


Summer 2014

A Framework for Web Object Self-Preservation

Charles L. Cartledge
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/computerscience_etds

 Part of the [Computer Sciences Commons](#), and the [Digital Communications and Networking Commons](#)

Recommended Citation

Cartledge, Charles L.. "A Framework for Web Object Self-Preservation" (2014). Doctor of Philosophy (PhD), dissertation, Computer Science, Old Dominion University, DOI: 10.25777/60a5-2c24
https://digitalcommons.odu.edu/computerscience_etds/24

This Dissertation is brought to you for free and open access by the Computer Science at ODU Digital Commons. It has been accepted for inclusion in Computer Science Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

**A FRAMEWORK FOR WEB OBJECT
SELF-PRESERVATION**

by

Charles L. Cartledge
A.E.E.T. December 1972, University of Alaska
B.E.E.T. June 1974, Oregon Institute of Technology
M.S. December 2007, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2014

Approved by:

Michael L. Nelson (Director)

Hussein M. Abdel-Wahab (Member)

B. Danette Allen (Member)

Chester E. Grosch (Member)

Yaohang Li (Member)

Michele C. Weigle (Member)

ABSTRACT

A FRAMEWORK FOR WEB OBJECT SELF-PRESERVATION

Charles L. Cartledge
Old Dominion University, 2014
Director: Dr. Michael L. Nelson

We propose and develop a framework based on emergent behavior principles for the long-term preservation of digital data using the web infrastructure. We present the development of the framework called unsupervised small-world (USW) which is at the nexus of emergent behavior, graph theory, and digital preservation. The USW algorithm creates graph based structures on the Web used for preservation of web objects (WOs). Emergent behavior activities, based on Craig Reynolds' "boids" concept, are used to preserve WOs without the need for a central archiving authority. Graph theory is extended by developing an algorithm that incrementally creates small-world graphs. Graph theory provides a foundation to discuss the vulnerability of graphs to different types of failures and attack profiles. Investigation into the robustness and resilience of USW graphs lead to the development of a metric to quantify the effect of damage inflicted on a graph. The metric remains valid whether the graph is connected or not. Different USW preservation policies are explored within a simulation environment where preservation copies have to be spread across hosts. Spreading the copies across hosts helps to ensure that copies will remain available even when there is a concerted effort to remove all copies of a USW component. A moderately aggressive preservation policy is the most effective at making the best use of host and network resources.

Our efforts are directed at answering the following research questions:

1. *Can web objects (WOs) be constructed to outlive the people and institutions that created them?*

We have developed, analyzed, tested through simulations, and developed a reference implementation of the unsupervised small-world (USW) algorithm that we believe will create a connected network of WOs based on the web infrastructure (WI) that will outlive the people and institutions that created the

WOs. The USW graph will outlive its creators by being *robust* and continuing to operate when some of its WOs are lost, and it is *resilient* and will recover when some of its WOs are lost.

2. *Can we leverage aspects of naturally occurring networks and group behavior for preservation?*

We used Reynolds' tenets for "boids" to guide our analysis and development of the USW algorithm. The USW algorithm allows a WO to "explore" a portion of the USW graph before making connections to members of the graph and before making preservation copies across the "discovered" graph. Analysis and simulation show that the USW graph has an average path length ($L(G)$) and clustering coefficient ($C(G)$) values comparable to small-world graphs. A high $C(G)$ is important because it reflects how likely it is that a WO will be able spread copies to other domains, thereby increasing its likelihood of long term survival. A short $L(G)$ is important because it means that a WO will not have to look too far to identify new candidate preservation domains, if needed. Small-world graphs occur in nature and are thus believed to be *robust* and *resilient*. The USW algorithms use these small-world graph characteristics to spread preservation copies across as many hosts as needed and possible.

USW graph creation, damage, repair and preservation has been developed and tested in a simulation and reference implementation.

Copyright, 2014, by Charles L. Cartledge, All Rights Reserved.

ACKNOWLEDGMENTS

This is to all the ones who supported me and pushed me through. I am the sum of parts of all these:

- Dr. Nelson without whose patience, guidance, direction, and faith none of this would have been possible.
- Dr. Abdel-Wahab whose positive attitude, willingness to listen, and answer questions made all of our conversations a delight.
- Dr. Allen whose system oriented focused comments and questions were always constructive and appreciated.
- Dr. Grosch who was always ready and available to teach me how to express myself and my ideas.
- Dr. Li and our many discussions about making predictions when data sets were too large to manipulate in a reasonable time.
- Dr. Weigle whose insights, attention to detail, and willingness to explore different data presentation techniques and ideas are greatly appreciated.
- Dr. Martin Klein for his practical and sound advice about the ins and outs of the Ph.D. process.
- The Web Science Digital Library group, a coven of students under Dr. Nelson's tutelage. The stories we shared have changed me and broadened my appreciation of the sacrifices many have made on this trail towards "PhDhood."
- Mary, my wife and best friend, without whose patience, understanding, and support I could not have made it this far. She has sacrificed time with friends and family, deferred trips to far flung places, spent endless hours reading about "wandering nodes," and endless nights alone while I ran just one more simulation. Now that this part of my life has come to an end, it is time for me to start repaying my debt to her (with interest).

My hope is that I can pay forward the help, guidance, consistency, and tolerance that has been given to me.

This work supported in part by the NSF, Project 370161.

TABLE OF CONTENTS

	Page
LIST OF TABLES.....	xiii
LIST OF FIGURES	xxv
CHAPTER	
1. INTRODUCTION	1
1.1. COMPARISON OF PRESERVATION ATTRIBUTES	2
1.2. TECHNOLOGY ENABLERS	5
1.3. LIFESPAN LIMITATIONS	10
1.4. SAFEGUARDING OF CONTENT	12
1.5. RESEARCH QUESTIONS.....	14
2. BACKGROUND	15
2.1. BOIDS	15
2.2. SMALL-WORLD.....	18
2.3. CURRENT PRACTICE IN DIGITAL PRESERVATION	18
2.4. SUMMARY	26
3. RELATED WORKS	29
3.1. EMERGENT BEHAVIOR	29
3.2. GRAPH THEORY	32
3.3. DIGITAL LIBRARIES AND WEB PRESERVATION	60
3.4. SUMMARY	79
4. INTRODUCTION TO UNSUPERVISED SMALL-WORLD (USW).....	80
4.1. INTRODUCTION TO JOSIE MCCLURE	80
4.2. TENETS OF THE USW ALGORITHM	80
4.3. EXPECTED CONTRIBUTIONS	81
4.4. THE USW CREATE, ATTACK, AND REPAIR LIFE CYCLE	82
5. THEORY	83
5.1. INTRODUCTION	83
5.2. GRAPH THEORY	83
5.3. COMMUNICATIONS.....	121
5.4. COPYING	143
5.5. ATTACK VERSUS FAILURE	148
5.6. EFFECT OF CLUSTERING COEFFICIENT EQUATION SELEC- TION	171
5.7. SUMMARY	187

6.	SIMULATION	188
6.1.	INTRODUCTION	188
6.2.	POLICIES	188
6.3.	CONTROL VALUES	188
6.4.	COMPARISON OF USW AND RANDOM GRAPHS	190
6.5.	GROWTH FUNCTION THEORETIC	191
6.6.	ORTHOGONALITY OF FAMILIES, FRIENDS, AND HOSTS	194
6.7.	DESIGN DECISIONS	195
6.8.	REDUCING THE PROBLEM SPACE	212
6.9.	CREATING AN UNSUPERVISED SMALL-WORLD GRAPH	237
6.10.	ANALYSIS OF DAMAGE VS. REPAIR	238
6.11.	ATTACKING AN UNSUPERVISED SMALL-WORLD GRAPH	239
6.12.	REPAIRING AN UNSUPERVISED SMALL-WORLD GRAPH	241
6.13.	REPEATEDLY ATTACKING A USW GRAPH	244
7.	REFERENCE IMPLEMENTATION	253
7.1.	INTRODUCTION	253
7.2.	DIFFERENCES BETWEEN USW THEORY AND REFERENCE IMPLEMENTATION	253
7.3.	IMPLEMENTATION ENVIRONMENT	253
7.4.	USW IMPLEMENTATION	266
7.5.	RESULTS	273
8.	FUTURE WORK	275
8.1.	IMPROVED CONNECTION SELECTION	275
8.2.	HANDLING MULTIPLE WO MAILBOXES	277
8.3.	IMPROVED AGGREGATE RESOURCE UPDATES	278
8.4.	IMPROVED REPLICATION PROCESSES	279
8.5.	IMPROVED SECURITY	280
9.	CONCLUSION AND CONTRIBUTIONS	281
9.1.	CONCLUSION	281
9.2.	CONTRIBUTIONS	285
APPENDICES		
A.	ALGORITHMS	287
A.1.	MAIN ALGORITHMS AND FUNCTIONS	287
A.2.	SUPPORTING ALGORITHMS AND FUNCTIONS	302
A.3.	FUTURE WORK	309
B.	USW EVENTS	311
B.1.	EVENT 101. GET ENTRYPOINT'S REM.	312
B.2.	EVENT 102. GET FRIEND OF FRIEND'S REM.	313
B.3.	EVENT 103. SEND FRIEND REQUEST.	314
B.4.	EVENT 104. WO UPDATES OWN REM.	316

B.5. EVENT 105. RETRIEVE FRIEND REQUEST.	318
B.6. EVENT 106. PROCESS FRIEND REQUEST.	321
B.7. USW GRAPH AFTER WANDERING.	323
B.8. EVENT 201. COPY REQUEST TO FRIEND.	324
B.9. EVENT 202. GET COPY REQUEST MESSAGE.	332
B.10. EVENT 203. COPY SERVICE REQUEST.	341
B.11. EVENT 204. COPY LOCATION RETURNED	349
B.12. EVENT 205. COPY LOCATION SENT.	350
B.13. EVENT 206. RETRIEVE COPY LOCATION.	352
B.14. EVENT 207. UPDATE REM WITH COPY'S LOCATION.	354
B.15. EVENT 208. BROADCAST COPY LOCATION TO FAMILY.	356
B.16. EVENT 209. RETRIEVE COPY LOCATION MESSAGE.	358
B.17. EVENT 210. UPDATE REM WITH LOCATION OF NEW COPY.	360
B.18. EVENT 301. UPDATE MAILBOX TIME CHECKED.	362
B.19. EVENT 302. ENSURE PARENT IS ACCESSIBLE.	364
B.20. EVENT 303. ENSURE THAT FRIENDS ARE ACCESSIBLE.	365
B.21. EVENT 304. ENSURE THAT FAMILY MEMBERS ARE ACCESSIBLE.	366
B.22. EVENT 305. ENSURE THAT THE PROGENITOR IS ACCESSIBLE.	367
C. COMMAND LINE ARGUMENTS	368
C.1. USW SIMULATOR.	368
C.2. PRESERVE ME! ROBOT	377
D. VOCABULARY.	383
D.1. DIGITAL PRESERVATION TERMS	383
D.2. GRAPH THEORETIC TERMS	385
D.3. USW TERMS.	405
D.4. MISCELLANEOUS TERMS	408
E. USW MATHEMATICAL TERMS.	410
F. DEGREE HISTOGRAMS	418
F.1. INTRODUCTION	418
F.2. HISTOGRAMS BASED RESULTING FROM USING THE SAME FIRST WO	419
F.3. HISTOGRAMS BASED RESULTING FROM USING A RANDOM FIRST WO	430
F.4. HISTOGRAMS BASED RESULTING FROM USING LAST ADDED WO	441
G. MESSAGE PROPAGATION FIGURES	452
H. USW DEGREE DISTRIBUTIONS	489
I. USW DISCONNECTION	499

J. USW PATH LENGTH HISTOGRAMS	509
K. USW SYSTEM DESIGN CONSIDERATIONS	519
K.1. INTRODUCTION	519
K.2. DISCUSSION	519
K.3. PRESERVATION EFFECTIVENESS ANALYSIS	521
K.4. COMMUNICATION COST ANALYSIS	538
K.5. SUMMARY	538
L. LIST OF ALGORITHMS	546
M. LIST OF LISTINGS	547
REFERENCES	548
VITA	567

LIST OF TABLES

Table	Page
1. Increases in Intel transistor density as a function of time.	7
2. Increase in standards based LAN speeds has been roughly 45% per year.	8
3. Listing, interpreting and implementing Craig W. Reynolds' flocking concepts in the USW framework.	17
4. Measured and computed metrics for real-world small-world graphs.	18
5. Comparing Cooper and Garcia-Molina concept of trading networks and the USW.	58
6. Maximum number of edges based on directivity and self loops.	99
7. A summary list of useful equations.	100
8. Comparing the expected average path length and expected clustering coefficient for lattice, small-world and random graphs.	101
9. Comparing Albert, Jeong and Barabási's raw s to our damage metric for a collection of test graphs.	107
10. A list of symbols used for the analysis of USW communications.	122
11. The number of WOs at selected distances.	124
12. Computing the number of selections needed to achieve 0.95 likelihood of a message being received by all WOs based on different WO selection criteria.	132
13. Subjective evaluation of communications mechanisms and WO selection criteria.	132
14. A comparison and analysis of the number of selections needed to reach a level of confidence versus the order of the graph based on uniform random selection.	135
15. Summary of the number of epochs needed to get the sample message to all WOs.	139
16. Number of trials to send a message.	141

17.	Aggregated data categories based on an expected update rate.	145
18.	Recommended copying actions based on resource size and update frequency.	147
19.	Comparing the betweenness of edges based on the neighborhood discovered from a central vertex.	155
20.	Comparing the betweenness of vertices based on the neighborhood discovered from a central vertex.	157
21.	Comparing the degree of each vertex based on the neighborhood discovered from a central vertex.	158
22.	Damage to the discovered subgraph of path length 3 based on $A_{E,*}$ attack profiles.	163
23.	Damage to the discovered subgraph of path length 3 based on $A_{V,*}$ attack profiles.	168
24.	Damage to the discovered subgraph of path length 3 based on $A_{D,*}$ attack profiles.	174
25.	Efficacy of various attack profiles.	174
26.	Various clustering coefficients definitions and their sources.	177
27.	Number of triangles and triples per node from the test graph.	181
28.	Results of exercising the igraph and sna packages with the sample graph.	182
29.	USW simulation parameters used to generate graphs to be analyzed with igraph.	184
30.	Comparing the regions for the expected average path length ($L(G)$) and expected Clustering Coefficient ($C(G)$) for lattice, small-world and random graphs.	187
31.	Comparing USW and random graph metrics.	191
32.	Summary of various graph characteristics ($n = 75, \beta = 0.85, \gamma = 0.10$) as a function of selection method and constant <i>established</i> WO.	201
33.	Summary of various graph characteristics ($n = 75, \beta = 0.85, \gamma = 0.10$) as a function of random selection method to select the <i>established</i> WO.	202
34.	Summary of various graph characteristics ($n = 75, \beta = 0.85, \gamma = 0.10$) as a function of using the last previously entered WO as the <i>established</i> WO.	203

35.	How often a WO is chosen as USW graph grows.	204
36.	Trending 50% and 70% based on USW graph size.	204
37.	Combinatorial explosion of possible USW control parameters.	212
38.	USW simulation values used to reduce USW problem space.	231
39.	CC and APL logical (raw) data.	235
40.	CC and APL logical (normalized) data.	236
41.	Alternating turns between attacker and graph.	238
42.	USW simulation values used to create a USW graph.	239
43.	Comparison of pre and post attack USW graph metrics.	241
44.	Comparison of pre and post repair USW graph metrics.	244
45.	Differences between USW simulation and reference implementation.	254
46.	Domains that provided HTML pages for USW implementation testing.	256
47.	Comparing USW implementation and random graph metrics.	273
48.	Comparing USW implementation copies and desired copies.	273
49.	Robot execution times for various implementation order USW graphs.	274
50.	Analysis of Josie's metadata.	284
51.	USW simulator command line arguments.	368
52.	Preserve Me! robot command line arguments.	377
53.	Number of triangles and triples per node from the test graph.	401
54.	Results using different approaches for computing the clustering coefficients for the sample graph.	402
55.	USW related mathematical symbols.	410
56.	The effect on $C(G)$ and $L(G)$ based on various values for β and γ	510
57.	Named conditions for total system host preservation capacity h_{cap} in relation to total system c_{soft} and c_{hard}	522

58. The effectiveness of various preservation policies based on named host capacity conditions.	544
---	-----

LIST OF FIGURES

Figure	Page
1. A 1907 photograph of Josie McClure.	3
2. Hardware continues to improve (Moore’s “Law”).	8
3. Hardware continues to improve (network speed).	9
4. Hardware continues to improve (magnetic mass storage).	9
5. Average US life expectancy for both sexes, 1940 - 2010.	12
6. Naturally occurring small-world graph: nematode <i>Caenorhabditis ele-</i> <i>gans</i> neuron connections.	19
7. Man-made small-world graph: Western Electricity Coordinating Council.	19
8. Organic small-world graph: ENRON e-mail exchanges.	20
9. Relationship between URI, resource, and representation.	22
10. OAIS Reference Model functional model.	26
11. Navigational figure orienting USW graph life aspects.	27
12. USW is at the nexus of multiple disciplines.	30
13. Watts and Strogatz lattice evolves to a random graph.	43
14. Representative lattice graph and associated degree distribution.	87
15. Representative Erdős-Rényi random graph and associated degree distri- bution.	88
16. Representative Albert-Barabási scale free graph and associated degree distribution.	89
17. Representative Watts – Strogatz small-world graph and associated degree distribution.	90
18. Barabási graph — disconnected after 1 removal.	91
19. Erdős-Rényi graph — disconnected after 116 removals.	92
20. Watts – Strogatz graph — disconnected after 75 removals.	93

21.	The degree distribution plot for $\beta = 0.0$ and $\gamma = 1.0$	94
22.	The disconnection plot for $\beta = 0.0$ and $\gamma = 1.0$	95
23.	Notional diagrams for test cases 100 , $90,10$, $90\dots 1$ and $80\dots 2$	108
24.	Notional diagrams for test cases $50,50$, $50,49,1$, $50,40,10$ and $50,30,10,10$.	109
25.	Notional diagrams for test cases $50\dots 5$, $20\dots 20$, $16\dots 1$ and $10\dots 10$.	110
26.	Notional diagrams for test cases $10\dots 9$ and $1\dots 1$	111
27.	Sample two triangles graph.	112
28.	Sample totally disconnected graph.	113
29.	Sample 3 pairs graph.	114
30.	Sample 1 LCC and 6 singletons graph.	115
31.	Sample fully connected graph.	116
32.	Sample cave man graph.	117
33.	Sample butterfly graph.	118
34.	Sample off balance graph.	119
35.	Sample clusters graph.	120
36.	A “toy” graph that is tractable by hand.	123
37.	The number of selections required to reach any particular a	130
38.	The cumulative probability distribution (CPD) for the selection process.	131
39.	A comparison of different degree distributions and an assessment of how suitable the graph is for message distribution.	138
40.	Preservation copy guidelines.	144
41.	The sample graph presented to each attack profile.	149
42.	The first graph component that will be removed based on different attack profiles (1 of 2).	150
43.	The first graph component that will be removed based on different attack profiles (2 of 2).	151

44.	The effects of different path lengths starting from a fixed vertex in discovering the global graph.	154
45.	The effects of the $A_{E,L}$ attack profile on the sample graph.	161
46.	The effects of the $A_{E,H}$ attack profile on the sample graph.	162
47.	Damage to the discovered graph of path length 3 based on $A_{E,*}$ attack profiles.	164
48.	The effects of an $A_{V,L}$ attack profile on the sample graph.	166
49.	The effects of an $A_{V,H}$ attack profile on the sample graph.	167
50.	Damage to the discovered subgraph of path length 3 based on $A_{V,*}$ attack profiles.	169
51.	Markedly different graphs resulting from the differences in choosing $A_{V,H}$ or $A_{V,L}$ attack profiles.	170
52.	The effects of an $A_{D,L}$ attack profile on the sample graph.	172
53.	The effects of on $A_{D,H}$ attack profile on the sample graph.	173
54.	Damage to the discovered subgraph of path length 3 by based on $A_{D,*}$ attack profiles.	175
55.	The sample graph after removing the fifth discovered node using $A_{D,*}$ attack profiles.	176
56.	Watts – Strogatz small-world graph analysis (normalized data).	180
57.	Watts – Strogatz small-world graph analysis (raw data).	180
58.	Sample undirected graph used to demonstrate effects of different Clustering Coefficient definitions.	181
59.	Average path lengths and clustering coefficients for various sized USW graphs.	183
60.	The effect of varying β and γ on a USW graph of size 10.	185
61.	Path length histogram.	186
62.	Creation, wandering, and active maintenance phases and policy navigation aide.	189
63.	Comparing USW and random graph metrics (raw data).	192

64.	Comparing USW and random graph metrics (normalized data).....	192
65.	USW WO “friendship” links.....	195
66.	USW WO families.....	196
67.	USW hosts.....	197
68.	Creation, wandering, and active maintenance phases and policy A.....	197
69.	Creation, wandering, and active maintenance phases and policy B.....	199
70.	Harmonic series of 4 points showing which WOs are selected 50% and 75% of the time.....	205
71.	Harmonic series of 100 points showing which WOs are selected 50% and 75% of the time.....	206
72.	Harmonic series of 1,000 points showing which WOs are selected 50% and 75% of the time.....	207
73.	Harmonic series of 10,000 points showing which WOs are selected 50% and 75% of the time.....	208
74.	Creation, wandering, and active maintenance phases and policies C and E.	210
75.	A snapshot of the least aggressive replication policy.....	214
76.	The growth of a $n_{\max} = 500$ WO system captured at various time-steps.	216
77.	Least aggressive replication policy. System stabilization at $T_{\text{step}} = 334$..	218
78.	Moderately aggressive replication policy. System stabilization $T_{\text{step}} = 554$.	219
79.	Most aggressive replication policy. System stabilization at $T_{\text{step}} = 300$..	220
80.	Showing total messages sent and received by an early node, a mid-simulation node and all WOs.....	221
81.	Theoretical truth table showing all possible copying conditions.....	226
82.	Number of preservation copies based on a specific maximum copying condition.....	227
83.	Number of preservation copies per host based on a specific maximum copying condition.....	228

84.	Histogram of preservation copies based on a specific maximum copying condition.	229
85.	Histogram of preservation copies per host baed on a specific maximum copying condition.	230
86.	A specific minimum copying condition WO status.	232
87.	A specific minimum copying condition, a histogram.	233
88.	A specific minimum copying condition, sorted by WO status.	234
89.	Relationship of damage and repair on graph health.	247
90.	CC of USW simulation.	248
91.	Degrees of USW simulation.	249
92.	APL of USW simulation.	250
93.	Temporal degree histogram of USW simulation.	251
94.	Graph metrics after repeated low percentage (5%) attacks and repairs. . .	252
95.	Graph metrics after repeated high percentage (50%) attacks and repairs.	252
96.	Sample HTML page from arXiv domain.	256
97.	Sample HTML page from flickr domain.	257
98.	Sample HTML page from gutenber domain.	258
99.	Sample HTML page from radiolab domain.	259
100.	The Preserve Me! Viz display.	265
101.	USW client basic “make connection” popup.	267
102.	USW client basic make connection popup.	268
103.	USW client basic make copies popup.	269
104.	USW client basic copies request popup.	270
105.	USW client basic with copies popup.	271
106.	Histogram of an implemented USW graph.	274

107. OAIS Reference Model functional entities.	283
108. OAIS Information Model.	283
109. USW contributions to graph theory, emergent behavior, and preservation.	286
110. Wandering events and messages, event 101.	312
111. Wandering events and messages, event 102.	313
112. Wandering events and messages, event 103.	314
113. Wandering events and messages, event 104.	316
114. Wandering events and messages, event 105.	318
115. Wandering events and messages, event 106.	321
116. USW graph after wandering events.	323
117. Copying events and messages, event 201.	324
118. Copying events and messages, event 202.	332
119. Copying events and messages, event 203.	341
120. Copying events and messages, event 204.	349
121. Copying events and messages, event 205.	350
122. Copying events and messages, event 206.	352
123. Copying events and messages, event 207.	354
124. Copying events and messages, event 208.	356
125. Copying events and messages, event 209.	358
126. Copying events and messages, event 210.	360
127. Maintenance events and messages, event 301.	362
128. Maintenance events and messages, event 302.	364
129. Maintenance events and messages, event 303.	365
130. Maintenance events and messages, event 304.	366
131. Maintenance events and messages, event 305.	367

132.	Histogram of representative degree distributions of 1000 node graphs built using random, power law, small-world and USW processes.	392
133.	Representative lattice graph and associated degree distribution.	395
134.	Representative Erdős-Rényi random graph and associated degree distribution.	396
135.	Representative Albert-Barabási scale free graph and associated degree distribution.	397
136.	Representative Watts – Strogatz small-world graph and associated degree distribution.	398
137.	A “small-world” graph exists along the continuum between a regular lattice and a random graph.	399
138.	Sample undirected graph used to demonstrate effects of different Clustering Coefficient definitions.	401
139.	Degree histogram for $n * \gamma$, always using same first WO.	420
140.	Degree histogram for $max(1, \ln(n * \gamma))$, always using same first WO. . . .	421
141.	Degree histogram for $max(1, \ln(n) * \gamma)$, always using same first WO. . . .	422
142.	Degree histogram for $max(0, \ln(n * \gamma))$, always using same first WO. . . .	423
143.	Degree histogram for $max(0, \ln(n) * \gamma)$, always using same first WO. . . .	424
144.	Degree histogram for $max(1, \log_2(n * \gamma))$, always using same first WO. . . .	425
145.	Degree histogram for $max(1, \log_2(n) * \gamma)$, always using same first WO. . . .	426
146.	Degree histogram for $max(0, \log_2(n * \gamma))$, always using same first WO. . . .	427
147.	Degree histogram for $max(0, \log_2(n) * \gamma)$, always using same first WO. . . .	428
148.	Degree histogram for $5 + \log_2(n * \gamma)$, always using same first WO.	429
149.	Degree histogram for $n * \gamma$, always using a random first WO.	431
150.	Degree histogram for $max(1, \ln(n * \gamma))$, always using a random first WO. . . .	432
151.	Degree histogram for $max(1, \ln(n) * \gamma)$, always using a random first WO. . . .	433
152.	Degree histogram for $max(0, \ln(n * \gamma))$, always using a random first WO. . . .	434

153. Degree histogram for $\max(0, \ln(n) * \gamma)$, always using a random first WO.	435
154. Degree histogram for $\max(1, \log_2(n * \gamma))$, always using a random first WO.	436
155. Degree histogram for $\max(1, \log_2(n) * \gamma)$, always using a random first WO.	437
156. Degree histogram for $\max(0, \log_2(n * \gamma))$, always using a random first WO.	438
157. Degree histogram for $\max(0, \log_2(n) * \gamma)$, always using a random first WO.	439
158. Degree histogram for $5 + \log_2(n * \gamma)$, always using a random first WO. . . .	440
159. Degree histogram for $n * \gamma$, always using using last added WO.	442
160. Degree histogram for $\max(1, \ln(n * \gamma))$, always using using last added WO.	443
161. Degree histogram for $\max(1, \ln(n) * \gamma)$, always using using last added WO.	444
162. Degree histogram for $\max(0, \ln(n * \gamma))$, always using using last added WO.	445
163. Degree histogram for $\max(0, \ln(n) * \gamma)$, always using using last added WO.	446
164. Degree histogram for $\max(1, \log_2(n * \gamma))$, always using using last added WO.	447
165. Degree histogram for $\max(1, \log_2(n) * \gamma)$, always using using last added WO.	448
166. Degree histogram for $\max(0, \log_2(n * \gamma))$, always using using last added WO.	449
167. Degree histogram for $\max(0, \log_2(n) * \gamma)$, always using using last added WO.	450
168. Degree histogram for $5 + \log_2(n * \gamma)$, always using using last added WO.	451
169. Degree distribution, size = 1000 $\beta = 0.0 \gamma = 0.0$	453
170. Sequential, size = 1000 $\beta = 0.0 \gamma = 0.0$	454
171. Random selection, size = 1000 $\beta = 0.0 \gamma = 0.0$	455
172. Degree biased selection, size = 1000 $\beta = 0.0 \gamma = 0.0$	456
173. Degree distribution, size = 1000 $\beta = 0.0 \gamma = 0.5$	457
174. Sequential, size = 1000 $\beta = 0.0 \gamma = 0.5$	458

175. Random selection, size = 1000 $\beta = 0.0$ $\gamma = 0.5$	459
176. Degree biased selection, size = 1000 $\beta = 0.0$ $\gamma = 0.5$	460
177. Degree distribution, size = 1000 $\beta = 0.0$ $\gamma = 1.0$	461
178. Sequential, size = 1000 $\beta = 0.0$ $\gamma = 1.0$	462
179. Random selection, size = 1000 $\beta = 0.0$ $\gamma = 1.0$	463
180. Degree biased selection, size = 1000 $\beta = 0.0$ $\gamma = 1.0$	464
181. Degree distribution, size = 1000 $\beta = 0.5$ $\gamma = 0.0$	465
182. Sequential, size = 1000 $\beta = 0.5$ $\gamma = 0.0$	466
183. Random selection, size = 1000 $\beta = 0.5$ $\gamma = 0.0$	467
184. Degree biased selection, size = 1000 $\beta = 0.5$ $\gamma = 0.0$	468
185. Degree distribution, size = 1000 $\beta = 0.5$ $\gamma = 0.5$	469
186. Sequential, size = 1000 $\beta = 0.5$ $\gamma = 0.5$	470
187. Random selection, size = 1000 $\beta = 0.5$ $\gamma = 0.5$	471
188. Degree biased selection, size = 1000 $\beta = 0.5$ $\gamma = 0.5$	472
189. Degree distribution, size = 1000 $\beta = 0.5$ $\gamma = 1.0$	473
190. Sequential, size = 1000 $\beta = 0.5$ $\gamma = 1.0$	474
191. Random selection, size = 1000 $\beta = 0.5$ $\gamma = 1.0$	475
192. Degree biased selection, size = 1000 $\beta = 0.5$ $\gamma = 1.0$	476
193. Degree distribution, size = 1000 $\beta = 1.0$ $\gamma = 0.0$	477
194. Sequential, size = 1000 $\beta = 1.0$ $\gamma = 0.0$	478
195. Random selection, size = 1000 $\beta = 1.0$ $\gamma = 0.0$	479
196. Degree biased selection, size = 1000 $\beta = 1.0$ $\gamma = 0.0$	480
197. Degree distribution, size = 1000 $\beta = 1.0$ $\gamma = 0.5$	481
198. Sequential, size = 1000 $\beta = 1.0$ $\gamma = 0.5$	482
199. Random selection, size = 1000 $\beta = 1.0$ $\gamma = 0.5$	483

200. Degree biased selection, size = 1000 $\beta = 1.0$ $\gamma = 0.5$	484
201. Degree distribution, size = 1000 $\beta = 1.0$ $\gamma = 1.0$	485
202. Sequential, size = 1000 $\beta = 1.0$ $\gamma = 1.0$	486
203. Random selection, size = 1000 $\beta = 1.0$ $\gamma = 1.0$	487
204. Degree biased selection, size = 1000 $\beta = 1.0$ $\gamma = 1.0$	488
205. The degree distribution for $\beta = 0.0$ and $\gamma = 0.0$	490
206. The degree distribution for $\beta = 0.5$ and $\gamma = 0.5$	491
207. The degree distribution for $\beta = 1.0$ and $\gamma = 1.0$	492
208. The degree distribution for $\beta = 0.0$ and $\gamma = 0.0$	493
209. The degree distribution for $\beta = 0.5$ and $\gamma = 0.5$	494
210. The degree distribution for $\beta = 1.0$ and $\gamma = 1.0$	495
211. The degree distribution for $\beta = 0.0$ and $\gamma = 0.0$	496
212. The degree distribution for $\beta = 0.5$ and $\gamma = 0.5$	497
213. The degree distribution for $\beta = 1.0$ and $\gamma = 1.0$	498
214. The disconnection plot for $\beta = 0.0$ and $\gamma = 0.0$	500
215. The disconnection plot for $\beta = 0.5$ and $\gamma = 0.5$	501
216. The disconnection plot for $\beta = 1.0$ and $\gamma = 1.0$	502
217. The disconnection plot for $\beta = 0.0$ and $\gamma = 0.0$	503
218. The disconnection plot for $\beta = 0.5$ and $\gamma = 0.5$	504
219. The disconnection plot for $\beta = 1.0$ and $\gamma = 1.0$	505
220. The disconnection plot for $\beta = 0.0$ and $\gamma = 0.0$	506
221. The disconnection plot for $\beta = 0.5$ and $\gamma = 0.5$	507
222. The disconnection plot for $\beta = 1.0$ and $\gamma = 1.0$	508
223. The path length histogram for $\beta = 0.0$ and $\gamma = 0.0$	510
224. The path length histogram for $\beta = 0.5$ and $\gamma = 0.5$	511

225.	The path length histogram for $\beta = 1.0$ and $\gamma = 1.0$	512
226.	The path length histogram for $\beta = 0.0$ and $\gamma = 0.0$	513
227.	The path length histogram for $\beta = 0.5$ and $\gamma = 0.5$	514
228.	The path length histogram for $\beta = 1.0$ and $\gamma = 1.0$	515
229.	The path length histogram for $\beta = 0.0$ and $\gamma = 0.0$	516
230.	The path length histogram for $\beta = 0.5$ and $\gamma = 0.5$	517
231.	The path length histogram for $\beta = 1.0$ and $\gamma = 1.0$	518
232.	Notional host system design diagram relating host capacity to WO preservation needs.	521
233.	Effectiveness of least aggressive preservation policy in famine system capacity condition.	523
234.	Effectiveness of moderately aggressive preservation policy in famine system capacity condition.	524
235.	Effectiveness of most aggressive preservation policy in famine system capacity condition.	525
236.	Effectiveness of least aggressive preservation policy in boundary-low system capacity condition.	526
237.	Effectiveness of moderately aggressive preservation policy in boundary-low system capacity condition.	527
238.	Effectiveness of most aggressive preservation policy in boundary-low system capacity condition.	528
239.	Effectiveness of least aggressive preservation policy in straddle system capacity condition.	529
240.	Effectiveness of moderately aggressive preservation policy in straddle system capacity condition.	530
241.	Effectiveness of most aggressive preservation policy in straddle system capacity condition.	531
242.	Effectiveness of least aggressive preservation policy in boundary-high system capacity condition.	532

243. Effectiveness of moderately aggressive preservation policy in boundary-high system capacity condition.	533
244. Effectiveness of most aggressive preservation policy in boundary-high system capacity condition.	534
245. Effectiveness of least aggressive preservation policy in feast system capacity condition.	535
246. Effectiveness of moderately aggressive preservation policy in feast system capacity condition.	536
247. Effectiveness of most aggressive preservation policy in feast system capacity condition.	537
248. Number of messages exchanged based on preservation policies in famine system capacity condition.	539
249. Number of messages exchanged based on preservation policies in boundary-low system capacity condition.	540
250. Number of messages exchanged based on preservation policies in straddle system capacity condition.	541
251. Number of messages exchanged based on preservation policies in boundary-high system capacity condition.	542
252. Number of messages exchanged based on preservation policies in feast system capacity condition.	543

CHAPTER 1

INTRODUCTION

My name is JosieP and I've slept in a drawer for twenty years. Artie kept me there. It was warm in the winter, cool in the summer and it was always dark and protected. Sometimes we would look out over the ocean and watch the waves. Later, we would look at the mountains and the desert. Artie liked me and cared for me. She took me with her everywhere. We traveled all over the world together. And then there were three of us.

Artie married Bert. He was nice. He took care of things. When Artie started to get sick, Bert took care of her and me. When Artie got sicker, we did not go out as much any more. The time between looking at the mountains got longer and longer. I heard Artie and Bert talking, but it did not make much sense. I stayed in the drawer and kept company with the others. When I did see Artie, I do not think that she knew who I was.

Then she stopped visiting me. I heard Bert talking with other people and everything seemed to be sad and quiet. Bert started moving things. I would see him sometimes, but he never talked to me. Not the way that Artie did.

Bert took some of my friends away and I got lonely. I think that he still liked me, but it was a very quiet time.

I heard Bert talking to David. David is Artie's nephew. Bert was saying that Artie did not live there anymore. Artie needed more help and care than Bert could give, so she was living somewhere else. Bert said that he was going to live there too, soon. Bert told David that there was so much stuff to get rid of he did not know what to do. He had already given a lot of things to friends and family members, but there was still a lot left. David said that he would take it all; everything. All Bert had to do was put it in a box and David would care for it, protect it, and pass it on to the next generation. I was not sure what all that was about, but it sounded good.

One day Bert came and got me. He got a lot of my friends as well. He picked me up and carefully laid me in a box. Then it went dark. Then cold and noisy. Then bumpy and cold. Then quiet, warm and moist.

I heard the box open. The sound of tape being pulled back and some cardboard tearing. David's voice started talking, "this is a picture of my Father's brother Robert. He ..." Things were being lifted off me and I could breath again. "These are pictures of Artie in South Africa, ..." And then there was light. "This is a picture of my Grandmother's sister Josie." And I could see.

It was a nice room. Yellow and brown walls and a few cat hairs in the air. David looked nice. His nose was not too large, the glasses fit his face, and his thinning silver hair was the right length. David passed me to his wife. She seems nice, too. She has nice eyes and says that I look nice in my muslin dress. She passes me on to their son. In many ways he was like me when I was that age. Mildly curious about the old things and the old stories. I bet he'd rather be doing almost anything other than listening to David talk about people who have passed.

David is talking about Bert and the promise made to him to protect things for the next generation. David is saying that he is not sure how he will do that but that he hopes to find a way to keep that promise so that a hundred years from now, their great grandchildren will be able to see the three of them.

Now I begin to understand. I am a picture of Josie McClure (Figure 1 on the following page), born 1892. I have been passed down through the years, safe and secure in darkened places, protected from the elements and not handled too much. And, I am here to see and be touched by my sister's great grandchild. David has made a promise to Bert, a man now since gone, to continue to protect me so that I can see David's great grandchildren.

The rest of this story belongs to David. David who grew up in an analog era; now with one foot there and one in the digital era is going to find a way to pass me on to generations unknown who have both feet firmly planted in the digital era.

1.1 COMPARISON OF PRESERVATION ATTRIBUTES

In Josie's analog age, popularity can lead to destruction and loss. Active use and enjoyment almost always leads to damage due to "normal wear and tear." In an analog world, benign neglect allows the original to be enjoyed far into the future in much the same way as when it was created. In a digital age, things are different.

David is faced with a different set of problems dealing with preserving things in the digital age. In the analog age, benign neglect eliminated "wear and tear" so things that were protected from the elements, insects, fire, and rot still remained.



(a) front

(b) back

Figure 1. A 1907 photograph of Josie McClure. An analog artifact in a digital age. Penciled on the back “Josie McClure picture taken Feb 30, 1907 at Poteau, I.T. Fifteen years of age When this was taken weighed 140 lbs.” The penciled information on the back is “metadata” associated with the picture. The same data converted to current place and current calendar would be: Josie McClure picture taken Mar 2, 1907 at Poteau, Oklahoma. Fifteen years of age when this was taken weighed 140 lbs.

As long as no one used them, things would stay around for a long time. In the digital age, things that are not used are lost. They may exist on a physical medium somewhere, but there may not be any means to read the medium. They may exist in a proprietary format, and the program to interpret the file may be lost. They may exist in a file that can be read, but “bit rot” [1, 2] may have set in and the file may become too corrupt to be processed. Data preservation is different in the two different ages.

“Conventional archiving distinguishes between conservation (which looks after individual artifacts) and preservation (which retains the content if the original artifact decays or is destroyed).”

William Y. Arms [3]

The tenets of digital preservation [3] will apply to Josie in the digital age:

- *Replication and refreshing*: aims to preserve a precise sequence of bits. A digital representation of Josie as a JPEG file would be copied without modification from one storage device to another.
- *Migration*: preserves the content at a semantic level, but not necessarily the specific sequence of bits. A digital representation of Josie as a JPEG file would be converted to a PNG formatted representation.
- *Emulation*: provides an environment where the original sequence of bits can be used. A digital representation of Josie as a JPEG file is viewed using software that runs a program that supports viewing JPEG files.

An important part of the digital preservation process is deciding what to collect, what to store, what to preserve for the future and what to discard. Compounding the problem is that it is nearly impossible to predict what will be important in the future. The simplest way to address these questions is to err on the side of ignorance and preserve everything.

Digital images and content that are popular are copied from one context to another and migrate from one format to another. This “handling” does not damage or destroy the original, but in fact helps to ensure its existence. Popularity also leads to emulation. In a digital world, benign neglect leads to permanent loss because

hardware, data formats and the software needed to read the data are constantly changing in a very short time.

David could convert Josie's analog image into a digital one, and then address digital preservation issues. Periodically he could *refresh* the bits by copying the digital data from one media (an internal hard disk, external drive, CD-ROM, DVD, or some solid state device) to another. If he were to monitor and predict changes in digital image format, he could *migrate* the bits from an old format to a newer one. Or he could maintain a computer system with complete operating system, a suite of applications and instructions for future generations on how to use the system. This complete operating environment could also be *emulated* in some sort of advanced simulation. David could do any and all of these things, until he was unable to because of health, financial constraints, or death. The digital representation of Josie will survive as long as David, as the curator survives.

1.2 TECHNOLOGY ENABLERS

In the digital age, technology enablers directly impact digital preservation. These enablers include:

- *Moore's Law*: states that transistor density doubles about every 24 months. This continuing increase in density is enabled by decreasing line width (thinner traces), more efficient component construction techniques and improved manufacturing processes. Moore in 1965 [4] predicted that by 1975 there would be as many as 65,000 components on an integrated circuit (IC). He looked at a number of existing IC manufacturing technologies and put forth the idea that the unit cost per technology would decrease as the number of components increased until the component density was such that the manufacturing defects would obviate any gains by increasing the density. When this happens, then another technology would be brought to the fore and the process would repeat itself yielding greater and greater densities. Based on these monetary considerations and his analysis of past efforts, he predicted a doubling in IC density about every other year.

Describing Moore's prediction using text is interesting, it is even more striking when plotted (Figure 2 on page 8 and Table 1 on page 7) [5, 6].

- *Nielson's Law*: states that Internet bandwidth (network speed) increases by a

factor 45% every year (Figure 3 on page 9 and Table 2 on page 8) [7],

- *Kryder's Law*: relates to the density of hard drives and indirectly the cost of magnetic storage, decreasing by approximately 45% every year (Figure 4 on page 9) [8, 9]. This is a fundamental aspect to the long-term preservation planning [10, 11, 12, 13].

“The density of hard drives increases by a factor of 1,000 every 10.5 years (doubling every 13 months).”

Mark Kryder [9]

The exponential decline in storage costs has changed the way that individuals and companies view storage. As the cost for storage approaches \$0.00 individuals will tend to save more and larger files and images. Companies will cease to charge for storing data at their locations and will look to other revenue avenues (i.e., advertising or added service value) as a way to make a profit, and

- *Broadband access*: will be available in more than 94% [14] of the homes in the US.

The cumulative effects of these enablers permit companies such as flickr, Gmail, shutterfly, or ImageShack to offer to store large amounts of digital data for free. These companies offer this storage because their income is derived from the sale of advertising targeted at the user. As the cost of storage medium approaches \$0.00, the limiting factor on how long the institution or repository may live and how long the institution is willing support the maintenance cost of the archive will be their revenue stream. Metcalfe's Law states:

“... connect any number, n , of machines - whether computers, phones or even cars - and you get n squared potential value. Think of phones without networks or cars without roads. Conversely, imagine the benefits of linking up tens of millions of computers and sense the exponential power of the telecoms.”

George Gilder [15]

Table 1. Increases in Intel transistor density as a function of time. The continued exponential increases in transistor density (≈ 2.843 for this period) is the basis for Moore's Law.

Year	Name	Number of transistors	Trace width (μm)	Die (mm^2)	Density trans/ (mm^2)
1971	4004	2,300	10.000	12	192
1972	8008	3,500	10.000	14	250
1974	8080	4,500	6.000	20	225
1978	8086	29,000	3.000	33	879
1979	8088	29,000	3.000	33	879
1982	Intel 286	134,000	1.500	49	2,735
1985	intel 386	275,000	1.500	104	2,644
1989	Intel 486	1,200,000	1.000	N/S	—
1993	Pentium	3,100,000	0.800	N/S	—
1995	Pentium Pro	5,500,000	0.350	N/S	—
1997	Pentium II	7,500,000	0.250	N/S	—
1998	Celeron	7,500,000	0.250	N/S	—
1999	Pentium III	9,500,000	0.250	123	77,236
2000	Pentium 4	$4 \cdot 10^{+07}$	0.180	112	375,000
2001	Itanium	$3 \cdot 10^{+07}$	0.180	N/S	—
2001	Xeon	$4 \cdot 10^{+07}$	0.180	90.3	465,116
2002	Itanium 2	$2 \cdot 10^{+08}$	0.180	374	588,235
2004	Itanium 2 (9MB cache)	$6 \cdot 10^{+08}$	0.130	N/S	—
2006	Itanium 2	$2 \cdot 10^{+09}$	0.090	N/S	—
2007	Itanium 2	$2 \cdot 10^{+09}$	0.090	N/S	—
2008	Xeon	$2 \cdot 10^{+09}$	0.045	N/S	—
2010	Core I7	$1 \cdot 10^{+09}$	0.032	263	4,448,669
2010	Quad-core Itanium	$2 \cdot 10^{+09}$	0.065	N/S	—
2010	8 Core	$2 \cdot 10^{+09}$	0.045	684	3,362,573
2010	10 Core	$3 \cdot 10^{+09}$	0.032	512	5,078,125
2012	Itanium 9500	$3 \cdot 10^{+09}$	0.032	544	5,698,529
2013	Core 17-4770K	$1 \cdot 10^{+09}$	0.022	177	7,909,605

Table 2. Increase in standards based LAN speeds has been roughly 45% per year.

Year	Mbps	IEEE Standard
1983	10	802.3a
1995	100	802.3u
1998	1,000	802.3y
2003	10,000	802.3ae
2011	40,000	802.3bg
2014(est.)	400,000	802.3bj

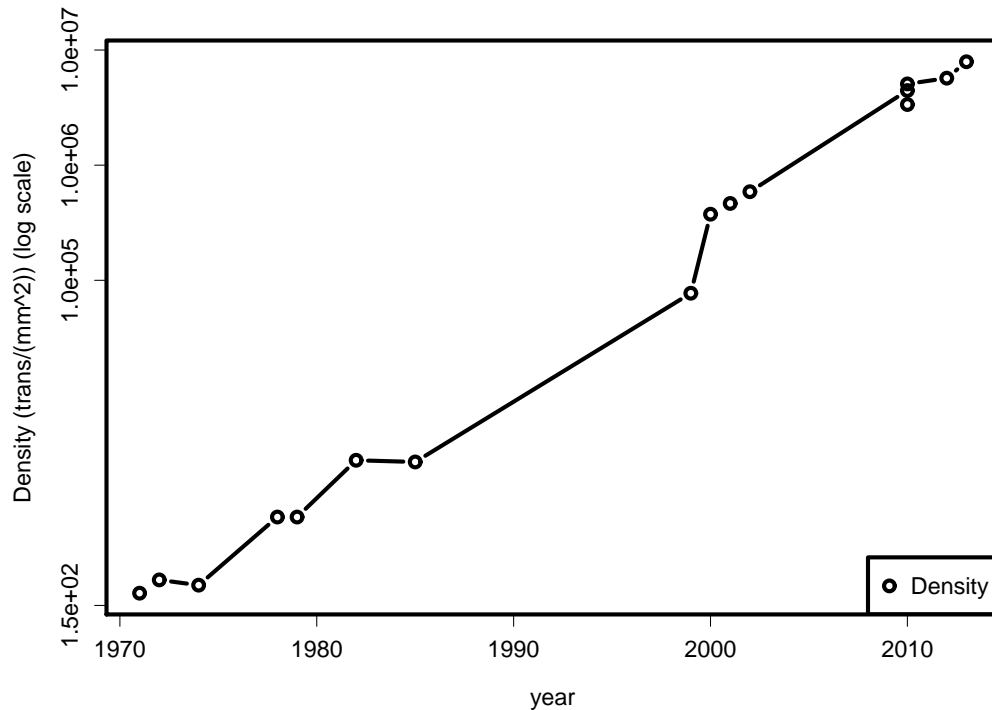


Figure 2. Hardware continues to improve (Moore’s “Law”). Moore’s “Law” of doubling transistor density every 18 to 24 months has held true for 45 years. Improvements in thermal cooling efficiencies and changes in the ways chips are manufactured from 2D to 3D are expected to enable the “Law” to continue.

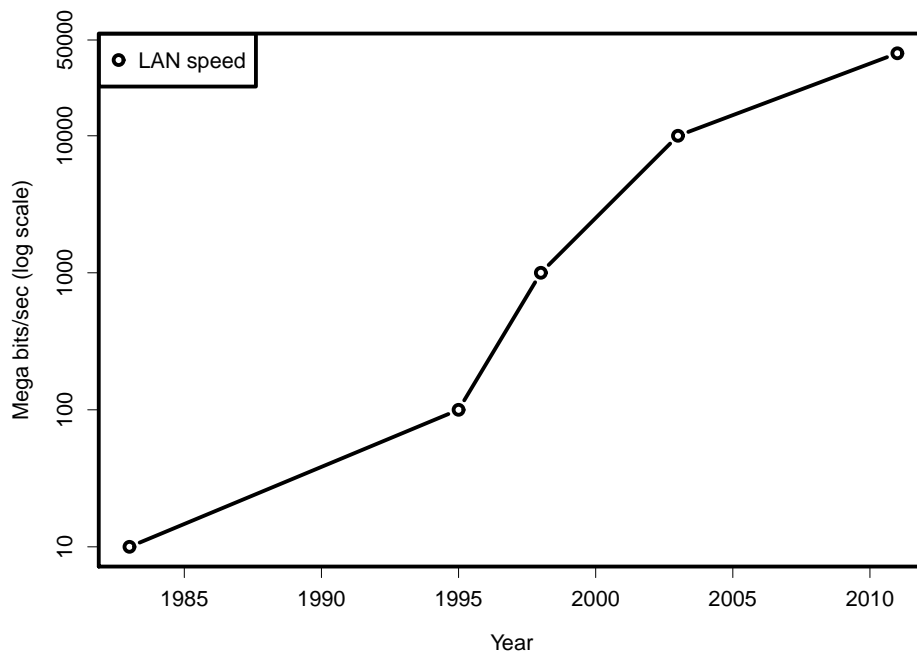


Figure 3. Hardware continues to improve (network speed). Standardized network speeds have been increasing exponentially 45% per year from 1983 - 2014.

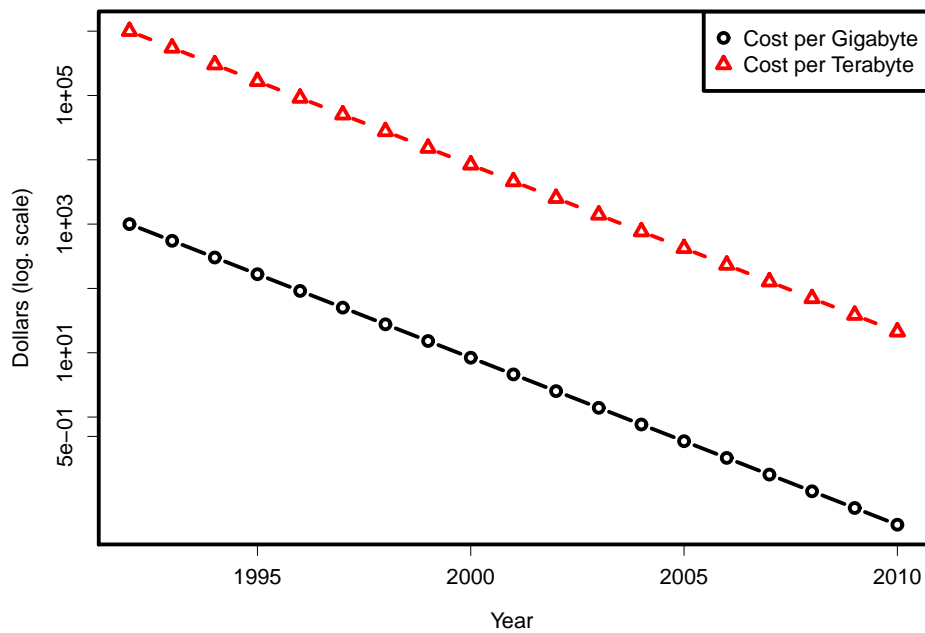


Figure 4. Hardware continues to improve (magnetic mass storage). The cost of magnetic mass storage has been declining at a 45% rate over the last decade.

by replacing “telecoms” with “computer resources,” the technological enablers will have an enormous and probably unpredictable impact.

1.3 LIFESPAN LIMITATIONS

As a broad statement, there are many different types of memory organizations that can act as repositories. They include: *individuals* acting by themselves in their own best interest, *commercial organizations* operating for a profit, *educational and academic* institutions acting in the pursuit of knowledge and tradition, *non-profit organizations* (such as the Internet Archive and Wikipedia) acting because they perceive a need that is not being filled, *museums* whose main charge to collect and preserve items of interest for future generations, and *government institutions* that produce and collect artifacts for the betterment of the population.

Human life expectancy in the US is currently about 77 years [16]. The life expectancy trend is nearly constant, or increasing slightly (Figure 5 on page 12). When addressing David’s desire to preserve the digital representation of his analog artifact, he can only reasonably expect that a human would be interested and able to act as an archivist for 20 to 30 years in the middle of their life. This 20 to 30 year period corresponds roughly to one generation.

We believe that commercial institutions may not be a much better choice for long-term viability. Currently (circa 2013), the US economy is recovering from the worst recession since the Great Depression of 1929. Institutions that were once thought to be invulnerable to outside influences and considered among the strongest in the world have gone through bankruptcy proceedings and if they emerge will be radically different entities than what they were. A digital object given to a commercial institution may not be any more likely to survive beyond one generation than if a single human were charged with looking after it.

Educational and academic institutions are also subject to being lost. Frederick College operated in Portsmouth, VA from 1958 until it closed in 1968 [17]. Carolina College for Women operated in Maxton, NC from 1907 until it closed in 1926 for financial reasons [18]. Antioch College in Yellow Springs, OH is an interesting case of a college returning from the dead [19]. Antioch College was founded in 1852 and closed in 2008 due to lack of enrollment. Antioch alumni raised enough money to restart the college in 2011 and is currently accepting students. We have examined the historical record and found that if an institution survives longer than 23 years

(one generation) then it has a higher likelihood of surviving longer. But first, it has to survive for one generation.

Government institutions are not immune either. The Texas Superconducting Super Collider lasted but a few years [20, 21]. The National Aeronautics and Space Administration (NASA), chartered to give its information and discoveries the widest possible publications [22], faces budget cuts and constraints. The Base Realignment and Closure Commission (BRAC) was charged with closing military installations [23]. When a governmental institution closes, the question arises as to who is responsible for the corporate knowledge of the closed institution.

The National Science Foundation (NSF) has recently added the requirement that all proposals include a Data Management Plan (DMP) [24]. The plan must describe the manner in which data and results from the NSF effort are to be disseminated to the community at large. While ensuring that the results of an effort are made available to members of the community, the DMP does not address how the data will be managed after the lifetime of the organization.

Each of these types of institution are subject to external pressures and may not be able to live up to their commitment to preserve digital data into the future. As stated by William Y. Arms (with our emphasis added):

*“Tomorrow we could see the National Library of Medicine abolished by Congress, Elsevier dismantled by a corporate raider, the Royal Society declared bankrupt, or the University of Michigan Press destroyed by a meteor. All are highly unlikely, **but over a long period of time unlikely events will happen.**”*

William Y. Arms [25]

The solution that David has to arrive at must meet and overcome all these limitations.

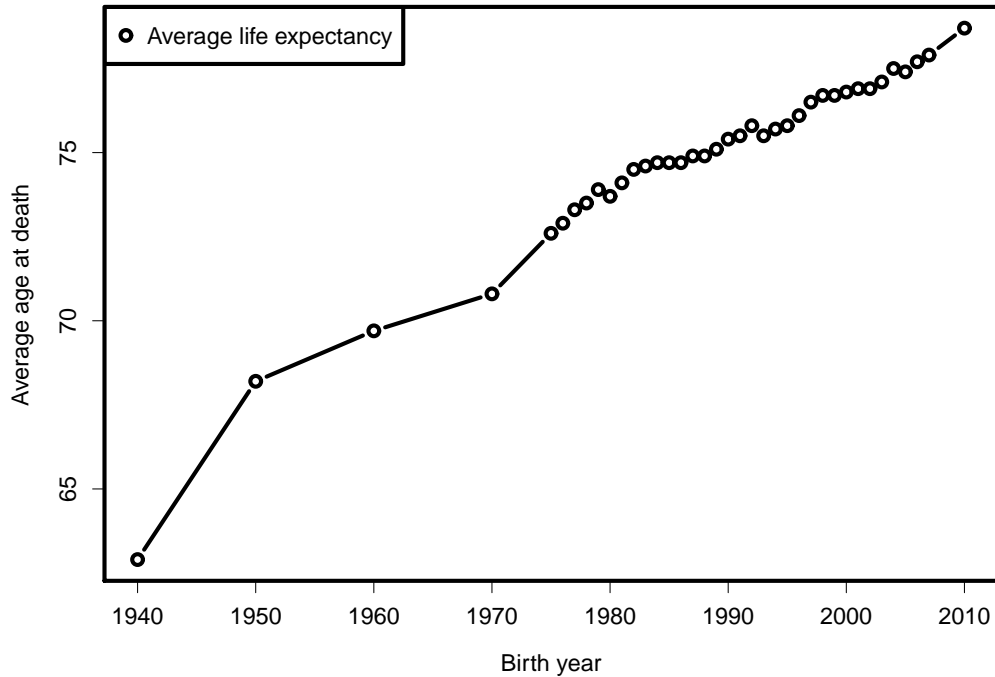


Figure 5. Average US life expectancy for both sexes, 1940 - 2010. The rate of life expectancy increase remains nearly constant of ≈ 0.2 years of age per calendar year.

1.4 SAFEGUARDING OF CONTENT

Every institution has to safeguard its contents against a host of events by which the data could be lost. A partial list of these events include:

1. *Change in operations*: due to closing of the institution, changing the institution's business model such that they are no longer interested in preserving some or all their data, or something as simple as inability to perform the required maintenance because of insufficient resources [26],
2. *Internet Service Provider's information technologies (IT) policies and practices*: that could deem that a particular type of legal digital data was no longer going to be hosted or supported [27],
3. *Failure of a critical piece of IT infrastructure*: the loss of a disk drive that had not been backed up recently would result in loss of data since its last backup [28],

4. *Human error*: whether due to negligence, malicious intent or simple misunderstanding [29],
5. *Censorship*: whether by active governmental censorship of the Internet [30], or ex post facto revisionist historians purging the records of data and ideas that do not fit the current dogma,
6. *Natural catastrophe*: including the traditional fire [31, 32], wind, water [33] and temperature. Changes in global climate may cause institutions to rethink their locations based on rising sea levels, weather patterns and population distributions, and
7. *Technological obsolescence*: of hardware and software without which access to the digital objects and what they represent would be impossible. Software obsolescence can be mitigated by continuous migration of files in the “old” format to the “new” format (specific examples include the image formats Kodak PhotoCD, Kodak RAW, and PICT [34]). The number of old formats will continue to increase because newer and better formats are continuing to be developed. Eventually the market place will decide that there is no financial incentive to support an old format and therefore there will not be a migration path available. Emulation of the systems that supported the old formats is an alternative way to ensure access to old data.

One way to increase the likelihood that data will be available and understandable for a long time, is to use commonly available technology and require minimal external support. An outstanding example of the application of these tenets is the Domesday Book [35] written in 1085 and still accessible today after almost a 1,000 years. An example of what can happen if there is too much dependence on specialized technology and esoteric external support is the BBC’s Domesday Project [36]. Between 1984 and 1986, the BBC compiled static and multimedia data from across the UK. The data was presented in 1986 using laser disks and a network of BBC micro computers. By 2002 there were fears that the data would be unusable because of the reliance on specialized technology [37]. The Creative Archiving at Michigan and Leeds: Emulating the Old on the New (CAMiLEON) project was created to access the data before it became totally irretrievable [38]. The CAMiLEON project used hardware and software emulation to retrieve the data [39].

1.5 RESEARCH QUESTIONS

Based on the previous sections, our research questions are:

1. *Can web objects (WOs) be constructed to outlive the people and institutions that created them?*
2. *Can we leverage aspects of naturally occurring networks and group behavior for preservation?*

We will address these questions by examining related works dealing with “emergent behavior,” graph-theory focusing on small-world graphs, and long term digital preservation. Based on these related works, we will develop a theory of unsupervised incremental graph creation that results in graphs with small-world properties. We will use a graph agnostic metric to measure damage to a graph caused by the removal of an edge or vertex and, based on that metric identify, the most advantageous attack profile that could be used by an entity bent on destroying a graph. This damage metric can be used to quantify damage to a graph, and the obverse of the metric can be used to quantify the most advantageous edge to restore to a graph after the graph has been attacked. The damage metric can serve as the heart of a “game” between an attacker and the graph to predict the long term future of the graph.

CHAPTER 2

BACKGROUND

2.1 BOIDS

Reynolds in 1987 created and defined the behavior of “boids” by which he sought to establish that three simple rules were sufficient to simulate the complex behaviors of schools of fish, flocks of birds, and herds of animals [40]. The rules themselves are simple, but the behaviors that emerge from the rules are complex and realistic. The salient feature is that these rules are scale-free: only the neighbors are accounted for in the computation; knowing the entire size of the group, or network is not required. A precise definition of neighbor does not exist and it is reasonable to say that it depends on the specific species that the boid represents and can be a combination of distance between boids, relative position (in front, behind, to one side), environmental conditions and perhaps other factors. Application of the “boids” concept to computer animation resulted the movement of collections of objects (schools of fish, herds of animals, flocks of birds, etc.) in ways that mimicked real life and did not require monolithic programs that controlled each entity individually. Reynolds’ approach imbued each object with simple directives that it used to decide how and where to move, a by product of this object oriented freedom, was that occasionally the objects behaved in accordance with the directives, but not in accordance with the animators wishes.

Unsupervised Small-World will implement these rules to create self-preserving digital objects with similar complex emergent behaviors. Table 3 lists the rules that Reynolds proposed for boids (his term for bird-like objects) and our interpretation for USW web-objects(WOs). While it is not directly possible to implement these concepts in the world of Web Objects, it is possible to mimic them. His ideas are further expanded as follows.

Collision avoidance is perhaps the easiest rule to visualize the transcription from boids to WOs. WOs flocking to a new repository cannot overwrite each other (collide in physical storage), nor collide in name-spaces (have the same URI). This is orthogonal to the naming mechanism used: URN implementations such as handles

or DOIs, globally unique identifiers (GUIDS) or content addressable naming schemes [41].

With boids, the concept of *velocity matching* (the vector quantity referring to the combination of heading, speed and change in altitude) is to travel the same speed as your neighbors. This is perhaps the most difficult rule transformation. However, interpreting velocity as resource consumption (i.e., storage space) makes this rule more intuitive. Specifically, a WO should try to consume as much, and only as much, storage as everyone else. In resource-rich environments, making as many copies of yourself as you would like is easy. When storage becomes scarce, this becomes more difficult. So there must be a provision for WOs to delete copies of themselves from different archives to make room for late arriving WOs in low-storage situations. WOs will never delete the last copy of themselves to make room for new WOs, but they will delete copies of themselves to come down from a soft threshold (e.g., 10 copies) down to a hard threshold (e.g., 3). When resources become plentiful again, new copies can be made.

For boids, *flock centering* means staying near (but not colliding with) other flockmates. We interpret this similarly, with WOs attempting to stay near other WOs as they make copies of themselves at new repositories. In essence, when a WO learns of a new repository and makes a copy of itself there, it should tell the other WOs it knows so they will have the opportunity to make copies of themselves at the new location if they wish. Announcing the location of a new repository will thus cause WOs at other repositories that have not reached their upper limit on creating copies to flow to the new repository.

Collision avoidance and velocity matching are complementary and the combination of these two rules results in the boids moving in the same general direction at roughly the same speed. Flock centering drives the boids towards one another, and prevents the boids from flying apart. The interaction of these rules results in emergent behavior that appears realistic and reasonable. Because the boids are driven by these simplistic rules; implementation is relatively simple, of low complexity and results in boids that have geometric and kinematic state, but do not have a significant mental state.

Table 3. Listing, interpreting and implementing Craig W. Reynolds’ flocking concepts in the USW framework. We are taking Reynolds’ concepts and applying them to a different type of movement model. Reynolds was interested in mimicking the movement of herds and groups of animals, while USW WOs exist outside the physical realm and have different abilities and limitations.

Reynolds’ Concepts	<i>Definition</i>	<i>Concept as interpreted and implemented within USW</i>
Collision Avoidance	Avoid collisions with nearby flock-mates	Each WO has a unique name or number. WOs flocking to a new repository cannot overwrite each other (collide in physical storage), nor collide in namespaces (have the same URI).
Velocity Matching	Attempt to match velocity with nearby flock-mates	Interpreted as the consumption of system resources, therefore use only as many resources as those in your “flock.” In resource-rich environments (lots of storage space available on lots of hosts), making as many copies of yourself as you would like is easy. When storage becomes scarce, WOs must be able to delete copies of themselves from different archives to make room for late arriving WOs.
Flock Centering	Attempt to stay close to nearby flock-mate	A preservation copy will be made on repositories discovered by your “friends,” or by following your “friends” to new repositories.

Reynolds’ work [40] is the first important step in incorporating behavior and autonomy concepts to classical computer animation systems. He proposes a “bottom up” approach and designs a system where a global and complex behavior emerges from a combination of several simple individual behaviors. Reynolds obtains synthetic flocks of birds, where the birds avoid crashing among them, maintain a constant velocity and remain within the flock [42]. When Reynolds’ concepts are applied to groups of autonomous entities their collective emergent behavior appears to be under the control of an omnipotent, omnipresent, and controlling entity. In fact each entity responds to only the small part of the collective that it can detect. We believe that Reynolds’ approach to crowd (or flock) behavior (Table 3) is applicable outside the field of computer animations, and specifically to the world of “crowd

Table 4. Measured and computed metrics for real-world small-world graphs.

Entity	Order	size	Actual		Random graph		Ref.
			$C(G)$	$L(G)$	$C(G)$	$L(G)$	
C. elegans	248	511	0.21000	2.87	0.05000	2.62	[45]
WEEC	4,941	6,594	0.08010	18.99	0.00054	8.70	[48]
ENRON	148	500,000	0.44000	2.25	0.11000	2.00	[47]

sourced curation” of digital objects. We believe that the emergent behavior of digital objects will be comparable to the behavior of objects under the control of an omnipresent controller.

2.2 SMALL-WORLD

We are interested in the structural aspects of “small-world” graphs because of their occurrence in many diverse and unexpected areas. Small-world graphs were popularized in main stream computer science by Watts and Strogatz [43]. They were interested in taking a lattice graph and perturbing each edge based on a probability p and understanding what happened as the graph went from a totally regular lattice graph to a totally random graph. Their investigation revealed an area where p resulted in a graph with a relatively high average clustering coefficient $C(G)$ when compared to a random graph of the same order and size and an average path length $L(G)$ that approximated a random graph of the same order and size. They declared that this phenomenon constituted a “small-world” in the same manner as Stanley Milgram’s small-world [44]. From a mathematical perspective, Watts and Strogatz laid out the following criteria for a “small-world:”

$$L(G) \geq L(G)_{random} \quad (1)$$

$$C(G) \gg C(G)_{random} \quad (2)$$

Small-world graphs also appear in nature (Figure 6 on the following page), in consciously created man-made entities (Figure 7 on the next page), and in unconsciously created man-made entities (Figure 8 on page 20). Metrics for these graphs (order, size, $C(G)$, etc.) are available in Table 4.

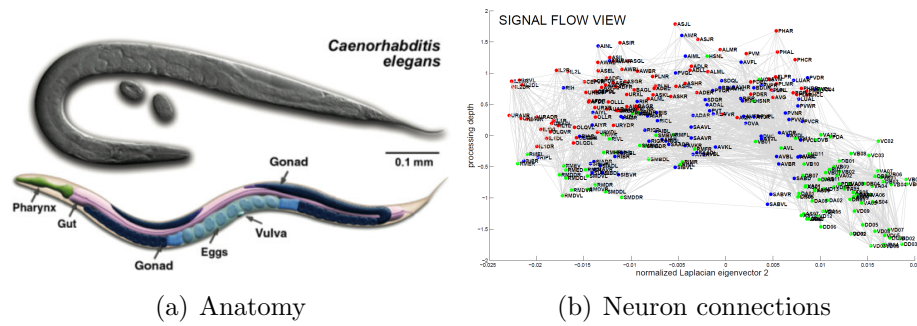


Figure 6. Naturally occurring small-world graph: nematode *Caenorhabditis elegans* neuron connections. Images from [45].

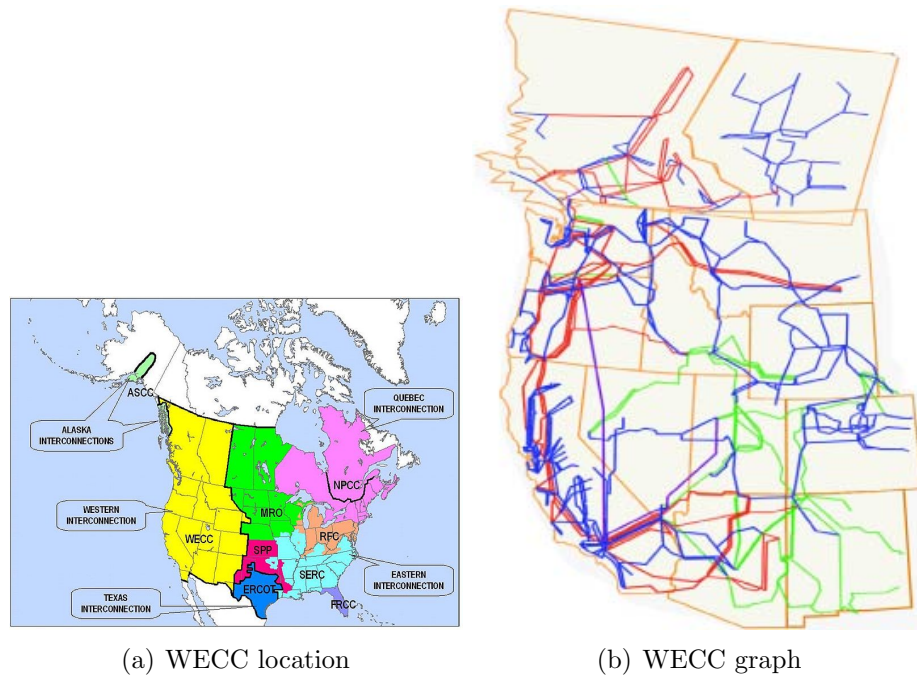


Figure 7. Man-made small-world graph: Western Electricity Coordinating Council. Images from [46].

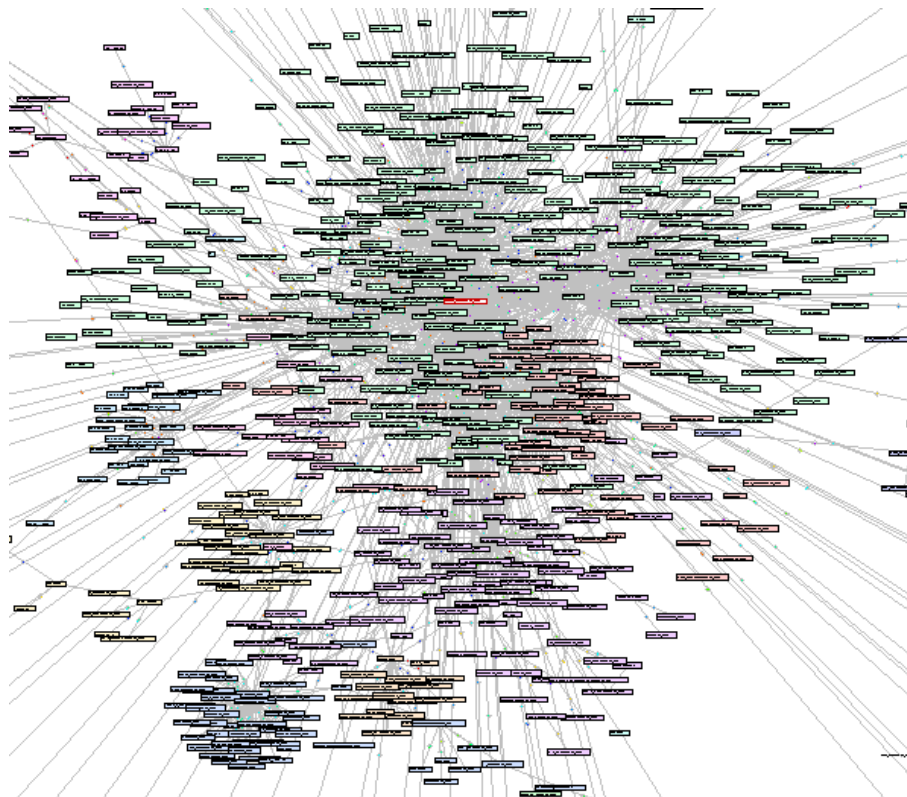


Figure 8. Organic small-world graph: ENRON e-mail exchanges. Image from [47].

2.3 CURRENT PRACTICE IN DIGITAL PRESERVATION

2.3.1 INTRODUCTION

Current practices in digital preservation range from institutions and organizations attempting to preserve their intellectual property (with varying degrees of success [49]) at one end of the spectrum to the promotion of digital preservation using cloud-based technology at the other [50]. These extremes could be characterized as ad-hoc (and probably doomed to failure because they are ad-hoc), or the creation of product that is geared to the preservation of medical and financial data and less for general needs of society.

In the following sections, we discuss another approach based on the application of current well-known standards and architectures combined with the intent of a repository to create an algorithm that will preserve data for a long time.

2.3.2 WORLD WIDE WEB CONSORTIUM (W3C) WEB ARCHITECTURE

The World Wide Web Consortium (W3C) has put forth an architectural recommendation [51] that discusses the core design issues for the World Wide Web (Web) to provide sufficient scalability, efficiency and utility resulting in a remarkable information space as the original technologies have evolved to increasingly complex and diverse system. Some of these design issues are: the use of *Uniform Resource Identifier (URI)*, the interaction between a URI and a resource, and the separation of content, presentation, interaction, and data formats.

A URI identifies one resource. A resource is used in the broadest of senses and can refer to something as limited as a single file, or as complex as initiating a long series of events “behind the scenes.” Multiple URIs may point to the same resource, but a URI can only identify a single resource.

An interaction is the communication exchange between servers at a given point in time. An agent is charged with carrying out some task on the Internet. Protocols on the Web are based on an exchange of messages (communication) between agents, where each message may include data and metadata about a resource. It is then up to a protocol, such as HTTP, to dereference the URI. The relationship between a URI, the resource that it identifies, and what the resource represents is shown in Figure 9 on the following page [51]. The result is a representation of the state of the resource. The format of the resource’s representation is dependent on a negotiation between the requesting agent and the responding server.

2.3.3 DIGITAL REPOSITORIES

“A repository is a network-accessible storage system in which digital objects may be stored for possible subsequent access or retrieval. The repository has mechanisms for adding new digital objects to its collection (depositing) and for making them available (accessing), . . . The repository may contain other related information, services, and management systems.”

Kahn and Wilensky [52]

We are concerned about digital repositories. That is, a generalized data repository where the collection is entirely digital. These repositories are composed of

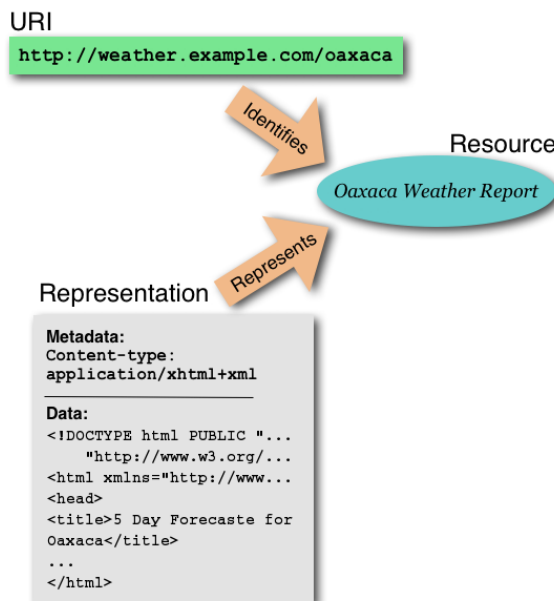


Figure 9. Relationship between URI, resource, and representation. URIs identify resources. When a URI is dereferenced, a representation of the resource’s state is returned. Image from [51].

hardware, software, and data that exist in an environment where communication between each repository is possible and supported without requiring human intervention.

Reich and Rosenthal in [53] identify two different models of digital repositories, *centralized* and *decentralized*. The *centralized* model contains a very small number of tightly controlled and administered repositories that do the entire job of preservation, requiring expensive hardware, and a sophisticated and highly trained technical staff. The cost of preserving data in this model is borne by a few, and providing ready access to their data may not be their highest priority. Centralized repositories are more concerned with the preservation of the bits, rather than access to the bits.

The *decentralized* model envisages a large number of loosely controlled and loosely administered repositories where each repository is only responsible for part of the preservation process. These repositories would have inexpensive hardware and require only minimally trained staff. The cost of preserving data would be borne by many and in proportion to the amount of its priorities and resources. Decentralized repositories are concerned with preserving access to the bits and less so with the bits themselves.

Kahn and Wilensky in [52] put forth an idea that is an extension of the *de-centralized* repository where a user could deposit a digital object in one or more repositories from which it may be made available to others. The idea that anyone could be a curator was extended by McCown in [54] and becomes a client-assisted preservation system. McCown's ideas are an extension of the original Open Archives Initiative (OAI) Object Reuse and Exchange (OAI-ORE) [55, 56] for instantiating aggregations of Web resources.

2.3.4 PRESERVATION TECHNIQUES

Preservation of digital data is a multidimensional problem. Jeff Rothenberg in [57], identifies the following technological dimensions to the problem:

1. *Digital media suffer from physical decay and obsolescence,*
2. *Digital documents are inherently software-dependent,*
3. *Additional considerations* include issues of corruption of information, privacy, authentication, validation, and preserving intellectual property rights.

Rothenberg concludes the report by making a case that the best approach to address these issues is to document the environment needed to run the software and then in the future recreate these environments as part of sophisticated software emulators. These emulators would run on machines as yet undesigned and unbuilt.

Kenneth Thibodeau in [58], an article about the vision of an Electronic Records Archive (ERA) at the National Archives and Records Administration (NARA) documents some of the early efforts at the San Diego Supercomputer Center (SDSC) dealing with the long-term issues of building a management architecture to support the preservation of arbitrarily structured sets of virtually any type of electronic records. SDSC's ideas and techniques lead to the Persistent Object Preservation approach. Key to this approach is the requirement that the digital object be internally self-documenting using ideas based on the Extensible Markup Language (XML). Self-documenting digital objects could be more easily processed by the Universal Virtual Computer (UVC) [59]. Digital archivists (DA) have a number of roles and responsibilities within the digital library (DL) [60]. These include:

- *Appraisal and Selection:* All systems have a finite (albeit large) amount of storage space, therefore it is not possible to store all WOs. The archivist is

responsible for the selection of those DO items that are deemed to be most valuable.

- *Accession*: Once a DO has been selected for storage, it must be prepared for the storage in the archives.
- *Storage*: The placement of the DO onto a media of some type. Consideration is also given to the anticipated frequency of access, number of redundant copies, and some sort of hierarchical organization.
- *Access*: Ensuring that the DL is accessible via a network with the appropriate bandwidth and protocols for delivering the WOs.
- *System engineering*: Defining and maintaining the interlocking requirements of media and data formats, hardware and software upon which the DL depends.

DA have to take a long-term view of the WOs under their purview. With this comes the realization that both WOs and the media on which they live will eventually become obsolete. In order to meet the DL's responsibility for long-term preservation and to address the continuing obsolescence problem, archivists must have strategies to migrate their WOs from the old to the new. These are a few migration strategies that are similar to (but different than preservation strategies):

Change media: Current magnetic and optical technology is subject to "bit rot" [61] and if left unattended will eventually corrupt enough to the media so that the DO will become unrecoverable. As more stable media becomes available, WOs on older and less stable media have to be copied from the old to the new. (Similar to *replication and refreshing*.)

Change format: A multiplicity of data formats can become unmanageable. A DL could decide to change or convert a DO from its original format to a more manageable and "standard" format. (Similar to *migration*.)

Incorporate standards: DLs, like any other user of digital data, benefits from adherence to well published and accepted standards.

Build migration paths: DL archivists can communicate and educate DO creators about better and more efficient techniques that can be used in the creation of DOs. Incorporating the ideas of digital preservation early makes the inevitable later migrations easier.

The Consultive Committee for Space Data Systems developed a reference model for an Open Archival Information System (OAIS) as a standard for how archive systems should be organized and operated [62]. A portion of their recommendation is called the “Functional Model” (Figure 10 on the next page). A summary of the major activities of the model are:

- *Ingest*: provides the services and functions to accept Submission Information Packages (SIPs) from Producers and prepare the contents for storage and management within the Archive,
- *Archival Storage*: provides the services and functions for the storage, maintenance and retrieval of Archival Informational Packages (AIPs).
- *Data Management*: provides the services and functions for populating, maintaining, and accessing both Descriptive Information which identifies and documents Archive holdings and administrative data used to manage the Archive,
- *Administration*: provides the services and functions for the overall operation of the Archive system,
- *Preservation Planning*: provides the services and functions for monitoring the environment of the OAIS, providing recommendations and preservation plans to ensure that the information stored in the OAIS remains accessible to, and understandable by, the Designated Community over the Long-Term, even if the original computing environment becomes obsolete.
- *Access*: provides the services and functions that support Consumers in determining the existence, description, location and availability of information stored in the OAIS,

A Submission Information Package (SIP) is a package of information about the digital artifact that a producer would submit with the artifact. A Dissemination Information Package (DIP) is disseminated with the artifact to the consumer. Archive Information Packages (AIPs) are not part of the USW algorithm. In the USW algorithm, the SIP is represented by metadata and ORE REsource Maps (Appendix D on page 383), created when the digital object becomes a member of the USW graph. The WO’s DIP will contain whatever data is associated with the WO’s retrieval by the standard WI retrieval mechanisms.

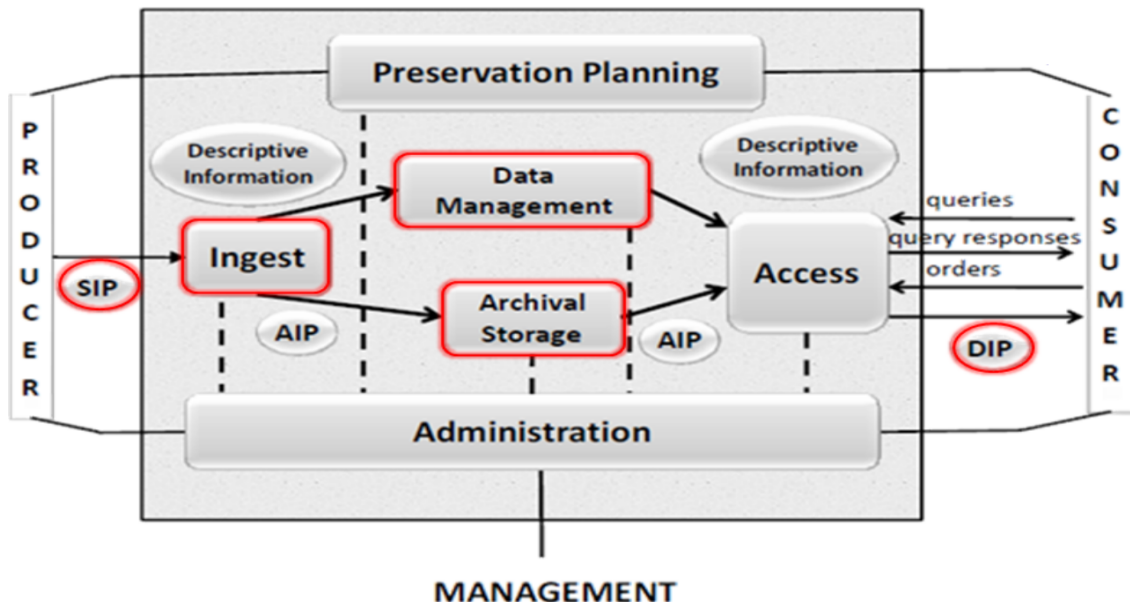


Figure 10. OAIS Reference Model functional model. The USW algorithm directly supports *ingest*, *data management*, and *archival storage* functions of the OAIS reference model. The *SIP* is created when the WO is created, and the *DIP* is created by the process that retrieves the WO. Image is taken from [62].

The USW algorithm reflects the intent of the OAIS ingest functionality.

2.4 SUMMARY

The current Web architecture has proven itself to be both robust and scalable. Therefore it can be expected to continue to support protocols that conform to Web standards and practices. Significant limitations in the current approaches to the long-term preservation of digital data have been identified. These procedural and mechanistic limitations apply to personal, commercial and non-commercial entities.

A framework using Web standards and practices that is not constrained by personal and institutional entities will be presented in the following chapters. That framework is called unsupervised small-world (USW).

The following chapters will describe specific aspects of the USW life cycle. These aspects are:

- *Creation*: when the USW graph is created and grows.
- *Attack*: when the USW graph is attacked by an adversary bent on disconnecting the graph.

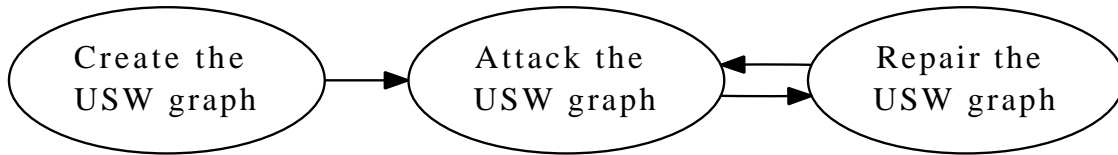


Figure 11. Navigational figure orienting USW graph life aspects.

- *Repair*: when the USW graph detects and recovers from damage caused by the attacker.

These aspects are shown in the navigational diagram (Figure 11). This image will appear in each chapter that deals with *creation*, *attack*, and *repair*. After a USW graph repairs itself, it could be subject to another attack, which is why there is a cycle between the attack and repair aspects.

During the *creation* and *repair* aspects of the USW graph's life, individual USW WOs will actively engage in fulfilling the migration strategies laid out above. Specifically,

Change media: every time a copy is made on a new host; presumably, a new media will be used. The media may be magnetic, optical, or some other technology, but the media will not be the same as the media where the original WO was located. Copying the WO to a new media is at the very heart of changing the media.

Change format: the USW algorithm is predicated on WOs exchanging messages (to create copies, to manage USW roles, etc.) to create and maintain the USW graph. A message could also announce to the WOs that a service exists to change data from one standard format to another. A WO receiving this message would then be able to change the format of some of its constituent parts.

Incorporate standards: the USW algorithm is predicated on adherence to Internet standards to host the WOs and to enable communication between the WOs. Services that can change the format of data presumably will be based on standards as well.

Build migration paths: the USW graph grows through the addition of new WOs.

These new WOs strengthen the existing graph by adding new potential preservation locations for existing WOs. New WOs can be added to the USW graph by DLs taking explicit action to include those WOs that they deem “worthy.” WOs could also be added to the graph by anyone by the addition of USW JavaScript “decorators” to existing Web pages. The addition of these “decorators” would enable anyone to be a DL curator [54].

CHAPTER 3

RELATED WORKS

Unsupervised Small-World (USW) brings together a disparate collection of collection of ideas and concepts from computer science and digital libraries. Ideas from graph theory (including metrics, robustness, and resiliency) and communication theory where message sender and receivers may be unknown when a message is sent. Each of these ideas and concepts are identified in the following sections.

3.1 EMERGENT BEHAVIOR

Emergent behavior systems are self-managing distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity to operators and users.

“The Autonomic Computing Paradigm has been inspired by the human autonomic nervous system. Its overarching goal is to realize computer and software systems and applications that can manage themselves in accordance with high-level guidance and direction from humans.”

Manish and Hariri [63]

Application of the USW algorithm results in a emergent behavior system.

- 2006

Harmen and Beneš [64] expanded on Reynolds’ rules to account for the behavior of a flock leader. They based their idea on watching the behavior of flocks where one bird would break away from the flock for a short time and act as a leader and then return to the flock and another assumes the role as leader. Their addition was to add a leadership attribute to each boid and that attribute changed value as the boid neared the periphery of the flock. USW incorporates their idea of changes in leadership roles to answer the question: Which WO is responsible for the preservation of a group of WOs (see Section 3.1.1 on the following page)?

3.1.1 DIRECTLY APPLICABLE

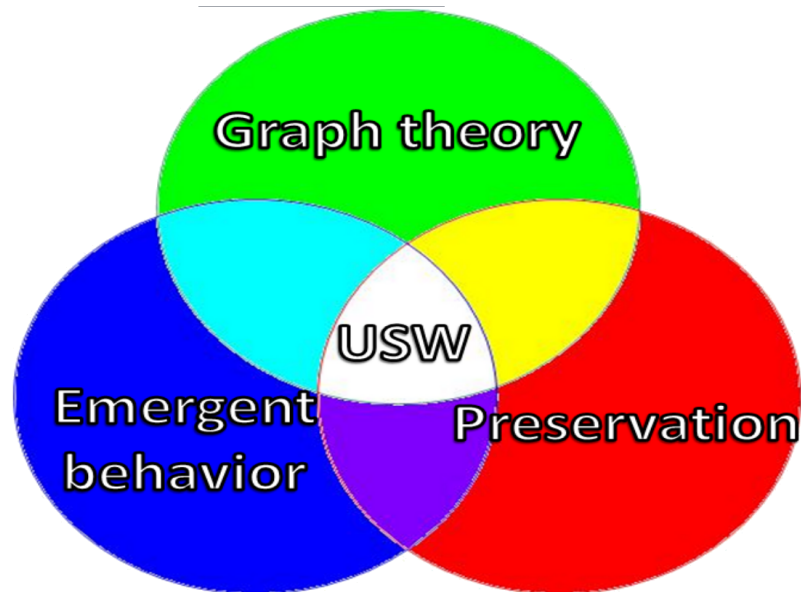


Figure 12. USW is at the nexus of multiple disciplines. The disciplines are: 1) *Graph theory*: mathematical structures used to model pairwise relations between objects, 2) *Emergent behavior*: movement of the inanimate, 3) *Preservation*: ensuring that digital information of continuing value remains accessible and useable.

Harmen and Beneš

Proper execution of the USW algorithm requires that a family have a single “active maintainer” at any particular point in time. If a family is split due to the network being split, then an outside entity would see there was more than one family, but to the family members in each partition, there would only be one family. As each WO is accessed, it executes a function that informs it, if it is the one responsible for the active maintenance of the USW graph (see Algorithm 9 on page 296). The act of becoming the active controller is at the heart of Harmen and Beneš [64].

3.1.2 COMMUNICATIONS

USW WOs must be able to communicate amongst themselves. There are many different communication models that could be employed. The challenge is finding an efficient model that has the following capabilities:

1. Point-to-point messaging,
2. One to many (multicast) messaging,

3. One to all (broadcast) messaging, and
4. One to unknown and unknowable recipients.

USW WOs require all of the above communications capabilities.

- *1989*

Carriero and Gelernter [65] propose a method of communication called Linda, between cooperating programs that does not require that either the sender or the receiver be explicitly identified. Rather, the sender attaches metadata to the message and places the combined digital structure into a *tuple space*. An unidentified receiver queries the tuple space looking for messages whose metadata match the query. The receiver then performs any necessary actions included in the message and may potentially return values to the tuple space for other processing or to return results to the sender. Using the concept of the tuple space: 1) the sender may never need to know the identity of the receiver, 2) the receiver may never know the identity of the sender, 3) messages can exist in tuple space after the sender ceases to exist and before any qualified receivers start to exist, 4) the same message can be sent to all members of an unknown group based on the metadata. USW uses the tuple space concept to send messages to unknown and as yet non-existent members of the USW graph (see Section 5.3 on page 121).

- *2002*

Iamnitchi et al. [66] look at how to locate data, in particular files in a peer-to-peer network in a fast and efficient manner. They approach the problem by examining the effectiveness of using a *gossip* based protocol that individual nodes can use to construct a “map” of the files in their local neighborhood. Based on these maps, they can quickly locate requested files. They put forth the idea that a small-world graph overlaying the actual graph structure would further improve the performance of the system, but do not offer a way to construct such a graph. USW could use the idea of employing a *gossip* based protocol to enable a WO to learn more about the total graph (Chapter 8 on page 275).

- *2004*

Newman [67] details an algorithm that can detect and display communities (clusters) in graphs with more than 1,000,000 vertices. The worst case execution time is $O((m+n)n)$ or $O(n^2)$ for sparse graphs. The algorithm identifies modules in the graph $\mathcal{Q} = \sum_i (e_{ii} - a_i^2)$ by partitioning the graph into communities based on the number of edges that connect the various communities. As the communities are identified, graph connected components of different sizes can be combined or distinguished.

- *2013*

Alam et al. [68], describe a mailbox style communications tailored to enable RESTful HTTP communication between sender and receiver. Messages are kept in persistent storage and are retrieved from storage based on search queries applied to the messages destination. Decoupling the absolute destination from the message allows for a message to be sent to unknown and unknowable recipients. This communications mechanism allows for temporal and locality separation between sender and receiver. USW uses the HTTP Mailbox as the prototype communication mechanism between WOs (see Section 5.3 on page 121). The USW implementation will use multiple mailboxes to spread the communications processing load across multiple independent hosts and to help minimize the possibility of a cascading failure [69].

3.2 GRAPH THEORY

Graph theory is the study of points (vertices or nodes) and lines (or edges) that connect them $G(V, E)$. Graphs are mathematical structures used to model the pairwise relationship between objects. Different relationships result in graphs that have different properties.

The USW algorithm will create graphs with small-world characteristics.

3.2.1 ROBUSTNESS AND RESILIENCE

Robustness and resilience describe the ability of the graph to remain connected and “functional” when some of its components have been compromised or removed. A “robust” graph is able to remain functional at some level after it has been damaged. A “resilient” graph is able to regain its functionality after it has been damaged and has had a chance to recover, or rebuild itself.

- *1990*

Najjar and Gaudiot [70] develop a probabilistic model of a network's resistance to disconnection based on the likelihood that any set of components can fail and then disconnect the system. Based on their analysis, they prove that graphs with a higher average degree are less likely to become disconnected when components start to fail. USW takes the idea of assigning a probability of failure to a component (either a friendship link, or a DO) and compute the resilience of the USW graph (see Section 3.2.5 on page 60).

- *1996*

Ang and Nadarajan [30] examine how Singapore has attempted to censor the content that its citizens can download from the Internet. They cite that Singapore has justified this censorship because of socio-political grounds by favoring caution and prevention over liberalism. Censorship is focused primarily on material going to homes more so than data going to companies, while material for public consumption is more heavily censored than material going to the home. Singapore has tried using separate servers and algorithmic based approaches. Singapore has been attempting to walk a thin line between fully controlling access by its citizens and enabling them to have all the benefits of the Internet. Inherently, USW will not have access controls, but the design and resiliency tests will take into account that the USW environment might be harmful to the USW's preservation intent.

- *2002*

Fiat and Saia [71] examine the problem of how to ensure that data is not effected by a censorship attack on the data content of Peer-to-Peer Content Addressable Networks. They look at networks that are resistant to censorship and spamming because data is not passed "homeward" if data from lower levels does not all agree. USW graphs are neither Peer-to-Peer nor is the content addressable, but they will ensure that the content they are charged with preserving are in fact the original contents. The ideas from this paper used to ensure that data is un-altered will be incorporated (see Section 3.2.5 on page 57).

Holme et al. [72] took Albert and Jeong and Barabási's [73] paper and expanded it by introducing the idea of using the *average inverse path length*

$(L(G)^{-1})$ as an approach to measure the vulnerability of a graph to different types of attacks. The use of $L(G)^{-1}$ as part of the metric to compute the $Damage(G)$ used in evaluating classical and USW graphs (see Section 5.2.4 on page 100).

Moreno et al. [74] investigate the stability of scale-free networks under node-breaking avalanches. They use the idea that a node has a fixed throughput capacity and that when this capacity is exceeded, then the node fails. The volume that the node was servicing is then spread back through the nodes that were feeding the failed node. This may cause these nodes to fail as well, resulting in an avalanche of failures because of the failure of one node. Based on simulations of medium sized graphs ($n = 10^5$ nodes), they conclude that almost 20 - 60% of their graphs nodes have to be removed prior to system collapse. As the average degree of the nodes increase the number of nodes that have to collapse increases (see Section 3.2.5 on page 60).

Holme and Kim [75] address the problem of vertex overload in an evolving system. They define vertex overload as a function of the number of geodesic paths that use a vertex and that the vertex can only support a fixed number of paths. If the number of paths exceeds the capacity of the vertex, then the vertex is defined as being overloaded and may breakdown. When a vertex breaks down, the paths that used that vertex are shifted to other vertices. If these other vertices are then overloaded, an avalanche of failures might occur result from a single failure (see Section 3.2.5 on page 59).

- 2004

Crucitti, Latora et al. [76] published a paper with the same title as Albert and Jeong and Barabási's [73], dealing with the same general topic, but proposing a metric they called *global efficiency*. Their global efficiency is *average inverse path length* $L(G)^{-1}$, but with a different name. They look at the behavior of a network (i.e., a graph that has a measurable flow along an edge) when a node or an edge is removed. Their premise is that the flow between nodes will always take the lowest cost path. In their models, each edge has a capacity and a tolerance factor. As edges/nodes are removed, the flow that was going through the removed component is spread out to other edges. The removal of a critical edge (high flow) and the redistribution of the flow through adjacent

edges can result in a cascade of failures as the increased flow causes additional edges to reach saturation. They investigated these phenomena for Erdős-Rényi random graphs and Barabási-Albert scale-free graphs using the same ideas of ID, IB, RD and RB as introduced in by Holme in [72]. Crucitti introduces the idea of *global efficiency* that has the same form and character as $L(G)^{-1}$.

$$E(G) = \frac{1}{n(n-1)} \sum_{i \neq j \in V} \frac{1}{d_{ij}} \quad (3)$$

Crucitti computes global efficiency after a node or an edge is removed, but they do not compare the current efficiency versus a connected graph's efficiency.

Petermann and Rios [77] explore how the tools used to discover the nature (connections between nodes via edges) can skew the data about the graph and therefore the analysis of the data may be suspect. Using the idea of the *traceroute* command to detect links between routers in real-world graphs, the authors argue that traceroute in general will report a short path (perhaps not the shortest path) between a source and terminus node. Because of selecting a short path between the source and terminus whenever possible, a tree like structure is returned from analyzing the data from the command. This tree like structure may represent the true nature of the graph because each router, node, or decision point may have more outgoing edges than are reported by traceroute. Petermann and Rios then construct several simulated graphs and show how using different discovery tools and techniques return different values for the same graph. They conclude that the discovery tool will bias the data and that a researcher needs to understand and account for these biases. USW will use the discovery tools that are part of the R *igraph* and *Matrix* packages [78, 79] to report on and evaluate the USW graphs.

Bollobás and Riordan [80] explore the robustness and vulnerability of the *linearized chord diagram* (LCD) and the diagram's robustness and vulnerability in the face of random damage and malicious attack. They show that an LCD graph is much more robust (and more vulnerable) than a classical random graph with the same number of edges. Also, under malicious attack, the critical portion p_c of vertices needed for a giant component is roughly 4 times as high for an LCD graph as a classical random graph. As part of their analysis, they investigate how effective an attacker can be when allowed to have only a

limited knowledge of the entire graph.

- *2005*

Criado, Flores et al. [81] propose to quantify the vulnerability of a graph based on the number of nodes, number of edges and the standard deviation of the degree of the nodes to random and intentional attacks.

Klau and Weiskircher [82] formalized Réka Albert and Hawoong Jeong and Albert-László Barabási's idea into the tuple $(S, \langle s \rangle)$. Their work is also a chapter in [83]. They provide a very nice survey of robustness and resilience metrics and ideas that have been advocated by various authors. A graph is robust if it is able to keep its basic functionality even when some of its components fail. Components can fail because of some random event, or because the component has been targeted to fail in order to cause damage to the graph. They explore and detail how to measure different aspects of a graph's robustness in the face of random failures and attacks and conclude stating that the ideal statistics for describing the robustness of a complex network depends on the application and the types of failures that are expected. Klau and Weiskircher hint that failure of a component in a real network may result in a cascade of failures across the network. None of the approaches provide a single unit-less value that describes the damage inflicted on a graph by the removal of an edge or node and the possible disconnection of the graph.

Link et al. [84] look into the parameters affecting the resilience of scale-free networks to random failures. They extend previous work by Cohen et al. in [85] by focusing on the Internet and estimating the percentage of nodes that must be removed during a random attack to cause a disconnection. They assign a likelihood of 0.5 deletion to each of the routers in the Internet and conclude that the Internet will collapse (as in no longer have a giant component) after the failure of 0.9 of the entire net. They generalize their results of various finite power-law networks based on analytical and empirical evidence. USW graphs should not have a power-law degree distribution, so much of the paper may not be directly applicable (see Section 3.2.5 on page 59).

- *2006*

Notetea and Pongor [86] proposed measuring the "robustness" of a network by computing the $L(G)^{-1}$ before and after a change is made to a graph under

consideration. If the robustness of the graph is improved, then the change becomes permanent. If the robustness decreases then the change is reverted. They focus on the evolution of a graph towards a new organization that is more robust or efficient. Their definition of efficiency is:

$$E(G) = \frac{1}{n(n+1)} \sum_{i \neq j \in V} \frac{1}{d_{ij}} \quad (4)$$

(which the same as other's $L(G)^{-1}$) and their definition of robustness $R(G) = \frac{E_t}{E}$. They use a genetic algorithm that starts with a random graph (100 nodes and 120 edges) and mutates and crossovers the graph until it reaches a “steady state” condition. A steady state was achieved when the goals of E , R and the maximum percentage of periphery nodes (those nodes with a degree of 1) was reached. E_t was computed after either 1 or 5 of the highest betweenness nodes were removed. Their idea of *robustness* R comes close to capturing our idea of a single number that measures the *health* of a graph. Health is the inverse of our idea of damage. Within USW, once edges are created, they are not removed or altered. The efficiency of the USW graph will increase by the addition of more nodes and edges.

3.2.2 ATTACK

Any component of a graph may fail at anytime. Examples of a graph failures include: a USW WO residing on a host whose power supply fails, or a network router that is shorted out due to flooding. An attack is a collection of targeted failures, whose collective goal is to disrupt the graph.

- 2000

Albert, Jeong and Barabási [73] look at the effect on the average (or expected) path length for scale-free networks (specifically snapshots of the Internet and the WWW) when the highest degreed node (be it an Internet router, or a well connected HTML page) is removed from the graph. Within their context, the Internet is a graph where routers equate to nodes and communications links equate to edges. They proposed a tuple metric (LCC, S, s) based on the proportion of the graph represented by the ratio of largest connected component LCC to the entire graph S and the mean size of all remaining fragments

$\langle s \rangle$. They conclude that these networks are tolerant of many random failures, but are very susceptible to the failure of a few critical elements because of their underlying structure. This type of sensitivity is common to scale-free networks. USW is not a scale-free graph and therefore should not be sensitive to targeted attacks. The sensitivity of the USW graph to targeted attacks will be evaluated (see Section 5.5 on page 148).

- *2002*

Motter and Lai [69] focus on the effects of cascading failures due to overloading of the Internet and power grids when using degree based attacks. In these types of graphs the traffic (be it either packets or electrical power) that was being serviced when a component fails is transferred to other components of the same type to which the failed component was connected. Their analysis shows that an attack, or a failure of an exceptionally heavily loaded component, may have a cascading failure effect on the other components. The possibility of a cascading USW graph failure when the messages are routed through a WO is a contributing factor to finding and using a different communications model.

- *2005*

Criado, Flores et al. [81] propose to quantify the vulnerability of a graph based on the number of nodes, number of edges and the standard deviation of the degree of the nodes to random and intentional attacks. Perhaps most importantly, they define the attributes of a vulnerability function in terms of the graph.

Their definition is:

Let \mathcal{G} be the set of all possible graphs with a finite number of vertices. A *vulnerability function* v is a function $v : \mathcal{G} \rightarrow [0, 1]$ verifying the following properties:

1. v is invariant under isomorphisms.
2. $v(G') \leq v(G)$ if G' is obtained from G by adding edges.
3. $v(G)$ is computable in polynomial time with respect to the number of vertices of G .

The equation they present to meet their definitions is:

$$v^{**}(G) = \exp\left\{\frac{\sigma}{n} + n - |E| - 2 + \frac{2}{n}\right\} \quad (5)$$

Supported by:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(k_i - \frac{2|E|}{n}\right)^2} \quad (6)$$

Equation 5 evaluates to the interval $[0,1]$. A value of 0 means that the graph is very robust (low vulnerability), while a value of 1 means that the graph is very vulnerable (not robust). Using equation 5 before and after a modification to a graph can be used as a way to measure what effect the change has had on the graph's vulnerability. If the vulnerability increases, then probably the change should not be finalized. While their system of equations meets their requirements, the equations do not report the type of damage that we are interested in measuring. Their definition of the attributes of a metric are in harmony with our intuition.

- *2006*

Kim et al. [87] examine the idea of increasing the resilience of a network as it grows by changing the connection criteria as a function of the life of the graph. They propose that a node be connected to an already existing node in the network based on the maximum, average, or minimum degree of the already connected node. These criteria require global knowledge of the graph. Based on their analysis, they recommend that new nodes be connected to an already existing node whose degree value approximates that of the total graph. USW nodes operate using only local knowledge and create additional edges in a random manner. In future work, an outside entity may steer WOs to selected WOs to increase the USW graph's resilience.

- *2008*

Lee et al. [88] look at how the topology of the graph affects which type of attack profile would be most effective. They propose a new metric, called *attack power* to quantify the effect of any of their attack profiles. They measure damage to their graph using degree distribution, average path length and vertex

cover. They enumerate some interesting attack profiles, but their approach does not address a disconnected graph. They use the autonomic system (AS) connectivity graphs from National Laboratory for Applied Network Research as their test graph. Based on this graph, they apply weights to each of the edges in the graph based on the amount of traffic along that edge. They then focus on three different types of failures. *Node failure* where an AS is lost due to some sort of hardware failure (i.e., power supply failure, accidental or deliberate misconfiguration, etc.). *Link failure* where adjacent ASes are not able to communicate because of hardware failure (such as the cutting of a cable), or electronic failure (such as DNS hacking, routing table poisoning, etc.). *Path failure* including DoS and routing table loops, resulting in a flooding of the path with packets to the extent that the communications links are unusable. Lee et al. then create different attack profiles based on these types of failures. Their attack profiles are:

1. *Random AS attack* — randomly choose an AS and remove it,
2. *Min-degree AS attack* — order the ASes by their degree connectivity and then start removing them from low degree to high degree order,
3. *Max-degree AS attack* — order the ASes by the degree connectivity and then start removing them from high degree to low degree order,
4. *Random edge attack* — randomly choose an edge and remove it,
5. *Min-weight edge attack* — order the edges by their weight and then start removing them from low weight to high weight order,
6. *Max-weight edge attack* — order the edges by their weight and then start removing them from high weight to low weight order,
7. *Random path attack* — randomly choose a path and remove it,
8. *Max-weight edge attack* — order all paths by weight and then remove paths in order from heaviest to lightest, and
9. *Max-length path attack* — order all paths by length and then remove paths in order from longest to shortest.

After each attack, the effect on the graph is quantified by a metric they labeled as “attack power.” Attack power reports the effect of each attack on the number of components that fail in the system. We treat Lee’s path failure as

a limited case of our edge failure (see Section 5.5.3 on page 153). Path failure is based on the path at the start of the attack where the path meets some sort of criteria and then a series of edges are removed based on these criteria. The limitation is that the set of criteria used to identify the path in the first place, may not be valid after the removal of the first edge in the path. Their conclusion is that the Internet is more vulnerable than a random graph to path based attacks. Our method of testing the robustness of the USW graphs will be slightly different than those proposed in [88] in that we will reevaluate the effect of an attack after each edge or WO removal to ensure that only the latest information about the graph is used.

Kownacki [89] examines the robustness of planar random graphs to targeted attacks. He focuses on connected cluster distribution when all faces whose number is greater than a cut-off k_{max} are removed. His investigations point to a percolation value of $k_c = 15.8$ that above which the connectivity of the graph is greatly affected and below which the graph remains largely undamaged. His experiments were based on using graphs ranging in size from 1,000 and 32,000. He concludes that his simulated graphs are “rather” robust with up to $\approx 20\%$ of the nodes having to be removed prior to the disconnection of the graph. USW graphs are not planar and therefore Kownacki’s analysis and results are not directly applicable, but they do serve as a benchmark to be measured against.

3.2.3 SMALL-WORLD

A small-world graph by definition has the following properties [90]:

1. Is of a large order n ,
2. The order is much much greater than the average degree and much much greater than 1 $n \gg \langle k \rangle \gg 1$,
3. The order is much much greater than the maximum degree of any vertex $n \gg k_{max}$,
4. Is sparsely connected $|E| \ll n^2$,
5. $L(G) \sim \frac{\ln(n)}{\ln(k)}$, and

6. Whose $C(G) \gg C(G)_{random}$

USW graphs will be meet the definition of a small-world graph.

- 1998

Watts and Strogatz [43] take a regular lattice graph of fixed size and randomly “re-wire” edges based on a probability p . They compare the average path length $L(G)$, and the average clustering coefficient $C(G)$ of the “re-wired” graph to the original graph. As p varies from 0 to 1, the normalized $L(G)$ and $C(G)$ are nearly 1 when p is nearly 0 and nearly 0 when p is nearly 1. These two regions correspond to a regular graph and random graph regions, respectively. Between these extremes, there is a region where the $L(G)$ begins to drop from the regular region to the random region while the value of $C(G)$ remains in regular region (Figure 13 on the next page). This area where $C(G)$ is $\gg L(G)$ is designated as the “small-world” region. Any graph that exhibits this kind of relationship between normalized $C(G)$ and $L(G)$ values is, by definition, a small-world graph. USW graphs are small-world graphs based on this definition.

- 1999

Watts [90] provides an overview of the small-world graph related phenomena and the derivation of region of average path length and average clustering coefficient that characterize a small-world graph. The definition that Watts gives for a small-world graph is:

“A small-world graph is a large N , sparsely connected, decentralized graph ($n \gg k_{max} \gg 1$) that exhibits a characteristic path length close to that of an equivalent random graph ($L \approx L(G) \sim \frac{\ln(n)}{\ln(k)}$), yet with a clustering coefficient much greater ($C \gg C(G) \sim \frac{k}{n}$).”

Watts – Strogatz [90]

Watts puts forth the conjecture that any graph that meets the average path length and clustering coefficient requirements of his definition is a small-world graph regardless of how it was constructed. USW graphs are repeatedly evaluated against Watts’ definition to ensure that they meet the requirements of a small-world.

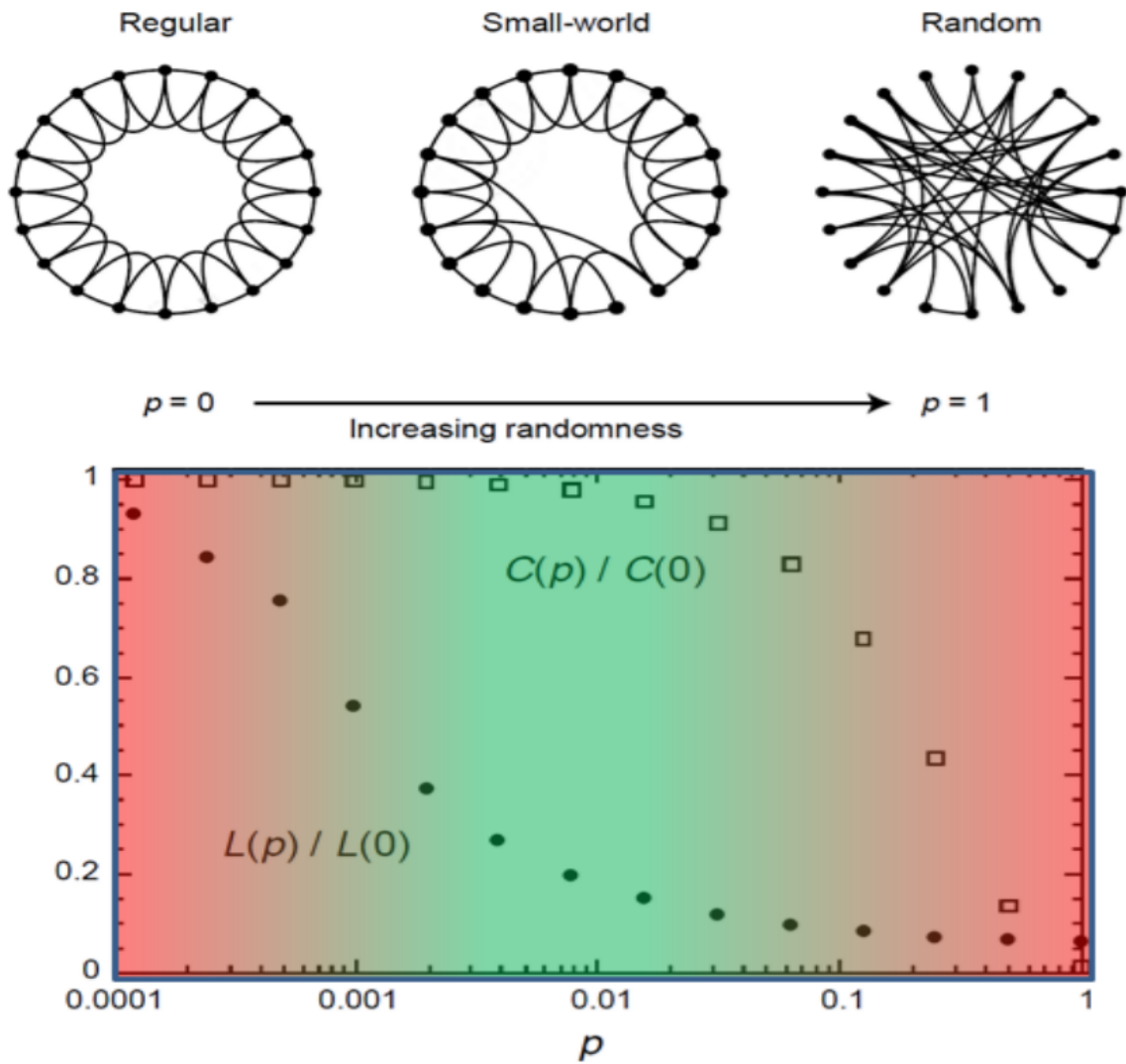


Figure 13. Watts and Strogatz lattice evolves to a random graph. The graph evolves as the probability p of rewiring an edge increases from 0 to 1. A small-world exists between the total regularity of a lattice and the total randomness of a random graph (both shown in red).

- 2000

Amaral et al. [91] put forth the position that scale-free graphs are in fact a case of small-world graphs. They further classify these special case small-world graphs into: *scale-free* characterized by a degree distribution that decays as a power law, *broad-case* networks characterized by a degree distribution that has a power law regime followed by a sharp cut-off, and *single-scale* networks characterized by a degree distribution with a fast decaying tail. They explored this area based on the question: “What are the reasons for such a tight range of possible structures for small-world graphs?” The answer put forth is that preferential attachment leads to growing networks with power law degree distributions, and that preferential attachment can be affected by two factors: *aging* of the vertices, and *cost* of adding additional edges to a vertex. USW will create small-world graphs, but it has not been classified yet based on Amaral’s taxonomy.

Border et al. [92] examine 200 million pages with 1.5 billion links from the WWW and conclude that the Web is considerably more intricate than had been suggested by smaller scale experiments. Based on the real-world data they used, they compute that the fraction of the Web with i in-links is proportional to $\frac{1}{i^{2.1}}$. Interestingly enough, they find that their in-degree data more closely fits a Zipf distribution than a power-law distribution. Another interesting observation is that given a random start and finish page, one get from the start page to the finish page only about 25% of the time. USW friendship links are bidirectional, so some of the ideas from this paper are not applicable. Based on particular values of β and γ , the USW degree distributions will be analyzed to see if they fit a power-law distribution (see Section 5.2.1 on page 84).

Callaway et al. [93] look at predicting the percolation point in random graphs where the degree distribution is other than Poisson. Those efforts are applicable to real-world graphs (such as the Internet and others). They give exact solutions for a variety of cases, including site and bond percolation and to models in which the occupation probabilities depend on the vertex degree. Part of their motivation is to compute the point in graph development where a large giant connected component appears that permits the underlying graph of become fully functional (see Section 3.2.5 on page 57).

Kleinberg [94] claims to prove that there are no decentralized algorithms using only local knowledge that can construct a graph with small world properties for clustering coefficients and average path length. He started with a lattice based graph and uses messages that collect intermediary data as they are passed from node s to t . Based on the information collected in these messages, long jumps/links can be constructed that shorten the overall average path length. Our USW approach is similar in that the USW node actively collects information about the structure of the graph while in the wandering phase and passively from newer wandering nodes after it has become connected (Appendix A on page 287).

Kleinberg [95] algorithmically bridges the gap between Stanley Milgram's [44] and Watts and Strogatz's [43] small-world phenomena. Kleinberg uses an underlying lattice as a baseline where each node represents a potential message sender and receiver. Each sender and receiver has directed links to nodes close to them and a very small set of links to nodes that are far away. In this sense, close and far are determined by the lattice distance between the sending and receiving nodes. Kleinberg then poses the question: *Why should arbitrary pairs of strangers be able to find short chains of acquaintances that link them together?* Kleinberg approaches solving this by analytically sending messages from any sender to any receiver. The sender knows the location of the receiver and its own close and distant links. As the message is passed from node to node, all nodes that touch the message record that fact in the message. As the message is routed through the lattice, each node begins to accumulate a more complete understanding of all links in the graph. Kleinberg uses this model to prove that there are no decentralized algorithms using only local knowledge that can construct a graph with small-world properties for clustering coefficients and average path length. Our USW approach is similar in that the USW node actively collects information about the structure of the graph while in the wandering phase and passively from newer wandering nodes after it has become connected (Appendix A on page 287).

Mathias and Gopal [96] investigate the creation of small-world graphs arising from base graphs other than lattices. They focus on using simulated annealing to create graphs that have a trade-off between maximal connectivity and minimal wiring and settle on an alternative creation that connects small hubs to

create a larger graph that has the desired small-world characteristics of high clustering coefficients and short average path length. USW simulations use their ideas of connecting subgraphs into a larger whole while evaluating the effectiveness of damage metrics.

Newman [97] provides a short review of small-world (SW), their characteristics and techniques by which they can be created. He starts by contrasting SW with random graphs and then proceeds to discuss the SW model of Watts – Strogatz. Newman provides average path length limits based on his previous works. He concludes with SW models based on graphs with a limited number of “well connected” nodes a discussion of the World Wide Web as SW by Albert et al. and models by Kleinberg based on Manhattan distance (L_1) computations. USW continuously evaluates its self in terms of average path length ($L(G)$) and clustering coefficient ($C(G)$) and remains within the bounds developed by Newman.

- *2002*

Holme and Kim [48] look at the problem of growing scale-free graphs with high clustering based on the use of a control parameter. They look to fill the void between classic Watts – Strogatz small-world graphs that have high clustering but without a power law degree distribution and the Barabási and Albert (BA) scale-free graphs that have low clustering and a power law degree distribution. Holme and Kim achieve this combination distribution by taking the BA graph construction algorithm using preferential attachment and adding an additional random edge between the new node and the neighbors of the node that the new one is attaching to. Holme and Kim focus on controlling the clustering coefficient in a scale-free graph based on a control parameter. The USW algorithm creates small-world graphs based on two control parameters (β, γ) and application of Policy C.

Newman et al. [98] attempt to model social networks via the use of random graphs. In some cases, their efforts based on degree distributions match well with real data, while in others they match less well. They found that large random graph has a degree distribution that can be expressed by a Poisson distribution that does not match well with the degree distribution of a random graph. They were able to exactly compute the size of the largest connected

component S and the average size of the remaining components $\langle s \rangle$ based on a specified degree distribution. Using these data, they proposed a way to compute the average path length.

- *2004*

Barrat et al. [99] put forth the idea that more can be learned about “large real-world” graphs if the edges between the nodes in the graph are assigned a “weight” that is appropriate for the graph. They base their claim after studying the International Air Transportation Association database of seats available between airports and the author citation database from e-Print Archive. Using information particular to the graph (number of seats, number of co-authored papers, etc.), they compute a weight for each edge. Based on this weighted edge, the more critical and important edges can be determined, where before the edges may have been overlooked. USW will use this type of idea about edge centrality to test the graph’s resilience to high edge centrality attacks (see Section 5.5 on page 148).

Hui et al. [100] consider how to construct a structured peer-to-peer network with small-world clustering and path length characteristics. They present a small-world overlay protocol (SWOP) that is used to maintain a network consisting of head nodes, inner nodes, long links and cluster links. They use a SHA-1 function to ensure that each node has a unique identifier. This hashing function is also used when a node wants to join the network. SWOP has various sub protocols including: Join Cluster, Leave Cluster, Stabilize Cluster and Object Lookup. The idea of using SHA-1 as a way to generate a unique ID is appealing, but multiple protocols does not fit well with our current USW model. Their Object Lookup protocol may be applicable to locating a USW DO that has a particular data load.

Kleinberg [94] takes a lattice graph and creates long links originating from selected nodes s based on the distance to other selected nodes t based on the distance between the nodes. Kleinberg states that these “long links” are useful in the design of peer-to-peer file sharing systems on the Internet. An effective routing algorithm can arise based on purely locally gleaned information. Our USW algorithm is different from Kleinberg’s in that the USW environment does not have a concept of distance. We will investigate a variant of Kleinberg’s

idea of distance by giving connection preference to nodes that are directly connected to a node that the USW node connects to (Section 6.7.3 page 200).

- *2005*

Bonato [101] provides a survey of various models for growth of the Internet (he calls it W), including preferential attachment, off line, copying and growth/deletion. He provides a list of desirable W graph properties, including:

1. *On-line property.* The number of nodes and edges changes with time.
2. *Power law degree distribution.* The degree distribution follows a power law, with an exponent $\beta > 2$.
3. *Small-World property.* The average distance (or diameter) is much smaller than the order of the graph.
4. *Many dense bipartite subgraphs.* The number of distinct bipartite cliques or cores is large when compared to a random graph with the same number of nodes and edges.

USW uses some of the techniques that Bonato discusses in his copying model when a USW WO makes connections to some of the WOs that it has learned about during its *wandering* phase (Appendix A on page 287). USW uses some of the techniques discussed in the growth/deletion model when assessing the robustness of the USW graph (see Section 5.2.1 on page 84).

- *2008*

Cont and Tanimura [102] investigate the creation of graphs with small-world properties that do not start from an initial lattice. They provide a technique for constructing graphs that meet the clustering coefficient and average path length requirements to be classified as a small-world starting from a collection of disconnected clusters and then connecting them. They do not present a construction technique corresponding to the USW approach where the nodes are added one at a time and make connections based only on the information that they gather.

3.2.4 MISCELLANEOUS

- 1949

Luce and Perry [103] provide an early example of using an adjacency matrix to identify cliques in a graph. The authors focus on the problem of three or more nodes n having a relationship such that $(n_i \rightarrow n_j \rightarrow n_k \rightarrow n_i)$ only. By the use of matrix multiplication, they provide a technique that says $x_{ij}^{(n)} = c$, iff there are c distinct n -paths from i to j and an element i is contained in a clique iff the i^{th} entry of the main diagonal of S^3 is positive. USW is more concerned about the clustering coefficient $C(G)$ than the presence of cliques. The techniques and approaches outlined by Luce and Perry are useful for computing $L(G)$ during simulations and evaluations.

- 1973

Fiedler [104] explores the second smallest eigenvalue ($a(G)$) of the adjacency matrix $A(G)$ algebraic connectivity of the graph G . Results of this exploration come to be known as the *Fiedler* vector. The *Fiedler* vector is used to explain the relationship between the second eigenvalue and the usual vertex and edge connectivities. USW uses the *Fiedler* vector when evaluating which edge or node to remove to cause the greatest amount of damage to the USW graph when it is under attack.

- 1976

Milgram [44] reported on two experiments where he started a chain letter in the American Mid-West with a target in the Eastern US. The chain letter was to be passed from one person to another, only if the first person knew the second. The fewest number of intermediate people in the chain was 2, while the highest was 11. There was a pronounced peak at 6 intermediaries. Milgram's paper is credited with measuring the anecdotal phenomenon where complete strangers find that they have someone in common. The peak in the reported data, led to the phrase "we are all separated by six degrees of freedom." Our USW graphs may have more than 6 degrees (based on β , γ , and n), or edges between any random node, but the average path distribution length mimics that measured by Milgram.

- 1993

Huberman and Adamic [105] examine data from Alexa and Infoseek crawls of

the Internet. Based on this data, they present a power-law distribution for the number of links to a particular site. Using this data a growth factor $-\beta$ can be estimated using the equation $p(k) = p(k_0) * (\frac{k}{k_0})^{-\beta}$. As an extension, the number of sites that that would be expected to be located after N crawls is $N * p(k)$. USW creates small-world graphs, vice preferential attachment graphs that are the main source of data for this evaluation. Construction of the USW graph based on the USW algorithm will create a graph that will increase in density $\rho(G)$ as more WOs are added (see Section 5.3.2 on page 125).

- *1997*

Randić and DeAlba [106] look to quantify the distinction between sparse and dense graphs, where before these were discussed as “qualitative” attributes of graphs and therefore open for interpretation. They develop their equation by examining the ratios $\frac{E}{E^*}$ (the ratio of the actual number of edges over the maximal number of edges) and $\frac{Z^*}{Z}$ (the ratio of the size of the adjacency matrix over the number of 0 entries in the adjacency matrix). Using these two ratios, they propose the metric $\rho(G) = (\frac{n^2}{2m} - 1)(1 - \frac{1}{n})$. If $\rho(G) < 1$ then the graph is sparse. If $\rho(G)$ is > 1 then the graph is dense. Because $\rho(G)$ can never equal 1, there are no ambiguous conditions. USW graphs start out as sparse, but become more dense as they grow (see Section 5.3.2 on page 125).

- *2001*

Barabási [107] provides a statistical overview of the Web and general graph theory. He describes directed and undirected graphs, the salient characteristics of Poisson and power-law graphs, and estimates on the upper limit on the size of the Web (19 clicks from one end to another). He addresses the differences between static and dynamic graphs and how most graph theoretical work has been directed towards static graphs. How the Internet can be attacked and the effectiveness of various attack profiles (edge vs. node) would be in causing a disconnection and wide spread disruption. USW graphs are dynamic and small-world. The paper is written towards a general population, but has ideas and thoughts views that have been incorporated into USW implementation and testing (see Section 5.5 on page 148).

Brandes [108] presents a faster algorithm to compute betweenness centralities. Computing betweenness is a $O(n^3)$ time and $O(n^2)$ space problem, but this

approach brings it down to $O(nm)$ time or $O(nm + n^2 \log n)$ time and $O(n + m)$ space making the computation of much larger graphs practical. These techniques were used during simulation and graphs of size 10,000. The R language igraph library [78] was used when creating the reference implementation robot (Chapter 7 on page 253).

Cooper and Garcia-Molina [109] investigate the idea that autonomous repositories may enter into trading agreements with each other to provide preservation services by trading data between themselves. They conclude that the key issue is how a site can determine with whom to enter to a trading agreement with. Based on the agreements that can exist between different sites and the way that these agreements may be advertised, a complete preservation trading network could be created. USW will match very closely to many of the ideas of the trading network (see Section 3.2.5 on page 57). USW hosts provide *autonomous archive sites*, *digital storage* and indirectly *a trading network*. USW WOs provide *archiving clients* and *automation*.

Goh et al. [110] investigate the problem of data packet transport in scale-free networks whose degree distribution follows a power law with the exponent γ . Using the idea of vertex betweenness that computes the betweenness of each vertex based on the number of geodesic paths that use that particular vertex, they find that the load distribution follows a power-law distribution the exponent $\delta \approx 2.2$. Based on their investigations, they conjecture that the load exponent is a generic quantity that can be used to characterize scale-free networks. USW used some of their ideas about how the degree distribution affects message transmission as part of an investigation into the effectiveness of message and communication costs as new WOs are added to an existing graph (see Section 5.3 on page 121).

- *2002*

Menczer [111] looks at the linkage structure of Web hypertext pages and the content topology of Web pages. Menczer defines a lexical distance between Web pages using Information Retrieval techniques and then argues that the Web is built in a cognitive manner using the lexical distance. In effect, Web page creators are more likely to link to pages that are similar to themselves.

Menczer validated his approach by using 150,134 URLs from the Open Directory Project, and re-creating the power law distributions seen there. USW will keep the idea of linking to nodes based on their content as an alternative way for the wandering node to make its first connection (Chapter 8 on page 275).

- *2003*

Newman [112] provides a survey of random graphs and shows how well, or not the random graph's degree distribution matches real-life networks/graphs. He discusses random graphs with specified degree distributions, directed and undirected graphs, graph clustering and the application of these ideas to epidemiological simulation models.

Wang and Chen [113] provide an easy to read and understand survey of random, small-world and scale-free graphs. They provide equations for the distinguishing characteristics of each type of graph and address the "Achilles heel" — robustness versus fragility of scale-free networks. How scale-free are robust (as in the network is still able to function reasonably well) in the face of the loss of a very high percentage (80%) of randomly selected nodes, but it is very sensitive to a targeted attack against a very small percentage of nodes. They also address the idea of synchronization of a graph, where adding just a few links to an existing graph can have profound effects on the average path length, and can convert a regular graph into a small-world one. USW graphs have been evaluated in light of the ideas in this paper (see Section 5.5 on page 148).

Newman and Girvan [114] investigate how to detect communities in a graph without any foreknowledge of the graph. They approach the problem by computing the edge betweenness centrality $c_B(e)$ for entire graph, identifying that edges as being between two communities, and then removing that edge. The process is repeated until all edges have been removed and all nodes are isolated. This decomposition will identify all communities in the graph that are greater than or equal to some arbitrary n . Betweenness can be based on the number (or percentage) of geodesic paths that use an edge, by the flow through an edge (if there are different flows through different edges), or the number of random walks (a Markov approach) that use a particular edge. USW graphs have been evaluated based on ideas and techniques from this

paper (Section 5.5 page 148).

Radicchi et al. [115] tackle the problem of how to detect communities in large networks in a fast and efficient manner. They define a community as a portion of the graph (a subgraph) where the connections between nodes (the edge density) is higher than the average in the total graph. Radicchi et al. use an agglomerative algorithm that builds communities up from nodes, as opposed to a divisive algorithm where the graph is cut into subgraphs based on the edge's centrality value. The computational difference between the two approaches is that $O(M^2N)$ for the agglomerative algorithm and $O(MN)$ for the divisive. The agglomerative is too computationally expensive for large graphs, while the divisive is practical.

- 2005

Aberer et al. [116] propose a reference model for the study of P2P networks. They authors claim that the diversity of approaches to describe P2P networks has been brought about by the origins of the different most popular P2P networks creators. These creators come from different communities (networking, databases, distributed systems, graph theory, and others) and as such the terminology used to describe each aspect of each system is inconsistent. This inconsistency results in confusion when comparing different P2P architectures and makes it difficult to properly evaluate each type of network using the same set of criteria. The reference model takes into account the key components of P2P design consideration:

1. *Efficiency*: Routing should incur a minimum number of overlay hops (with minimum “physical” distance) and the bandwidth (number and size of messages) for constructing and maintaining the overlay should be kept minimal.
2. *Scalability*: The concept of scalability includes many aspects. We focus on numerical scalability, i.e., very large numbers of participating peers without significant performance degradation.
3. *Self-organization*: The lack of centralized control and frequent changes in the set of participating peers requires a certain degree of self-organization, i.e., in the presence of churn the overlay network should self-reconfigure

itself towards stable configurations. This is a stabilization requirement as external intervention typically is not possible.

4. *Fault-tolerance*: Participating nodes and network links can fail at any time. Still all resources should be accessible from all peers. This is typically achieved by some form of redundancy. This is also a stabilization requirement for the same reason as above. Fault-tolerance implies that the partial failure property of distributed systems [19] is satisfied, i.e., even if parts of the overlay network cease operation, the overlay network should still provides an acceptable service.
5. *Cooperation*: Overlay networks depend on the cooperation of the participants, i.e., they have to trust that the peers they interact with behave properly in respect to routing, exchange of index information, quality of service, etc.

USW graphs, while not strictly a P2P network in the classical sense, have been evaluated based on *efficiency* (see Section 5.3 on page 121), *scalability* (Appendix A on page 287), *self-organization* (Appendix A on page 287), *fault-tolerance* (see Section 5.5 on page 148) and *cooperation* (Appendix A on page 287).

Borgatti [117] provides an interesting view on how different types of flows through a network can result in different centrality values for edges or nodes in that network. The types of flows he studied in his simulations include: *used goods, money, gossip, e-mail, attitudes, infection, and packages*. Data being sent using these flows are also affected by the flow processes. These processes include: if the data is actually moved or copied from one node to another, whether there is only one data package at a time or more than one, and whether the data takes the shortest path or not. Each combination of the type of flow and the influencing process is addressed with real-world examples and analysis. USW nodes may learn about nodes in their local area via a gossip protocol and will use an infection model when engaged in forwarding messages (Chapter 8 on page 275).

Leskovec et al. [118] explore the phenomena that over time graphs become more dense with edges being added super linearly with respect to the number of nodes. The result being that the average path length decreases as a function

time as well. In particular they examine the arXiv citation graph, the patents citation graph, the autonomous systems graph and the affiliation graph. In all cases, the diameter shrinks as a function of time. They propose different models to describe this behavior. The models are:

1. *Community guided attachment* where members of the community are “directed” that they should cite a certain number of already existent members of the community resulting in self-reinforcing behavior, and
2. *A forest fire model* based on the idea that some papers have an extra ordinary large number of out links (citations to others) that anyone accessing them, might also access a large number of other papers.

USW graphs demonstrate community guided attachment behavior based on Policy C.

White and Smyth [119] look towards finding communities in a graph by applying Laplacian techniques to the adjacency matrix of a graph. Using this approach they claim that they are able to find more accurate communities of subgraphs than could be found using hierarchical techniques. They tested their techniques with real-world data from WordNet, American college football teams, and clustering of authors publishing in Neural Information Processing Systems (NIPS). USW graphs have small-world properties, and the characteristics of $L(G)$ and $C(G)$ are more important. A community may develop based on Policy C, but it is not guaranteed.

- *2006*

Borgatti and Everett [120] provide a view of conceptualizing and measuring centrality values from a graph-theoretic perspective. They approach the problem by focusing on a node’s involvement in the walk structure of a graph. They measure this along four axis: type of nodal involvement, type of walk considered, property of walk assessed and choice of summary measurement. USW used many of their involvement ideas during the centrality measurement and evaluations when considering how to attack a graph (see Section 5.5 on page 148).

Newman [121] puts forth a technique for detecting community structures by examining the adjacency matrix for a graph and then the spectral partitioning

of that matrix. By recursively applying spectral partitioning techniques to the derivatives of the adjacency graph, the full network can be subdivided into modules/clusters in a fast and accurate manner.

Newman [122] provides an extensive treatment of the properties of many different types of graphs including random, Markov and small-world. For each type of graph, he discusses and presents equations defining clustering coefficients, expected degree distributions, mixing patterns, network resilience and degree correlation. Newman has brought into one place many divergent ideas and formulations. USW made use of his equations and approaches in the evaluation of its graphs.

- *2008*

Halim et al. [123] evaluate small-world networks as structured overlay networks (SON). They compare SON to small-world networks (SWN) and random networks and seek to explain how each responds to node and link failures. They conclude that the number of edges in any graph increases the robustness of the graph, that SON graphs have high maintenance costs as compared to SWN and that SWN can function as replacements for SONs. Their simulation and evaluation of SWNs leads to the conclusion that increasing the number of edges increases the robustness of the SWN matches our intuition.

Osvall and Bergstrom [124] use information theory to detect and describe clusters of connected nodes in a directed graph using random walks. They demonstrate their technique by performing citation analysis on 6,434,916 citations in 6,128 scientific journals and identify 88 modules/clusters. Using their techniques, they can design maps of graphs at an appropriate level for a particular visualization need. USW may apply their techniques in order to provide “maps” of the autonomously created graphs based on the content of each WO (Chapter 8 on page 275).

Zweig and Zimmermann [125] propose a protocol that enables the graph to change its overall degree distribution based on locally detected failures. Essentially, if a node detects a failure in one of its $N_i(v)$ neighbors, then it will create additional links to other members of its k neighborhood. If there is extensive damage to the graph because it is under attack, then there will be a dramatic shift in the total degree distribution of the graph. USW may use

these repair techniques if it detects that there is significant damage within the local k subgraph (Chapter 8 on page 275).

- *2010*

Karsai et al. [126] put forth the argument that the spreading dynamics of a small-world network are slow, even with the numerous short length paths that are characteristic of a small-world. They use empirical data and a susceptible-infected (SI) model to evaluate their proposition. A way to view their contribution from a USW perspective is to replace their “infection” with “message passing.” Their model and concepts dovetail nicely with (see Section 5.3 on page 121) and support the need for faster and more capable communication models.

3.2.5 DIRECTLY APPLICABLE

Callaway

USW graphs are dependent on interactions between WOs for the graph to grow through making new friendship links between WOs, thereby enabling more preservation copies to be made. Experimentation with constructing USW graphs has shown that this interaction should be regular and widespread. For instance, if a list of WOs were added to the system in a linear manner through one gateway, and that gateway were only accessed after all the WOs had been added to the system, then connections will be made to only the gateway and not to any other WOs. Once all the WOs had been added to the system and the list of WOs (including the gateway) were accessed a second time, then more connections would be made. In a sense, the order in which WOs are added to the system and most particularly when and how often the gateway is accessed could create the appearance of a percolation activity.

Cooper and Garcia-Molina

Cooper and Garcia-Molina [109] enumerate ideas they believe are key to a trading network. The USW algorithm fits very closely with some of their ideas (Table 5 on the next page).

Table 5. Comparing Cooper and Garcia-Molina concept of trading networks and the USW.

Concept	C&G-M expansion	USW Interpretation
An archive site	An autonomous provider of an archival storage service.	Each USW WO host is a potential autonomous provider of an archival storage service. Each host makes its own decisions based solely in its own criteria.
A digital collection	A set of related digital material that is managed by an archive site.	A USW WO is a single digital entity. A collection of these entities on a host can be considered a Cooper Garcia-Molina collection.
Archival storage	Storage systems used to store digital collections.	Each USW WO copy is a preservation copy.
Archiving clients	Users that deposit collections into the archive, and retrieve archived data.	The USW client facing software will support users adding to the WO collection.
Trading network	A local site must connect to remote sites and propose trades.	The USW algorithm does not support or address this need.
Automation	The archive should operate as automatically as possible, while allowing librarians or archivists to oversee its operation and adjust its configuration.	Each USW WO operates automatically and will manage preservation copies automatically.

Fiat and Saia

USW preserves web pages that the user decides are worth curation. It is totally possible that the web page is not under the user's control and therefore may be changed after the user has curated the page. This can happen based on the following events:

1. The user decides to add the web page to the USW graph,
2. Preservation copies of the web-page are made in accordance with the USW algorithm,
3. The web page's creator decides to change the content of the original web page.

The web page that the curator preserved is now different than the page visible under normal conditions. Solving the problem of resource synchronization is beyond the scope of this dissertation. The USW algorithm will ensure that the web page that was originally curated is preserved. This is akin to cache synchronization in a large multi-processor system.

Holme and Kim

USW WOs will have infinite capacity, so the likelihood of an avalanche of failures as described by Holme and Kim is 0. An alternative way to interpret their work is to interpret the number of geodesic paths as the number of paths a broadcast message would take from one WO to all WOs. Using this interpretation, then failure of the WO that most paths traverse could cause an avalanche of failures. We have implemented a Linda [65] communications to eliminate this possibility. Implementing Linda style communications also allows for more efficient WO to a WO group communication mechanism and allows messages to be sent to WOs that currently do not exist.

Link

Analysis of the probability of a USW graph being disconnected by the loss of a WO (and all the edges that are incident to the WO) has been taken to the limit during the resiliency and robustness analysis. During the analysis, the $Damage(G)$ that the graph would suffer when any of the WOs was deleted was computed. The WO that was resulted in the highest $Damage(G)$ was removed from the graph, and

the process was repeated until the graph was disconnected. This analysis is directly related to the analysis elsewhere (see Section 3.2.5 on the following page).

Manku

The USW algorithm requires that WOs exchange messages amongst themselves. Ideally this would be direct WO-to-WO communications, but current WI limitations prevent this type of communication. Generally, a server of some sort is required to access a WO, and that assumes that the URI of the intended recipient is known when the sender needs to send the message. Because USW communication is asynchronous, and because the intended recipients may not be known when the message is sent, and because the same message may be intended for multiple recipients, a different messaging scheme is needed. The USW reference implementation uses a Linda [65] communications model.

Moreno

USW are small-world graphs composed of WOs that are assumed to have infinite capacity and therefore should be immune to avalanche types of failures. But if the capacity were characterized as the inability for USW nodes to contact their friends then this could result in a bottle neck preventing WO-to-WO communication and could be treated as a form of avalanche behavior. WO-to-WO communication is central to the USW algorithm, a store-and-forward communications model would allow for the possibility of an avalanche type of failure. To overcome this type of design limitation, a tuple based style of communications based on Linda [65] was implemented (see Section 5.3 on page 121).

Najjar and Gaudiot

Analysis of the probability of a USW graph being disconnected by the loss of a WO (and all the edges that are incident to the WO) has been taken to the limit during the resiliency and robustness analysis. During the analysis, the $Damage(G)$ that the graph would suffer when any of the WOs was deleted was computed. The WO that was resulted in the highest $Damage(G)$ was removed from the graph, and the process was repeated until the graph was disconnected.

3.3 DIGITAL LIBRARIES AND WEB PRESERVATION

A digital library is a library whose collections are stored electronically. The collection can be specific to a particular topic, or unclassified and free form. The USW algorithm takes some of the functions of a digital library geared towards preservation of web pages and exposes them to crowd-sourcing. Digital libraries have evolved from custom applications tailored for one institution or environment, to general purpose systems with extensive functionality. The USW algorithm supports and implements many of the standard digital library functions and exposes some of those functions to curators outside of the library.

- 1989

Bearman [127] raises a series of questions and issues about the role of archivists in culture. He postulates that their current perceived role as keeper of the past in order to serve some ill-defined greater good at an unknown time in the future is incorrect. Bearman feels that archivists are keepers of very incomplete fragments of the past and that usually these fragments are without context and that the context is vital to truly understanding the past. Based on the rate of production that might be of archival interest and the rate of accession by archivists, the volume of material that should be archived will overwhelm any archiving effort. Bearman believes that the very limited number of people that use archives is the result of a combination of the archivist's view that they are keepers of the past and therefore not overly concerned about the present and that archives are not constructed in a way that encourages the general public from accessing the information and data in the archives. Part of his summary is:

“Archivists cannot adequately describe what they currently hold or will acquire if they continue to employ current methods based on examination of their holdings, even if they rely on only the highest level of top down description. To increase the effectiveness of description and control by the necessary order of magnitude, archivists will need to identify that information which can be obtained from outside, and import it into their systems automatically. This information will provide access by provenance, based on the nature of the activities documented, and by the structure of information systems. They will also need to design systems that capture administrative

data throughout the life-cycle of records, both before and after they come into archival custody, and use that information in the on-going control and management of the records. . . . In redirecting ourselves to this effort, we need to revisit our rhetoric, removing the unconvincing references to our role in preserving evidence for posterity, and replacing them with our role in focussing and connecting the past and the present. Instead of envisioning ourselves as victims of an information explosion, we need to emphasize a vision of archives, libraries and museums joining to bring about an information implosion.”

- 1995

Arms [128] puts forth eight key concepts in the architecture of digital libraries. They are:

1. *The technical framework exists within a legal and social framework*
2. *Understanding of digital library concepts is hampered by terminology*
3. *The underlying architecture should be separate from the content stored in the library*
4. *Names and identifiers are the basic building block for the digital library*
5. *Digital library objects are more than collections of bits*
6. *The digital library object that is used is different from the stored object*
7. *Repositories must look after the information they hold*
8. *Users want intellectual works, not digital objects*

The USW algorithm address concepts 3, 5, and 6 (see Section 3.3.2 on page 77).

Lesk [129] provides a survey of most of the common and recurring problems posed when preserving digital objects. These include:

1. *Media Problems*: not only the lifetime of different media, but the ability to read and access the data on the media.
2. *Format Problems*: once data on media is accessed, it must then be interpreted. The wide variety of different formats, both open source and

proprietary, could lead to the creation of a new profession: *digital paleographer*.

3. *Selection*: the almost exponential explosion in the amount of digital data and the limited number of professional digital archivists means that not all data will, or can be archived and therefore a selection process must be created.
4. *Cooperation*: because of the volume of data that needs to be preserved, repositories may decide to “divide” up the work and focus on specific areas and then provide the results of their efforts to their partners.
5. *Fairness*: if repositories are to cooperate, then each must feel that they are being expected to contribute a reasonable amount of effort based on their available resources and expected return from other repositories.
6. *Legalities*: often digital data is encumbered with licenses and fees. These legal restrictions will have to be addressed and solved.

USW’s implementation addresses the media and formation problems (Appendix A on page 287). Because USW WOs are autonomous entities; the selection, cooperation and fairness concerns should be controlled by access to the WOs. USW does not address the legalities issue.

- *1996*

Waters and Garrett [60] detail the findings of Task force on Digital Archiving. The Task Force was created by the Commission on Preservation and Access and the Research Libraries Group to investigate the means of ensuring “continued access indefinitely into the future of records stored in digital electronic form.” The report identifies specific examples of how digital data was lost due to accident, changes in technology, loss of explanatory information (metadata) and loss of organizations. They detail the requirements of a “digital archivist” to include: appraisal and selection, accession, storage management, access and systems engineering. Some of the mitigation strategies that they proposed are to fend off the likelihood of data loss and to support the digital archivist include: changing media, changing format, incorporation of standards, building migration paths, and the use of processing centers. Many of the threats and mitigation strategies from this report have been echoed by many others since

the report was delivered. USW addresses a few of these areas (to wit, changing media and format), while others require organizational intervention and support and are outside of the USW arena.

- *1997*

Daniel and Lagoze [130] extend the Warwick Framework (WF) to manage a possible unrestricted growth in complexity. The WF originated from an attempt, at the Second Invitational Metadata Workshop, to define an extension mechanism for the Dublin Core Metadata Element Set. Named after the site of the workshop in Warwick, U.K., the WF tackles the extension problem by aggregating typed metadata packages into containers. The extensions by the authors include having a catalog of internal components inside a container, resulting in a self describing package. This catalog can be used to specify the relationship between the container's internal and external components. These relationships are called Distributed Active Relationships (DARs). External applications can take advantage of these DARs and automatically process the data in the container. USW incorporates many of the DAR ideas including no essential distinction between data and metadata, multiple relationships between the containing WO and other WOs, and the location of other resources are independent of the current WO.

- *1998*

Goldberg and Yianilos [131] present an idea that would use the Internet as long term, distributed, archival storage. This Internet based memory system is called Intermemory. In their model, a user donates a certain amount of storage S for a finite period of time and then is permitted to utilize an amount of storage s ($s \ll S$). Data would be dispersed across all the Intermemory subscribers and the data would continue to migrate as time goes on. Because of this dispersal, removal or censoring of data in the Intermemory would get increasingly difficult and the likelihood of total removal is vanishingly small. USW will provide the same level of protection from censorship by being a crowd-based opportunistic protocol that would make locating and modifying all copies of a WO extremely difficult. The USW algorithm does not provide, nor preclude access controls. Access control may be layered on by USW hosts. Rothenberg [2] starts with a hypothetical letter to his grandchildren saying, if

you can read this CD-ROM, you will find the key to the family fortune. His grandchildren have only seen CDs in old movies and are now faced with the problem of finding a mechanism to read the CD, software to interpret the data on the CD and then what to do with that knowledge. Rothenberg reviews the several significant data losses and postulates the idea that digital data never dies, it just becomes irretrievable. This loss may be due to the inability to mechanically handle the media, or that the software needed to correctly interpret the bit stream is unavailable. Rothenberg puts forth the argument that a data file can be considered a program that contains instructions (format instructions, image placement, etc.) and data (text, image, metadata, etc.) and therefore anything that can properly interpret the program instructions can be used to emulate the original program. This idea of emulation can be extended to the hardware as well. Rothenberg states that digital data will last forever, or five years whichever comes first. Rothenberg advocates for the use of well known standards to provide a method of bootstrapping future programs to be able to interpret non-standard bit streams, of using these standards to document the hardware and software needed to define emulators and that digital data (along with its attendant digital explanations) be migrated on a regular and frequent basis. USW uses standards based techniques [132, 133, 134, 135] to help ensure the USW data.

- *2000*

Waugh et al. [136] provide a brief overview and explanation of the Victorian Electronic Record Strategy (VERS) that is being tested within the Victorian government to address the difficulties of digital preservation. VERS distinguishes itself from other preservation systems, in that it is in active use by government agencies as opposed to theoretical systems. VERS relies on data driven encapsulation. The key elements of the encapsulation are:

- *Simple and Self Documenting* — the encapsulation must be capable of being read and understood by a human using the simplest computer tools.
- *Self Sufficient* — the encapsulation must include all the information required to preserve the digital information.
- *Content Documentation* — the encapsulation must contain sufficient documentation to enable a future user to find or write software to access the

preserved information.

- *Organizational Preservation* — the encapsulation must support the inclusion of information that addresses the organizational issues involved in continued use of the preserved information.

USW meets all the VERS key elements, except *organizational preservation* because USW WOs are autonomous entities.

- *2001*

Nelson and Maly [137] put forth the position that information in complex digital objects are “first class citizens” and that decoupling these citizens from the confines of the digital libraries that contain them will result in richer digital library (DL) experiences for users. They introduce the idea of “Buckets” as aggregative, intelligent, object-oriented constructs for publishing in digital libraries. Buckets are an extension of the “Smart Object, Dumb Archive” model. Bucket design goals are: aggregation, intelligence, self-sufficiency, mobility, heterogeneity and archive independence. USW is a continuation of the ideas that from the basis of “Buckets” and focuses on imbuing the USW digital objects with the intelligence to autonomously create small-world graphs, to migrate from the host where the WO currently lives to another host that has more space available and WOs that can be heterogeneously across the graph.

Reich and Rosenthal [53] provide an overview of the Lots of Copies Keeps Stuff Safe (LOCKSS) system. LOCKSS is a peer-to-peer network of minimal hardware and software computer installations that provide long term caching services to libraries and archives. When a user at one of the LOCKSS institutions requests a web page, the request is forwarded to the originating publisher. If the request fails, it is then serviced by the local institution. The LOCKSS installations at each institution monitor the health of each other’s pages and maintain all caches via a reputation based mechanism that is designed to operate slowly and to be very difficult to corrupt or subvert. USW incorporates the ideas of multiple copies to ensure long term availability and the implicit idea that copies should be distributed across multiple hosts to reduce the likelihood that accident or attack will compromise all copies.

Thibodeau [58] focuses on digital preservation as it pertains to the National Archives and Records Administration (NARA). An important distinction is

made between preserving a file (say a map) and the context of how that map was used (say in planning the bombardment of a specific target). NARA is chartered and directed to be responsible for the life-cycle maintenance of the records for all three branches of the Government until the end of the Republic. The author proposes an Electronic Records Archive (ERA) based on XML standards for addressing what is considered a moving preservation target that is growing both quantitatively and in complexity and in directions that are not entirely predictable. The author states that NARA is making and maintaining partnerships with other government agencies, private businesses and academia to address these challenges. USW uses standards based techniques [132, 133, 134, 135] to facilitate and support its preservation needs and to be able to move to newer technologies as they become available.

- *2002*

Nelson and Allen [138] report on the long monitoring of 1,000 digital library (DL) objects. Twenty World Wide Web accessible DLs were chosen and from each DL, 50 objects were chosen at random. The DLs were checked three times a week for just over 1 year for a total of 161 data samples. During this time span, they found 31 objects (3% of the total) that appear to no longer be available. During their testing, they did not make any statement as to the “correctness” or “completeness” of the objects other than to compare their size. If the 3% per year loss is a reasonable presentation of the expected object loss at a DL, then 50% of the original objects will not be available after 22 years. The USW algorithm is created to provide a data-centric long term method to preserve data and therefore should not be subject to the same kinds of losses that repository-centric approaches. As the USW algorithm operates, it monitors the status of each of its preservation copies and works to ensure that enough copies are present so that the rate of loss per year will not exceed the rate of preservation copies created.

- *2003*

Kephart and Chess [139] provide a high level vision of autonomic computing. They put forth the position that as systems become more interconnected and diverse, architects will become less able to anticipate and design interactions to cover all eventualities. Because of this, the software will have to be able to

adapt to and overcome these unforeseen conditions. Autonomic systems will have four self management aspects. These aspects are:

1. *Configuration*: the ability to follow high level policies,
2. *Healing*: the ability to detect, diagnose and repair system failures,
3. *Optimization*: the continuous process of improvement and enhancing performance, and
4. *Protection*: defend against malicious attacks or cascading failures

USW WOs implement all of these aspects.

Lipscomb [140] discusses how demographic factors are affecting the number of active librarians. The factors are the average age of the current professional librarian is increasing, in step with the general population and that soon a majority of the librarians will be eligible to retire. Second, there are fewer new librarians entering the profession to fill the ranks of those that leave the profession. Lipscomb predicts that ongoing recruiting efforts may have negative impacts on the profession, even as the need for more professionals increases. USW could be of use to these professionals by reducing the amount of manual data maintenance that is required once a WO is curated. Additionally, an implemented USW will make it easier to curate WOs.

Markwell and Brooks [141] examine the expected life of links to education related sites for 6 years starting in 2000. Based on the monthly sampling of 515 URLs, they estimate that the “link rot” follows the equation $E = E_0 * e^{-0.013 * m}$ where E_0 is the original links used in a course curriculum, and m is the number of months since all the links were established and E is the number of links that can be expected to still be viable. They estimate that the half-life of their education related links to be 60 months. The USW algorithm is created to provide a data-centric long term method to preserve data and therefore should not be subject to the same kinds of losses that repository-centric approaches. As the USW algorithm operates, it monitors the status of each of its preservation copies and works to ensure that enough copies are present so that the rate of loss per year will not exceed the rate of preservation copies created.

Paskin [142] examines the question “What is a copy?” Specifically, the copy term is used in the generic sense of an imitation or reproduction of an original.

Paskin concludes that two digital entities are never the same in any absolute sense and can be considered copies of each other only in the context of some defined purpose. It is the definition of this purpose that can place some restrictions on the copy's use and availability. Use and restriction are at the heart of the Digital Rights Management (DRM). The difficulty of handling DRM issues with respect to long term preservation of digital data is that the data will become increasingly encumbered with strings that identify how the copy can be used within a DRM environment. USW uses a loose definition of "copy." When a USW DO is created, it will not be a bit for bit copy of the original object that is to be preserved, but instead contains a subset of all the object's components. By preserving only a subset of the object, then DRM issues should be avoided.

- *2004*

Hunter and Choudhury [143] put forth the idea that comparing the metadata of digital objects in a central repository and be useful in detecting, or predicting an object's obsolescence. Based on this detection, a message could be sent to a relevant agent and obsolescence could be avoided. By making this agents available using the machine-process-able ontology web language service (OWL-S) these agents could be discovered automatically. They call this system Preservation service Architecture for New Media and Interactive Collections (PANIC). Their approach is very different from our USW, in that the USW model relies on peer-to-peer communication to alert members about agents and does not rely on a central repository.

- *2005*

Anderson et al. [144] document their experiences at Stanford University when they participated in the Archive Ingest and Handling Test (AIHT). Stanford focused on the assessment process. They quickly realized that a purely human-mediated interrogative process or tool would not scale and that some sort of automated mechanism was required. Based on their experience working with more than 55,000 files, they feel that an automated assessment process is clearly the only efficient means to collect technical information about large numbers of files. Additionally, they feel that cooperative experiments like the AIHT are the best way to test how a system will work with "live data." A fully

implemented USW will aid the assessment process by involving the user in the curation process (Chapter 7 on page 253) and by automating the conversion from one data format to another (Chapter 8 on page 275).

Shirky [145] presents an overview of the Archive Ingest and Handling Test (AIHT). The AIHT was a project from the National Digital Information Infrastructure and Preservation Program to better understand which aspects of digital preservation are institution-specific, and which are more general and applicable across institutions. AIHT initially focused on, what they thought would be a trivial problem, the transfer of 60,000 digital records from George Mason University (GMU) to the Library of Congress. This “trivial” exercise turned out to be fraught with problems: file identifiers that are not, requirements that are not, curation triage that will not, and “small” errors that cause the system to fail when taken to scale. These types of problems are also possible when the digital records are exported from one system to another. There are two conclusions from the paper: (1) Data-centric is better than tool-centric or process-centric at large scale, and (2) Preservation is an outcome. The final conclusion from the AIHT is that there is a pressing need for continual comparative testing preservation tools and technologies. USW fits squarely into the idea that data should have the “desire” and the tools to preserve itself.

Baker et al. [61] explore the challenges associated with the preservation of digital data on various hardware alternatives in the face of hardware/software failures, environmental threats, accidental erasure by humans, and changes in institutional policies and directions. Their solution to mitigate against these threats is to replicate data across autonomous sites, minimize the per-site engineering costs, and design for long term scalability. In many ways they are promoting the philosophy of the Lots of Copies Keeps Stuff Safe (LOCKSS) system [53, 146]. USW follows these design philosophies by operating at high level thereby maintaining a level of hardware and software independence, by keeping copies across multiple independent sites to mitigate against human and institutional errors, and by designing each USW WO to act autonomously so that the system will scale.

McCown et al. [147] explored the persistence of URLs cited in articles published in D-Lib Magazine. They selected 452 articles that had a total of 4,387 unique URL references. These URLs were sampled three times a week for 6

months. Approximately 28% URLs failed to resolve at the start of the monitoring. This increased to approximately 38% by the end of the monitoring period. Based on this data, they estimated that the half-life of a URL referenced in a D-Lib Magazine article is approximately 10 years. Additionally, they found that references URLs in the .net, .edu or country-specific top-level domains were lost faster. The USW algorithm is created to provide a data-centric long term method to preserve data and therefore should not be subject to the same kinds of losses that repository-centric approaches. As the USW algorithm operates, it monitors the status of each of its preservation copies and works to ensure that enough copies are present so that the rate of loss per year will not exceed the rate of preservation copies created.

Nelson et al. [148] detail their experience at Old Dominion University (ODU) as participants in the Archive Ingest and Handling Test (AIHT). The AIHT was a project from the National Digital Information Infrastructure and Preservation Program to better understand which aspects of digital preservation are institution-specific, and which are more general and applicable across institutions. ODU was the only member of the test that was a non-library to participate in the test and therefore focused on alternative archiving techniques, including:

- *Self-archiving Object*
- *Archive Models and Granularity*
- *Archive Ingest*
- *Format Conversion*

Nelson et al. encoded the content to be archived as an MPEG-21 DIDL complex objects with web access to the content. Their archival data model used internal identifiers in place of the original identifiers and did not delete the original files when they are migrated to new formats. Each individual file to be archived is mapped to an MPEG-21 DIDL “Component” element. They believed that metadata generation tools will continue to evolve, so their philosophy was to be able to process the original file through an arbitrary number of introspective programs and store their respective output in separate MPEG-21 DIDL “Descriptor” elements. Throughout the project, they took a decidedly “data-centric” view of archiving. USW has made use of many of

the ideas and lessons learned from ODU's experience with the AIHT by using REsource Maps [133] to document the current state of the WO and therefore laying the framework for keeping a series of data transformations as the WO ages (Chapter 8 on page 275).

Rosenthal et al. [149] layout the minimum requirements for a digital preservation system in sufficient detail to obtain a certification as a ISO-9000-like process. To approach the problem of assuring the long term availability of digital data by identifying and proposing solutions and processes to single points of failure, automatic upgrades to media, hardware and software as the current items become obsolete; bearing in mind that the data will be accessed infrequently. Because of the infrequent access, the data must be audited on a periodic basis. All of these problems and issues are addressed from a "bottom-up" approach, focusing on what the system should not do. They bring forth the idea that the failure of a digital preservation system will be determined in a finite time, while its success will be forever unproven.

Shirky [150] presents the final results of the Library of Congress' Archive Ingest and Handling Test (AIHT). The test covered three main areas: ingest and markup of a digital archive; export and sharing of that same digital archive from the tested preservation regime; and conversion of digital objects from one format to another. Based on the results of the efforts of the five different digital repositories (George Mason University, Harvard University, John Hopkins University, Old Dominion University, and Stanford University), three areas of continued investigation and effort were identified. The areas are: validation of shared effort; a need for regular, comparative testing; and work to reduce the cost of sharing the preservation burden. USW can be actively used to support reducing the cost of preservation by automating the conversion of digital formats and migrating digital data from one environment to another.

- *2006*

Moore et al. [151] examine the requirements for digital repository audit checksums as part of the assessment of organizational infrastructure, repository functions, community use and technical infrastructure. These assessments are expressed as a series of rules that can be mapped to various potential failures of an organization. The failures include technology failure, technology evolution,

collection evolution and organizational failures. A large number of checksum related commands are identified that are applicable to these failures. USW could incorporate some sort of checksumming or MD5 hash of the digital data to increase the confidence that the data has not been corrupted (Chapter 8 on page 275).

- *2008*

Allison et al. [152] put forth Simple Web-service Offering Repository Deposit (SWORD) as a lightweight protocol for repository deposit. It was developed over short time span during mid 2007 with a very ambitious schedule to provide capability to support a series of use cases. The use cases include an author using a desktop application to submit a file to a mediated deposit service, computer or experiment generated output stored as a “save as” type of function, simultaneous depositing of the same digital data to multiple repositories, and inter-repository digital exchanges. SWORD met all these requirements in a very short period of time. USW WOs could make use of the ideas of using standard and existing repository interfaces in order to preserve copies (Chapter 8 on page 275).

Lagoze et al. [153] recasts the repository-centric notion of digital object preservation to that of a digital object being a bounded aggregation of Web resources. USW will use many of the ideas from this paper that deal with the mechanics of preserving data. In particular, USW will use Object Re-Use & Exchange (ORE) resource maps (REMs) to manage the data associated with the USW WOs. USW will use the ORE idea of links to frame, design, and track the resources that constitute the original digital object that was preserved.

- *2009*

McCown et al. [28] surveyed 52 individuals that had lost their web sites through a number of hardware, software and human failures. Using a collection of web repositories (the Internet Archive, Google, Live Search, and Yahoo) collectively known as the web infrastructure (WI) and the software package Warrick, portions of many of these lost sites were able to be recovered. Based on these experiences and the belief that the reasons that the original sites were lost would continue to occur, the idea of lazy preservation is proposed. Lazy preservation uses the idea that there are web crawling repositories routinely

caching copies of the visible web in their local storage and that these repository stores could be used to recreate lost sites. Because this is an ad-hoc operation by the crawlers, there is no guarantee that a complete site will be crawled, nor that all site components will be available in one repository. Lazy preservation and site reconstruction using the Warrick software cannot replace dedicated preservation activities, but preservation will occur at virtually no cost to the site being preserved. USW takes the idea of lazy preservation and extends it to be crowd-based preservation. When a user decides that a site should be preserved, it will be initially entered into the USW graph and then actively, automatically, and completely preserved there after.

McCown et al. [54] looks at spreading the curation process across all Internet users versus relying on the activities of a few selected curators associated with digital repositories. McCown proposes client side software called ReMember which uses Object Reuse and Exchange (ORE) Resource Maps (ReMs) for describing aggregations of web resources. These resources are then stored at different web infrastructure (WI) accessible locations so that they can be retrieved at a later time. USW uses ReMs to record the location of various WO components and the idea of using the WI to store preservation copies.

- *2011*

Wojcik et al. [154] provide a perspective on distributed digital curations from the Persistent Archives Testbed (PAT) project and the Archive Ingest and Handling Test (AIHT). A number of fundamental difficulties that arose across the different archives are identified, including the differences between file names and the effect on the way different software packages (and operating systems) restrict and constrain names. Each digital library tested their sites with one or more archival functions (appraisal, accessioning, arrangement, description, preservation, or access). The sites were able to exchange data using a common software infrastructure while maintaining each site's independence. Based on the lessons learned, their conclusion is that long term sustainability is probably beyond the capability of most individual archival repositories, and that the expertise required to stay abreast of changes in technology may be too overwhelming. Because of these issues, a distributed multi-repository partnership or consortia might be the best alternative. USW is designed to be repository independent and to be able to migrate the data load of each WO

as needs arise. USW directly addresses many of the issues raised in [154].

Digital preservation deals with ensuring that digital data is available for as long as necessary. USW supports key aspects of digital preservation of data refreshing and data migration by continually making preservation copies on different WO hosts.

3.3.1 DIGITAL REPOSITORIES

Digital repositories are the electronic “bookshelves” of a digital library. The USW graph serves as the “bookshelf” where the USW WOs are the “books” in the digital library.

- 1990

The Committee on Government Operations [155] takes the National Archives and Records Administration (NARA) to task about some of the areas where the NARA had not done as well as they were expected. Included in the report are details about how the National Military Command Center Information Processing system (NIPS) files were lost because the application that read the files lived on an IBM mainframe that was no longer manufactured and therefore the Agent Orange Task force was not able to accurately report the effects on Viet Nam era veterans of Agent Orange. Other problems that were enumerated include; loss of census data because files were compressed in an ad hoc and non-documented way, how United States Railway Authority Case Tracking/Document Management System data was lost because the latest version of the database software could not read the previous version’s data, how data was expected to be lost shortly because it was punched into Hollerith cards, and how testimony and diary summaries from the Watergate affair requires a specific version of software that was no longer available. An unidentified archivist is quoted as saying,

“The problem with preservation of electronic media is that the media can easily be preserved longer than the capability of reading the signals recorded on them. Magnetic and optical media for recording sound, image and data are all subject to market forces and technological change which are occurring at a rate that requires us to continuously recopy media to newer media to newer physical and logical formats in order to preserve access.”

- *2003*

Lynch [156] speaks to the need for institutional repositories. In his view, an institutional repository is a set of services that an organization (such as a university) offers to the members of its community for the management and dissemination of digital material created by the institution and its members. Key among the services provided by the repository is the management of technological changes, and the migration of digit content from old technology to new. Lynch argues that an institutional repository can address near-term problems of making institutional data available while the author is part of the institution, and will continue to do so after the author has left. The institution at its core acting like long term corporate memory. Unlike long term memory, an institutional repository could fail for many reasons, including changes in policy, management failure or incompetence, or technical problems. USW WOs augment the preservation capabilities of an institutional repository and lessen the impact of change in policy or technical problems.

- *2005*

Heery and Andersen [157] report on a review of UK digital repositories. They enumerate the primary functions of a digital repository for supporting current research and the need for the repository to support as yet unidentified research needs. Heery and Andersen identify repositories based on their content type:

1. Raw research data
2. Derived research data
3. Full text pre-print scholarly papers
4. Full text peer-reviewed final drafts of journal/conference proceedings papers e-theses
5. Full text original publications (institutional or departmental technical reports)
6. Learning objects
7. Corporate records (staff and student records, licenses, etc.)

and by the coverage of these types:

1. Personal (authors personal archive)
2. Journal (output of a single journal or group of journals)
3. Departmental
4. Institutional
5. Inter-institutional (regional)
6. National
7. International

and by target user group:

1. Learners
2. Teachers
3. Researchers

USW supports their digital repository tenets by ensuring that digital data is preserved for future and unknown uses.

- *2006*

Heery and Powel [158] examine the state of digital repositories in the UK in 2006 and lay out areas that need attention to meet the 2010 vision in which a high percentage of scholarly work in the UK will be widely available. They identify areas that need attention and development, in the areas of policy, culture, and working practices. For different media types (academic papers, geospatial data, learning materials, and raw data), they spell out the 2010 vision, what is the state of the art now, and how to get from where they are to where they need to be. Integral to their vision is that repository holdings be open and accessible to queries from outside the repository using well known and established protocols (such as OAI Protocol for Metadata Harvesting). USW process is complementary with their stated visions. USW digital objects could be held in a digital repository and be made available via OAI techniques and procedures.

3.3.2 DIRECTLY APPLICABLE

Arms

The USW algorithm represents a change from archive and repository centric preservation to data-centric preservation. As such, some of the concepts enumerated by Arms in [128] are applicable while others are not. The concepts are:

1. The technical framework exists within a legal and social framework.
2. Understanding of digital library concepts is hampered by terminology.
3. The underlying architecture should be separate from the content stored in the library.

Within the USW environment, a copy request with data is sent to a friend WO, who in turn passes the data onto its copy service. The service has complete liberty as to how the data is stored on its host, and where it is located. The location returned by the copy service will be used by the host to retrieve a representation of the original data, but how it is stored may be totally independent of where it is stored.

4. Names and identifiers are the basic building block for the digital library.
5. Digital library objects are more than collections of bits.

The USW algorithm ensures that preservation copies of data are made across distinct hosts. Candidate preservation hosts are identified by the friendship connections that WOs make based on their activity in the USW graph. These connections can be considered as metadata about the data being preserved. The USW algorithm ensures that the metadata and the actual data are preserved, the data without the metadata is just a collection of bits without form or organization.

6. The digital library object that is used is different from the stored object.

The way that a USW WO is presented to the user, may be totally different than the way the WO is stored internally.

7. Repositories must look after the information they hold.

USW removes the requirement for a repository to look after the information they hold. The removal of this requirement opens up a world of preservation possibilities.

8. Users want intellectual works, not digital objects.

3.4 SUMMARY

We have identified those authors and works germane to the Unsupervised Small-World framework. We build the USW theory, simulation, and reference implementation on these related works.

“We are like dwarfs standing on the shoulders of giants and so are able to see more and see farther than the ancients.”

Bernard of Chartres, 12th-century philosopher [159]

CHAPTER 4

INTRODUCTION TO UNSUPERVISED SMALL-WORLD (USW)

4.1 INTRODUCTION TO JOSIE MCCLURE

We have chosen the photograph of Josie McClure to juxtapose the problems of preserving data in an analog age (from the early 1900s) with the growing problem of preserving data in a digital age (post 2000). We have enumerated the things that have, and have not to occur to data in the analog age for the data to survive. As well as the preservation things that must occur in the digital age.

Josie must transition from the analog to digital age to survive for the next hundred years. Her image must be scanned into a digital format and put on the Web to become a Web Object (WO). That WO will then become a part of the USW graph by executing the USW algorithm.

4.2 TENETS OF THE USW ALGORITHM

The USW algorithm is intended to provide a method for WOs to outlive the people and the institutions that create them. To achieve that goal, the algorithm:

- *Does not depend on global knowledge*: meaning that there is
 - No omnipotent enforcer of the actions that a WO will take, and
 - No omnipresent monitor of the state of the USW graph ensuring that each WO remains healthy and vital.
- *Opportunistic preservation*: meaning that a WO will engage in its USW graph and preservation activities when accessed by an outside entity. This implies that it may be a very long time between accesses and therefore a very long time between preservation activities.

- *Self-describing Web Objects*: meaning that all data necessary to preserve the WO and to perform USW activities are in the WO itself. The WO does not require, nor is it dependent on outside entities for interpretation and understanding.

Each of these tenets will be expanded upon in the following sections.

4.3 EXPECTED CONTRIBUTIONS

We make the contributions to Computer Science in the following areas:

- *Graph theory*,
- *Preservation*, and
- *Emergent behavior*

as supported by theoretical analysis, verified by simulation, and a reference implementation (Figure 12 on page 30).

4.3.1 THEORETICAL

We make the contributions to Computer Science theory in the following areas:

- Quantify the damage to a graph caused by the loss of a vertex or node,
- Identification, quantification, and qualification of the different ways that an attacker can damage a graph by the removal of edges or vertices,
- Quantify the amount of repair opportunities that the graph must have in order to repair itself after it has been damaged by an attacker,
- Incrementally create a small-world graph by the addition of nodes using only locally gleaned information.

4.3.2 SIMULATION

Two different simulators are used to verify and validate the theories we have developed. The first simulator is an in-memory and messages driven application used to create and analyze graphs of 1,000 vertices or more. The in-memory simulator allows graphs of these sizes to be created and analyzed in a fairly short time frame.

The second simulator is a robot that drives the reference implementation. The reference implementation is performance constrained to about 1 message per second, so graphs of a few hundred become extremely time consuming to create and evaluate.

4.3.3 REFERENCE IMPLEMENTATION

We created a reference implementation was created to demonstrate and validate theories and algorithms in a setting that more closely approximates the Web than a pure simulation. We have collected all the algorithms (Appendix A on page 287) and messages (Appendix B on page 311) needed to create the reference implementation in the appendices.

4.4 THE USW CREATE, ATTACK, AND REPAIR LIFE CYCLE

The life cycle of a preservation graph can be viewed as:

- *Create*: where a node (vertex) is added to an already existing graph. In the limit, the USW algorithm supports the creation of the graph when there are 0 nodes.
- *Attack*: where the graph is attacked by an enemy.
- *Repair*: when the graph is allowed to repair itself.
- *Repeated attacks and repairs*: when the attacker and the graph “play a game” by alternating attack and repair turns

These life cycle stages can be drawn as a “state diagram” (Figure 11 on page 27), and this diagram will be used as a “navigation” tool throughout this dissertation.

CHAPTER 5

THEORY

5.1 INTRODUCTION

USW is at the convergence of many different and significant computer science disciplines. These include:

- *Emergent behavior*: movement of the inanimate, Reynolds' ideas and concepts for "boids,"
- *Graph theory*: the unattended construction of small-world graph structures that mimic those found in nature and in some man-made social organizations, and
- *Preservation*: the long-term preservation of digital data focusing on web pages.

To provide the context of understanding the contributions of this research, we first briefly review the status of how objects are stored in repositories as well as the nature and types of various networks or graphs.

Each of the preservation approaches listed above (see Section 2.3 on page 20) inherently relies on human and institution intervention in the digital preservation activities of refreshing and migration [60, 57]. Digital preservation activities of emulation and metadata attachment are outside our context in this dissertation. Over time humans die and their personal archives can become lost, institutions may lose funding or have a change in ownership and therefore be unable to continue their preservation activities. As the amount of digital data continues to grow (at potentially an exponential rate), the organizational and human cost to keep up with traditional approaches can become overwhelming. An alternative approach is to revisit the definition of a WO and to incorporate into that definition the idea that the WO is empowered to make replication copies of itself for the purposes of preservation.

5.2 GRAPH THEORY

Our approach for the construction of a small-world network of WOs for self preservation is different than what others have used or proposed. We make use of the definition of a small-world graph as one that has a high *clustering coefficient* when compared to a randomly created graph and an average path length that is proportional to the number of nodes in the graph [43]. The Watts-Strogatz approach to constructing such a graph is to take a lattice graph of degree k and order n and perturb the links to create a graph with small-world characteristics. Some approaches make connections between nodes based on the proportion of the destination node's degree count [160, 161, 162], a kind of preferential attachment or fitness policy. Yet another type of approach takes an existing graph and then grows a small-world by the addition of new links [163, 95]. Or, by connecting a node to a fixed number of vertices based on their degree [164], or even creating a small-world graph from a random one [165]. Newman in [97] provides a survey of small-world graph construction techniques. Our USW approach can use preferential attachment to select the first node when adding a new node to an existing graph. But after the first WO selection, the USW algorithm controls where the WO fits into the graph and how many edges are created to other WOs in the system. USW is the only small-world graph creation algorithm that we know of where connections are made between WOs based only on information that the WO gleans prior to making its first connection.

5.2.1 ROBUSTNESS THEORETIC

In discussions about the graphs created by the USW algorithm, questions about their robustness and resiliency need to be addressed. We will compare various robustness and resiliency of the USW graph to graphs constructed by more classical techniques, such as Barabási's preferential attachment, Erdős-Rényi's random and Watts – Strogatz small-world. Each graph type is evaluated by counting the number of edges that an omnipotent attacker could remove before the graph becomes disconnected. Within this chapter, disconnection is defined as there being one or more nodes that are unreachable from any other node in the graph. In the trivial case; the removal of a node with a degree of one from a graph of infinite size means that the graph is disconnected. Other definitions are possible, but were not evaluated. The chapter concludes with how the work represented here can be expanded and enhanced.

Discussion

In much of the literature, the terms *resiliency* and *robustness* are used almost interchangeably. For this dissertation they have very different meanings. Their meanings are:

Resilience: the power or ability to return to the original form after being bent, damaged, or attacked.

Robustness: strongly or stoutly built. The ability of a system to remain functioning under a range of conditions (Appendix D on page 383).

Within this section, we are interested in the robustness of a graph, i.e. how much damage can a graph sustain and still remain connected? Resilience, by its definition, implies a period of recovery after some sort of damage. This in turn implies some sort of game where in one turn a totally knowledgeable attacker wreaks some sort of damage to the graph and in the next turn the graph is allowed to repair itself. These two steps alternate until the graph becomes disconnected (meaning that the attacker has won), or the graph degree distribution and average path lengths oscillate between some set of values (meaning the graph has won).

Test Environment

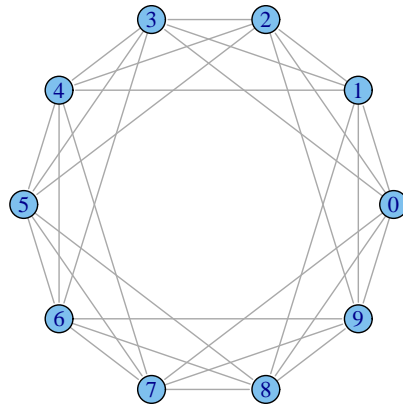
The R package *igraph* [78] was used to generate representative Albert – Barabási, Erdős-Rényi, and Watts – Strogatz graphs. The package was chosen because it: 1) has library calls to generate each graph type thereby speeding up the generation phase of the analysis, and 2) uses a widely accepted implementation. For each graph two types of data were collected and analyzed. The data are the degree distribution and an iterative graph disconnection process to quantify the particular graph’s robustness.

Results and Evaluation

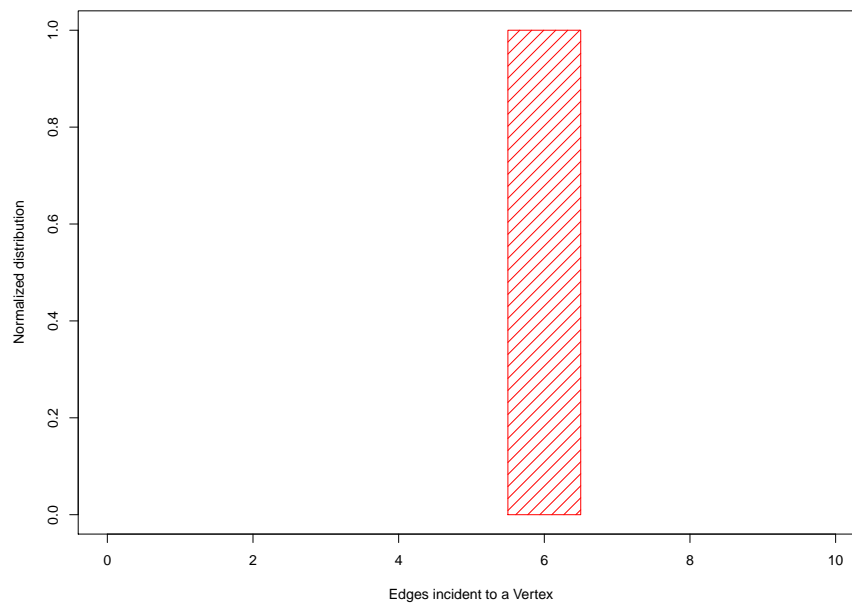
Degree distribution and robustness results for these “standard” graphs are shown in Figures 14 on page 87 through 17 on page 90. After the graphs were created, the next point of interest was to evaluate their robustness in the face of directed attack by an adversary who has total knowledge of the graph.

For each of the “standard” graphs, a game was constructed. The single edge with the highest centrality $c_B(E)$ when computing the shortest path between all vertices was computed based on techniques from [114]. This edge was removed. Average path length $L(G)$ and diameter $D(G)$ values were computed and saved. These three steps were repeated until the graph became disconnected (i.e., the attacker won). Plots for the “standard” graphs are shown (Figure 18 on page 91 through Figure 20 on page 93). For each graph type, the green circles represent the average path length for that graph after the edge with the highest “betweenness” was removed. The red squares are the diameter of the graph after the edge was removed. For all graphs, the edge with the highest “betweenness” was repeatedly identified and removed until the graph was disconnected. As expected, the Barabási graph became disconnected after the removal of a single edge. Disconnection for the Barabási and the Erdős-Rényi followed very closely the lower bounds of their respective degree distribution plots. Barabási preferential attachment graphs display a power-law degree distribution where the number of higher degreed nodes declines at an ax^k ($k < 0$) rate. Erdős-Rényi degree distribution is a normal random bell curve centered approximately at pn , where p is the probability of connecting two nodes. Watts – Strogatz small-worlds start out as regular graphs where all nodes are of the same degree. After the initial graph is created, each edge $e \in E(g)$ is evaluated for potential rewiring one end to a different node. No new edges are added to the original graph, but end points are moved, thus the total of the node degrees starts off as kn and remains that after the rewiring. The Watts – Strogatz graph took many more edge removals before disconnection, probably because of the relatively high number of vertices that were highly degreed.

After establishing a set of criteria for graph evaluation, USW graphs were subjected to the same game. A set of 9 USW graphs were constructed with 100 nodes and permutations of 0, 0.5, 1.0 for β and γ . Their degree distributions and robustness measurements are shown in Figures 21 on page 94 and 22 on page 95 respectively. A more complete set of degree distribution and disconnections based on various combinations of β and γ are found in Appendices H on page 489 and I on page 499 respectively.

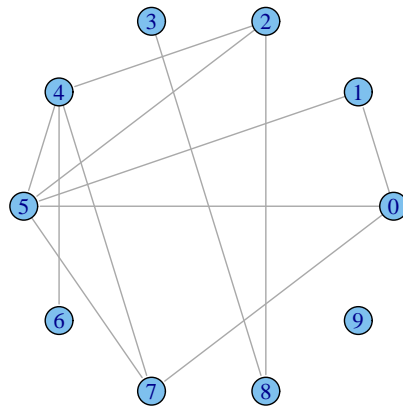


(a) Lattice graph

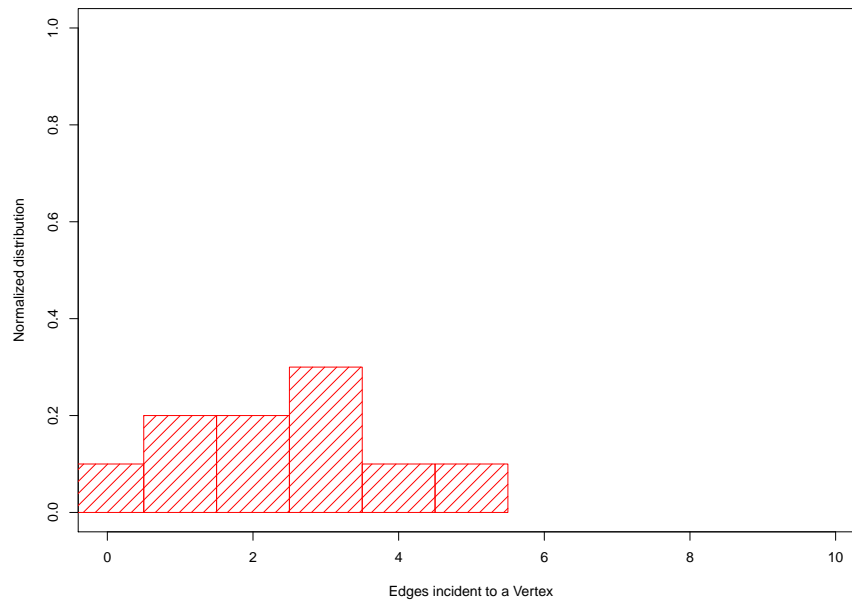


(b) Lattice graph degree distribution

Figure 14. Representative lattice graph and associated degree distribution.

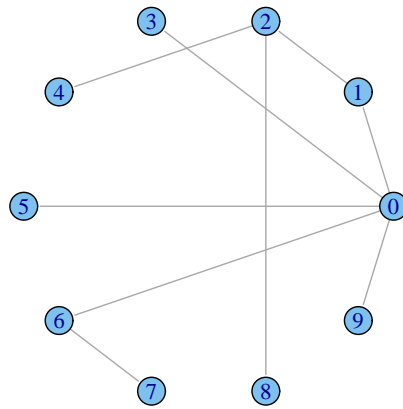


(a) Erdős-Rényi random graph

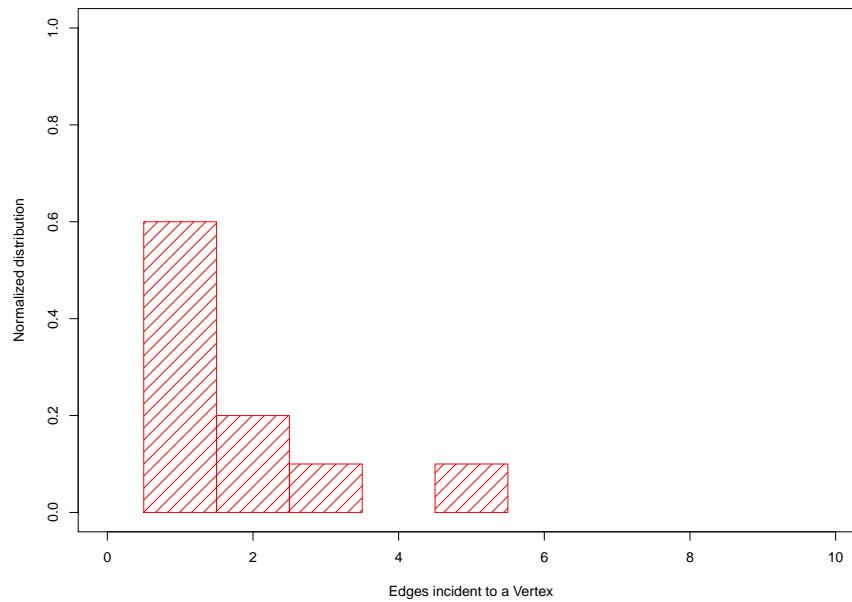


(b) Erdős-Rényi random graph degree distribution

Figure 15. Representative Erdős-Rényi random graph and associated degree distribution.

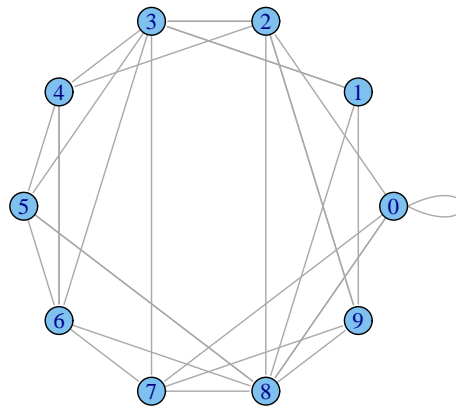


(a) Albert-Barabási scale free

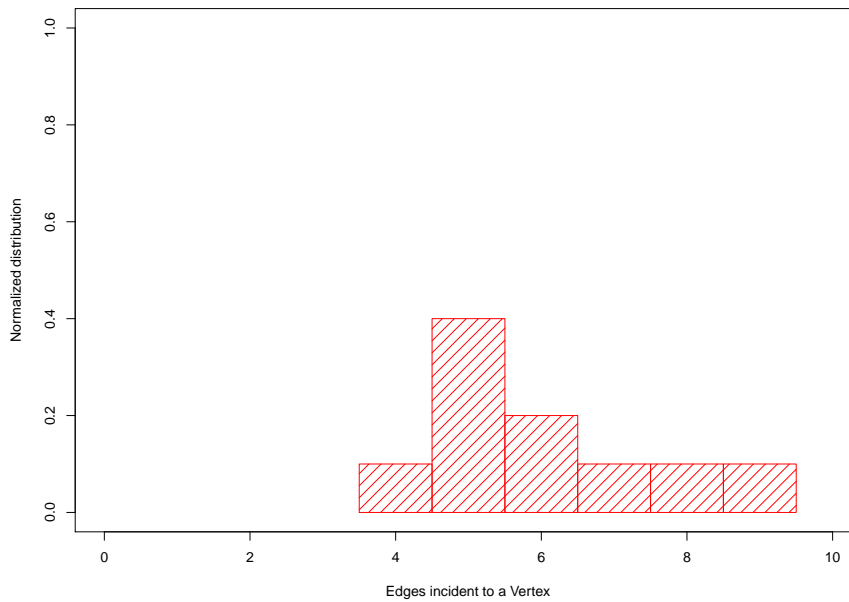


(b) Albert-Barabási scale free degree distribution

Figure 16. Representative Albert-Barabási scale free graph and associated degree distribution.



(a) Watts – Strogatz small-world graph



(b) Watts – Strogatz small-world graph degree distribution

Figure 17. Representative Watts – Strogatz small-world graph and associated degree distribution.

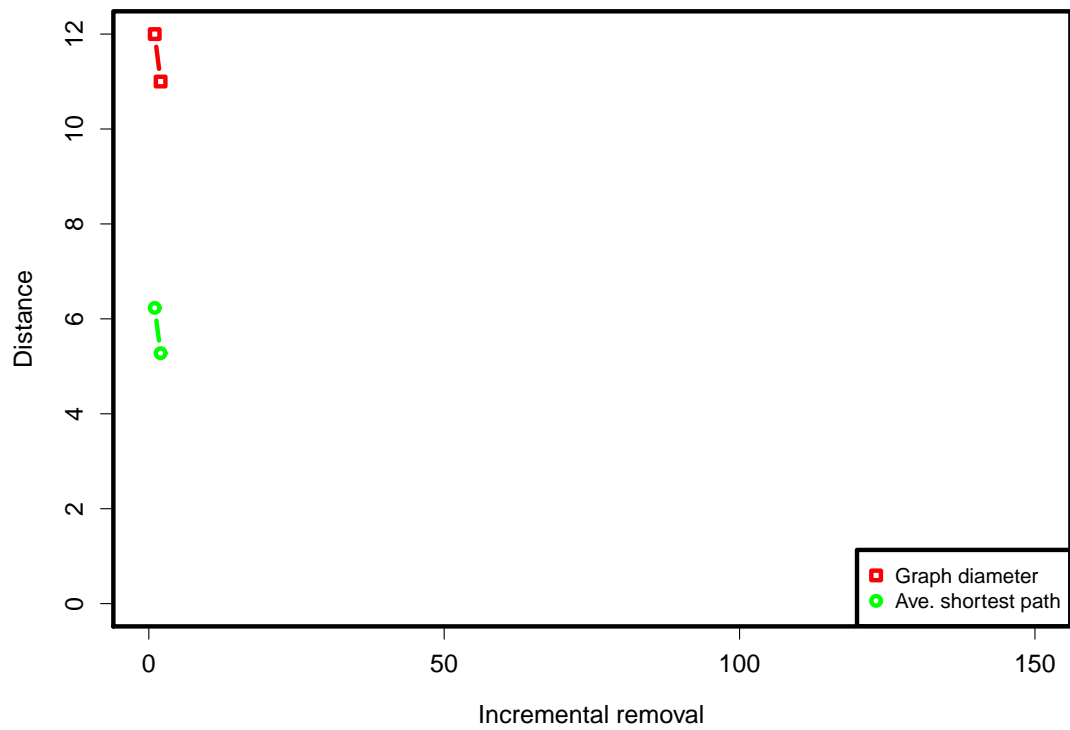


Figure 18. Barabási graph — disconnected after 1 removal. The disconnection of various “classical” graphs with the same number of nodes.

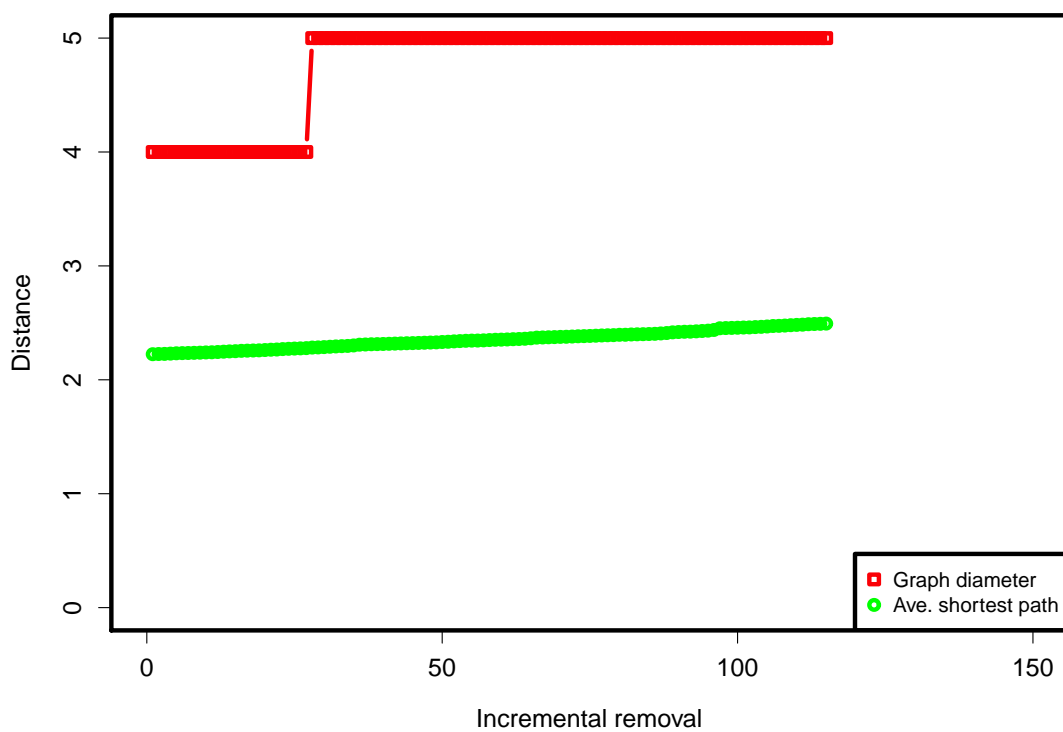


Figure 19. Erdős-Rényi graph — disconnected after 116 removals. The disconnection of various “classical” graphs with the same number of nodes.

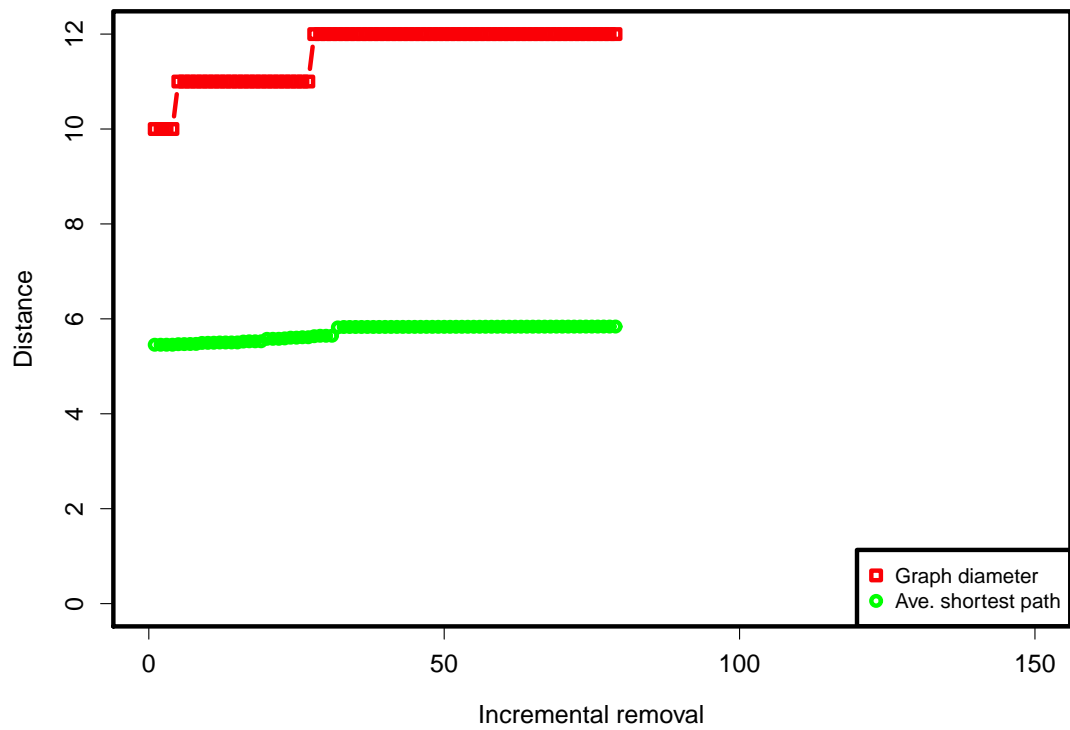


Figure 20. Watts – Strogatz graph — disconnected after 75 removals. The disconnection of various “classical” graphs with the same number of nodes.

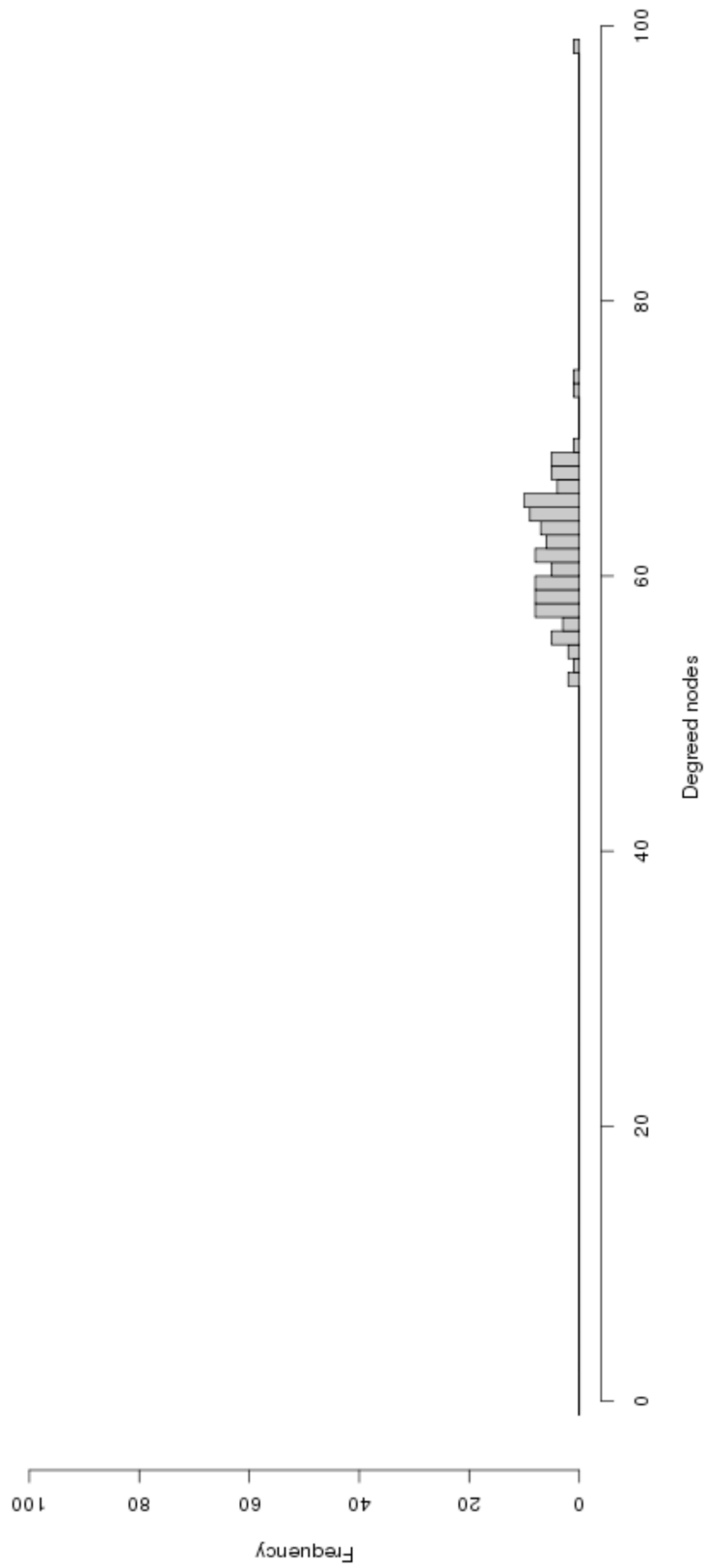


Figure 21. The degree distribution plot for $\beta = 0.0$ and $\gamma = 1.0$. Plots with midrange values of β and γ clearly show their random number origins.

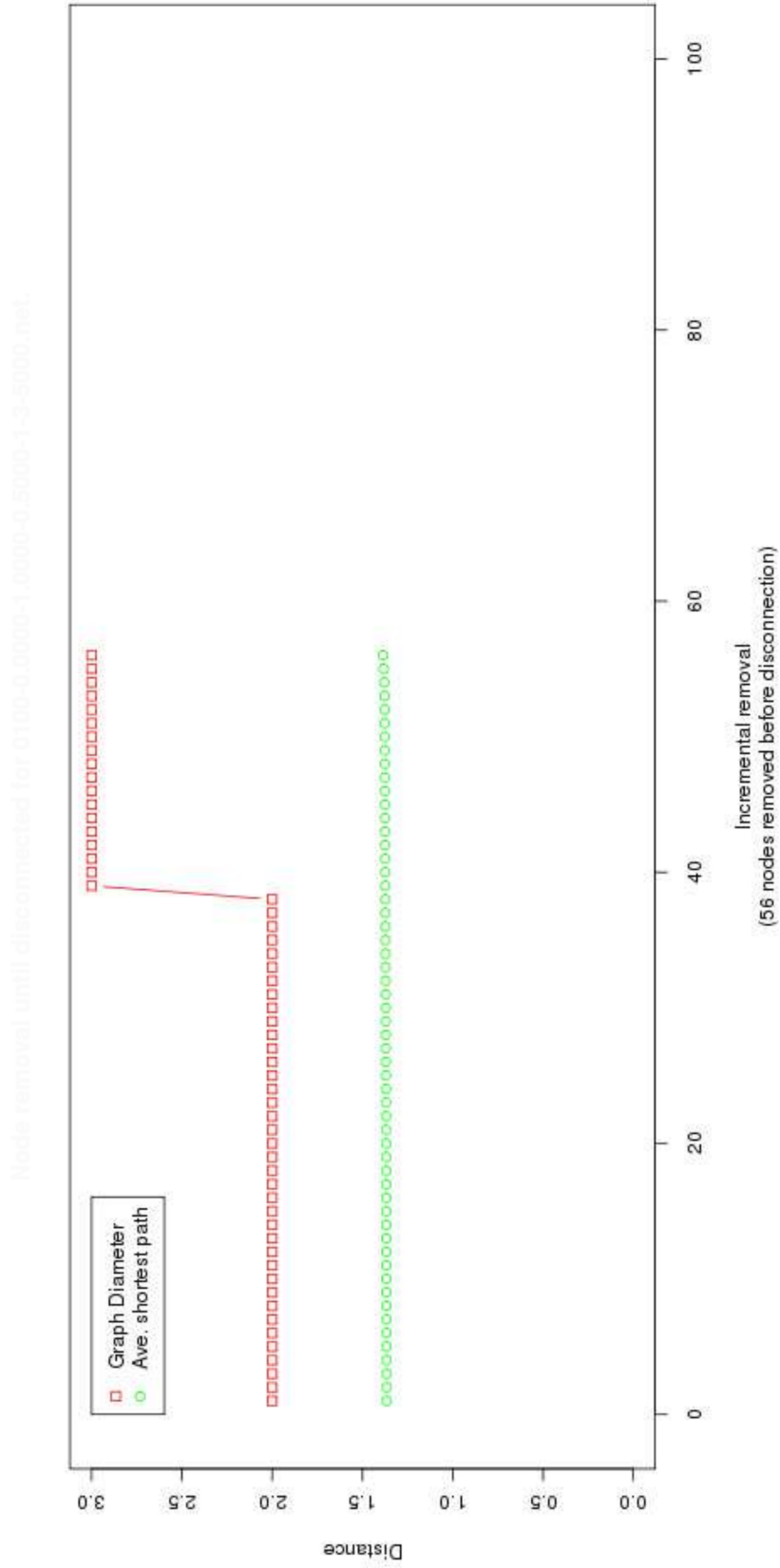


Figure 22. The disconnection plot for $\beta = 0.0$ and $\gamma = 1.0$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

One of the common factors in describing graphs and their robustness in the face of directed attacks is the k -edge connectivity.

“A graph is k -edge connected if every disconnecting set has at least k edges.”

Douglas B. West [166]

As can be seen in Figure 21 on page 94, β of 0 (regardless of γ) results in a degree distribution with a great deal of very low degreed nodes. As γ increases, the degree “hump” moves away from 0 and starts to widen and become lower.

Summary

Based on these simulations, robustness is a k -edge connection problem. The terminology from [167] uses $\lambda(G)$ notation when talking about edge-connectivity, and refers to any edge that is being removed as a “bridge.”

5.2.2 COMPARISON OF CONNECTED AND DISCONNECTED METRICS

The graph theory field is festooned with a wide variety of graph types. The graphs in this dissertation are limited to those that meet these requirements:

1. undirected,
2. simple,
3. self loops are not permitted, and
4. may have more than one component.

Connected metrics

Here we list a collection of characteristic metrics for connected graphs. In many cases the characteristic does not have meaning, or a computable value when the graph is not connected. Detailed explanation of each term is Appendix D.2 on page 385.

- *Average path length*($L(G)$)

- *Average inverse path length* ($L(G)^{-1}$)
- *Centrality, betweenness of an edge* ($c_B(e) = \sum_{s \neq t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}}$)
- *Centrality, betweenness of an edge relative to all edges in a graph* ($c_B(E) = \max(c_B(e) | e \in E)$)
- *Centrality, betweenness of a vertex* ($c_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$)
- *Centrality, betweenness of a vertex relative to all vertices in a graph* ($c_B(V) = \max(c_B(v) | v \in V)$)
- *Centrality, closeness of a vertex* ($c_C(u) = \frac{n-1}{\sum_{u \neq v \in V} d(u,v)}$)
- *Centrality, degree* ($c_D(v)$)
- *Clustering coefficient* ($C(G)$)
- *Degree* (k)
- *Diameter* ($D(G)$)
- *Eccentricity of a node* ($\epsilon(u)$)
- *Eccentricity of a graph* ($\epsilon(G)$)
- *Path length* ($d(u, v)$)
- *Radius of a graph* ($r(G)$)
- *Triangles based on a node* ($\lambda(v)$)

Disconnected metrics

Here we list a collection of characteristic metrics for disconnected graphs. In many cases the connected graph characteristic does not have meaning, or is not computable when the graph is disconnected. Detailed explanation of each term is in Appendix D.2 on page 385.

- *Average inverse path length* ($L(G)^{-1}$)
- *Constrained average path length* ($L(G) = \frac{1}{n(n-1)} \sum_{\substack{u \neq v \in V \\ 0 < d(u,v) < \infty}} d(u, v)$)

The effect of directivity and self loops

Many of the graph metric equations use the number of edges in the graph, but often the authors do not specify how the edges are selected or limited. Table 6 on the following page identifies how many edges can be used based on two criteria: whether or not the edges are directed or whether or not the graph permits edges back to the originating vertex. Based on these restrictions, the number of edges can range from $\frac{n*(n-1)}{2}$ to $n * (n + 1)$.

5.2.3 DERIVATION OF VARIOUS EQUATIONS RELATING TO RANDOM GRAPHS

The number of edges in a fully connected graph is bounded by the number of vertices in the graph (Equation 7).

$$|E| = \frac{n(n-1)}{2} \quad (7)$$

The number of edges in a random graph is dependent on the probability ρ that there is an edge between the two nodes (Equation 8).

$$|E| = \rho \frac{n(n-1)}{2} \quad (8)$$

The average degree of the nodes in the graph $\langle k \rangle$ is twice the number of the edges in the graph $|E|$ (Equation 9), which can be reduced to a function of the probability ρ that nodes will be connected (Equation 11).

$$\langle k \rangle = \frac{2|E|}{n} = \frac{2\rho n(n-1)}{2n} \quad (9)$$

$$= \frac{2\rho n(n-1)}{2n} \quad (10)$$

$$= \rho(n-1) \quad (11)$$

Table 6. Maximum number of edges based on directivity and self loops. A sample three node graph is used to illustrate the maximum number of edges a graph can have based on whether edges are directed or not and whether the graph permits edges that originate and return to the same node. The number of edges that can be used in various graph theoretical computations can range from $\frac{n*(n-1)}{2}$ to $n*(n+1)$. The apparently redundant double edges when directed edges are allowed and self loops are permitted reflect that there is two-way communication. In effect, the node is “talking” to itself.

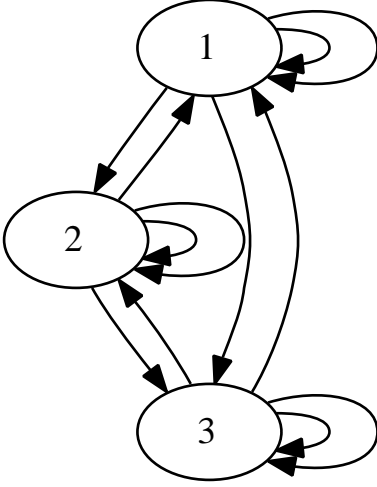
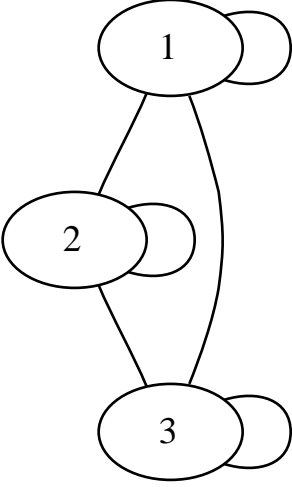
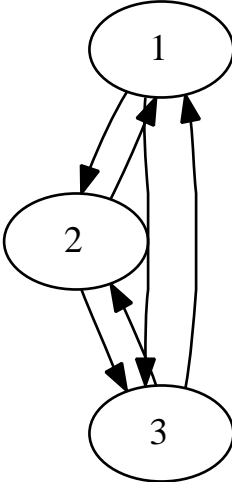
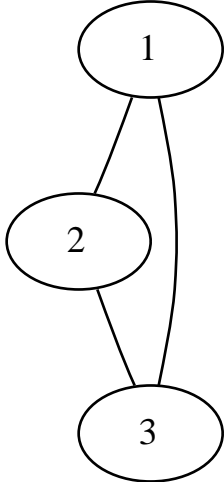
		Are directed edges permitted?	
		Yes	No
Self loops permitted?	Yes		
	$ E_{max} = n * (n + 1) = 12$	$ E_{max} = \frac{n*(n+1)}{2} = 6$	
Self loops permitted?	No		
	$ E_{max} = n * (n - 1) = 6$	$ E_{max} = \frac{n*(n-1)}{2} = 3$	

Table 7. A summary list of useful equations. These equations are useful for estimating n , $\langle k \rangle$ and n for random graphs.

Given	Estimate	
$L(G), \langle k \rangle$	$n = L(G)^{\langle k \rangle}$	(15)
$L(G), n$	$\langle k \rangle = \frac{\lg n}{\lg L(G)}$	(16)
$n, \langle k \rangle$	$L(G) = \sqrt{\langle k \rangle n}$	(17)
$n, \langle k \rangle$	$C(G) = \frac{\langle k \rangle}{n}$	(18)

The relationship between average degree of the graph $\langle k \rangle$, the number of nodes n in the graph n , and the average path length $L(G)$ is easily shown (Table 7).

$$L(G)^{\langle k \rangle} = n \quad (12)$$

$$\langle k \rangle \lg L(G) = \lg n \quad (13)$$

$$\langle k \rangle = \frac{\lg n}{\lg L(G)} \quad (14)$$

5.2.4 DAMAGE ASSESSMENT

A number of metrics reporting on different structural aspects a graph $G(V, E)$ have been proposed and used when comparing graphs. Some of these metrics are summarized in Table 8 on the following page. Many of these metrics are applicable when the graph is connected, but the definition of a graph does not require that

Table 8. Comparing the expected average path length and expected clustering coefficient for lattice, small-world and random graphs. Small-world region of values for $L(G)$ and $C(G)$ are wide and not exact. The relationship between $L(G)$ and $C(G)$ graph values are the real determinates as to whether any particular graph is a small-world [113].

Graph type	$L(G)$	$C(G)$	See
Random	$L(G) \sim \frac{\ln(n)}{\ln(k)}$	$C(G) \sim \frac{k}{n}$	[43]
Scale free	$L(G) = A \ln(n - B) + C$	<i>Not predictable</i>	[170]
Lattice	$L(G) \sim \frac{n}{2k} \gg 1$	$C(G)_{Lattice} = \frac{(3k-3)}{2(2k-1)}$	[43, 171]
Small-world	$\frac{n}{2k} > L(G) > \frac{\ln(n)}{\ln(k)}$	$C(G) \sim \frac{3}{4}$	[43]

it be connected. Our interest is in quantifying the damage to a graph after one or more components have been removed. None of the available metrics are applicable to quantifying damage when the graph that results from the removal may be disconnected.

Menger's theorem [168, 167] sets the limit on the number of vertices or edges that must be removed to cut the graph. The greater the $\kappa(G)$ or $\lambda(G)$ that must be removed, the greater the connectivity and the robustness of the graph. The following relationship between vertex $\kappa(G)$, edge $\lambda(G)$ and degree $\delta(G)$ connectivity holds true for all non-trivial graphs [169]:

$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

Discussion

The idea of quantifying the damage to a graph $G(V, E)$ because of failure or attack is at the heart of quantifying the robustness of the graph. Intuitively, robustness is the ability of the graph to continue to perform its function when some number of its elemental components have been lost due to some attack profile. (See Section 5.5.5 on page 159 for a description of different attack profiles.) We will now investigate how a graph performs with some small examples and then extend the results to larger graphs. The investigation uses these metrics:

- *Average path length* ($L(G)$)
- *Average inverse path length* ($L(G)^{-1}$)
- *Clustering coefficient* ($C(G)$)
- *Damage* ($Damage(G)$)

Approach

The approach is to create a series of test graphs (some easy, some hard, some pathological) and see how the metrics work and what kinds of insights the graphs revealed. Things to look for in the images, using Figure 27 on page 112 as an example, include:

1. Each figure has three parts. They are:
 - (a) The initial graph (Figure 27(a) on page 112). This graph may be connected or not depending on the sample that is being tested.
 - (b) The consolidated graph (Figure 27(b) on page 112). Many of the graph metrics will not work on a disconnected graph, therefore the initial graph is made into a connected graph. The components of the disconnected initial graph are ordered based on their size. The highest degree vertices in the two largest components are connected with a new edge. The process of ordering and connecting is repeated until there is a single component. This new graph is called the consolidated graph. Vertices from the initial graph that are connected via a new edge are colored in red.
 - (c) The action plot (Figure 27(c) on page 112). For the purposes of this discussion, an action is defined as the addition or removal of an edge. Depending on the size of the initial graph and the number of edges, the number of actions per graph may be different.
2. The action graph has several interesting parts. Starting first with the three vertical divisions. The divisions are:
 - (a) *Consolidation* (labeled “C”). During this phase of the action, the graph is being *consolidated*. Edges are being added as necessary to create a single component. The number of edges added is 1 less than the original number of components.

- (b) *Adding* (labeled “Adding”). After the graph is connected, additional edges are added until the graph is fully connected. The number of additional edges necessary depends on the initial graph’s fragmentation.
- (c) *Destruction* (labeled “Destruction”). After the graph is fully connected, edges are removed until the graph is fully disconnected.

The action plot shows the history of the graph from initial creation, through extensive construction and finally total destruction.

A number of interesting data items are plotted on the action plot. They are:

- (a) Average Inverse Path Length (AIPL) ($L(G)^{-1}$). The AIPL increases relatively rapidly during the Consolidation phase, slowly increases to 1 during the Adding phase, and then decreases to 0 during Destruction. The math behind the AIPL is such that a totally disconnected graph has an AIPL of 0 and a fully connected graph has an AIPL of 1. During the Consolidation phase, the graph has more and more edges being added until it is connected and during Adding, the effects of the additional edges are less pronounced. During Destruction, interesting things happen with the curve.

The selection of which edge to remove is based on the same attack profile ($A_{E,H}$) Edge High Betweenness. This is the attack profile that a determined attacker would use given total knowledge of the graph and the resources to take it to destruction. During Destruction the AIPL curve has pronounced and repeated steps downward. The steps occur when some component becomes disconnected from the graph. The number of edges necessary to disconnect a component decreases as the graph becomes less connected. In Figure 27(c) on page 112, it takes 5 removals to isolate the first node, four to isolate the second, three the third and so on until the last two nodes are isolated.

- (b) Damage ($Damage(G)$). The damage curve is computed based on the assumption that the initial graph is being compared to a fully connected graph. As more edges are added to the initial graph, the damage decreases. Because adding more edges during the Adding phase brings the consolidated graph into closer alignment with the fully connected graph,

the damage curve rapidly approaches 0 and remains close to 0 for a significant portion of the Destruction phase. The damage curve has steps in it, for the same reason as the AIPL curve, but the damage curve increases from 0 to 1.

Attributes of $Damage(G)$ metric [172] are:

- i. Different fragmentation cases result in different numerical values,
- ii. Test cases where the size of the fragments have been scaled, and the entire graph (for instance, increased by a factor of 10 or 0.1) should result in the same value,
- iii. The value is useful without additional information about the graph (i.e., the metric is graph independent and does not require knowledge of the graph in a different state),
- iv. The metric should be *unitless*.

$Damage(G)$ test cases are summarized in (Table 9 on page 107), and in Tables 23 on page 108 through 26 on page 111.

- (c) Clustering Coefficient ($C(G)$). During the Consolidation phase, the CC curve is very dependent on the initial graph. Sometimes it starts high and then drops while other times it stays low for the the entire phase. In all cases, during Adding, the CC increases to 1 as more and more edges are added and more and more triangles are created in the graph. During Destruction, the curve has “waves” as a component becomes more isolated. When a component becomes totally isolated, the CC curve bounces back up to 1 because the node no longer is a part of the computations. This “wave” action continues until the last three nodes are connected by two edges and no triangles are possible.
- (d) Average Path Length ($L(G)$). The APL is dependent on a connected graph and therefore has no meaning during the Consolidation phase. During the Adding phase, it decreases to 1 as more and more edges are added to the graph. The graph is still connected during the beginning of the Destruction phase until the first component is isolated.

Equations 19 on the following page through 24 on the next page were derived from Albert, Jeong and Barabási (AJB) [73], and are the basic definitions for the number of nodes n in the graph at any point in time. At that point in time, there is

a set of clusters s in the graph. If the graph is connected then there is one cluster. In [73], the node with the highest degree is removed (along with its adjacent edges) and all values are computed again. n starts at an initial value and is decremented at each time step until all nodes are disconnected.

Equation 21 is the number of clusters (components) in the set of clusters c . Equation 22 identifies the size of the largest connected component LCC in c . Equation 23 is the ratio (percentage) of the size of LCC to the current n . Equation 24 is the mean size of all the remaining clusters (i.e., less the LCC) in the graph.

$$n \stackrel{\text{def}}{=} \text{number of nodes in } G \quad (19)$$

$$c \stackrel{\text{def}}{=} \text{set of clusters in } G \quad (20)$$

$$m = |c| \quad (21)$$

$$LCC = \max(|\langle c \rangle|) \quad (22)$$

$$S = \frac{|LCC|}{n} \quad (23)$$

$$s = \frac{n - |LCC|}{m - 1} \quad (24)$$

The various characteristics in equations 19 through 24 are subject to some mathematical constraints. These constraints are:

$$1 \leq |LCC| \leq n \quad (25)$$

$$m_{min} = \begin{cases} 1 & \text{when } |LCC| == n \\ 2 & \text{otherwise} \end{cases} \quad (26)$$

$$m_{max} = \begin{cases} 1 & \text{when } |LCC| == n \\ n - LCC & \text{otherwise} \end{cases} \quad (27)$$

$$m_{min} \leq m \leq m_{max} \quad (28)$$

$$1 \leq j \leq m \quad (29)$$

In addition to the mathematical constraints, there are a series of logical constraints. These constraints are:

1. $s < |LCC|$ (Equation 24)
2. S will always be in the range $\frac{1}{n} \leq S \leq 1$ (Equation 23)

3. If $|LCC| == 1$ then $\forall c : |c_i = 1 \implies m = n$ meaning that anytime where $m == n$ and $|LCC| \neq 1$ is a contradiction and can not happen.
4. If $|LCC| == \frac{n}{2} \implies m_{max} = \frac{n}{2}$ where $\forall c_i : |c_i| == 1$.
5. If $|LCC| == \frac{n}{j} \implies m_{max} = \frac{n}{j}$ where $\forall c_i : |c_i| == 1$.
6. If $|LCC| == (n - 1) \implies m = 2$.

Constraint 2 on the preceding page limits $|LCC|$ between n and 1. The $|LCC|$ will equal n when the graph is connected (i.e., the graph has not been fragmented). LCC will equal 1 when the graph is totally disconnected (i.e., the graph is composed of only nodes and no edges). Equation 28 on the previous page limits the number of fragments m to between 1 and n . Equation 29 on the preceding page limits the number of fragments to the greater of 1 (when the graph is totally connected; i.e. one cluster) or n (when the graph is totally disconnected). Albert, Jeong and Barabási (AJB) were interested in the fraction f of their graphs that had to be removed to cross a percolation threshold that would cause the graph to become severely fragmented. We are interested in the continuum of the graph's performance while it is connected and after it is disconnected. The percolation threshold is of passing interest, while the ideas that they espouse serve as starting point for our investigation.

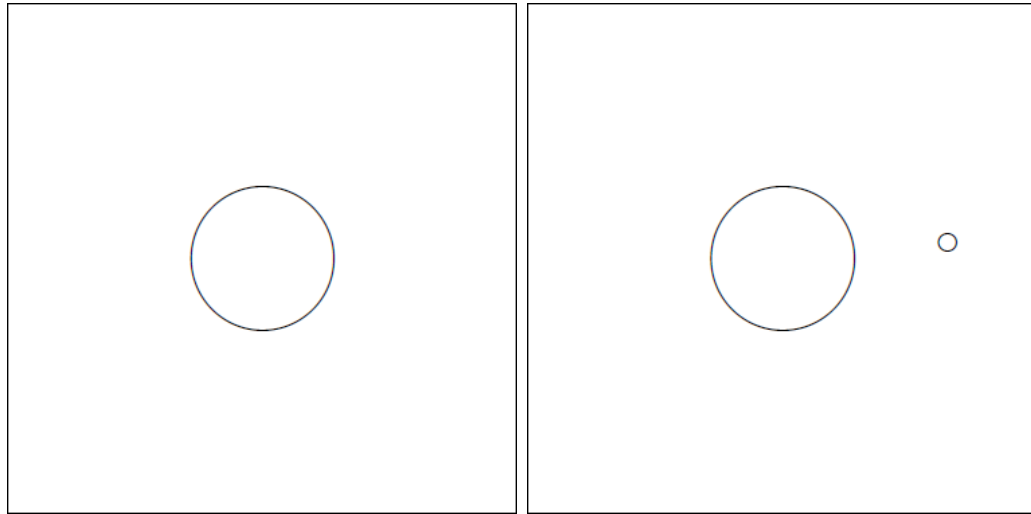
Summary

There are a few things that can be said.

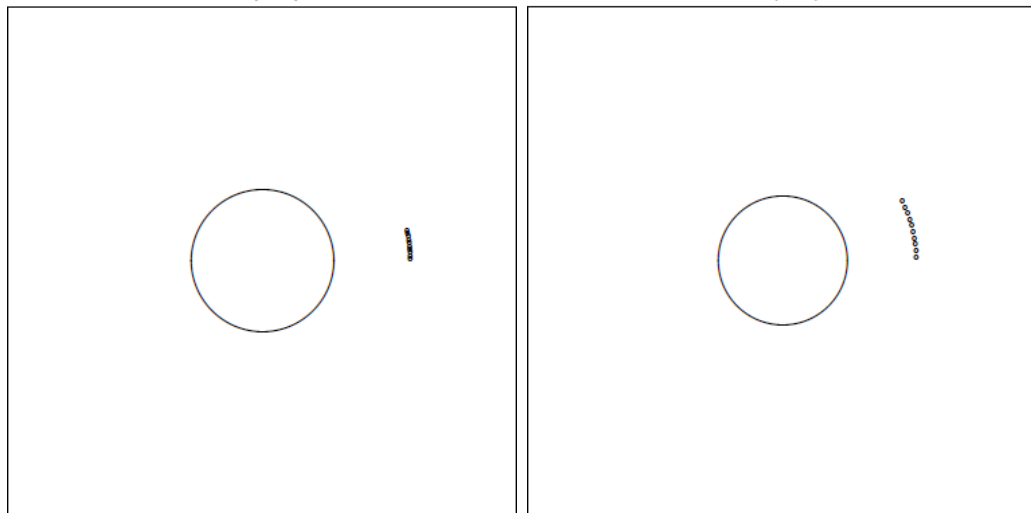
1. The Damage $Damage(G)$ curve behaves in a reasonable way during all phases.
2. Based on analysis, the AIPL $L(G)^{-1}$ curve behaves in a reasonable way. The curve's behavior during the Destruction phase was unexpected but understandable. Each "step" in the curve during the Destruction phase is the result of one more component being disconnected. When the graph is totally disconnected, the curve goes to 0.
3. The Clustering Coefficient $C(G)$ curve was reasonable in most cases. The case of the two triangles (Figure 27 on page 112) dip during the Adding phase is due to the number of edges in the graph increasing, but their placement does not increase how well-connected the graph has become. In all other cases the

Table 9. Comparing Albert, Jeong and Barabási’s raw s to our damage metric for a collection of test graphs. Raw s [73] and $Damage(G)$ [172] are evaluated as surrogates for the “health” of the graph. A healthy graph would have a value close to 0, while a totally disconnected graph would have a value of 1.

Name	s	$Damage(G)$
<i>100</i>	NaN	0.00
<i>90,10</i>	10.00	0.14
<i>90...1</i>	1.00	0.16
<i>80...2</i>	2.00	0.31
<i>50,50</i>	50.00	0.39
<i>50,49,1</i>	25.00	0.40
<i>50,40,10</i>	25.00	0.46
<i>50,30,10,10</i>	16.67	0.52
<i>50...5</i>	5.00	0.64
<i>20...20</i>	20.00	0.66
<i>16...1</i>	8.40	0.78
<i>10...10</i>	10.00	0.81
<i>10...9</i>	9.00	0.82
<i>1...1</i>	1.00	1.00

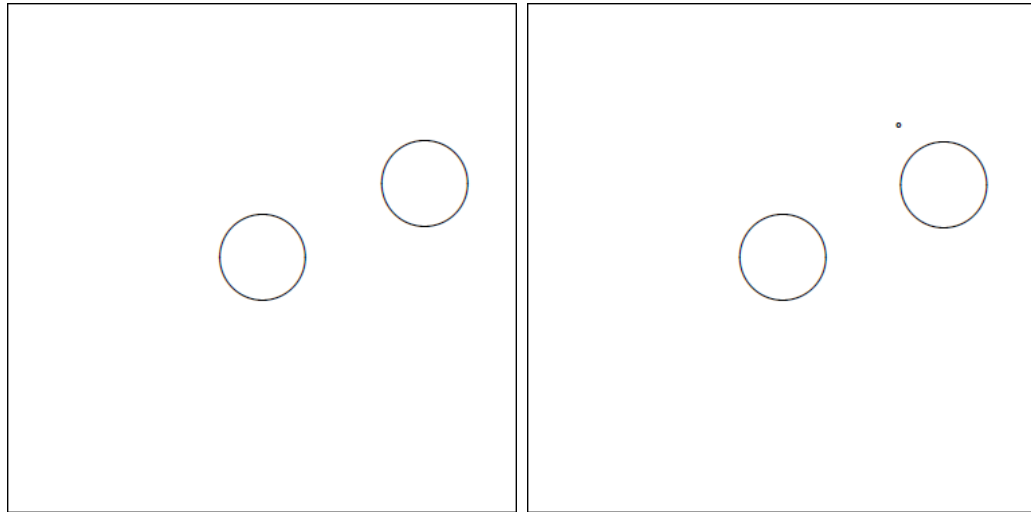


(a) 100 diagram, $s = \text{NaN}$, $\text{Damage}(G) = 0.00$ (b) 90,10 diagram, $s = 10.00$, $\text{Damage}(G) = 0.14$

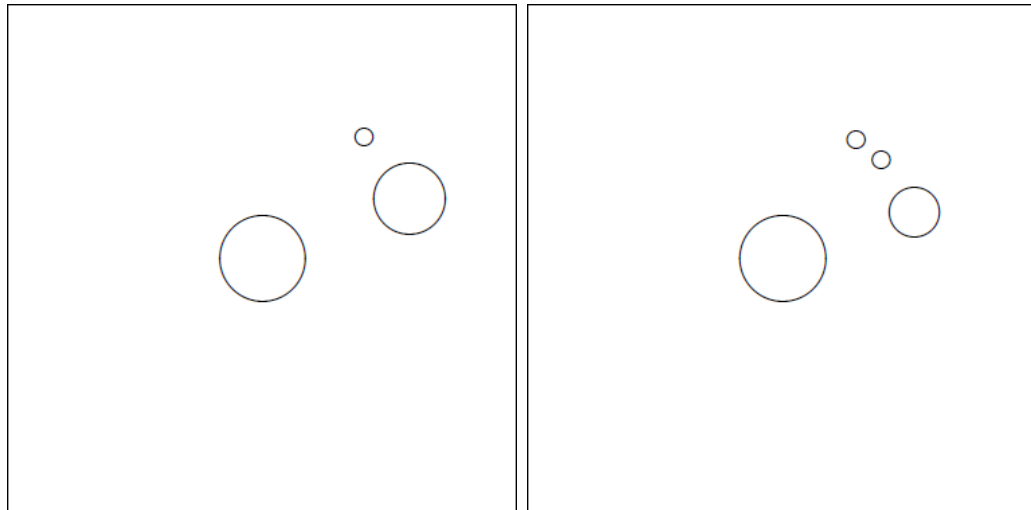


(c) 90 ... 1 diagram, $s = 1.00$, $\text{Damage}(G) = 0.16$ (d) 80 ... 2 diagram, $s = 2.00$, $\text{Damage}(G) = 0.31$

Figure 23. Notional diagrams for test cases *100* , *90,10* , *90...1* and *80...2* . The entire graph is contained within the square. The LCC is represented by the large inner circle, smaller fragments are represented by the outer circles. The circles represent the relative sizes of the different fragments.

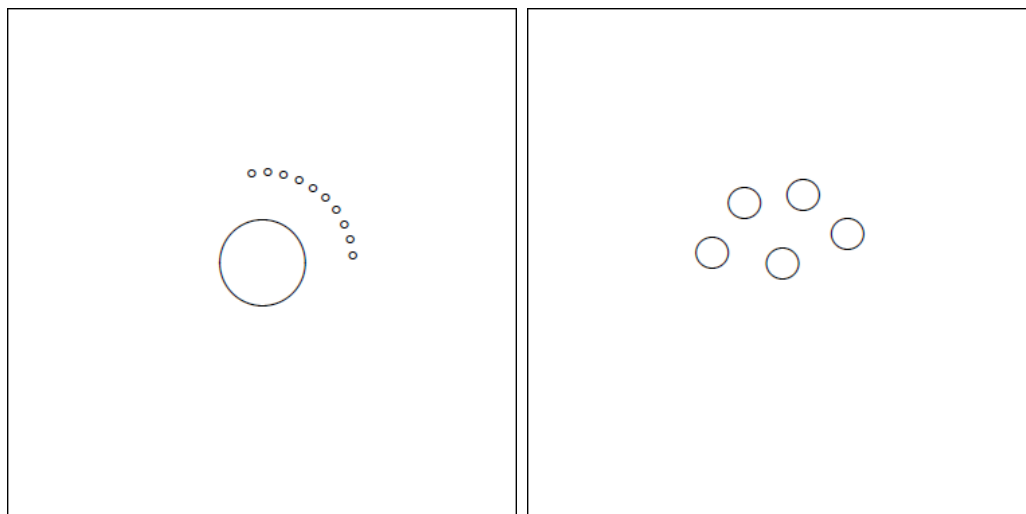


(a) 50,50 diagram, $s = 50.00$, $\text{Damage}(G) = 0.39$ (b) 50,49,1 diagram, $s = 25.00$, $\text{Damage}(G) = 0.40$

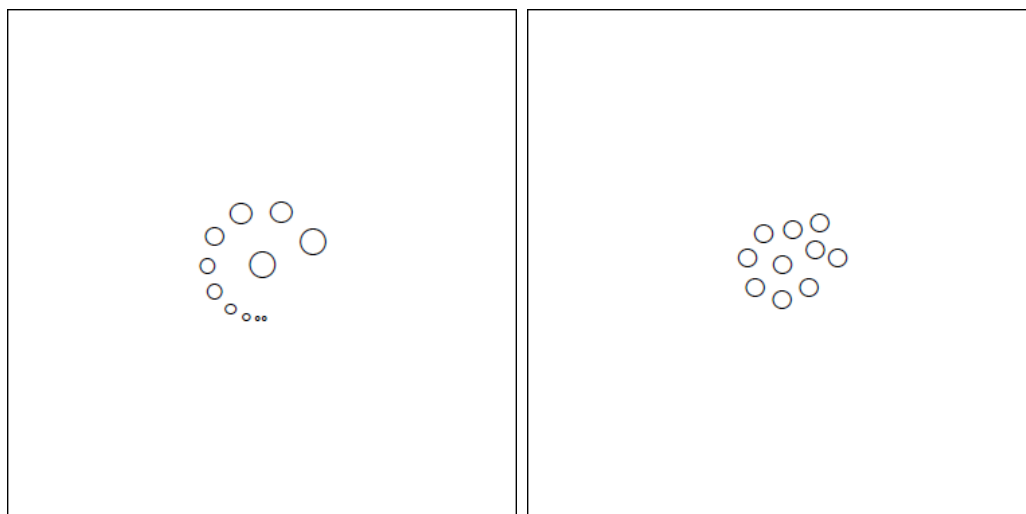


(c) 50,40,10 diagram, $s = 25.00$, $\text{Damage}(G) = 0.46$ (d) 50,30,10,10 diagram, $s = 16.67$, $\text{Damage}(G) = 0.52$

Figure 24. Notional diagrams for test cases $50,50$, $50,49,1$, $50,40,10$ and $50,30,10,10$. The entire graph is contained within the square. The LCC is represented by the large inner circle, smaller fragments are represented by the outer circles. The circles represent the relative sizes of the different fragments.



(a) 50 ... 5 diagram, $s = 5.00$, $\text{Damage}(G) = 0.64$ (b) 20 ... 20 diagram, $s = 20.00$, $\text{Damage}(G) = 0.66$



(c) 16 ... 1 diagram, $s = 8.40$, $\text{Damage}(G) = 0.78$ (d) 10 ... 10 diagram, $s = 10.00$, $\text{Damage}(G) = 0.81$

Figure 25. Notional diagrams for test cases $50 \dots 5$, $20 \dots 20$, $16 \dots 1$ and $10 \dots 10$. The entire graph is contained within the square. The LCC is represented by the large inner circle, smaller fragments are represented by the outer circles. The circles represent the relative sizes of the different fragments.

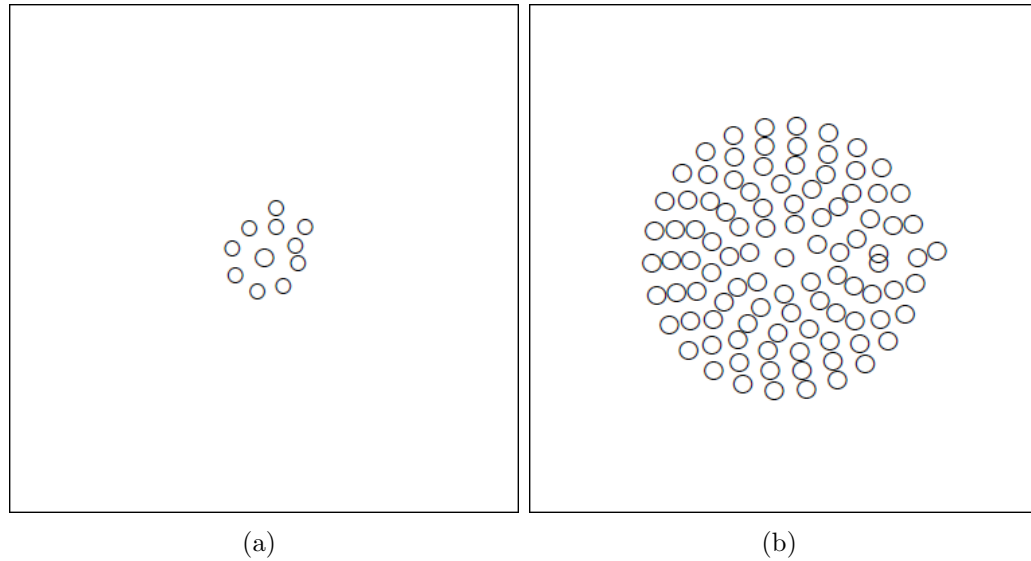


Figure 26. Notional diagrams for test cases $10 \dots 9$ and $1 \dots 1$. The entire graph is contained within the square. The LCC is represented by the large inner circle, smaller fragments are represented by the outer circles. The circles represent the relative sizes of the different fragments.

behavior during Adding make sense. The curve's behavior during destruction was unexpected and quite delightful.

4. The Average Path Length $L(G)$ curve's behavior during the Adding phase was to be expected, while its behavior during the first part of destruction was unexpected. In all cases, the $L(G)$ increased until the first disconnection because the average path increased.

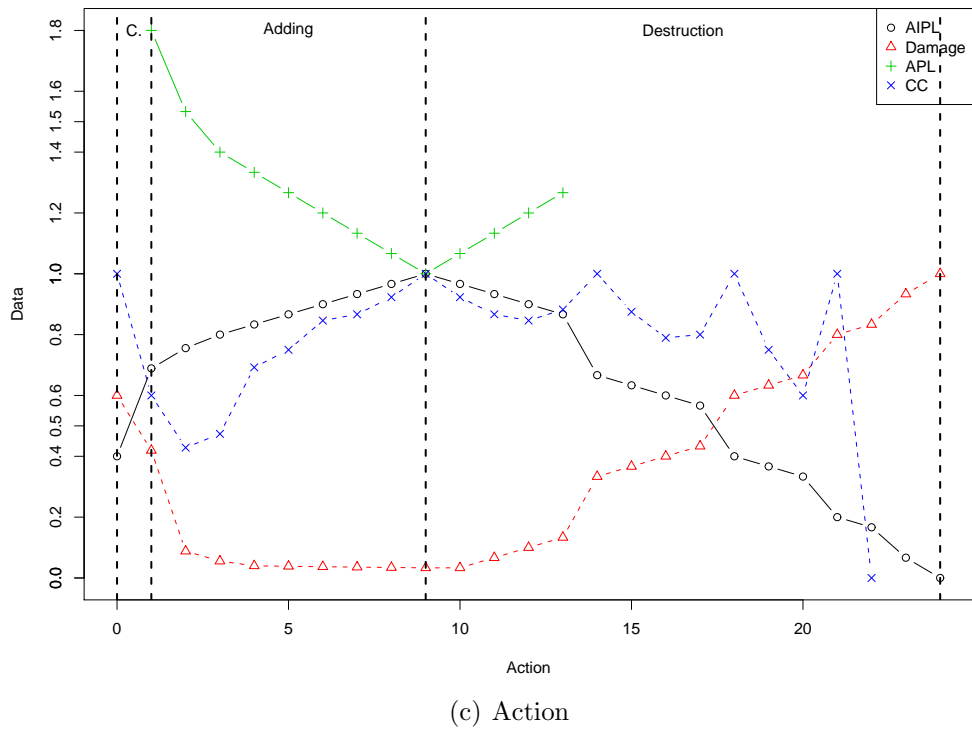
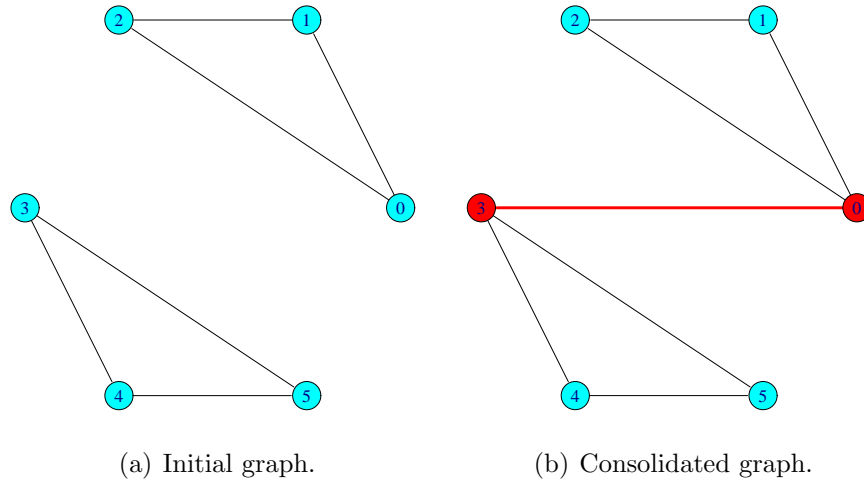


Figure 27. Sample two triangles graph. Initially 2 clusters, 6 nodes and 6 edges.

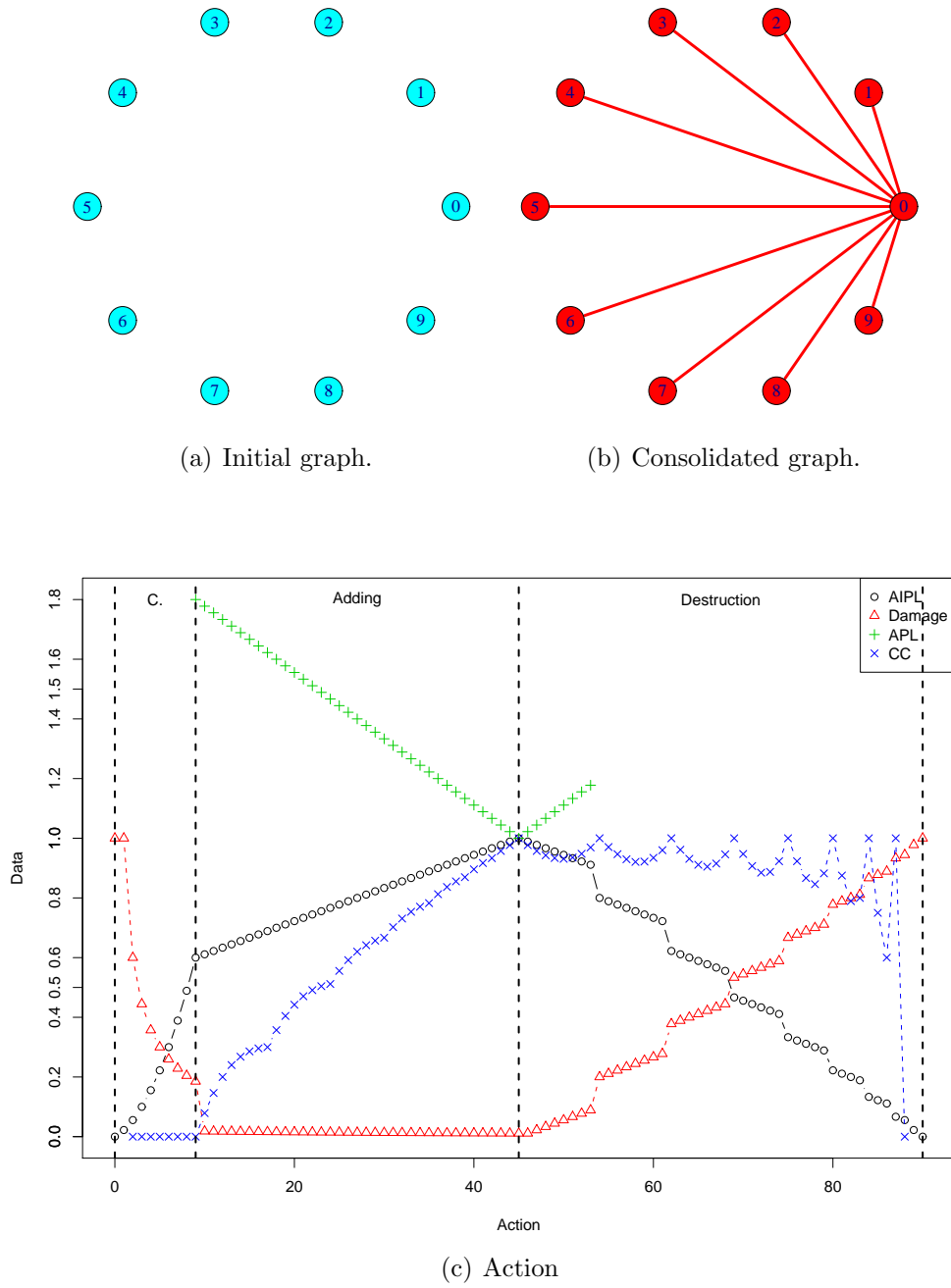


Figure 28. Sample totally disconnected graph. Initially 10 clusters, 10 nodes and 0 edges.

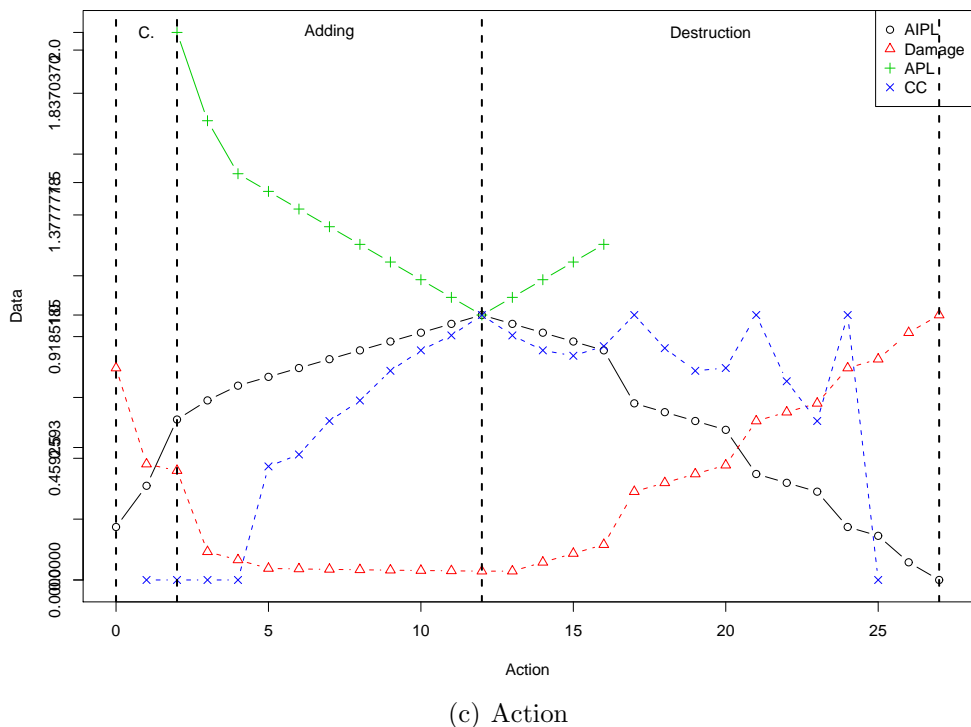
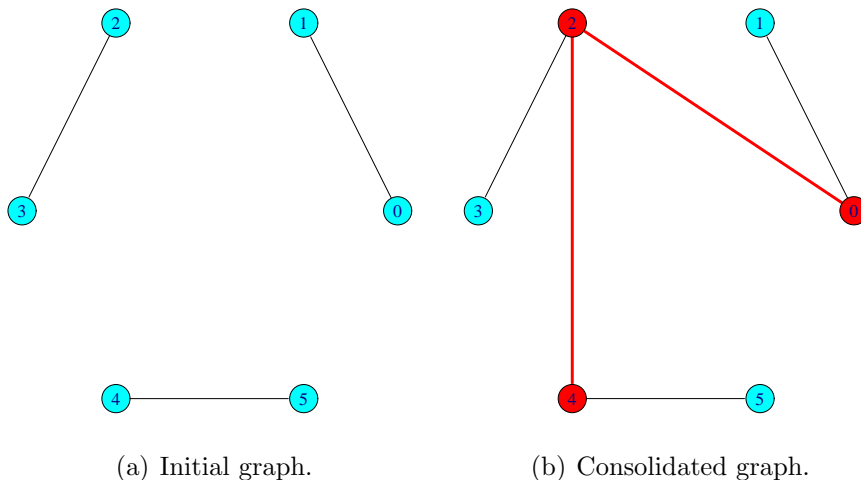


Figure 29. Sample 3 pairs graph. Initially 3 clusters, 6 nodes and 3 edges.

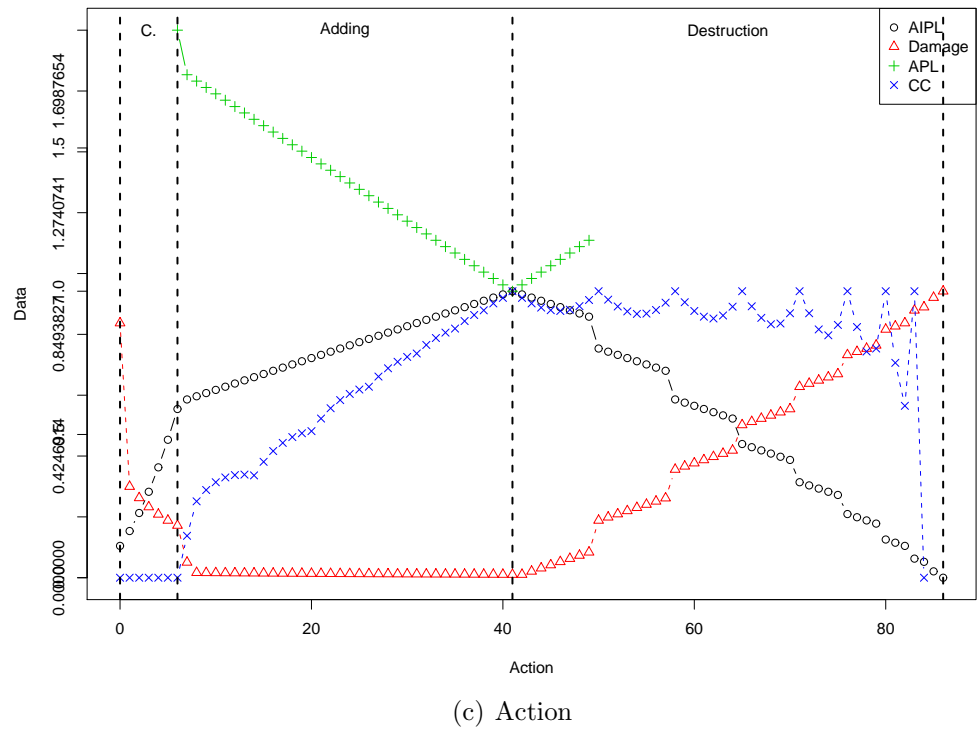
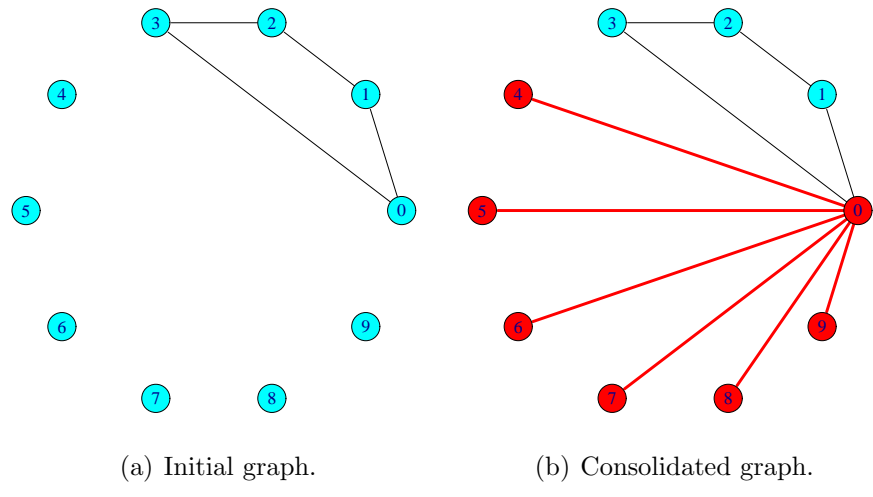


Figure 30. Sample 1 LCC and 6 singletons graph. Initially 7 clusters, 10 nodes and 4 edges.

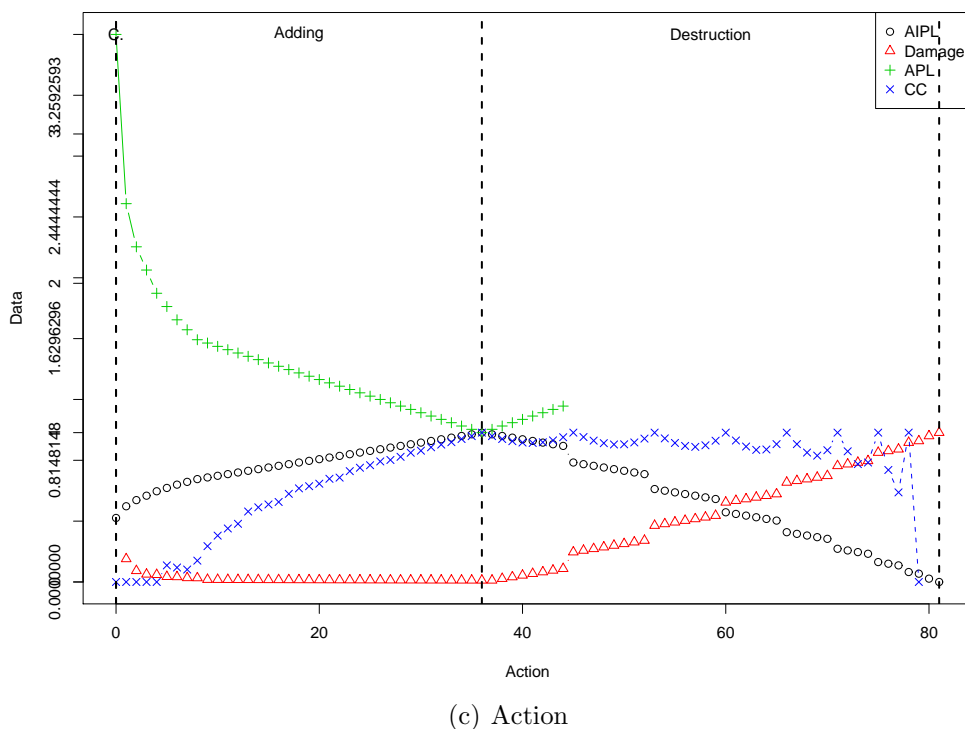
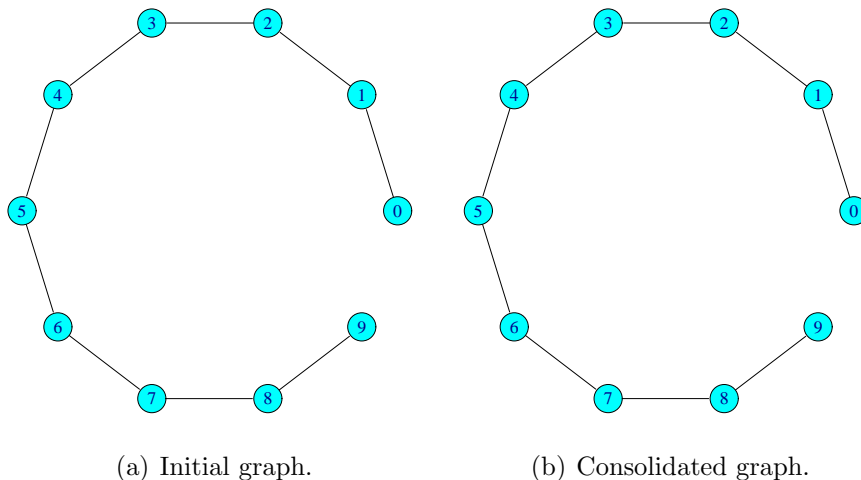


Figure 31. Sample fully connected graph. Initially 1 cluster, 10 nodes and 9 edges.

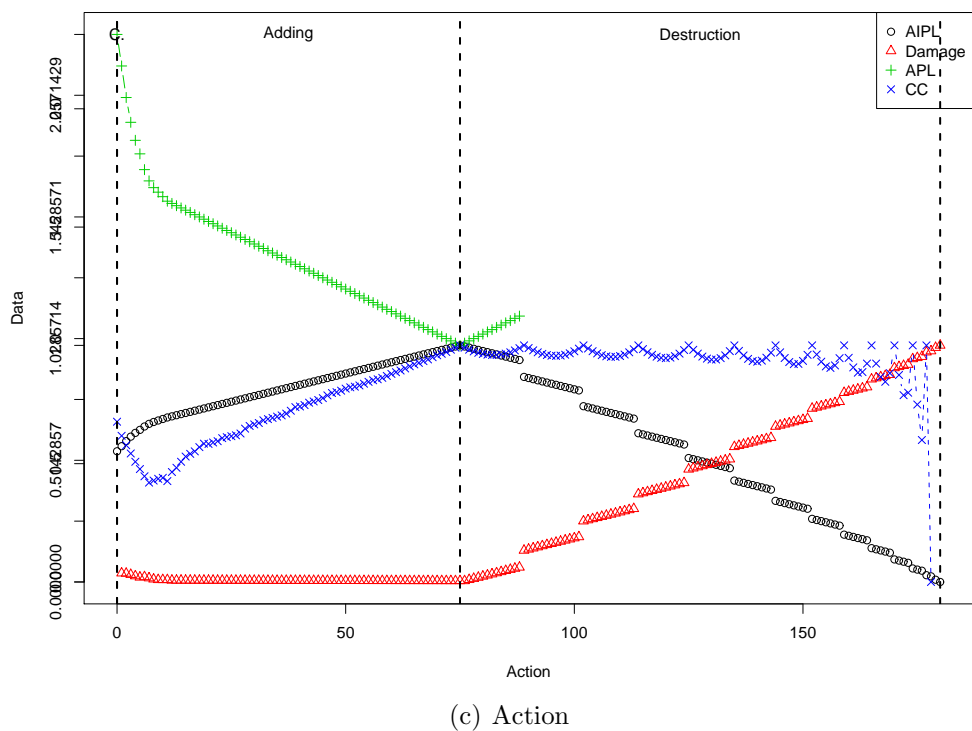
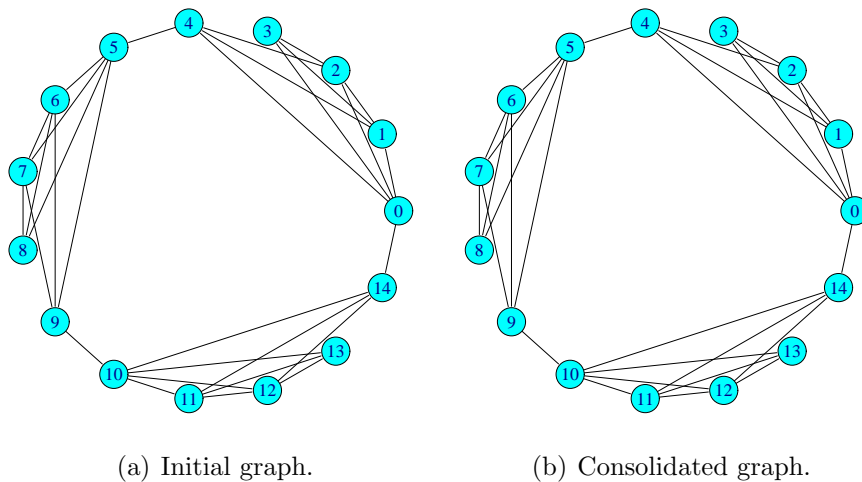


Figure 32. Sample cave man graph. Initially 1 cluster, 15 nodes and 30 edges.

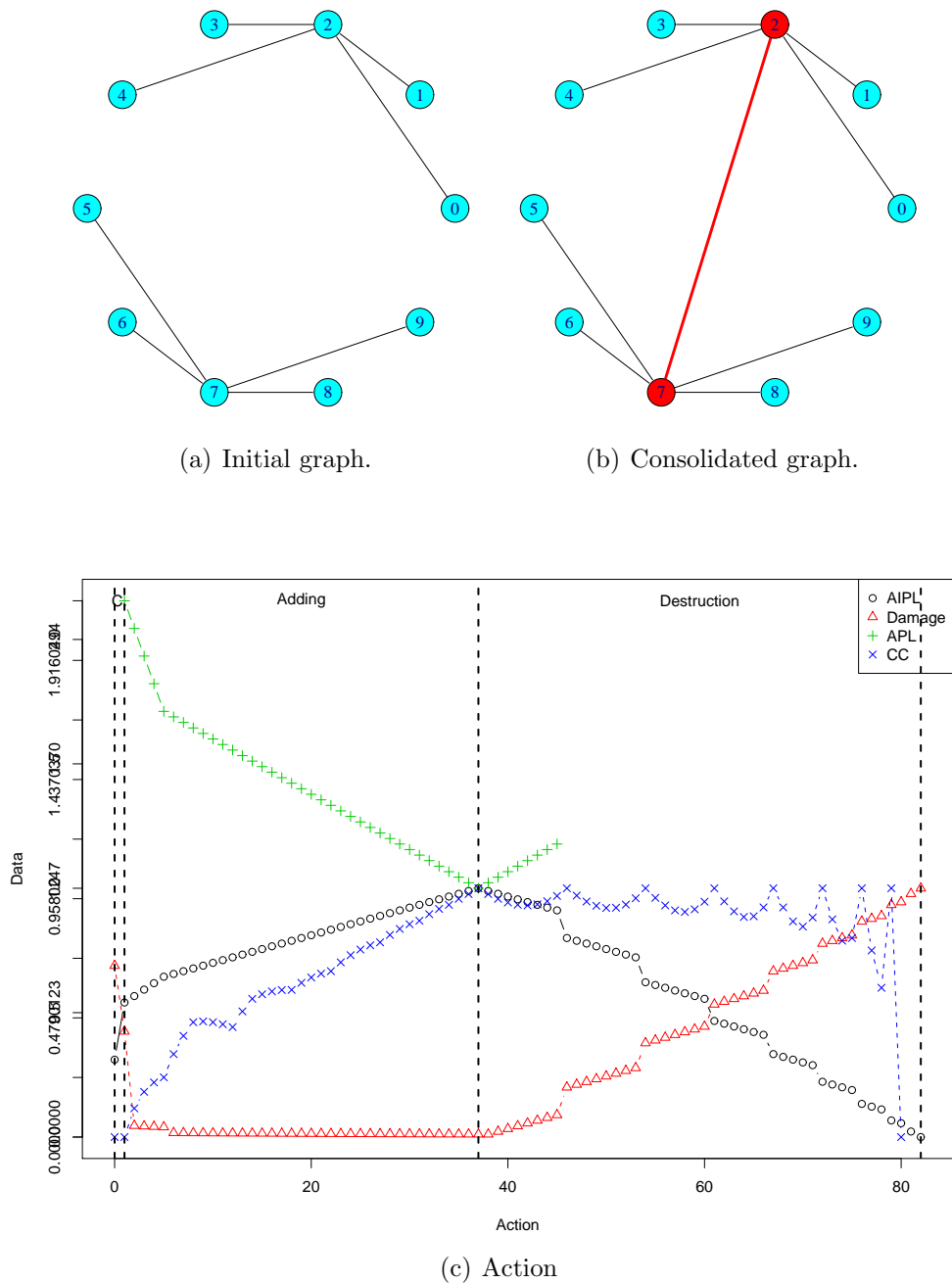


Figure 33. Sample butterfly graph. Initially 2 clusters, 10 nodes and 8 edges.

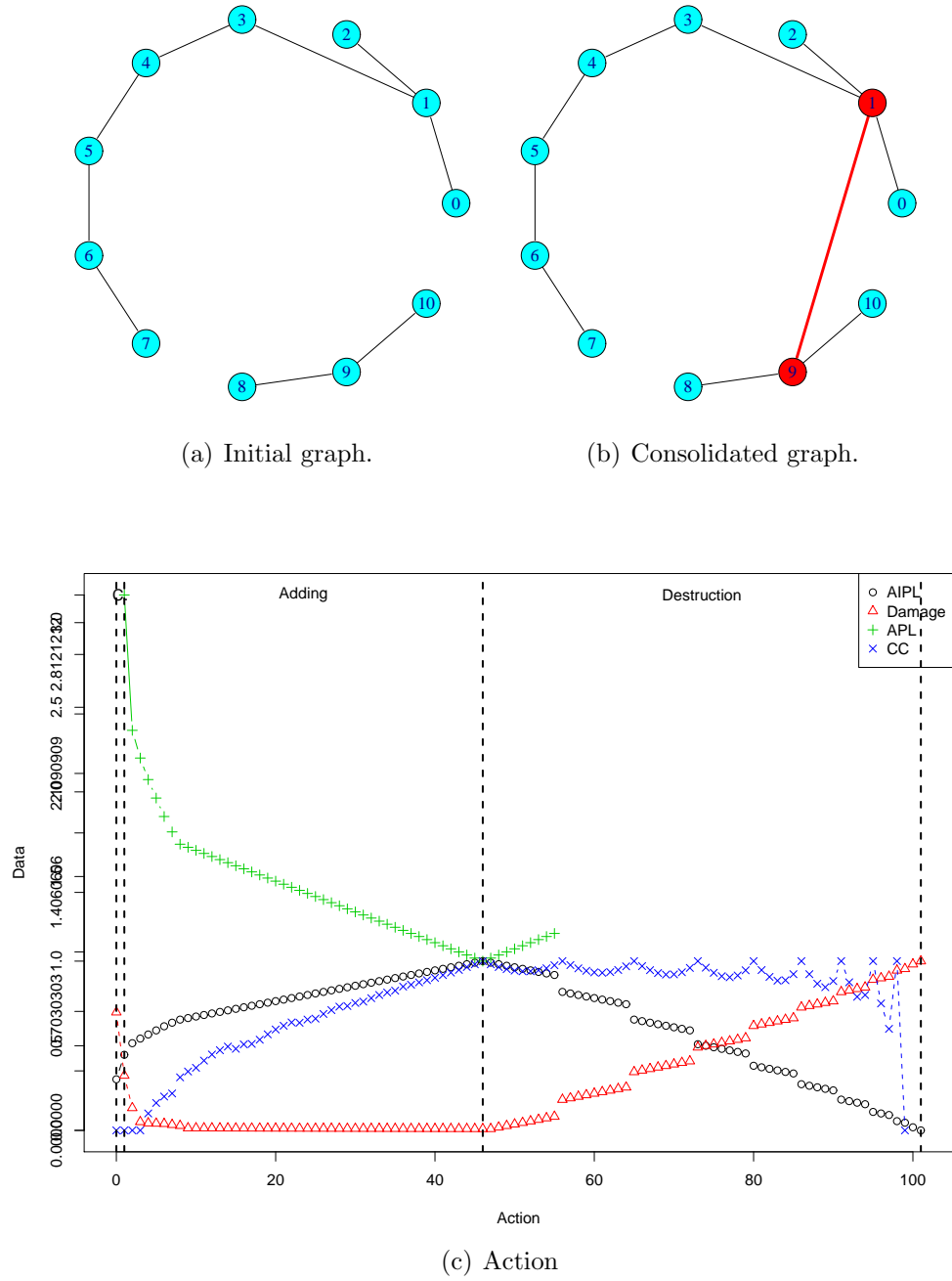


Figure 34. Sample off balance graph. Initially 2 clusters, 11 nodes and 9 edges.

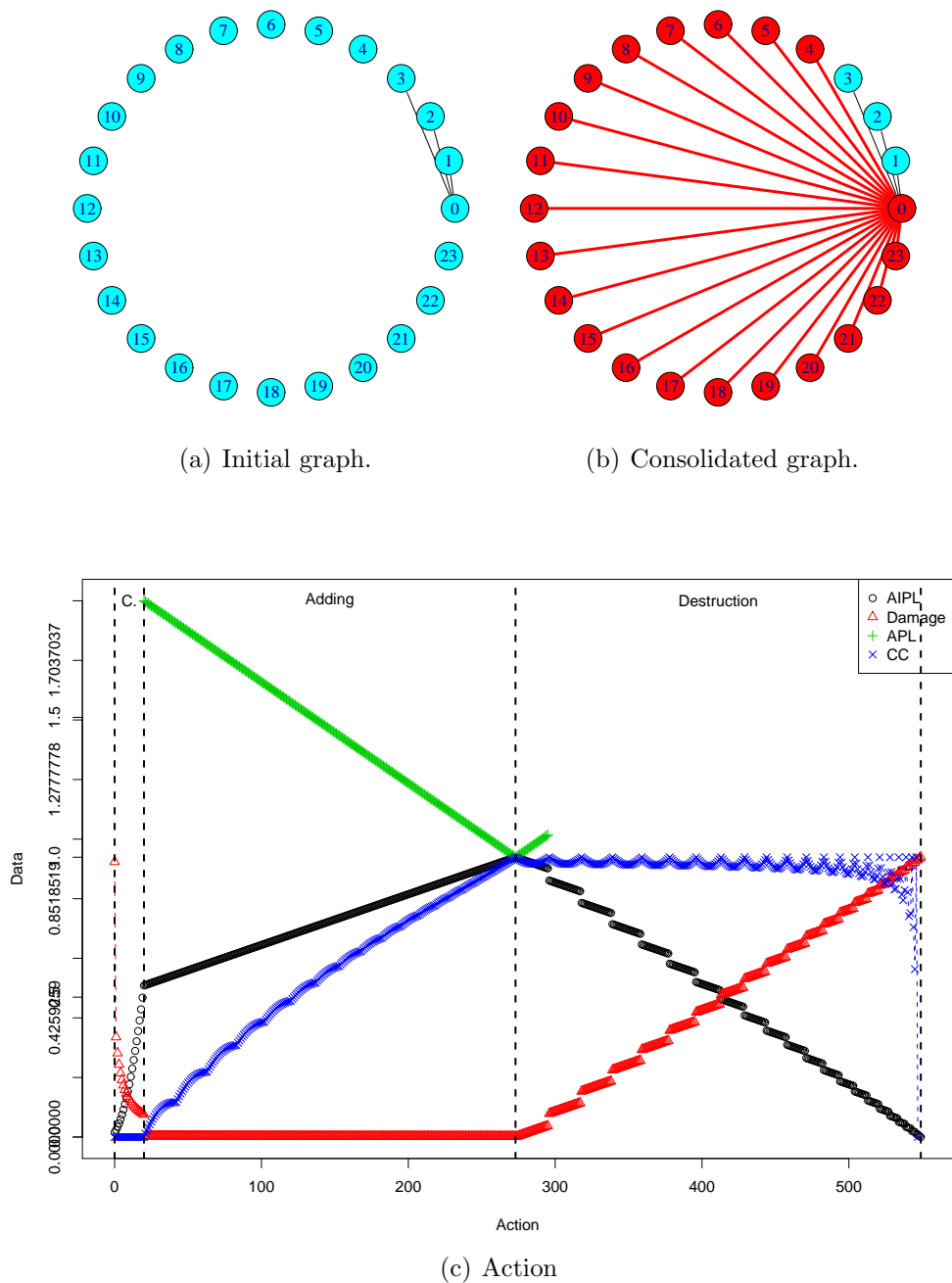


Figure 35. Sample clusters graph. Initially 21 clusters, 24 nodes and 3 edges.

5.3 COMMUNICATIONS

5.3.1 INTRODUCTION

We investigate the problem of sending a message from a sending WO to receiving WO (there may be more than one intended recipient) by looking at the factors that affect the path length and looking that the likelihood that the message will actually be delivered based on the path length.

5.3.2 DISCUSSION OF MESSAGE PATH

During the analysis of the USW graphs, considerable time was spent trying to quantify the number of hops it would take a message to “travel” from any particular WO to all the WOs. This section defines the problem with a hand tractable example, derives the underlying equations and presents a closed form equation to answer the base question:

1. *How many hops are necessary to get from a “root” web object (WO) to reach all WOs in a Unsupervised Small-World (USW) graph?*

Sample problem and definitions

A sample graph (Figure 36 on page 123) will be used to develop equations that can be applied to larger graphs. A small number of symbols are used to describe characteristics of the graph (Table 10 on the next page).

Analysis

The number of WOs at any number of hops from WO# 1 in the sample graph is shown in Table 11 on page 124. Examination of the total summation equation in the table, reveals that is a *geometric series*:

$$n = \sum_{i=0}^m \langle k \rangle^i \quad (30)$$

and that solving for m would be the solution to the problem. The derivation of equation 21 on page 105 shows how to compute the number of hops m needed to

Table 10. A list of symbols used for the analysis of USW communications.

Symbol	Meaning
a	The likelihood that a connection will be made after an unknown number of connection attempts.
e_{new}	The number of new edges added to the USW graph after the wandering WO makes all of its new connections.
j	The number of random number selections required to achieve a .
$\langle k \rangle$	The average degree of all the WOs excluding those WO that are leaf nodes. In the sample graph, WOs whose number is greater than 4 are leaf nodes because no other nodes can be reached from them.
l	The likelihood that a connection will be made based on the current random number and β .
m	The number of constant distance rings required to have a graph of size n where all interior WOs have a constant average degree of $\langle k \rangle$.
n	The number of WOs in the USW.
n_d	The order of the discovered USW graph.
β	The threshold that a random number has to exceed for the newly introduced WO to make its first connection (i.e., as long as the random number is below β , the newly introduced WO will “wander” through the USW graph).
γ	The percentage of WOs that the no longer wandering WO will make connections to.
$\rho(G)$	The of the number edges currently in the graph to the total number there could be.
$ \Gamma_v _{\mathcal{V}}$	The vertices that are directly connected to the WO of interest.

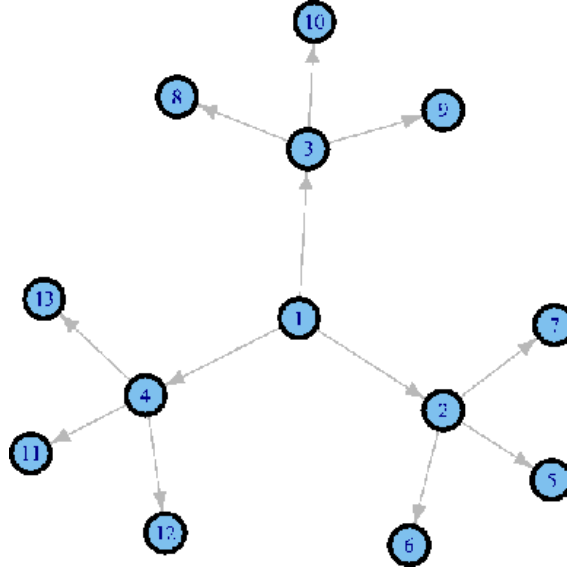


Figure 36. A “toy” graph that is tractable by hand. Node 1 wants to reach all the nodes in the graph. The question is: *how many hops (i.e., how distant) are all the WOs in the graph?*

reach all WOs in a USW graph with n WOs and when the graph has an average degree of $\langle k \rangle$.

$$\sum_{i=0}^m \langle k \rangle^i = \frac{1 - \langle k \rangle^{m+1}}{1 - \langle k \rangle} \quad (31)$$

$$n = \frac{1 - \langle k \rangle^{m+1}}{1 - \langle k \rangle} \quad (32)$$

$$n * (1 - \langle k \rangle) = 1 - \langle k \rangle^{m+1} \quad (33)$$

$$\langle k \rangle^{m+1} = 1 - n * (1 - \langle k \rangle) \quad (34)$$

$$\langle k \rangle^{m+1} = 1 - n + n * \langle k \rangle \quad (35)$$

$$(m + 1) \log(\langle k \rangle) = \log(1 - n + n * \langle k \rangle) \quad (36)$$

$$m = \frac{\log(1 - n + n * \langle k \rangle)}{\log(\langle k \rangle)} - 1 \quad (37)$$

$$numberOfEdges = m - 1 \quad (38)$$

$numberOfEdges$ is the number of edges that a message would have to traverse to go from WO#1 to all WOs in the graph. Once the message was received by the most distant WO, it would not go any further.

Table 11. The number of WOs at selected distances.

Distance(m)	Number WOs(n)	Equation
0	1	$\langle k \rangle^0$
1	3	$\langle k \rangle^1$
2	9	$\langle k \rangle^2$
Total	13	$\sum_{m=0}^2 \langle k \rangle^m$

Other considerations

A number of things can be surmised based on the above analysis and the fundamental questions. They are:

1. *Graph omnipotence*: If n and $\langle k \rangle$ are given, then the entire graph must have been explored.
2. *Graph exploration*: If the graph has been explored, then the average path length for the graph can be computed using:

$$d(u, v) : u, v \in V \quad (39)$$

$$d(u, v) = |E(P)|, E(P) = \{u_0u_1, u_1u_2, \dots, v_{-1}v_0\} \quad (40)$$

$$L(G) = \frac{1}{n * (n - 1)} * \sum_{i,j}^n d(v_i, v_j) \quad (41)$$

3. *Average path length*: Is a more realistic value for how many hops are required than *numberOfEdges* from Equation 38 on the preceding page.
4. *Distances in a USW graph*: Distances between WOs in a USW graph are independent of how the USW graph is constructed.
5. *Availability of graph related information*: If n and $\langle k \rangle$ are given, then why not all the information about the entire graph? If the label for each node in the USW graph was its canonical name, then all information would be available and the basic problem becomes trivial.

The WO's name is sufficient for the underlying Internet architecture to make the distance between any WOs a constant 1.

A different approach

The text and equations in the previous sections are correct in as far as they go. Based on further analysis, it is possible that there are other approaches that are as equally valid.

The following sections approach the problem starting with posing slightly different questions, followed by how the Unsupervised Small-World is constructed, how Unsupervised Small-World WOs could be selected by an external user or agent, how the Unsupervised Small-World WOs could communicate messages from one to another, and concludes with an evaluation of the different scenarios.

Posing a different question

The original question was:

1. *How many hops are necessary to get from a “root” web object (WO) to reach all WOs in a Unsupervised Small-World (USW) graph?*

and could be open to interpretation. So it has been replaced by the following questions:

1. *How many hops are necessary to get a singular message from a “root” WO to all WOs in a Unsupervised Small-World (USW) graph?*
2. *How many hops are necessary to get the same message from a “root” WO to all WOs in a Unsupervised Small-World (USW) graph?*

The first replacement question speaks to a “unicast” message from one sender to one receiver. The second question speaks to a “broadcast” message from one sender to all WOs. In fact, the second replacement question is a special case of the first question. The time to send a message to the most remote WO (there are many different ways to define what is remote) will always be less than the time to send the same message to those WOs that are less remote. We will focus on solving the first question.

Unsupervised Small-World construction

The construction of a USW graph is dependent on the control parameters β and γ . As the new WO wanders through the USW graph, it collects information about

the USW structure. After making its first connection, the WO will make connections to γ of the USW WOs that it has discovered. Based on these definitions:

$$l = 1 - \beta \quad (42)$$

$$a = 1 - \beta^j \quad (43)$$

$$\beta^j = 1 - a \quad (44)$$

$$j \log(\beta) = \log(1 - a) \quad (45)$$

$$j = \frac{\log(1 - a)}{\log(\beta)} \quad (46)$$

$$n_d = j * \langle k \rangle \quad (47)$$

$$e_{new} = \gamma * n_d + 2 \quad (48)$$

When the wandering node is connected into the USW graph, the USW graph has grown by 1 WO and e_{new} edges. The factor of two in equation 48 accounts for the two edges added when the wandering WO made its first connection.

The density of a graph $\rho(G) = \frac{m}{\binom{n}{2}}$ [167] can be used as a surrogate for $\langle k \rangle$. $\langle k \rangle$ is an important descriptor of the USW graph and has a profound impact on the how much of the graph is known by any single WO. Therefore the question of how e_{new} affects $\rho(G)$ is important. (In the interest of clarity, $|E|$ will be used vice m for the following equations.)

$$\rho(G)_{Old} \stackrel{?}{=} \rho(G)_{New} \quad (49)$$

$$\frac{|E|}{n(n-1)} \stackrel{?}{=} \frac{|E| + e_{new}}{(n+1)n} \quad (50)$$

$$|E|(n^2 + n) \stackrel{?}{=} (|E| + e_{new})(n)(n-1) \quad (51)$$

$$|E|n^2 + |E|n \stackrel{?}{=} |E|n^2 + e_{new}n^2 - n|E| - ne_{new} \quad (52)$$

$$|E|n \stackrel{?}{=} e_{new}n^2 - n|E| - ne_{new} \quad (53)$$

$$|E| \stackrel{?}{=} e_{new}n - |E| - e_{new} \quad (54)$$

$$2|E| \stackrel{?}{=} e_{new}n - e_{new} \quad (55)$$

$$2|E| \stackrel{?}{=} e_{new}(n-1) \quad (56)$$

$$\frac{2|E|}{(n-1)} = e_{new} \quad (57)$$

Based on equation 57, the following prediction can be made about $\rho(G)$:

$$\rho(G)_{New} will \begin{cases} \text{remain the same} & \text{if } e_{new} = \frac{2|E|}{(n-1)} \\ \text{decrease} & \text{if } e_{new} < \frac{2|E|}{(n-1)} \\ \text{increase} & \text{if } e_{new} > \frac{2|E|}{(n-1)} \end{cases} \quad (58)$$

As $\rho(G)$ goes, so goes $\langle k \rangle$.

Unsupervised Small-World WO selection options

When a WO is activated, or selected by someone (or some agent) browsing the Web, the WO engages in a series of maintenance activities. Two of these activities are to read messages and to send messages. For this discussion, we focus on the sending of messages and assume that the WO acts appropriately on whatever messages it reads.

A WO can be selected based on one of these conditions:

1. *Uniform random selection:* Every WO has exactly the same likelihood of being selected as every other WO.

$$\text{success} = \frac{1}{n} \quad (59)$$

$$a = 1 - \left(\frac{n-1}{n}\right)^j \quad (60)$$

$$j = \frac{\log(1-a)}{\log \frac{n-1}{n}} \quad (61)$$

Based on equation 61, the larger graph becomes, the more number of random selections it will take to reach a predefined expectation of success.

2. *Degree based selection:* A vector is created which contains each WO its degree $d(v) = k$ number of times. Meaning that if a WO has a degree of 4, then it would be in the vector 4 times. Therefore,

$$\text{success} = \frac{\min(d)}{\sum d} \quad (62)$$

$$a = 1 - \left(\frac{\sum d - \min(d)}{\sum d}\right)^j \quad (63)$$

$$j = \frac{\log(1-a)}{\log \frac{\sum d - \min(d)}{\sum d}} \quad (64)$$

Based on equation 64 on the preceding page, the more unbalanced the degree distribution of the graph, the more degree biased selections will be needed to select the least connected WO.

3. *Age based selection:* A vector is created which contains each WO a number of times based on its age. If the newest WO is normalized to one, the oldest will be greater than one. If the oldest is normalized to one, then the newest will be greater than one. Therefore, the formulation for computing j is exactly the same as the technique used in equation 64 on the previous page, only substituting one for $\min(d)$ and $\sum age$ for $\sum d$. As the graph ages, it will take more and more selections to select the WOs with an age of one.
4. *Popularity:* Some WOs are more “popular” than others. The definition of “popular” is open for discussion, but it will rank order all the WOs to some standard. Using that standard and making the least “popular” WO as a one, then the same formulation used in equation 64 on the preceding page is applicable, substituting one for $\min(d)$ and $\sum popularity$ for $\sum d$.

Unsupervised Small-World WO communication mechanisms

There are three basic communications mechanisms, gossip based [173], store-and-forward and bus style. Gossip is a variation on store-and-forward and will be considered as part of the store-and-forward discussion. In the store and forward technique, a message is received and if it is not addressed to the recipient, it is forwarded on to another intermediary recipient. In the bus technique, a signal is sent out that a message is coming, everyone on the bus listens and if the message is addressed to them then they take an action.

These data fields are assumed to be in the message, they may or may not be used based on the technique being discussed:

1. *To:* The intended recipient.
2. *From:* Who originated the message.
3. *Via:* Who has seen the message so far.
4. *Time to live (TTL):* How many WOs have seen this message.

Store and forward (e-mail style communications)

These are the different store-and-forward techniques that could be employed:

1. *Randomly forward*: All fields in the message (other than To) are ignored. The next single recipient of the message is selected at random from all the $|\Gamma_v|_{\mathcal{V}}$. Messages sent using this technique will take $\theta(n^3)$ [174] attempts to reach the intended recipient.
2. *Forward to everyone*: All fields in the message (other than To and TTL) are ignored. Each time the message is received, the TTL field is decremented. If the TTL reaches 0, the message is silently discarded. Otherwise, the message is forwarded to everyone in the $|\Gamma_v|_{\mathcal{V}}$. This flooding will occur at each node until the TTL reaches 0 and the number of messages in circulation at any time will be $\text{TTL} * \langle k \rangle$. If the TTL is not large enough, the message may never be received.
3. *Forward to others that have not seen the message*: The From and Via fields of the message are examined to see if they are not in the $|\Gamma_v|_{\mathcal{V}}$ of the current recipient. If the TTL is not 0, then the message is forwarded to all who have not seen the message yet. The number of messages in circulation will be less than $\text{TTL} * \langle k \rangle$ and messages may die sooner than the TTL based on not being forwarded back to WOs on the Via list.

As can be seen in Table 12 on page 132, the number of selections necessary to achieve a 0.95 confidence of the least likely WO being selected can vary considerably based on the selection criteria. The number of selections required to ensure that a message is received by the “lowest evaluated” recipient is the product of the communications mechanism and the recipient selection criteria.

The representative selection models were evaluated to determine how many selections would be necessary to reach a 0.95% probability of reaching the “least” likely WO (Figure 37 on the following page). The vertical green line is 0.95% and each of the horizontal green lines indicates the number of selections required to reach the acceptable level. All of the curves in the figure have the same basic shape, only offset vertically from each other.

The cumulative probability distribution (CPD) for the representative selection models is show in Figure 38 on page 131. The figure shows two plots of the same

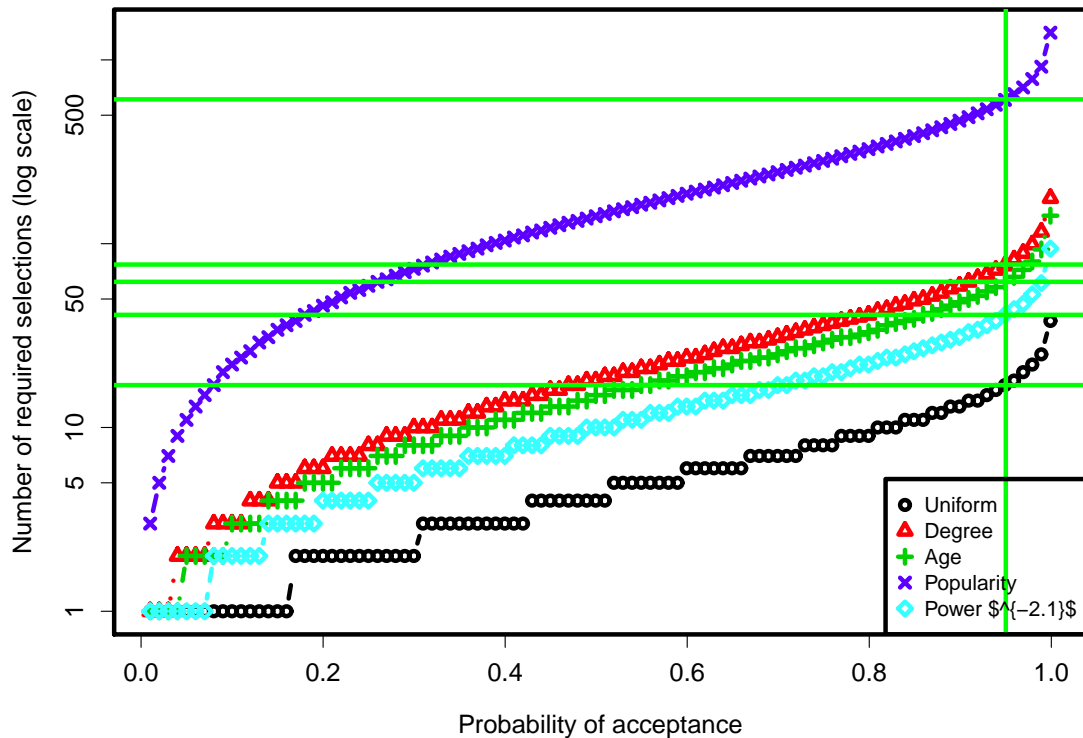


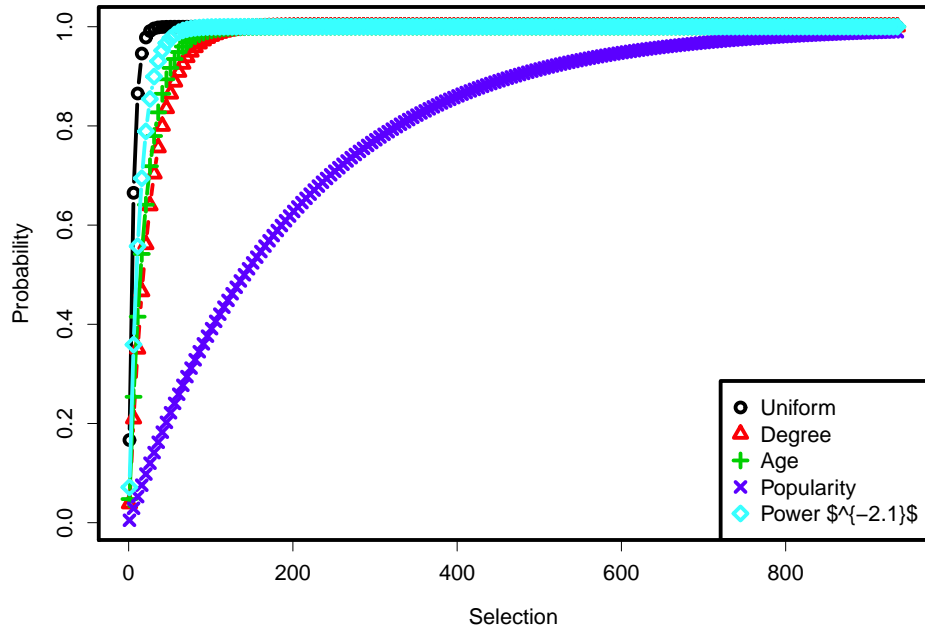
Figure 37. The number of selections required to reach any particular a .

data. As seen in Figure 38(a) on the following page, the curves for all but the Popularity selection are almost stacked on top of one another. In Figure 38(b) on the next page, the X-axis is a log scale in order to show the activity in the lower region.

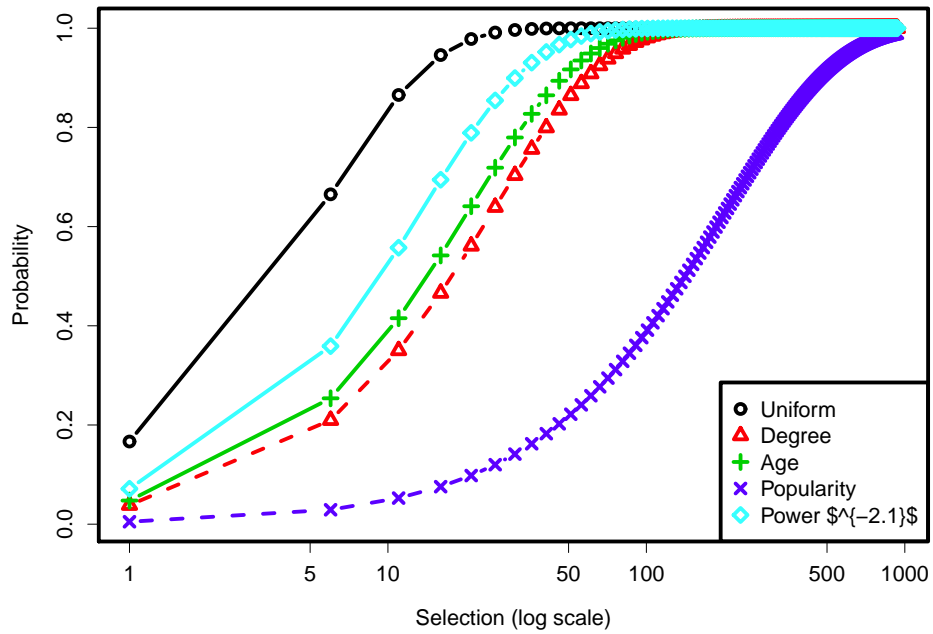
A subjective evaluation of the performance of all the a communications techniques based on the WO selection criteria is shown in Table 13 on page 132.

Bus

Using a bus style communications, the time for a message to go from the sender intended recipient after getting control of the bus, is 1. Getting control of the bus will be dependent on n due to contention for control. If the WOs are too talky this should not be an issue.



(a) Normal CPD



(b) Log CPD

Figure 38. The cumulative probability distribution (CPD) for the selection process.

Table 12. Computing the number of selections needed to achieve 0.95 likelihood of a message being received by all WOs based on different WO selection criteria.

Item	Uniform	Degree	Age	Popularity	Power ^{-2.1}
WO-1	1	2	1	1	1
WO-2	1	10	2	1	1
WO-3	1	10	3	1	1
WO-4	1	10	4	1	1
WO-5	1	10	5	100	5
WO-6	1	10	6	100	5
min	1	2	1	1	1
\sum	6	52	21	204	14
j	17	77	62	610	41

Table 13. Subjective evaluation of communications mechanisms and WO selection criteria.

Comms. mechanism	Perf.	Selection Criteria			
		Uniform	Degree	Age	Popularity
Randomly forward	$\theta(n^3)$	Bad	Bad	Bad	Worst
Forward to everyone	$\theta(n^2)$	N/A	N/A	N/A	N/A
Forward to other that have not seen the mes- sage	$\theta(n^2)$	Bad	Bad	Bad	Best
Bus	$\theta(1)$	Best	Better	Better	Better
Tuple based	$\theta(1)$	Best	Better	Better	Better

Tuple based

Carriero and Gelernter in [65, 175, 176] provide an explanation and an overview of the Linda communication model as implemented in various languages and for representative applications. Linda's communication model is summarized in:

“If two processes need to communicate, they do not exchange a variable; instead, the data producing process generates a new data object (called a tuple) and sets it adrift in a region called tuple space.”

Nicholas Carriero and David Gelernter [65]

Asynchronously the sender writes a message into the *tuple space* (perhaps) without specifically identifying the intended recipient. Instead, attached to each message are a set of “characteristics” of the intended recipient. A candidate recipient queries the tuple space to retrieve any messages there that match the recipient's characteristics and processes the messages returned by the query. Using a bus style communications, the time for a message to go from the sender intended recipient after the message is in tuple space is 1. The size of the tuple space will affect query time.

The number of “hops” that a message must take to reach all WOs in a Unsupervised Small-World (USW) graph has been computed. But another question has been raised:

1. *How long will it take for the same message from a “root” WO to reach a given percentage of all WOs in a Unsupervised Small-World (USW) graph?*

The ultimate answer depends on the number of users/agents accessing the graph per unit time, and the number of accesses required for 95% assurance that all USW WOs have received the message will be between $\theta(1)$ and $\theta(n^3)$. The number of users/agents and how active they are per unit time will determine how long it takes for a message to be received by a particular WO.

Summary

Equation 38 on page 123 shows how to compute the number of hops m needed to reach all WOs in a USW graph with n WOs and when the graph has an average degree of $\langle k \rangle$. The equation will work for any sized graph, but is very sensitive to

all WOs having the same $\langle k \rangle$. Table 14 on the following page shows the number of uniform random selections necessary to reach different CPD based on graph order. The values come from the USW simulator.

The number of USW WO selections necessary to ensure that the WO ranking lowest based on the selection criteria has the form $\frac{\log(1-a)}{\log \frac{\sum \text{criteria} - \min(\text{criterion})}{\sum \text{criteria}}}$, where a is the acceptance level. The selection criteria (uniform random, age, degree, popularity, etc.) was a significant impact on the number of selections needed to reach the acceptance level. The effectiveness of the communications mechanism is a multiplicative factor in the number of selections that are necessary. Communications mechanisms can operate in the range of $\theta(1)$ to $\theta(n^3)$.

WO selection based on uniform random selection using bus, or tuple space style communications results in the fewest number of WOs needed to be selected to reach any given acceptance threshold.

With an estimate of the number of WOs that must be selected to achieve the acceptance threshold and a estimate of the number of WOs that will be accessed per unit time; the length of time to reach the acceptance threshold is a simple division. The simplest technique to reduce the time for a message to be received by the intended recipient is to have an agent that constantly activates the USW WOs.

Table 14. A comparison and analysis of the number of selections needed to reach a level of confidence versus the order of the graph based on uniform random selection.

Order	0.80	Ratio	0.85	Ratio	0.90	Ratio	0.95	Ratio
200	32,349	162	38,132	191	46,281	231	60,213	301
75190	4,549,567,472	60,508	5,362,788,653	71,323	6,508,959,532	86,567	8,468,351,591	112,626
150180	18,149,767,646	120,853	21,393,982,742	142,456	25,966,447,106	172,902	33,783,126,565	224,951
225170	40,800,695,651	181,200	48,093,694,400	213,588	58,372,598,820	259,238	75,944,501,990	337,276
300160	72,502,215,425	241,545	85,461,763,246	284,721	103,727,220,017	345,573	134,952,224,608	449,601
375150	113,254,124,730	301,890	133,497,950,890	355,852	162,030,021,356	431,907	210,805,917,981	561,924
450140	163,058,219,713	362,239	192,204,374,539	426,988	233,283,572,545	518,247	303,508,925,376	674,255
525130	221,910,540,450	422,582	261,576,366,439	498,117	317,482,208,210	604,578	413,053,875,971	786,575
600120	289,814,633,455	482,928	341,618,107,037	569,250	414,631,002,270	690,913	539,447,371,086	898,899
675110	366,768,575,975	543,272	432,327,329,892	640,381	524,727,203,885	777,247	682,685,831,796	1,011,222
750100	452,775,961,690	603,621	533,708,271,044	711,516	647,775,954,448	863,586	842,775,947,207	1,123,551
825090	547,824,350,594	663,957	645,746,266,873	782,637	783,759,456,336	949,908	1,019,694,562,079	1,235,859
900080	651,939,555,919	724,313	768,471,744,649	853,782	932,714,639,934	1,036,257	1,213,489,723,949	1,348,202
975070	765,109,408,636	784,671	901,870,360,163	924,929	1,094,624,095,296	1,122,611	1,424,138,781,957	1,460,550
1050060	887,289,017,347	844,989	1,045,889,197,822	996,028	1,269,423,597,355	1,208,906	1,651,558,177,362	1,572,823
1125050	1,018,586,844,114	905,370	1,200,656,106,943	1,067,202	1,457,268,320,235	1,295,292	1,895,949,796,356	1,685,214
1200040	1,158,887,837,991	965,708	1,366,035,471,581	1,138,325	1,657,993,663,249	1,381,615	2,157,099,488,508	1,797,523
1275030	1,308,232,827,852	1,026,041	1,542,075,418,646	1,209,442	1,871,658,039,309	1,467,933	2,435,083,250,767	1,909,824
1350020	1,466,666,123,576	1,086,403	1,728,828,178,269	1,280,594	2,098,324,841,520	1,554,292	2,729,983,559,465	2,022,180
1425010	1,634,148,119,200	1,146,763	1,926,247,065,045	1,351,743	2,337,937,406,559	1,640,646	3,041,726,693,918	2,134,530
1500000	1,810,707,964,705	1,207,139	2,134,366,439,422	1,422,911	2,590,537,438,620	1,727,025	3,370,366,912,534	2,246,911

5.3.3 DISCUSSION OF THE LIKELIHOOD THAT A MESSAGE WILL BE RECEIVED.

A fundamental way to measure the performance of any communication system is: How long will it take for the same message from a “root” WO to reach a given percentage of all WOs in a Unsupervised Small-World (USW) graph?

This section delves into answering that question. A number of different sized USW graphs were created, with different values of β and γ and three different USW graph stimulation policies. Message delivery to all WO is based on the degree distribution of the graph and equation that predicts the probability that a particular WO will receive the message at any particular time is derived.

Analysis

The analysis is based on running the USW simulator with varying graph sizes, varying values for β and γ and varying how an outside entity would “ping” a WO within the the graph. The graph was allowed to be created as determined by β and γ before a message was introduced.

Three different “ping” selection policies were simulated. They are:

- *Sequential*: every WO in the graph was selected. This served as a baseline to assist in the evaluation of the other policies.
- *Random*: a WO was selected at random from all the WOs in the graph.
- *Degree biased*: the WO as selected partially on its degree k .

In order to be able to select a WO to “ping” based on its degree, a degree distribution set of the entire graph had to be created. Along with the degree set, another data structure maintained a list of WOs based on their degree. After the set was created, a vector was created that had the same number of entries per degree as the degree from the set. If the set looked like:

$$\{3, 6, 9\}$$

Then the vector would look like:

$$333666666999999999$$

A member of the vector was chosen in a uniform random manner. The degree that was chosen was used as a pointer to the list of WOs that had that degree. From the list of WOs that all have the chosen degree, one WO was chosen in a uniform random manner.

WO #0 was selected to send a message to all WOs. When a WO was “pinged” it would do its normal housekeeping and check for Linda style messages. The “pinged” WO would check to see if it had received the message before and would print a log message stating that this was the first time it had received the message, or if this was an old message. The log message stating this was a new message was captured for the analysis.

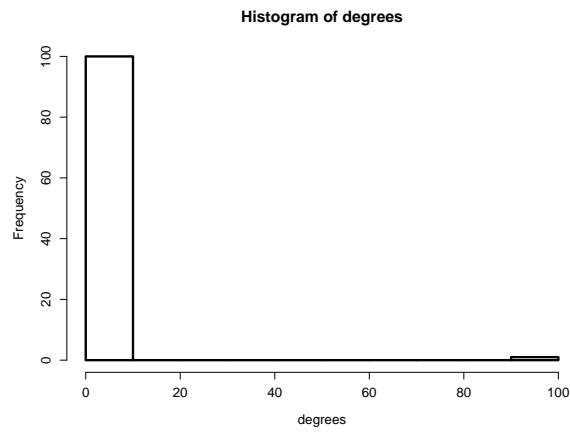
Initial testing of the algorithm showed that it was nearly impossible for all the WOs to be pinged without some duplicate pings and that the entire selection process would have to be repeated a number of times to raise the likelihood that all WOs would have a chance to be “pinged.” Each of these iterations is called an epoch in recognition that there is considerable time involved in pinging all the WOs in the graph. The pinging process runs for 20 epochs.

The simulator was run and the number of epochs needed to for all the WOs to acknowledge they had received the message is summarized in Table 15 on page 139. As expected using the Sequential approach has all WOs receiving the message in 1 epoch. Random selection was able to reach all WOs in about 10 epochs under all β and γ values. Degree biased selection quickly fell apart, in many cases requiring greater than 20 epochs. (Closer analysis may show that those cases where 19 epochs were reported, may also really need more than 20.)

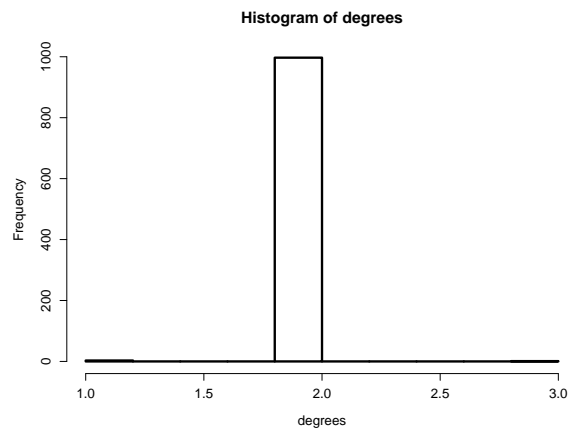
Looking at the cases where the WOs were not able to receive the message in 20 epochs, a common characteristic came into view. The degree distribution was heavily skewed into a long head or a long tail (Figure 132 on page 392). Where the degree distribution was not heavily skewed, the message was delivered to all WOs within 20 epochs.

A complete set of degree distribution and delivery times were computed (Appendix G on page 452).

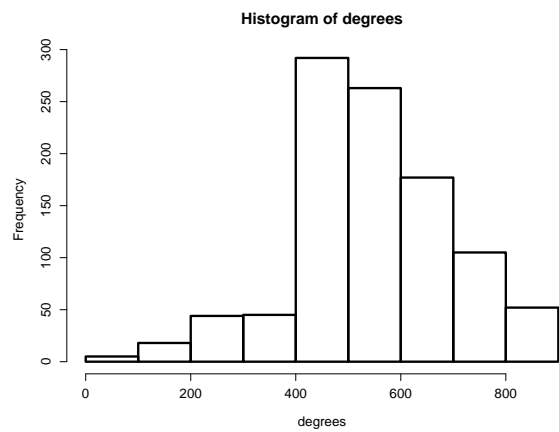
Probabilistic analysis We investigate the likelihood that a message will be delivered as a binomial distribution probabilistic problem with slight modifications. Classic binomial distribution meets the following assumptions:



(a) Long tail (bad)



(b) Long head (bad)



(c) Majority about the same (OK)

Figure 39. A comparison of different degree distributions and an assessment of how suitable the graph is for message distribution.

Table 15. Summary of the number of epochs needed to get the sample message to all WOs.

Selection method	β	γ	Graph size		
			100	500	1000
Sequential	0.0	0.0	1	1	1
	0.0	0.5	1	1	1
	0.0	1.0	1	1	1
	0.5	0.0	1	1	1
	0.5	0.5	1	1	1
	0.5	1.0	1	1	1
	1.0	0.0	1	1	1
	1.0	0.5	1	1	1
	1.0	1.0	1	1	1
Random	0.0	0.0	6	9	10
	0.0	0.5	8	7	9
	0.0	1.0	5	8	7
	0.5	0.0	6	9	10
	0.5	0.5	5	7	8
	0.5	1.0	9	7	8
	1.0	0.0	6	9	10
	1.0	0.5	8	6	8
	1.0	1.0	5	9	7
Degree biased	0.0	0.0	>20	>20	>20
	0.0	0.5	14	20	19
	0.0	1.0	11	14	19
	0.5	0.0	>20	>20	>20
	0.5	0.5	7	14	15
	0.5	1.0	8	19	19
	1.0	0.0	7	9	19
	1.0	0.5	17	15	17
	1.0	1.0	>20	>20	>20

1. There are only two possible outcomes for each trial,
2. The probability of success is the same for each trial,
3. There are n trials, where n is a constant,
4. The n trials are independent.

The classic binomial distribution [177] is described by:

$$b(x; n, p) = \binom{n}{k} p^x (1 - p)^{n-x} \quad (65)$$

Equation 65 returns the likelihood that there are x “successes” out of n trials, the likelihood of a trial succeeding is p .

We are interested in applying the binomial distribution under two different scenarios. The scenarios are:

1. A message is sent to a single member of a “family” simulating that a sending WO wants to convey a message to a member of another family and then has no interest in the message there after.
2. A message is sent to all “family” members simulating that an active maintainer WO has information that all family members need to know.

To address these scenarios, the classic binomial distribution assumptions are modified to:

1. There are only two possible outcomes for each trial,
2. The probability of success is the same for each trial,
3. ~~There are n trials, where n is a constant,~~ for our analysis, we will be computing n , therefore;
4. The n trials are independent, and
5. A probability of success c is given

To see the effect of these assumptions, we create a small USW graph with the following characteristics:

1. There are nine non-participating WOs,

Table 16. Number of trials to send a message. The number of trials to remains constant for families B and C for both scenarios because reception by one of member of each family is sufficient for the message to be passed along to the next member in the path. The number of trials is marked different for the receiving WO because in the first scenario only one WO has to receive the message, but in the second there is at least a c likelihood that all members of the receiving family gets the message.

	Scenario 1	Scenario 2
WO	n	n
B	8	8
C	11	11
Recv.	14	84,948

2. There is one sending WO,
3. The sending WO has a friendship link to the B family,
4. The B family has a friendship link to the C family,
5. The C family has friendship links with all members of the intended receiving family.

The store and forward message path is:

$$\text{Sending WO} \rightarrow \text{B} \rightarrow \text{C} \rightarrow \text{Receiving family}$$

The desired probability of success is 0.95.

The number of trials to meet c is shown in Table 16.

Summary

Based on data from the USW simulator, the probability that a particular WO will receive a message is:

$$P(WO) = \frac{1}{|WO_k|} * \frac{k_{WO}}{\sum k_{WO}}$$

The number of “pings” that might be required to ensure that all WOs get the message, is at least $\frac{1}{P(WO)}$ and if $P(WO)$ is small, some multiple of the least number of pings. Because selection of WO to ping is a random function, there is no guarantee

that all WOs will in fact be pinged, regardless of how many epochs pass. All that can be hoped for is that the probability of not being “pinged” is acceptably low.

Degree Distribution and Message Delivery Times

Figures 169 on page 453 through 204 on page 488 show a collection of data for a USW graph of order 1000. Data shown include:

1. *Degree distribution*: the various combinations of β and γ result in very different degree distributions. The distribution provides insight into the other plots.
2. *Sequential*: the number of epochs necessary for 100% of the WOs to receive the message if the WOs are accessed sequentially. This plot is always a straight line, but it is included for completeness.
3. *Random*: the number of epochs necessary for 100% of the WO to receive the message if the WOs are accessed randomly. The curve will asymptotically approach 100%.
4. *Degree biased selection*: the WOs with higher degrees k are preferentially selected over those with lessor degrees. Some WOs will never get the message because their k is too low relative to other WOs.

5.3.4 SUMMARY

These general statements can be made:

1. Sequentially accessing all WOs will ensure that the message is received in one epoch. (Trivial case.)
2. Extremely skewed degree distributions will ensure that at least some of the WOs will never get the message. (Pathological case.)

Between these two extremes, if the system were given enough time then all WOs would get the message. It becomes a design decision as to how much energy must be put into the system to reach an acceptable level of likelihood that enough WOs have received the message.

5.4 COPYING

Discussion

Unsupervised Small-World does not attempt to make a “bit-by-bit” copy of the original resource [142]. Instead, the resource that will be preserved is examined by an entity outside the USW environment to identify those components that comprise the essence of the resource. A REsource Map (REM) [133] is created to capture the identified components. A crucial part of the REM is the list of aggregated resources that could be used to recreate the essence of the original resource.

Aggregated resources fall into different categories (Table 17 on page 145). Each category will be treated differently by the USW “copying” process.

The size of the resource (or more precisely the data associated with the resource), also impacts the preservation method. Notionally, the following resource data size relationships exist:

Tiny << Medium << Large << Huge.

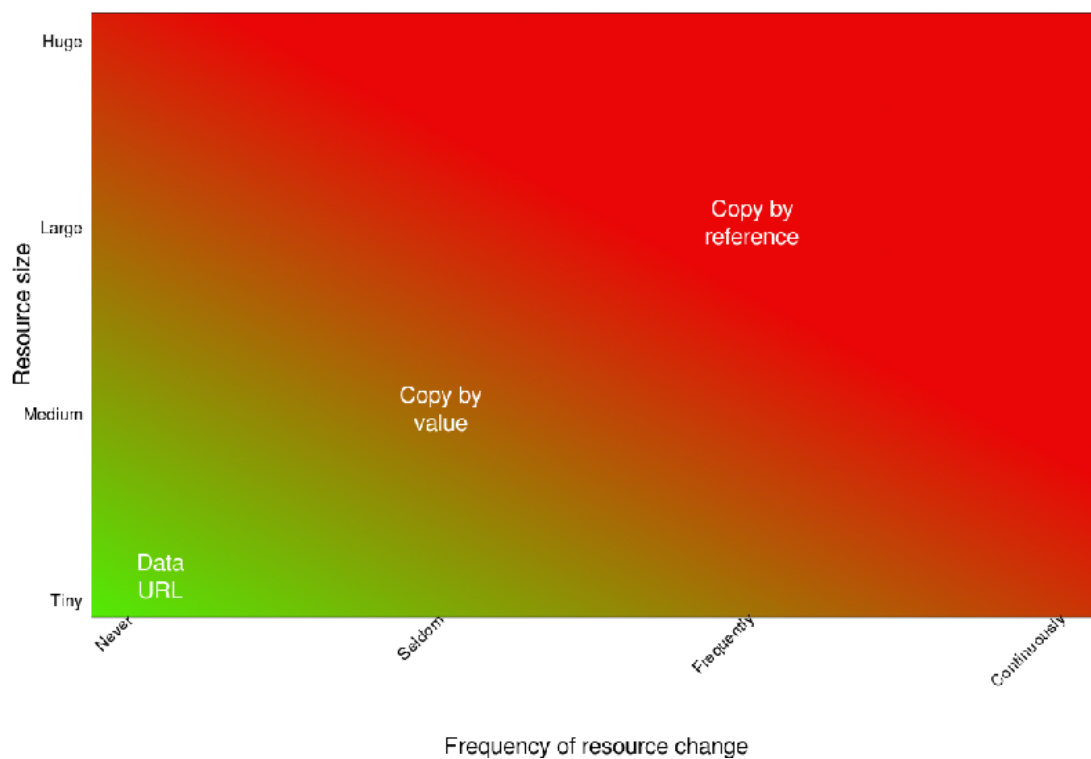


Figure 40. Preservation copy guidelines. Green means that data from the resource is safe for the life of the WO. Red means that data is at grave risk and may be vulnerable to loss at any time. A continuum exists between green and red. The preservation copy guideline to be followed will be stated by originating REM creator. Subsequent ReM modifiers will always be able to enforce a more conservative approach, but should never adopt a looser approach.

Table 17. Aggregated data categories based on an expected update rate. Each category will be treated differently during the USW copying process.

Freq. of Update	Example	How treated	Preservation Impact
<i>Continuously</i>	Real time streaming video from a camera	The original URI is maintained. Copy by Reference	The aggregated resource will be available only as long as the original URI is valid.
<i>Frequently</i>	Number of passengers on a mass transit bus	The original URI is maintained or the resource is copied to a new location. Copy by Reference or Copy by Value	If the original is maintained then loss of the single resource is catastrophic. If the resource is copied then the resource is likely to be different than the original and it is more likely to survive.
<i>Seldom</i>	A CV	The resource is copied to a new location. Copy by Value	If the original is maintained then loss of the single resource is catastrophic. If the resource is copied then the resource is likely to be the same as the original and it is more likely to survive.
<i>Never</i>	A digital signature	These internalized data are copied intact and without modification as part of the copied resource to the new host. Data URI	As long as the copied resource is available, then the internalized data will be available.

Absolute values for each of the terms is almost meaningless as something that is considered as large today might be considered only medium in the not too distant future.

The “preservation space” is a combination of the orthogonal aspects of resource size and resource change rate (Figure 40 on page 144 and Table 18 on the following page).

To capture the attributes of size and method of preservation, a single identifier format will be used:

`copy_{size}_{frequencyOfUpdate}`

Resulting in the following copying and preservation identifiers:

- *copy_tiny_never*
- *copy_medium_never*
- *copy_large_never*
- *copy_huge_never*
- *copy_tiny_seldom*
- *copy_medium_seldom*
- *copy_large_seldom*
- *copy_huge_seldom*
- *copy_tiny_frequently*
- *copy_medium_frequently*
- *copy_large_frequently*
- *copy_huge_frequently*
- *copy_tiny_continuously*
- *copy_medium_continuously*
- *copy_large_continuously*
- *copy_huge_continuously*

Table 18. Recommended copying actions based on resource size and update frequency.

		Update frequency			
		Never	Seldom	Frequently	Continuously
Size	Tiny	Data URI	Copy by Value	Copy by Reference or Copy by Value	Copy by Reference
	Medium	Copy by Value	Copy by Reference or Copy by Value	Copy by Reference	Copy by Reference
	Large	Copy by Reference or Copy by Value	Copy by Reference	Copy by Reference	Copy by Reference
	Huge	Copy by Reference	Copy by Reference	Copy by Reference	Copy by Reference

Summary

WOs can ensure that preservation copies are spread across unique hosts by using unique copy services and assuming that unique services, in fact, copy to unique places. By maintaining a list of copy services and the copy service that was used to create a new copy, when a new copy is needed then it is easy to identify a copy service that has not been used already and request that a new copy be made there.

Due to the nature of asynchronous communication between the Copy Requesting WO and the Copy Servicing WO, it is possible that several Requests could be sent resulting in one or more Copy Location messages.

5.5 ATTACK VERSUS FAILURE

5.5.1 EXPLANATION

Errors and attacks remove components from a system. The distinguishing characteristic between the two types of losses is how components are selected. This characteristic can be explained by using a computer network as a graph. The network is a graph where vertices are represented by routers, switches and computers, while edges are represented by the connections between the vertices, either wired or wireless connections.

The loss of a router through hardware failure, mis-configuration, or the severing of the communications links to the router can be considered to be accidental. An error is the accidental loss of a component from a system. The simultaneous loss of a set of routers, perhaps without a readily apparent reason, could be considered to be an attack. An attack is the deliberate loss of components, or a component, from a system.

The survivability of a graph to error or attack depends on the underlying structure of the graph (for example scale-free or exponential). Scale-free graphs are very robust in the face of random failures, but are very susceptible to attacks [73]. Where exponential graphs have just the opposite behavior.

5.5.2 SELECTION OF GRAPH COMPONENT TO ATTACK

The components that an attacker chooses to remove from an existing graph can be called an attack profile. Each of the different attack profiles is presented with

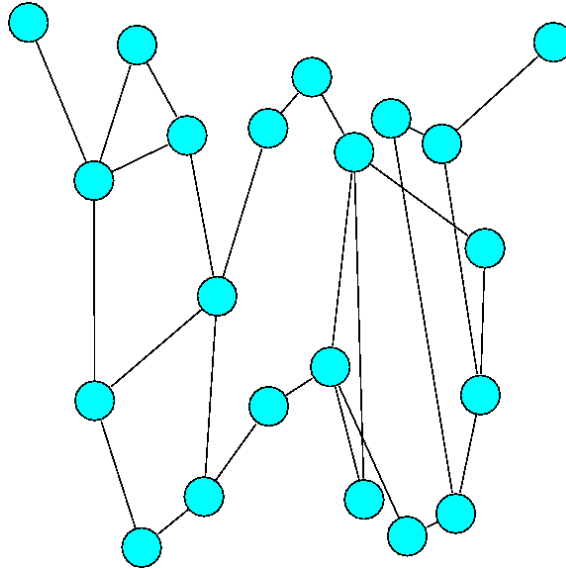


Figure 41. The sample graph presented to each attack profile. This is graph is small enough that it can be visualized easily, characteristics can be computed manually and yet has interesting features.

the same graph (Figure 41). The attack profile continues to execute until the graph is disconnected. In those cases where there are multiple graph components with the same value (vertices of the same degree, edges with the same betweenness, etc.), the attack profile is recursively applied and the total number of deletions is reported. Figures 42 on the following page and 43 on page 151 show the sample graph prior to the deletion of the first attack profile specific element. Each attack profile assumes that the attacker has complete (i.e., global) knowledge of the graph and so is able to make decisions that are most advantageous to the attacker. How this knowledge is obtained is outside this discussion. The goal of each attack profile is the disconnection of the graph, where disconnection is defined as no path exists from vertex i to vertex j $i \not\leftrightarrow j : \exists_{i,j} \in V$. Therefore a graph with only one vertex is still connected and removing a vertex that is connected to only one other vertex does not disconnect the graph.

Ultimately there are only two graph components that an attacker can attack, edges or vertices. The selection of which of these components to attack has to be based on some metric rather than random selection. Holme and Kim [72] looked at how an attacker could maximize the damage to a graph by one of two approaches. The approaches being:

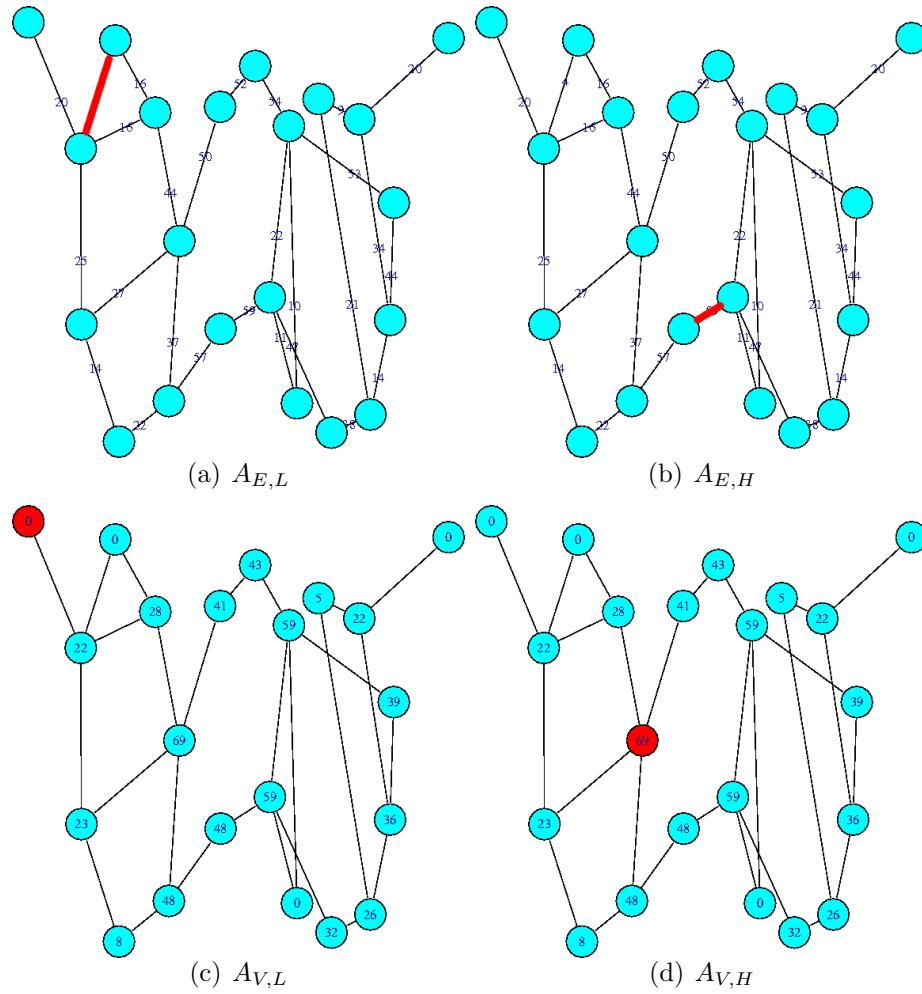


Figure 42. The first graph component that will be removed based on different attack profiles (1 of 2). Each profile selects a different component to be removed. In each of these figures, the first component to be removed is shown in red. In cases where more than one component has the appropriate qualities to qualify it for removal; selection of which component to remove is based on random selection.

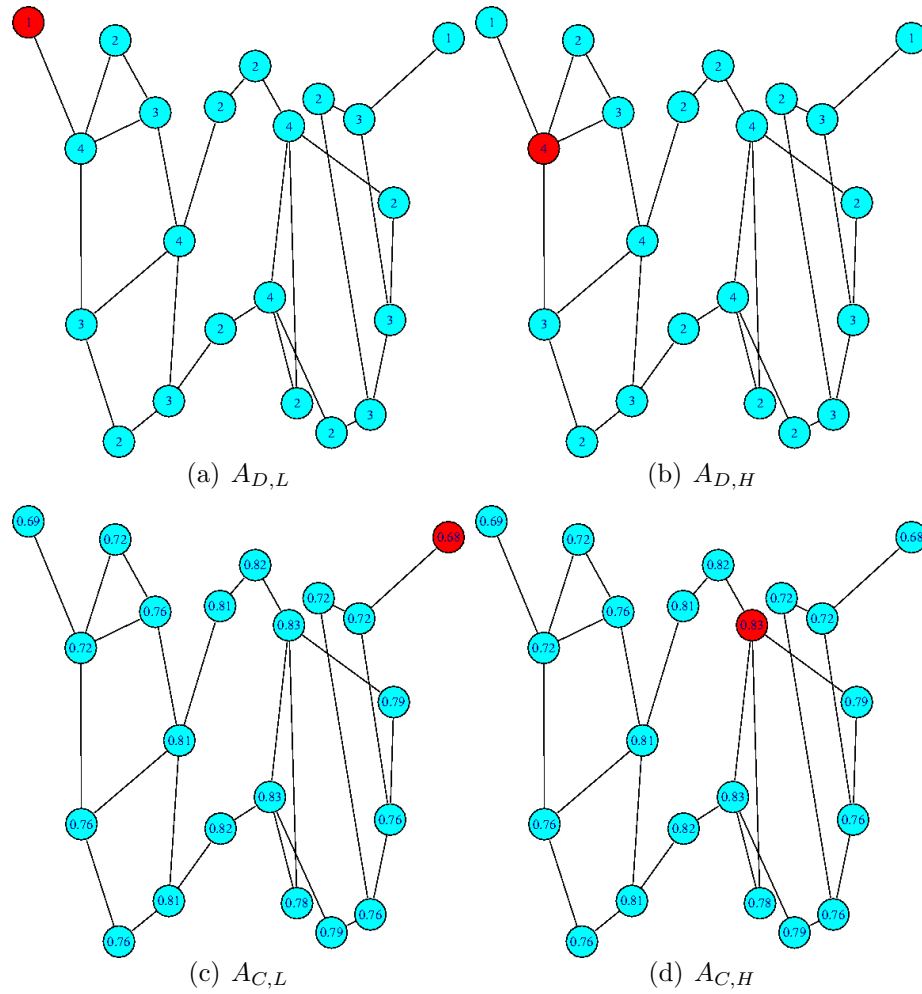


Figure 43. The first graph component that will be removed based on different attack profiles (2 of 2). Each profile selects a different component to be removed. In each of these figures, the first component to be removed is shown in red. In cases where more than one component has the appropriate qualities to qualify it for removal; selection of which component to remove is based on random selection.

1. To remove the vertex with the highest initial degree (ID)

$$c_D(v) = d(v) \quad (66)$$

2. Or, the vertex with the highest in-betweenness centrality (IB)

$$c_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (67)$$

Their idea about betweenness can be extended to include removing the edge with the highest in-betweenness centrality

$$c_B(e) = \sum_{s \neq t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}} \quad (68)$$

Lee et al. in [88] put forth failures in a network as being either **node**, **link**, or **path** related. Their **node** corresponds to our **vertex** and their **link** to our **edge**. And, their **path** to our **betweenness**. The betweenness of a component is a measurement of the component's contribution to all the shortest paths δ_{st} in the graph. The higher the betweenness value, the more shortest paths use that component.

In the following sections, we will use a sample graph to show the effects of an attacker's limited knowledge of the global graph on which component to remove.

5.5.3 SIZE OF SUBGRAPH TO EVALUATE

An attacker has to select a graph component to attack, and identifying which component to remove is based on the attacker's knowledge of some portion of the graph. The attacker's knowledge can range from a single component to complete knowledge of the graph. One approach to gaining knowledge of a graph's organization is to identify a vertex and then determine those vertices that are at a path length distance of 1 edge from the initial vertex. This process is repeated again and again until the attacker decides to stop increasing the path length.

In Figure 44 on page 154, vertex 5 is the source vertex and is colored red. The path length is initially set to 1 and the attacker now knows about the vertex set $\{4, 5, 6, 8, 9\}$ (Figure 44(a) on page 154). All attacker discovered vertices are colored

pink. As the path length increases from 2 (Figure 44(b) on the following page) to 4 (Figure 44(d) on the next page), more and more of the global graph becomes known. As readers, we know what the global graph looks like because we have an omnipotent view point. The attacker does not enjoy this view and must blindly continue to work outwards from his initial vertex. The attacker must expend time and energy to increase his knowledge of the graph, until at some point he will have spent “enough” and believes that sending additional time will not be worth the effort.

The attacker uses this limited local knowledge of the global graph to select the component whose removal will cause the greatest damage to the graph. If the path length is increased enough, the entire graph will be discovered. Barabási hypothesized that the entire World Wide Web could be discovered with a path length of 19 [162]. The resources for attempting to conduct such a discovery may be too large to be practical.

Edge selection

The selection of an edge to remove from the graph is based on how much of the graph that the attacker has discovered. As the discovered graph becomes larger and larger (as measured by the path length from an initial/central) vertex to the rest of the graph (Figure 44 on the following page), the more closely the betweenness value of the edge is to the edge’s betweenness value for the entire graph. The edge betweenness value for all edges in the global graph and for the discovered subgraph is shown in Table 19 on page 155. In the table, the first two columns are the vertices that are connected by an edge. The third column is the edge betweenness for that edge based on the global graph. The remaining columns show the edge betweenness value as the path length from the central vertex gets longer and longer. In those cases where the discovered subgraph has not discovered a particular vertex in the global graph, the edge betweenness value is marked with a — indicating no value possible. It is interesting to see how the value of an edge changes as the size of the graph changes. In most cases the value of an edge decreases as graph size increases.

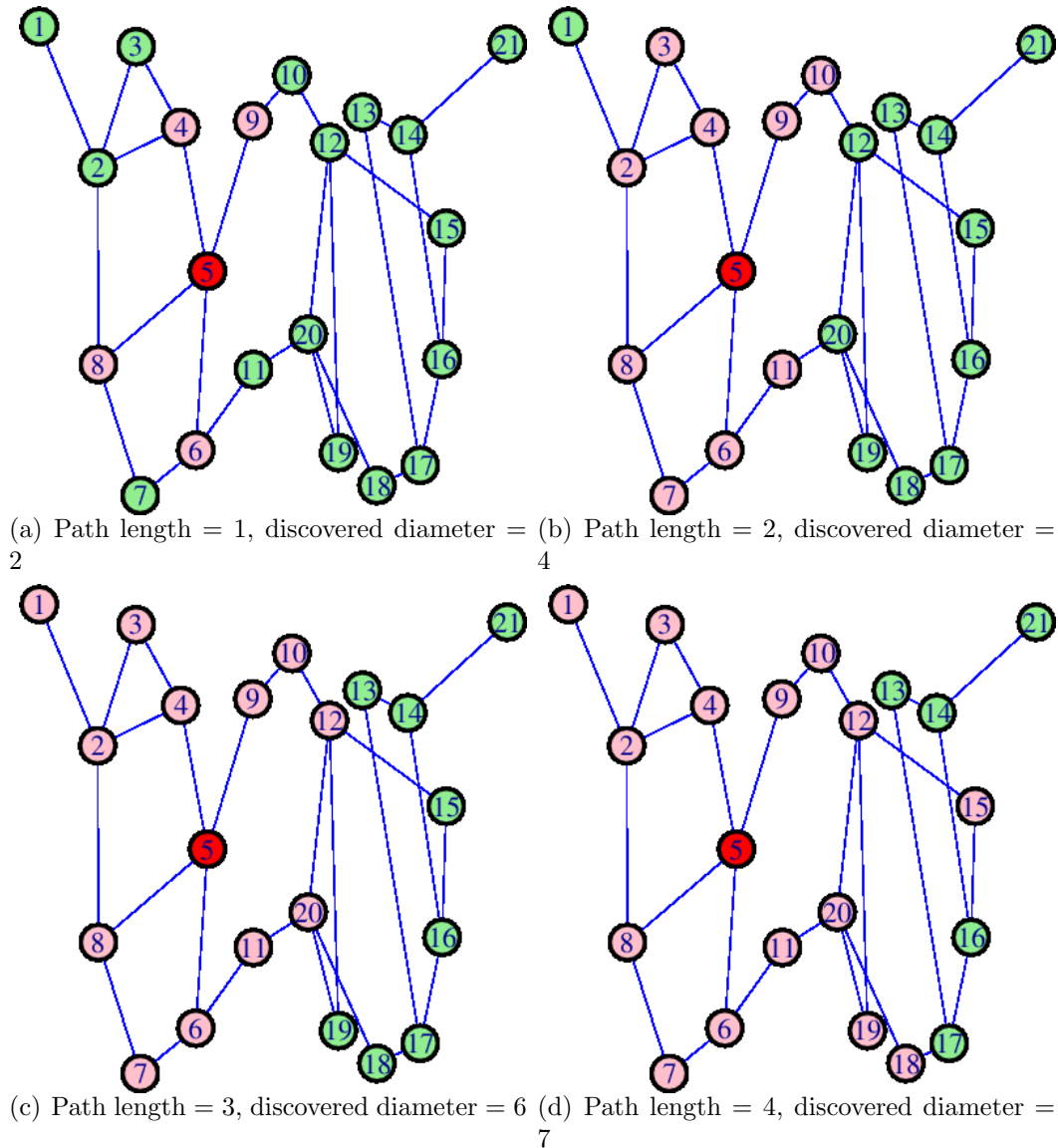


Figure 44. The effects of different path lengths starting from a fixed vertex in discovering the global graph. Vertex 5 is the center vertex. Each sub-figure shows the subgraph that is discovered based on the path length from the center vertex as the path length increments from 1 to 4. The diameter of the discovered subgraph is at most twice the path length. As the path length increases, more and more of the global graph is discovered.

Table 19. Comparing the betweenness of edges based on the neighborhood discovered from a central vertex. The size of the neighborhood increases from 1 to 4 based around vertex 5 (Figure 44 on the previous page). As the size of the neighborhood gets closer and closer to the global graph, the betweenness values get closer and closer to the global values. Those edges that have not been discovered because they belong to a portion of the global graph that has not been discovered are marked with a —.

Source node	Dest. node	Edge Betweenness	Path length 1	Path length 2	Path length 3	Path length 4
1	2	20.00	—	—	12.00	15.00
2	3	4.00	—	3.00	4.00	4.00
2	4	15.63	—	4.00	8.83	11.30
2	8	24.03	—	7.67	14.83	18.37
3	4	16.00	—	6.00	8.00	11.00
4	5	43.97	4.00	13.33	21.17	29.63
5	6	36.47	4.00	14.17	19.33	26.80
5	8	26.80	4.00	8.50	13.83	18.47
5	9	49.63	4.00	16.00	23.00	31.63
6	7	21.57	—	5.50	9.33	14.23
6	11	56.37	—	9.00	19.00	34.37
7	8	13.90	—	6.83	9.67	11.57
9	10	51.63	—	9.00	17.00	28.63
10	12	53.63	—	—	11.00	25.63
11	20	58.37	—	—	13.00	31.37
12	15	52.67	—	—	—	15.00
12	19	9.63	—	—	—	6.63
12	20	21.67	—	—	7.00	15.00
13	14	8.67	—	—	—	—
13	17	20.67	—	—	—	—
14	16	33.33	—	—	—	—
14	21	20.00	—	—	—	—
15	16	43.67	—	—	—	—
16	17	13.67	—	—	—	—
17	18	37.33	—	—	—	—
18	20	46.33	—	—	—	15.00
19	20	10.37	—	—	—	8.37

Vertex selection

The selection of a vertex to remove from the graph is based on how much of the graph that the attacker has discovered. As the discovered graph becomes larger and larger (as measured by the path length from a initial/central) vertex to the rest of the graph (Figure 44 on page 154), the more accurate the computed value betweenness value of the vertex is to the vertex's betweenness value for the entire graph. The betweenness value for all vertices in the global graph and for the discovered subgraph is shown in Table 20 on the next page. In the table, the first column is the vertex number. The second column is the vertex's betweenness value based on the global graph. The remaining columns show the vertex betweenness value as the path length from the central vertex gets longer and longer. In those cases where the discovered subgraph has not discovered a particular vertex in the global graph, the vertex betweenness value is marked with a — indicating no value possible. It is interesting to see how the value of an vertex changes as the size of the graph changes. In most cases the value of an vertex decreases as graph size increases. One notable exception is the vertex 2. As the graph size increases, that vertex's betweenness increases and decreases and yet in the global graph, its value is less than in some of the subgraphs.

Degree selection

Discovering the degree of a node is based on the idea that the nodes exchange messages between themselves and that the attacker can intercept these messages. As the attacker intercepts more and more messages; a node's neighbors (aka, degree) can be determined. The degree of a node can be used as a criterion to determine if the node is worthy of attack.

The degrees for the discovered graph based on differing path lengths is shown in Table 21 on page 158. The first column is the vertex number. The second column is the vertex's global degree. The remaining columns show the degree of each of the discovered vertices as the path length increases. If the vertex has not been discovered based on a particular path length then the marker — is used to indicate that no data is available. It is interesting to note that the degree of a vertex always increases as the path length increases until the global degree value is reached. Once the global value is reached, it remains constant.

5.5.4 ATTACK PROFILE NOTATION

Table 20. Comparing the betweenness of vertices based on the neighborhood discovered from a central vertex. The size of the neighborhood increases from 1 to 4 based around vertex 5 (Figure 44 on page 154). As the size of the neighborhood get closer and closer to the global graph, the betweenness values get closer and closer to the global values. Those vertices that have not been discovered because they belong to a portion of the global graph that has not been discovered are marked with a —. The betweenness values have been normalized to the range (0,1) to allow comparisons across different sized graphs.

Node	Vertex Be- tween- ness	Path length 1	Path length 2	Path length 3	Path length 4
1	0.00	—	—	0.00	0.00
2	0.32	—	0.13	0.42	0.37
3	0.00	—	0.00	0.00	0.00
4	0.41	0.00	0.33	0.40	0.40
5	1.00	1.00	1.00	1.00	1.00
6	0.69	0.00	0.46	0.55	0.66
7	0.11	—	0.08	0.11	0.12
8	0.33	0.00	0.33	0.40	0.36
9	0.59	0.00	0.37	0.43	0.49
10	0.62	—	0.00	0.24	0.43
11	0.69	—	0.00	0.31	0.55
12	0.86	—	—	0.09	0.52
13	0.07	—	—	—	—
14	0.31	—	—	—	—
15	0.56	—	—	—	0.00
16	0.52	—	—	—	—
17	0.38	—	—	—	—
18	0.47	—	—	—	0.00
19	0.00	—	—	—	0.00
20	0.85	—	—	0.12	0.60
21	0.00	—	—	—	—

Table 21. Comparing the degree of each vertex based on the neighborhood discovered from a central vertex. The size of the neighborhood increases from 1 to 4 based around vertex 5 (Figure 44 on page 154). As the size of the neighborhood get closer and closer to the global graph, the betweenness values get closer and closer to the global values.

Vertex	Degree	Path len. 1	Path len. 2	Path len. 3	Path len. 4
1	1	—	—	1	1
2	4	—	3	4	4
3	2	—	2	2	2
4	3	1	3	3	3
5	4	4	4	4	4
6	3	1	3	3	3
7	2	—	2	2	2
8	3	1	3	3	3
9	2	1	2	2	2
10	2	—	1	2	2
11	2	—	1	2	2
12	4	—	—	2	4
13	2	—	—	—	—
14	3	—	—	—	—
15	2	—	—	—	1
16	3	—	—	—	—
17	3	—	—	—	—
18	2	—	—	—	1
19	2	—	—	—	2
20	4	—	—	2	4
21	1	—	—	—	—

An attacker can target any graph component for removal based on the damage estimate or other criteria and whether to use the highest, or lowest valued component based on those criteria. We introduce the notation $A_{C,V}$ as a shorthand way to identify a specific profile. The first subscript in $A_{C,V}$ is the metric that is being used to select a component $C \in \{E, V, D, *\}$ for *edge*, *vertex*, *degree* or *any* respectively. The second subscript is the value of the metric that is being used $V \in \{L, M, H, R, *\}$ for *low*, *medium*, *high*, *random* or *any* respectively. The notation $A_{D,H}$ means that the attacker is using a profile that targets nodes based on their degree D and chose the highest H valued one.

5.5.5 EFFECTIVENESS OF DIFFERENT ATTACK PROFILES

The damage to a graph by fragmentation can be calculated (Equation 111 on page 387) using the fragmented graph and approximating the graph without fragmentation. A connected graph is created from the fragmented graph by adding an edge between each of the highest degreed nodes of each fragment. As each edge is added to coalesce the fragments into a larger and larger connected component, the highest degreed node may change based on the order in which the fragments are coalesced. Therefore the highest degreed node in the coalescing component must be evaluated after each fragment addition. At the end of the collation process, there will be a single connected component containing the same number of nodes as the fragmented graph and one additional edge for each of the original fragments.

As the original graph becomes more and more fragmented, its AIPL will decrease. The AIPL of the connected approximation will decrease and the $Damage(G)$ will increase as well. This behavior is readily apparent when edges are removed from the original graph in order to create the fragments. When vertices are removed, the behavior is similar, until the last vertex is removed. In the limiting case, AIPL of the fragmented graph with one fragment and one node in that fragment, is the same as the AIPL of a connected component with one node. Using Equation 111 on page 387 results in a value of 0, meaning that the graph is undamaged.

Edge selection

The attacker can compute the betweenness of any edge in the subgraph that he has discovered, see Table 19 on page 155. Based on these computed betweenness values, the attacker can select either the highest or lowest valued edge to remove.

After the removal of this edge, the betweenness values can be recomputed for the newly modified subgraph and the process repeated again and again until there are no edges left in the discovered graph (the discovered graph is totally destroyed).

Figures 45 on the following page and 46 on page 162 show the effects of repeatedly applying attack $A_{E,L}$ or $A_{E,H}$ profile to the discovered subgraph of path length 3. In each figure, the betweenness value of each edge is written on the edge. The edge with the lowest (Figure 45 on the following page) or highest (Figure 46 on page 162) betweenness value is highlighted in red, prior to it being removed. After the removal of the edge, the betweenness values of all the remaining edges is computed shown in the next sub-figure, along with the next edge that has been selected for removal. The four sub-figures in Figures 45 on the following page and 46 on page 162 show this process. When two or more edges have the same betweenness value, the selection of which edge to remove it totally random.

Attack profile $A_{E,L}$ tends to attack the periphery of the graph, while profile $A_{E,H}$ tends to attack the core of the graph. Either profile will result in a fully disconnected graph with the same number of removals, selecting the highest valued edge causes more damage quicker.

Table 22 on page 163 lists the computed damage to the discovered subgraph after the removal of either the highest or lowest betweenness valued edge. Figure 47 on page 164 shows the damage plotted against the deletion. There are 16 edges in the discovered subgraph and damage is total upon the removal of the last edge.

Vertex selection

The attacker can compute the betweenness of any vertex in the subgraph that he has discovered Table 20 on page 157. Based on these computed betweenness values, the attacker can select either the highest or lowest valued vertex to remove. After the removal of this vertex, the betweenness values can be recomputed for the newly modified subgraph and the process repeated again and again until there are no vertices left in the discovered graph (the discovered graph is totally destroyed).

Figures 48 on page 166 and 49 on page 167 show the effects of repeatedly applying $A_{V,L}$ or $A_{V,H}$ profile to the discovered subgraph of path length 3. In each figure, the betweenness value of each vertex is written in the vertex. The vertex with the lowest (Figure 48 on page 166) or highest (Figure 49 on page 167) betweenness value is highlighted in yellow, prior to it being removed. After the removal of the vertex,

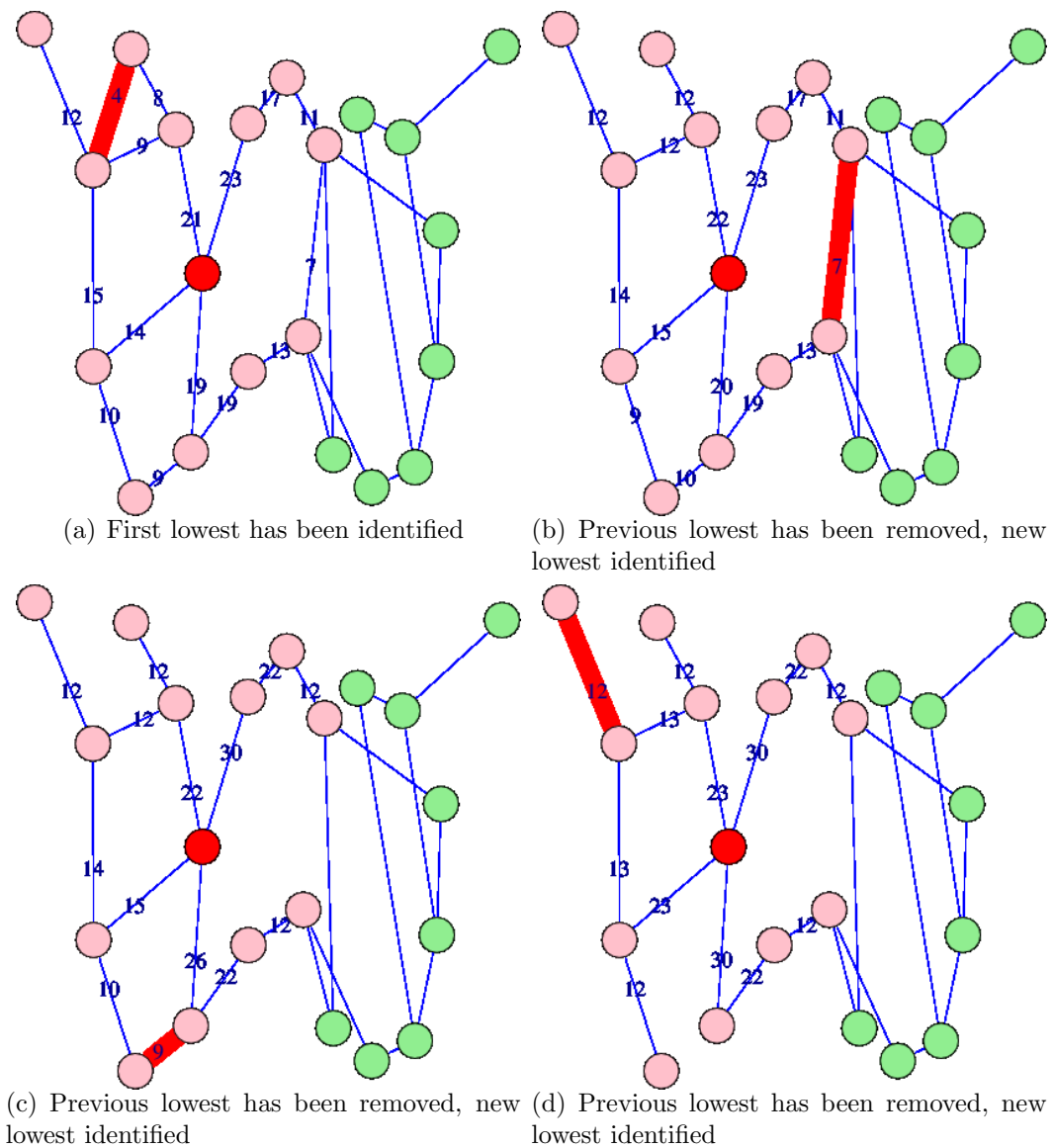


Figure 45. The effects of the $A_{E,L}$ attack profile on the sample graph. Vertex 5 is the center vertex and is marked in red. The discovered graph is at a path length of 3 from the center vertex and is marked in pink. The edge with the lowest betweenness value is marked in red. After each deletion, all edge betweenness values are recomputed because the graph has changed. Some of the edges are unlabeled because the attacker has not “discovered” them.

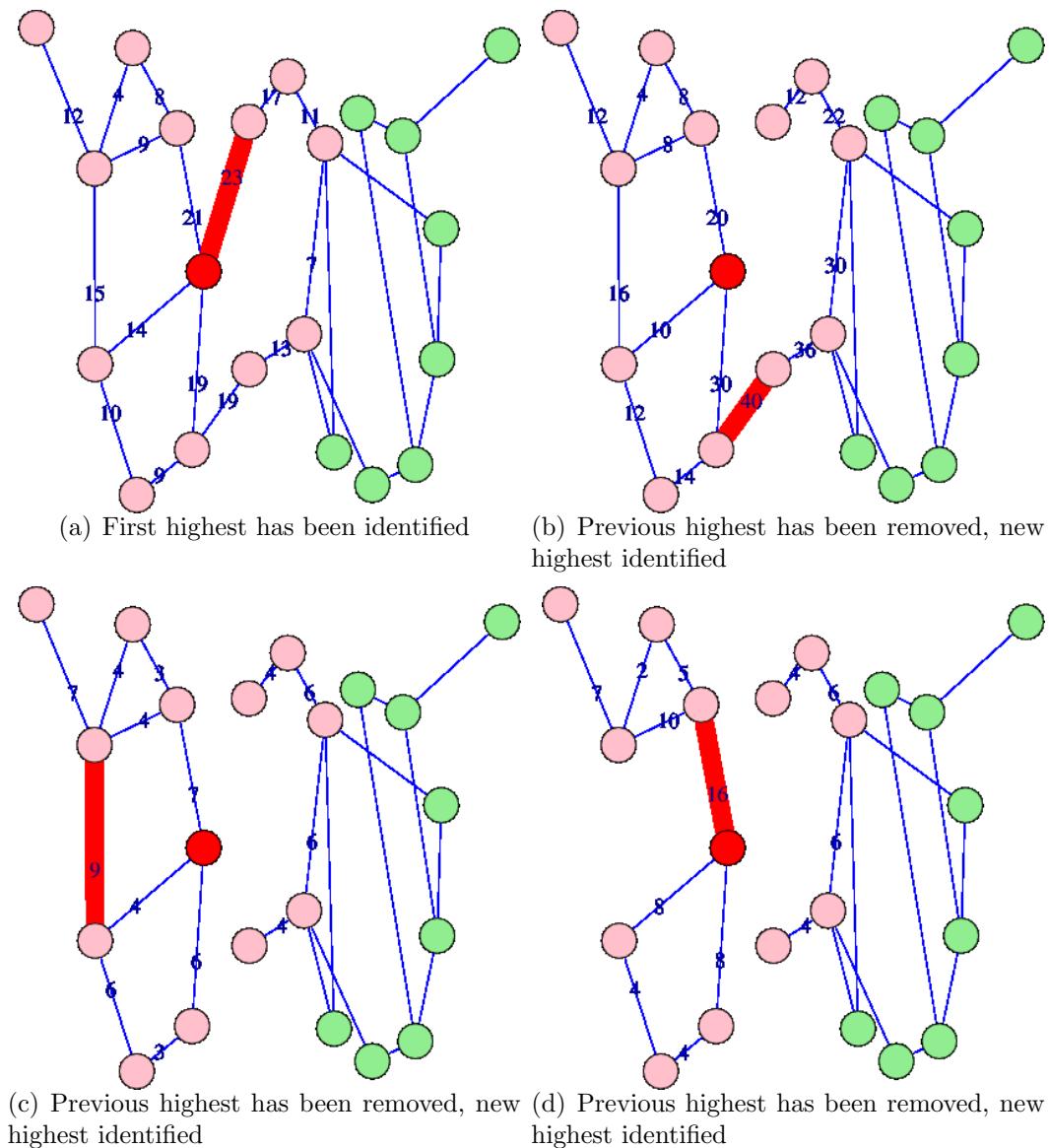


Figure 46. The effects of the $A_{E,H}$ attack profile on the sample graph. Vertex 5 is the center vertex and is marked in red. The discovered graph is at a path length of 3 from the center vertex and is marked in pink. The edge with the highest betweenness value is marked in red. After each deletion, all edge betweenness values are recomputed because the graph has changed. Some of the edges are unlabeled because the attacker has not “discovered” them.

Table 22. Damage to the discovered subgraph of path length 3 based on $A_{E,*}$ attack profiles. The betweenness of each edge is recomputed after the removal of either the highest or lowest betweenness valued edge. The process is repeated again and again until all edges are removed.

Deletion	Local damage due to $A_{E,H}$	Global damage due to local damage by $A_{E,H}$	Local damage due to $A_{E,L}$	Global damage due to local damage by $A_{E,L}$
0	0.00	0.00	0.00	0.00
1	0.10	0.06	0.02	0.01
2	0.36	0.31	0.07	0.03
3	0.41	0.33	0.10	0.05
4	0.57	0.41	0.21	0.12
5	0.65	0.50	0.23	0.13
6	0.70	0.53	0.34	0.19
7	0.72	0.54	0.44	0.26
8	0.78	0.57	0.54	0.32
9	0.82	0.62	0.62	0.38
10	0.83	0.62	0.71	0.43
11	0.87	0.64	0.77	0.48
12	0.89	0.65	0.83	0.52
13	0.92	0.67	0.89	0.57
14	0.95	0.68	0.93	0.61
15	0.97	0.69	0.97	0.65
16	1.00	0.70	1.00	0.70

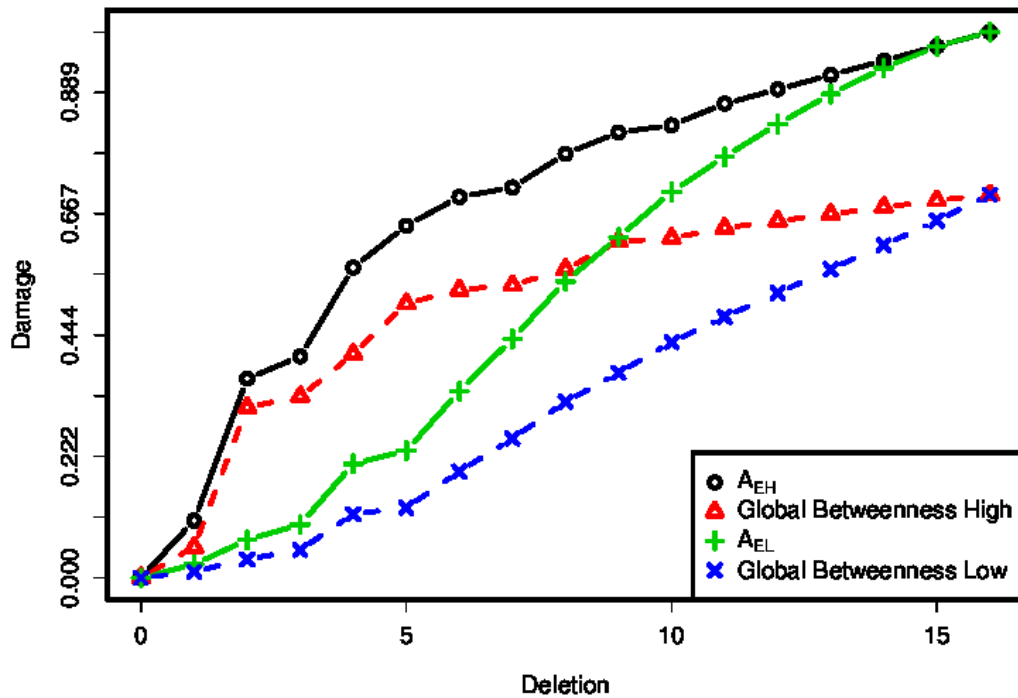


Figure 47. Damage to the discovered graph of path length 3 based on $A_{E,*}$ attack profiles. The “local” values are those that come from the discovered graph, while the global values are from the total graph. Damage inflicted on the discovered graph when using the high edge betweenness value and the resulting impact on the total graph are shown in black and red respectively. In a similar manner, damage caused by choosing the low betweenness is shown in the green and blue lines respectively. The betweenness of each edge is recomputed after the removal of either the highest or lowest betweenness valued edge. The process is repeated again and again until all edges are removed.

the betweenness values of all the remaining vertices are computed and shown in the next sub-figure, along with the next vertex that has been selected for removal. The four sub-figures in Figures 48 on the next page and 49 on page 167 show this process. When two or more vertices have the same betweenness value, the selection of which edge to remove it totally random.

Attack profile $A_{V,L}$ tends to attack the periphery of the subgraph. While attack profile $A_{V,H}$ tends to attack the core of the graph. While both selection choices will result in a fully disconnected graph with the same number of removals, selecting the highest valued vertex causes more damage quicker.

The betweenness computation, removal and damage computation process is shown in Table 23 on page 168 and Figure 50 on page 169. The global high line in Figure 50 on page 169 goes flat after the fifth deletion while the global low line continues to increase. This behavior is explained by looking at Figures 51(a) on page 170 and 51(b) on page 170. By the fifth high deletion, the discovered and global graphs are disconnected and further local deletions do not affect the global graph. In Figure 51(a) on page 170, the discovered and global graphs are still connected and local deletions will affect the global graph.

Degree selection

The attacker can compute the degree of any vertex in the subgraph that he has discovered Table 21 on page 158. Based on these values, the attacker can select either the highest or lowest valued vertex to remove. After the removal of this vertex, the degree values can be recomputed for the newly modified subgraph and the process repeated again and again until there are no vertices left in the discovered graph (the discovered graph is totally destroyed).

Figures 52 on page 172 and 53 on page 173 show the effects of repeatedly applying attack $A_{D,L}$ or $A_{D,H}$ profiles to the discovered subgraph of path length 3. In each figure, the degree value of each vertex is written in the vertex. The edge with the lowest (Figure 52 on page 172) or highest (Figure 53 on page 173) betweenness value is highlighted in yellow, prior to it being removed. After the removal of the vertex, the degree values of all the remaining vertices are computed shown in the next sub-figure, along with the next vertex that has been selected for removal. The four sub-figures in Figures 52 on page 172 and 53 on page 173 show this process. When two or more vertices have the same degree value, the selection of which

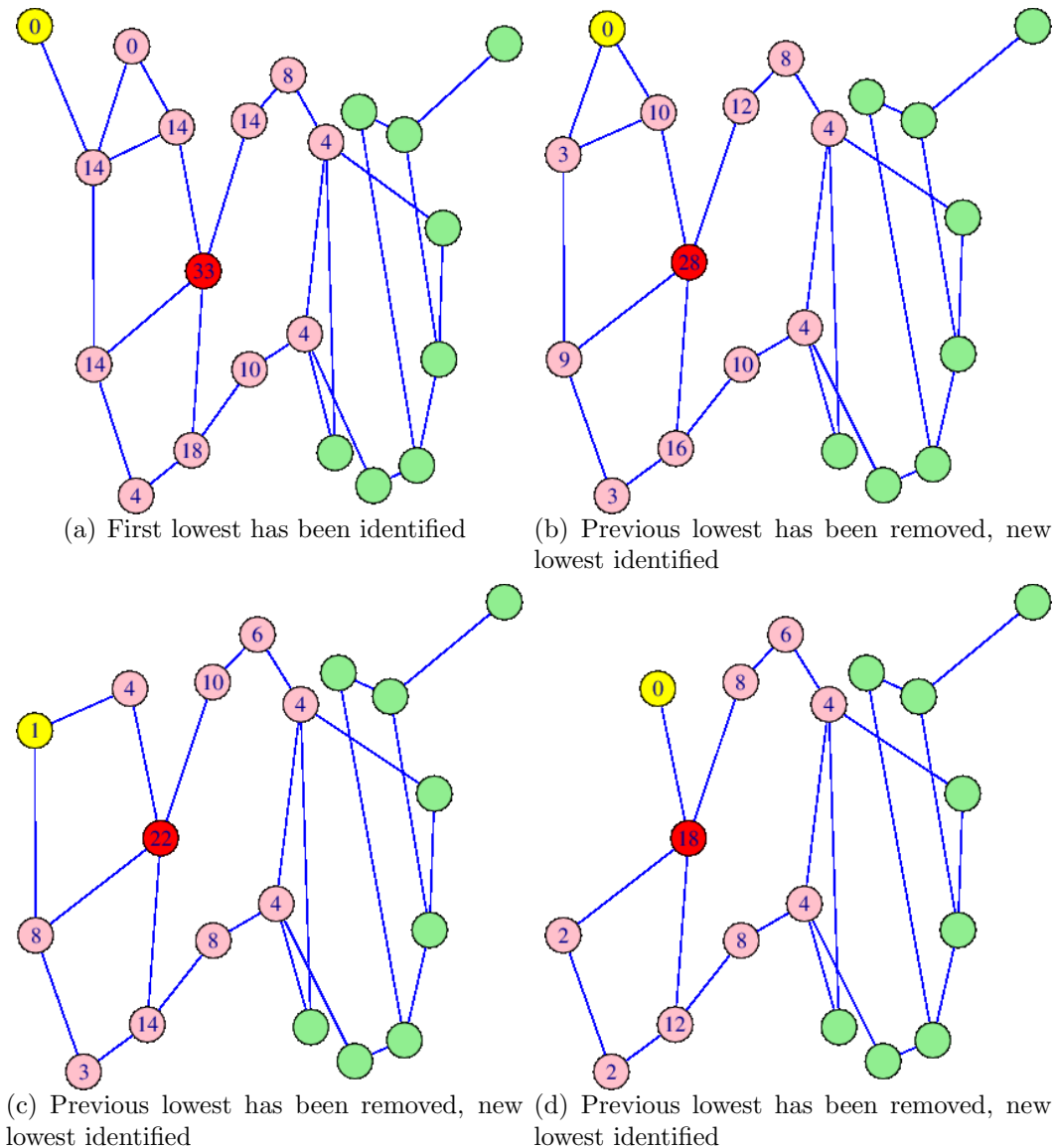


Figure 48. The effects of an $A_{V,L}$ attack profile on the sample graph. Vertex 5 is the center vertex and is shown in red. The discovered graph, in pink is at a distance of 3 from the center vertex. Each vertex is labeled with the number of shortest paths that go use that vertex. The vertex with the lowest betweenness is drawn in yellow. Each time, the lowest valued vertex is removed from the discovered graph and all betweenness values for the discovered graph are recomputed. If there is more than one vertex with the same low value, one is selected at random for removal. Some of the vertices are unlabeled because the attacker has not “discovered” them.

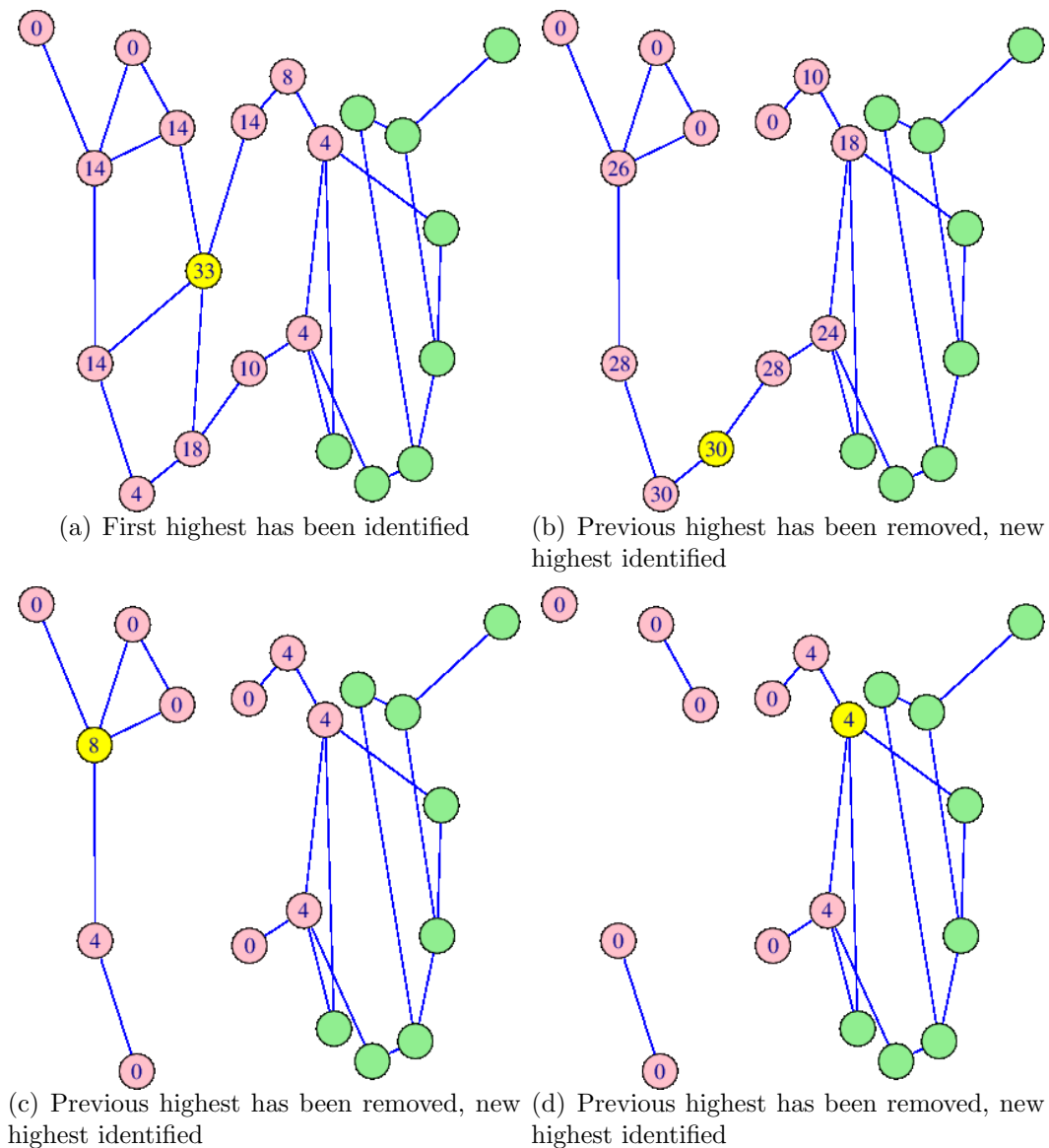


Figure 49. The effects of an $A_{V,H}$ attack profile on the sample graph. Vertex 5 is the center vertex. The discovered graph is at a distance of 3 from the center vertex. The vertex with the highest betweenness is drawn in yellow. Each time, the highest valued vertex is removed from the discovered graph and all betweenness values for the discovered graph are recomputed. If there is more than one vertex with the same high value, one is selected at random for removal. Some of the vertices are unlabeled because the attacker has not “discovered” them.

Table 23. Damage to the discovered subgraph of path length 3 based on $A_{V,*}$ attack profiles. The betweenness of each vertex is recomputed after the removal of either the highest or lowest betweenness valued vertex. The process is repeated again and again until all vertices are removed.

Deletion	Local damage due to $A_{V,H}$	Global damage due to local damage by $A_{V,H}$	Local damage due to $A_{V,L}$	Global damage due to local damage by $A_{V,L}$
0	0.00	0.00	0.00	0.00
1	0.29	0.17	0.12	0.07
2	0.57	0.41	0.24	0.14
3	0.78	0.51	0.36	0.22
4	0.89	0.68	0.47	0.28
5	0.89	0.68	0.58	0.35
6	0.92	0.70	0.66	0.41
7	0.92	0.70	0.77	0.48
8	0.95	0.71	0.83	0.54
9	0.95	0.71	0.89	0.59
10	0.97	0.72	0.93	0.64
11	0.97	0.72	0.97	0.69
12	1.00	0.77	1.00	0.77

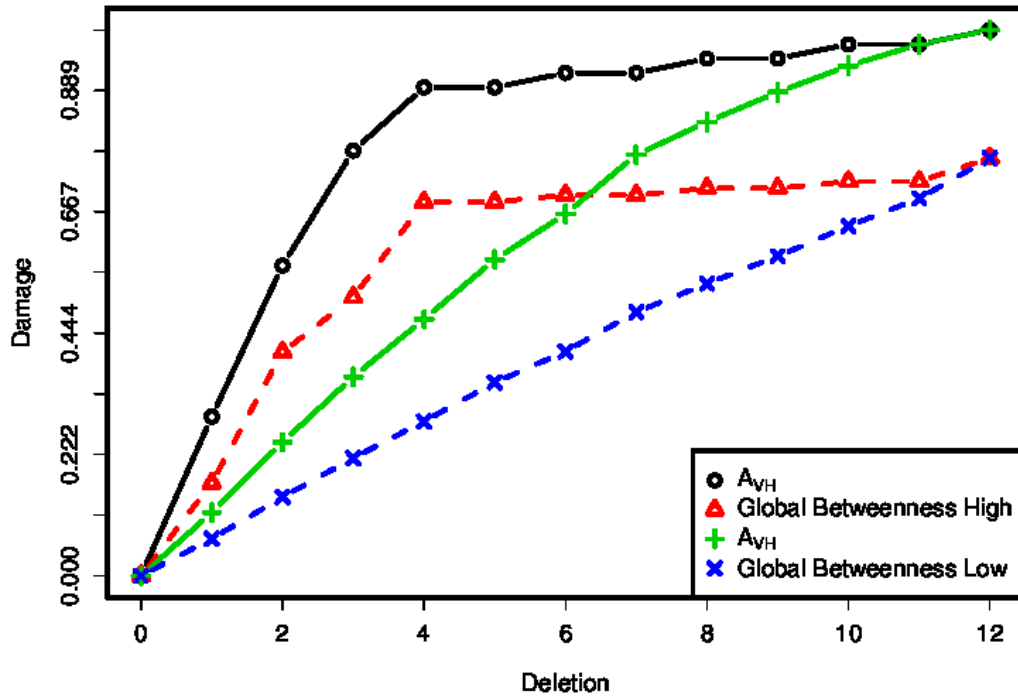


Figure 50. Damage to the discovered subgraph of path length 3 based on $A_{V,*}$ attack profiles. The “local” values are those that come from the discovered graph, while the global values are from the total graph. Damage inflicted on the discovered graph when using the high vertex betweenness value and the resulting impact on the total graph are shown in black and red respectively. In a similar manner, damage caused by choosing the low betweenness is shown in the green and blue lines respectively. The betweenness of each vertex is recomputed after the removal of either the highest or lowest betweenness valued vertex. The process is repeated again and again until all vertices are removed. Damage to the global graph is flat from deletion 4 through 11, while the local damage increases due to the selection of the particular high valued vertices to remove. The low betweenness option does not show this type of behavior. The system of graphs for high and low selection is shown in Figure 51 on the following page.

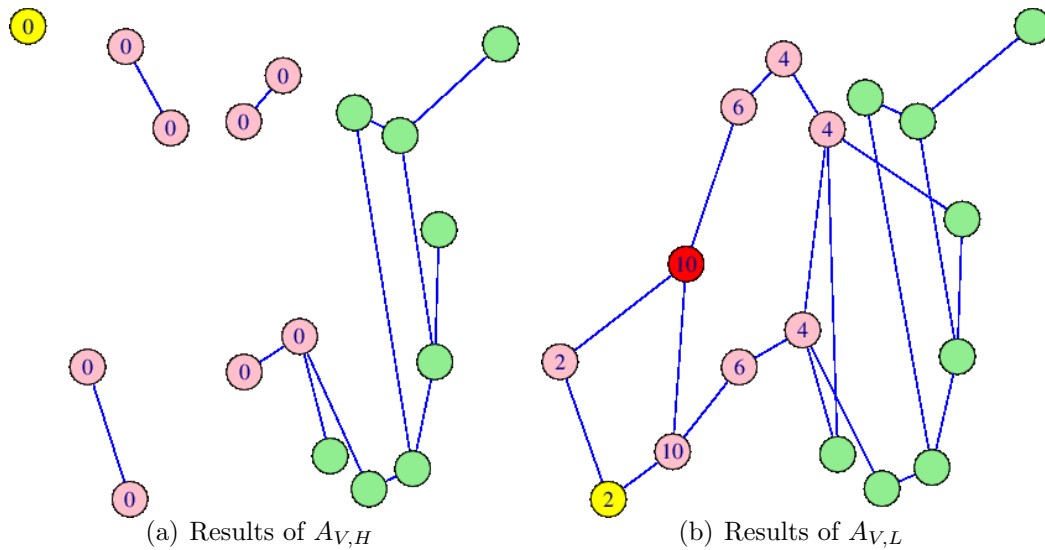


Figure 51. Markedly different graphs resulting from the differences in choosing $A_{V,H}$ or $A_{V,L}$ attack profiles. Both sub-figures show the sample graph after 4 deletions based on $A_{V,H}$ or $A_{V,L}$ attack profiles. Continued deletions in the discovered graph (in pink) in the high betweenness case (Figure 51(a)), will have only marginal effect on the global graph (the union of pink and green). Deletions in the discovered graph in low betweenness case (Figure 51(b)) will continue to affect the union of the pink and the green nodes because the two graphs (pink and green) are still connected. Some of the vertices are unlabeled because the attacker has not “discovered” them.

edge to remove it totally random.

Attack profile $A_{D,L}$ tends to attack the periphery of the subgraph. While attack profile $A_{D,H}$ tends to attack the core of the graph. While both selection choices will result in a fully disconnected graph with the same number of removals, selecting the highest valued vertex causes more damage quicker.

The betweenness computation, removal and damage computation process is shown in Table 24 on page 174 and Figure 54 on page 175. The Global High line in Figure 54 on page 175 goes flat after the fifth deletion while the Global Low line continues to increase. This behavior is explained by looking at Figures 55(a) on page 176 and 55(b) on page 176. Using a $A_{D,H}$ profile, the discovered and global graphs are disconnected and further local deletions do not affect the global graph. Using $A_{D,L}$ profile in Figure 55(b) on page 176 results in the discovered and global graphs still being connected, so any deletions on the discovered graph affect the global graph.

5.5.6 SUMMARY

All node based attacks ($A_{V,*}$, $A_{D,*}$) will totally destroy the discovered graph. All edge based attacks $A_{E,*}$ will cause the discovered graph to be totally disconnected. The two attack philosophies differ in their efficacy and are summarized in Table 25 on page 174. Attack efficacy was computed by integrating the area under the $A_{*,H}$ and dividing it by the area under the $A_{*,L}$ curves (Equation 69).

$$efficacy = \frac{\int_1^n Damage(A_{*H})}{\int_1^n Damage(A_{*L})} \quad (69)$$

In all cases the $A_{*,H}$ attack profiles were more destructive than the $A_{*,L}$ profile. HTTP/HTML protocols preclude $A_{E,*}$ (see Section 6.11.2 on page 240).

If the attacker's goal is to disconnect the graph by repeated use of the same attack profile, then the most effective profiles in order are: $A_{E,H}$, $A_{V,H}$ and $A_{D,H}$.

5.6 EFFECT OF CLUSTERING COEFFICIENT EQUATION SELECTION

Since being introduced by Watts and Strogatz [43] the clustering coefficient for a graph ($C(G)$) is a staple topic when discussing Small-World graph characteristics. Their definition was written in the caption of one figure and therefore is open for

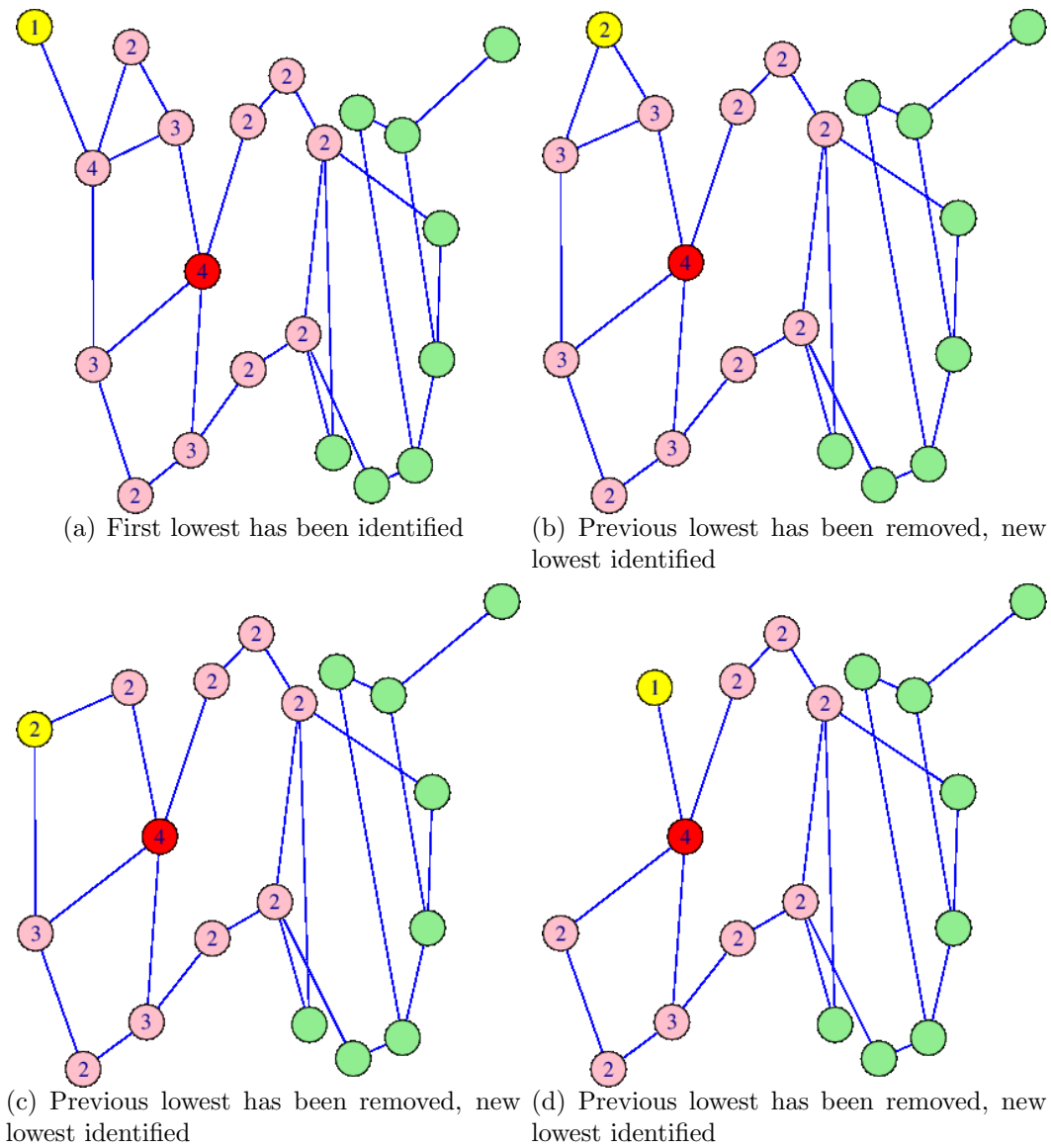


Figure 52. The effects of an $A_{D,L}$ attack profile on the sample graph. Vertex 5 (marked in red) is the center vertex. The discovered graph is at a distance of 3 from the center vertex. The vertex with the lowest degree is marked in yellow. In the case where multiple vertices have the same degree value (Figure 52(b)), random choice is used to select one vertex as the next one to be removed. Removal of a vertex causes a reduction in the degree values of all of the removed vertex's neighbors. This change in the degree values of potentially many vertices requires that the relative order of the vertices be evaluated after each removal. Some of the vertices are unlabeled because the attacker has not "discovered" them.

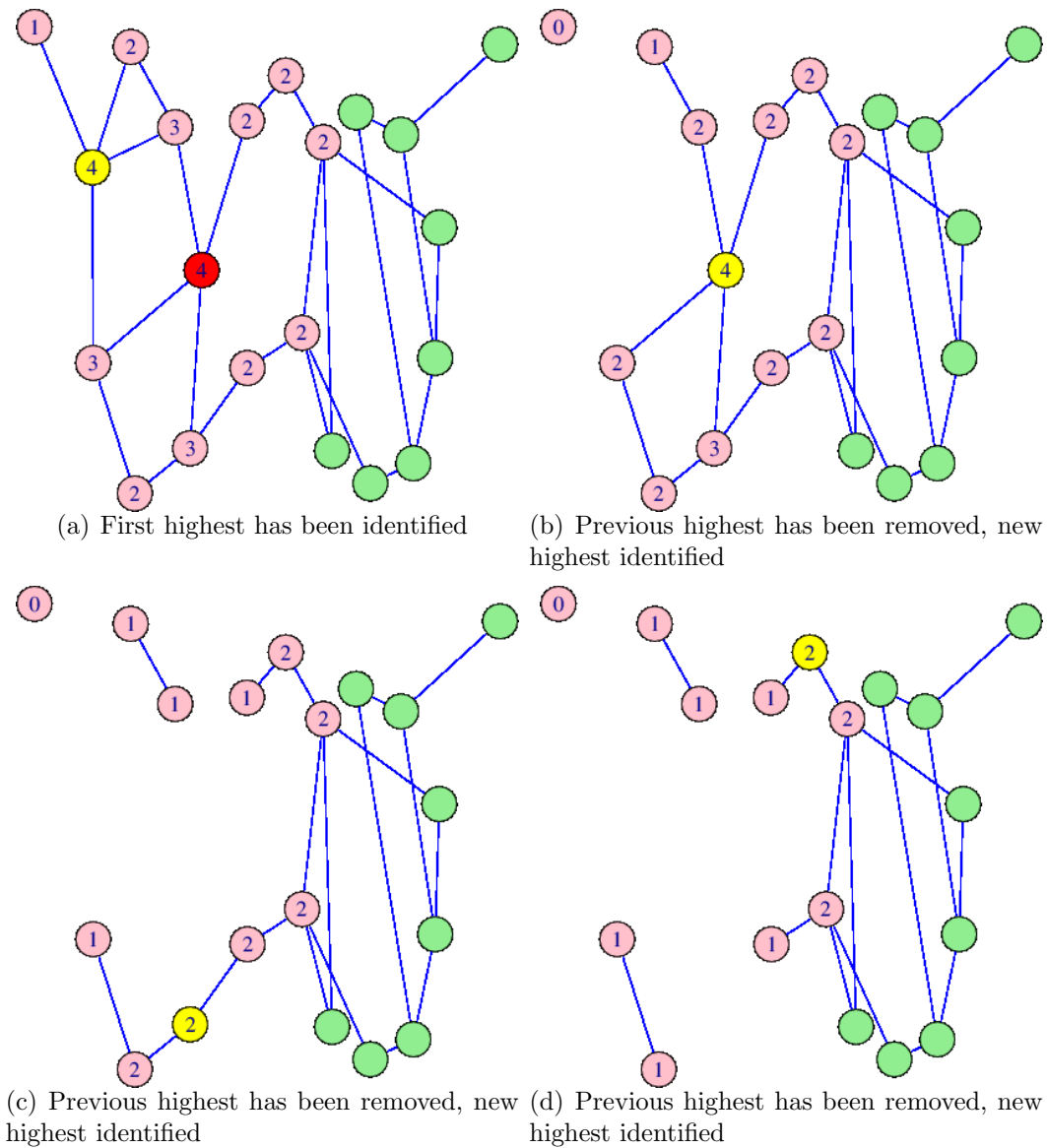


Figure 53. The effects of on $A_{D,H}$ attack profile on the sample graph. Vertex 5 (marked in red) is the center vertex. The discovered graph is at a distance of 3 from the center vertex. The vertex with the highest degree is marked in yellow. In the case where multiple vertices have the same degree value (Figure 53(c)), random choice is used to select one vertex as the next one to be removed. Removal of a vertex causes a reduction in the degree values of all of the removed vertex's neighbors. This change in the degree values of potentially many vertices requires that the relative order of the vertices be evaluated after each removal. Some of the vertices are unlabeled because the attacker has not "discovered" them.

Table 24. Damage to the discovered subgraph of path length 3 based on $A_{D,*}$ attack profiles. The degree of each vertex is computed after each deletion. A vertex's degree value will change if one of its immediate neighbor vertices has been removed. The removal of a neighbor will reduce the degree of all its neighbors by one. This change in the degree of all neighboring vertices may affect the relative order of all vertices based on their respective degree. The process is repeated again and again until all edges are removed.

Deletion	Local damage due to $A_{D,H}$	Global damage due to local damage by $A_{D,H}$	Local damage due to $A_{D,L}$	Global damage due to local damage by $A_{D,L}$
0	0.00	0.00	0.00	0.00
1	0.27	0.16	0.12	0.07
2	0.61	0.37	0.24	0.14
3	0.78	0.51	0.36	0.22
4	0.88	0.62	0.47	0.28
5	0.95	0.74	0.58	0.35
6	0.97	0.75	0.66	0.41
7	1.00	0.76	0.77	0.48
8	1.00	0.76	0.83	0.54
9	1.00	0.76	0.89	0.59
10	1.00	0.76	0.93	0.64
11	1.00	0.76	0.97	0.69
12	1.00	0.76	1.00	0.77

Table 25. Efficacy of various attack profiles. In general, regardless of the attack profile utilized, attacking the highest valued component is the most destructive.

Attack Profile	Attacks	Efficacy
$A_{E,H}$	The core of the graph	1.43
$A_{E,L}$	The periphery of the graph	1.00
$A_{V,H}$	The core of the graph	1.42
$A_{V,L}$	The periphery of the graph	1.0
$A_{D,H}$	The core of the graph	1.40
$A_{D,L}$	The periphery of the graph	1.00

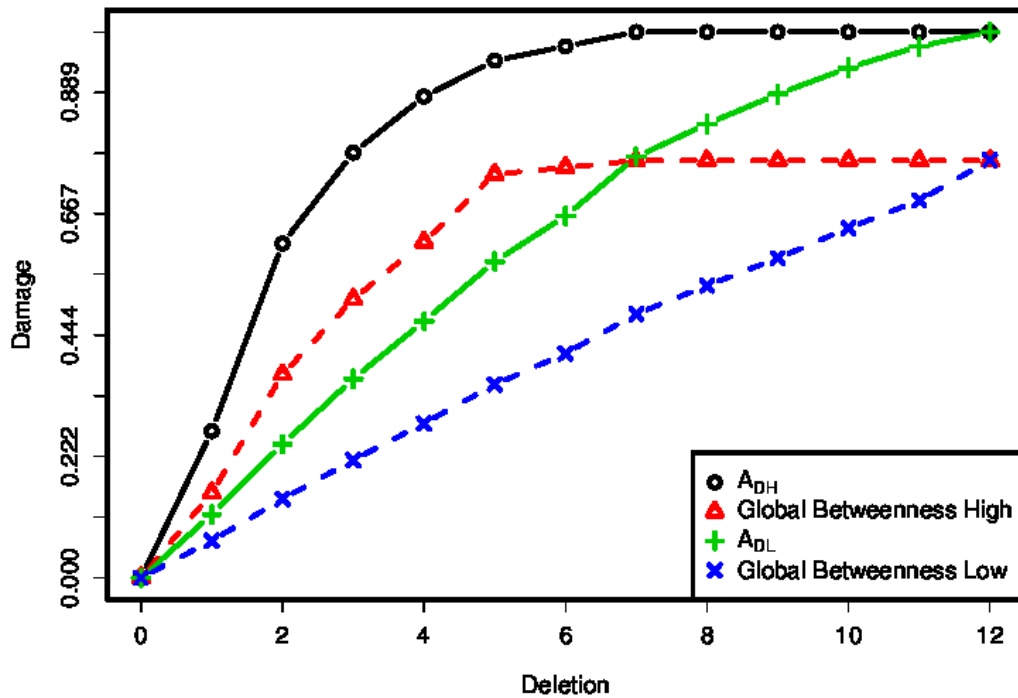


Figure 54. Damage to the discovered subgraph of path length 3 by based on $A_{D,*}$ attack profiles. The degree of each vertex is computed after each deletion. A vertex's degree value will change if one of its immediate neighbor vertices have been removed. The removal of a neighbor will reduce the degree of all its neighbors by one. This change in the degree of all neighboring vertices may affect the relative order of all vertices based on their respective degree. The process is repeated again and again until all vertices are removed. The flat area on the Global High line is related to the discovered and global graphs becoming disconnected (Figure 55 on the next page).

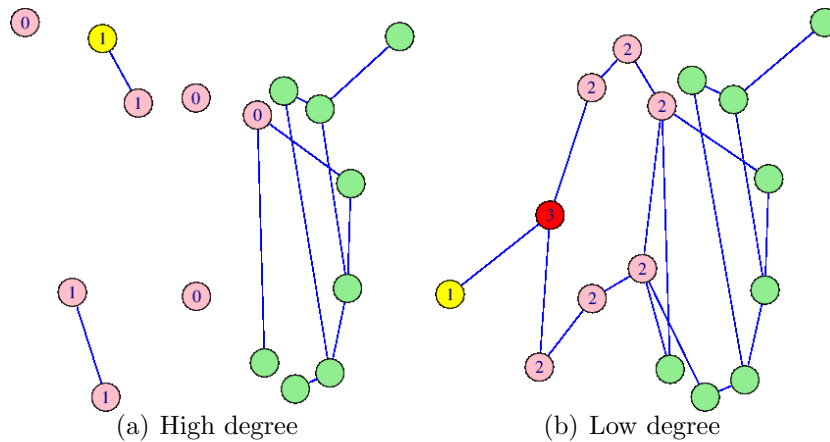


Figure 55. The sample graph after removing the fifth discovered node using $A_{D,*}$ attack profiles. The undiscovered graph is drawn in green. The central vertex, where it remains is drawn in red (Figure 55(b)). The vertex that will be deleted next is drawn in yellow. While each graph shows the effects of five deletions, selecting the highest degreed node to delete results in a graph that is disconnected (Figure 55(a)). Focusing on the lowest degreed node results in damage to the periphery and a graph that is still connected (Figure 55(b)). Some of the vertices are unlabeled because the attacker has not “discovered” them.

interpretation. Newman [178] presents equations for two different $C(G)$ s and provides a formulaic interpretation of Watts’ definition. The equations for Watts and Newman have been coded into the R packages, as well as one formulation whose source I have yet to discover. Each $C(G)$ definition, natural language interpretation, and evaluation are given in the following subsections.

5.6.1 CLUSTERING COEFFICIENT DEFINITIONS

A literature search identified several different definitions for clustering coefficient (Table 26 on the next page).

Table 26. Various clustering coefficients definitions and their sources.

Cluster Coefficient Definition	Equation, Natural language, Ref. (Eq.)
C_{Global}	$C(G)_{Average} = \frac{3 * \Sigma \text{NumberOfTrianglesInTheGraph}}{\text{NumberOfConnectedTriples}}$ <p>The total number of triangles in the graph divided by the total number of triples in the graph. [178, This is the ratio of the means] (3) [179] (1)</p>
$C(G)_{AverageLocal}$	$C_i = \frac{\text{NumberOfTrianglesConnectedToVertex}i}{\text{NumberOfTriplesCenteredAtVertex}i}$ $C(G)_{AverageLocal} = \frac{1}{n} \sum_i \frac{\text{NumberOfTrianglesConnectedToVertex}i}{\text{NumberOfTriplesCenteredAtVertex}i}$ <p>The summation of the ratio of triangles at each node divided by the number of triples that that node belongs to averaged over all nodes. [178, This is the means of the ratios.] [43]</p>

(Continued on the next page.)

Table 26. (Continued from the previous page.)

Cluster Coefficient Definition	Equation, Natural language, Ref. (Eq.)
	$C_i = \frac{\text{NumberOfTrianglesConnectedToVertex}i}{\text{NumberOfTriplesCenteredAtVertex}i}$
$C(G)_{AverageTriples}$	$C(G)_{AverageTriples} = \frac{1}{\sum_i, C_i \neq 0} \sum_i \frac{\text{NumberOfTrianglesConnectedToVertex}i}{\text{NumberOfTriplesCenteredAtVertex}i}$ <p>The summation of the ratio of triangles at each node divided by the number of triples that that node belongs to averaged over nodes that belong to a triple. Equation “reverse engineered” based on analysis of results using test environments.</p>
$C(G)_{Lattice}$	$C(G)_{Lattice} = \frac{(3k-3)}{2(2k-1)} = C(G) \sim \frac{3}{4} \text{ for large } k$ <p>$C(G)$ is purely dependent on the number of nodes (k) that each node is linked to. $C(G)_{Lattice}$ and $C(G)_{LatticeRewired}$ are special cases of C_{Global} because the number of triangles and triples can be computed directly because the number of connections (aka, degrees) is given by the value of k. [171] (pg. 8)</p>
$C(G)_{LatticeRewired}$	$C_{LatticeRewired} = \frac{(3k-3)}{(4k-2)} * (1-p)^3$ <p>$C(G)$ is initially dependent on the number of nodes (k) that each node is linked to and then the probability that an edge will be rewired. [171] (8)</p>

(Last page.)

5.6.2 TEST ENVIRONMENT

The purpose of this investigation was to validate the custom code written as part of my research into USW construction. During this research, we learned that R had different packages (igraph [78] and sna [180]) that would/could compute a graph's $C(G)$; so we undertook to compare the various R libraries. The evaluation process was to construct and evaluate Watts – Strogatz graphs, evaluate a small graph that lends itself to hand computations, and finally to evaluate Unsupervised Small-World (USW) graphs.

Part of the investigation, led to re-creating Watts – Strogatz ring lattices of size 2000 (n) and connections (k) equalling 10 where the probability of re-wiring varied from 0 to 1. These values meet the requirement that [43]:

$$n \gg k \gg \ln(n) \gg 1$$

The results are shown in Figure 56 on the following page. The $C(G)$ line is very dark because there are eight symbols (one for each “type” in the igraph transitivity function) being plotted nearly on top of each other. The raw data is shown as well (see Figure 57 on the next page). The Watts – Strogatz small-world graphs were created, average path length $L(G)$, and different cluster coefficient $C(G)$ values are computed using the *transitivity* function from the R igraph package.

One of the interesting things in Figure 57 on the following page is that when the probability approaches 1, the average path length approaches:

$$l = \log(n)$$

Figure 58 on page 181 is an artificial undirected graph. It was created because it is small enough to be validated by hand and yet has enough interesting features to be useful when discussing the various $C(G)$ definitions. Table 27 on page 181 lists the number of triangles and triples that are used by the $C(G)$ equations (Table 26 on page 177). The results of evaluating the sample graph with all R language igraph and sna options are shown in Table 28 on page 182.

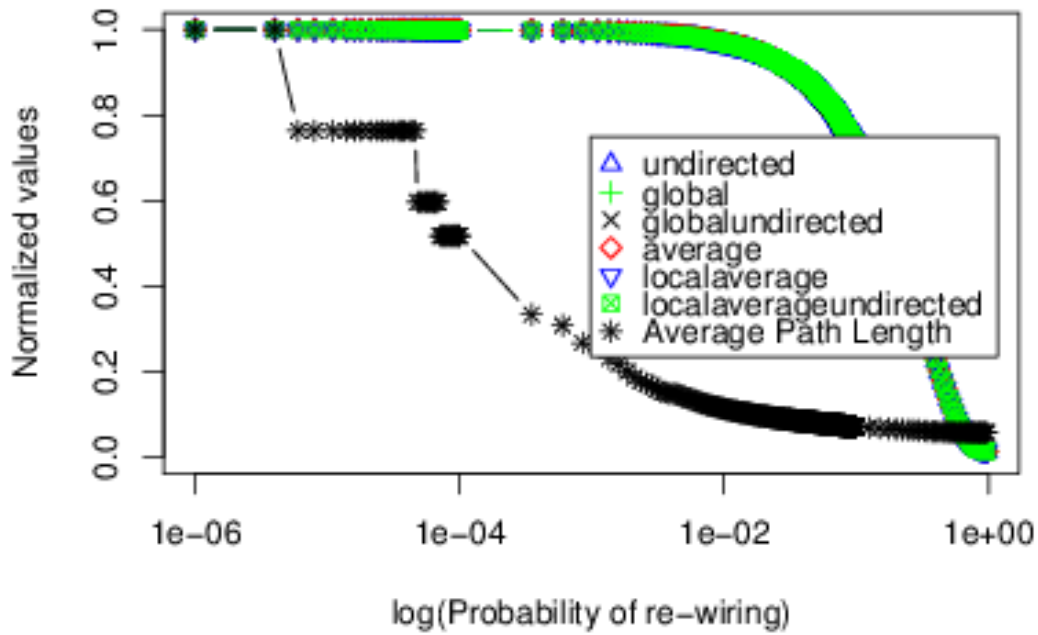


Figure 56. Watts - Strogatz small-world graph analysis (normalized data).

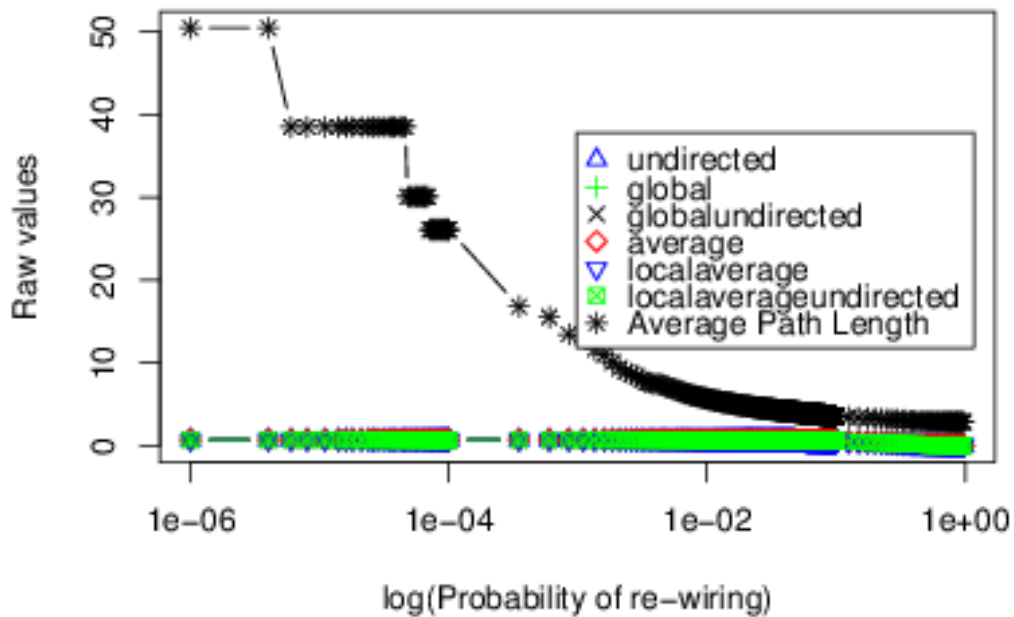


Figure 57. Watts - Strogatz small-world graph analysis (raw data).

Table 27. Number of triangles and triples per node from the test graph.

Node	Num. of Triangles per Node	Num. of Triples per Node	$\frac{\Sigma \text{Triangles}}{\text{Triples}}$
1	1 — {2, 1, 3}	1 — {2, 1, 3}	$\frac{1}{1} = 1$
2	1 — {1, 2, 3}	1 — {1, 2, 3}	$\frac{1}{1} = 1$
3	1 — {1, 3, 2}	6 — {1, 3, 2}, {1,3,4}, {4,3,5}, {1,3,5}, {2,3,5}, {2,3,4}	$\frac{1}{6} = 0.166666667$
4	0	0	$\frac{0}{0} = \text{Not a number}$
5	0	0	$\frac{0}{0} = \text{Not a number}$

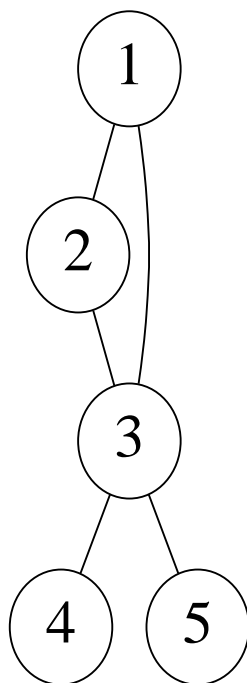
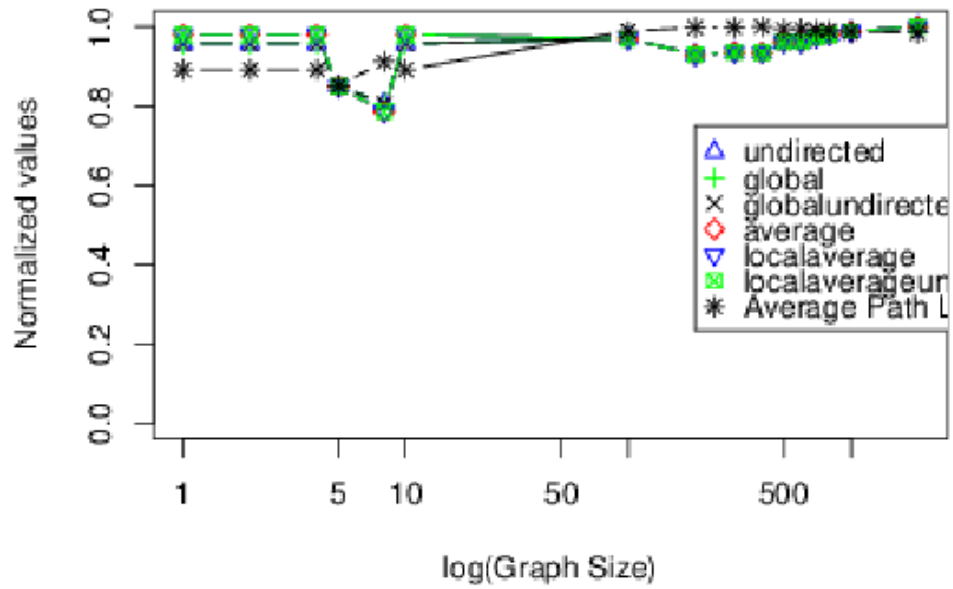


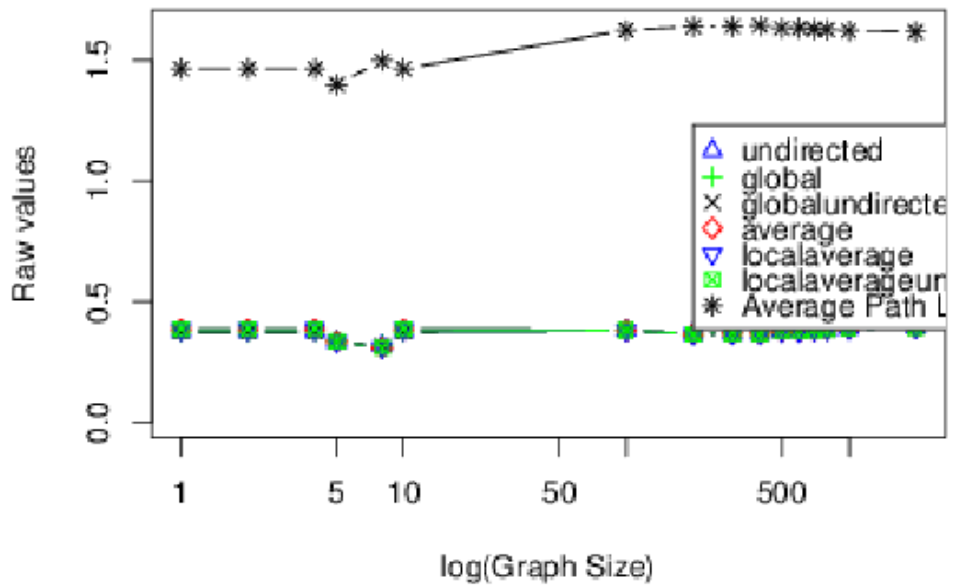
Figure 58. Sample undirected graph used to demonstrate effects of different Clustering Coefficient definitions.

Table 28. Results of exercising the igraph and sna packages with the sample graph.

Package, function control	Underlying Clustering Coefficient equation	Numerical result	Notes
igraph, transitivity(type = ...)			
null	C_{Global}	0.375	
undirected	C_{Global}	0.375	Same as global
global	C_{Global}	0.375	
globalundirected	C_{Global}	0.375	Same as global
localundirected	C_i of $C(G)_{AverageLocal}$	{1, 1, 0.166667, NaN, NaN}	A vector of C_i values used to compute $C(G)_{AverageLocal}$
local	C_i of $C(G)_{AverageLocal}$	{1, 1, 0.166667, NaN, NaN}	A vector of C_i values used to compute $C(G)_{AverageLocal}$
average	$C(G)_{AverageTriples}$	0.7222	
localaverage	$C(G)_{AverageTriples}$	0.7222	
localaverageundirected	$C(G)_{AverageTriples}$	0.7222	
sna, gtrans(measure = ...)			
null	C_{Global}	0.375	
weak	C_{Global}	0.375	The transitive constraint corresponding to: $a \rightarrow b \rightarrow c \Rightarrow a \rightarrow c$
strong	$C(G)_{AverageLocal}$	0.4333	The transitive constraint corresponding to: $a \rightarrow b \rightarrow c \Leftrightarrow a \rightarrow c$
weakcensus	N/A	6	The number of transitive triads used in computing “weak” transitivity.
strongcensus	N/A	26	The number of transitive triads used in computing “strong” transitivity.



(a) Normalized USW data.



(b) Raw USW data.

Figure 59. Average path lengths and clustering coefficients for various sized USW graphs.

Table 29. USW simulation parameters used to generate graphs to be analyzed with igraph. A complete list of simulator command line options are provided in Appendix C on page 368.

Parameter	Value	Notes
β	0.9	The value that the random number has to exceed to connect to the current non-wandering node
γ	60%	Percentage of nodes used after connecting
vertices	Varied	The number of nodes in the system was varied to see the effect of $C(G)$ and average length
1 st node choice	4	Always pick the highest degreed node to start with when looking for a connection
hosts	5000	The number of hosts that the vertices could be spread across

Initially we investigated the USW algorithm with limited β and γ values and a very small graph ($n = 10$). The effects of varying β and γ from 0.0 to 1.0 are shown in Figures 60 on the next page and 61 on page 186. The USW initial node policy chosen was to have each wandering node start with the same initial non-wandering node, node 0 circled in red in Figure 60 on the next page. Holding γ to 0.0 and varying β from 0.0 to 1.0 created graphs that evolved from star to non-closed ring lattices. Increasing γ from 0.0 to 1.0 added a greater and greater number of additional edges to the graph until the graph is an almost fully connected entity.

We investigated larger USW graphs with the command line parameters shown in Table 29. Based on earlier investigations these values seemed to create Small-World graphs. Figures 59(a) on the previous page and 59(b) on the preceding page show the normalized and raw values respectively. In prior work, we had focused on the shape of the $C(G)$ curve as indicating the presence of a small-world being present.

5.6.3 RESULTS AND EVALUATION

Packages igraph and sna can take as input either a Graphviz [181] or a Pajek [182] graph file and then manipulate the resulting data structure. The results of exercising the packages are shown in Tables 28 on page 182. The sna documentation states that the measurement type “strong” is the most commonly used, but does

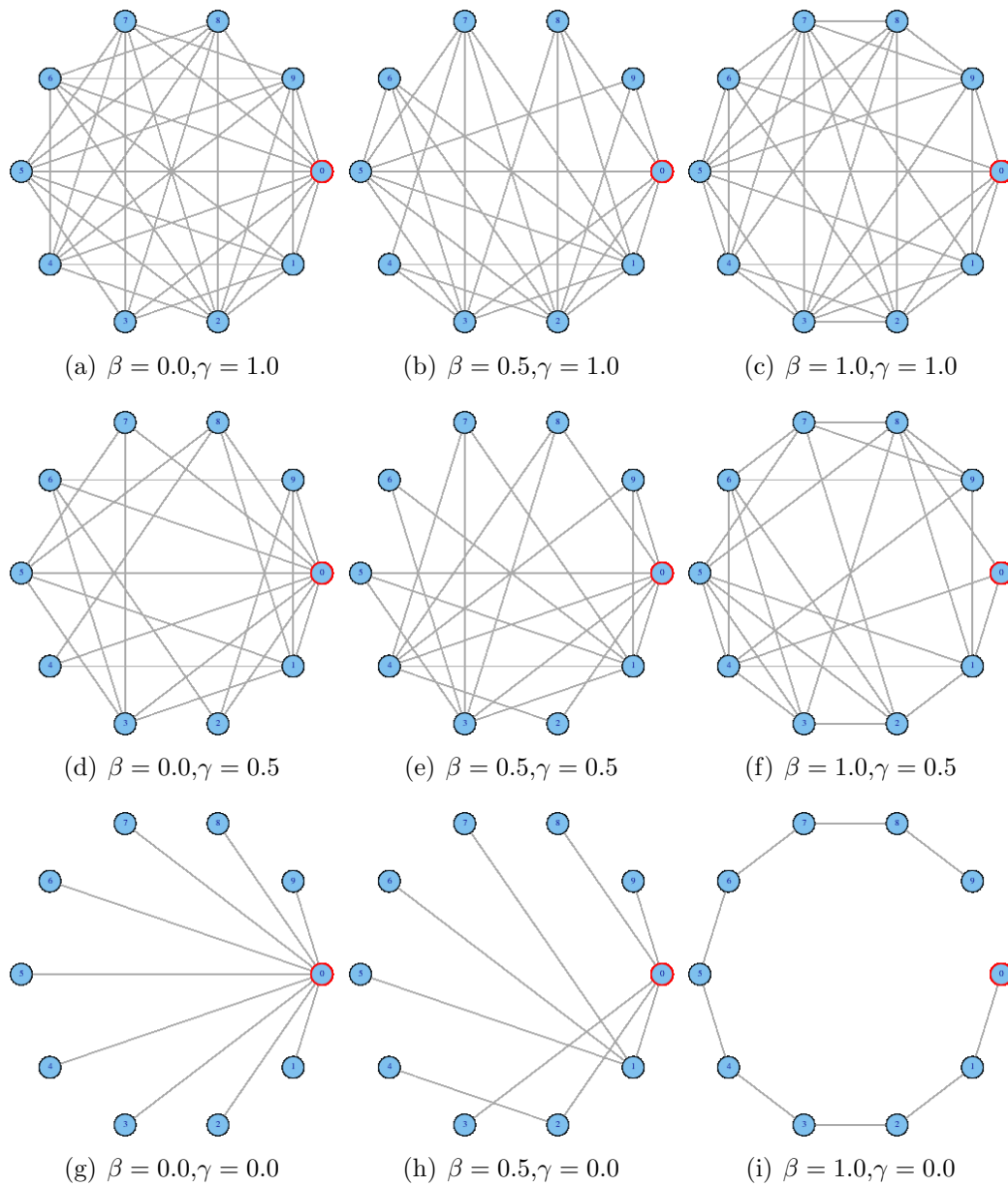


Figure 60. The effect of varying β and γ on a USW graph of size 10. β is the threshold that a random number must exceed for a “wandering” node to attach (make an undirected edge) to a “non-wandering” node. As the “wandering” node attempts to attach to a non-wandering node, it maintains an internal data structure of nodes that it has failed to attach to and nodes that it will attempt to attach to in the future. After the wandering node attaches to a non-wandering node, connections are made to a γ portion of the previously and not yet visited non-wandering nodes. For all runs, the same initial node (node 0, circled in red) was used as the first initial node. When β and γ are 0, the graph takes on a “star” shape. As β increases towards 1.0, the graph becomes more linear.

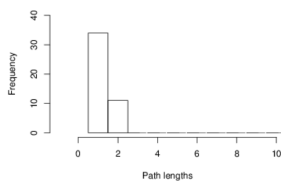


Figure 61. Path length histogram. A more complete set of histograms are in Appendix J on page 509.

not provide any justification for the statement. Both packages reference Wasserman and Faust [183] as their justification for their differing approaches.

Comparing the average $C(G)$ values computed with the custom code to those returned with the igraph transitivity type set to “average,” “localaverage” and “localundirectedaverage.” All values compared exactly out to 4 decimal places. The other values for type agreed out to 3 decimal places. Therefore, probably any could be used and be in close agreement.

This is a quote from Newman comparing the $C(G)$ of structured and random graphs (*In his context, Newman is referring to our C_{Global} or $C(G)_{AverageLocal}$*):

“In general, regardless of which definition of the clustering coefficient is used, the values tend to be considerably higher than for a random graph with a similar number of vertices and edges. Indeed, it is suspected that for many types of networks the probability that the friend of your friend is also your friend should tend to a non-zero limit as the network becomes large, so that $C = O(1)$ as $n \Rightarrow \infty$. On the random graph, by contrast, $C = O(n^{-\infty})$ for large n (either definition of C) and hence the real-world and random graph values can be expected to differ by a factor of order n .”

M. E. J. Newman [178]

Table 30 on the following page summarizes the relationships and regions of values that define lattice, small-world and random graphs. Exact thresholds from one graph type to another do not exist and the transition from one type to another can be very gradual. Small-world region of values for $L(G)$ and $C(G)$ are wide and not exact. The relationship between $L(G)$ and $C(G)$ compared to lattice and random graph values are the real determinates as to whether any particular graph is a small-world

Table 30. Comparing the regions for the expected average path length ($L(G)$) and expected Clustering Coefficient ($C(G)$) for lattice, small-world and random graphs.

Graph type	$L(G)$ (average path length)	$C(G)$ (clustering coefficient)
Lattice	$\frac{N}{2 * K}$	$\frac{3}{4}$
Small-world	$\frac{N}{2 * K} > L(G) > \frac{\ln(N)}{\ln(\langle k \rangle)}$	$\frac{3}{4} > C(G) > \frac{K}{N}$
Random	$\sim \frac{\ln(N)}{\langle k \rangle}$	$\frac{\langle k \rangle}{N}$

[113]. For the purposes of this discussion, the most important thing is that the definition of $C(G)$ be explicitly given and agreed to.

5.6.4 SUMMARY

We will use $C(G)_{AverageLocal}$ for offline analysis of USW graphs, because:

1. It is available in an “off the shelf” package, and
2. It is what Watts and Strogatz used.

We will use custom code for repeated bulk analysis of USW simulated graphs because it is a compiled function versus a scripted function.

5.7 SUMMARY

This dissertation is at the intersection of emergent behavior, graph theory, digital preservation (Figure 12 on page 30). USW algorithm will:

1. Take the emergent behavior tenets and apply them to WOs on the Internet,
2. Create graphs of WOs that are resilient, robust, and autonomous,
3. Communicate effectively and efficiently between one sender and an unknown and unknowable collection of receivers, and
4. Make and create copies of WOs taking into account the update rates of the WO’s aggregate components.

CHAPTER 6

SIMULATION

6.1 INTRODUCTION

We have enumerated and expanded up a set of theories that form the foundation of the USW algorithm (Chapter 5 on page 83). Based upon those theories, we:

1. Identify and describe a small set of USW related policies,
2. Identify and describe a single control variable,
3. Discuss how graph related metrics will change because of the USW algorithm, and
4. Explain the orthogonality of USW WOs and the hosts where they live.

6.2 POLICIES

Policies are courses of action used in different circumstances. The USW algorithm has a set of policies dealing with how to select the first WO, how to select the next WO, how to select friends, how to propagate preservation copies, and so on. Within each policy are a number of different approaches that could be “hardwired” into a WO when it is created. The policy helps guide the behavior of the WO in both “active” and “passive” maintenance roles.

Different policies are in place at different phases during the life of a WO (Figure 62 on the next page).

6.3 CONTROL VALUES

The single most controlling variable in the USW algorithm is β . A secondary variable is γ . We will explain each.

β controls how much of the existing USW graph is “explored” prior to a “wandering” WO making its first connection (Equation 70 on page 190). β is the threshold

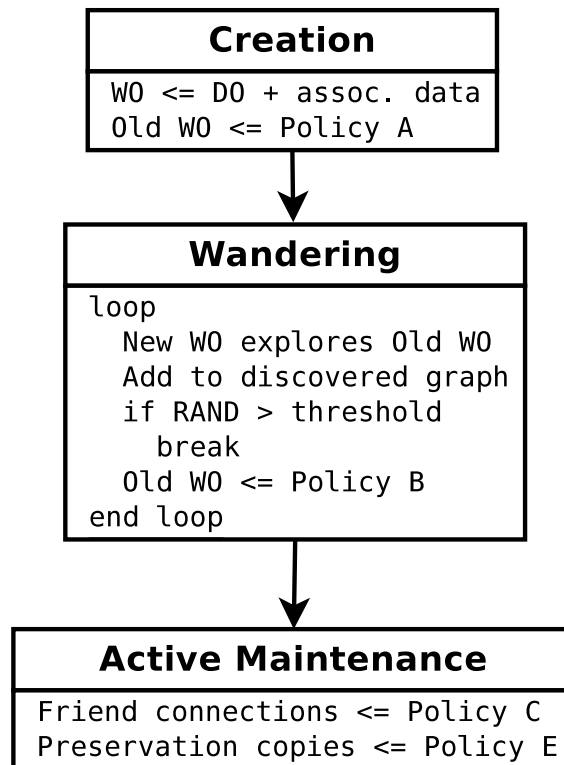


Figure 62. Creation, wandering, and active maintenance phases and policy navigation aide. This is the basic navigation figure that will be annotated when each policy is discussed.

that a locally generated random number must exceed for a wandering WO to make its first connection to an established WO. As the wandering WO attempts to attach to an established WO, it maintains an internal data structure of WOs that it has failed to attach to (the *visitedSet*) and WOs that it will attempt to attach to in the future (the *toBeVisitedSet*). After the wandering WO attaches to an established WO, friend connections are made to a portion of the $visitedSet \cup toBeVisitedSet$. The size of $visitedSet \cup toBeVisitedSet$ is the size of the “discovered” USW graph (Equation 71). The distinction between the “explored” and the “discovered” portions of the USW graph, is that a when a new WO is identified then it becomes part of the “discovered” graph. When a WO is examined and its friend connections are identified, it becomes part of the “explored” graph. If β is a large value less than 1, then the “wandering” WO will have have the opportunity to “discover” many WOs.

$$n_{explored} = \frac{1}{1 - \beta} \quad (70)$$

$$n_{discovered} = \langle k \rangle * n_{explored} \quad (71)$$

γ is a multiplicative factor applied to the size of $visitedSet \cup toBeVisitedSet$ as part of computing how many WOs to make friendship connections to based on the selection method.

At a macro level, β determines how much of the USW graph will be explored and γ determines how much of the discovered graph is remembered.

6.4 COMPARISON OF USW AND RANDOM GRAPHS

We exercised the USW algorithm and compared its results ($C(G)$ and $L(G)$) with a random graph with the same number of vertices and edges (n and $|E|$). The random graphs were created using the *erdos.renyi.game()* function from the R igraph library [78]. We compared and tabulated the results (Table 31 on the following page). Raw and normalized data are plotted in Figures 63 on page 192 and 64 on page 192. Examination of the normalized data (Figure 64 on page 192), validates that the USW algorithm creates graphs that meet the requirements of a small-world, where:

$$C(G)_{USW} \gg C(G)_{random} \quad (72)$$

$$L(G)_{USW} \approx L(G)_{random} \quad (73)$$

Table 31. Comparing USW and random graph metrics.

Vertices	Edges	Avg. Deg.	USW APL	USW CC	Rnd. APL	Rnd. CC
50	275	11.000	1.776	0.358	1.849	0.211
100	623	12.460	1.874	0.266	2.067	0.124
200	1,379	13.790	1.931	0.193	2.295	0.072
300	2,148	14.320	1.952	0.149	2.434	0.045
400	2,910	14.550	1.964	0.121	2.534	0.036

6.5 GROWTH FUNCTION THEORETIC

Any discussion about USW growth at time i is predicated on the conditions at time $i - 1$ (see Equations 74 through 80).

$$G(V, E) \equiv \text{A USW graph exists} \quad (74)$$

$$\text{(perhaps with 0 WOs or 0 edges).} \quad (75)$$

$$n_{(i-1)} \equiv \text{orderUSWgraph}(|V(G)|) \quad (76)$$

$$\langle k_{(i-1)} \rangle \equiv \text{averagedegree} \quad (77)$$

$$\rho(G)_{(i-1)} = \frac{n_{(i-1)} \langle k_{i-1} \rangle}{n_{(i-1)}(n_{(i-1)} - 1)} \quad (78)$$

$$C(G)_{Average}(i - 1) \equiv \text{averageclusteringcoefficient} \quad (79)$$

$$= \frac{|E(G)|}{n_{(i-1)}(n_{(i-1)} - 1)} \quad (80)$$

6.5.1 CURRENT CONDITION

After a number of WOs have been added to the USW graph, the current condition is established and one WO is added to the system. The state of the system changes

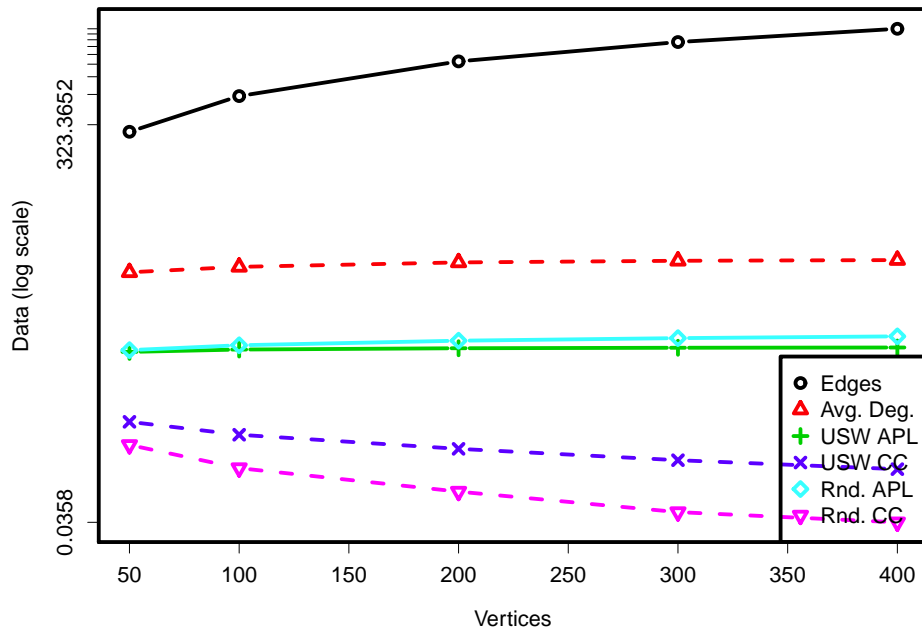


Figure 63. Comparing USW and random graph metrics (raw data).

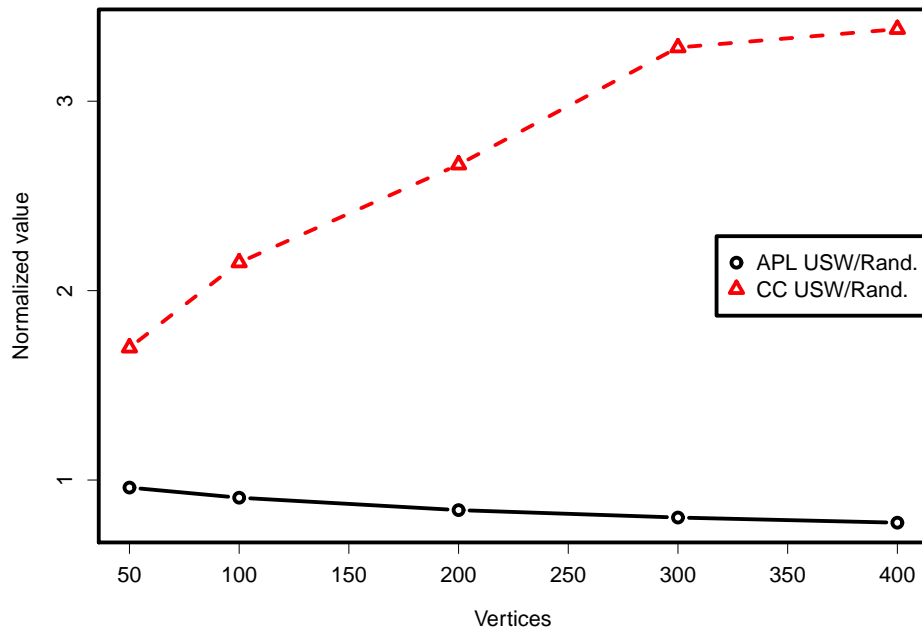


Figure 64. Comparing USW and random graph metrics (normalized data).

(see Equations 81 through 88).

$$n_i = n_{(i-1)} + 1 \quad (81)$$

$$n_{explored} = \frac{1}{1 - \beta} \quad (82)$$

$$n_{discovered} = n_{explored} * \langle k \rangle_{(i-1)} \quad (83)$$

$$1 \leq x \leq n \quad (84)$$

$$k_{added} = f(\gamma, x) \quad (85)$$

$$1 \leq k_{added} \leq x \quad (86)$$

$$\langle k \rangle_i = \langle k \rangle_{(i-1)} + \frac{k_{added}}{n_i + 1} \quad (87)$$

$$C(G)_{Average(i)} = \frac{|E(G)| + k_{added}}{n_i * (n_i + 1)} \quad (88)$$

Where:

1. $n_{explored}$ (Equation 82) is the number of *established* WOs contacted by the *wandering* WO,
2. $n_{discovered}$ (Equation 83) is the number of *discovered* WOs,
3. k_{added} (Equation 85) is the number of edges added to the system based on γ and the size of the discovered graph. k_{added} is bounded between 1 and the $n_{discovered}$,
4. $\langle k_i \rangle$ grows by the addition of number of new edges divided by the new number of WOs.

6.5.2 EFFECT OF ADDITIONAL WOS

After WOs have been added to the system, graph metrics change (see Equations 89 on the next page through 92 on the following page). The graph continues to grow and evolve as more and more WOs are added. The addition of a single WO results in at least 1 undirected edge being added to the system, and that incremental amount is added to the current $\langle k \rangle$ (Equation 90 on the next page) resulting in an every increasing $\langle k \rangle$. An increasing $\langle k \rangle$ means that the average density $\rho(G)$ increases as well (Equation 92 on the following page).

$$n_i \equiv \text{orderUSW graph} \quad (89)$$

$$\langle k_i \rangle = \langle k_{i-1} \rangle + \frac{k}{n_i + 1} \quad (90)$$

$$\rho(G)_i = \frac{n * \langle k_i \rangle}{n * (n - 1)} \quad (91)$$

$$= \frac{\langle k_i \rangle}{n - 1} \quad (92)$$

6.5.3 GROWTH THEORETIC SUMMARY

USW graph density is directly dependent on β and over time will increase towards 1. The USW growth function $f(\gamma)$ controls how fast $\rho(G)$ will increase. $L(G)$ is indirectly dependent on β and will decrease over time to 2 (Equation 95).

$$\rho(G) \nearrow \text{ is directly dependent on } \beta \text{ will increase over time towards } 1 \quad (93)$$

$$C(G)_{Average} \text{ follows the same path as } \rho(G). \quad (94)$$

$$L(G) \searrow \text{ will decrease over time towards } 2. \quad (95)$$

6.6 ORTHOGONALITY OF FAMILIES, FRIENDS, AND HOSTS

The USW logically consists of connected WOs called “friends,” and collections of preservation WOs called “families.” These connected WOs sets share connections, but they are not connected via the traditional HTTP navigational links. WOs can not exist in a vacuum. All WOs require an Web Infrastructure (WI) for persistent storage and communication mechanisms. The WI provide hosts on which WO can live. A host may have one or many WOs. The relationship of WOs and hosts can be thought of as USW layers (Figure 65 on the following page and Figure 66 on page 196).

The upper limit of preservation copies that a WO can create is dependent on the number of friend connections that the WO has (see Section 6.3 on page 188) and the number of unique hosts that those friends reside upon. For example, if WO #4 and WO #5 both reside on host #3 (Figure 67 on page 197), then they can not

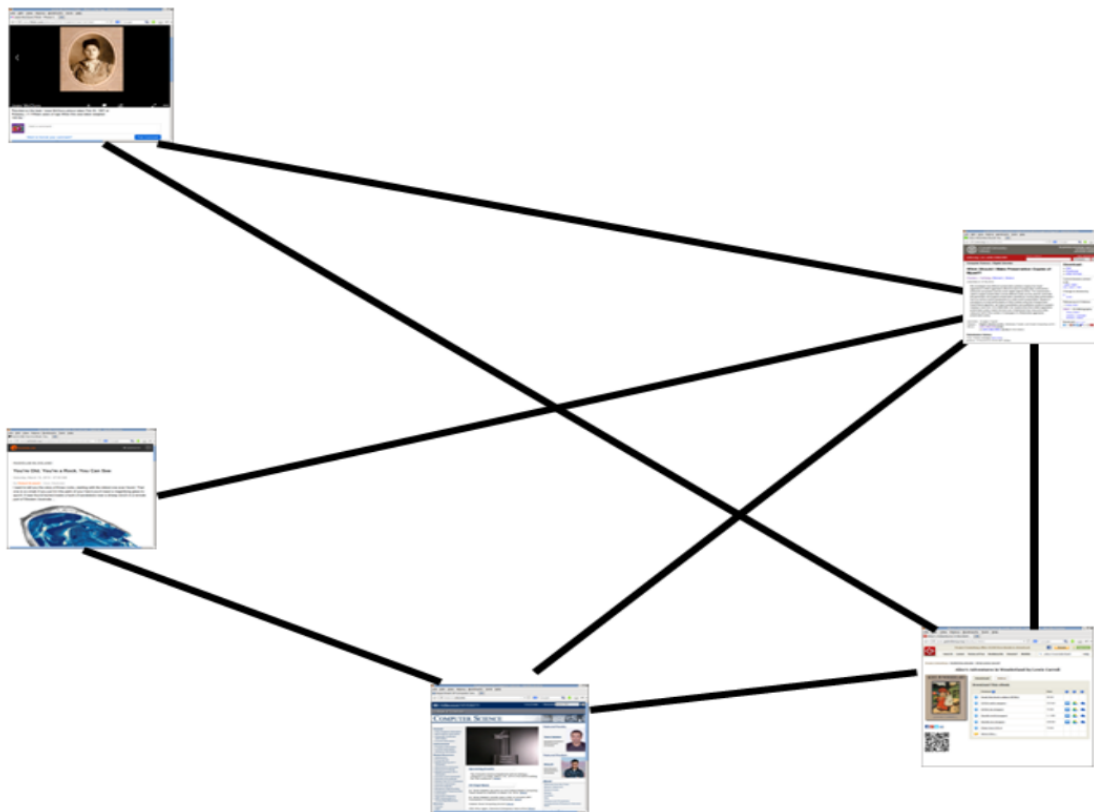


Figure 65. USW WO “friendship” links. A WO may have many friends. Friendship links are shown as solid black lines. Friendship links are not the same as HTML links.

make preservation copies on that host, regardless of their friendship links.

6.7 DESIGN DECISIONS

The purpose of the USW algorithm is to allow for WOs to engage in the long-term preservation of digital data, without the requirement of continuous human interaction. Critical to solving this problem is the idea that the individual USW components be self-reliant and self-organizing. A number of design decisions have to be made to meet the self-organizing goals. In the following sections, we identify and investigate a number of these design questions and present various alternatives to each major design question.

6.7.1 CHOOSING THE INITIAL WO

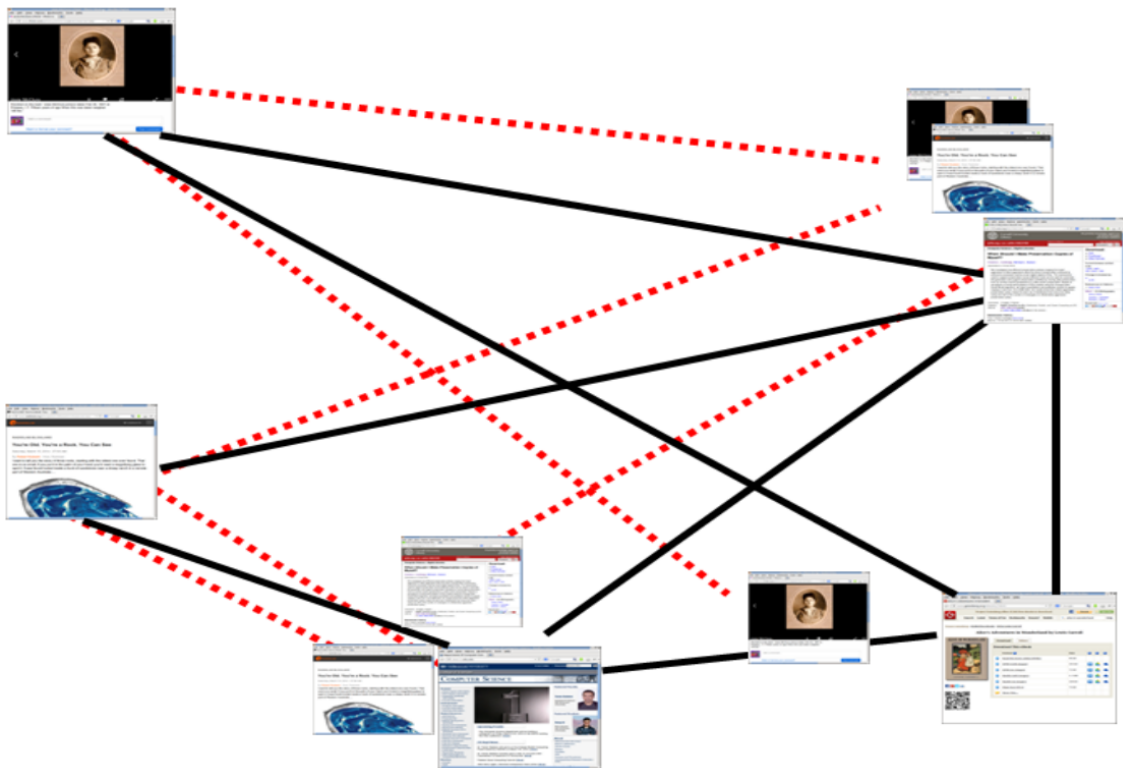


Figure 66. USW WO families. A WO may have many family members. Family links are shown as dashed red lines.

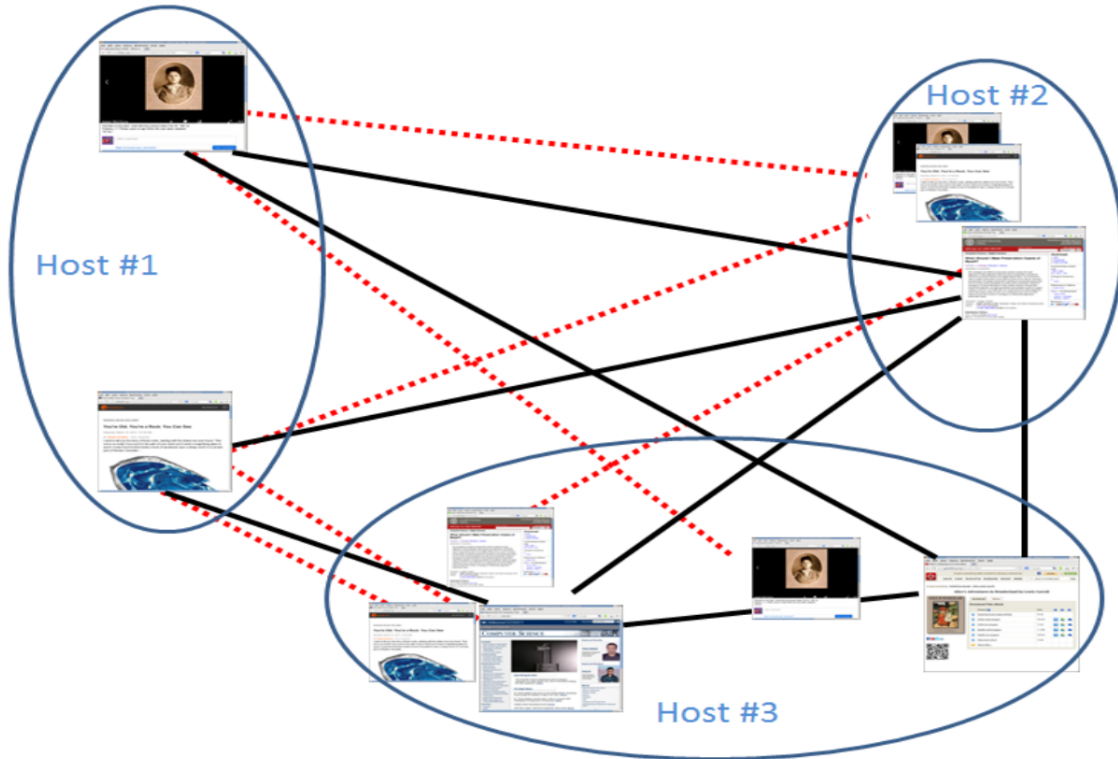


Figure 67. USW hosts. Hosts are shown as solid blue ovals.

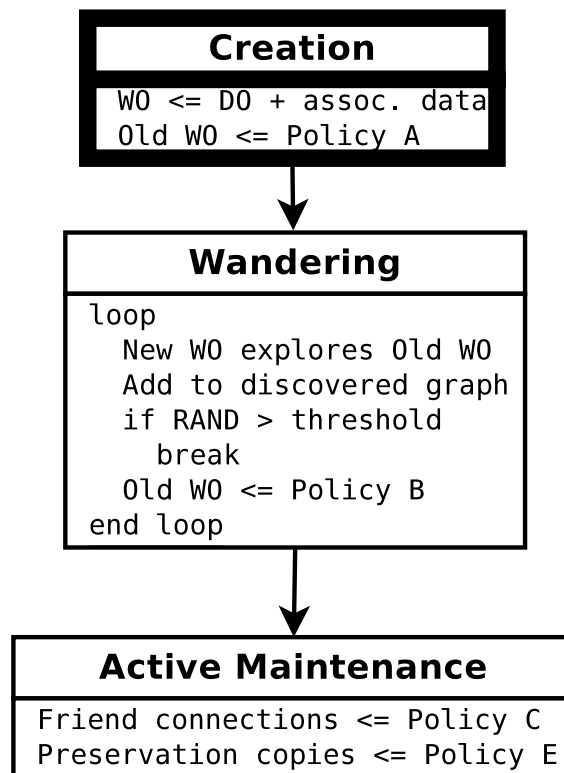


Figure 68. Creation, wandering, and active maintenance phases and policy A.

The initial WO is the WO that a “wandering” WO is introduced to when trying to find a connection in the USW graph. The initial WO is selected via some mechanism outside the USW algorithm (Figure 68 on the preceding page). This is a list of possible approaches:

1. *Use a “well-known” WO:* The same WO is used as the initial WO for all “wandering” WOs. If all USW implementations were to use the same “well-known” WO, then it is likely that a power law graph will start to grow, vice the desired small-world graph. Using a “well-known” WO is easy to implement.
2. *Random selection:* An initial WO is selected at random from all available WOs in the graph. This approach requires that the external process used to select the initial WO have total knowledge of the current USW graph. As the graph becomes larger over time, the effort and energy to explore the entire graph may become excessive.
3. *Preferential attachment:* The existent USW graph is explored and the ordering of the WOs (based on some criteria) is used for determining the initial WO. A preferential attachment example based on “degreeness” is as follows:
 - (a) The USW graph is explored and a list of WOs that have been discovered and visited is maintained in a *visited list*.
 - (b) For each WO in the *visited list*, some collection of metrics (such as degreeness) is maintained.
 - (c) One WO is selected based on:
 - i. The highest degree (the rich get richer [184]), or
 - ii. The lowest the degree, or
 - iii. Some probabilistic distribution.

6.7.2 CHOOSING THE NEXT WO

When the “wandering” WO encounters a new candidate attachment WO, it will explore the WO and acquire a list of WOs that the candidate is currently attached to. This action will increase the WO’s “discovered” portion of the total USW graph. These WOs may become part of the list of “to be visited” WOs that the “wandering” WO will visit, if it does not attach itself to the current candidate WO (Equation 98).

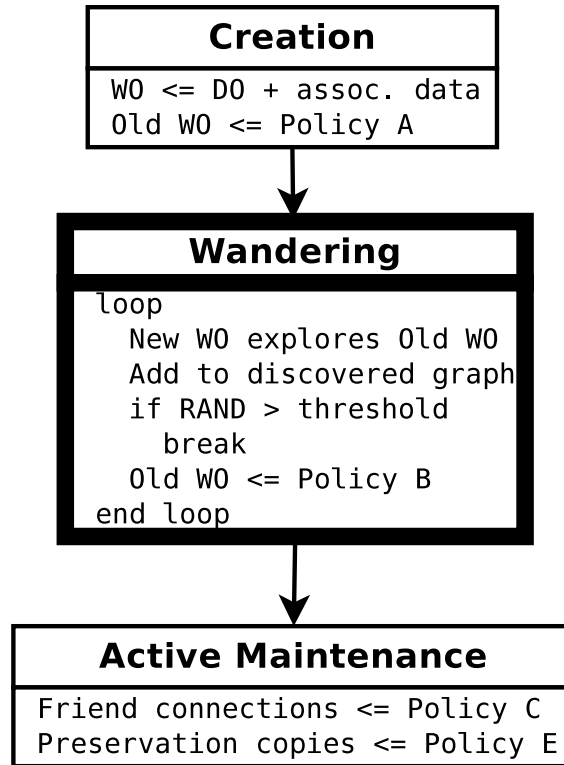


Figure 69. Creation, wandering, and active maintenance phases and policy B.

$$\{WOset\} \equiv \text{WOs connected to discovered WO} \quad (96)$$

$$NewlyDiscoveredSet = \{WOset\} \setminus visitedSet \cup toBeVisitedSet \quad (97)$$

$$toBeVisitedSet = toBeVisitedSet \cup \{NewlyDiscoveredSet\} \quad (98)$$

The set of WOs are added to the existing set of “to be visited” WOs is in accordance with one of several possible approaches (Figure 69) (Equation 98).

This is a list of possible approaches when selecting the next WO to be visited:

1. *FIFO processing*: the *NewlyDiscoveredSet* is appended to *toBeVisitedSet* and the next WO to be visited is at the head of the *toBeVisitedSet* list.
2. *LIFO processing*: the *NewlyDiscoveredSet* is prepended to *toBeVisitedSet* and the next WO to be visited is at the end of the *toBeVisitedSet* list.

3. *Random processing*: the *NewlyDiscoveredSet* is appended to *toBeVisitedSet* and the next WO to be visited is selected at random from the *toBeVisitedSet* list.

6.7.3 DECIDING HOW MANY CONNECTIONS TO MAKE

Discussion

The “wandering” WO discovers information about the USW graph, including building a list of the WO that it has explored and discovered. At the end of its “wandering” phase, it will make connections to some number of the WOs it has discovered. There are a number of different ways to compute how many of these connections to make. Some of these approaches are evaluated below (Equation 99).

$$\text{friendsToBe} = \begin{cases} n * \gamma & \text{if selection method} = 1 \\
 \max(1, \ln(n * \gamma)) & \text{if selection method} = 2 \\
 \max(1, \ln(n) * \gamma) & \text{if selection method} = 3 \\
 \max(0, \ln(n * \gamma)) & \text{if selection method} = 4 \\
 \max(0, \ln(n) * \gamma) & \text{if selection method} = 5 \\
 \max(1, \log_2(n * \gamma)) & \text{if selection method} = 6 \\
 \max(1, \log_2(n) * \gamma) & \text{if selection method} = 7 \\
 \max(0, \log_2(n * \gamma)) & \text{if selection method} = 8 \\
 \max(0, \log_2(n) * \gamma) & \text{if selection method} = 9 \\
 5 + \log_2(n * \gamma) & \text{if selection method} = 10 \end{cases} \quad (99)$$

The evaluation was based on creating a USW graph with the same creation parameters ($n = 75, \beta = 0.85, \gamma = 0.10$) and the selection method was varied. Table 32 on the next page has the data resulting from using a “constant” (i.e., well known) established WO and shows selected graph characteristics as the selection method changed. For each resulting graph, a degree distribution histogram (of *in*, *out*, and *combined* degrees) is created. Table 33 on page 202 has the data resulting from selecting a “random” WO as the established WO and shows selected graph characteristics as the selection method changed. Table 34 on page 203 has the data resulting from using the last added WO as the established WO and shows selected graph characteristics as the selection method changed.

Table 32. Summary of various graph characteristics ($n = 75, \beta = 0.85, \gamma = 0.10$) as a function of selection method and constant *established* WO. The USW algorithm requires an “established” WO that all other WOs are initially introduced to. Therefore, there is an initial WO and then 75 new WOs are introduced. A USW graph may not have a total of 76 WOs due to the fidelity of the prototype environment (e.g., HTTP timeouts or other systemic problems). On average, it takes about 12 events to create an edge and the system is averaging 111 events per minute (≈ 1.85 events per second).

	Selection method									
	1	2	3	4	5	6	7	8	9	10
$C(G)$	0.09	0.065	0.068	0.052	0.000	0.104	0.066	0.104	0.000	0.304
$L(G)$	2.660	2.092	2.141	2.061	2.058	2.074	2.100	2.022	2.057	1.819
Time(min.)	29	29	27	25	14	48	27	30	13	101
Events	3,234	3,332	3,100	2,698	1,638	4,151	3,143	3,560	1,604	11,287
Min(k)	1	2	2	1	1	2	2	1	1	6
$\langle k \rangle$	3.7	3.7	3.6	3.1	1.9	4.5	3.5	4.1	1.9	13.6
Max (k)	24	67	60	68	60	67	66	68	61	75
Std. dev. (k)	4.4	7.6	6.9	7.8	7.3	7.5	7.4	7.8	7.3	8.6
Actual. WOs	73	76	75	73	66	76	76	72	67	76
Edges	269	284	267	226	127	343	269	295	129	1037
Degree hist. fig.	139	140	141	142	143	144	145	146	147	148

All of the *log* graphs showed one, or possibly two WOs that were extremely well-connected. So well-connected, that the effort to slow the connection growth effectively transferred all the growth to a single WO. If the user were to attempt to service this well-connected WO, then all the concerns enumerated in the Introduction would come to the front.

This connectedness may be the result of the way the USW graphs were created. To wit:

1. All WOs used the same initial established WO to start the USW algorithm. This mimics using a “well-known” WO as the starting point. If the starting WO was selected at random (requiring knowledge of the entire USW graph) then results may be different.
2. All WOs used the same series of random numbers. Random numbers are used

to determine when to make the first USW WO connection (thereby beginning the transition from “wandering” to “established”) and for selecting the initial set of WOs to form “friendship links” with. Having a different series of random numbers would make a difference in the USW graph.

Table 33. Summary of various graph characteristics ($n = 75, \beta = 0.85, \gamma = 0.10$) as a function of random selection method to select the *established* WO. The USW algorithm requires an “established” WO that all other WOs are initially introduced to. Therefore, there is an initial WO and then 75 new WOs are introduced. A USW graph may not have a total of 76 WOs due to failures of the prototype environment (e.g., HTTP timeouts or other systemic problems). On average, it takes about 12 events to create an edge and the system is averaging 111 events per minute (≈ 1.85 events per second). The *established* WO was selected from all the WO currently in the USW graph when the *wandering* WO as introduced.

	Selection method									
	1	2	3	4	5	6	7	8	9	10
$C(G)$	0.070	0.008	0.008	0.000	0.000	0.040	0.040	0.008	0.006	0.000
$L(G)$	2.966	2.434	2.434	3.832	3.690	2.684	2.684	2.434	3.269	3.690
Time(min.)	92	85	173	69	67	93	97	72	64	205
Events	3,646	3,908	3,908	2,900	2,802	4,002	3,805	3,158	2,802	8,737
Min(k)	1	2	2	1	2	1	4	2	2	1
$\langle k \rangle$	3.5	3.9	3.9	2.1	1.9	8.1	4.1	3.9	2.7	1.9
Max (k)	18	37	37	15	21	33	33	37	20	21
Std. dev. (k)	2.7	5.6	5.6	2.6	3.4	4.7	4.7	5.6	3.0	3.4
Actual. WOs	76	75	75	76	76	75	75	75	76	76
Edges	264	290	290	163	148	304	304	290	202	148
Degree hist. fig.	149	150	151	152	153	154	155	156	157	158

Thoughts on random selection

The question about which WO would be chosen and how often it would be chosen as the USW grows has interesting and unexpected aspects. As the graph grows, each WO has exactly the same probability of being selected as the *established* WO as any other. Table 35 on the following page shows the probability that any of the 4 WOs in the USW will be selected as the graph grows from 1 WO to 4. When there is 1 WO, then the probability that it will be selected is 1. When there are two WOs,

Table 34. Summary of various graph characteristics ($n = 75, \beta = 0.85, \gamma = 0.10$) as a function of using the last previously entered WO as the *established* WO.

	Selection method									
	1	2	3	4	5	6	7	8	9	10
$C(G)$	0.094	0.005	0.004	0.0000	0.000	0.000	0.004	0.000	0.000	0.353
$L(G)$	2.835	2.412	2.413	7.936	7.969	7.969	2.412	7.969	7.943	1.913
Time(min.)	91	84	94	60	62	50	89	61	62	201
Events	4,197	3,944	3,931	2,790	2,797	4,099	3,934	2,797	2,784	8,382
Min(k)	1	2	2	1	1	1	2	1	1	6
$\langle k \rangle$	4.4	3.9	3.9	2.0	2.0	2.0	3.9	2.0	1.9	11.2
Max (k)	30	38	38	5	5	5	38	5	5	25
Std. dev. (k)	3.9	5.7	5.7	1.7	1.7	1.7	5.7	1.7	1.6	4.9
Actual. WOs	76	76	76	76	76	76	76	76	76	76
Edges	334	295	293	149	150	150	294	150	148	855
Degree hist. fig.	159	160	161	162	163	164	165	166	167	168

then each as a probability of $\frac{1}{2}$, and so on. When the graph has reached 4 WOs, the first WO will have been selected 52% of the time.

The behavior of WO #1 is interesting and is in fact a harmonic series (Equation 100) [185]:

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} = \sum_{i=1}^n \frac{1}{i} \quad (100)$$

The harmonic series displays interesting characteristics when evaluated to see which WOs are selected most often. Figures 70 on page 205 through 73 on page 208 show that WO#1 has a full harmonic series sum and each WO after that is missing their respective first harmonic series terms. Each figure has two Y-axes. The left-hand axis is the harmonic value of the WO. The right-hand axis is the cumulative probability density (CPD) function of the individual WOs harmonic number normalized to the sum of all the harmonic numbers. Additionally, each figure has horizontal lines at 50% and 75% CPD intersecting the CPD curve and then vertical lines showing where that CPD percentage is met on the X-axis. As the USW graph grows larger, the 50% value tends towards 18% of the order of the USW graph, while the 75% tends towards the 38%.

Table 35. How often a WO is chosen as USW graph grows.

#WO graph	in USW	WO #				Σ
		1	2	3	4	
	1	1				1
	2	$\frac{1}{2}$	$\frac{1}{2}$			1
	3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$		1
	4	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	1
	Σ	2.08	1.08	0.58	0.25	4
	Prob.	0.52	0.27	0.15	0.06	1

Table 36. Trending 50% and 70% based on USW graph size.

Order USW graph	50% WO	% order	75% WO	% order
4	1	0	2	50
100	19	19	39	39
1,000	187	18.7	383	38.3
10,000	1,867	18.67	3,824	38.24
100,000	18,668	18.668	38,241	38.241

Based on the limited test cases, 50% of the randomly selected WOs will be in the first 19% USW WOs (Table 36).

Thoughts on last WO selection

Examining the degree histograms (Figure 149 on page 431 through Figure 158 on page 440), the histogram given by $5 + \log_2(n * \gamma)$ (and shown in Figure 158 on page 440) exhibit the type of degree clustering and $C(G)$ that are typical of “classical” Watts – Strogatz [43] small-worlds.

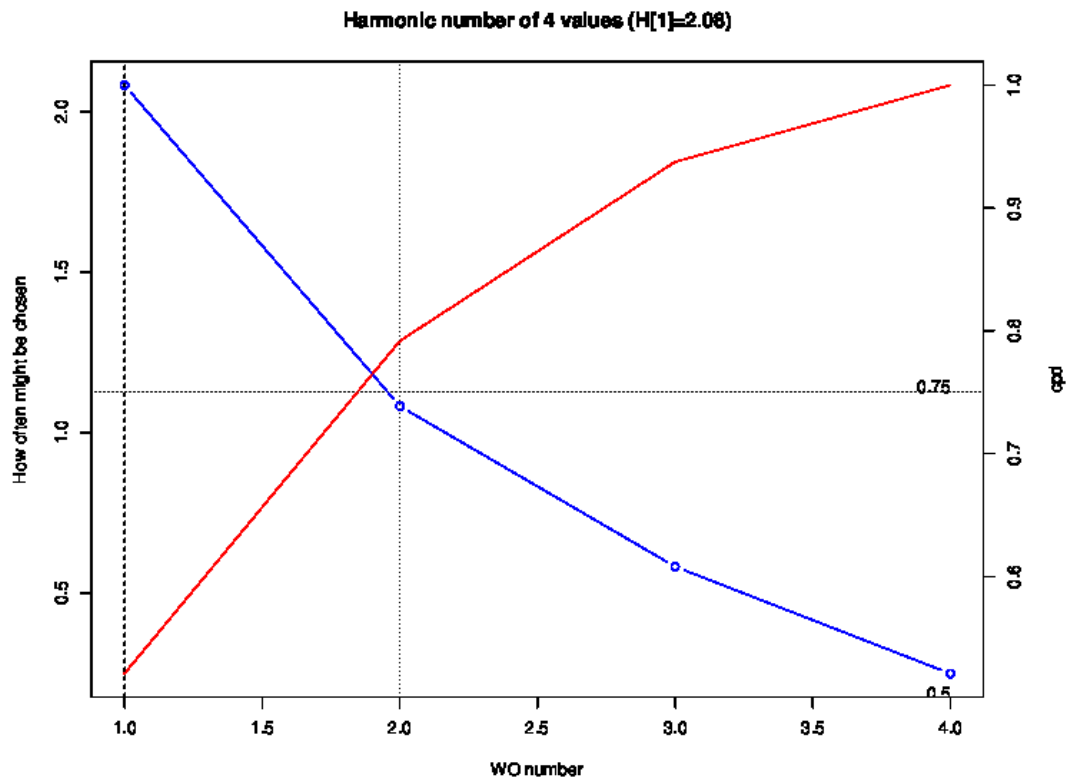


Figure 70. Harmonic series of 4 points showing which WOs are selected 50% and 75% of the time.

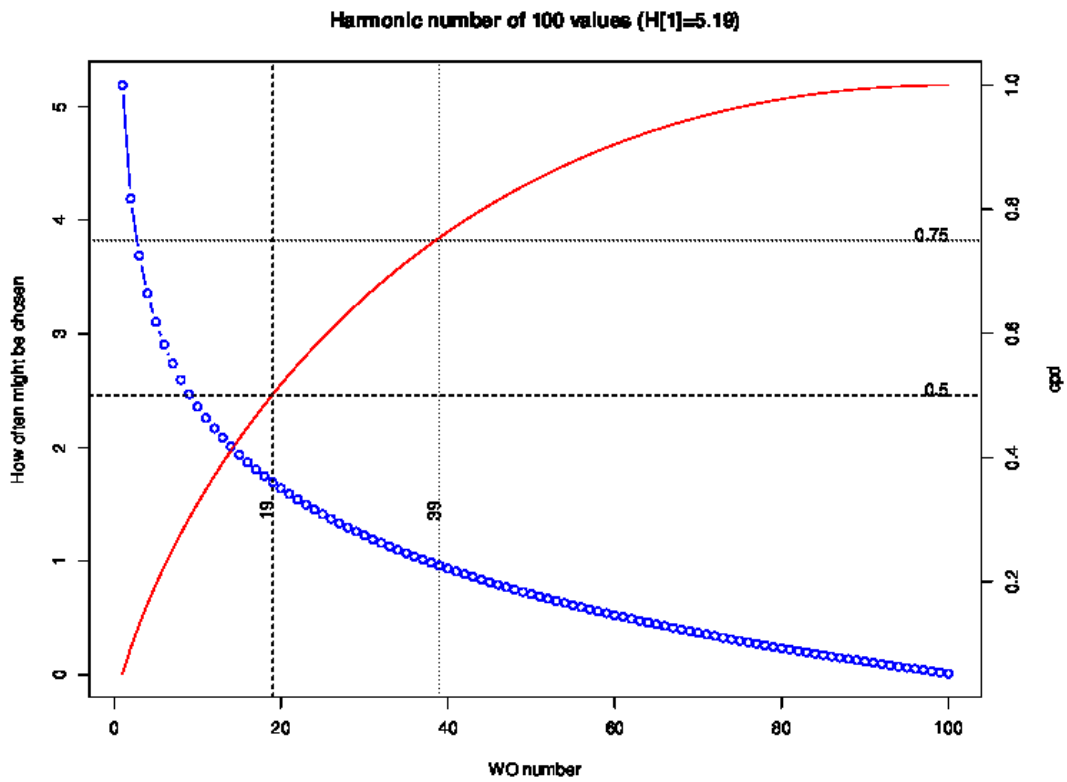


Figure 71. Harmonic series of 100 points showing which WOs are selected 50% and 75% of the time.

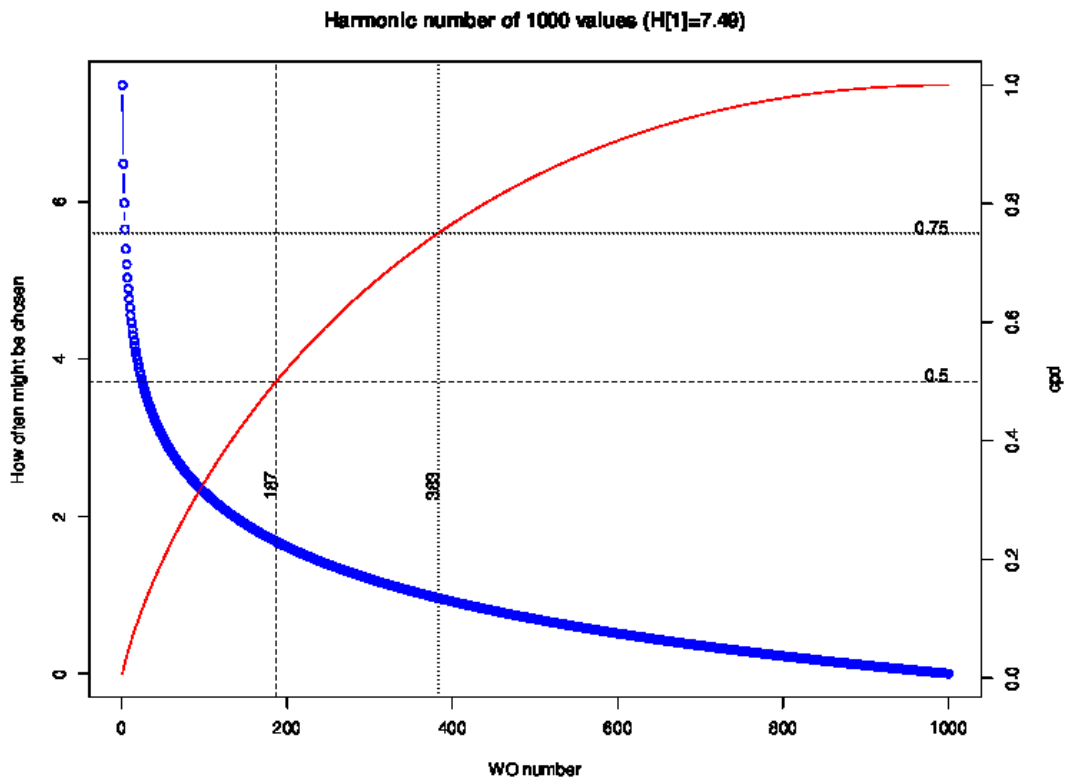


Figure 72. Harmonic series of 1,000 points showing which WOs are selected 50% and 75% of the time.

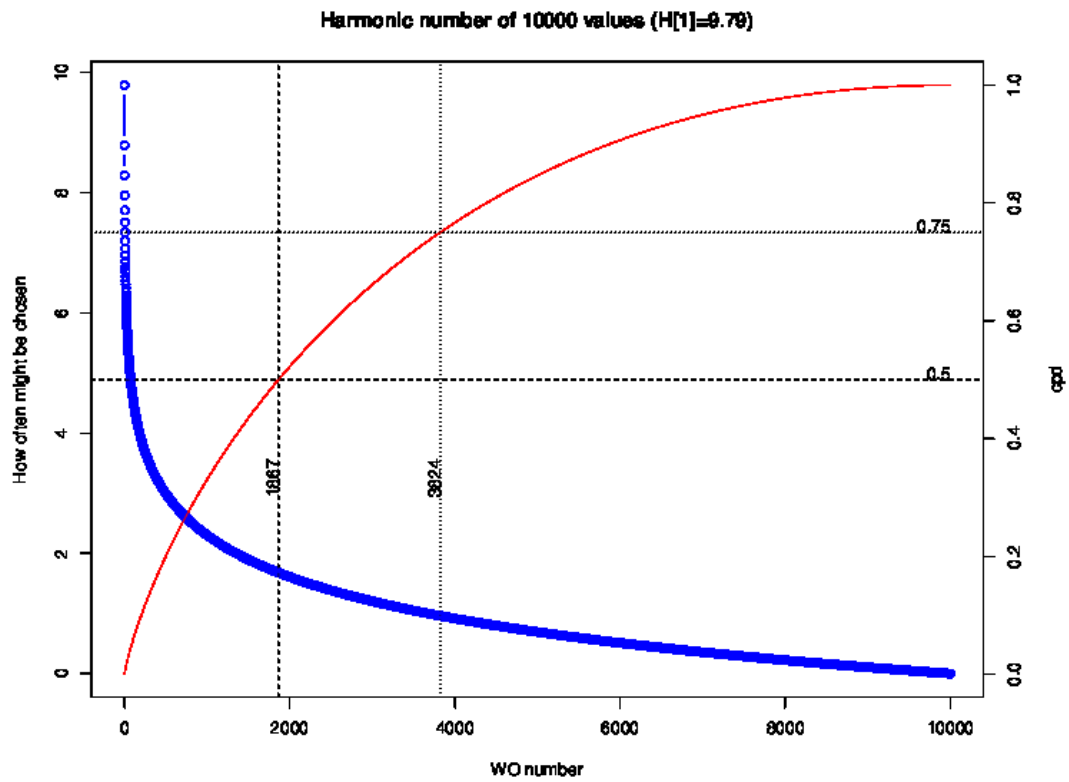


Figure 73. Harmonic series of 10,000 points showing which WOs are selected 50% and 75% of the time.

Evaluation

1. $\langle k \rangle$ is not an adequate descriptor: all of the degree distribution histograms have a very long power law tail. A power law distribution might be more descriptive.
2. Lower log base results in more left-handed skewing: based on the actions of the *max* function. The *max* function returns one of three values:
 - 0 if $\log(n * \gamma)$ is less than 0 (happens when n is less than $\frac{1}{\gamma}$) based on selection method,
 - 1 if $\log(n * \gamma)$ is less than 1 (happens when n is less than $\frac{1}{\gamma}$) based on selection method,
 - $\log(n * \gamma)$ otherwise.
3. *Artificial offset* results in a much denser graph (as in many edges relative to the other approaches) and more *friends* per WO. Each *friends* means that the User Interface in a fielded system will take more time to process all *friends*.
4. *Using the last WO* and the *artificial offset* results in a graph in more general keeping with a “classical” small-world.

6.7.4 CHOOSING CONNECTIONS

After the “wandering” WO has made a connection to the candidate WO, it may be left with WOs in its *toBeVisitedSet* and its *visitedSet* lists. Attaching itself to WOs in the *toBeVisitedSet* list is akin to making attachments into the future (because the “wandering” WO would visit these WOs sometime in the future). Attaching itself to WO in the *visitedSet* list is akin to attaching itself to the past (because the “wandering” WO has already visited those WOs before and did not make a connection). We will evaluate different approaches in order to identify which approach is best for selecting which WOs to connect to.

After the number of connections to make has been determined (Equation 99 on page 200), which WOs to select has to be addressed. There are three different lists of WOs from which *friend* connections can be made. The lists are:

1. $\{WOset\}$: WOs connected to the candidate WO,

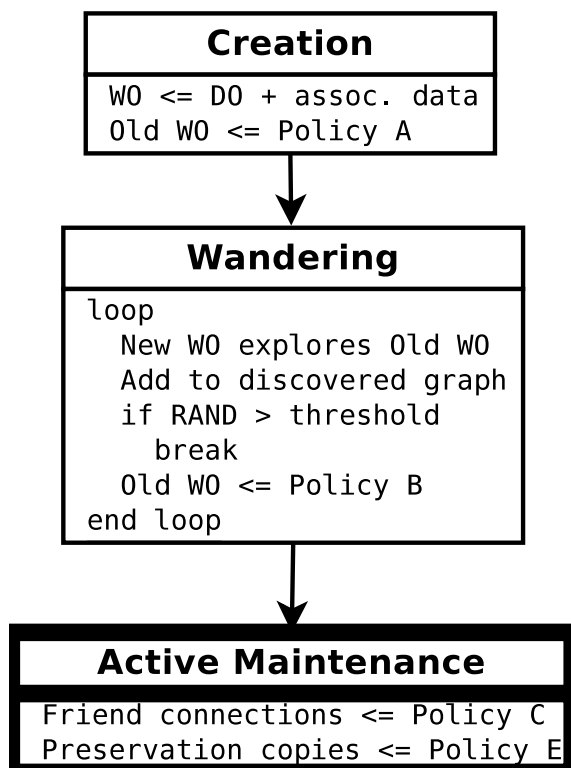


Figure 74. Creation, wandering, and active maintenance phases and policies C and E.

2. *visitedSet*: WOs that the wandering WO has explored, but not connected to
3. *toBeVisitedSet*: WOs that the wandering WO has not explored.

Friendship connections are made to *friendsToBe* WOs using the following approaches:

1. *Randomly*: select *friendsToBe* of WOs in $visitedSet \cup toBeVisitedSet$.
2. *FIFO*: select *friendsToBe* of WOs from $visitedSet \cup toBeVisitedSet$.
3. *LIFO*: select *friendsToBe* of WOs from $visitedSet \cup toBeVisitedSet$.
4. *Preferentially connect* to candidate WO's *friendsToBe* friends from $\{WOset\}$.
If $friendsToBe > |\{WOset\}|$ then the remaining connections are selected based on "random selection."

6.7.5 CHOOSING WHEN TO MAKE A PRESERVATION COPY

An active maintenance WO is responsible for ensuring that enough preservation copies have been created and for ensuring that the copies are spread across unique hosts (Figure 74 on the previous page). When the WO determines that additional preservation copies need to be made to get closer to the maximum number of preservation copies that it was directed to achieve when it was created, there are several different approaches that it can take. During the life of the maintenance WO, it will have many opportunities to make preservation copies. At each of these opportunities, it can be:

1. *Polite* and attempt to find room for a single preservation copy,
2. *Moderately aggressive* and add enough preservation copies to immediately reach the minimum number of copies and after that only add one more at a time to reach the maximum number, or
3. *Aggressive* and attempt to make the maximum number of preservation copies every opportunity.

The *polite* policy keeps the number of communications between the WOs to a minimum, but takes the longest time to reach the maximum. The *moderately aggressive* increases the communication activity until the minimum number is reached

Table 37. Combinatorial explosion of possible USW control parameters. If the step resolution for both β and γ is 0.01, then the table shows the number of combinatorial possible USW algorithm control parameters. If the resolution is made finer, then the number of combinations will increase.

Parameter	Values
Policy A	3
Policy B	3
Policy C	4
Selection method	10
β	$\frac{1}{\text{resolution}}$
γ	$\frac{1}{\text{resolution}}$
Combinations	3,600,000

and after that communication activity drops off. Using the *moderately aggressive* approach, the minimum number is reached quickly, while it takes longer to reach the maximum. The *aggressive* approach maximizes communication at all times and reaches the maximum number of copies faster than the other approaches.

6.8 REDUCING THE PROBLEM SPACE

In Section 6 on page 188, we identified a collection of policies and control parameters that are essential to the USW algorithm. Each policy has a finite number of values, and the control parameters are floating point numbers $[0, 1]$. The combinatorial product of these parameters is too large for practical exploration (Table 37).

The size of the USW control parameter space must be reduced to allow practical testing and analysis. The following sections will address each policy and identify values that meet the small-world criteria for $C(G)_{Average}$ and $L(G)$.

6.8.1 CHOOSING THE INITIAL WO

Policy A deals with how to select the initial WO that a wandering WO is initially introduced to in order to start exploring and discovering the USW graph. We have enumerated a number of different approaches that could be used (see Section 6.7.1 on page 195).

6.8.2 CHOOSING THE NEXT WO

Policy B deals with different ways the wandering WO can select the next WO to be explored from the wandering WO's internal list of WOs it had discovered. We have enumerated a number of different approaches that could be used (see Section 6.7.2 on page 198).

6.8.3 CHOOSING CONNECTIONS

Policy C deals with computing how many WOs the wandering WO should select from its *visitedSet* and *toBeVisitedSet* after the wandering WO has made its first connection. We have enumerated a number of different approaches that could be used (see Section 6.7.4 on page 209).

6.8.4 CHOOSING WHEN TO MAKE A PRESERVATION COPY

We investigate how different replication policies ranging from least aggressive to most aggressive affect the level of preservation achieved by autonomic processes used by smart web objects (WOs). Based on simulations of small-world graphs of WOs created using the Unsupervised Small-World algorithm, we report quantitative and qualitative results for graphs ranging in size from 10 to 5,000 WOs. Our results show that a moderately aggressive replication policy makes the best use of distributed host resources and that the communication costs for selected replication policies only differ by 18% for very small graphs and less than 5% for larger graphs.

Just as there are orthogonal views of a collection of WOs; within a simulation there can be different views of time. In our event driven simulation, a simulation event is equivalent to simulation time $S_e \equiv S_t$. A time slice shows the state of the system after every 10 S_e ($T_{\text{slice}} = 10S_e$). A time step is a regular offset into the simulation after an initial offset ($T_{\text{step}} = S_e3500 + T_{\text{slice}} * \text{step}$). The T_{slice} value was chosen to facilitate analysis of the data from the simulation. The T_{step} offset of 3,500 was chosen because many of the initial events in the simulation dealt with the configuration of internal data structures and not with the actions of the WOs or the hosts.

1. **Least aggressive** — a WO will only make a single replication copy at a time, regardless of how many copies are needed, how many opportunities are available to the WO at a particular time and will continue to make single copies until it reaches c_{hard} .

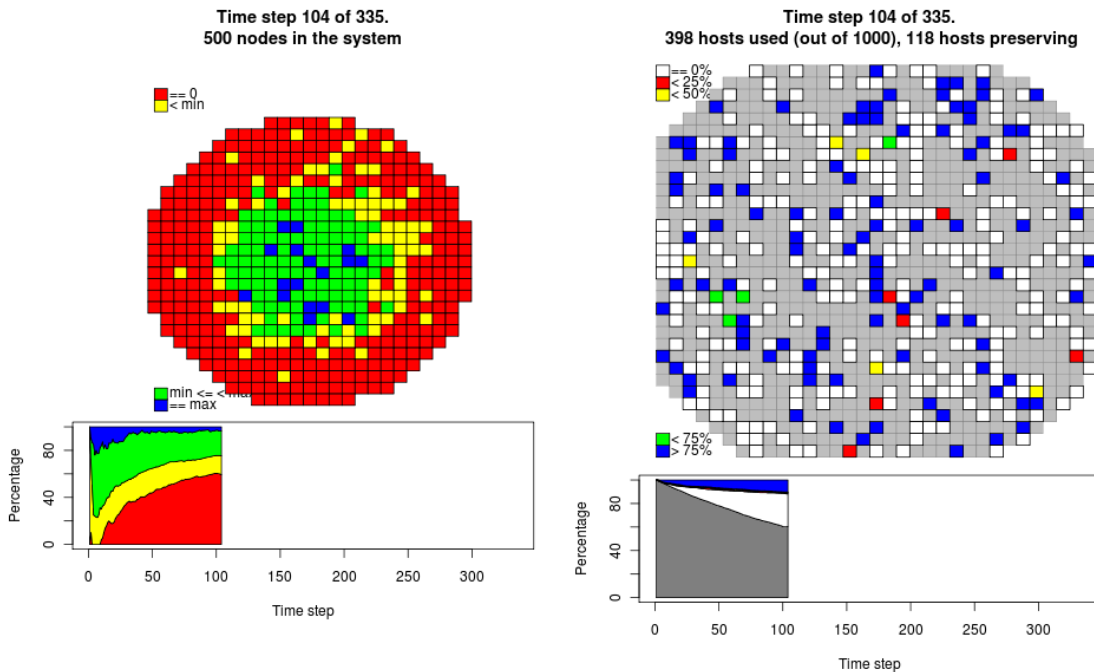


Figure 75. A snapshot of the least aggressive replication policy. WOs are shown on the left and hosts are shown on the right. The colors show the state of the WO's preservation copies, or host's preservation capacity used at the time of the measurement. Under each circular plot is a T_{step} histogram. Above each circular plot is a status line showing T_{step} , how many WOs are in the system or how many hosts are active and preserving data.

2. **Moderately aggressive** — a WO will make as many copies as it can to reach c_{soft} when it makes its first connection, then it will fall back to *least aggressive* policy.
3. **Most aggressive** — a WO will make as many copies as it can to reach c_{hard} when it makes its first connection, then it will fall back to *least aggressive* policy.

The effect of both the moderately and most aggressive replication behaviors is that after reaching their respective goals, they behave like the least aggressive.

Figure 75 on the previous page serves as a legend for the sub-figures in Figures 76 on the following page and 78 on page 219 and shows WO and host replication status as a function of T_{step} . Figure 75 on the preceding page is divided into four areas. The left half of Figure 75 on the previous page shows WO related data, while the right half shows host data. WOs are created sequentially and added to the model.

In Figure 76 on the following page, WOs are added in a spiral fashion starting at the center of the “circular” plot, and newer WOs are plotted in a circular manner from the center. This presentation is much the same as the rings of a tree, in that the oldest are in center and the youngest are on the outer edge.

The preservation status of a WO is approximated by the color assigned to the WO. Initially the WO has $c = 0$ copies and is colored red. As the WO creates copies, the color changes to yellow. When the WO reaches c_{soft} , the color changes to green. When c_{hard} is reached, the WO turns blue. The rules of the model (Table 3 on page 17) permit the killing of one WO’s replication copies for the sake of creating a room for copy of a WO that needs to reach its c_{soft} (i.e., if a $\text{WO}_{i,c,h}$ has more than its c_{soft} and $\text{WO}_{j,c,h}$ has not reached its c_{soft} , then $\text{WO}_{i,c,h}$ will sacrifice one of its copies so that the other WO can move closer to c_{soft}). Sacrificing a preservation copy for the betterment of the whole is the embodiment of *velocity matching*. The effect of this behavior is that a WO can change color from red to yellow to green and then possibly to blue. If the WO changes to blue, it might oscillate between green and blue as its preservation copies oscillate between c_{soft} and c_{hard} . A WO will never sacrifice a copy if it has not exceeded its c_{soft} . The histogram under the WO circular plot shows the percentage of WOs in each of the different preservation copy states as a function of T_{step} .

The preservation utilization status of a host is shown in the right half of Figure 75 on the preceding page. The universe of possible hosts is constant and is represented by the entire right half plot. Hosts that are not being used are shown in grey. The placement of the host in the figure is based on the host’s sequential number in the model. Those hosts that are used are drawn in one of five colors. If the host is used in the model, but is not hosting any preservation copies then it is colored white. If less than 25% of the host’s capacity is used then it is colored red. Similarly, it is yellow if less than 50% is used, green if less than 75% and blue if greater than 75%. The histogram on the host’s side shows the percentage of the hosts that are in any of the particular states.

The model has $n_{\text{max}}=500$, $c_{\text{soft}}=3$, $c_{\text{hard}}= 5$, $h_{\text{max}} = 1000$, $h_{\text{cap}} = 5$. The model runs until it reaches a steady state. A steady state is defined as: all WOs are unable to locate candidate hosts on which to store preservation copies. Steady state is reached at different times based on the replication policy. In all cases, all n_{max} WOs have been introduced into the model by $T_{\text{step}} = 100$.

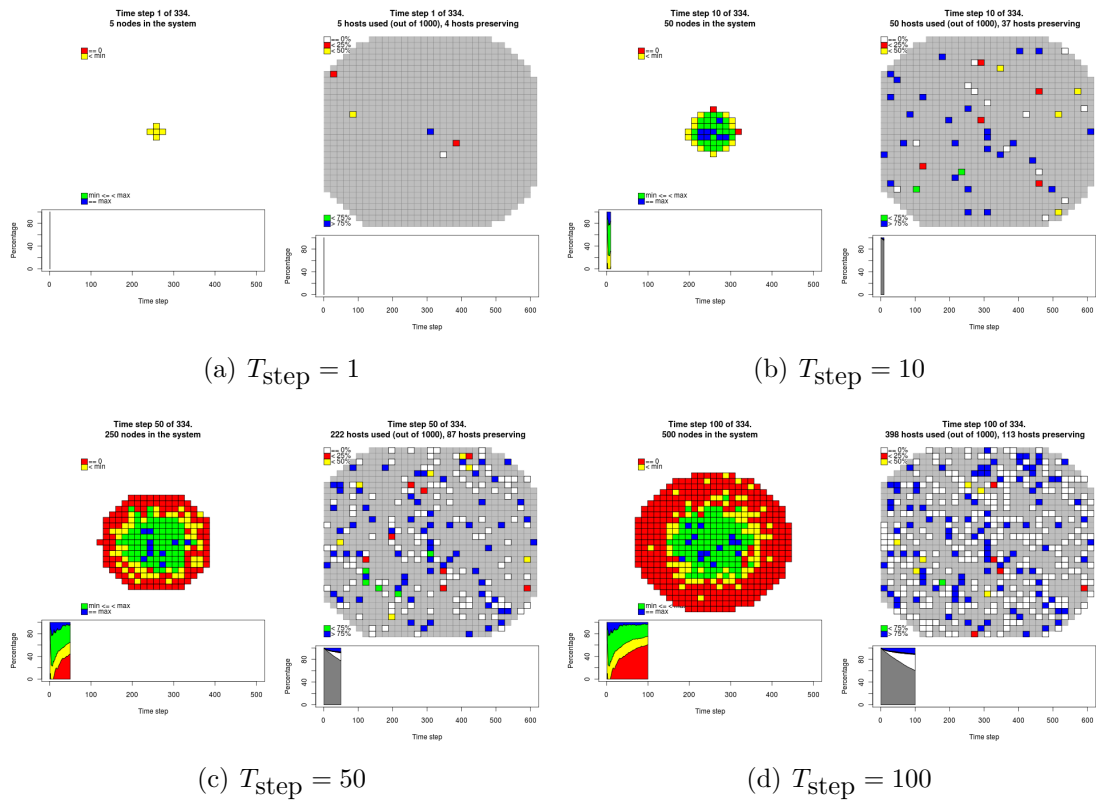


Figure 76. The growth of a $n_{\max} = 500$ WO system captured at various time-steps. The left half of each sub-figure shows the “tree ring” growth of the WO’s portion of the system. The WO and host histograms show the percentage of WO and hosts that are in their respective states as a function of time. All WOs have been created and assigned to a host by $T_{\text{step}} = 100$.

The initial WO is plotted in the center of the left-hand upper quadrant of each composite, Figure 76(a) shows the first 5 WOs in the system. The one in the center is the oldest WO, while the others are younger. The five WOs currently in the system (Figure 76(a)), live on hosts in the system. Hosts can live anywhere on the network and where a particular host is drawn information. The hosts in Figure 76(a) have a finite capacity that their respective system administrators have allocated to the preservation of copies of “foreign” WOs. Foreign copies are copies of WOs that originated on another host and are being preserved on the local host.

At any point in time during the simulation, there will likely be a difference in the number of preservation copies that the WOs want to create and the preservation capacity of all the hosts. Reynolds’ rules (Table 3 on page 17) attempt to balance these two requirements over time. Figure 76(a) on the previous page indicates that the WOs have each made some number of copies (they are colored yellow vice red) and those copies are spread across some of the hosts in a non-even manner. One host has used all its capacity (as shown in blue), while one has not used any (as shown in white). The remaining hosts have used something in between those two extremes (they are yellow and red). In Figure 76(a) on the preceding page, the histograms do not show too much information because the figure shows the $T_{\text{slice}} = 1$ of system growth.

In Figure 76(b) on the previous page, $T_{\text{slice}} = 10$. The tree ring growth of the WOs is becoming more apparent. Older WOs have had more opportunities to make preservation copies of themselves, therefore there is more green and blue in the center of the WO plot. Many of the hosts are have reached h_{cap} , as indicated by the number of blue hosts. The histograms are starting to become filled with data. The WO histogram is starting to show that the percentage of the WOs that have made some, but not all their preservation copies (those in yellow) is starting to grow, while the percentage of those that have reached their goals is lessening. The hosts histogram is starting to show that the percentage of the hosts that have been discovered and added to the system (the grey area), is starting to decrease. A WO will be local to exactly one host. A host may have more than one WO local to it. A WO will not put a preservation copy on any host that it lives on, or that already has a preservation copy of itself.

In Figure 76(c) on the preceding page, $T_{\text{slice}} = 50$. The tree ring presentation of the WO success at preservation is becoming more pronounced. Younger WOs are

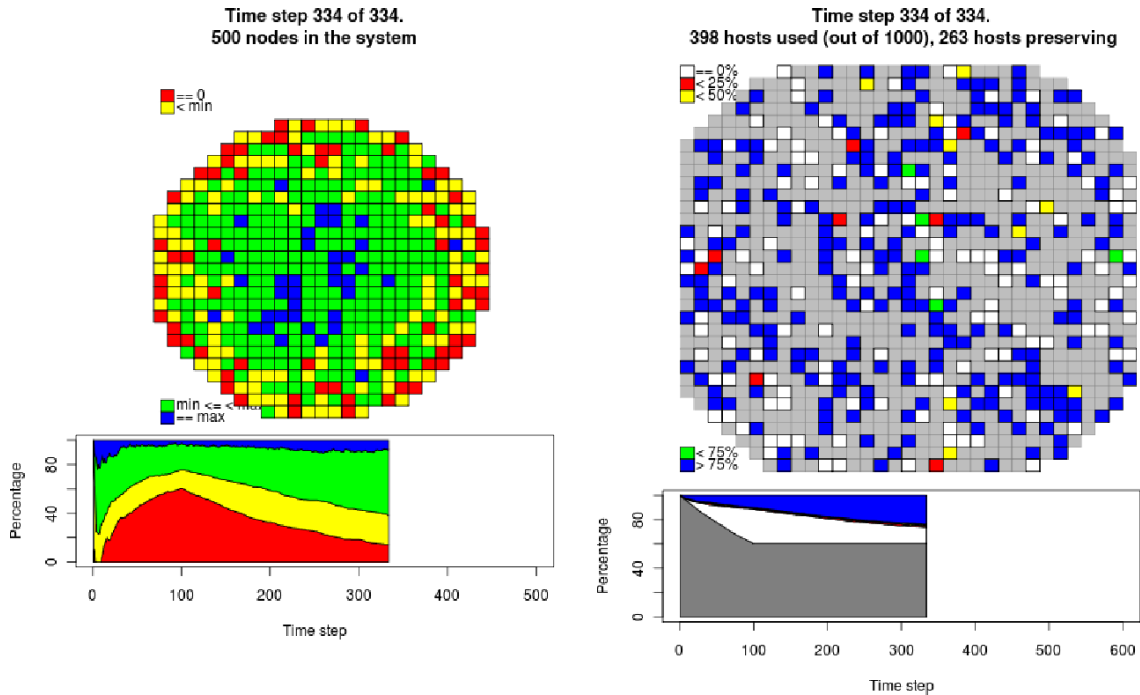


Figure 77. Least aggressive replication policy. System stabilization at $T_{\text{step}} = 334$. Using a moderately aggressive policy results in a higher percentage of WOs meeting their preservation goals sooner and makes more efficient use of limited host resources sooner.

struggling to make copies, while the old ones are maintaining their copies. More of the hosts are being brought into the system (the percentage of grey hosts is decreasing), but a significant percentage of the hosts are not being used for preservation (those shown in white).

In Figure 76(d) on the previous page, $T_{\text{slice}} = 100$. All WOs have been introduced into the system. The tree ring preservation effect is still evident, and some of the new WOs have been fortunate enough to make some number of preservation copies (as shown by the yellow markers in the sea of red). The percentage of hosts that are still not preserving any WOs is still significant, and the percentage of hosts that have reached h_{cap} is holding constant. The system will continue to evolve until it reaches a steady state, when those WOs that have preserved as many copies of themselves as they can based on their knowledge of hosts that have excess preservation capacity. The final time slice for this particular graph is shown in Figure 77.

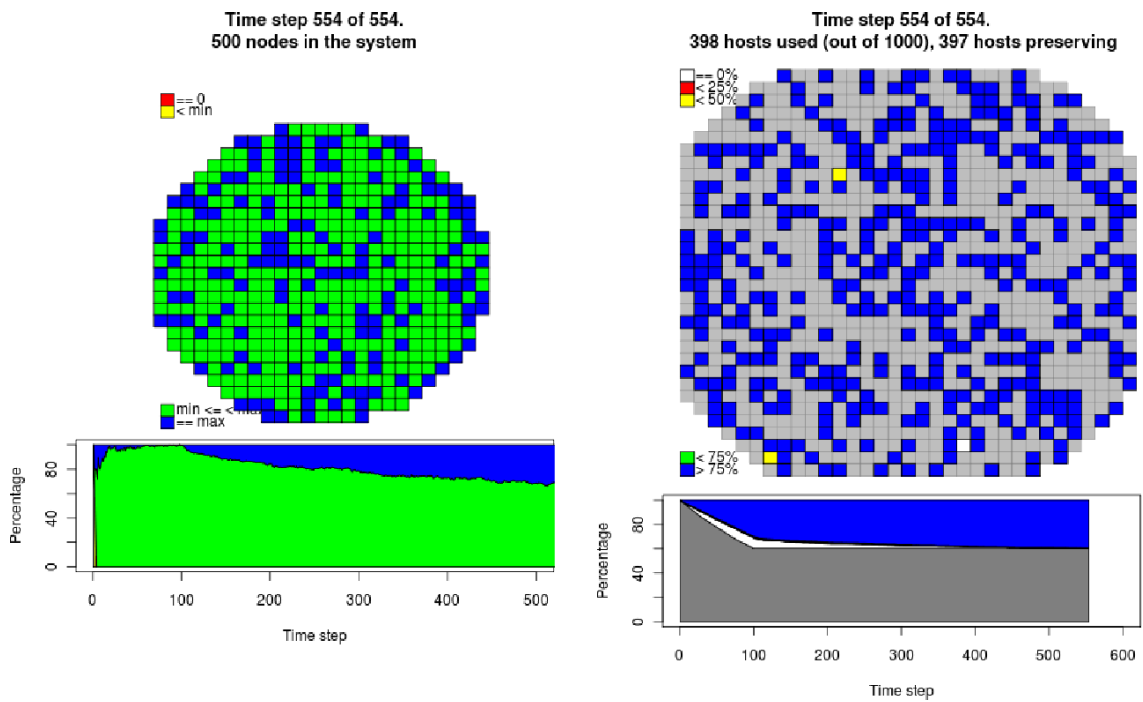


Figure 78. Moderately aggressive replication policy. System stabilization $T_{\text{step}} = 554$. Using a moderately aggressive policy results in a higher percentage of WOs meeting their preservation goals sooner and makes more efficient use of limited host resources sooner.

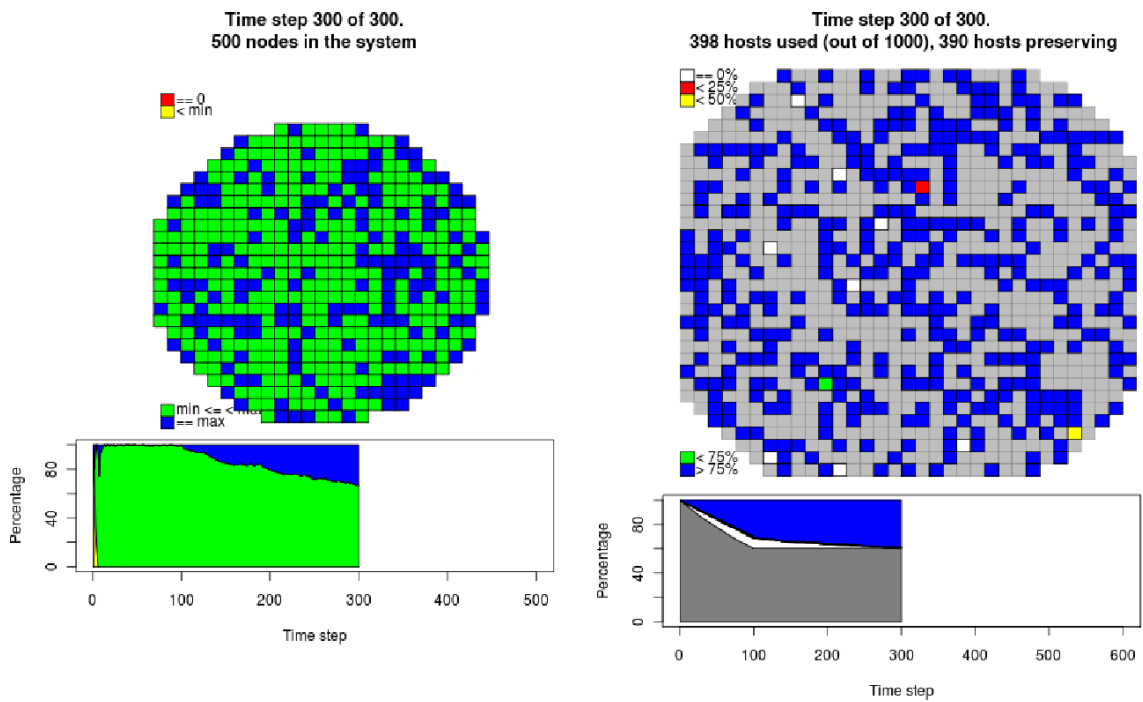
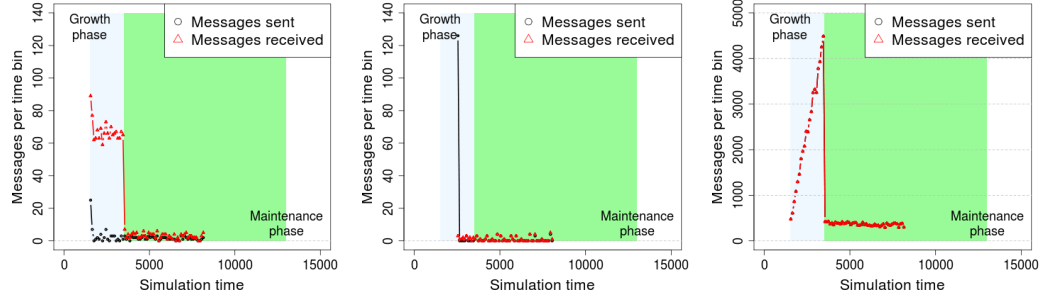
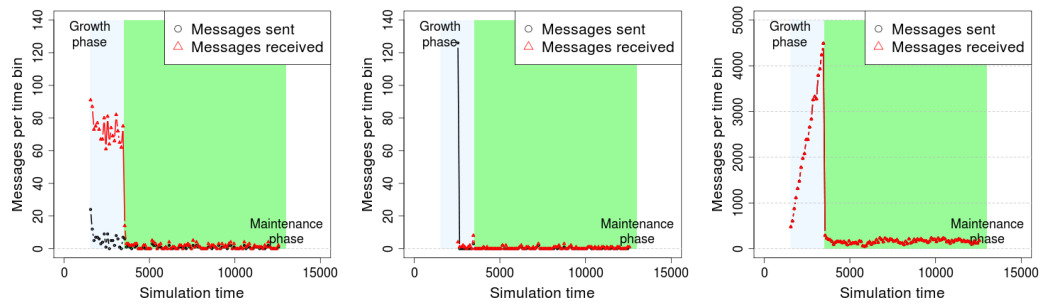


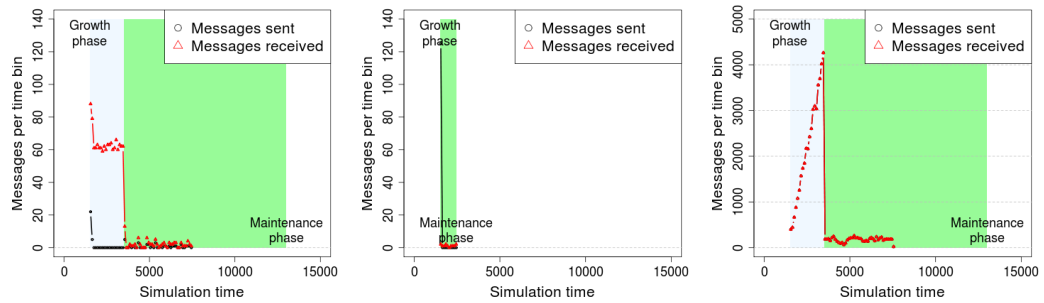
Figure 79. Most aggressive replication policy. System stabilization at $T_{\text{step}} = 300$. Using a moderately aggressive policy results in a higher percentage of WOs meeting their preservation goals sooner and makes more efficient use of limited host resources sooner.



(a) $WO_{1,c,h}$, replication policy 1. (b) $WO_{250,c,h}$, replication policy 1. (c) Sum of all WOs, replication policy 1.



(d) $WO_{1,c,h}$, replication policy 2. (e) $WO_{250,c,h}$, replication policy 2. (f) Sum of all WOs, replication policy 2.



(g) $WO_{1,c,h}$, replication policy 3. (h) $WO_{250,c,h}$, replication policy 3. (i) Sum of all WOs, replication policy 3.

Figure 80. Showing total messages sent and received by an early node, a mid-simulation node and all WOs. The shape of the message sent curves (in black) for the early node is different based on the replication policy (see Figures 80(a), 80(d) and 80(g)). While the shape of messages received curve (in red) remains almost the same. This behavior is contrasted with the mid-simulation node (see Figures 80(b), 80(e) and 80(h)). The mid-simulation node message sent curve is constant regardless of the replication policy. The growth and maintenance phases are shown in light blue and light green respectively.

6.8.5 PRESERVATION STATUS WHEN THE SYSTEM REACHES STEADY STATE

Figure 78 on the next page shows the steady state condition of the same system using the three different replication policies. All WOs have been introduced into the system by $T_{\text{step}} = 100$ (as shown by the “kink” in the percentage of hosts that are used histogram). Each replication policy resulted in a significantly different time to reach a steady state. A steady state in the system is achieved when the WOs have made as many preservation copies as they are able to based on the number of friends that they have acquired when the system was in a growth phase and the number of unique hosts that those friends live on. The WOs are programmed with $c_{\text{soft}} = 3$, $c_{\text{hard}} = 5$ and $h_{\text{cap}} = 5$. The hosts have enough preservation capacity to accommodate the preservation needs of the WOs. If the WO can locate enough unique hosts via its friends, then it will be able to meet its preservation goals. These representative values for number of WOs, desired preservation levels and host preservation capacity were chosen to illustrate the interaction between the WOs as they move preservation copies from one host to another while attempting to maximize the preservation needs of most of the WOs.

The least aggressive policy reaches steady state after $T_{\text{step}} = 334$ (Figure 77 on page 218) and a significant percentage of the WOs have not been able to make any preservation copies (as shown by the lower-most (red) band in the histogram). If the system were to be forced to operate longer; based on the downward trend of the two lower-most bands, it might be possible for the system to achieve a higher percentage of preservation. As shown in the node half of the figure, many of the hosts are not preserving any WOs and those hosts that are preserving have reached their capacity.

The moderately aggressive policy reaches steady state after $T_{\text{step}} = 554$ (Figure 78 on page 219). Prior to $T_{\text{step}} = 100$, most of the WOs have made most of their preservation copies. After $T_{\text{step}} = 100$, the percentage achieving c_{hard} slowly increases until steady state at $T_{\text{step}} = 554$. The hosts’ preservation capacity is used by the WOs in the system almost as quickly as the hosts come on line. This is indicated by the very narrow white region between the unused host region and the totally used region. At steady state, only a very few of the hosts have not been totally used (as shown by the few host usage squares that are neither blue or grey).

The most aggressive policy reaches steady state after $T_{\text{step}} = 300$ (Figure 79 on page 220). Close examination of the host histograms in Figures 78 on page 219 and 79 on page 220 show almost identical behavior both prior to $T_{\text{step}} = 100$ and at steady state. Comparing the host usage plot in the two figures show that slightly more hosts have unused capacity based on a most aggressive policy than a moderately aggressive policy (390 versus 397). Based on n_{max} WOs in the system, the difference between the two policies host under utilization does not appear to be significant.

6.8.6 COMMUNICATION PHASES WHILE THE SYSTEM STRIVES TO REACH STEADY STATE

From the WO's perspective, there are two distinct phases of communication. The first is when the WO is *wandering* through the graph and collecting information from WOs that are already connected into the graph, called the *growth* phase. The second is after the WO is connected into the graph and is based on the current replication policy, and is called the *maintenance* phase. During the growth phase, the WO is aggressively communicating with other WOs. While in the maintenance phase, the WO is responding to queries and communications from other WOs. This change in communication patterns occurs at $T_{\text{step}} = 100$ in Figure 78 on page 219. $T_{\text{step}} = 100$ in Figure 78 on page 219 corresponds to approximately $S_t = 3500$ in Figure 80 on page 221. Figure 80 on page 221 shows the communications for 2 different WOs and the system in total as a function of the replication policy. $WO_{1,c,h}$ and $WO_{250,c,h}$ were chosen to represent the messaging profiles of all WOs in order to see if the profile changes as a function of when a WO enters the system. Time in Figure 80 on page 221 runs until $S_t = 15000$ and messages are counted in bins sized to 100 simulation events.

Looking at Figures 80(a) on page 221, 80(b) on page 221, 80(d) on page 221, 80(e) on page 221, 80(g) on page 221 and 80(h) on page 221, there is a marked difference in the communication curves between $WO_{1,c,h}$ and $WO_{250,c,h}$. These curves (with only minor differences) are consistent across all replication policies. $WO_{1,c,h}$ (the earliest WO introduced into the system), sends a rather modest number of messages $O(2n)$ to WOs that are also in the system as $WO_{1,c,h}$ attempts to create preservation copies. Under the least aggressive policy (Figure 80(a) on page 221), $WO_{1,c,h}$ sends a few messages per time bin until the system enters the maintenance phase. The number

of messages sent during the moderately aggressive policy is nominally the same (Figure 80(d) on page 221). While the most aggressive policy results in messages for just a couple of time bins and then virtually no messages are sent (Figure 80(g) on page 221). Regardless of the replication policy, the number of messages that $WO_{1,c,h}$ receives is about the same.

Comparing the message curves for $WO_{1,c,h}$ and $WO_{250,c,h}$ indicates that the system discovered by the later WO is very different than the one discovered by the earliest WO. The late arriving node has more than enough opportunities to satisfy its preservation goals within the $T_{\text{slice}} = 0$. $WO_{250,c,h}$ sends all of its messages in one time bin and virtually nothing thereafter. This behavior is constant across all replication policies and indicates that the late arriving WOs are able to connect with another WO in very short order (within one time bin) and almost immediately enter into the maintenance phase of their existence. The maintenance phase of the system corresponds to a combination of the *velocity matching* and *flocking centering*.

The system is in a growth phase from about $T_{\text{slice}} = 1500$ to $T_{\text{slice}} = 3500$ as shown by the rising curves in the “Sum of all WOs” sub-figures 80(c) on page 221, 80(f) on page 221 and 80(i) on page 221. During the growth phase, the *wandering* node is sending and receiving a lot of messages while attempting to make its initial connection into the graph. After $T_{\text{slice}} = 3500$, the system is in a maintenance phase when the system is attempting to balance the preservation needs of the WOs with the capacity of the hosts. Comparing the messages curves for the entire system Figures 80(c) on page 221, 80(f) on page 221 and 80(i) on page 221 shows that there is no qualitative difference between the number of messages sent and received in the system based on replication policy. The nuances of the message curves for early WOs is lost as the size of the system increases.

6.8.7 MESSAGES SENT AND RECEIVED AS THE SYSTEM GROWS IN SIZE

Figures 78 on page 219 and 80 on page 221 showed the efficacy and communication costs associated with a system with $n_{\text{max}} = 500$ and $h_{\text{max}} = 1000$. These values allowed the simulation to execute quickly, therefore enabling more options and combinations to be investigated. After determining that at least a moderately aggressive replication policy enabled a high percentage of WOs to meet at least their c_{soft} goals, the next area of investigation was to determine how the total number of

messages changes as a function of system size. Figure 80 on page 221 clearly shows that there are two different types of communication curves reflecting the different types of communication during the growth and maintenance phases. During the maintenance phase, the WOs are attempting to spread their replication copies out across all the unique hosts in their friend's network. One of the contributing factors to this spreading is the limited capacity of the hosts to support preservation. In order to remove the effects of maintenance communications and focus purely on the effect of the number of WOs in the system, a series of simulations were run where $h_{\text{cap}} = 2 * n_{\text{max}}$. This ensured that there would be room on the host for any WO that discovered the host via one of their friends. Based on the simulations, the total number of messages exchanged during the growth phase approximates $O(n^2)$ and the incremental messaging cost of each new WO to the system is $O(2n)$.

6.8.8 COPY CREATION SUMMARY

We have shown that implementing Reynolds' "boid" model that a limited number of rules can result in an emergent behavior system where web objects (WOs) behave in a manner that works towards the betterment of the whole by occasionally sacrificing an individual. Using simulations, we investigated different policies that WOs could use when making preservation copies of themselves. The policies were: (1) be least aggressive and only attempt to make a single copy at a time, (2) be moderately aggressive and initially make at least a minimum number of copies and then revert to policy (1), or (3) be most aggressive and make as many copies as possible at every opportunity and then revert to policy (1).

There are two distinct communication message curves: one prior to all the WOs being introduced into the system and one after. The system's growth period is characterized by many messages being sent from the *wandering* WO and few being received while the WO attempts to make its appropriate number of preservation copies. The maintenance period is characterized by a relatively few number of messages as the WO is directed to sacrifice its preservation copies for the greater good of the graph, and subsequently having to create copies anew. There are distinct differences between the growth message curves of new and late arriving WOs, based on the replication policy. The number of messages exchanged between WOs is virtually independent of the replication policy used. The difference between the maximum and minimum number of messages was only 18% when the USW graph

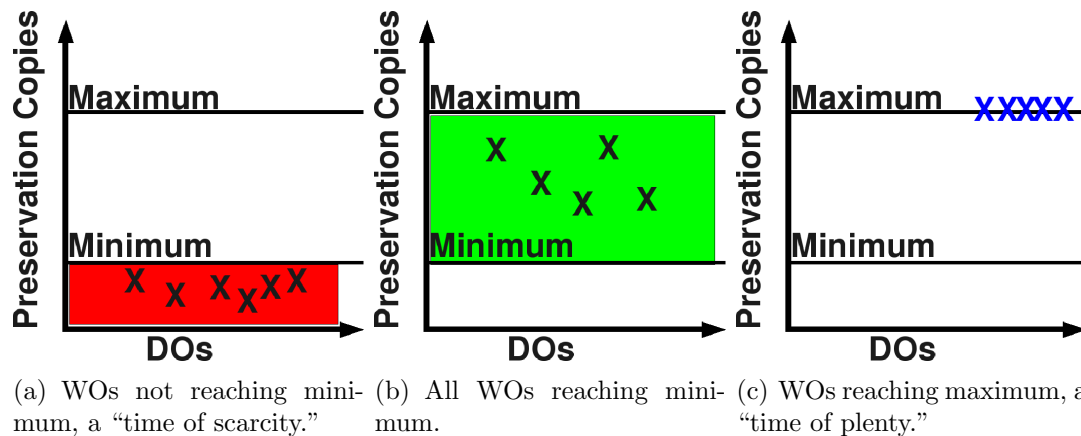


Figure 81. Theoretical truth table showing all possible copying conditions. Showing the combinations of number of preserved copies versus the minimum and maximum desired copies. The two horizontal lines represent the minimum and maximum number of desired preservation copies.

had 100 WOs. As the size of the graph increased to 5,000 WOs, the difference varied from 1% to 8%.

Based on simulations of 500 WOs and potentially 500 hosts with limited preservation capacity; a moderately aggressive replication policy enabled the WOs to attain the same preservation percentage in the same time frame as the most aggressive policy and to more slowly exhaust the preservation capacity of the supporting hosts. A moderately aggressive replication policy will make as many copies as quickly as it can to reach a minimum number of preservation copies and then it will change its behavior to making only one copy at a time until it has reached the its desired maximum number of copies.

We provide a set of USW system design considerations based on WO preservation needs and the preservation capacity of the hosts in Appendix K on page 519. Figure 81 is a truth table showing the relationship between WO preservation desires and host capacity. Figures 82 on the following page and 86 on page 232 show the results of a specific simulation, resulting in only 57% of the WOs achieving their preservation needs while 35% of the host are at their maximum preservation capacity.

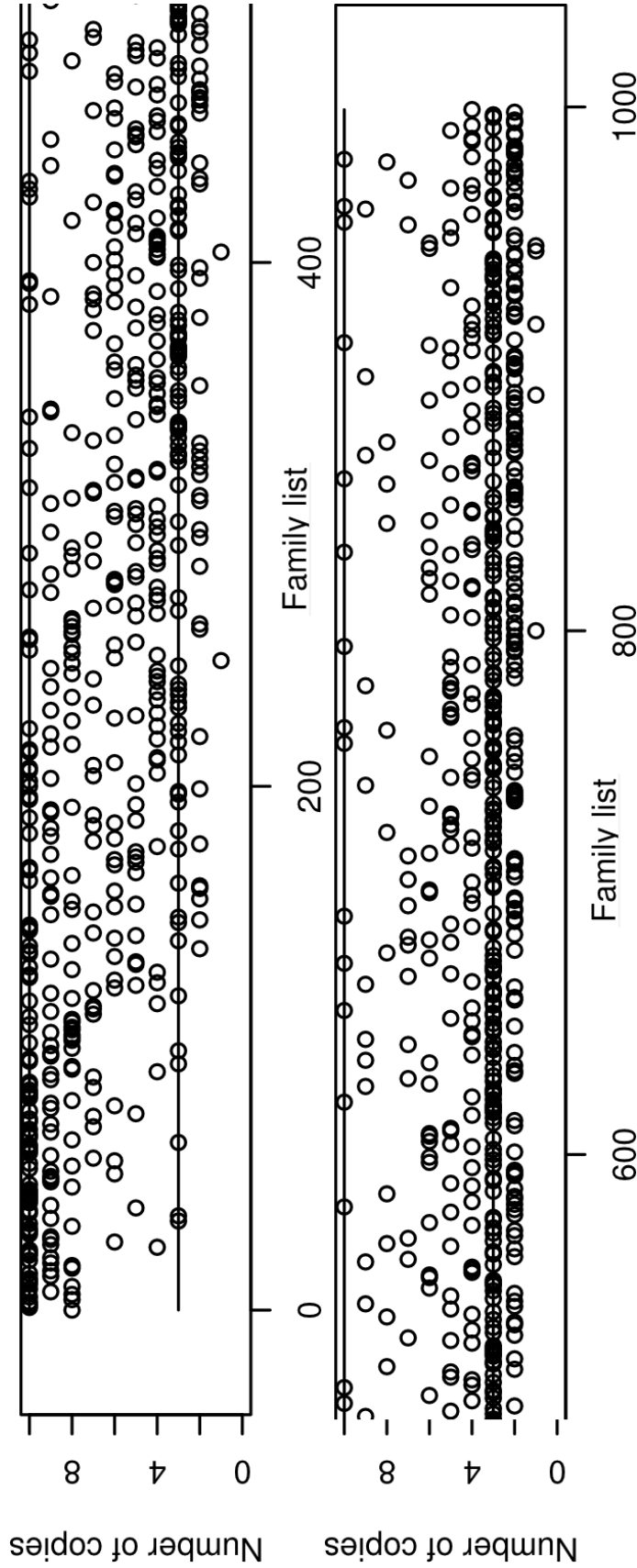


Figure 82. Number of preservation copies based on a specific maximum copying condition. A case where 35% of the hosts are at their maximum preservation capacity and 58% of the WOs have preserved their minimum number of copies. WOs are assigned to a random host.

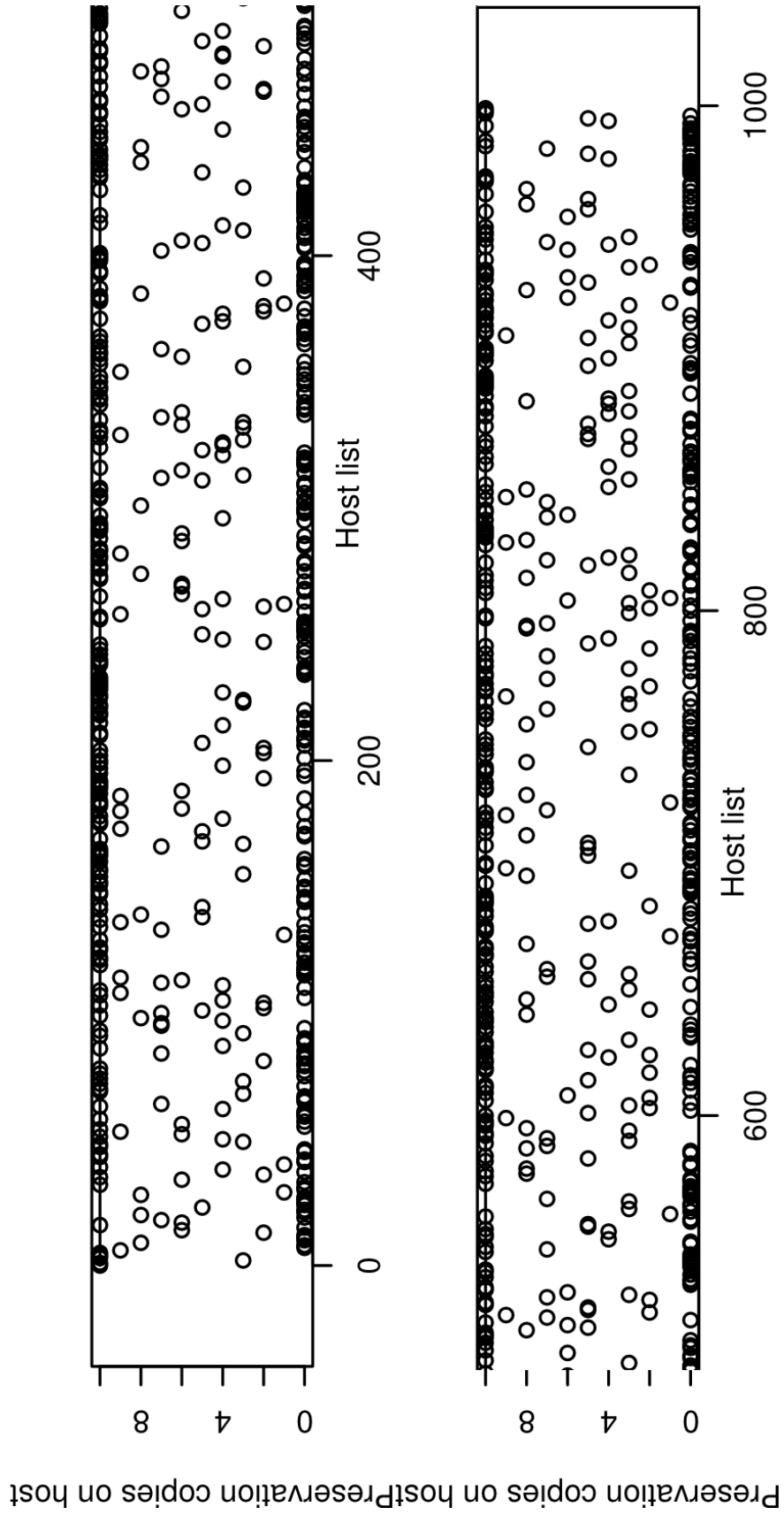


Figure 83. Number of preservation copies per host based on a specific maximum copying condition. A case where 35% of the hosts are at their maximum preservation capacity and 58% of the WOs have preserved their minimum number of copies. WOs are assigned to a random host.

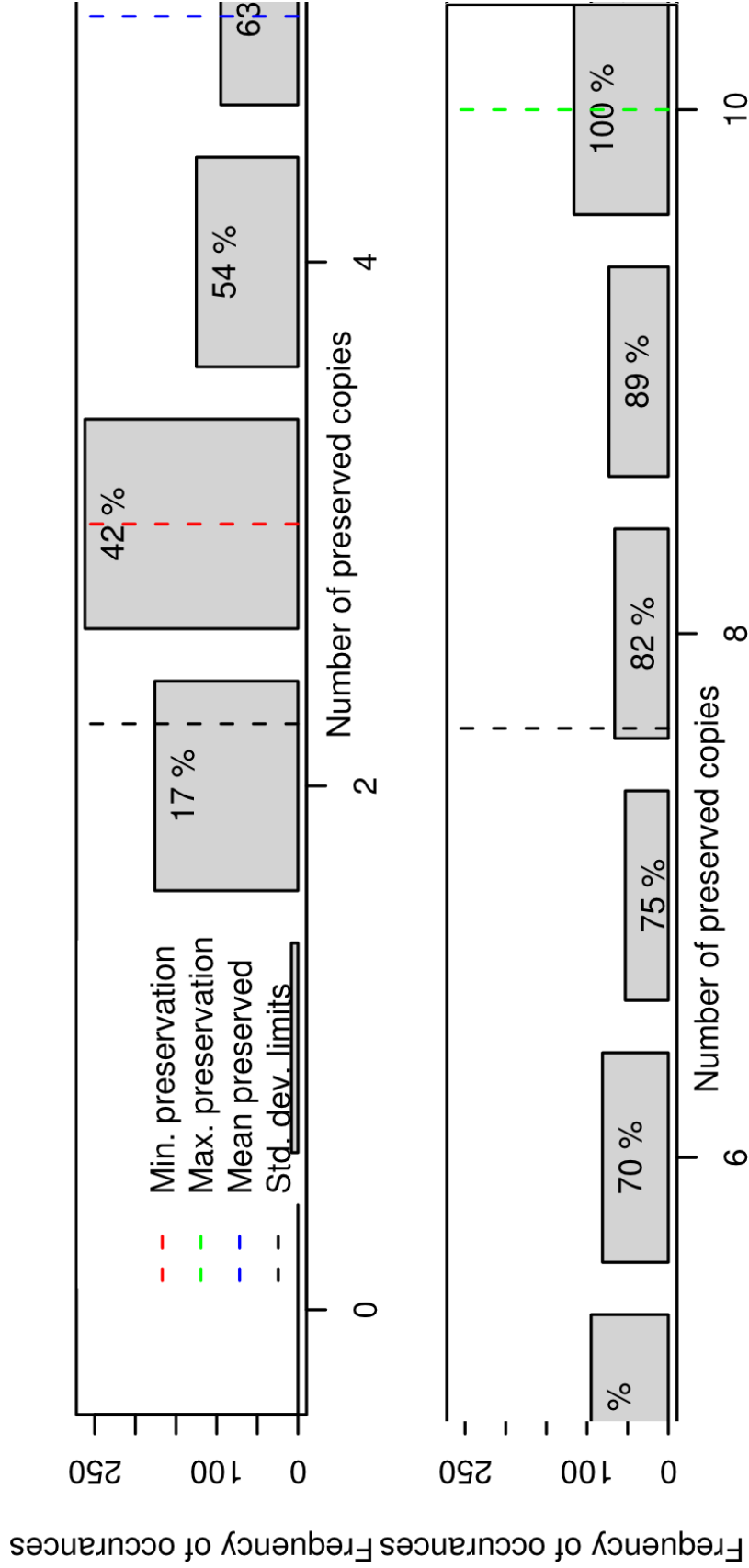


Figure 84. Histogram of preservation copies based on a specific maximum copying condition. A case where 35% of the hosts are at their maximum preservation capacity and 58% of the WOs have preserved their minimum number of copies. WOs are assigned to a random host.

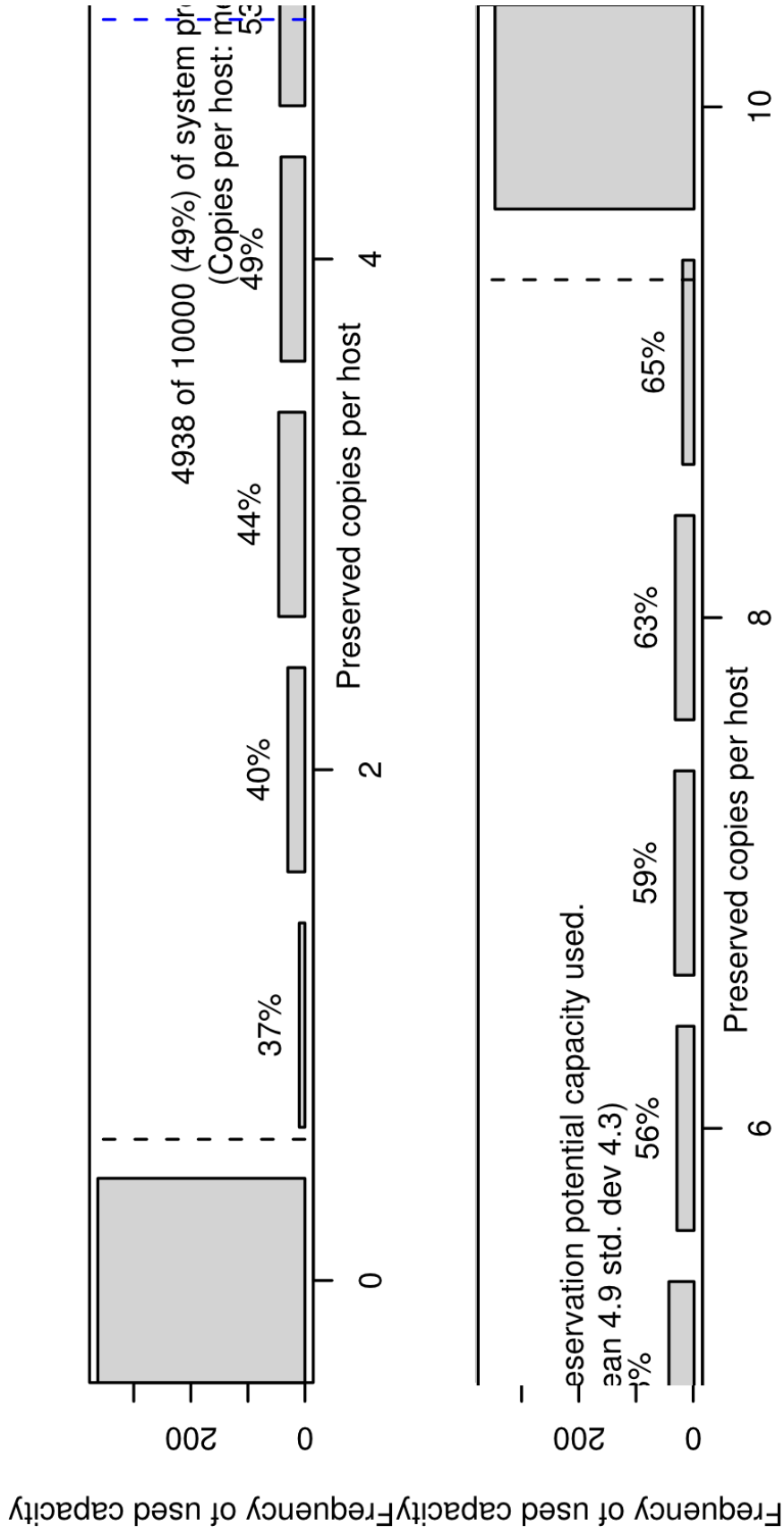


Figure 85. Histogram of preservation copies per host based on a specific maximum copying condition. A case where 35% of the hosts are at their maximum preservation capacity and 58% of the WOs have preserved their minimum number of copies. WOs are assigned to a random host.

Table 38. USW simulation values used to reduce USW problem space.

Parameter	Min. value	Max. value	Inc. value
Policy A	1	5	1
Policy B	1	3	1
Policy C	1	4	1
Sample function	1	10	10
β	0.8	0.8	N/A
γ	0.5	0.5	N/A
Size	100	100	N/A

6.8.9 COMPUTATION AND ANALYSIS

Theoretically, the size of the USW problem space is unlimited because of the infinite values through which β and γ could be taken through. In order to reduce the computational problem space, they and the number of WOs, were held to a fixed value, while the policies were exercised for all values (Table 38). The fixed values for β and γ were based on experience with previous USW simulations as generating graphs that tended to meet the small-world criteria for $C(G)_{Average}$ and $L(G)$.

For all permutations, the $C(G)_{Average}$ and $L(G)$ were computed and compared to a random graph of the same size and number of edges. The number of times where the USW graph $C(G)_{Average}$ exceeded the random graph and where the USW graph $L(G)$ approximated the random graph was enumerated (Table 39 on page 235 and Table 40 on page 236).

6.8.10 SUMMARY

In summary; we have made the following design decisions to reduce the problem space:

1. *Choosing the initial WO:* A well-known WO will be used for ease of implementation and testing. We will call this **Policy A**.
2. *Choosing the next WO:* FIFO processing will be used for ease of implementation and testing. We will call this **Policy B**.
3. *Deciding how many connections to make:* Using a log function concentrates

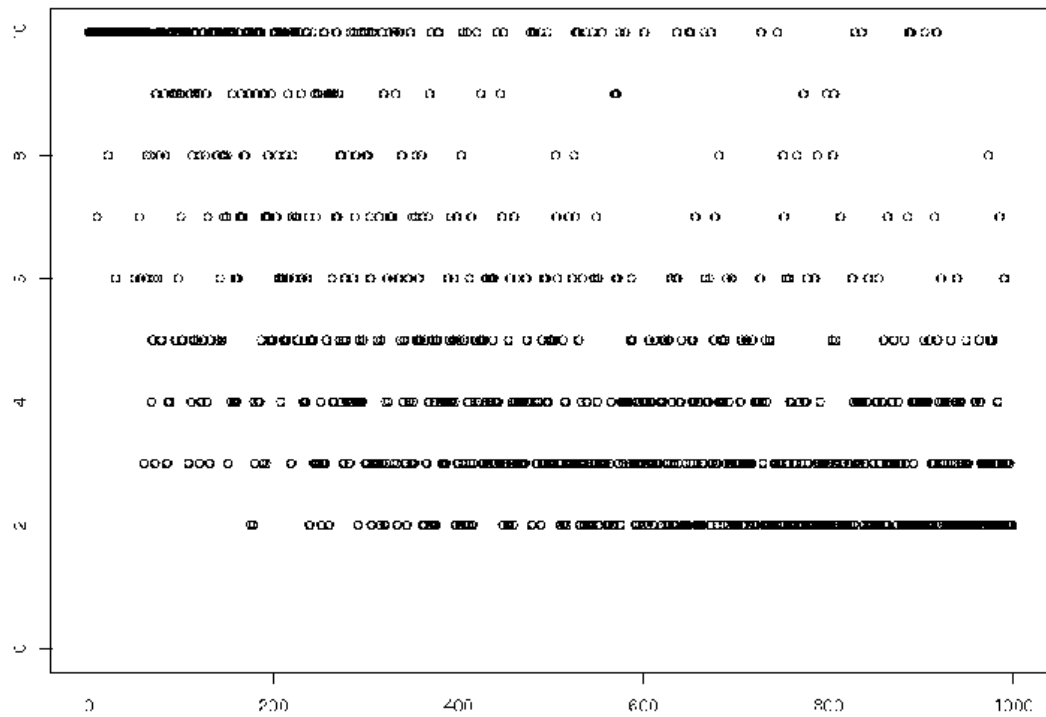


Figure 86. A specific minimum copying condition WO status. A case where 65% of 1,000 WOs have preserved at least their minimum number of copies. WOs 0 through 1000 have between 2 and 10 preservation copies each.

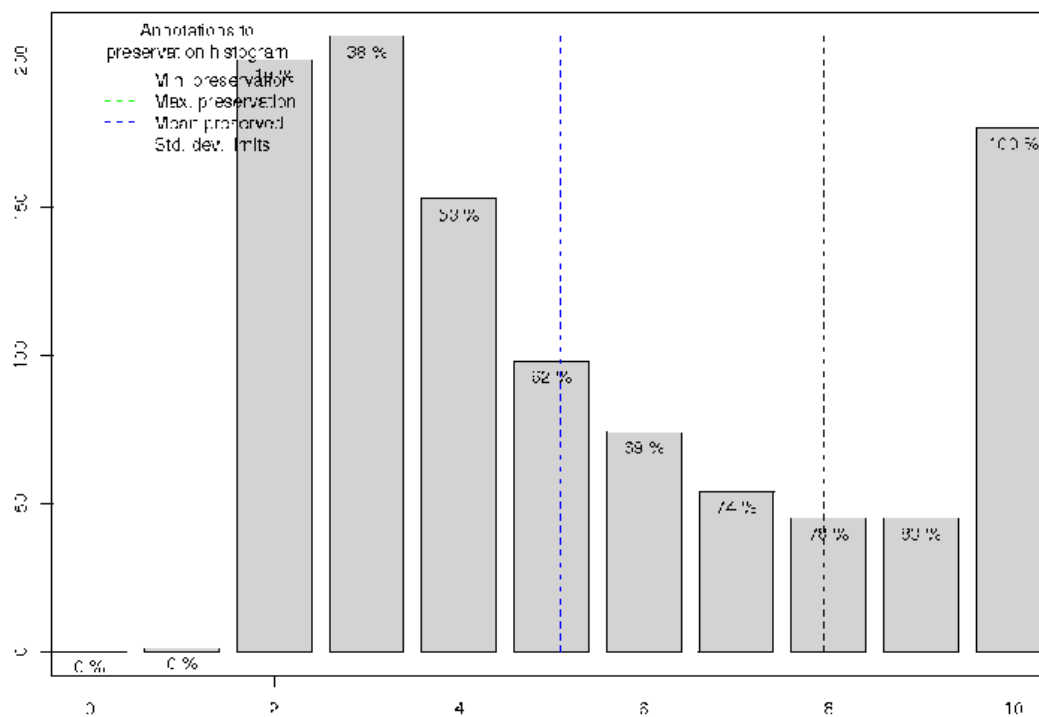


Figure 87. A specific minimum copying condition, a histogram. A case where 65% of 1,000 WO have preserved at least their minimum number of copies. Histogram of how many WO achieved a particular level of preservation copies. Preservation copies mean and standard deviation values are shown in blue and black respectively.

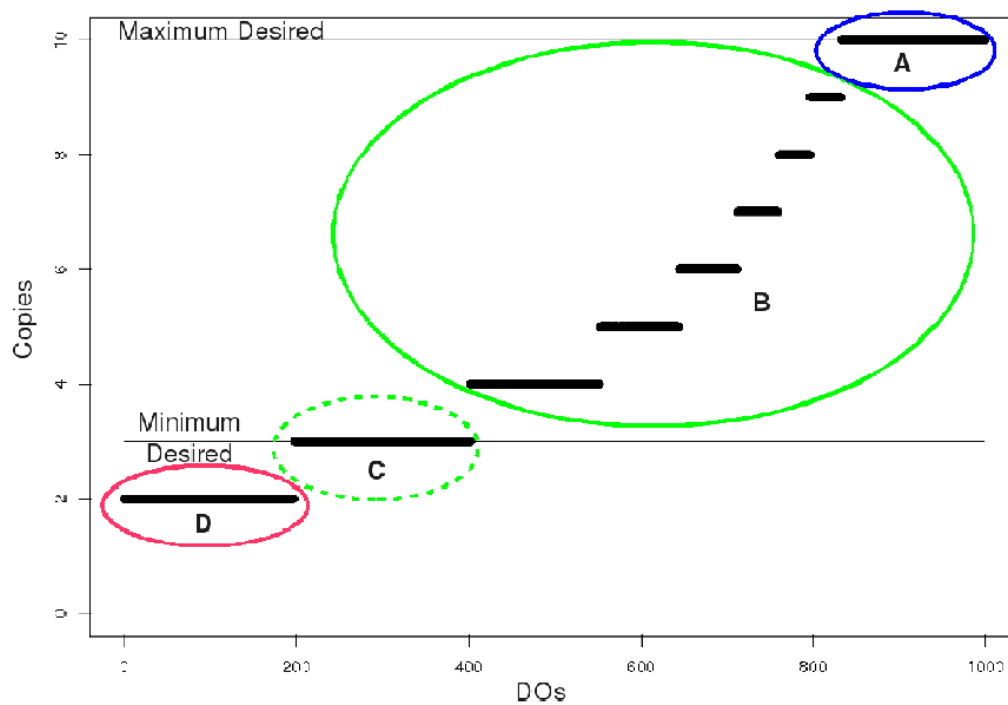


Figure 88. A specific minimum copying condition, sorted by WO status. A case where 65% of 1,000 WOs have preserved at least their minimum number of copies. WOs sorted by preservation copies highlighting those WOs that are in a “time of scarcity,” (red) or “time of plenty,” (blue) or in between (green). See Section 6.8.8 on page 225 for full explanation of the different regions.

Table 39. CC and APL logical (raw) data. The number of times the USW CC was greater than a random graph with the same number of vertices and edges, and the USW APL was approximately the same as a random graph APL. The numeric values for A correspond to choosing: 1) random WO, 2) most connected WO, 3) highest degreed WO, 4) lowest degreed WO. The numeric values for B correspond to next WO selection: 1) FIFO, 2) LIFO, 3) random selection. The numeric values for C correspond to how to select WO friends: 1) random selection, 2) FIFO selection, 3) LIFO selection, 4) prefer friends of friend.

Policy		Policy A				
C	B	1	2	3	4	5
	1	8	8	8	8	8
1	2	8	8	8	8	8
	3	8	8	8	8	8
	1	2	2	2	2	2
2	2	0	0	0	0	0
	3	0	0	0	0	0
	1	2	2	2	2	2
3	2	0	0	0	0	0
	3	2	0	2	2	2
	1	3	3	3	3	3
4	2	3	3	3	3	3
	3	3	3	3	3	3

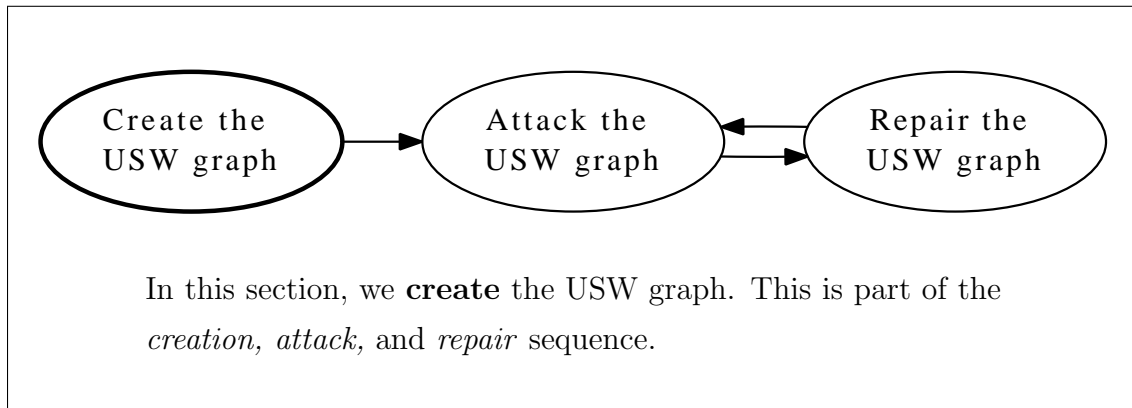
Table 40. CC and APL logical (normalized) data. The numeric values for A correspond to choosing: 1) 2) random WO, 3) most connected WO, 4) highest degreed WO, 5) lowest degreed WO. The numeric values for B correspond to next WO selection: 1) FIFO, 2) LIFO, 3) random selection. The numeric values for C correspond to how to select WO friends: 1) random selection, 2) FIFO selection, 3) LIFO selection, 4) prefer friends of friend.

Policy		Policy A				
C	B	1	2	3	4	5
1	1	1.0	1.0	1.0	1.0	1.0
	2	1.0	1.0	1.0	1.0	1.0
	3	1.0	1.0	1.0	1.0	1.0
2	1	0.3	0.3	0.3	0.3	0.3
	2	0.0	0.0	0.0	0.0	0.0
	3	0.0	0.0	0.0	0.0	0.0
3	1	0.3	0.3	0.3	0.3	0.3
	2	0.0	0.0	0.0	0.0	0.0
	3	0.3	0.0	0.3	0.3	0.3
4	1	0.4	0.4	0.4	0.4	0.4
	2	0.4	0.4	0.4	0.4	0.4
	3	0.4	0.4	0.4	0.4	0.4

the number of friendship links into a smaller set of WOs but does not cure the systemic problem of algorithmically limiting the number links. Combining an artificial lower bound on the number of WOs to force a higher out degree and using the last introduced WO as the *USW established* WO results in a graph in rough approximation with a classic small-world graph.

4. *Choosing connections*: Preferential attachment will be used to connect to the friends of the established WO that the wandering WO makes it first connection to, then *random selection* will be used for ease of implementation and testing. We will call this **Policy C**.
5. *Deciding when to make a preservation copy*: A *polite* preservation will be used for ease of implementation and testing. We will call this **Policy E**.

6.9 CREATING AN UNSUPERVISED SMALL-WORLD GRAPH



We have reduced the USW parameter space (see Section 6.8 on page 212) to a manageable size. We will use the data from Table 40 on the preceding page as a guide for selecting appropriate values to create a USW graph for further analysis. During this analysis, we will:

1. *Create* the graph,
2. *Attack* the graph by removing 10% of the WOs in the graph using the $A_{V,H}$ profile,
3. *Repair* the graph examining a giving the graph 2 opportunities to detect the loss of WOs and to take action to repair the graph,

Table 41. Alternating turns between attacker and graph.

Turn	Health
0	$G = 1$
1	$G = G * Damage(G)$
2	$G = G * r(G)$
3	$G = G * Damage(G)$
4	$G = G * r(G)$
\vdots	\vdots

4. Repeat the *attack* and *repair* sequence will be repeated until the graph reaches a steady state.

During each step of the process (*create*, *attack*, and *repair*), the graph will be analyzed ($C(G)_{Average}$, $L(G)$, and $\langle k \rangle$, degree as a function of time), and the data reported.

6.10 ANALYSIS OF DAMAGE VS. REPAIR

We investigate the relationship between the damage $Damage(G)$ an attacker can cause to a graph, and how resilient $r(G)$ the graph has to be to recover from the damage. We approach the problem through analysis and simulation.

The following assumptions apply:

1. $Damage(G)$ affects a constant of the USW graph
2. $r(G)$ remains constant as relative to $Damage(G)$
3. The “health” of the graph (based on some measure) is 1.0 before any damage is done.

We view the interplay between the attacker and graph as a game where each player alternates turns. The attacker causes $Damage(G)$ damage to the graph, and the graph repairs itself $r(G)$ before the attacker’s next turn (Table 41).

$Damage(G)$ is under control of the attacker, and $r(G)$ is a function of maintenance activity in the USW graph. The amount of maintenance activity between attacker turns is dependent on the rate of accessing the USW WOs and the length of time between attacker turns. If the time between attacks is small, then there

might be a very small number of maintenance activities. The relationship between $Damage(G)$ and $r(G)$ is not straight forward (Equation 101). As an example of the interaction between $Damage(G) = 0.75\%$ and $r(G)$, a game was conducted for all three conditions of $r(G)$ for a 350 turn game (Figure 89 on page 247).

$$r(G) = \begin{cases} > \frac{1}{Damage(G)} & \text{then graph will improve} & (101a) \\ = \frac{1}{Damage(G)} & \text{then graph will stabilize} & (101b) \\ < \frac{1}{Damage(G)} & \text{then graph will collapse} & (101c) \end{cases}$$

6.10.1 CREATING THE USW GRAPH

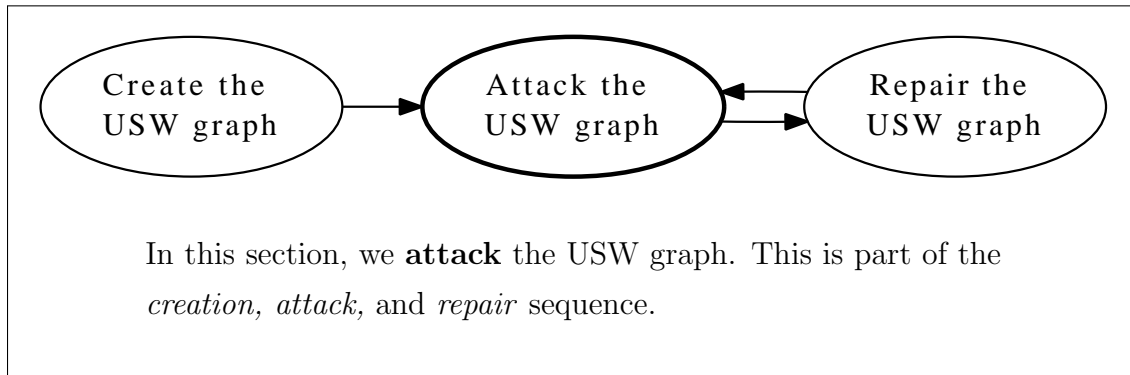
We use a single set of parameters to create the graph (Table 42). The $C(G)$, degree distribution, $L(G)$ for the graph is show in Figures 90 on page 248, 91 on page 249, and 92 on page 250 respectively. The degree distribution changes and evolves over time (Figure 93 on page 251).

Table 42. USW simulation values used to create a USW graph.

Parameter	Value
Policy A	1
Policy B	2
Policy C	1
Sample function	1
β	0.8
γ	0.5
Size	1500

The USW algorithm successfully created a graph that meets that small-world criteria for $C(G)_{Average}$, $L(G)$. The graph will be attacked and analyzed.

6.11 ATTACKING AN UNSUPERVISED SMALL-WORLD GRAPH



6.11.1 INTRODUCTION

We assume that an external entity causes damage to the USW graph. This entity could be as simple as the failure of a host to maintain a WO, or as complex as an adversary bent on removing some WOs for some reason. As we have shown in an earlier section (see Section 5.5.5 on page 159), $A_{E,H}$ is the most effective attack profile to damage and disconnect a graph. Due to the WO's internal structure, a $A_{E,H}$ attack profile requires that very specific changes be made to the WO and we assume that this is neither permitted by the hosts on which the targeted WO lives, nor is it practical for the attacker to implement.

A far simpler attack profile to implement is the $A_{V,H}$.

6.11.2 ATTACKING THE USW GRAPH

We simulate the actions of the attacker by examining the USW graph and removing the 10% of the discovered WOs meeting the $A_{V,H}$ attack profile. (Current technology enables an adversary to explore an Internet graph of significant size [186], but we assume that the adversary does want to draw attention to the attack.) After the WOs were removed, a set of graph metrics were measured (Table 43 on the following page). HTTP/HTML protocols preclude the $A_{E,*}$ attack profiles because removing an edge requires that both ends of the edge be modified so that the edge does not exist. HTTP/HTML protocols do not permit such fine grained strategies, only the total removal of a WO is supported.

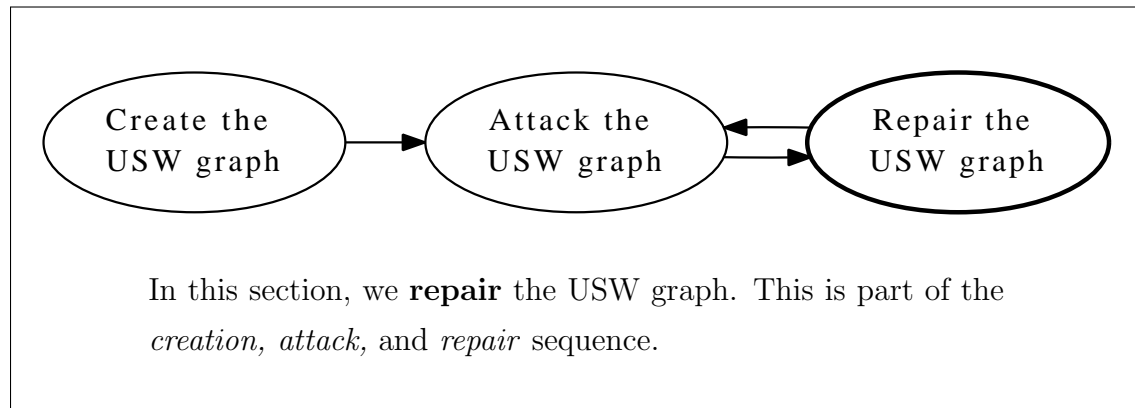
6.11.3 SUMMARY

Table 43. Comparison of pre and post attack USW graph metrics.

Parameter	Pre. attack	Post attack
n	300	224
$ E $	2,392	1,732.0
$\langle k \rangle$	15.95	15.5
$\langle k \rangle \sigma$	12.5	11.1
$C(G)$	0.1	0.2
$L(G)$	2.3	2.2

We have simulated a focused $A_{V,H}$ on attack on a USW graph and measured the results of the attacks. We measured various graph parameters to quantify the effect of the attack.

6.12 REPAIRING AN UNSUPERVISED SMALL-WORLD GRAPH



We limit the problem of repairing a USW graph to detecting the loss of a WO and making one of the missing WOs the active maintainer. This is opposed to adding a new WO replace the missing one. We take this approach to focus on the ability of the USW graph to maintain preservation copies of each WO, rather than on the ability of the USW graph to accept new WOs. Adding new WOs to an existing USW graph was discussed in Section 6.9 on page 237.

6.12.1 DETECTING USW GRAPH LOSSES AND GAINS

Detecting the loss of a WO

When a WO is non-reachable, then it is assumed to be lost. A WO will be either an active maintainer or a passive maintainer. An active maintainer is responsible for ensuring that there are enough preservation copies for the family. A passive maintainer sends notification messages to the active maintainer about things it has detected and that the active maintainer needs to take action upon. Active and passive maintainer WOs compose a “family” relationship.

The active maintainer in a family will communicate with active maintainer in another family. These communications are between “friends.”

The loss of a passive maintainer will be detected by the active maintainer during the active maintainer’s normal action. The loss of an active maintainer will be detected by a passive maintainer from the same family. When the active maintainer loss is detected, one of the family’s passive maintainers will assume active maintainer responsibilities.

Detecting a disconnected USW graph

A USW graph could become disconnected because of something as simple as misconfiguring a critical router or as complex as massive attack.

It is not possible for a WO to detect when the USW graph is disconnected. A disconnection is only possible by viewing a graph from an external viewpoint (the viewer has to be outside the graph to see that the graph is disconnected). A WO does not have an external viewpoint, it will never be able to detect that the graph is disconnected.

A WO will be able to detect:

1. Its active maintainer is not reachable,
2. Some (perhaps all) of its friends are not reachable.

6.12.2 DETECTING A RECONNECTED DISCONNECTED USW GRAPH

A USW graph reconnecting means that at least one USW family that was disconnected is now connected. This can happen after the graph was disconnected, each family has grown somewhat, but not all members of each family have been

accessed and subsequently have not updated their internal data structures with the latest family information. So there is at least one family member in each partition that has a “memory” of the family in the other partition. To the WO which has memory of the other partition, it behaves exactly as if it was never disconnected. Some of copies may not be accessible, and it will notify the family of that. It will add entries to additional entries about the copies that were made between the last time it was accessed and now. It will follow whatever maintenance responsibility is defined in its internal data structure. This may cause a newly accessed WO to assume parental responsibilities (with an announcement to the family). It may then relinquish responsibilities because some other WO is better suited (by definition) that it is. After some number of accesses of all the WOs in the newly reconstituted graph, there will be only a single active maintainer based on whatever maintenance approach is in effect.

If the two graphs have anything other than the “progenitor only” maintenance approach are separated for a long time, it is possible that both parts will have created the maximum number of copies as dictated by their internal data structures. When the two partitions unite, the processing of messages from the family mailboxes could result in the family having more than its maximum number. This is not an issue, because the maintaining WO will only make copies when there are fewer than the minimum number needed and less than the maximum. If there are more than the maximum, then no new copies will be created. Eventually the excess WOs in the family will die off and then the maintainer will set about making new copies.

6.12.3 REPAIRING THE USW GRAPH

We repair the USW graph by attempting to access all the active maintainer WOs. Those WOs that have been removed by the attacker will not be accessible, therefore we select one of the passive maintainers for that family to become the family’s active maintainer. The new active maintainer WOs have internal copies of the same USW creation parameters (see Section 6.9 on page 237) that the original active maintainers had, and assume their maintenance responsibilities for a family that now has one less family member due to the loss of the original active maintainer.

After all necessary active maintainers have been identified, all the WOs in the USW graph are accessed to perform their maintenance functions. Accessing all the WOs simulates either a very active USW graph, or a long time between attacks.

Table 44. Comparison of pre and post repair USW graph metrics.

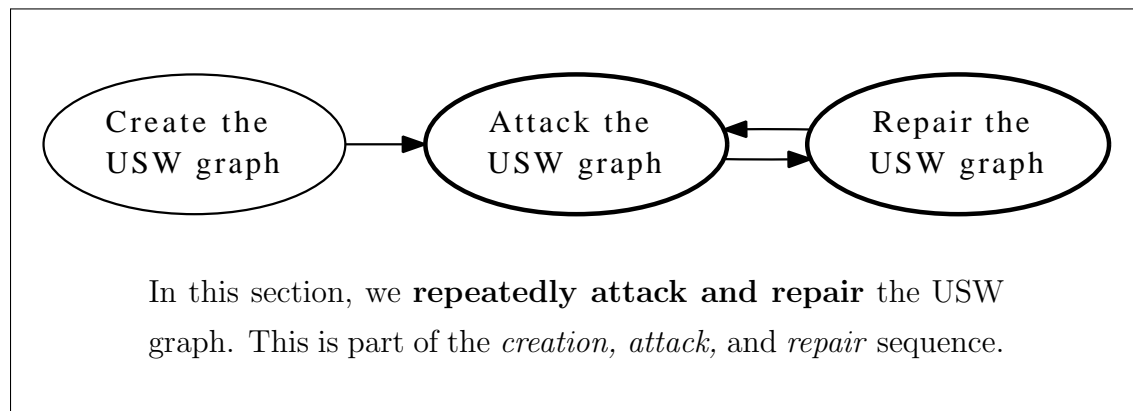
Parameter	Pre. repair	Post repair
n	224	300
$ E $	2,392	1,732.0
$\langle k \rangle$	15.46	9.3
$\langle k \rangle \sigma$	11.05	5.5
$C(G)$	0.16	0.2
$L(G)$	2.23	2.7

Following the universal accessing, various graph metrics are collected (Table 44).

6.12.4 SUMMARY

We have evaluated how a USW graph can be repaired predicated on accessing all available active maintenance WOs. The USW graph is both *robust* and *resilient*.

6.13 REPEATEDLY ATTACKING A USW GRAPH



Over the long life of the USW graph, it is possible that it might be attacked repeatedly. These attack repair cycles could be expected to repeat many, many times. We are interested in the long-term change in selected USW graph metrics.

For clarity, no additional WOs were added to the USW graph between the time it was attacked and when it was repaired. Preventing additional WOs from entering the system ensured that only the effects of the attack and repair processes were

captured.

6.13.1 REPEATEDLY ATTACKING THE USW GRAPH

$L(G)$ and $C(G)$ data for a USW graph that has been attacked using the $A_{V,H}$ profile and repaired at two different percentage levels are shown (Figure 94 on page 252 and Figure 95 on page 252). The symbols are identical for both figures, but only the first one has a legend because of way the data in the second figure covers most of the plotting surface.

In both figures, the reported $L(G)$ and $C(G)$ are shown as well as the computed $L(G)$ and $C(G)$ for a random graph with the same number of vertices and edges.

The reported $L(G)$ follows the see-saw action of increasing after each attack and then decreasing after each repair. This behavior continues for a few attack repair cycles and then tends to settle within a small range of values. The reported values are about 30% higher than the computed values. The reported is higher because it is based on real degree distribution data, while the computed assumes a uniform $\langle k \rangle$ distribution.

The $C(G)$ starts out a value and then decreases after an attack. When the graph is repaired, the $C(G)$ increases above its original value. The increase is due to the maintenance activities of the WOs. If a WO has not achieved the number of preservation copies that it desires, then it will explore the graph again looking for places to put the copies. Exploring the graph enables the WO to add more connections to other WOs and thereby increasing the $C(G)$ of the entire graph. The see-saw behavior of $C(G)$ settles out when the $L(G)$ settles out. In all cases, the computed $C(G)$ is lower than the reported because the reported is based on real degree distribution data, while the computed assumes a uniform $\langle k \rangle$ distribution.

6.13.2 SUMMARY

We have demonstrated that the USW graph is *robust* and *resilient* when alternately attacked and then allowed to repair itself. If a high percentage of the USW graph is damaged by the attack, then the USW algorithm results in more additional edges being added to the graph as compared to a low percentage attack. These additional edges are additional friendship connections made between active maintainers. The more connections there are, the closer the graph comes to being fully connected. As the USW graph approaches being fully connected, the closer $L(G)$

and $C(G)$ come to 1. The emergent behavior of the USW WOs will be controlled through the application of a few policies and one control variable. These policies and the control variable value are set at the time the WO is created and are immutable thereafter. This design is in keeping with Reynolds' minimal rule set for emergent behavior.

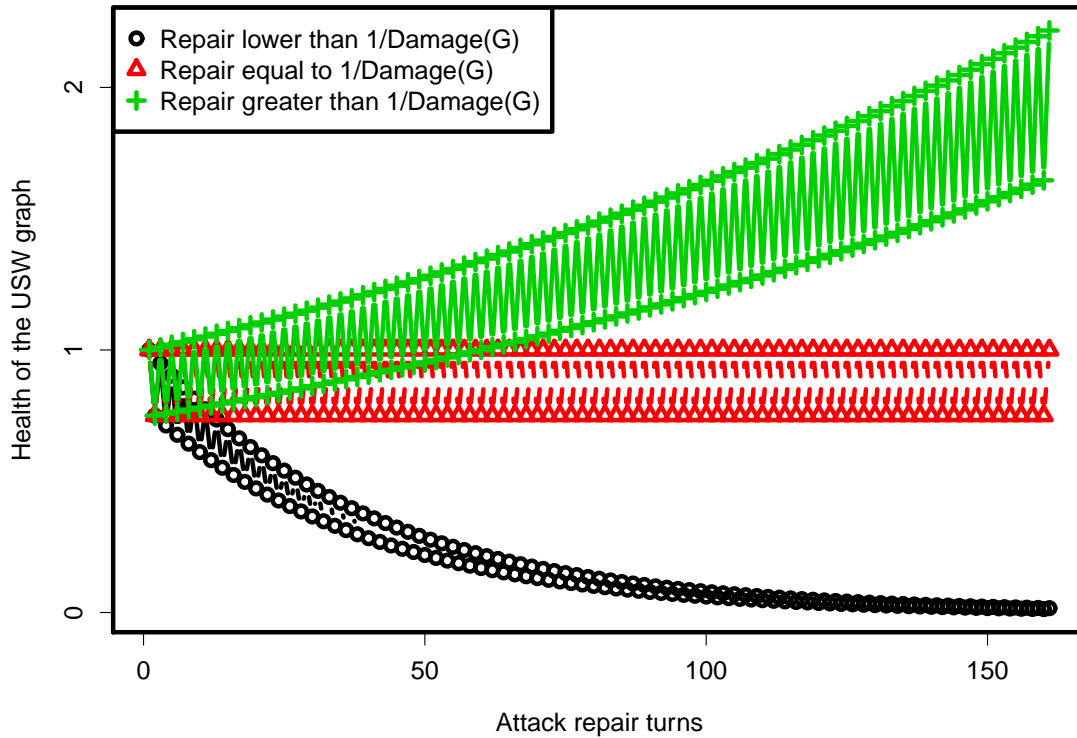


Figure 89. Relationship of damage and repair on graph health. A sample game between an attacker and the USW graph, where $\text{Damage}(G) = 0.75\%$ and three different values of $r(G)$ are used (Equation 101 on page 239). A low $r(G)$ will result in a graph whose health declines and asymptotically approaches 0. A high $r(G)$ will result in a graph that continues to improve. While a $r(G)$ which just matches the inverse of the $\text{Damage}(G)$ will restore the graph to its initial health. In all cases, the figure shows the health of the graph when it was damaged and then the health after active maintenance activities took place. These two actions inscribe an alternating low high area for each of the $r(G)$ values.

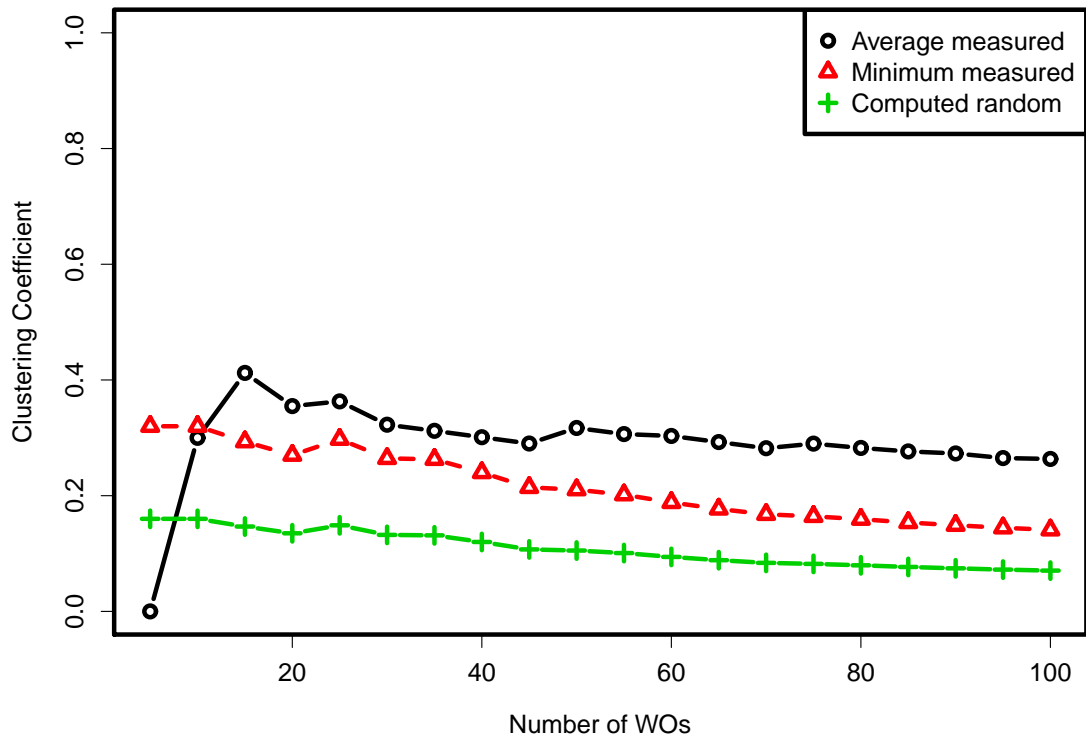


Figure 90. CC of USW simulation. After the graph has started growing, the measured USW $C(G)_{Average}$ remains significantly higher than the computed random graph with the same number of vertices and edges. The minimum measured $C(G)_{Average}$ provides a feel for the $C(G)_{Average}$ range of the total graph.

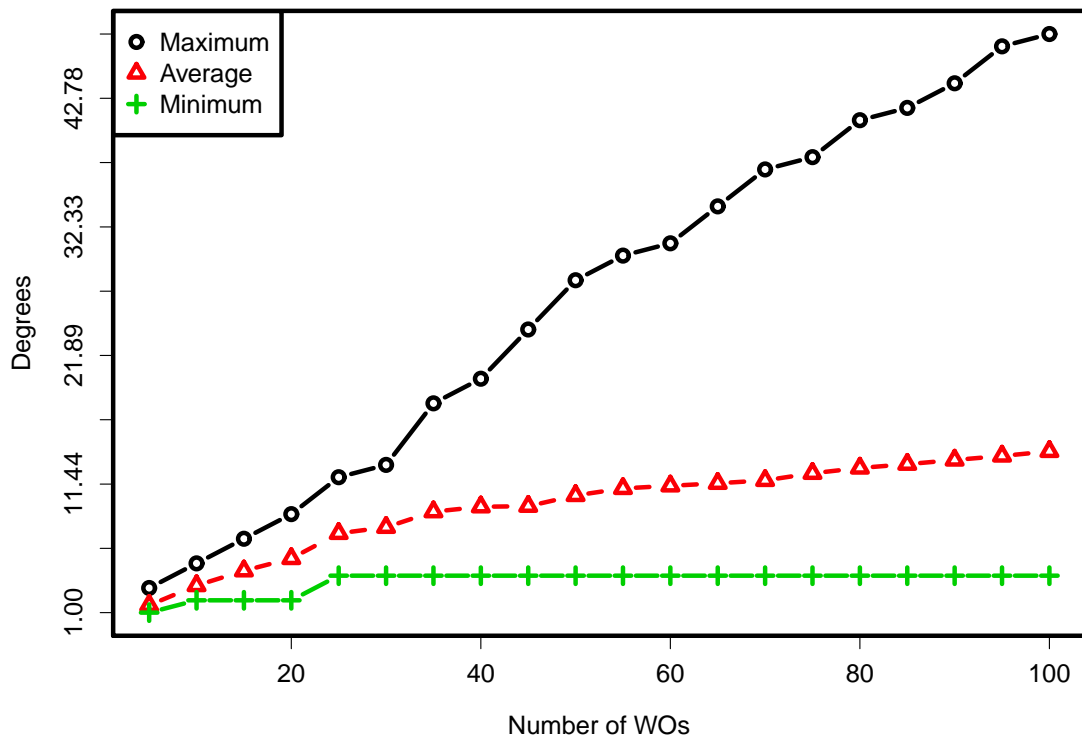


Figure 91. Degrees of USW simulation. The maximum degrees continues to rise as the number of WOs increases. This is due to the policy of always using the same first WO when introducing a wandering WO to the USW graph. It is a variant of “preferential attachment” [184] because in the beginning, the reciprocal connections between WOs tends to reinforce the edges going to the first WO. As the USW graph in real life, we expect that wandering WOs will be introduced to a variety of WOs and the phenomenon shown in the simulation will not occur.

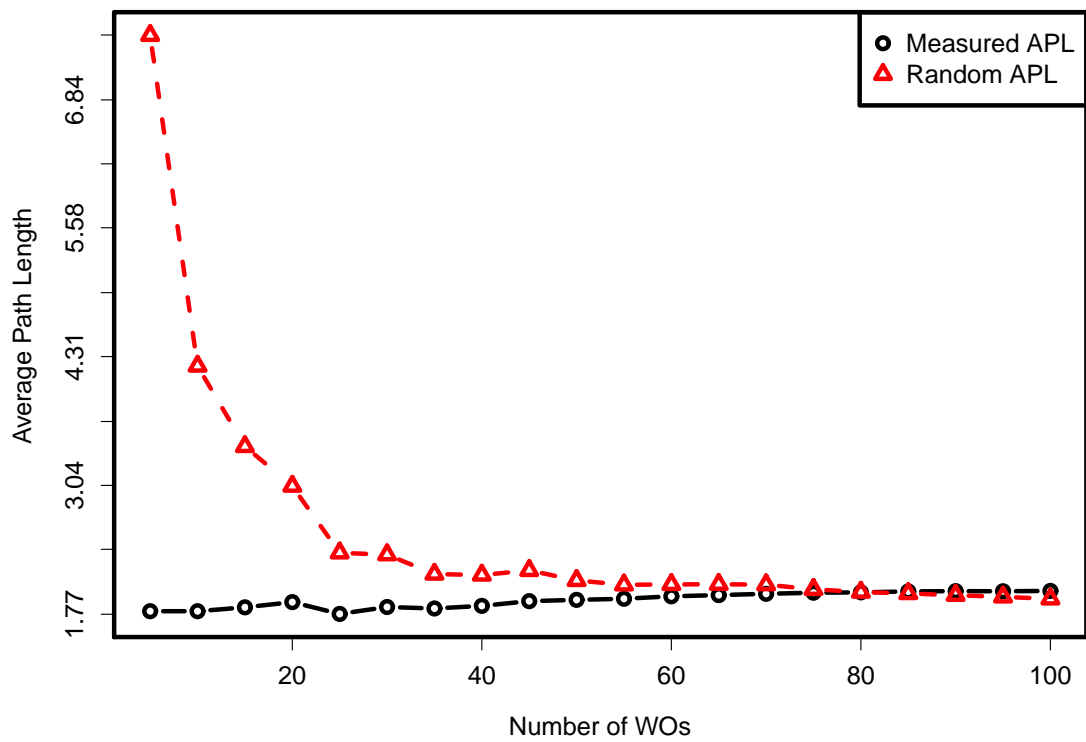


Figure 92. APL of USW simulation. The USW measured $L(G)$ very closely approximates the computed $L(G)$ for a random graph.

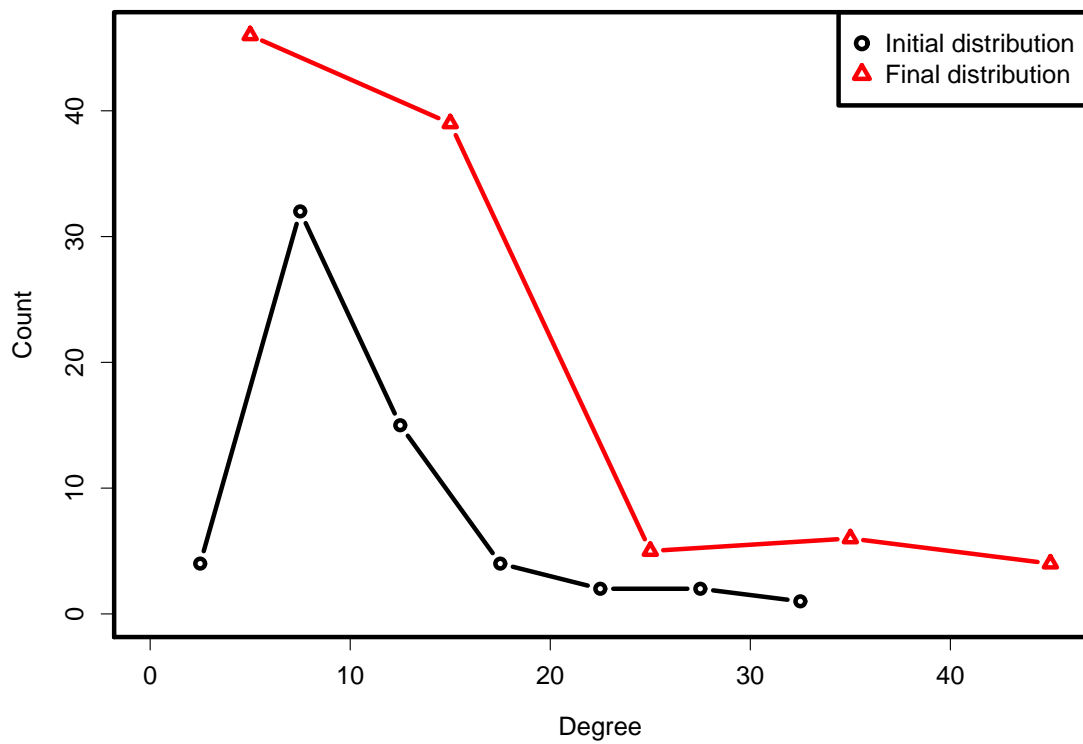


Figure 93. Temporal degree histogram of USW simulation. As the USW graph grows, the degree histogram it takes on a power law shape.

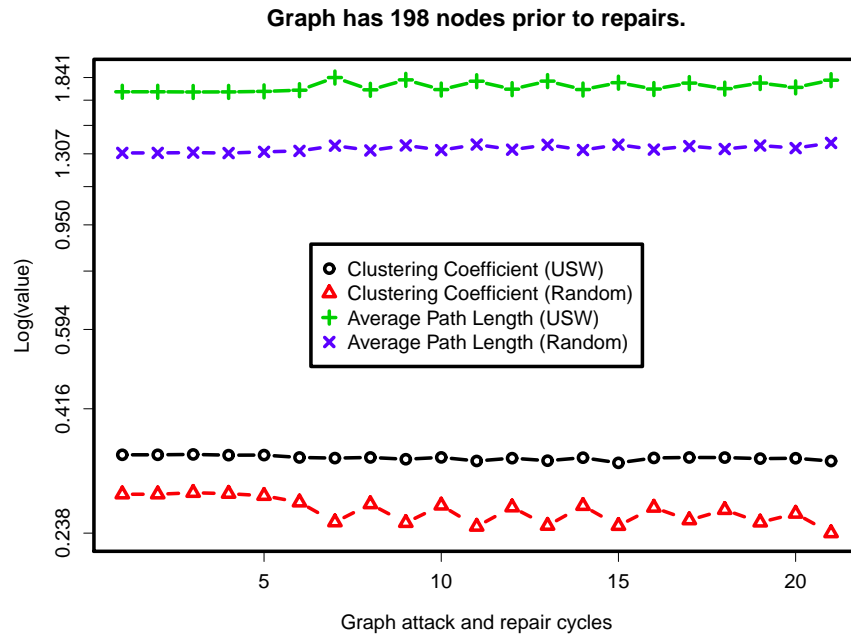


Figure 94. Graph metrics after repeated low percentage (5%) attacks and repairs.

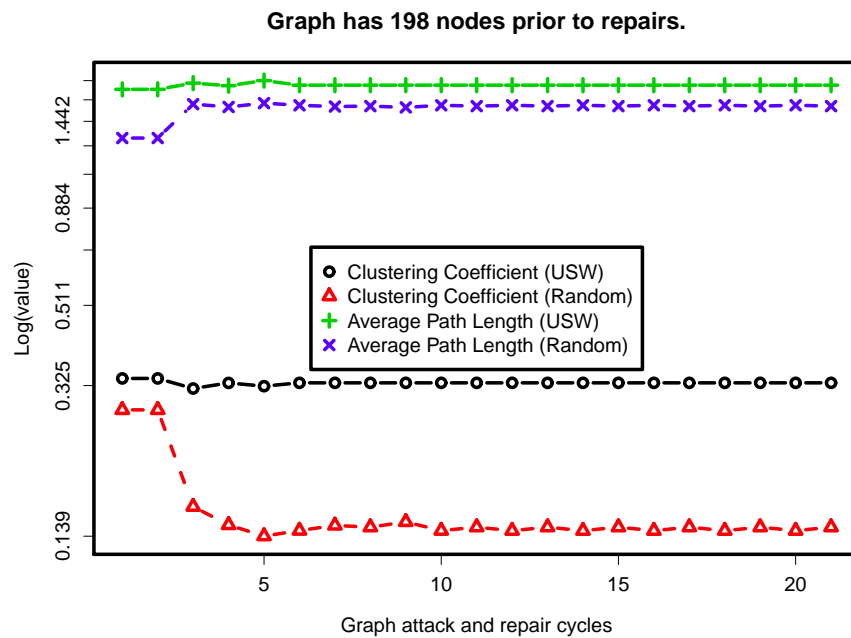


Figure 95. Graph metrics after repeated high percentage (50%) attacks and repairs.

CHAPTER 7

REFERENCE IMPLEMENTATION

7.1 INTRODUCTION

We have developed the USW algorithm and shown that it is capable and viable in a simulation environment. We now take it the next level by building a small scale reference implementation using a representative set of different web pages, and using the idea of “crowd-sourcing” via instrumented HTML pages to test the USW algorithm.

We create a mini-web environment based on HTML pages downloaded from four different domains. In our mini environment, we create the minimal necessary support services and communications infrastructure to allow WOs to create copies, modify themselves, and exchange messages.

After creating this environment, we compare the USW graph to a random graph of the same order and size.

7.2 DIFFERENCES BETWEEN USW THEORY AND REFERENCE IMPLEMENTATION

We developed the USW theory without any implementation constraints; we assumed that various critical USW aspects (communication, connections, etc.) were in place and working correctly. Many of these aspects were easily ignored when studying USW performance in a simulation environment. When moving from a simulation environment to a reference implementation, some of these aspects become problematic (Table 45 on the following page). The number and types of HTTP methods available also varies considerably on publicly available servers [187]. A potential by-product of the temporal aspects of moving from a simulation to a reference implementation is the possibility that the WOs that a WO is connected to will on average have more connections than the original WO [188]. This is partially due to WOs that the original WO is connected to can double-count WOs that are shared between WOs.

7.3 IMPLEMENTATION ENVIRONMENT

Table 45. Differences between USW simulation and reference implementation.

Area	USW Theory	HTTP/HTML reality	Impact
Communications	Instantaneous	Asynchronous	WOs have to be accessed to read and process messages. An HTTP mailbox is used to exchange messages [68].
Edges	Bidirectional	Directional	Some of graph metrics (such as $L(G)$) may not work because paths may not exist between all WOs. Extra time will be required to fully create bi-directional links.
Temporal effects	Non-existent	Present	The reference implementation may take considerably longer than the simulation environment to stabilize. A longer stabilization time would permit an attacker more time to mount an effective attack.

The minimum web infrastructure (WI) needed by the USW algorithm is:

1. A set of domains that has USW instrumented pages (instrumenting the pages changes the pages from digital objects to web objects),
2. A way for a WO to modify itself,
3. A way to create WOs, and
4. A way for WOs to communicate with each other.

Each of these needs is addressed in the following sections.

7.3.1 DOMAINS

Four different web domains were selected based on the variety of web page structures (Table 46 on the next page). In three of the domains, 100 pages were downloaded, while only 1 was downloaded from the fourth. Only one page was downloaded from *www.gutenberg.org* because of their stated monitoring of client activity and that any excessive activity would result in the black listing of the client's domain. If *www.gutenberg.org* decided there was too much activity from the *cs.odu.edu* domain, they may have decided to black list all of *odu.edu*. Based on previous USW algorithm analysis, the page from *www.gutenberg.org* was designated the "well known WO" that all new WOs would use as their gateway into the USW graph (Policy A).

As each page was down loaded from its original domain to its respective *cs.odu.edu* domain, it was instrumented with USW lines (see Listing 1), and a REM was created for the page. Each page's REM contains USW metadata and data specific to that page.

```
<link rel="resourcemap" type="application/atom+xml;type=
  entry" href="http://arxiv.cs.odu.edu/rems/arxiv
  -0704-3647v1.xml" />
<link rel="aggregation" href="http://arxiv.cs.odu.edu/
  rems/arxiv-0704-3647v1.xml#aggregation" />
<script src="http://www.cs.odu.edu/~salam/wsd1/uswdo/work
  /preserveme.js"></script>
```

Listing 1. USW implementation instrumentation. Each HTML page was instrumented with these lines.

Table 46. Domains that provided HTML pages for USW implementation testing.

Source domain	Num. of HTML pages	Destination domain
www.arxiv.org (Figure 96)	100	arxiv.cs.odu.edu
www.flickr.com (Figure 97 on the following page)	100	flickr.cs.odu.edu
www.gutenberg.org (Figure 98 on page 258)	1	gutenberg.cs.odu.edu
www.radiolab.org (Figure 99 on page 259)	100	radiolab.cs.odu.edu

The screenshot shows a Mozilla Firefox browser window displaying an arXiv page. The address bar shows the URL arxiv.org/abs/1202.4185. The page header includes the Cornell University Library logo and a search bar. The main content area features the title "When Should I Make Preservation Copies of Myself?" by Charles L. Cartledge and Michael L. Nelson, submitted on 19 Feb 2012. The abstract discusses preservation policies for digital objects. The right sidebar contains download options (PDF, PostScript, Other formats), current browse context (cs.DL), and references (NASA ADS). The submission history shows it was submitted on Sun, 19 Feb 2012 21:03:33 GMT (532kb).

Figure 96. Sample HTML page from arXiv domain.

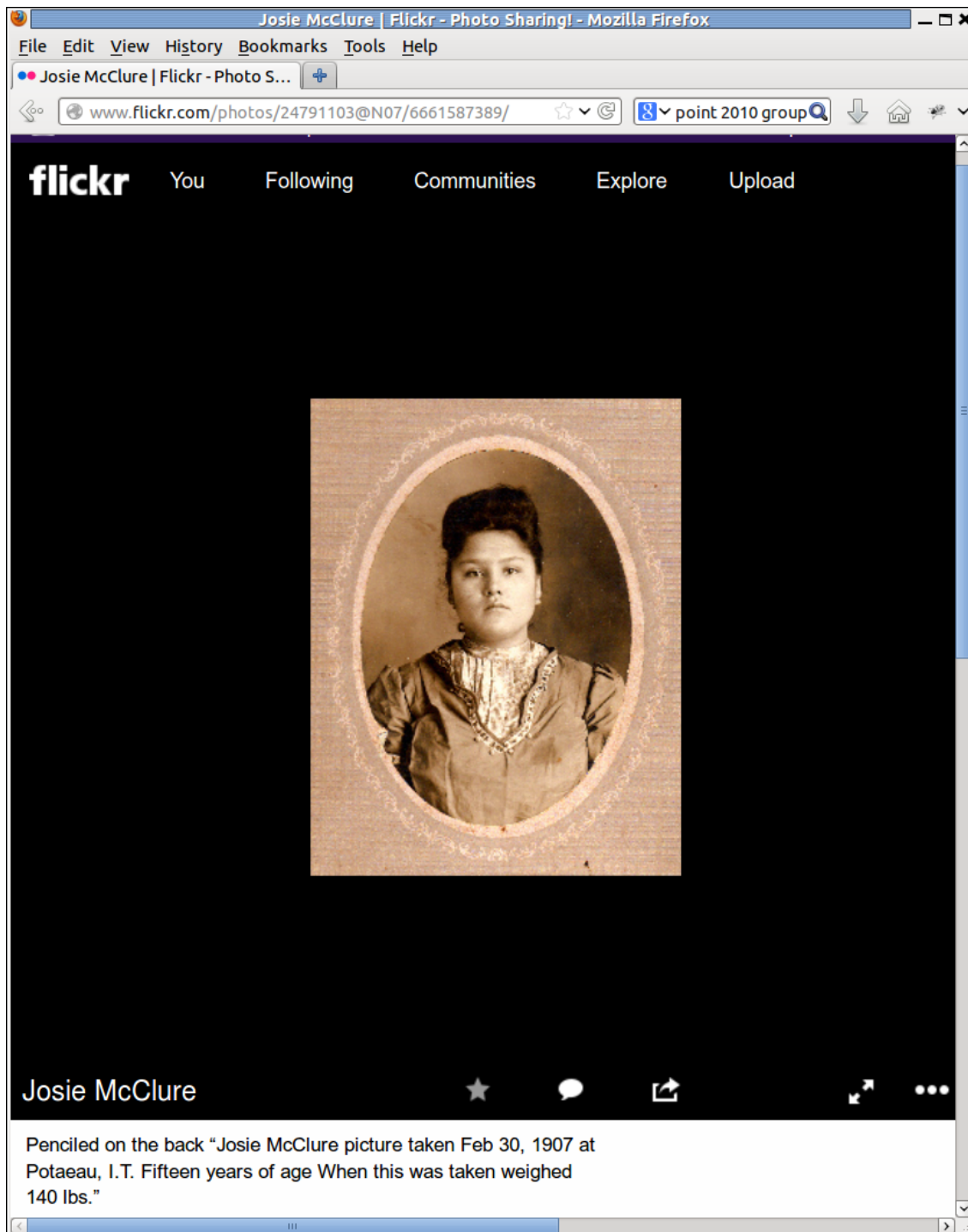


Figure 97. Sample HTML page from flickr domain.

The screenshot shows a Mozilla Firefox browser window with the address bar displaying `www.gutenberg.org/ebooks/19033`. The page title is "Alice's Adventures in Wonderland by Lewis Carroll - Free Ebook - Mozilla Firefox". The browser's menu bar includes "File", "Edit", "View", "History", "Bookmarks", "Tools", and "Help". The address bar contains the URL and a search icon. Below the address bar, there is a navigation bar with the Project Gutenberg logo, a search bar containing "alice s adventure in", and a "Help" button. The main content area features the Project Gutenberg logo, a search bar, and a "Help" button. The page title is "Alice's Adventures in Wonderland by Lewis Carroll". Below the title, there is a "Download" button and a "Bibrec" button. A table titled "Download This eBook" lists various download options with their sizes and icons for social media sharing. To the left of the table is a book cover for "ALICE IN WONDERLAND" by LEWIS CARROLL, and below it are social media icons and a QR code.

Project Gutenberg offers 45,263 free ebooks to download. [Donate](#) [Flattr this!](#)

[Search](#) [Latest](#) [Terms of Use](#) [Bookmarks](#) [Donate?](#) [Mobile](#) [Help](#)

[Project Gutenberg](#) › [45,263 free ebooks](#) › [28 by Lewis Carroll](#)

Alice's Adventures in Wonderland by Lewis Carroll

[Download](#) [Bibrec](#)

Download This eBook

Format ?	Size	?	?	?
Read this book online: HTML	90 kB			
EPUB (with images)	535 kB	?	?	?
EPUB (no images)	72 kB	?	?	?
Kindle (with images)	1.1 MB	?	?	?
Kindle (no images)	238 kB	?	?	?
Plain Text UTF-8	73 kB			
More Files...				

[ALICE IN WONDERLAND](#)
LEWIS CARROLL

[f](#) [+](#) [t](#) [p](#)

Figure 98. Sample HTML page from gutenberg domain.



Figure 99. Sample HTML page from radiolab domain.

7.3.2 SUPPORT SERVICES

A WO has to be able to modify itself (to keep track of its preservation copies, to maintain a list of friend connections, to “remember” the last time it checked for messages, and so on) and to make preservation copies of of WOs.

Edit Service

The REM Edit Service associated with each unique WO is an HTTP service and is accessed via an HTTP POST command (see Listing 2).

```
curl -m 120 -i -X POST --data-binary @/tmp/temp~.
  xxx3283ace9 -i -H "Sender: http://flickr.cs.odu.edu/
  rems/flickr-ceotty-8161751828.xml" -H "Content-type:
  application/patch-ops-error+xml" http://ws-dl-02.cs.
  odu.edu:10101/hm/http://gutenberg.cs.odu.edu/rems/
  gutenberg-pride-and-prejudice.xml 2>/dev/null "
```

Listing 2. Sample Edit Service request.

The Edit Service expects an XML patch directive [132] (see Listing 3).

```
PATCH /rems/gutenberg-pride-and-prejudice.xml HTTP/1.1
Host: gutenberg.cs.odu.edu
Content-type: application/patch-ops-error+xml
Content-length: 216

<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry"><link rel="http://wsdl.cs.odu.edu/uswdo/
  terms/friend" href="http://flickr.cs.odu.edu/rems/
  flickr-ceotty-8161751828.xml" title="Kittens" /></add>
</diff>
```

Listing 3. Sample Edit Service request.

Copy Service

The REM Copy Service associated with each unique WO is an HTTP service and is accessed via an HTTP POST command (see Listing 4).

```
curl -m 120 -i -H "Sender: http://arxiv.cs.odu.edu/rems/
  arxiv-0912-0201v1.xml" -X POST --data-binary @/tmp/
  temp~.xxx258b667a http://ws-dl-02.cs.odu.edu:10101/hm/
  http://gutenberg.cs.odu.edu/rems/gutenberg-pride-and-
  prejudice.xml 2>/dev/null "
```

Listing 4. Sample Copy Service request.

The Copy Service expects a complete REM (see Listing 5). Upon successful completion of the Copy request, the service returns the URL where the copy was created.

```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:oreatom=
  "http://www.openarchives.org/ore/atom/" xmlns:dcterms=
  "http://purl.org/dc/terms/" xmlns:dc="http://purl.org/
  dc/elements/1.1/" xmlns:rdf="http://www.w3.org
  /1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.
  org/2000/01/rdf-schema#" xmlns:ore="http://www.
  openarchives.org/ore/terms/" xmlns:foaf="http://xmlns.
  com/foaf/0.1/" xmlns:grddl="http://www.w3.org/2003/g/
  data-view#" xmlns:relationship="http://purl.org/vocab/
  relationship/" xmlns:usw="http://wsdl.cs.odu.edu/uswdo
  /terms/" grddl:transformation="http://www.openarchives
  .org/ore/atom/atom-grddl.xsl" xmlns:le="http://purl.
  org/atompub/link-extensions/1.0">
<id>tag:uswdo.cs.odu.edu,2012-11-01:arxiv-0912-0201v1</
  id>
<link rel="alternate" type="text/html" href="http://
  arxiv.cs.odu.edu/arxiv-0912-0201v1.html" />
<link rel="self" type="application/atom+xml" href="http:
  //arxiv.cs.odu.edu/rems/arxiv-0912-0201v1.xml" />
```

```

<link rel="edit" type="application/atom+xml" href="http://ws-dl-02.cs.odu.edu:10102/rem/edit/http://arxiv.cs.odu.edu/rem/arsiv-0912-0201v1.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/copy" type="application/atom+xml" href="http://ws-dl-02.cs.odu.edu:10102/rem/copy/http://arxiv.cs.odu.edu/" />
(lines removed)
<source>
<author>
<name>ODU WSDL ReM Generator</name>
<uri>http://ws-dl-02.cs.odu.edu/</uri>
</author>
</source>
<published>2013-07-30T18:25:13-04:00</published>
<updated>2013-07-30T18:25:13-04:00</updated>
<link rel="license" type="application/rdf+xml" href="http://creativecommons.org/licenses/by-nc/2.5/rdf" />
<rights>This Resource Map is available under the Creative Commons Attribution-Noncommercial 2.5 Generic license</rights>
<title>LSST Science Book, Version 2.0</title>
<author>
<name>LSST Science Collaborations</name>
</author>
(lines removed)
<category term="http://www.openarchives.org/ore/terms/Aggregation" label="Aggregation" scheme="http://www.openarchives.org/ore/terms/" />
<category term="2009-12-01T16:50:56+00:00" scheme="http://www.openarchives.org/ore/atom/created" />
<category term="2009-12-01T16:50:56+00:00" scheme="http://www.openarchives.org/ore/atom/modified" />
<category term="3" scheme="http://wsdl.cs.odu.edu/uswdo/terms/preservationCopiesMinimumNumber" />

```



```

<category term="5" scheme="http://wsdl.cs.odu.edu/uswdo/
  terms/preservationCopiesMaximumNumber" />
<category term="0.85" scheme="http://wsdl.cs.odu.edu/
  uswdo/terms/beta" />
<category term="0.10" scheme="http://wsdl.cs.odu.edu/
  uswdo/terms/gamma" />
(lines remove)
</entry>

```

Listing 5. Sample Copy Service REM. Many lines have been removed for clarity.

7.3.3 MESSAGE SERVER

WOs need to be able to communicate with each other, to send requests to make friendship connections, to send requests to make preservation copies, and to manage the active maintainer status across USW family members. Inside of each WOs' REM is the location of the mailboxes that that WO receives messages from (see Listing 6). Each WO receives messages from three logically different mailboxes and can be checked at different times. They are:

- *#self*: messages that are meant to be received by only the intended WO recipient. This is akin to a network point-to-point communication.
- *#family*: messages that are meant to be received by all members of a particular USW family. This is akin to a network multicast communication.
- *#all*: messages that are meant to be received by all WOs. This is akin to a network broadcast communication.

```

<link rel="http://wsdl.cs.odu.edu/uswdo/terms/httpmailbox
  #self" href="http://ws-dl-02.cs.odu.edu:10101/hm/http:
  //flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml"
  usw:last-checked="" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/httpmailbox
  #family" href="http://ws-dl-02.cs.odu.edu:10101/hm/
  tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty
  -8161751828" usw:last-checked="" />

```

```
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/httpmailbox
#all" href="http://ws-dl-02.cs.odu.edu:10101/hm/all"
usw:last-checked="" />
```

Listing 6. Sample mailbox lines from a REM.

The message server is based on work done by Alam [68, 189].

7.3.4 USW GRAPH VISUALIZATION

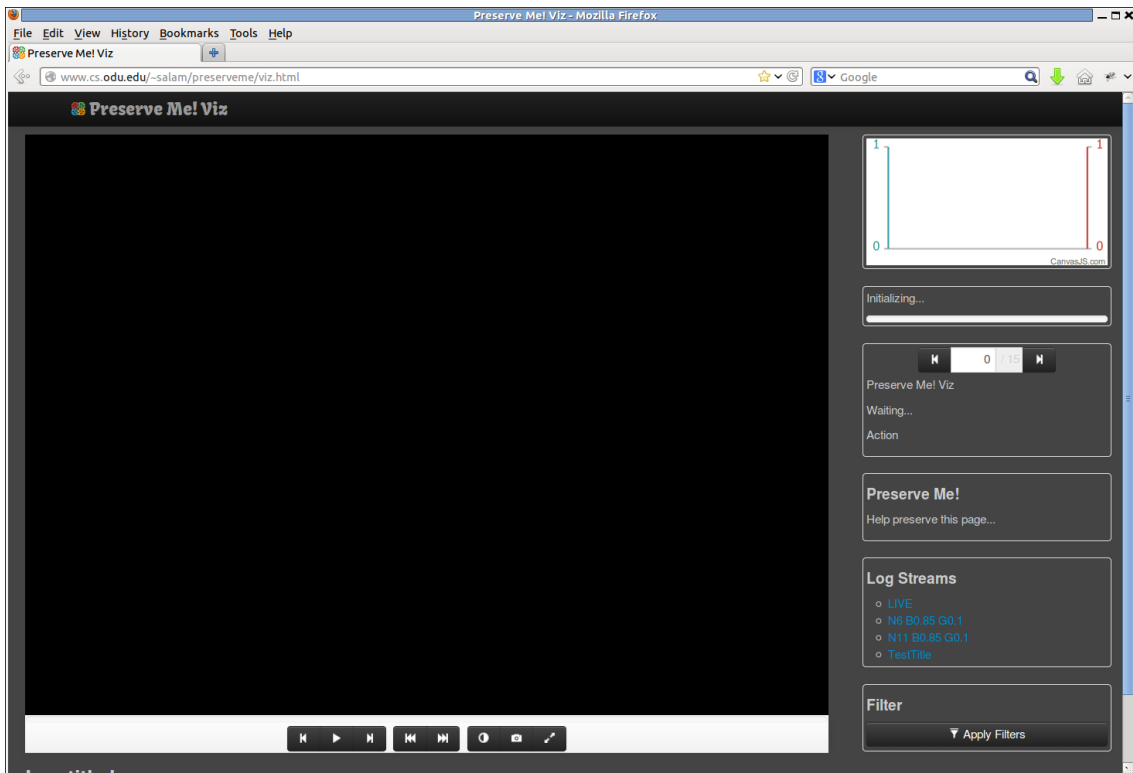
The USW graph exists on the surface of the WI and can be hard to visualize, and the idea of WOs communicating with one another requires an additional leap of faith. We created a USW visualizer called “Preserve Me! Viz” to address these concerns (Figure 100 on the following page). The visualizer receives messages sent by either the USW web page client or robot as JSON formatted objects and plots the information on the the display.

Data that is permanently displayed include:

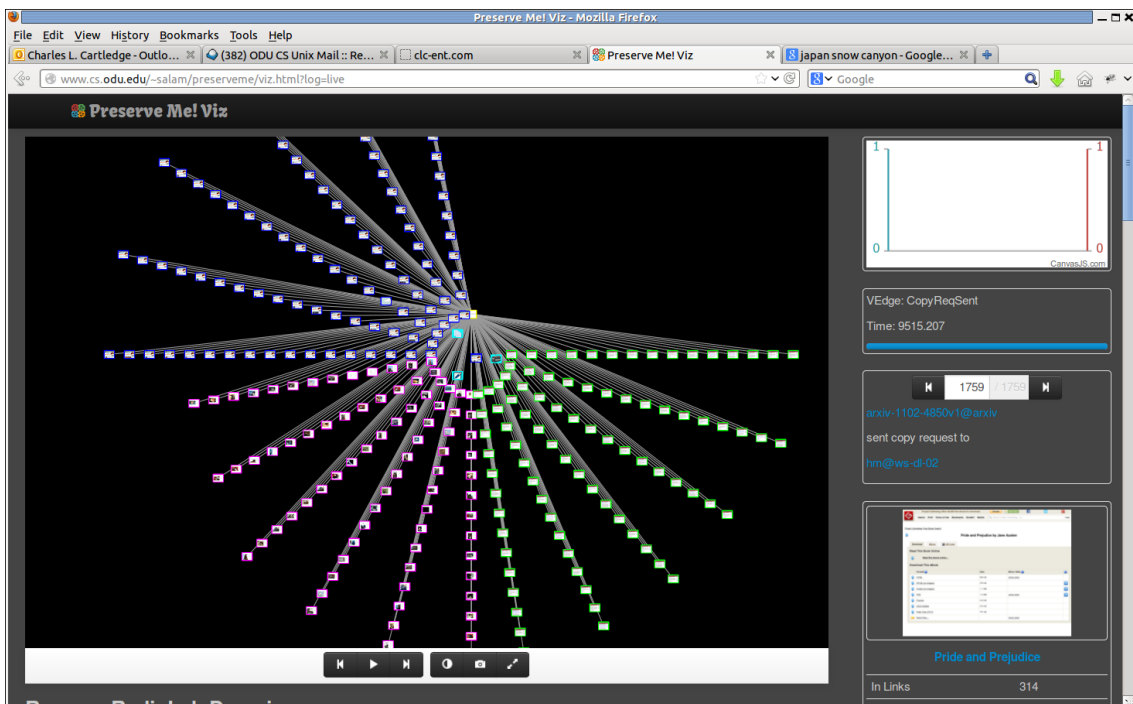
1. Icons representing WOs and services,
2. Friendship links between WOs,
3. Family links between family members,
4. Current $L(G)$ and $C(G)$,
5. The type of message just processed,
6. Time until the next message is processed,
7. The sender and receiver from the last message, and
8. If a previously recorded message stream is being displayed, then a title and associated explanatory text may be displayed.

Data that is intermittently displayed include:

1. Communication events between the sender and the mail service,
2. Communication events between the received and the mail service,
3. Requests and replies between a WO and the copy service, and



(a) A blank display



(b) A display with data

Figure 100. The Preserve Me! Viz display.

4. Directives sent from a WO and its edit service.

7.4 USW IMPLEMENTATION

USW WOs communicate with each other and are responsible for the maintenance of the USW graph and the preservation of copies of them selves. USW WOs lie dormant and fallow unless they are accessed and they can interact with each other. We envision two different ways of interacting with the USW graph. The first supports crowd-sourced preservation through the use of a pop-up window that a human being interacts with. This pop-up is the public face of the USW algorithm. The second way to interact with the graph is through the use of a robot or crawler. We have created a web page client and a robot.

Both the web page client and the robot implement the complete USW algorithm (Appendix A on page 287) and WO to WO communications (Appendix B on page 311).

Web page client

The web page client serves as the primary human interface into the USW algorithm. The USW algorithm can appear to be very complex to someone not totally versed in its nuances. Because the expected user will not be totally versed in the USW algorithm, two different views into the USW algorithm are presented, “Basic” and “Advanced.”

The “Basic” interface walks the user through the major actions that a USW WO would undertake:

1. Connect to the USW graph (Figure 101 on the next page),
2. Make connections to other WOs in the graph (Figure 102 on page 268) , and
3. Make and maintain preservation copies in the graph Figures 103 on page 269, 104 on page 270, and 105 on page 271.

USW robot

Primary inputs to the robot are the gateway WO to which all other WOs will be introduced and a list of HTML pages to be added to the USW graph. While

Preserve Me! - Mozilla Firefox

www.cs.odu.edu/~salam/preserveme/preserveme.html?rem=http%3A%2F%2Fgutenberg.cs.odu.edu%2Frem%2Fgutenberg-pride-and-prejudice.xml

Preserve Me! Advanced

Dynamic Linking of Smart Digital Objects Based on User Navigation Patterns

Author(s)
Aravind Elango, Johan Bollen, Michael L. Nelson

Status: **Disconnected!** (No friends) Updated: 7 months ago

I have no Friends!

This Resource Map is a lonely soul and it has got no life! Please help it connect with an existing network of web objects.
[Need Help?](#)

Using gateway

Connect to Network

Web Science and Digital Libraries Group
Department of Computer Science Old Dominion University
Norfolk VA - 23529

Figure 101. USW client basic “make connection” popup. The WO has determined that it does not have any friend connections and must make some.

Preserve Me! - Mozilla Firefox

www.cs.odu.edu/~salam/preserveme/preserveme.html?rem=http%3A%2F%2Farxiv.cs.odu.edu%2Frem%2Farxiv...

Preserve Me! Advanced

Dynamic Linking of Smart Digital Objects Based on User Navigation Patterns

Author(s)
Aravind Elango, Johan Bollen, Michael L. Nelson

Status: **Disconnected!** (No friends) Updated: 7 months ago

Connecting to the Network...

- Load gateway ResourceMap (Events: 101)
- Explore USW graph (Events: 102)
- Send friendship requests (Events: 104)
- Connect to the network (Events: 103)

Congratulations! Connected to the network succesfully. To start making copies, please Reload

Web Science and Digital Libraries Group
Department of Computer Science Old Dominion University
Norfolk VA - 23529

Figure 102. USW client basic make connection popup.

The screenshot shows a Mozilla Firefox browser window with the URL `www.cs.odu.edu/~salam/preserveme/preserveme.html?rem=http%3A%2F%2Farxiv.cs.odu.edu%2Fprems%2Farxiv`. The page header includes the 'Preserve Me!' logo and an 'Advanced' button. The main content area features a blue header for 'Dynamic Linking of Smart Digital Objects Based on User Navigation Patterns'. Below this, there is a thumbnail of the resource map, the author(s) 'Aravind Elango, Johan Bollen, Michael L. Nelson', and a status bar indicating 'Status: Danger! (Number of copies: 0)' and 'Updated: 2 minutes ago'. A prominent red warning box states 'Not Enough Copies!' and 'This Resource Map needs more copies to survive. Please help making more copies of it. [Need Help?](#)'. A large red button labeled 'Make More Copies' is positioned below the warning. The footer contains the text 'Web Science and Digital Libraries Group', 'Department of Computer Science Old Dominion University', and 'Norfolk VA - 23529'.

Figure 103. USW client basic make copies popup.

Preserve Me! - Mozilla Firefox

www.cs.odu.edu/~salam/preserveme/preserveme.html?rem=http%3A%2F%2Ffarxiv.cs.odu.edu%2Frem%2Ffarxiv-...

Preserve Me! Advanced

Dynamic Linking of Smart Digital Objects Based on User Navigation Patterns

Author(s)
Aravind Elango, Johan Bollen, Michael L. Nelson

Status: **Danger!** (Number of copies: 0) Updated: 4 minutes ago

Making Copies...

- Discover copy hosts (Events: 101, 102, 103, and 104)**
- Send copy requests (Events: 201)**

Copy request has been sent to the following:

1. <http://gutenberg.cs.odu.edu/rem/gutenberg-pride-and-prejudice.xml>
2. <http://flickr.cs.odu.edu/rem/flickr-ceotty-8161751828.xml>
3. <http://radiolab.cs.odu.edu/rem/radiolab-mix-and-match.xml>

A copy may be made when someone visits the link(s) listed above.

Web Science and Digital Libraries Group
Department of Computer Science Old Dominion University
Norfolk VA - 23529

Figure 104. USW client basic copies request popup.

Preserve Me! - Mozilla Firefox

www.cs.odu.edu/~salam/preserveme/preserveme.html?rem=http%3A%2F%2Fflickr.cs.odu.edu%2Frem%2Fflickr-arxiv-cs-0401029v1.xml

Preserve Me! Advanced

Dynamic Linking of Smart Digital Objects Based on User Navigation Patterns

Author(s)
Aravind Elango, Johan Bollen, Michael L. Nelson

Status: Safe! (Number of copies: 2) Updated: less than a minute ago

Not Enough Copies!

This Resource Map needs more copies to survive. Please help making more copies of it. [Need Help?](#)

[Make More Copies](#)

Existing Copies

- <http://flickr.cs.odu.edu/copyrems/arxiv-cs-0401029v1.xml>
- <http://gutenberg.cs.odu.edu/copyrems/arxiv-cs-0401029v1.xml>

Web Science and Digital Libraries Group
Department of Computer Science Old Dominion University
Norfolk VA - 23529

Figure 105. USW client basic with copies popup.

there are a number of additional arguments available (see Section C.2 on page 377), the gateway and the WO are the ones of greatest interest. The robot also supports the testing the loss of a WO or an entire domain to test the transfer of active maintenance from one WO to another.

Table 47. Comparing USW implementation and random graph metrics. The relative sizes of $\langle k \rangle$ and $\langle k \rangle \sigma$ indicate the possibility that the $\langle k \rangle$ distribution does not fit an ideal “normal” distribution. This possibility is confirmed by examining the degree histogram for this USW graph (Figure 106 on the following page).

Metric	USW	Random
n	156	(same)
$ E $	539	(same)
$\langle k \rangle$	6.91	(same)
$\langle k \rangle \sigma$	21.04	N/A
$C(G)$	0.07	0.04
$L(G)$	1.96	2.08

Table 48. Comparing USW implementation copies and desired copies.

	Copies
Total desired minimum	468
Actual number	164
Total desired maximum	780

7.5 RESULTS

We used the USW robot to create a USW graph for comparison with a random graph of the same order (Table 47). The robot was used to create USW graphs of different orders within the constraints of the number of live web pages that had been downloaded to the test environment. Execution time was recorded for each order (Table 49 on the following page). The execution times fit Equation 102. Table 48 lists the number of copies that were made as compared to system total c_{hard} and c_{soft} .

$$\text{hours} = 1.586918 + (-0.041927)n + (0.000373)n^2 \quad (102)$$

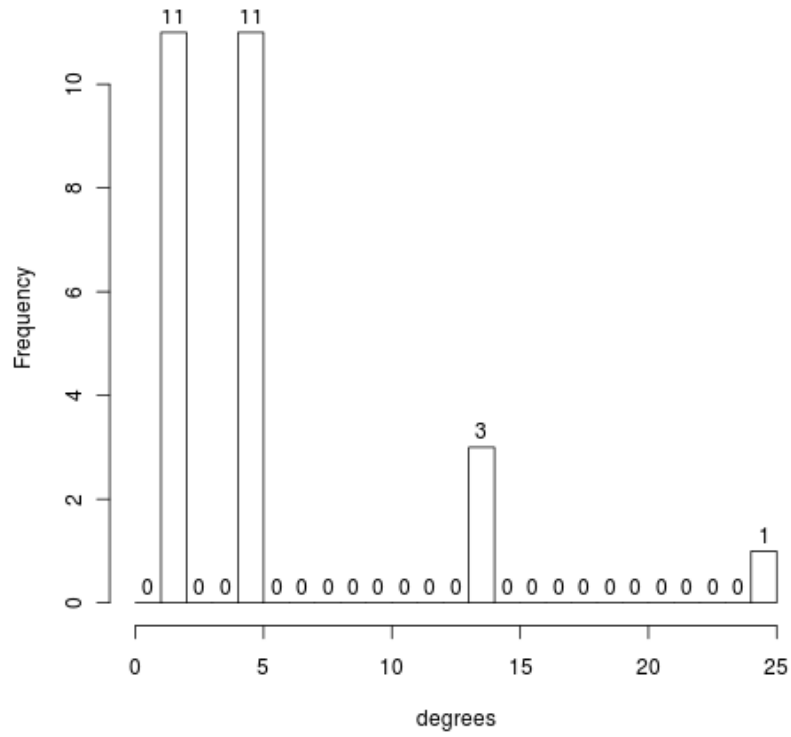


Figure 106. Histogram of an implemented USW graph.

Table 49. Robot execution times for various implementation order USW graphs.

Order	Hours
10	0.75
50	1.15
156	3.78
301	22.85

CHAPTER 8

FUTURE WORK

We have identified a number of areas where the USW precepts and algorithms can and should be expanded and improved. These areas are expanded in the following sections.

8.1 IMPROVED CONNECTION SELECTION

The current approach to creating unsupervised small-world (USW) graphs that rely on locally gathered graph data, arbitrary values of β , γ and selection policies to be used after a *wandering* web object (WO) makes its first connection to another WO in the USW graph. We present an idea for improving the USW graph creation that should result in graphs that more closely meet the quantitative requirements of a small-world than are currently being met.

8.1.1 INTRODUCTION

A USW web object (WO) communicates with WOs that are already connected in the USW graph and collects data about the $k = 1$ neighborhood around each of these established and connected WOs. During the process, the wandering WO maintains a list of WOs that it has contacted to ensure that established WOs are only contacted once. After a connection is made, a γ portion of the visited and not yet visited WOs are forcibly connected to the no-longer wandering WO.

8.1.2 DISCUSSION

At its core, the USW graph creation process is based on random selection. The initial connection is based on a random number compared to an acceptance threshold and all connections are then randomly selected. Connection after the initial one can be selected based on when they were discovered (LIFO or FIFO queues) or their degree, but at the core it is mostly a random selection. The decision on which connections to make after the initial one can be improved and even maximized.

Possible improvements are:

1. Currently the wandering WO contacts each established WO and requests the $k = 1$ neighbors of the established WO. This list of neighbors is then compared to the list of WO that have yet been visited and those that have already been visited. WOs that are not in either list are added to the list of WOs to be visited. The act of adding these newly discovered WOs just to the list of WOs to be visited, gives up a tremendous amount of information about the USW graph. If the connection information about between the established WO and its neighbors is retained, then the wandering WO can start to create a more accurate “map” of the USW graph. The size of this USW graph map will be limited by the number of established WOs that the wandering WO contacts.
2. When the wandering WO makes it first connection to an established WO, the “map” can be put to use.
3. Currently after the initial connection, a γ portion of the WOs that the wandering WO has learned about are forcibly connected to the wandering WO. The improvement would be to take the partial USW graph (when the USW graph becomes large, it seems very unlikely that the wandering WO will explore the entire graph before making its first connection) and make connections to the best γ WOs that reinforce the small-world aspects of high $C(G)_{Average}$ and acceptable $L(G)$ when compared with the USW graph prior to the wandering WO making its connections.
4. Computing the $L(G)$ is in worst case $\Theta(V^3 \lg V)$ (from [185]) and we assume that computing $C(G)_{Average}$ is equally as costly.
5. Watts – Strogatz in [43] plot normalized $C(G)_{Average}$ and $L(G)$ values and define the region where $C(G)_{Average}$ is high and the $L(G)$ is low as the small-world region. Watts and Strogatz were able to use normalized data because they started their exploration from a lattice graph which they proceeded to perturb. USW graphs do not start out as lattices, but we can use the idea of normalization and detection of large differences between normalized $C(G)_{Average}$ and normalized $L(G)$ as a value function to enable the wandering WO to make connections that optimize the value function.
6. The key to the improved processing is a nested For loop (Algorithm 19 page 308) that is used to evaluate each possible improvement to the graph. USW WOs

are ordered (see Algorithm 17 on page 305) after each change is made to the graph so that decisions are made on the most recent data available. Each change to the graph is evaluated (see Algorithm 18 on page 306) and then compared with an exponentially averaged value of the previous values. An exponential averager is used to compensate for the “noisiness” of the $C(G)_{Average}$ and $L(G)$ values. While the figures in Watts and Strogatz [43] show smoothly changing values, experimentation with other graphs have shown very “noisy” data. The dampening value of 0.7 is based on experience with other exponential averages. The WOs are searched in decreasing path length order. This order was chosen because reducing the $L(G)$ will have the greatest and most rapid improvement in the overall graph structure. As each change is evaluated, the first candidate WO that causes the smoothed exponential average to decrease causes the search to terminate. A connection request message is sent to the candidate WO and the local copy of the USW graph is updated assuming that the candidate WO will in fact make the connection.

7. WO ordering based on the current local graph, exhaustive evaluation of candidate WO connections and request for real connections repeats until the γ number of WOs has been identified and connected to.

The cost to compute $L(G)$ one time is $\Theta(V^3 \lg V)$. Because we will be computing it multiple times based on γ , the more likely computation time will be $\Theta(2 * V^{3+\gamma} \lg V)$ assuming that the time to compute $C(G)_{Average}$ is approximately the same as $L(G)$.

8.2 HANDLING MULTIPLE WO MAILBOXES

Currently a WO’s REM contains this single line (broken for clarity) giving the location of the WO personal mailbox (see Listing 7).

```
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#self"
href="http://ws-dl-02.cs.odu.edu:10101/hm/http:
//flickr.cs.odu.edu/rem/flickr-ceotty-8161751828.xml
"
```

Listing 7. Current WO personal mailbox.

The REM can be expanded to contain multiple lines such as:

```

    <link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#self"
    href="http://ws-dl-02.cs.odu.edu:10101/hm/http:
//flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml
"
    <link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#self"
    href="http://ws-dl-03.cs.odu.edu:10102/hm/http:
//flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml
"
    <link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#self"
    href="http://ws-dl-04.cs.odu.edu:10103/hm/http:
//flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml
"

```

Listing 8. Future work WO personal mailboxes.

The REM (see Listing 8) now declares three personal mailboxes. When a WO parses the REM as part of its persistent memory, it must treat mailboxes and possibly other data as a set of data and apply the appropriate processing to each entry in the list vice assuming that there is only one value.

The same approach can be used for any USW system parameters, to include *edit* and *copy* services.

8.3 IMPROVED AGGREGATE RESOURCE UPDATES

The USW process attempts to create enough copies of a WO's aggregated resources so that the essence of the WO is preserved for a long time. Resources may not be immutable, they may change at some rate relative to the expected life of the WO. They may change *often*, *infrequently*, *seldom*, or *never*. Each of these notional categories will be handled like this when a copy of WO is created:

- *Often*: an HREF to the original resource will be maintained intact.
- *Infrequently*: an HREF to the original resource may be maintained intact, or a copy of the resource will be made when the WO is copied.

- *Seldom*: a copy of the resource will be made when the WO is copied.
- *Never*: a copy of the resource will be incorporated into the WO.

8.4 IMPROVED REPLICATION PROCESSES

The basic premise of the USW process is that a “generic” web page is automatically instrumented with various pieces of JavaScript to entice the viewer to aid in the page’s long-term preservation [54].

When the first viewer elects to preserve the page, the hosting server makes a “snapshot” of the page and creates a web object (WO) from the page. The WO takes the form of an Atom based REsource Map (REM) [133] XML file. Individual resources in the original page are identified based on criteria unique to the hosting server. Copies of this REM are provided to copying services on other hosts and the REM is updated to reflect the location on the copying host of the original aggregated resources. This approach corresponds to “baseline synchronization” versus “incremental synchronization” [190].

8.4.1 DISCUSSION

At time t_1 , a web page with one resource (R_1) (for instance, a JPEG image) is selected by the viewer for long-term preservation using the USW process. The WO is copied to a number of additional servers resulting in multiple copies of R_1 .

At time t_2 , R_1 changes and becomes R'_1 . Within the web page, its identity remains the same, but the resource is changed (i.e., someone replaces the original JPEG image with another with the same name).

From the viewer’s perspective, the web page at times t_1 and t_2 are different things.

The USW process could recognize that R_1 has changed by comparing its current MD5 hash with the previous MD5. Given that the web page at time t_2 is not the same one as at t_1 , the WO could start the USW process again. This is the essence of replication.

If the USW process attempts to spread the R'_1 to its USW copies, then R_1 will be lost entirely. This loss occurs because the service making copy of the REM on its local host, can place copies of resources anywhere it chooses. Thus, it is not possible for the originating WO to predict where the copy of R'_1 would be placed. The only

choice that the originating WO has is to tell the copy service to replace R_1 with R'_1 . Resulting in the eventual loss of all copies of R_1 .

One possible solution is to serialize all synchronizations. Each “incremental synchronization” would be serialized (possibly using a UUID) and the serialization would become part of the WOs family name. This would result in:

1. Uniquely identifiable synchronization copies of the originating WO,
2. Large amounts of bookkeeping data (based on frequency of synchronizations).

Another possible solution is to use content based naming (CBN) to create unique names to replace all references to R [191]. Or alternatively, organizationally unique names that can be bridged to other organizations [192]. Routing and searching for CBN identified data will require investigation and experimentation beyond the scope of this dissertation. Using CBN techniques will change the details of the REMs, but will not change the USW precepts or concepts.

Resource synchronization was never a part of the USW baseline design. While possible approaches exist that might support synchronization, they each violate the basic premise in that they do not preserve what the original page viewer saw and wanted to preserve.

8.5 IMPROVED SECURITY

Currently the role of “active maintainer” is based on a comparing the respective WO’s creation timestamps. The WO with the earliest timestamp is designated as the active maintainer. An attacker could construct a false WO with an extremely early timestamp and assume the role of progenitor for any USW family. This is a security area that could be addressed. One possible approach would be to have all family members vote whether, or not to accept a WO’s claim as the active maintainer. Design and implementation of this voting process involving WOs that may, or may not be active at the same time could have application and impact in other areas of computer science.

CHAPTER 9

CONCLUSION AND CONTRIBUTIONS

9.1 CONCLUSION

We have identified two significant shortcomings in the current world of digital preservation. One is the expectation that institutions will preserve digital data for the long-term (greater than 100 years). We have shown many cases where institutions are unable or unwilling to preserve data they created, or were charged with maintaining for even as short as 20 years. The second shortcoming is selecting which digital data to preserve. The volume of digital data continues to grow at an exponential rate and will outstrip the capabilities of the curators in charge of the OAIS ingest process (Figure 107 on page 283) [193]. The cumulative effects of these shortcomings will be the loss of a significant portion of our personal and cultural heritage.

We have taken fundamental ideas from computer animation which have been used to foster emergent behaviors between independent entities and applied them to web objects. Specifically, we have reinterpreted Craig Reynolds' ideas of *collision avoidance*, *velocity matching*, and *flock centering* as seen to herd behavior and applied them as *name uniqueness*, *resource consumption*, and *host utilization*. The net effect of these ideas is to have web objects (WOs) that move and operate as independent members of a herd.

We have examined the underlying theory behind Barabási and Albert-László preferential attachment, Erdos-Renyi random, and Watts and Strogatz small-world graphs. We have studied each type of graph for their *robustness* and *resilience* in the face of *attacks* and *failures*. We have developed a metric to quantify the damage to the graph based on the loss of edges or vertices due to attack or failure. Based on the analysis of the many attack profiles that an adversary could use, and the occurrence of small-world graphs in nature and social context, we identified small-world graphs as both *robust* and *resilient*. Bringing together the ideas from computer animation and graph theory, we developed the unsupervised small-world algorithm

to create small-world graphs based on local knowledge. The USW graph serves as the framework for the long-term preservation of digital data.

We have identified and explored a different approach to solving the long-term preservation problem by:

1. Promoting the idea that everyone could be a curator [54] and initiating the ingest process,
2. Developing the USW protocol that addresses the administration and preservation planning management functions,
3. Encouraging “crowd-sourced” preservation by executing data management and archival storage functions based on users interacting with the USW web objects (WOs), and
4. Providing access to the preserved WOs using common web infrastructure (WI) search and retrieval technologies.

The USW algorithm creates a collection of WOs that:

1. Grows based on users deciding which WO should be preserved,
2. Automatically expands across new hosts based on their receptiveness to new WOs, and
3. Maintains WOs when faced with losses,

We have developed the need for and the theory behind the unsupervised small-world (USW) algorithm. We have developed a simulation that was used to test and refine various aspects of the USW algorithm. We have developed a small scale reference implementation of the USW algorithm using a variety of pages from different domains. We have developed a framework for crowd-sourced preservation.

The OAIS Informational Model (Figure 108 on the following page) makes explicit that OAIS can apply to physical and digital objects and that the object can be enhanced by the addition of outside knowledge via the Knowledge Base to the Information Object. Application of such outside information to the metadata on Josie’s picture (Figure 1 on page 3) enhances the image (Table 50 on page 284).

We started with and answered the following questions:

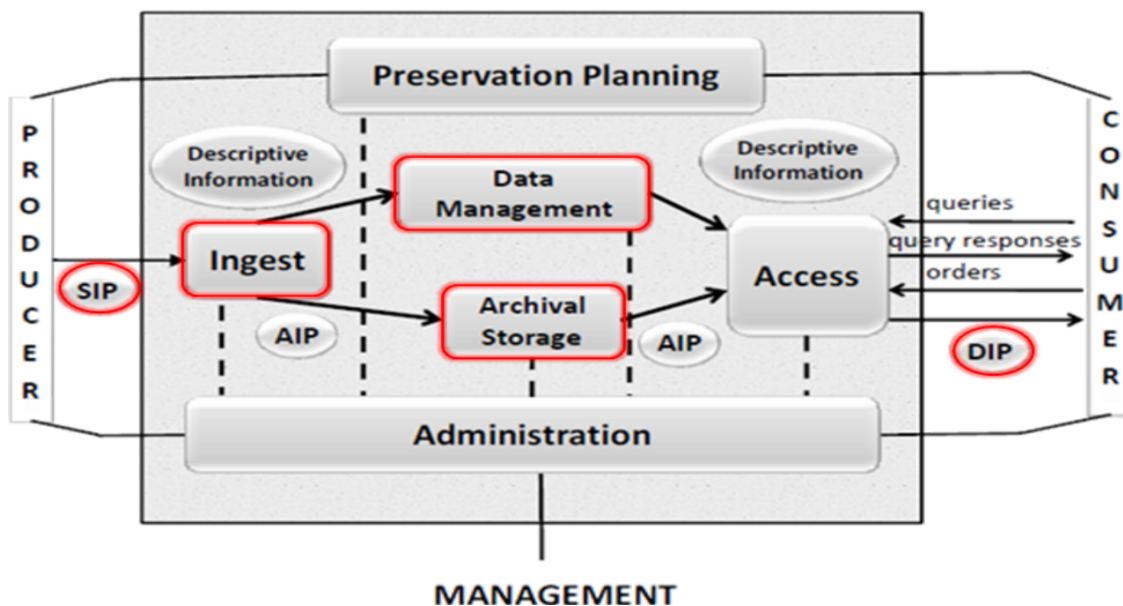


Figure 107. OAIS Reference Model functional entities. A Submission Information Package (SIP) is a package of information about the digital artifact that a producer would submit with the artifact. A Dissemination Information Package (DIP) is disseminated with the artifact to the consumer. Archive Information Packages (AIP) are not part of the USW algorithm. In the USW algorithm, the SIP is represented by metadata (including ORE REMs) created when the digital object becomes a member of the USW graph. The WO's DIP will contain whatever data is associated with the WO's retrieval by the standard WI retrieval mechanisms. The USW algorithm directly supports the Ingest, Data Management, and the Archival Storage functions of the reference model. Image taken from [62].

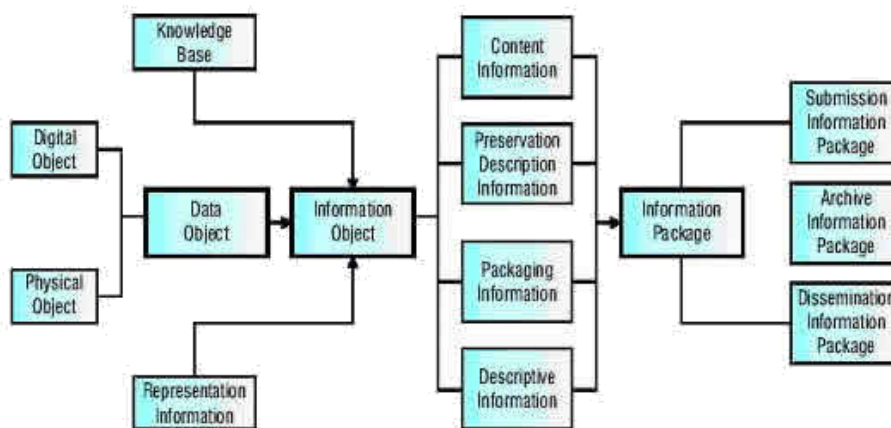


Figure 2

Figure 108. OAIS Information Model. Image taken from [194].

Table 50. Analysis of Josie’s metadata. Adding external knowledge via the knowledge base aspect of the OAIS Information Model enhances and enriches an object.

Metadata	Additional Information
Josie McClure	Her maiden name.
Feb 30,1907	In the late 1800s and early 1900 there were places where the calendar had more than 28 days in February. By today’s calendar, the date would be 2 March.
Poteau, I.T.	Poteau currently had a population of about 8520 in 2010. I.T. is an abbreviation for Indian Territories, the area that was to become Oklahoma.
Fifteen years of age ... weighed 140 lbs.	She was probably well fed and possibly overweight.

1. *Can web objects (WOs) be constructed to outlive the people and institutions that created them?*

We have developed, analyzed, tested through simulations, and developed a reference implementation of the unsupervised small-world (USW) algorithm that we believe will create a connected network of WOs based on the web infrastructure (WI) that will outlive the people and institutions that created the WOs. The USW graph will outlive its creators by being *robust* and continuing to operate when some of its WOs are lost, and it is *resilient* and will recover when some of its WOs are lost.

2. *Can we leverage aspects of naturally occurring networks and group behavior for preservation?*

We used Reynolds’ tenets for “boids” to guide our analysis and development of the USW algorithm. The USW algorithm allows a WO to “explore” a portion of the USW graph before making connections to members of the graph and before making preservation copies across the “discovered” graph. Analysis and simulation show that the USW graph has an average path length ($L(G)$) and clustering coefficient ($C(G)$) values comparable to small-world graphs. A high $C(G)$ is important because it reflects how likely it is that a WO will be able spread copies to other domains, thereby increasing its likelihood of long

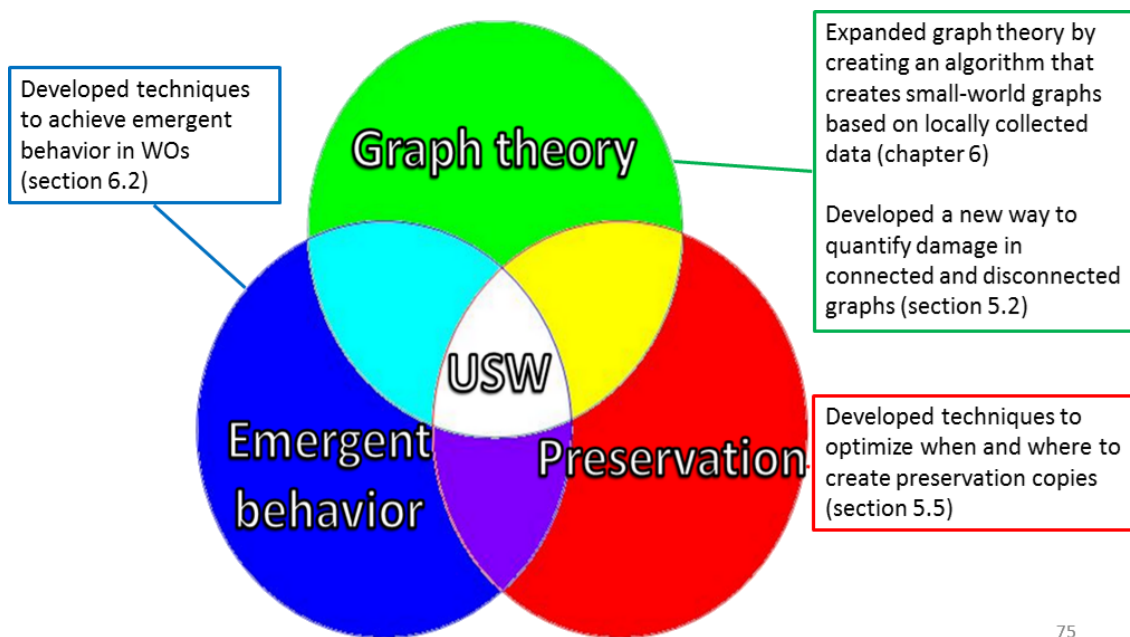
term survival. A short $L(G)$ is important because it means that a WO will not have to look too far to identify new candidate preservation domains, if needed. Small-world graphs occur in nature and are thus believed to be *robust* and *resilient*. The USW algorithms use these small-world graph characteristics to spread preservation copies across as many hosts as needed and possible.

9.2 CONTRIBUTIONS

We have made the following contributions to the field of Computer Science:

1. Expanded graph theory
 - (a) By creating an algorithm that creates small-world graphs based on locally collected data. Small-world graphs appear naturally in nature and in organic man made structures and relationships. Small-world graphs are both *robust* and *resilient*, and
 - (b) Developed a new way to quantify damage in connected and disconnected graphs.
2. Expanded digital preservation
 - (a) Developed a unified theory to optimize when and where to create preservation copies, and
 - (b) Analyzed the communication costs associated with preservation policies that different in their aggressiveness.
3. Developed techniques to apply the ideas of emergent behavior to digital preservation of web objects.

We have successfully combined disparate ideas from graph theory, digital preservation, and computer based emergent behavior to create a system for the preservation of web objects (Figure 109 on the next page).



75

Figure 109. USW contributions to graph theory, emergent behavior, and preservation.

APPENDIX A

ALGORITHMS

A collection of the algorithms that are at the heart of creating a USW graph. A complete list of all algorithms is given in APPENDIX L on page 546.

A.1 MAIN ALGORITHMS AND FUNCTIONS

The following algorithms, procedures, and functions are critical to understanding and implementing the USW algorithm in the simulator and the reference implementation.

A.1.1 USW CONCEPTUAL MODEL

The USW conceptual model one where a WO is created, added to an existing USW graph, makes some number of preservation copies, and waits to be accessed by some entity (see Algorithm 1 on the next page).

A WO is created by “wrapping” a digital object with metadata that has USW control parameters, communications mechanisms, and links to those parts of the DO that the entity creating the WO considers important (see Algorithm 3 on page 290). After the WO is created, the creator identifies an already existing USW WO that the new WO will use to start its exploration of the existing graph. The WO will explore the USW graph until it makes a connections to some of the WOs it has explored and discovered. Based on these connections, the wandering WO will make some number of preservation copies.

At some later time, any WO that is accessed from the USW graph will process any messages using the communication mechanisms that it was imbued with when it was created. These messages may update and change the WO’s internal data structure. A key distinction is whether the accessed WO is the responsible for the active or passive maintenance of its self and its family members.

After the WO performs its maintenance tasks, it updates the graph by returning itself (and its new internal data structures) to the original graph.

Algorithm 1 USW model.

```

1: procedure MODEL
2:    $wo \leftarrow Creation(do)$ 
3:    $data \leftarrow Wandering(wo, oldWO)$   $\triangleright$  The oldWO is given by the WO creator.
4:    $graph \leftarrow Connecting(wo, data)$ 
5:   while true do
6:      $wo \leftarrow WOSelectionProcess(graph)$ 
7:      $wo \leftarrow MessageMaintenance(wo)$   $\triangleright$  The wo is updated with applicable
       messages.
8:     if  $IsActiveMaintainer(wo) == \mathbf{true}$  then
9:        $wo \leftarrow ActiveMaintenance(wo)$ 
10:    else
11:       $wo \leftarrow PassiveMaintenance(wo)$ 
12:    end if  $\triangleright$  The USW graph is updated by the inclusion of the updated
       WO.
13:     $graph \leftarrow graph \cup wo$ 
14:  end while
15: end procedure

```

A.1.2 ACTIVE MAINTENANCE ALGORITHM

Each USW family has exactly one WO who is designated as the WO to affect active maintenance of the family. The primary purpose of active maintenance is to ensure that there are an adequate number of preservation copies available. The active maintainer examines its internal data structures to see how many copies have been created and where they are located (see Algorithm 2). If these copies are “bad” meaning they are unreachable, or corrupted, or invalid in some way, then a list of the bad copies is processed.

Each family should have internal data structures that track the active maintainer’s structures. If the active maintainer can not access a copy then the location of that copy must be removed from the all internal data structures for all family members. The next time a family member is accessed, it will receive and process the message and update its internal structure.

The active maintainer is responsible for creating additional copies as needed. If there are additional copies to be made, then the active maintainer creates the copies (see Algorithm 5 on page 292).

Algorithm 2 ActiveMaintenance() function. Returns an updated WO.

```

1: function ACTIVEMAINTENANCE(wo)
2:   currentBadCopies  $\leftarrow$  copies that are invalid
3:   while currentBadCopies  $\neq$  0 do  $\triangleright$  Remove copy from list of copies and all
      family members.
4:     badCopy  $\leftarrow$  PopFront(currentBadCopies)
5:     SendMessage(wo.family, “removecopy”, badCopy)
6:     currentBadCopies  $\leftarrow$  RemoveFront(currentBadCopies)
7:   end while
8:   neededCopies  $\leftarrow$  CopyNumberNeeded(wo)
9:   if neededCopies > 0 then
10:    Copy(wo)
11:  end if
      return wo
12: end function

```

A.1.3 CREATION FUNCTION

A web object (WO) is a digital object (DO) that has been augmented with USW specific data and metadata.

Included in the augmented data are:

1. *Limits on the number of preservation copies to create*: this includes both minimum and maximum limits
2. *Communication mechanisms*: the WO must be able to send and receive messages. Logically these messages can be:
 - Point-to-point (WO to WO messages), or
 - To family members only (multicast messages), or
 - To all WOs (broadcast messages).
3. *Specific USW control parameters*: including values for all policies, β , γ , name of the progenitor, and any other data required for implementation.

Each WO must have a globally unique identifier, a URI.

Algorithm 3 Creation() function. Returns a WO starting with a DO. Control and other data wrap the DO to create a WO.

```

1: function CREATION(do)
2:   wo  $\leftarrow$  do + controlParameters
3:   wo  $\leftarrow$  wo + name            $\triangleright$  Name assigned from a global naming function
   return wo
4: end function

```

A.1.4 CONNECTING FUNCTION

After a new WO explores and discovers as much of the USW graph as permitted, it must make friendship connections between itself and other selected WOs. At a minimum, a connection will be made between the new WO and the last WO it explored (see Algorithm 4).

While exploring the graph, the new WO maintained two sets of WOs. One, the *toBeVisitedSet* is a set of WOs that have been discovered, but have yet to be explored. The other, the *visitedSet* is a set of WOs that have been explored. These sets are completely disjoint. The function *SampleSize()* treats these sets as lists.

After making a connection to the last WO explored, the WO will make some number of connections to members of the *toBeVisitedSet* and *visitedSet* lists. Connections are bi-directional, so a connection between WO A and WO B requires that the internal data structures for both WOs be updated. The number of connections is dependent on the control parameters in the new WO.

Algorithm 4 Connecting() function. Returns a graph whose edges have been increased by the number of edges (connections) made by the no longer wandering WO.

```

1: function CONNECTING(newWO, data)
2:   newWO and data.oldWO connect to each other
3:    $\gamma'' \leftarrow \text{SampleSize}(wo, \text{newWO}.\text{"tobevisited"list})$ 
4:   while newWO has  $\gamma''$  names in its "to be visited" list do
5:     newWO selects node  $\rho$  from the list
6:     newWO and  $\rho$  connect to each other
7:   end while
8:    $\gamma' \leftarrow \text{SampleSize}(wo, \text{newWO}.\text{"visited"list})$ 
9:   while newWO has  $\gamma'$  names in its "visited" list do
10:    newWO selects node  $\rho$  from the list
11:    newWO and  $\rho$  connect to each other
12:  end while
13:  graph  $\leftarrow \text{graph} \cup \text{newWO}$ 
14:  return graph
14: end function

```

A.1.5 COPY ALGORITHM

A copy of a WO is a WO that is logically the same as the original WO, it is not a bit-by-bit copy of the original WO [142]. The intent of a copy is to increase the likelihood of the long-term preservation of the original WO. To increase this likelihood, copies are spread across to as many unique locations as possible.

The WO is imbued at creation with a copy creation policy, and minimum and maximum number of desired preservation copies. During the WO's lifetime, it will make copies, it will keep a record of their location, and will identify copy candidate locations based on the WO's connections to other WOs.

The WO determines if additional copies need to be created *CopyNumberNeeded()*, and if so identifies copy candidate locations based on where copies currently exist and where they could exist. If a set of candidate location exists, then a request is sent to one member of the set.

Algorithm 5 Copy() function. How to select where to make a copy. A copy will be created on a host that does not already have a copy on it.

```

1: procedure COPY(wo)
2:   copiesToCreate  $\leftarrow$  CopyNumberNeeded(wo)
3:   if copiesToCreate > 0 then
4:     currentCopyLocations  $\leftarrow$  wo.currentCopyLocations
5:     candidateCopyLocations  $\leftarrow$  wo.candidateCopyLocations
6:     possibleLocations  $\leftarrow$  candidateCopyLocations \ currentCopyLocations
7:     while copiesToCreate > 0 do
8:       if | possibleLocations | > 0 then ▷ Create a copy at the
          newCopyLocation
9:         newCopyLocation  $\leftarrow$  PopFront(possibleLocations)
10:        SendMessage(newCopyLocation, "copyrequest", wo)
11:       else ▷ Reaching this break can be used as an indication to begin
          "wandering" the graph again
12:         break
13:       end if
14:       copiesToCreate  $\leftarrow$  copiesToCreate - 1
15:     end while
16:   end if
17: end procedure

```

A.1.6 COPYNUMBERNEEDED FUNCTION

A WO will create some number of copies every chance it gets. The maximum number of copies it attempt to create is dependent on how many copies it has currently, the minimum and maximum number of copies it has been imbued to create, and its preservation policy (see Algorithm 6). The *CopyNumber()* implements Policy E logic.

Algorithm 6 *CopyNumberNeeded()* function. Returns the number of copies to make. The number of copies to make is determined by the copy creation policy and the current number of copies.

```

1: function COPYNUMBERNEEDED(wo)
2:   copyPolicy ← wo.copyPolicy
3:   minCopies ← wo.minCopies
4:   maxCopies ← wo.maxCopies
5:   currentNumber ← wo.currentNumberOfCopies
6:   copyNumber ← 0
7:   if currentNumber < maxNumber then
8:     if copyPolicy == “polite” then
9:       copyNumber ← 1
10:    else if copyPolicy == “moderately aggressive” then
11:      if currentNumber < minNumber then
12:        copyNumber ← minNumber – currentNumber
13:      else
14:        copyNumber ← 1
15:      end if
16:    else if copyPolicy == “aggressive” then
17:      copyNumber ← maxNumber – currentNumber
18:    end if
19:  end if
20:  return copyNumber
21: end function

```

A.1.7 ISACTIVEMAINTAINER FUNCTION

Each family has exactly one WO responsible for the active maintenance of the family. When the family was created, it was made of copies of the original WO known as the progenitor (see Algorithm 7). The progenitor WO will always be the active maintainer and will assume these responsibilities if another WO has designated itself as the active maintainer.

Algorithm 7 IsActiveMaintainer() function. Returns TRUE or FALSE if the current WO is the active maintainer. As a side effect, a new parent might be declared.

```

1: function ISACTIVEMAINTAINER(wo)
2:   returnValue ← false
3:   if wo.name == wo.progenitor then
4:     returnValue ← true
5:     if wo.name ≠ wo.parent then
6:       SendMessage(wo.family, "declareparent", wo.name)
7:     end if
8:   else if wo.name == wo.parent then
9:     returnValue ← true
10:  else if wo.parent is not accessible then
11:    SendMessage(wo.family, "declareparent", wo.name)
12:    returnValue ← true
13:  end if
14:  return returnValue
14: end function

```

A.1.8 MESSAGEMAINTENANCE FUNCTION

WOs communicate with each other via messages. Messages are used because current technology does not permit direct WO-to-WO interaction. WOs are imbued with various ways to send and receive messages (see Algorithm 3 on page 290). When a WO is activated, it queues all the messages that it meant for it based its communication mechanisms (see Algorithm 8). These messages are sorted into time order and processed sequentially. It is assumed that the communication mechanism keeps track of the messages the WO has received and will not send the same message twice. If the communication mechanism does not provide this service then the WO ensure that it only processes “new” messages.

Algorithm 8 MessageMaintenance() function. USW WO message maintenance.

```

1: function MESSAGEMAINTENANCE(wo)
2:   messages ← messages from communication wo.mechanism(s)
3:   messages ← SortByTimeSent(messages)
4:   while messages ≠ 0 do
5:     message ← PopFront(messages)      ▷ Messages can be received from
      family members, friends, and outsiders.  ▷ Process each message in
      the order sent.
6:   end while
      return wo
7: end function

```

A.1.9 PASSIVEMAINTENANCE ALGORITHM

Each family has exactly one active maintenance WO, all other family WOs engage in passive maintenance. A passive maintainer checks the state of family members and reports any problems to the active maintainer for action.

Algorithm 9 PassiveMaintenance() function. Returns an updated WO.

```

1: function PASSIVEMAINTENANCE(wo)
2:   currentBadCopies  $\leftarrow$  copies that are invalid
3:   while currentBadCopies  $\neq$  0 do       $\triangleright$  Notify the active maintenance WO
       about invalid copy.
4:     badCopy  $\leftarrow$  PopFront(currentBadCopies)
5:     SendMessage(wo.parent, "badcopy", badCopy.name)
6:     currentBadCopies  $\leftarrow$  RemoveFront(currentBadCopies)
7:   end while
       return wo
8: end function

```

A.1.10 SAMPLESIZE FUNCTION

The WO builds up internal data structures as it explores and discovers information about the USW graph. The WO uses this information in various ways when determining which friendship connections to make, where to make copies, etc. As the USW graph grows, these data structures could grow to considerable size. The *SampleSize()* function returns the number of entries to return from this universe of data based on criteria imbued at the WO's creation (see Algorithm 10).

Algorithm 10 *SampleSize()* function. Return the number of WOs to select from the WO list. There are many different ways to select ho many WOs to pick from a list.

```

1: function SAMPLESIZE(wo, listOfWOs)
2:   selector  $\leftarrow$  wo.selector
3:   universe  $\leftarrow$  listOfWOs
4:    $\gamma \leftarrow$  wo. $\gamma$ 
5:   if selector == 1 then
6:     sampleSize  $\leftarrow$  length(universe) *  $\gamma$ 
7:   else if selector == 2 then
8:     sampleSize  $\leftarrow$  max(1, ln(length(universe) *  $\gamma$ ))
9:   else if selector == 3 then
10:    sampleSize  $\leftarrow$  max(1, ln(length(universe))) *  $\gamma$ 
11:  else if selector == 4 then
12:    sampleSize  $\leftarrow$  max(0, ln(length(universe) *  $\gamma$ ))
13:  else if selector == 5 then
14:    sampleSize  $\leftarrow$  max(0, ln(length(universe))) *  $\gamma$ 
15:  else if selector == 6 then
16:    sampleSize  $\leftarrow$  max(1, log2(length(universe) *  $\gamma$ ))
17:  else if selector == 7 then
18:    sampleSize  $\leftarrow$  max(1, log2(length(universe))) *  $\gamma$ 
19:  else if selector == 8 then
20:    sampleSize  $\leftarrow$  max(0, log2(length(universe) *  $\gamma$ ))
21:  else if selector == 9 then
22:    sampleSize  $\leftarrow$  max(0, log2(length(universe))) *  $\gamma$ 
23:  else if selector == 10 then
24:    sampleSize  $\leftarrow$  5 + log2(length(universe) *  $\gamma$ )
25:  end if
      return sampleSize
26: end function

```

A.1.11 WANDERING ALGORITHM (HIGH LEVEL)

A new WO “wanders” through the USW graph gathering information about the graph’s structure as it explores each already existing WO it encounters. The new WO will wander until it has explored the entire graph, or a locally generated random number exceeds a threshold β that the WO was imbued with when it was created (see Algorithm 11).

The new WO explores an existing old WO and adds the old WO to a “visited” list. When exploring the old WO, the new WO discovers connections to other WOs. Connections to unexplored and already undiscovered WOs are added to the “to be visited” set. If there are no more WOs to be explored or a locally generated random number exceeds the new WOs β value then a new old WO is selected and the process continues. Otherwise, the function returns the last WO that was explored and the two lists.

Algorithm 11 Wandering() function. Returns a collection of data about the USW graph that the wandering WO discovered. Depending on the size of the graph and how long the WO wanders, the data returned may or may not reflect the total state of the graph.

```

1: function WANDERING(newWO, oldWO)
2:   connected = false
3:   while connected == false do
4:     newWO explores the oldWO
5:     newWO adds oldWO to “visited” list
6:     newWO adds oldWO’s list of “connected to WOs” to “to be visited” set
7:     newWO gets a value  $\zeta$  from a random number generator
8:     if there are names in “to be visited” list and  $\zeta < newWO.\beta$  then
9:       A different oldWO is selected from the “to be visited” list
10:    else
11:      connected = true
12:    end if
13:  end while
14:  return oldWO, “visited” list, “to be visited” list
15: end function

```

A.1.12 WANDERING ALGORITHM (LOW LEVEL)

A high level description of the Wandering algorithm was given in A.1.11 on the preceding page. A more detailed description is given in Algorithm 12 on the next page.

Algorithm 12 Wandering() detailed view. A WO is in the *wandering* state until it makes its first friendship link. After that it is in the *connected* state. The WO initiates communications with *non-wandering* or *established* WOs.

```

1:  $\beta \leftarrow newWO.beta$   $\triangleright$  The connection threshold.
2:  $\gamma \leftarrow newWO.gamma$   $\triangleright$  The amount of queues that will be used after the first
   connection.
3:  $growthSelector \leftarrow newWO.controlParameter$   $\triangleright$  The growth function selector.
4:  $odering \leftarrow newWO.controlParameter$   $\triangleright$  How to select WOs from a list.
5:  $oldWO$  =given by the WO creator  $\triangleright$  An initial WO that we (the creators of
   this WO) define.
    $\triangleright$  If this WO does not have any friends.
6: if  $FriendList == \emptyset$  then
    $\triangleright$  We initialize the queues of where we will go and where we have been.
7:    $toBeVisitedSet \leftarrow oldWO$ 
8:    $visitedSet \leftarrow \emptyset$   $\triangleright$  While we have WOs that we have not visited, we will
   work.
9:   while  $toBeVisitedSet \neq \emptyset$  do  $\triangleright$  Get the next one from the front of the
   queue.
10:     $potentialFriend \leftarrow PopFront(toBeVisitedSet)$   $\triangleright$  Make sure we talk
    to each WO exactly once.
11:     $visitedSet \leftarrow visitedSet \cup potentialFriend$   $\triangleright$  Get the potential friend's
    list of friends.
12:     $posFriends \leftarrow potentialFriend.friendList$   $\triangleright$  Identify WOs that are
    not in the  $visitedSet$  and not in the  $toBeVisitedSet$ .
13:     $additionalVisits \leftarrow posFriends \setminus visitedSet \setminus toBeVisitedSet$   $\triangleright$ 
    Append the new found WOs to our list of WOs to visit.
14:     $toBeVisitedSet \leftarrow toBeVisitedSet + additionalVisits$   $\triangleright$  If a random
    number exceeds our acceptance threshold or we have processed the
    last potential WO then we are ready to make a connection
15:     $\zeta \leftarrow$  from a random number generator
16:    if  $((\zeta > \beta)$  or  $(toBeVisitedSet == \emptyset))$  then  $\triangleright$  Make the list contain
    the WO that we last "talked to".
17:       $friendList \leftarrow potentialFriend$   $\triangleright$  Add to the list, some of the ones
      that the WO planned to and did talk to.
18:       $temp \leftarrow sampleSize(newWO, toBeVisitedSet)$ 
19:       $friendList \leftarrow friendList \cup toBeVisitedSet[temp]$ 
20:       $temp \leftarrow sampleSize(newWO, visitedSet)$ 
21:       $friendList \leftarrow friendList \cup visitedSet[temp]$ 
22:    else  $\triangleright$  Add the WO that we just talked to the to list of WOs that we
    have visited.
23:       $visitedSet \leftarrow visitedSet + potentialFriend$ 
24:    end if
25:  end while
26: end if
   return  $potentialFriend, visitedSet, toBeVisitedSet$ 

```

A.1.13 WOSELECTIONPROCESS FUNCTION

The USW algorithm relies on individual USW WOs to be accessed. Accessing a WO causes it to begin a maintenance process and injects “energy” into the USW system. An entity outside the USW system determines which WO to access (see Algorithm 13). The external entity could be human or robotic, and the WO selection technique is at the discretion of the entity.

Algorithm 13 WOSelectionProcess() Returns a single WO from the USW graph.

1: **function** WOSELECTIONPROCESS(*graph*) ▷ There are numerous ways to select a WO from the USW graph.

 Random selection would mimic a human browsing the USW graph as identified by some external source such as search engine.

 Ordered selection could be based on the time of WO creation, on the number of connections that already exist within the WO, the number of preservation copies that already exist, an estimate of the visitation rate for a WO, etc.

return wo

2: **end function**

A.2 SUPPORTING ALGORITHMS AND FUNCTIONS

Misc. things that are supporting and not really main stream enough to be raised to a higher level.

A.2.1 EVALUATING THE ROBUSTNESS AND RESILIENCY OF GRAPH

Algorithm 14 is an approach that returns whether or not a graph is (1) resilient to attack, and (2) is robust. This algorithm is used to measure the effectiveness of an attack profile and USW graph resiliency and robustness.

Algorithm 14 Evaluating the robustness and resiliency of graph.

```

1:  $g \leftarrow$  a graph
2:  $l \leftarrow$  maximum number of turns in the game
3:  $attackerPercent \leftarrow$  constant percentage
4:  $resiliencePercent \leftarrow$  constant percentage
5:  $gOrig \leftarrow g$ 
6: while  $l > 0$  do
7:   Report graph metrics on graph  $g$ 
8:    $gPrime \leftarrow removeHighestBetween(g, attackerPercent)$ 
9:   if  $disconnected(gPrime) == \mathbf{true}$  then
10:     Break From Loop
11:   end if
12:   Report graph metrics on graph  $gPrime$ 
13:    $g \leftarrow reconstitute(gOrig, gPrime, resiliencePercent)$ 
14:    $l \leftarrow l - 1$ 
15: end while
16: if  $disconnected(gPrime) == \mathbf{true}$  then
17:   Declare attacker the winner.
18: else
19:   Declare USW robust and resilient.
20: end if

```

A.2.2 REMOVEHIGHESTBETWEEN FUNCTION

Each WO in the USW graph was evaluated to identify the one with the highest centrality $c_B(v)$ (see Algorithm 15). This WO is then removed from the graph and the modified graph is returned.

Algorithm 15 removeHighestBetween() function. Removing the WO with the highest “betweenness” score.

```

1: function REMOVEHIGHESTBETWEEN(graph)      ▷ The WO with the highest
    betweenness value is identified.
2:    $WOId \leftarrow f(graph)$                 ▷ The WO is removed from the original graph.
3:    $gReduced \leftarrow graph - WOId$ 
    return gReduced
4: end function

```

A.2.3 RECONSTITUTE FUNCTION

Each WO in the attacked USW graph is assigned a random number as a local label (see Algorithm 16). All WOs are numerically sorted based on this numeric label. A fixed percentage of WOs in the attacked graph are replaced by the same WOs from the original graph.

Algorithm 16 *reconstitute()* function.

```

1: function RECONSTITUTE(gOrig, gAttacked, percentage)
2:   qNodes  $\leftarrow$  each node in the attacked graph is assigned a random number
      and sorted ascending by that number
3:   wNodes  $\leftarrow$  perCentage * length(qNodes)
4:   for node in wNodes do
5:     gAttacked[node]  $\leftarrow$  gOrig[node]
6:   end for
      return gAttacked
7: end function

```

A.2.4 ORDEREDQUEUE ALGORITHM

We may required a number of graph metrics for the USW graph depending on the particular circumstances. *OrderedQueue()* provides a single place where those metrics can computed and ordered as needed.

Algorithm 17 *OrderedQueue()* function. Returns an ordered queue of WOs from a USW graph. The queue can be ordered in different manners based on which graph characteristic is of interest. A few options are: path length from an source WO to all WOs, the degreeeness of each of the WOs or the age of the WO.

```

1: function ORDEREDQUEUE(g, sourceWO, queueType)
2:   returnValue  $\leftarrow$  null
3:   if queueType == "Path length longest" then
4:     pathLengths  $\leftarrow$  ComputePathLength(g, sourceWO)
5:     returnValue  $\leftarrow$  Sort(pathLengths, "descending")
6:   else if queueType == "Path length shortest" then
7:     pathLengths  $\leftarrow$  ComputePathLength(g, sourceWO)
8:     returnValue  $\leftarrow$  Sort(pathLengths, "ascending")
9:   else if queueType == "Degreeeness ascending" then
10:    degrees  $\leftarrow$  ComputeDegrees(g)
11:    returnValue  $\leftarrow$  Sort(degrees, "ascending")
12:  else if queueType == "Degreeeness descending" then
13:    degrees  $\leftarrow$  ComputeDegrees(g)
14:    returnValue  $\leftarrow$  Sort(degrees, "descending")
15:  else
16:    attribute  $\leftarrow$  SomeFuction(g)
17:    returnValue  $\leftarrow$  OrderingFunction(attribute)
18:  end if
19:  return returnValue
20: end function

```

A.2.5 VALUE ALGORITHM

$Value()$ provides a way to compare the normalized $C(G)_{Average}$ and $L(G)$ for a USW graph. The value returned by the function should remain positive, meaning that the normalized $C(G)_{Average}$ dominates. If the value returns a negative number, then there is significant differences in the $L(G)$ and further investigation into why is recommended.

Algorithm 18 $Value()$ function. Returns a single normalized value. The value is the difference between the clustering coefficient and the average path length. The functions CC and PL take a graph as input and return $C(G)_{Average}$ and $L(G)$ respectively.

```

1: function VALUE( $g, cc_{base}, pl_{base}$ )
2:    $cc \leftarrow ComputeClusteringCoefficient(g)$ 
3:    $pl \leftarrow ComputePathLength(g)$ 
4:    $nc \leftarrow cc/cc_{base}$ 
5:    $np \leftarrow pl/pl_{base}$ 
6:    $returnValue \leftarrow nc - np$ 
       return returnValue
7: end function

```

A.2.6 EVALUATING ALL CONNECTIONS

An exponential averager is used to select identify the set of WOs that have the greatest positive influence on both $C(G)_{Average}$ and $L(G)$.

Base values for $C(G)_{Average}$ and $L(G)$ from the original graph are computed. As well as the number of times the system will be evaluated. All δ_{st} value from the wandering WO to every other WO is computed and then ordered in a queue from longest to shortest. A connection is made from the wandering WO to the furthest WO and $C(G)_{Average}$ and $L(G)$ values for the tentative graph are computed. If the new values are better than the previous values, the process repeats. At the end of the tentative graph computations connection messages have been sent to a all WOs whose friendship connections improve the USW's small-world criteria.

Algorithm 19 EvaluateConnections() function. The newly connected WO evaluates the effect of connecting to each of the γ established WOs. The evaluation is used to select the ones to connect to have the greatest positive impact on $C(G)_{Average}$ and $L(G)$.

```

1: procedure EVALUATECONNECTIONS( $g, \gamma$ )
     $\triangleright$  The number of WOs that will become connected to the no-longer
    wandering wo
2:    $limit \leftarrow \gamma * |V(g)|$ 
3:    $dampening \leftarrow 0.7$ 
4:    $o \leftarrow 0$ 
5:   for  $o \leq limit$  do
6:      $oldDV \leftarrow 0$ 
7:      $cc_{base} \leftarrow ComputeClusterCoefficient(g)$ 
8:      $pl_{base} \leftarrow ComputePathLength(g)$ 
9:      $queueOfWOs \leftarrow orderedQueue(g, pathLengthLongest)$ 
10:     $e \leftarrow 0$ 
11:    for  $e \leq (limit - o)$  do
12:       $candidateWO \leftarrow PopFront(queueOfWOs)$ 
13:       $g' \leftarrow g \cup candidateWO$ 
14:       $newValue \leftarrow Value(g', cc_{base}, pl_{base})$ 
15:       $dampenedValue \leftarrow dampening * oldDV + (1 - dampening) * newValue$ 
16:      if  $dampenedValue > oldDV$  then
17:         $oldDV \leftarrow dampenedValue$ 
18:      else
19:        break
20:      end if
21:       $e \leftarrow e + 1$ 
22:    end for
23:     $sendMessage(candidateWO, "connect to me", WO)$ 
24:     $g \leftarrow g'$ 
25:     $o \leftarrow o + 1$ 
26:  end for
27: end procedure

```

A.3 FUTURE WORK

The following algorithms, procedures, and functions are critical to understanding and implementing the USW algorithm in other environments.

Algorithm 20 Important() function. Return TRUE or FALSE that a message is important. Provides a way to identify things that are important and to remove duplicates. There may be other criteria that define what makes a message *important*. For USW demonstration purposes, this procedure will always return **TRUE**, meaning that the message is important.

```

1: function IMPORTANT(message)      ▷ Determine if this message is important.
2:   returnValue ← FALSE
3:   if important message then
4:     returnValue ← TRUE
5:   end if
   return returnValue
6: end function

```

Algorithm 21 Wiki rd() implementation. Linda rd() equivalent implementation for MediaWiki.

```

1: LMT ← last modified date of this REM
2: MB ← message box wiki page of this REM
3: SMB ← shared multicast message box wiki page
4: CHANGES ← ∅
5: MSGs ← MB revisions after LMT ∪ filtered revisions from SMB
6: for each MSG in MSGs do
7:   if Important(MSG) then
8:     CHANGES ← CHANGES ∪ MSG
9:   end if
10: end for
11: Apply CHANGES to this REM

```

Algorithm 22 Wiki out() implementation. Linda the message box (*TMB*) out() equivalent implementation for MediaWiki.

```

1: if multicast message then
2:   TMB ← shared multicast message box wiki page
3: else
4:   TMB ← message box wiki page of target REM
5: end if
6: MSG ← message tuple
7: overwrite TMB with MSG

```

Algorithm 23 Gmail rd() implementation. Linda rd() equivalent implementation for Gmail.

```

1: MB ← gmail inbox of this REM
2: SMB ← shared multicast gmail inbox (via gateway)
3: MSGs ← MB unread messages ∪ filtered messages from SMB
4: CHANGES ← ∅
5: for each MSG in MSGs do
6:   if Important(MSG) then
7:     CHANGES ← CHANGES ∪ MSG
8:   end if
9: end for
10: apply CHANGES to this REM

```

Algorithm 24 Gmail out() implementation. Linda out() equivalent implementation for Gmail.

```

1: if multicast message then
2:   TMB ← shared multicast gmail inbox (via gateway)
3: else
4:   TMB ← gmail inbox of target REM
5: end if
6: MSG ← message tuple
7: email MSG to TMB

```

APPENDIX B

USW EVENTS

We have categorized and organized WO events and activities into an expected order to support implementation and provide additional clarification. For each event or activity, we have listed the following types of information:

- *Message Name*: message are exchanged between WOs to create friendship links, request that preservation copies be made, and so on. Where a message is exchanged, the message is identified. In those cases where a message is not exchanged (for instance when a `curl` command is used), then appropriate identifying information is provided.
- *Explanation*: a summary of what is happening the USW graph during this activity or event.
- *Example*: example text supporting whatever was identified in the *message name* item.

A complete list of listings is given in APPENDIX M on page 547.

B.1 EVENT 101. GET ENTRYPOINT'S REM.

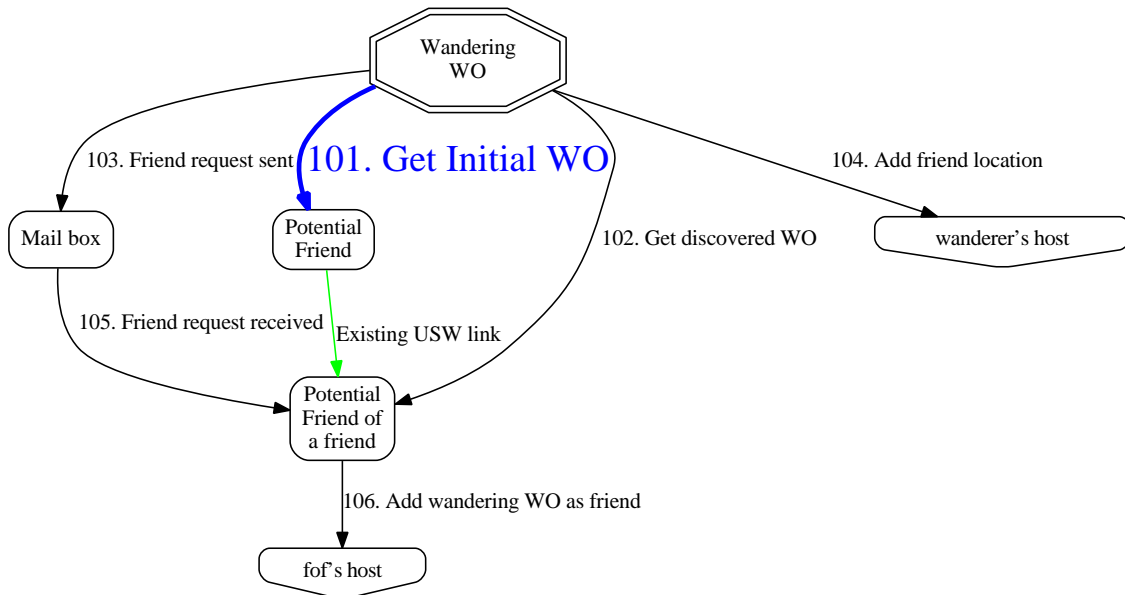


Figure 110. Wandering events and messages, event 101.

- *Message Name:* N/A, simple curl command
- *Explanation:* The “wandering” WO retrieves the entrypoint’s REM to extract USW graph information.
- *Example:*

```
curl -m 120 -i http://flickr.cs.odu.edu/
    flickr-ceotty-8161751828.html
```

B.2 EVENT 102. GET FRIEND OF FRIEND'S REM.

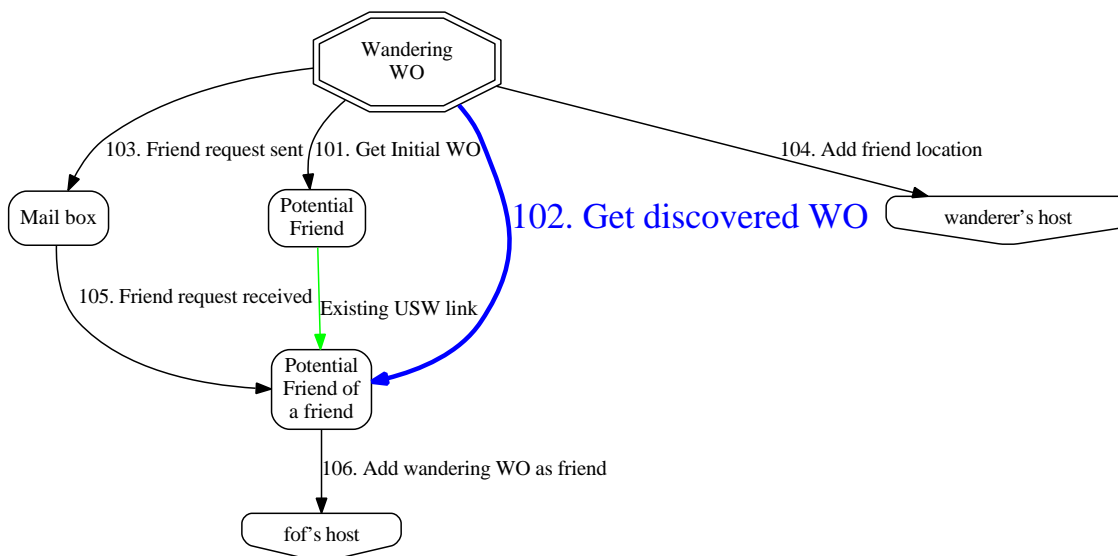


Figure 111. Wandering events and messages, event 102.

- *Message Name:* N/A, simple curl command
- *Explanation:* The “wandering” WO retrieves the discovered WO’s REM to extract USW graph information. The “wandering” WO will continue to explore the USW graph until an end condition is met: 1) the wandering WO’s random number exceeds β , or 2) the wandering WO explores the entire USW graph.
- *Example:*

```
curl -m 120 -i http://arxiv.cs.odu.edu/rems/
arxiv-0912-0201v1.xml
```

B.3 EVENT 103. SEND FRIEND REQUEST.

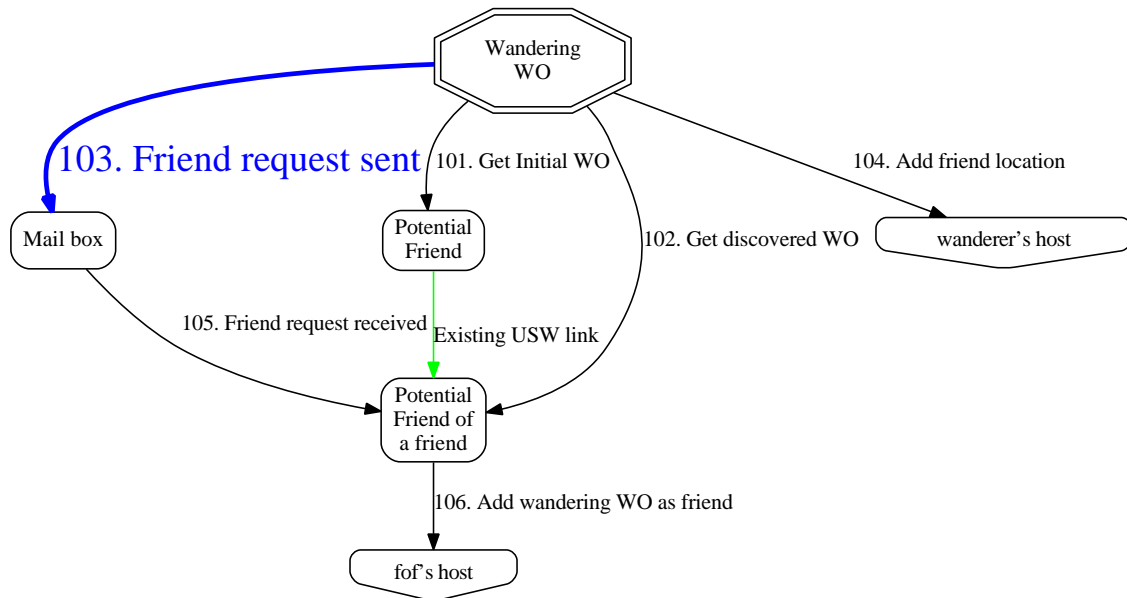


Figure 112. Wandering events and messages, event 103.

- *Message Name:* Friend Request
- *Explanation:* The wandering WO sends a friend request to an established WO.
- *Example:* (lines broken for clarity)

```

curl -m 120 -i -X POST --data-binary @/tmp/temp~.xxx9d0be5d -i
-H "Sender: http://flickr.cs.odu.edu/rems/
                                flickr-ceotty-8161751828.xml"
-H "Content-type: message/http"
http://ws-dl-02.cs.odu.edu:10101/hm/http:
//gutenberg.cs.odu.edu/rems/gutenberg-pride-and-prejudice.xml

```

```

PATCH /rems/gutenberg-pride-and-prejudice.xml HTTP
/1.1
Host: gutenberg.cs.odu.edu
Content-type: application/patch-ops-error+xml
Content-length: 216

```

```
<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
  <link rel="http://wsdl.cs.odu.edu/uswdo/terms/friend
    "
    href="http://flickr.cs.odu.edu/rems/flickr-ceotty
      -8161751828.xml"
    title="Kittens" />
</add>
</diff>
```

Listing 9. Sample Friend Request message. The Friend Request message is an asynchronous communication delivered to the recipient's personal mailbox. The recipient will receive and process the message at some later time. Contents of /tmp/temp~.xxx9d0be5d.

B.4 EVENT 104. WO UPDATES OWN REM.

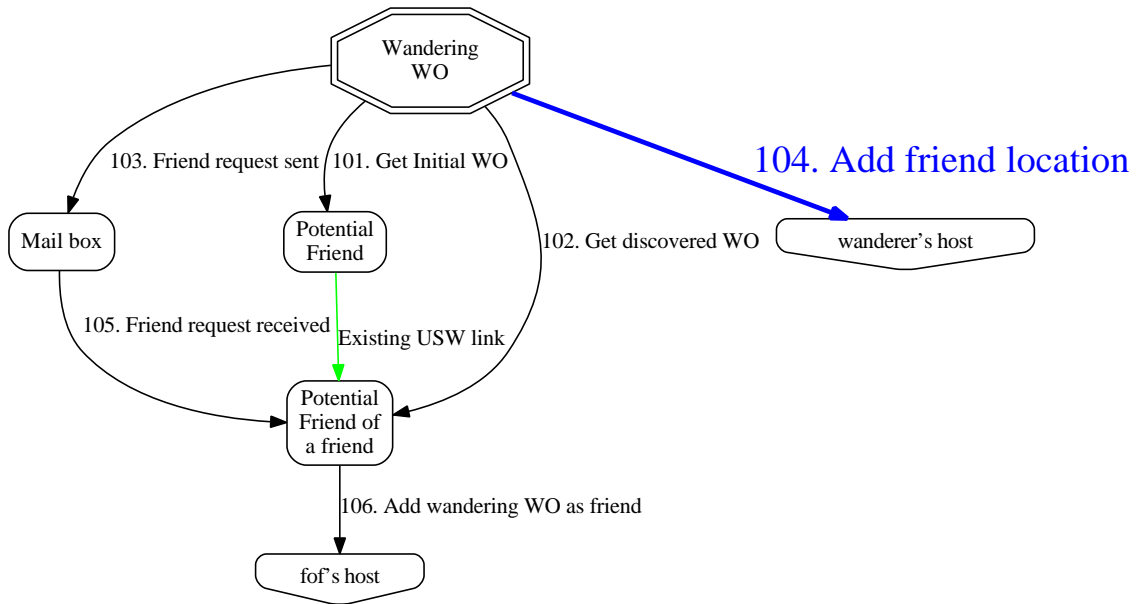


Figure 113. Wandering events and messages, event 104.

- *Message Name:* REM Patch
- *Explanation:* Send HTTP Patch directive to WO's edit service to update WO's REM.
- *Example:* (lines broken for clarity)

```
curl -m 120 -X POST -i --data-binary @/tmp/temp~.xxx23a40ffc
-H "Content-type: application/patch-ops-error+xml"
http://ws-dl-02.cs.odu.edu:10102/rem/edit/http:
//flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/friend"
href="http://gutenberg.cs.odu.edu/rems/gutenberg-
pride-and-prejudice.xml"
title="Pride and Prejudice" />
```

```
</add>  
</diff>
```

Listing 10. Sample Patch, adding a friend location. The patch directive is serviced by the requesting WO's *edit* service. Contents of `/tmp/temp~.xxx23a40ffc`.

B.5 EVENT 105. RETRIEVE FRIEND REQUEST.

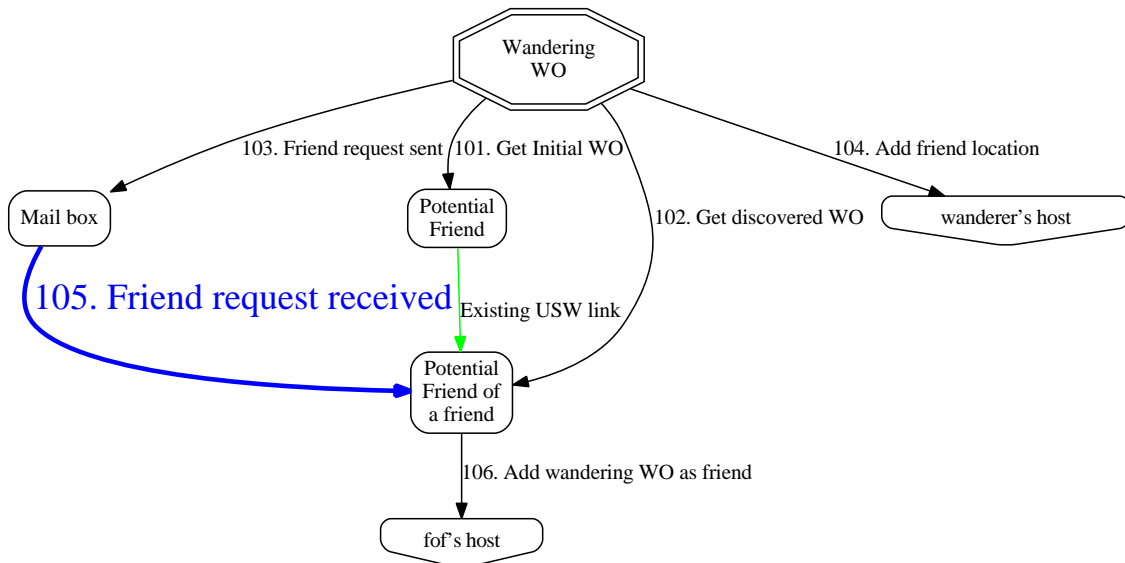


Figure 114. Wandering events and messages, event 105.

- *Message Name:* Retrieve Friend Request
- *Explanation:* The established WO services all of its mailboxes when it is activated. All messages are processed in a first sent - first processed order to ensure that an accurate state is achieved when the last message is processed.
- *Example:* (lines broken for clarity)

```

curl -m 120 -i -o -
  http://ws-dl-02.cs.odu.edu:10101/hm/http:
  //gutenberg.cs.odu.edu/remes/gutenberg-pride-and-prejudice.xml
-----
HTTP/1.1 200 OK\r
Server: HTTP Mailbox\r
Content-type: message/http\r
Date: Fri, 16 Aug 2013 18:17:37 GMT\r
Memento-Datetime: Fri, 16 Aug 2013 18:13:35 GMT\r
Via: sent by 68.10.149.64 on behalf of http://
  flickr.cs.odu.edu/remes/flickr-ceotty-8161751828.xml,
  delivered by http://ws-dl-02.cs.odu.edu:10101/hm/\r

```



```

Link: <http://ws-dl-02.cs.odu.edu:10101/hm/http://
  gutenbergs.cs.odu.edu/remss/gutenberg-pride-and-
  prejudice.xml>;
rel="current", <http://ws-dl-02.cs.odu.edu:10101/hm
  /id/f9db8321-7eae-43db-a241-0ee5ea5ec330>;
rel="self", <http://ws-dl-02.cs.odu.edu:10101/hm
  /id/08e32760-ac47-4194-a186-c3d79af50652>;
rel="first", <http://ws-dl-02.cs.odu.edu:10101/hm
  /id/77cef9fd-d3e1-4737-a59b-142a7936535c>;
rel="last", <http://ws-dl-02.cs.odu.edu:10101/hm
  /id/85088497-3a9a-4fb4-9c67-96e71e60d299>;
rel="next", <http://ws-dl-02.cs.odu.edu:10101/hm
  /id/08e32760-ac47-4194-a186-c3d79af50652>;
rel="previous"\r
Content-Length: 366\r
Connection: keep-alive\r
\r
PATCH /remss/gutenberg-pride-and-prejudice.xml HTTP
  /1.1
Host: gutenbergs.cs.odu.edu
Content-type: application/patch-ops-error+xml
Content-length: 216

<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
  <link rel="http://wsdl.cs.odu.edu/uswdo/terms/friend
    "
    href="http://flickr.cs.odu.edu/remss/flickr-ceotty
      -8161751828.xml"
    title="Kittens"
  />
</add>
</diff>

```

Listing 11. Complete Friend Request message. The receiving WO can use the “Link:” header to get an ordered list of messages from its mailbox.

B.6 EVENT 106. PROCESS FRIEND REQUEST.

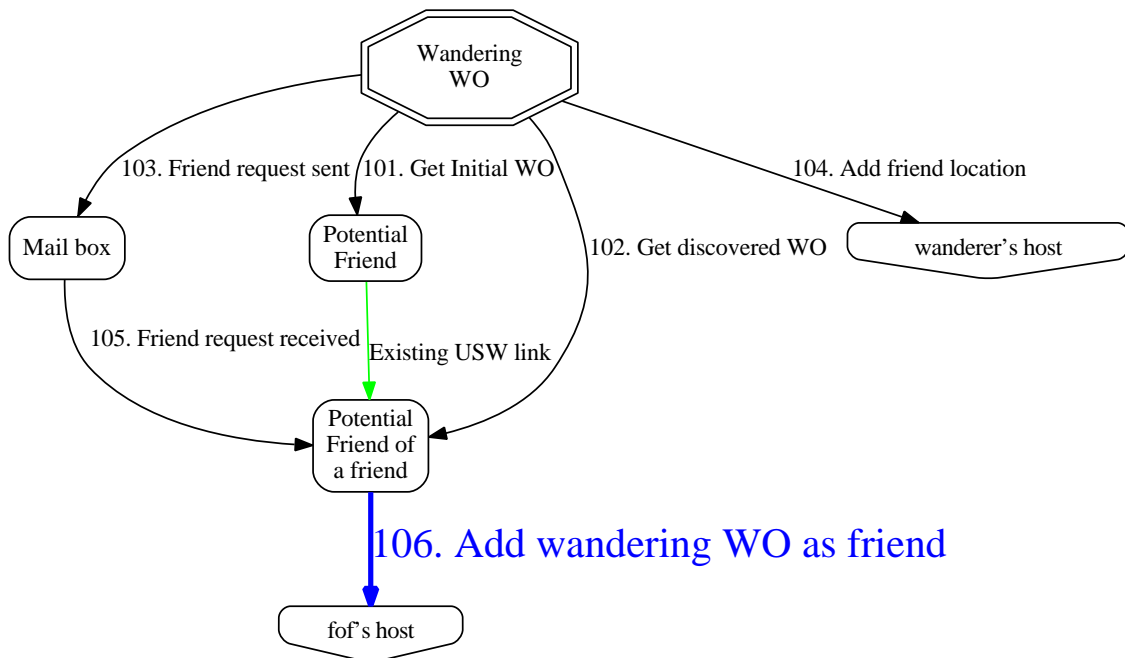


Figure 115. Wandering events and messages, event 106.

- *Message Name:* REM Patch
- *Explanation:* Send HTTP Patch directive to WO's edit service to update WO's REM.
- *Example:* (lines broken for clarity)

```
curl -m 120 -X POST -i --data-binary @/tmp/temp~.xxx9d03a55
-H "Content-type: application/patch-ops-error+xml"
http://ws-dl-02.cs.odu.edu:10102/rem/edit/http://
gutenberg.cs.odu.edu/rems/gutenberg-pride-and-prejudice.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/friend"
href="http://flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml"
```

```
    title="Kittens "  
  />  
</add>  
</diff>
```

Listing 12. Sample Friend Request message processing. The receiving WO uses its *edit* service to process the patch directive. Contents of `/tmp/temp~.xxx9d03a55`.

B.7 USW GRAPH AFTER WANDERING.

USW WO actions can be logically divided into different phases. One such division is pre- and post wandering. After “wandering,” a WO is connected to at least one other WO (Figure 116).

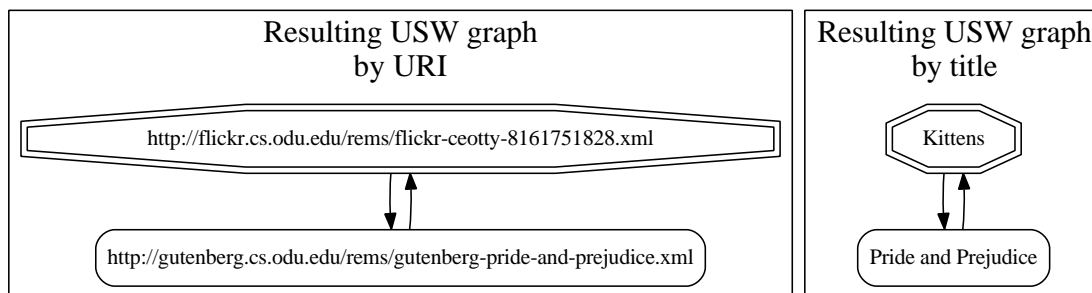


Figure 116. USW graph after wandering events.

B.8 EVENT 201. COPY REQUEST TO FRIEND

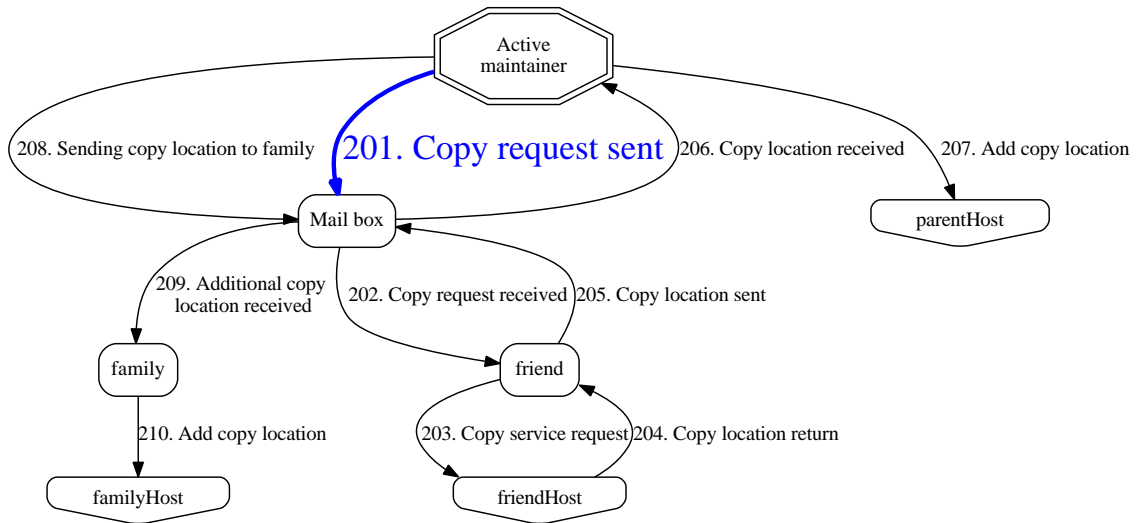


Figure 117. Copying events and messages, event 201.

- *Message Name:* Copy Request to Friend
- *Explanation:* A parental WO is charged with making and distributing copies across as many different domains as it knows about. It sends a Copy Request to a friend and requests that the friend make a copy of using the REM data the parental WO provides.
- *Example:* (lines broken for clarity)

```

curl -m 120 -i
  -H "Sender: http://flickr.cs.odu.edu/rems/
                                     flickr-ceotty-8161751828.xml"
  -H "Content-type: message/http"
  -X POST --data-binary @/tmp/temp~.xxx3bbc2b84
  http://ws-dl-02.cs.odu.edu:10101/hm/http://
  gutenberg.cs.odu.edu/rems/gutenberg-pride-and-prejudice.xml

```

```

POST http://ws-dl-02.cs.odu.edu:10102/rem/copy/http:
  //flickr.cs.odu.edu/ HTTP/1.1
Host: flickr.cs.odu.edu
Content-type: application/atom+xml

```

Content-length: 7681

```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
  xmlns:oreatom="http://www.openarchives.org/ore/atom/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ore="http://www.openarchives.org/ore/terms/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:grddl="http://www.w3.org/2003/g/data-view#"
  xmlns:relationship="http://purl.org/vocab/relationship/"
  xmlns:usw="http://wsdl.cs.odu.edu/uswdo/terms/"
  grddl:transformation="http://www.openarchives.org/ore/atom/atom-grddl.xsl"
  xmlns:le="http://purl.org/atompub/link-extensions/1.0">
  <id>tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty-8161751828</id>
  <link rel="alternate" type="text/html"
    href="http://flickr.cs.odu.edu/flickr-ceotty-8161751828.html" />
  <link rel="self" type="application/atom+xml"
    href="http://flickr.cs.odu.edu/rem/flickr-ceotty-8161751828.xml" />
  <link rel="edit" type="application/atom+xml"
    href="http://ws-dl-02.cs.odu.edu:10102/rem/edit/http://flickr.cs.odu.edu/rem/flickr-ceotty-8161751828.xml" />
```

```
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/copy"
      type="application/atom+xml"
      href="http://ws-dl-02.cs.odu.edu:10102/rem/copy/
http:
  //flickr.cs.odu.edu/" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
synchronize"
      type="message/http" href="http://ws-dl-02.cs.odu.
edu:10102/
  rem/synchronize/http:
  //flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#self"
      href="http://ws-dl-02.cs.odu.edu:10101/hm/http:
  //flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml"
      last-checked="2013-08-16T18:13:36+00:00"/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#all"
      href="http://ws-dl-02.cs.odu.edu:10101/hm/all"
      last-checked="2013-08-16T18:13:42+00:00"/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#family"
      href="http://ws-dl-02.cs.odu.edu:10101/hm/
tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty
-8161751828"
      last-checked="2013-08-16T18:13:39+00:00"/>
<link rel="http://www.openarchives.org/ore/terms/
describes"
      href="http://flickr.cs.odu.edu/rems/
  flickr-ceotty-8161751828.xml#aggregation" />
<source>
  <author>
```



```

    <name>ODU WSDL ReM Generator</name>
    <uri>http://ws-dl-02.cs.odu.edu/</uri>
  </author>
</source>
<published>2013-07-30T18:25:35-04:00</published>
<updated>2013-07-30T18:25:35-04:00</updated>
<link rel="license" type="application/rdf+xml"
  href="http://creativecommons.org/licenses/by-nc/2.5/rdf" />
<rights>This Resource Map is available under the
  Creative Commons Attribution-Noncommercial 2.5
  Generic license</rights>
<title>Kittens</title>
<author>
  <name>ceotty</name>
</author>
<category term="http://www.openarchives.org/ore/terms/Aggregation"
  label="Aggregation" scheme="http://www.openarchives.org/ore/terms/" />
<category term="2012-11-06T00:00:00-05:00"
  scheme="http://www.openarchives.org/ore/atom/created" />
<category term="2012-11-06T00:00:00-05:00"
  scheme="http://www.openarchives.org/ore/atom/modified" />
<category term="3"
  scheme="http://wsdl.cs.odu.edu/uswdo/terms/preservationCopiesMinimumNumber" />
<category term="5"
  scheme="http://wsdl.cs.odu.edu/uswdo/terms/preservationCopiesMaximumNumber" />
<category term="0.85"

```

```
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/beta"
  />
<category term="0.10"
  scheme="http://wsdl.cs.odu.edu/uswdo/terms/gamma"
  />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
family#parent"
  type="application/atom+xml" title="Kittens"
  href="http://flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
family#parentOriginal"
  type="application/atom+xml" title="Kittens"
  href="http://flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
  href="http://flickr.cs.odu.edu/flickr-ceotty
-8161751828.html"
  type="text/html" title="Kittens" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
  href="http://www.flickr.com/photos/ceotty
/8161751828/"
  type="text/html" title="Kittens | Flickr - Photo
Sharing!" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
  href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e.jpg"
  type="image/jpeg" title="[Medium 500] Kittens"
  usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e.jpg"
```

```
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_s.jpg"
    type="image/jpeg" title="[Square 75] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_s.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_q.jpg"
    type="image/jpeg" title="[Square 150] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_q.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_t.jpg"
    type="image/jpeg" title="[Thumbnail] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_t.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_n.jpg"
    type="image/jpeg" title="[Small 240] Kittens"
```

```
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_n.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_m.jpg"
    type="image/jpeg" title="[Small 320] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_m.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_z.jpg"
    type="image/jpeg" title="[Medium 640] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_z.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_c.jpg"
    type="image/jpeg" title="[Medium 800] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_c.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
```

```

    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_1.jpg"
    type="image/jpeg" title="[Large 1024] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_1.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
friend"
    href="http://gutenberg.cs.odu.edu/remes/gutenberg-
pride-and-prejudice.xml"
    title="Pride and Prejudice"/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
copyService"
    href="http://gutenberg.cs.odu.edu/remes/gutenberg-
pride-and-prejudice.xml"
    domain="gutenberg.cs.odu.edu"/>
</entry>

```

Listing 13. Sample Copy Request message.. The message contains the entirety of the WO that is requesting a copy be made on a friend WO's host. Because communication between the sending and the receiving WOs is asynchronous, sending the "value" of the requesting WO rather than a "reference" ensures that a complete WO is available to create a copy. Contents of /tmp/temp~.xxx3bbc2b84.

B.9 EVENT 202. GET COPY REQUEST MESSAGE.

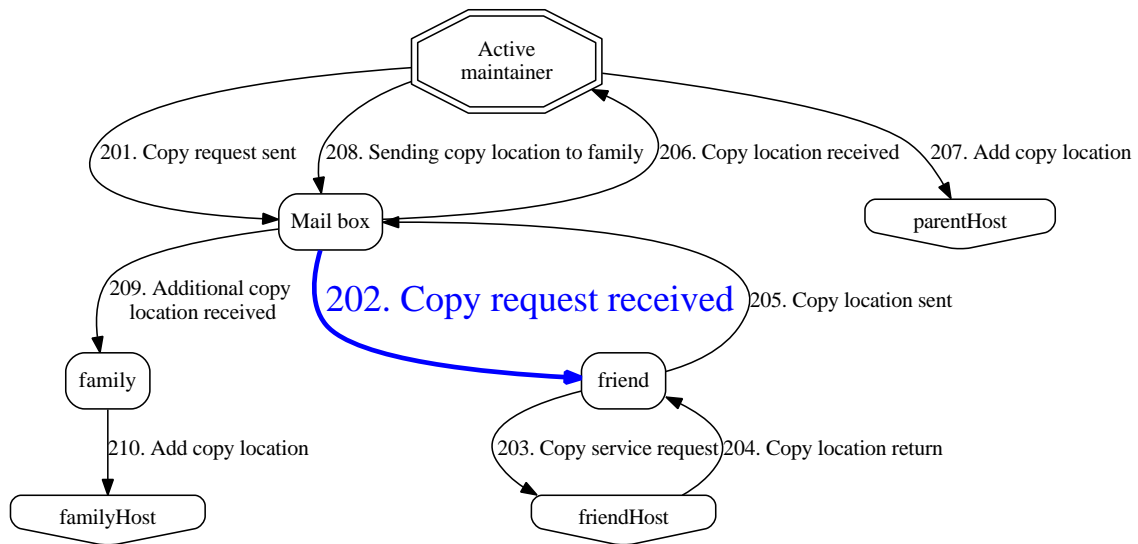


Figure 118. Copying events and messages, event 202.

- *Message Name:* Retrieve Copy Request
- *Explanation:* A parental WO sends a request to an established WO to make a copy of the parent based on the data in the message. The established WO services all of its mailboxes when it is activated. All messages are processed in a first sent - first processed order to ensure that an accurate state is achieved when the last message is processed.
- *Example:* (lines broken for clarity)

```
curl -m 120 -i -o - - http://ws-dl-02.cs.odu.edu:10101/hm/http://gutenberg.cs.odu.edu/remsgutenberg-pride-and-prejudice.xml
```

```
HTTP/1.1 200 OK\r
Server: HTTP Mailbox\r
Content-type: message/http\r
Date: Fri, 16 Aug 2013 18:17:36 GMT\r
Memento-Datetime: Fri, 16 Aug 2013 18:13:50 GMT\r
Via: sent by 68.10.149.64 on behalf of http://flickr.cs.odu.edu/remsflickr-ceotty-8161751828.xml,
```

```

delivered by http://ws-dl-02.cs.odu.edu:10101/hm/\r
Link: <http://ws-dl-02.cs.odu.edu:10101/hm/http:
//gutenberg.cs.odu.edu/rem/gutenberg-pride-and-
prejudice.xml>;
rel="current", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/85088497-3a9a-4fb4-9c67-96e71e60d299>;
rel="self", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/08e32760-ac47-4194-a186-c3d79af50652>;
rel="first", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/77cef9fd-d3e1-4737-a59b-142a7936535c>;
rel="last", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/31341132-3abd-412a-874d-6b2f4bae1cb1>;
rel="next", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/f9db8321-7eae-43db-a241-0ee5ea5ec330>;
rel="previous"\r
Content-Length: 7844\r
Connection: keep-alive\r
\r
POST http://ws-dl-02.cs.odu.edu:10102/rem/copy/http:
//flickr.cs.odu.edu/ HTTP/1.1
Host: flickr.cs.odu.edu
Content-type: application/atom+xml
Content-length: 7681

```

Listing 14. Complete Copy Request message. The receiving WO can use the “Link:” header to get an ordered list of messages from its mailbox.

```

POST http://ws-dl-02.cs.odu.edu:10102/rem/copy/http:
//flickr.cs.odu.edu/ HTTP/1.1
Host: flickr.cs.odu.edu
Content-type: application/atom+xml
Content-length: 7681

HTTP/1.1 200 OK
Date: Fri, 16 Aug 2013 18:13:48 GMT
Server: Apache/2.2.15 (Red Hat)

```

Last-Modified: Fri, 16 Aug 2013 18:13:47 GMT

ETag: "5671e6-1d00-4e4148d3e6ccc"

Accept-Ranges: bytes

Content-Length: 7424

Connection: close

Content-Type: text/xml

```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
  xmlns:oreatom="http://www.openarchives.org/ore/atom/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ore="http://www.openarchives.org/ore/terms/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:grddl="http://www.w3.org/2003/g/data-view#"
  xmlns:relationship="http://purl.org/vocab/relationship/"
  xmlns:usw="http://wsdl.cs.odu.edu/uswdo/terms/"
  grddl:transformation="http://www.openarchives.org/ore/atom/atom-grddl.xsl"
  xmlns:le="http://purl.org/atompub/link-extensions/1.0">
  <id>tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty-8161751828</id>
  <link rel="alternate" type="text/html"
    href="http://flickr.cs.odu.edu/flickr-ceotty-8161751828.html" />
  <link rel="self" type="application/atom+xml"
    href="http://flickr.cs.odu.edu/remsflickr-ceotty-8161751828.xml" />
```



```
<link rel="edit" type="application/atom+xml"
  href="http://ws-dl-02.cs.odu.edu:10102/rem/edit/
http:
  //flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/copy"
  type="application/atom+xml"
  href="http://ws-dl-02.cs.odu.edu:10102/rem/copy/
http:
  //flickr.cs.odu.edu/" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
synchronize"
  type="message/http" href="http://ws-dl-02.cs.odu.
edu:10102/
  rem/synchronize/http:
  //flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#self"
  href="http://ws-dl-02.cs.odu.edu:10101/hm/http:
  //flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml"
  last-checked="2013-08-16T18:13:36+00:00"/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#all"
  href="http://ws-dl-02.cs.odu.edu:10101/hm/all"
  last-checked="2013-08-16T18:13:42+00:00"/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#family"
  href="http://ws-dl-02.cs.odu.edu:10101/hm/
  tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty
-8161751828"
  last-checked="2013-08-16T18:13:39+00:00"/>
```

```

<link rel="http://www.openarchives.org/ore/terms/
describes "
  href="http://flickr.cs.odu.edu/rems/
  flickr-ceotty-8161751828.xml#aggregation" />
<source>
  <author>
    <name>ODU WSDL ReM Generator</name>
    <uri>http://ws-dl-02.cs.odu.edu/</uri>
  </author>
</source>
<published>2013-07-30T18:25:35-04:00</published>
<updated>2013-07-30T18:25:35-04:00</updated>
<link rel="license" type="application/rdf+xml"
  href="http://creativecommons.org/licenses/by-nc
/2.5/rdf" />
<rights>This Resource Map is available under the
  Creative Commons Attribution-Noncommercial 2.5
  Generic license</rights>
<title>Kittens</title>
<author>
  <name>ceotty</name>
</author>
<category term="http://www.openarchives.org/ore/
terms/Aggregation "
  label="Aggregation" scheme="http://www.
openarchives.org/ore/terms/" />
<category term="2012-11-06T00:00:00-05:00"
  scheme="http://www.openarchives.org/ore/atom/
created" />
<category term="2012-11-06T00:00:00-05:00"
  scheme="http://www.openarchives.org/ore/atom/
modified" />
<category term="3"

```

```
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/
preservationCopiesMinimumNumber " />
<category term="5"
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/
preservationCopiesMaximumNumber " />
<category term="0.85"
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/beta "
/>
<category term="0.10"
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/gamma "
/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
family#parent "
    type="application/atom+xml" title="Kittens"
    href="http://flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
family#parentOriginal "
    type="application/atom+xml" title="Kittens"
    href="http://flickr.cs.odu.edu/rems/flickr-ceotty
-8161751828.xml" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates "
    href="http://flickr.cs.odu.edu/flickr-ceotty
-8161751828.html"
    type="text/html" title="Kittens" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates "
    href="http://www.flickr.com/photos/ceotty
/8161751828/"
    type="text/html" title="Kittens | Flickr - Photo
Sharing!" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates "
```

```
href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e.jpg"
type="image/jpeg" title="[Medium 500] Kittens"
usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e.jpg"
modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_s.jpg"
type="image/jpeg" title="[Square 75] Kittens"
usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_s.jpg"
modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_q.jpg"
type="image/jpeg" title="[Square 150] Kittens"
usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_q.jpg"
modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_t.jpg"
type="image/jpeg" title="[Thumbnail] Kittens"
usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_t.jpg"
modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
```

```
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
  href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_n.jpg"
  type="image/jpeg" title="[Small 240] Kittens"
  usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_n.jpg"
  modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
  href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_m.jpg"
  type="image/jpeg" title="[Small 320] Kittens"
  usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_m.jpg"
  modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
  href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_z.jpg"
  type="image/jpeg" title="[Medium 640] Kittens"
  usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_z.jpg"
  modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
  href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_c.jpg"
  type="image/jpeg" title="[Medium 800] Kittens"
  usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_c.jpg"
```

```

    modified="2012-11-06T00:00:00-05:00" length="
    1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_1.jpg"
    type="image/jpeg" title="[Large 1024] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_1.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
    1024000" md5="" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
friend"
    href="http://gutenberg.cs.odu.edu/remss/gutenberg-
pride-and-prejudice.xml"
    title="Pride and Prejudice"/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
copyService"
    href="http://gutenberg.cs.odu.edu/remss/gutenberg-
pride-and-prejudice.xml"
    domain="gutenberg.cs.odu.edu"/>
</entry>

```

Listing 15. Corresponding Copy Request message.

B.10 EVENT 203. COPY SERVICE REQUEST.

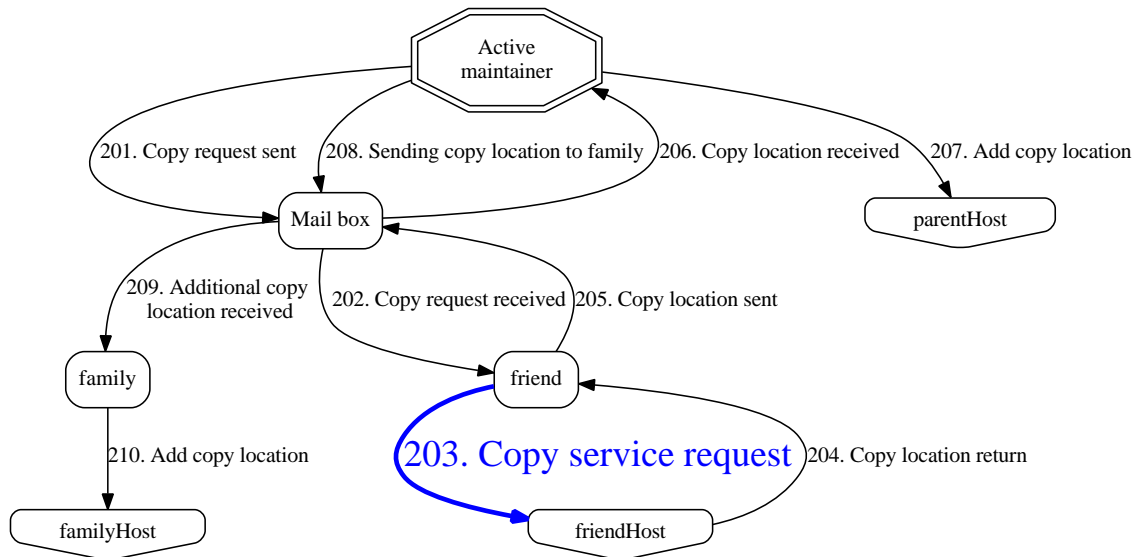


Figure 119. Copying events and messages, event 203.

- *Message Name:* Copy Service Request
- *Explanation:* A parental WO requests that a friend WO create a copy of the parent using the friend's copy service. The exact location of the copy is a function of the service and is unknowable to the requester.
- *Example:* (lines broken for clarity)

```

curl -m 120 -i -X POST --data-binary @/tmp/temp~.xxx4ac1f0a7
-H "Content-type: application/atom+xml"
http://ws-dl-02.cs.odu.edu:10102/rem/copy/
http://gutenberg.cs.odu.edu/

```

```

<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
  xmlns:oreatom="http://www.openarchives.org/ore/atom/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"

```

```

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:ore="http://www.openarchives.org/ore/terms/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:grddl="http://www.w3.org/2003/g/data-view#"
xmlns:relationship="http://purl.org/vocab/
relationship/"
xmlns:usw="http://wsdl.cs.odu.edu/uswdo/terms/"
grddl:transformation="http://www.openarchives.org/
ore/atom/atom-grddl.xsl"
xmlns:le="http://purl.org/atompub/link-extensions
/1.0">
<id>tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty
-8161751828</id>
<link rel="alternate" type="text/html"
href="http://flickr.cs.odu.edu/flickr-ceotty
-8161751828.html" />
<link rel="self" type="application/atom+xml"
href="http://flickr.cs.odu.edu/rem/flickr-ceotty
-8161751828.xml" />
<link rel="edit" type="application/atom+xml"
href="http://ws-dl-02.cs.odu.edu:10102/rem/edit/
http:
//flickr.cs.odu.edu/rem/flickr-ceotty
-8161751828.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/copy"
type="application/atom+xml"
href="http://ws-dl-02.cs.odu.edu:10102/rem/copy/
http:
//flickr.cs.odu.edu/" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
synchronize"

```



```

    type="message/http" href="http://ws-dl-02.cs.odu.edu:10102/rem/synchronize/http://flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/httpmailbox#self"
    href="http://ws-dl-02.cs.odu.edu:10101/hm/http://flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml"
    last-checked="2013-08-16T18:13:36+00:00"/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/httpmailbox#all"
    href="http://ws-dl-02.cs.odu.edu:10101/hm/all"
    last-checked="2013-08-16T18:13:42+00:00"/>
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/httpmailbox#family"
    href="http://ws-dl-02.cs.odu.edu:10101/hm/tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty-8161751828"
    last-checked="2013-08-16T18:13:39+00:00"/>
<link rel="http://www.openarchives.org/ore/terms/describes"
    href="http://flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml#aggregation" />
<source>
  <author>
    <name>ODU WSDL ReM Generator</name>
    <uri>http://ws-dl-02.cs.odu.edu/</uri>
  </author>
</source>
<published>2013-07-30T18:25:35-04:00</published>
<updated>2013-07-30T18:25:35-04:00</updated>
<link rel="license" type="application/rdf+xml"

```

```

    href="http://creativecommons.org/licenses/by-nc/2.5/rdf" />
<rights>This Resource Map is available under the
    Creative Commons Attribution-Noncommercial 2.5
    Generic license</rights>
<title>Kittens</title>
<author>
    <name>ceotty</name>
</author>
<category term="http://www.openarchives.org/ore/terms/Aggregation"
    label="Aggregation" scheme="http://www.openarchives.org/ore/terms/" />
<category term="2012-11-06T00:00:00-05:00"
    scheme="http://www.openarchives.org/ore/atom/created" />
<category term="2012-11-06T00:00:00-05:00"
    scheme="http://www.openarchives.org/ore/atom/modified" />
<category term="3"
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/preservationCopiesMinimumNumber" />
<category term="5"
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/preservationCopiesMaximumNumber" />
<category term="0.85"
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/beta" />
<category term="0.10"
    scheme="http://wsdl.cs.odu.edu/uswdo/terms/gamma" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/family#parent"
    type="application/atom+xml" title="Kittens"

```

```
    href="http://flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/family#parentOriginal"
    type="application/atom+xml" title="Kittens"
    href="http://flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml" />
<link rel="http://www.openarchives.org/ore/terms/aggregates"
    href="http://flickr.cs.odu.edu/flickr-ceotty-8161751828.html"
    type="text/html" title="Kittens" />
<link rel="http://www.openarchives.org/ore/terms/aggregates"
    href="http://www.flickr.com/photos/ceotty/8161751828/"
    type="text/html" title="Kittens | Flickr - Photo Sharing!" />
<link rel="http://www.openarchives.org/ore/terms/aggregates"
    href="http://farm8.staticflickr.com/7131/8161751828_bafa8b207e.jpg"
    type="image/jpeg" title="[Medium 500] Kittens"
    usw:synchronize="http://farm8.staticflickr.com/7131/8161751828_bafa8b207e.jpg"
    modified="2012-11-06T00:00:00-05:00" length="1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/aggregates"
    href="http://farm8.staticflickr.com/7131/8161751828_bafa8b207e_s.jpg"
    type="image/jpeg" title="[Square 75] Kittens"
    usw:synchronize="http://farm8.staticflickr.com/7131/8161751828_bafa8b207e_s.jpg"
```

```
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_q.jpg"
    type="image/jpeg" title="[Square 150] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_q.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_t.jpg"
    type="image/jpeg" title="[Thumbnail] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_t.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_n.jpg"
    type="image/jpeg" title="[Small 240] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_n.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_m.jpg"
    type="image/jpeg" title="[Small 320] Kittens"
```

```
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_m.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_z.jpg"
    type="image/jpeg" title="[Medium 640] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_z.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_c.jpg"
    type="image/jpeg" title="[Medium 800] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_c.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://www.openarchives.org/ore/terms/
aggregates"
    href="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_l.jpg"
    type="image/jpeg" title="[Large 1024] Kittens"
    usw:synchronize="http://farm8.staticflickr.com
/7131/8161751828_bafa8b207e_l.jpg"
    modified="2012-11-06T00:00:00-05:00" length="
1024000" md5="" />
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/
friend"
```

```
    href="http://gutenberg.cs.odu.edu/rems/gutenberg-  
pride-and-prejudice.xml "  
    title="Pride and Prejudice"/>  
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/  
copyService "  
    href="http://gutenberg.cs.odu.edu/rems/gutenberg-  
pride-and-prejudice.xml "  
    domain="gutenberg.cs.odu.edu"/>  
</entry>
```

Listing 16. Copy Request Message sent to receiving WO copy service. Each WO has the URI of its copy service in its REM. Contents of /tmp/temp~.xxx4ac1f0a7.

B.11 EVENT 204. COPY LOCATION RETURNED

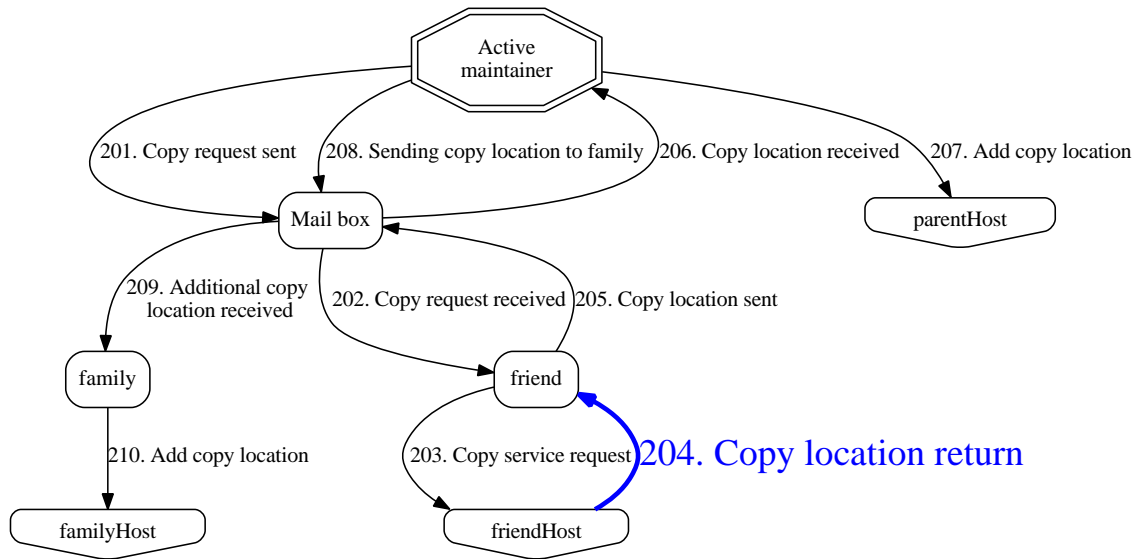


Figure 120. Copying events and messages, event 204.

- *Message Name:* Copy location return
- *Explanation:* The copy request is made by passing REM data to a copy service. The service returns the location of the copy via an HTTP location header.
- *Example:* (lines broken for clarity)

```

HTTP/1.1 201 Created\r
Content-Type: text/html; charset=utf-8\r
Location: http://gutenberg.cs.odu.edu/copyrems/flickr\r
          -ceotty-8161751828.xml\r
Content-Length: 0\r
Connection: keep-alive\r
Server: thin 1.5.0 codename Knife\r
\r

```

Listing 17. Copy service returns copy's URI. The copy URI is returned to the requesting WO.

B.12 EVENT 205. COPY LOCATION SENT.

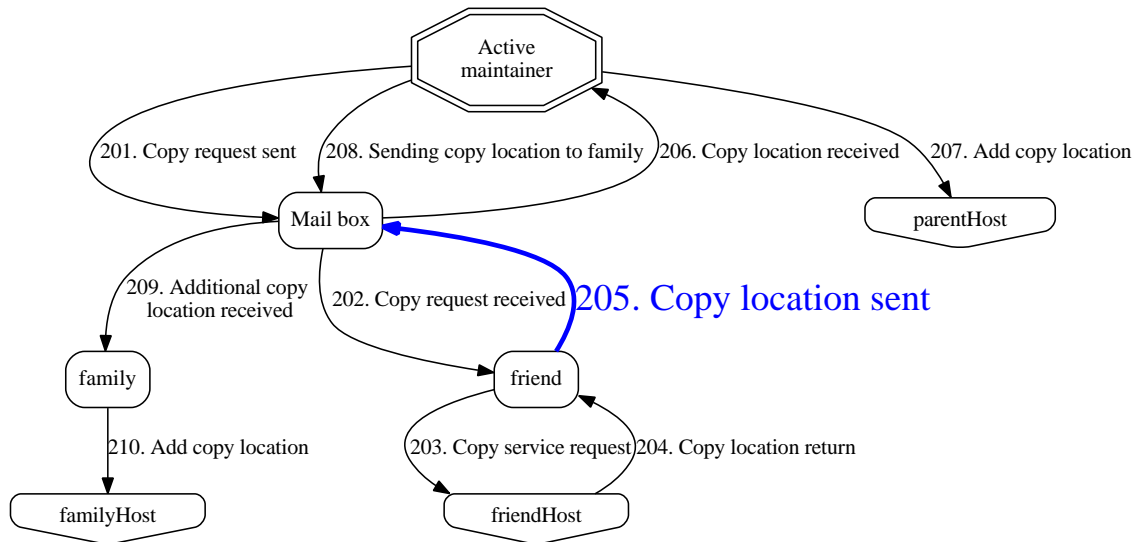


Figure 121. Copying events and messages, event 205.

- *Message Name:* Copy location sent
- *Explanation:* The location of the newly created copy is returned to the requester in the form of a REM patch directive.
- *Example:* (lines broken for clarity)

```

curl -m 120 -i -X POST --data-binary @/tmp/temp~.xxxdc8b74 -i -H
  "Sender: http://gutenberg.cs.odu.edu/rems/
                                     gutenberg-pride-and-prejudice.xml"
  -H "Content-type: message/http"
  http://ws-dl-02.cs.odu.edu:10101/hm/
  http://flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml
  
```

```

PATCH /remsflickr-ceotty-8161751828.xml HTTP/1.1
Host: flickr.cs.odu.edu
Content-type: application/patch-ops-error+xml
Content-length: 223
  
```

```

<?xml version="1.0" encoding="UTF-8"?>
  
```



```
<diff>
<add sel="entry">
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/family"
  href="http://gutenberg.cs.odu.edu/copyrems/flickr-
  ceotty-8161751828.xml"
  title="Kittens" />
</add>
</diff>
```

Listing 18. Copy URI is sent back to requesting WO. The URI of the copy is returned to the originating WO. Contents of /tmp/temp~.xxxdc8b74.

B.13 EVENT 206. RETRIEVE COPY LOCATION.

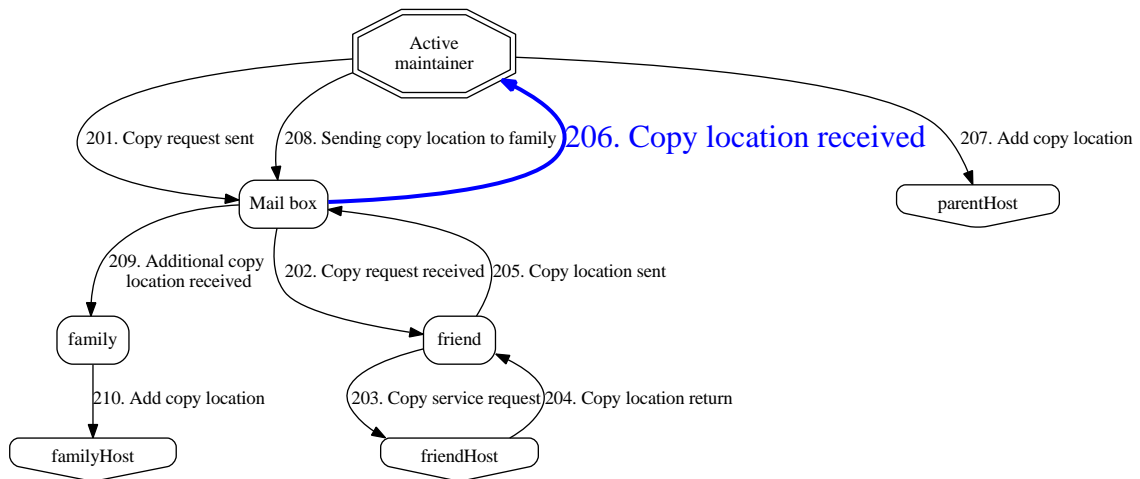


Figure 122. Copying events and messages, event 206.

- *Message Name:* Retrieve copy location
- *Explanation:* A copy request was sent to a friend. The friend may, or may or may not make the copy. If the friend makes the copy, then it will send the copy's location to the originator.
- *Example:* (lines broken for clarity)

```
curl -m 120 -i -o - http://ws-dl-02.cs.odu.edu:10101/hm/http://flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml
```

```
HTTP/1.1 200 OK\r
Server: HTTP Mailbox\r
Content-type: message/http\r
Date: Fri, 16 Aug 2013 18:20:15 GMT\r
Memento-Datetime: Fri, 16 Aug 2013 18:18:07 GMT\r
Via: sent by 68.10.149.64 on behalf of http://gutenberg.cs.odu.edu/rems/gutenberg-pride-and-prejudice.xml,
delivered by http://ws-dl-02.cs.odu.edu:10101/hm/\r
Link: <http://ws-dl-02.cs.odu.edu:10101/hm/http://
```

```

flickr.cs.odu.edu/remes/flickr-ceotty-8161751828.xml>
;
rel="current", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/89574459-38f8-4d65-a5bd-5604acd25e7a>;
rel="self", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/89574459-38f8-4d65-a5bd-5604acd25e7a>;
rel="first", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/89574459-38f8-4d65-a5bd-5604acd25e7a>;
rel="last"\r
Content-Length: 365\r
Connection: keep-alive\r
\r
PATCH /remes/flickr-ceotty-8161751828.xml HTTP/1.1
Host: flickr.cs.odu.edu
Content-type: application/patch-ops-error+xml
Content-length: 223

<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
  <link rel="http://wsdl.cs.odu.edu/uswdo/terms/family
  "
  href="http://gutenberg.cs.odu.edu/copyremes/flickr-
  ceotty-8161751828.xml"
  title="Kittens"
  />
</add>
</diff>

```

Listing 19. Copy URI is retrieved by the originating WO.

B.14 EVENT 207. UPDATE REM WITH COPY'S LOCATION.

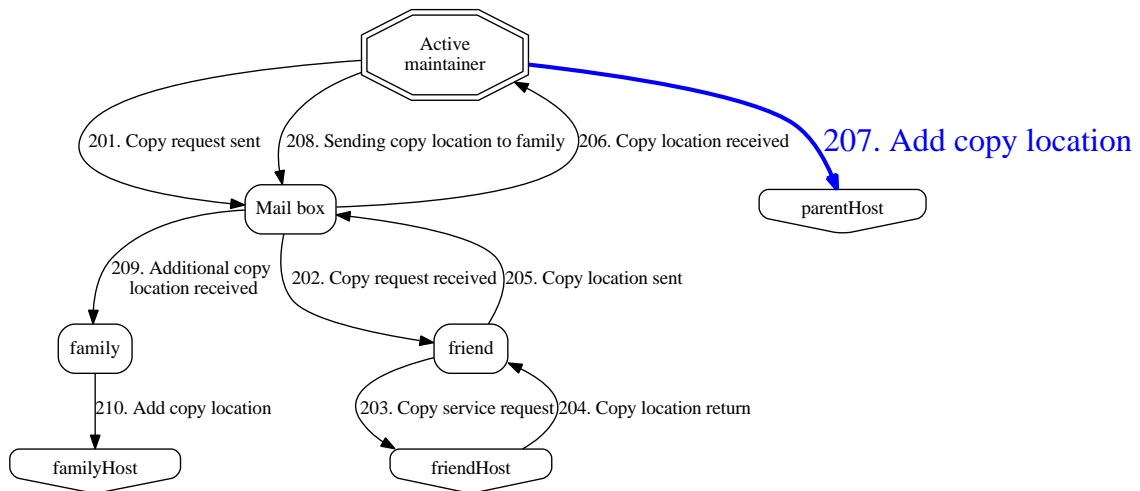


Figure 123. Copying events and messages, event 207.

- *Message Name:* Update REM with copy location
- *Explanation:* The WO will use its edit service to update its REM with the copy location received from the mail box.
- *Example:* (lines broken for clarity)

```

curl -m 120 -X POST -i --data-binary @/tmp/temp~.xxx7a2b1006
-H "Content-type: application/patch-ops-error+xml"
http://ws-dl-02.cs.odu.edu:10102/rem/edit/http:
//flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml
  
```

```

<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
  <link rel="http://wsdl.cs.odu.edu/uswdo/terms/family
  "
  href="http://gutenberg.cs.odu.edu/copyrems/flickr-
  ceotty-8161751828.xml"
  title="Kittens"
/>
  
```

```
</add>  
</diff>
```

Listing 20. Originating WO is updated with copy's URL. Contents of
/tmp/temp~.xxx7a2b1006.

B.15 EVENT 208. BROADCAST COPY LOCATION TO FAMILY.

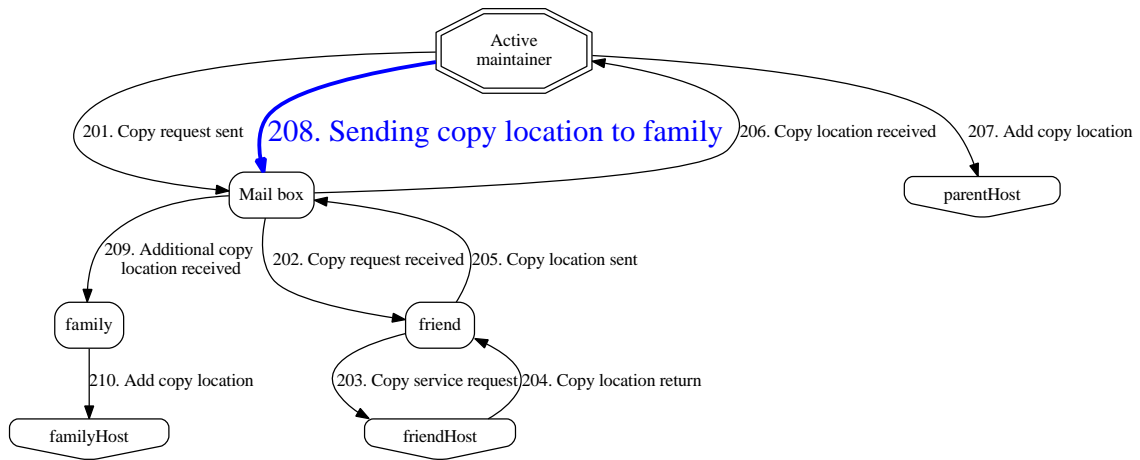


Figure 124. Copying events and messages, event 208.

- *Message Name:* Broadcast copy location to family
- *Explanation:* The parental WO will broadcast the location of any new copies to all family members by using the family mailbox.
- *Example:* (lines broken for clarity)

```

curl -m 120 -i -X POST --data-binary @/tmp/temp~.xxx3553e8e5 -i
-H "Sender: http://flickr.cs.odu.edu/rems/
                                     flickr-ceotty-8161751828.xml"
-H "Content-type: message/http"
http://ws-dl-02.cs.odu.edu:10101/hm/
tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty-8161751828
  
```

```

PATCH /remself HTTP/1.1
Host: selfHost
Content-type: application/patch-ops-error+xml
Content-length: 223
  
```

```

<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
  
```

```
<link rel="http://wsdl.cs.odu.edu/uswdo/terms/family
"
href="http://gutenberg.cs.odu.edu/copyrems/flickr-
ceotty-8161751828.xml"
title="Kittens"
/>
</add>
</diff>
```

Listing 21. Copy's URI is sent to all of the originating WO "family" members. The originating WO may not know the URI of all family members because new members may be created after the message is sent, or the USW family may become disconnected at some time. Contents of /tmp/temp~.xxx3553e8e5.

B.16 EVENT 209. RETRIEVE COPY LOCATION MESSAGE.

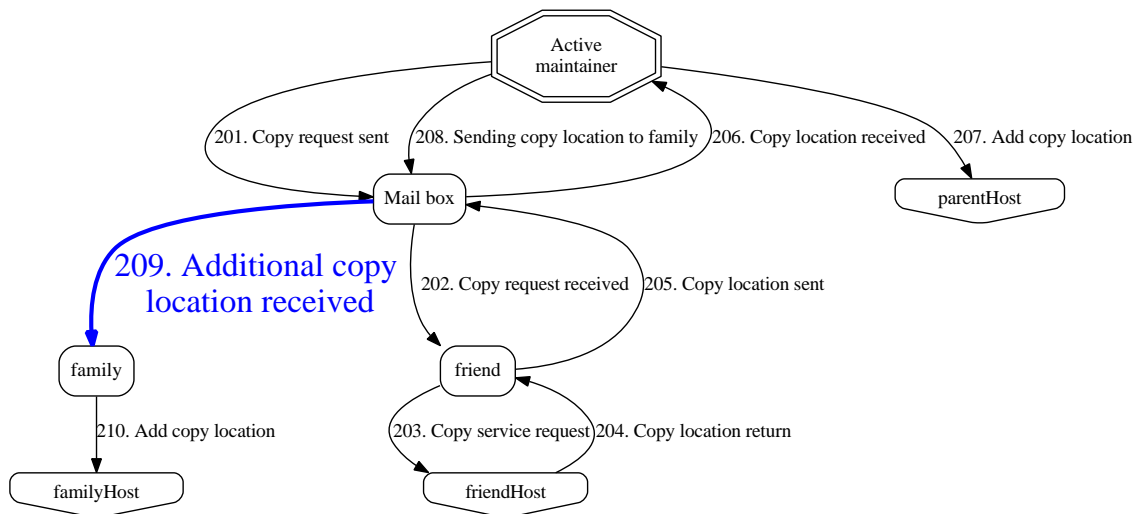


Figure 125. Copying events and messages, event 209.

- *Message Name:* Retrieve copy location message
- *Explanation:* The parental WO sends the location of any new copy to all family members via the family mailbox.
- *Example:* (lines broken for clarity)

```
curl -m 120 -i -o - http://ws-dl-02.cs.odu.edu:10101/hm
    /tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty-8161751828
    2>/dev/null
```

```
HTTP/1.1 200 OK\r
Server: HTTP Mailbox\r
Content-type: message/http\r
Date: Sat, 17 Aug 2013 20:10:33 GMT\r
Memento-Datetime: Sat, 17 Aug 2013 20:10:30 GMT\r
Via: sent by 68.10.149.64 on behalf of
    http://flickr.cs.odu.edu/remes/flickr-ceotty
    -8161751828.xml ,
    delivered by http://ws-dl-02.cs.odu.edu:10101/hm/\r
Link: <http://ws-dl-02.cs.odu.edu:10101/hm
```



```

/tag:uswdo.cs.odu.edu,2012-11-01:flickr-ceotty
-8161751828>;
rel="current", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/7a2ef2a8-dd5d-440a-8e1a-2972ea7d4106>;
rel="self", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/7a2ef2a8-dd5d-440a-8e1a-2972ea7d4106>;
rel="first", <http://ws-dl-02.cs.odu.edu:10101/hm
/id/7a2ef2a8-dd5d-440a-8e1a-2972ea7d4106>;
rel="last"\r
Content-Length: 332\r
Connection: keep-alive\r
\r
PATCH /rems/self HTTP/1.1
Host: selfHost
Content-type: application/patch-ops-error+xml
Content-length: 223

<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
  <link rel="http://wsdl.cs.odu.edu/uswdo/terms/family
  "
  href="http://gutenberg.cs.odu.edu/copyrems/flickr-
  ceotty-8161751828.xml"
  title="Kittens "
  />
</add>
</diff>

```

Listing 22. Family member retrieves new copy URI location. There may be many messages to be serviced.

B.17 EVENT 210. UPDATE REM WITH LOCATION OF NEW COPY.

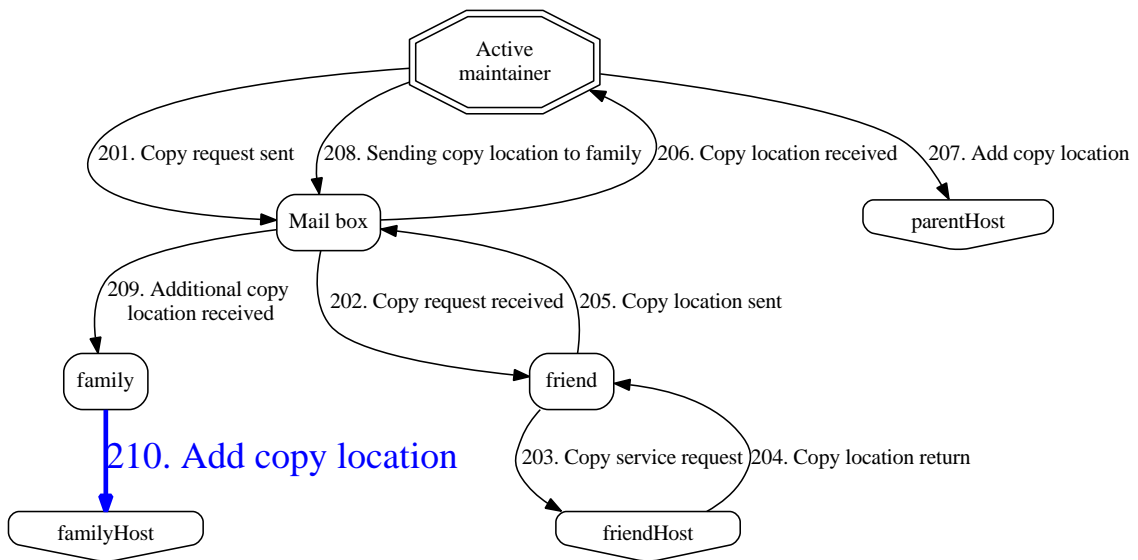


Figure 126. Copying events and messages, event 210.

- *Message Name:* REM Patch
- *Explanation:*
- *Example:* (lines broken for clarity)

```

curl -m 120 -X POST -i --data-binary @/tmp/temp~.xxx7a2b1006
-H "Content-type: application/patch-ops-error+xml"
http://ws-dl-02.cs.odu.edu:10102/rem/edit/http:
//flickr.cs.odu.edu/rem/rem/flickr-ceotty-8161751828.xml
  
```

```

<?xml version="1.0" encoding="UTF-8"?>
<diff>
<add sel="entry">
  <link rel="http://wsdl.cs.odu.edu/uswdo/terms/family
  "
  href="http://gutenberg.cs.odu.edu/copyrems/flickr-
  ceotty-8161751828.xml"
  title="Kittens"
  
```

```
 />  
</add>  
</diff>
```

Listing 23. New copy URI added to existing family member. The patch directive is serviced by the family member's *edit* service. Contents of `/tmp/temp~.xxx7a2b1006`.

B.18 EVENT 301. UPDATE MAILBOX TIME CHECKED.

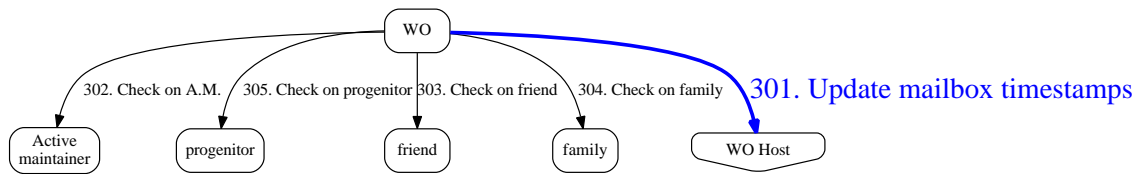


Figure 127. Maintenance events and messages, event 301.

- *Message Name:* REM patch
- *Explanation:* A WO updates the time it last checks each of its mailboxes. This update time is used to control how often a mail box is checked to reduce processing load and network usage. Whenever time is involved across different platforms, time synchronization can be an issue. When maintaining the last date time indicating when a mailbox was checked two times are available, from the requesting WO's host, and from the mailbox server. It is recommended that the time from the DATE general-header field (part of the HTTP header set) be used [195].
- *Example:* (lines broken for clarity)

```

curl -m 120 -X POST -i --data-binary @/tmp/temp~.xxx210e7129
-H "Content-type: application/patch-ops-error+xml"
http://ws-dl-02.cs.odu.edu:10102/rem/edit/http:
//flickr.cs.odu.edu/rem/flickr-ceotty-8161751828.xml

```

```

<?xml version="1.0" encoding="UTF-8"?>
<diff>
<replace sel="/entry/link[@rel='http://wsdl.cs.odu.
edu/uswdo/terms/httpmailbox#self']">
<link rel='http://wsdl.cs.odu.edu/uswdo/terms/
httpmailbox#self'
href="http://ws-dl-02.cs.odu.edu:10101/hm/http:
//flickr.cs.odu.edu/rem/flickr-ceotty-8161751828.
xml"
usw:last-checked="2013-08-17T21:10:29+00:00"

```

```
 />  
</replace>  
</diff>
```

Listing 24. Updating the mailbox last time checked timestamp. Contents of
/tmp/temp~.xxx210e7129.

B.19 EVENT 302. ENSURE PARENT IS ACCESSIBLE.

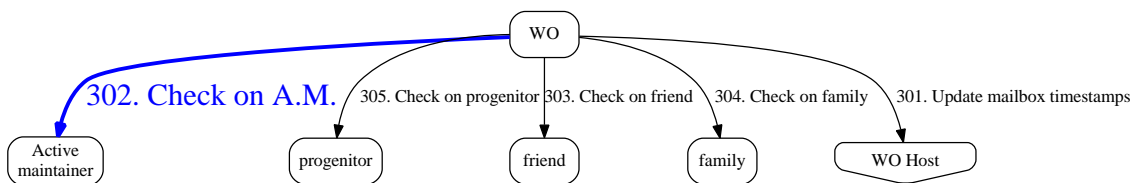


Figure 128. Maintenance events and messages, event 302.

- *Message Name:* curl command
- *Explanation:* Each WO will ensure that its parent is accessible. If the parent is not accessible, then the WO may take family related corrective actions.
- *Example:* (lines broken for clarity)

```
curl -m 120 -i
    http://flickr.cs.odu.edu/flickr-ceotty-8161751828.html
```

B.20 EVENT 303. ENSURE THAT FRIENDS ARE ACCESSIBLE.

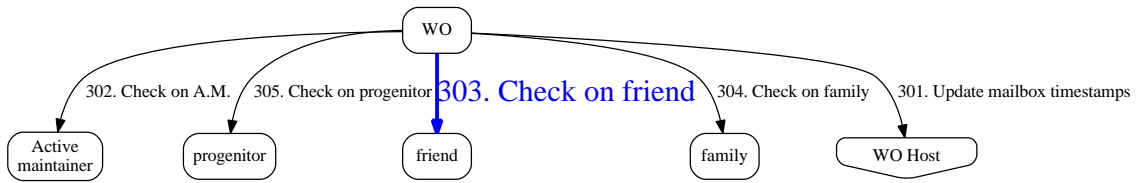


Figure 129. Maintenance events and messages, event 303.

- *Message Name:* curl command
- *Explanation:* Each WO will ensure that its friends are accessible. If a friend is not accessible, then the WO may take family related corrective actions.
- *Example:* (lines broken for clarity)

```

curl -m 120 -i -o -
    http://gutenberg.cs.odu.edu/rems/
                                gutenber-pride-and-prejudice.xml
  
```

B.21 EVENT 304. ENSURE THAT FAMILY MEMBERS ARE ACCESSIBLE.

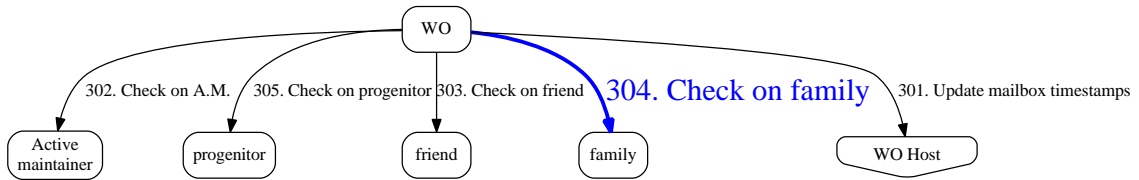


Figure 130. Maintenance events and messages, event 304.

- *Message Name:* curl command
- *Explanation:* Each WO will ensure that its family members are accessible. If a family member is not accessible, then the WO may take family related corrective actions.
- *Example:* (lines broken for clarity)

```

curl -m 120 -i -o -
  http://gutenberg.cs.odu.edu/copyrems/
      flickr-ceotty-8161751828.xml
  
```


B.22 EVENT 305. ENSURE THAT THE PROGENITOR IS ACCESSIBLE.

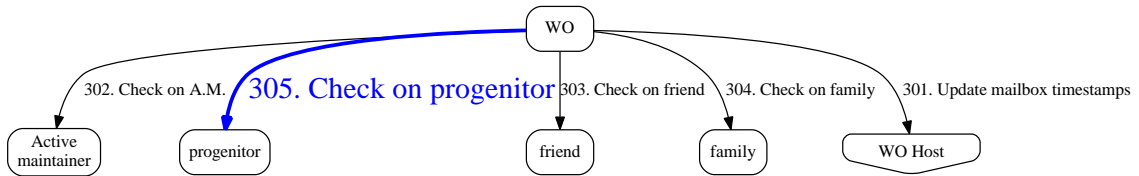


Figure 131. Maintenance events and messages, event 305.

- *Message Name:* curl command
- *Explanation:* Each WO will ensure that the family progenitor is accessible. If the progenitor is not accessible, then the WO may take family related corrective actions.
- *Example:* (lines broken for clarity)

```
curl -m 120 -i -o -
  http://flickr.cs.odu.edu/rems/flickr-ceotty-8161751828.xml
```

APPENDIX C

COMMAND LINE ARGUMENTS

The command line arguments to the USW simulator and robot.

C.1 USW SIMULATOR

Table 51 is a complete list of all command line arguments understood by the USW simulator.

Table 51. USW simulator command line arguments.

Arg.	Definition	Default	Min. value	Max. value
a	The probability threshold used when a preferential attachment mode of selecting the first node is used.	0.5	0	1
b	The threshold value (β) is the against which a locally generated random number is compared to determine if the NEW node will be connected to the current OLD node.	0.5	0	1
c	The file name where graph preservation and description data will be written.	Not used	N/A	N/A

(Continued on the next page.)

Table 51. (Continued from the previous page.)

Arg.	Definition	Default	Min. value	Max. value
d	The number of desired edges used during resiliency repairs.	10	1	1,000.0
f	How to select WOs to be “pinged” to collect message statistics. (see Note I on page 374)	2	0	3
g	The source of the random numbers used throughout the system. (see Note II on page 374)	1	1	4
k	The k size of the neighborhood to looking for the friends of a WO.	1	1	1,000.0
m	The minimum desired number of WO copies per family.	3	1	125,000.0
n	What procedure to be used to choose the next host to assign a parental WO to. (see Note III on page 374)	1	1	2

(Continued on the next page.)

Table 51. (Continued from the previous page.)

Arg.	Definition	Default	Min. value	Max. value
o	The type of resiliency repair technique that will be applied when attempting to reconstruct a graph. (see Note IV on page 374)	1	1	3
p	The percentage (γ) of the “visited” and “to be visited” lists that are used after the wandering node has found a home.	0	0	1
r	The random number seed, required for development to ensure that each run uses the same series of random numbers per run.	123,457.0	1	1,000,000.0
s	The size of the graph.	10	10	1,000,000.0
t	The maximum percent size of the graph that a node being reconnected because of resilience reconstitution will attempt to connect to.	0	0	1

(Continued on the next page.)

Table 51. (Continued from the previous page.)

Arg.	Definition	Default	Min. value	Max. value
w	The command line argument that has all control parameters necessary to reconstitute a graph during resilience reconstitution.	Not Used	NULL	NULL
A	Which first node selection selection policy to execute. (see Note V on page 374)	1	1	5
B	Which next first node selection policy to execute. (see Note VI on page 375)	1	1	3
C	How to select nodes from the <i>toBeVisitedList</i> to forcibly connect to. (see Note VII on page 375)	1	1	3
D	How to select nodes from the <i>visitedList</i> to forcibly connect to is the same as the 'C' argument. The 'D' command line argument is ignored.	2	2	2

(Continued on the next page.)

Table 51. (Continued from the previous page.)

Arg.	Definition	Default	Min. value	Max. value
E	What policy to use when attempting to place WOs on hosts. (see Note VIII on page 375)	1	1	3
F	The name of the input Pajek file.	Not Used	NULL	NULL
G	The name of the output Graphviz file.	Not Used	NULL	NULL
H	The name of the output host data file.	Not Used	NULL	NULL
J	The name of the SWAA output data file.	Not Used	NULL	NULL
L	The number of WO copies to attempt to create during graph reconstruction.	10	1	125,000.0
M	The maximum number of WO copies to attempt to create.	10	1	125,000.0
N	The number of hosts in the system.	100	0	2,000.0
O	Collect images.	Not Used	NULL	NULL
P	The name of the output Pajek file.	Not Used	NULL	NULL

(Continued on the next page.)

Table 51. (Continued from the previous page.)

Arg.	Definition	Default	Min. value	Max. value
R	Boolean whether or not to reset the random number sequence for each new WO.	TRUE	N/A	N/A
S	The name of the file that contains the sequence of random numbers to be used (see Note IX on page 375).	N/A	N/A	N/A
T	Method to be used to compute number of WOs to connect to after the first connection. (see Note X on page 375)	1	1	10
V	The name of the output visitation data file.	Not Used	NULL	NULL
W	Boolean whether or not to create a resiliency control string for use during resiliency reconstruction.	FALSE	N/A	N/A
Z	Boolean whether or not to collect messages	FALSE	N/A	N/A

(Last page.)

Notes:

- I. The possible values for command line option 'f' selection of WOs to be “pinged” to collect message statistics:
 1. Not used.
 2. Uniform random
 3. High degree preference
 4. Sequential

- II. The possible values for command line option 'g' source of random numbers:
 1. Not used.
 2. Use random numbers based on default seed
 3. Use random numbers based on specific seed
 4. Use random numbers read from file
 5. Use random numbers based on system clock

- III. The possible values for command line option 'n' selection of next host number:
 1. Assign new host based on mod total number of hosts
 2. Assign new host based on random selection from total hosts

- IV. The possible values for command line option 'o' selection of resiliency repair technique:
 1. Start with highest degreed node
 2. Start with lowest degreed node
 3. Select nodes at random

- V. The possible values for command line option 'A' first node selection selection policy:
 1. Always use the first node as the start node
 2. Pick a random start node
 3. Preferential attachment as the start node
 4. Always pick the highest degreed node as the start node

5. Always pick the lowest degreed node as the start node
- VI. The possible values for command line option 'B' next first node selection policy to execute.
 1. FIFO node selection for next OLD node
 2. LIFO node selection for next OLD node
 3. Random node select for next OLD node
 - VII. The possible values for command line option 'C' how to select nodes from the *toBeVisitedList* to forcibly connect to.
 1. Randomly select some number of nodes from the *toBeVisitedList*
 2. FIFO select some number of nodes from the *toBeVisitedList*
 3. LIFO select some number of nodes from the *toBeVisitedList*
 4. Connect to established nodes neighbors than random from the *toBeVisitedList*
 - VIII. The possible values for command line option 'E' how to attempt to create preservation copies
 1. Be polite and attempt to find room for a single preservation copy
 2. Be moderately aggressive and only add one more copy or enough to get to minimum desired
 3. Be aggressive and attempt to make our maximum number of preservation copies
 - IX. The possible values for command line option 'S' defining the name of the file that contains random numbers.
 - The default name of the file that contains the series of random numbers to be read from is: `/home/chuck/Temp/SmallWorld/randomNumbers.data`
 - Any other valid filename. Depending on the characters in the file name, the 'S' argument may need to be escaped.
 - X. The possible values for command line option 'T ' select which function to use to compute number of WOs to connect to

1. $m = \gamma * |toBeVisitedList, visitedList|$
2. $m = \max(1, \lg_e(|toBeVisitedList, visitedList| * \gamma))$
3. $m = \max(1, \lg_e(|toBeVisitedList, visitedList|) * \gamma)$
4. $m = \max(0, \lg_e(|toBeVisitedList, visitedList| * \gamma))$
5. $m = \max(0, \lg_e(|toBeVisitedList, visitedList|) * \gamma)$
6. $m = \max(1, \lg_2(|toBeVisitedList, visitedList| * \gamma))$
7. $m = \max(1, \lg_2(|toBeVisitedList, visitedList|) * \gamma)$
8. $m = \max(0, \lg_2(|toBeVisitedList, visitedList| * \gamma))$
9. $m = \max(0, \lg_2(|toBeVisitedList, visitedList|) * \gamma)$
10. $m = 5 + \lg_2(|toBeVisitedList, visitedList| * \gamma)$

C.2 PRESERVE ME! ROBOT

Table 52 is a complete list of command line arguments understood by the USW robot.

Table 52. Preserve Me! robot command line arguments.

Arg.	Definition	Default	Min. value	Max. value
a	List of additional WOs to be added to the system	NULL	NULL	NULL
d	The name of the script file used to sequence various commands. (see Note I on page 380)	NULL	NULL	NULL
e	USW entry WO.	http://arxiv.cs.odu.edu/arxiv-0704-3647v1.html	NULL	NULL
m	Should message server be reset on start-up?	FALSE	FALSE	TRUE
o	The name of the movie file.	NULL	NULL	NULL
q	List of WOs to be added to the system (see Note II on page 381)	NULL	NULL	NULL
r	List of WOs to be added to the system (see Note II on page 381)	NULL	NULL	NULL

(Continued on the next page.)

Table 52. (Continued from the previous page.)

Arg.	Definition	Default	Min. value	Max. value
s	List of WOs to be added to the system (see Note II on page 381)	NULL	NULL	NULL
t	List of WOs to be added to the system (see Note II on page 381)	NULL	NULL	NULL
v	Title of the USW graph and displayed by the Preserve Me! application.	NULL	NULL	NULL
A	Which first node selection selection policy to execute. (see Note III on page 381)	1	1	6
C	Explicitly incorporate this list of Copy REMs into the process.	NULL	NULL	NULL
D	Which domains should be deleted after the graph is created.	NULL	NULL	NULL
E	Which copy creation attitude to take. (see Note IV on page 381)	1	1	3
F	Which WOs to delete after the USW graph has stabilized?	NULL	NULL	NULL

(Continued on the next page.)

Table 52. (Continued from the previous page.)

Arg.	Definition	Default	Min. value	Max. value
I	How many times to attempt to process all mailbox messages before working on the next WO.	1	1	infinite
L	Limit the USW graph size to a maximum of this number of WOs, regardless of how many WOs are identified.	-1	1	infinite
M	Should REMs be reset on start-up?	FALSE	FALSE	TRUE
P	A list of predefined WOs that will be added to the system.	NULL	NULL	NULL
R	Should the list of WOs be randomized to mimic random users and random events?	FALSE	FALSE	TRUE
S	Method to be used to compute number of WOs to connect to after the first connection. (see Note V on page 381)	1	1	10

(Continued on the next page.)

Table 52. (Continued from the previous page.)

Arg.	Definition	Default	Min. value	Max. value
T	Initial system to known set of conditions based on the test condition number. (see Note VI on page 382)	0	0	4
V	Text file containing explanatory text.	NULL	NULL	NULL
X	The name of the file to restore the USW graph from.	NULL	NULL	NULL
V	Name of the file that has explanatory text used by the Preserve Me! application.	NULL	NULL	NULL
Z	Maximum number of times all WOs in the USW graph should be loaded to check their mailboxes.	0	1	infinite

(Last page.)

Notes:

I. The possible values for command line option 'd ' commands that the robot will execute in sequence:

1. *fileOfWOsToDelete*: same as the 'D ' option.

2. *fileOfDomainsToDelete*: same as the 'd ' option.
 3. *replaceParentHREF*: deprecated (do not use) send commands to all WOs to change their existing parent WO reference to the new parent WO.
 4. *replaceParentProgenitorHREFOrig*: deprecated (do not use)
 5. *replaceParentHREFNew*: send commands to all WOs to change their existing parent WO reference to the new parent WO.
 6. *replaceParentProgenitorHREFNew*: send commands to all WOs to change their existing progenitor WO reference to the new progenitor WO.
- II. All domains lists (options 'q ', 'r ', 's 'and 't ') are combined into a single list for internal processing. Different options were provided for ease of testing .
- III. The possible values for command line option 'A' first node selection selection policy:
1. Always use the first node as the start node
 2. Pick a random start node
 3. Preferential attachment as the start node
 4. Always pick the highest degreed node as the start node
 5. Always pick the lowest degreed node as the start node
 6. Always pick the last created node as the start node
- IV. The possible values for command line option 'E ' how to attempt to create preservation copies
1. Be polite and attempt to find room for a single preservation copy
 2. Be moderately aggressive and only add one more copy or enough to get to minimum desired
 3. Be aggressive and attempt to make our maximum number of preservation copies
- V. The possible values for command line option 'S ' select which function to use to compute number of WOs to connect to
1. $m = \gamma * |toBeVisitedList, visitedList|$

2. $m = \max(1, \lg_e(|toBeVisitedList, visitedList| * \gamma))$
3. $m = \max(1, \lg_e(|toBeVisitedList, visitedList|) * \gamma)$
4. $m = \max(0, \lg_e(|toBeVisitedList, visitedList| * \gamma))$
5. $m = \max(0, \lg_e(|toBeVisitedList, visitedList|) * \gamma)$
6. $m = \max(1, \lg_2(|toBeVisitedList, visitedList| * \gamma))$
7. $m = \max(1, \lg_2(|toBeVisitedList, visitedList|) * \gamma)$
8. $m = \max(0, \lg_2(|toBeVisitedList, visitedList| * \gamma))$
9. $m = \max(0, \lg_2(|toBeVisitedList, visitedList|) * \gamma)$
10. $m = 5 + \lg_2(|toBeVisitedList, visitedList| * \gamma)$

VI. The possible values for command line option 'T ' facilitate the creation of predefined test conditions

1. Defer to options 'm 'and 'M '.
2. Reset the message server, reset all REMs with $\beta = 1$ and $\gamma = 1$.
3. Reset the message server, reset all REMs with $\beta = 0$ and $\gamma = 0$.
4. Reset the message server, reset all REMs with $\beta = 1$ and $\gamma = 0$.
5. Reset the message server, reset all REMs with $\beta = 0$ and $\gamma = 1$.

APPENDIX D

VOCABULARY

D.1 DIGITAL PRESERVATION TERMS

“A major difficulty in any newly emerging discipline, such as digital preservation, is the lack of a precise and definitive taxonomy of terms. Different communities use the same terms in different ways which can make effective communication problematic. The following working set of definitions are those used throughout the handbook and are intended to assist in its use as a practical tool. These definitions will not necessarily achieve widespread consensus among the wide ranging communities the handbook is aiming at, they are offered here as a mechanism to avoid potential ambiguities in the body of the handbook rather than as a definitive gloss. Where they have been taken from existing glossaries, this has been acknowledged.”

Neil Beagrie and Maggie Jones [196]

The following definitions are from “The Handbook for Preservation Management of Digital Materials” [196]:

Access

As defined in the handbook, access is assumed to mean continued, ongoing usability of a digital resource, retaining all qualities of authenticity, accuracy and functionality deemed to be essential for the purposes the digital material was created and/or acquired for.

Digital

Archiving This term is used very differently within sectors. The library and archiving communities often use it interchangeably with digital preservation. Computing professionals tend to use digital archiving to mean the process of backup and ongoing maintenance as opposed to strategies for long-term digital preservation. It is this latter richer definition, as defined under digital preservation which has been used throughout this handbook.

Digital Preservation

Refers to the series of managed activities necessary to ensure continued access to digital materials for as long as necessary. Digital preservation is defined very broadly for the purposes of this study and refers to all of the actions required to maintain access to digital materials beyond the limits of media failure or technological change. Those materials may be records created during the day-to-day business of an organization; "born-digital" materials created for a specific purpose (e.g. teaching resources); or the products of digitization projects. This handbook specifically excludes the potential use of digital technology to preserve the original artifacts through digitization.

- *Long-term preservation* - Continued access to digital materials, or at least to the information contained in them, indefinitely.
- *Medium-term preservation* - Continued access to digital materials beyond changes in technology for a defined period of time but not indefinitely.
- *Short-term preservation* - Access to digital materials either for a defined period of time while use is predicted but which does not extend beyond the foreseeable future and/or until it becomes inaccessible because of changes in technology.

Emulation

A means of overcoming technological obsolescence of hardware and software by developing techniques for imitating obsolete systems on future generations of computers.

Metadata

Information which describes significant aspects of a resource. Most discussion to date has tended to emphasize metadata for the purposes of resource discovery. The emphasis in this handbook is on what metadata are required successfully to manage and preserve digital materials over time and which will assist in ensuring essential contextual, historical, and technical information are preserved along with the digital object.

Migration

A means of overcoming technological obsolescence by transferring digital resources from one hardware/software generation to the next. The purpose of

migration is to preserve the intellectual content of digital objects and to retain the ability for clients to retrieve, display, and otherwise use them in the face of constantly changing technology. Migration differs from the refreshing of storage media in that it is not always possible to make an exact digital copy or replicate original features and appearance and still maintain the compatibility of the resource with the new generation of technology.

Reformatting

Copying information content from one storage medium to a different storage medium (media reformatting) or converting from one file format to a different file format (file re-formatting).

Replication and refreshing

Copying information content from one storage media to the same storage media.

D.2 GRAPH THEORETIC TERMS

A vocabulary of graph theoretic terms and ideas that are applicable:

Average inverse path length ($L(G)^{-1}$)

The inverse of the mean of all the shortest paths in the graph between all nodes u and v . Because the shortest path between vertices in two disconnected components is ∞ , the inverse is 0 and therefore is a valid value that does not cause the computation to fail. A larger AIPL means that the distance between nodes is on average shorter [72].

$$L(G)^{-1} = \frac{1}{n(n-1)} \sum_{u \neq v \in V} \frac{1}{d(u,v)} \quad (103)$$

AIPL is also known as average inverse shortest path (AISP) [197] and average inverse shortest path length (AISPL) [198].

Average path length ($L(G)$)

The mean of all paths lengths between all vertices in the graph.

Centrality

A centrality measurement is a way of quantifying the notion that some components of a graph are more important than others. Some centrality measurements are based purely on data that is available at the graph component level and is invariant with respect to the rest of the graph; these are called *local* centrality measurements. *Global* centrality measurements are dependent on the structure of the graph in to-to.

Centrality, betweenness of an edge ($c_B(e)$)

The proportion of shortest paths between nodes s and t that use edge e [72, 199].

$$c_B(e) = \sum_{s \neq t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}} \quad (104)$$

Centrality, betweenness of an edge relative to all edges in a graph ($c_B(E)$)

The edge that has the highest centrality of all edges is the edge that is most used by all shortest paths in the graph.

$$c_B(E) = \max(c_B(e) | e \in E) \quad (105)$$

Centrality, betweenness of a vertex ($c_B(v)$)

The proportion of shortest paths between nodes s and t that use vertex v [72, 86, 200, 199].

$$c_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (106)$$

Centrality, betweenness of a vertex relative to all vertices in a graph ($c_B(V)$)

The vertex that has the highest centrality of all vertices is the vertex that is used by the most shortest paths in the graph.

$$c_B(V) = \max(c_B(v) | v \in V) \quad (107)$$

Centrality, closeness of a vertex

How close (fewest number of edges) u is to all other vertices [78].

$$c_C(u) = \frac{n-1}{\sum_{u \neq v \in V} d(u, v)} \quad (108)$$

Centrality, degree ($c_D(v)$)

The number of edges incident to a vertex.

$$c_D(v) = d(v) \quad (109)$$

Degreeness only makes sense for vertices. A vertex with a high degreeness is central to a local portion of the graph, but not to the graph in to-to.

Clustering coefficient($C(G)$)

Is a measure of graph's local structure. It is a value between 0 and 1 for a vertex that reflects the percentage of vertices that it and another vertex that it is directly connected to share in common [201, 43].

Component, connected

A *connected component* is a group of vertices in a graph that are reachable from one another. A *strongly connected component* is a group of vertices in a graph that are mutually reachable from one another [202].

Connected

Any vertex can be reached from any other vertex via a path δ_{st} of some length. The term *connected* means that there is a series *edges* between any arbitrary nodes source s and terminus t that can be used to get from node s to t [167].

Constrained average path length ($L(G) = \frac{1}{n(n-1)} \sum_{\substack{u \neq v \in V \\ 0 < d(u,v) < \infty}} d(u,v)$)

The average of all shortest path lengths between nodes u and v , given that there is a path between u and v . The lower an CAPL, the fewer edges on average there are between nodes.

$$L(G) = \frac{1}{n(n-1)} \sum_{\substack{u \neq v \in V \\ 0 < d(u,v) < \infty}} d(u,v) \quad (110)$$

Damage ($Damage(G)$)

The ratio of the AIPL of the current graph to a fully connected graph of the same size. [172].

$$Damage(G) = 1 - \frac{L(G_{fragmented})^{-1}}{L(G_{unfragmented})^{-1}} \quad (111)$$

Degree (k)

The number of edges incident to a node.

$$d(v) = k \quad (112)$$

Density ($\rho(G)$)

The ratio for the edges in the graph to the number of edges in a fully connected graph [106].

$$\rho(G) = \left(\frac{n^2}{2m} - 1\right)\left(1 - \frac{1}{n}\right) \quad (113)$$

Diameter ($D(G)$)

The maximal shortest path between any vertices u and v . [201].

$$D(G) = \max\{d(u, v) : u, v \in V\} \quad (114)$$

Directed

The edge connecting nodes s and t is unidirectional.

Disconnected

A graph is *disconnected* when source node s and terminus node t cannot be reached by a path.

Eccentricity of a graph ($\epsilon(G)$)

The maximal eccentricity of all nodes u in G .

$$\epsilon(G) = \max\{\epsilon(u) : u \in V\} \quad (115)$$

Eccentricity of a node ($\epsilon(u)$)

The maximal distance between vertex u and any other vertex v . [201, 166].

$$\epsilon(u) = \max\{d(u, v) : v \in V\} \quad (116)$$

Edge (E)

A connection between two vertices. For the purposes of this paper, an edge is an infra-structural link.

Fragment, cluster or component

Are used interchangeably and mean a set of *nodes* (there may be only 1 node) that are connected to each other. A graph G may have more than one *component*.

Graph($G(V, E)$)

A composite structure made of vertices and edges. $G(V, E)$ is an ordered pair of disjoint sets (V, E) such that E is a subset of V^2 of the unordered pairs of V [167]. In this discussion, a graph is composed of WOs that have infra-structural links. Infra-structural links are separate and distinct from HTML navigational links.

Neighbor

t is an immediate neighbor to s because they are separated by a path of length one edge.

Network

Is a *graph* with different weights assigned to each edge. By default, all *edges* in a *graph* have a weight of 1. While, *edges* in a *network* may have different weights.

Node

The same as a vertex.

Order of a graph(n)

The number of vertices in the graph.

Path length($|\delta_{st}|$)

The number of edges in a path from a source s vertex to a terminus t vertex.

Path length($d(u, v)$)

The number of edges in a path P from a starting node u to terminating node v [167].

$$d(u, v) = |E(P)|, E(P) = \{u_0u_1, u_1u_2, \dots, v_{-1}v_0\} \quad (117)$$

Path(δ_{st})

The sequence of edges from a source s vertex to a terminus t vertex.

Planar

The graph can be drawn in such a way that no edges cross one another [203].

Radius of a graph($r(G)$)

The minimal eccentricity of all vertices in G [201, 166].

$$r(G) = \min\{\epsilon(u) : u \in V\} \quad (118)$$

Resiliency

The capacity of a system to absorb disturbance and reorganize while undergoing change so as to still retain essentially the same function, structure, identity, and feedbacks [204].

Robustness

The ability of a system to remain functioning under a range of conditions [204].

Self loops

An edge that starts and ends at the same source node.

Simple

There is only one *edge* between any adjacent nodes.

Size($|E|$)

The number of edges in a graph.

Triangles based on a node($\lambda(v)$)

The number of subgraphs of the graph G that have exactly three nodes and three edges and one of the nodes is v [201].

$$\lambda(v) = |\{\Delta \mid v \in V_\Delta\}| \quad (119)$$

Undirected

The edge connecting nodes s and t is bidirectional. t is an immediate neighbor to s because they are separated by one edge and the same edge connects t to s .

Vertex(V)

an elemental entity. When we talk about about a vertex, it is a DO in the

KWF sense. The terms *vertex* and *node* are used interchangeably and mean the same thing.

D.2.1 SELECTED TYPES OF GRAPHS

Graphs can be classified by many different and overlapping criteria including the presence or absence of well defined structural elements. Randić and DeAlba [106] provide an extensive list of different classifications. Within this paper, we are interested in the classifying graphs by their degree distributions. Those processes can be purely random, preferential attachment, classical Watts – Strogatz small-world, or our USW construction process.

Each of these processes generates a graph with distinctively different degree distributions, clustering coefficients ($C(G)$) and expected average path lengths ($L(G)$). Figure 132 on the next page is a plot of representative degree distributions for each of the types of graphs that we are interested in. In Figure 132 on the following page, the red circles are characteristic of a power law distribution and of a preferential attachment degree distribution graph. The black x's are from a small-world graph and look very much like a random distribution because the underlying methodology for creating the edges is random. The difference between a small-world distribution and a random one is the smallness of the degree distribution and having a mean μ that is same as the underlying lattice that was used as the base. While this small-world distribution is ± 4 , a similar random one is ± 10 . A random graph degree distribution is shown with the green triangles, whose μ is centered at $p * n$ and a Poisson distribution for the rest of the degreed nodes.

1. *Preferential attachment*

A special case of *Power Law* graphs. Preferential attachment graphs grow over time by the addition of new vertices.

2. *Power law*

A graph where the likelihood that node u is connected to node v based on the number of edges incident to node v ($p(k) = ck^{-\delta} | \delta > 0, c > 0$). In a power law graph, the more connections a node has, the greater the likelihood that more connections will be made to that node. This is essence of the notion of “preferential attachment” where the “rich get richer.” This accretion process results in a few well-connected nodes and a majority that are very poorly

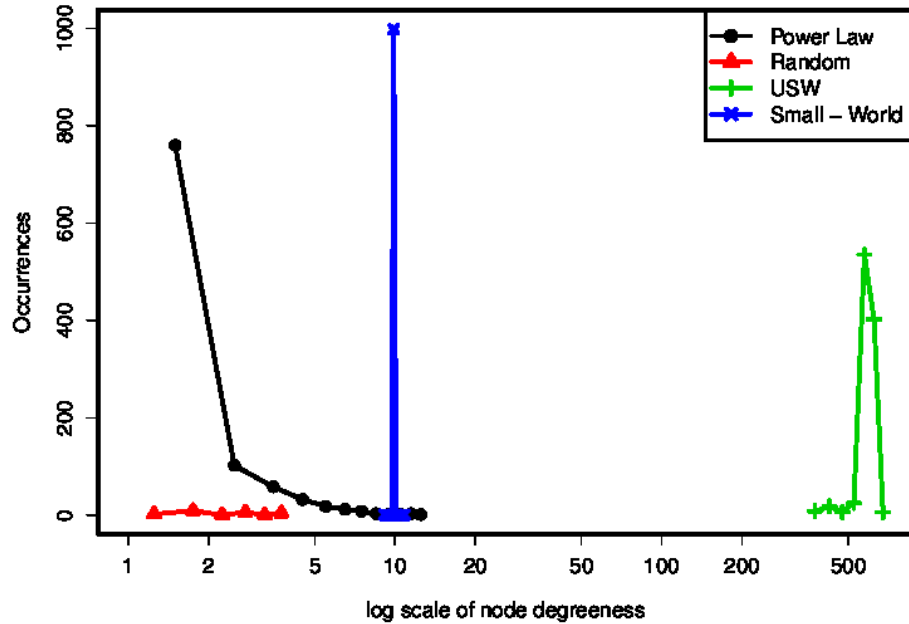


Figure 132. Histogram of representative degree distributions of 1000 node graphs built using random, power law, small-world and USW processes. The degree distribution for a random graph is based on the probability that an edge is created between two nodes. If the probability is low, then few edges are created, but the degrees of the nodes with edges reflects the type or random distribution that was used to create the random number. Power law $p(k) = Ck^{-\alpha}$ degree distributions can show a long tail based on α . Classical small-world graphs have a very narrow degree distribution because the underlying graph was initially a lattice. The shape and location of the unsupervised small-world (USW) graph degree distribution is a function of the β and γ values used to create the graph.

connected and is evidenced by a significant skewing of the graph's degree distribution towards the left (low end of the distribution) (Figure 135(b) on page 397). Power law graphs are characterized by the fraction of their vertices that have a specified degree k . In general, the degree distribution of a power law graph is given by: $p(k) = ck^{-\delta} | \delta > 0, c > 0$.

3. *Regular*

A graph where the minimum number of edges of any vertex $\Delta(G)$ equals the maximum number of edges $\Delta(G)$ of any vertex. All vertices have the same number of edges k (i.e., the same degree).

4. *Lattice*

A graph where every node at location (a,b) has edges to nodes at locations $(a-D,b)$, $(a+D,b)$, $(a, b-D)$ and $(a,b+D)$ where D is some constant, but arbitrary offset. A lattice graph is a regular graph, therefore each node has the same number of incident edges. The resulting degree distribution has only a single value (Figure 133(b) on page 395).

5. *Random*

A graph created by some random process [205, 206] where the likelihood that node u is connected to node v based on a random probability. These stochastic connections are made from any node u to any node v . The center of a random graph's degree distribution is at the product of n and $p(k)$ (Figure 134(b) on page 396). A random graph is one that is generated . At the end of the random graph construction process, the graph may not be connected.

6. *Social networks*

A graph where the average CC is high and the graphic structure mimics human activities.

7. *Small-world*

A graph where the average clustering coefficient $C(G)$ is high and the average path length $L(G)$ is low [43]. Small-world graphs were introduced in [43, 97] by beginning with a k -lattice and rewiring each edge with a probability p . Small-world graphs may be planar or non-planar and there is a greater than 0 probability that the resulting graph will not be simple and nor connected.

Small-World graphs have distinctive average path length and clustering coefficient properties (Table 8 on page 101).

D.2.2 CONNECTED GRAPH METRICS

Here we review a collection of characteristic metrics for connected graphs. In many cases the characteristic does not have meaning, or a computable value when the graph is not connected.

Average path length ($L(G)$)

The average of all shortest (geodesic) path lengths between nodes u and v . The lower an APL, the fewer edges on average there are between nodes. [201]

$$L(G) = \frac{1}{n(n-1)} \sum_{\substack{u \neq v \in V \\ 0 < d(u,v) < \infty}} d(u,v) \quad (120)$$

Clustering coefficient [201, 43].

The likelihood that two neighbors of v are connected.

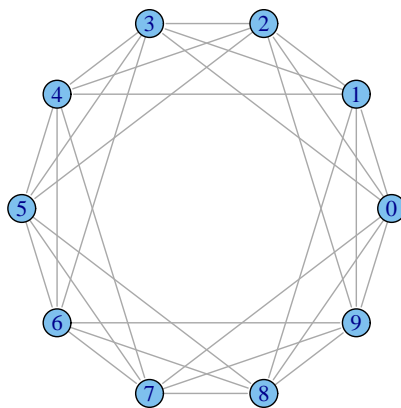
$$c(v) = \frac{2 * \lambda(v)}{d(v)^2 - d(v)} \quad (121)$$

A complete discussion of clustering coefficients can be found in section D.2.3.

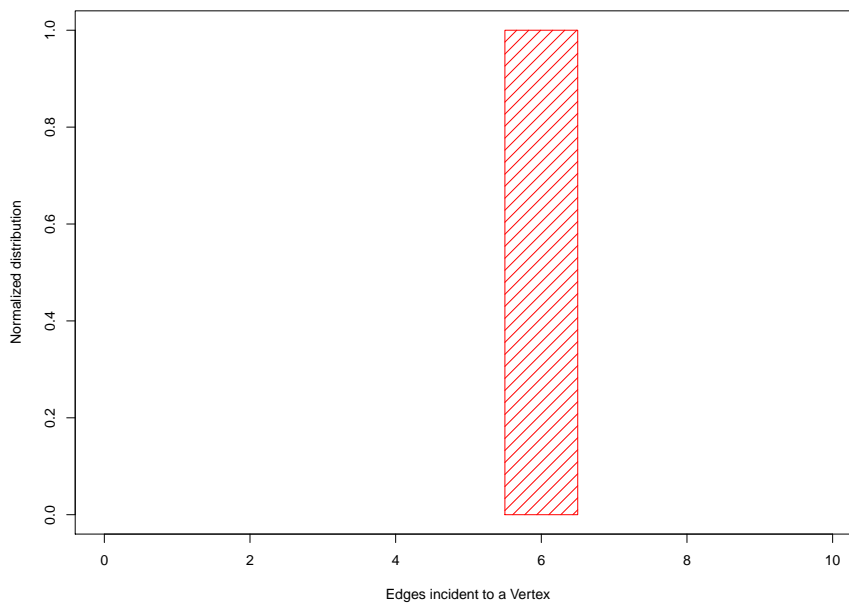
Equations 120, 114 on page 388 and 116 on page 388 are directly related to the length of the path between nodes u and v (Equation 117 on page 389). Equations 104 on page 386, 105 on page 386, 106 on page 386, 107 on page 386, 121, 116 on page 388 and 118 on page 390 are indirectly related to the path length.

D.2.3 CLUSTERING COEFFICIENTS

A clustering coefficient $C(G)$ is an expression of the likelihood that two nodes that are connected to node v are also connected directly to each other [201]. Since being introduced by Watts and Strogatz [43] $C(G)$ has been a staple characteristics topic when discussing Small-World and other graph types. Watts and Strogatz's definition was written in the caption of one figure and therefore open for interpretation. Their $C(G)$ definition and others are given in the following paragraphs.

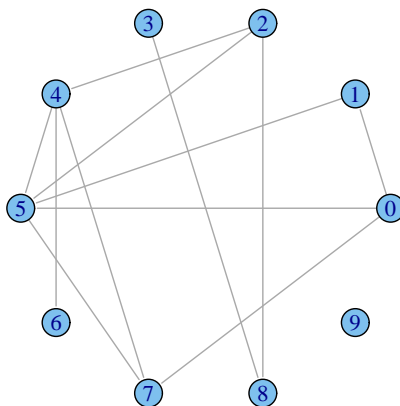


(a) Lattice graph

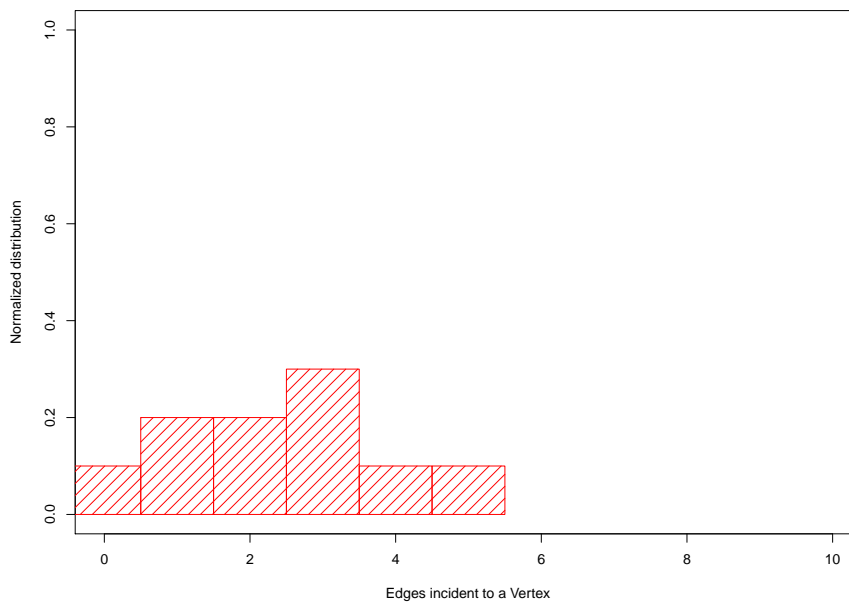


(b) Lattice graph degree distribution

Figure 133. Representative lattice graph and associated degree distribution.

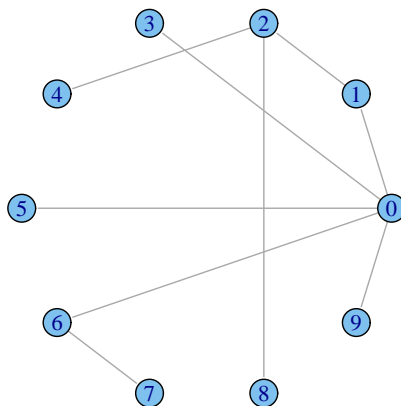


(a) Erdős-Rényi random graph

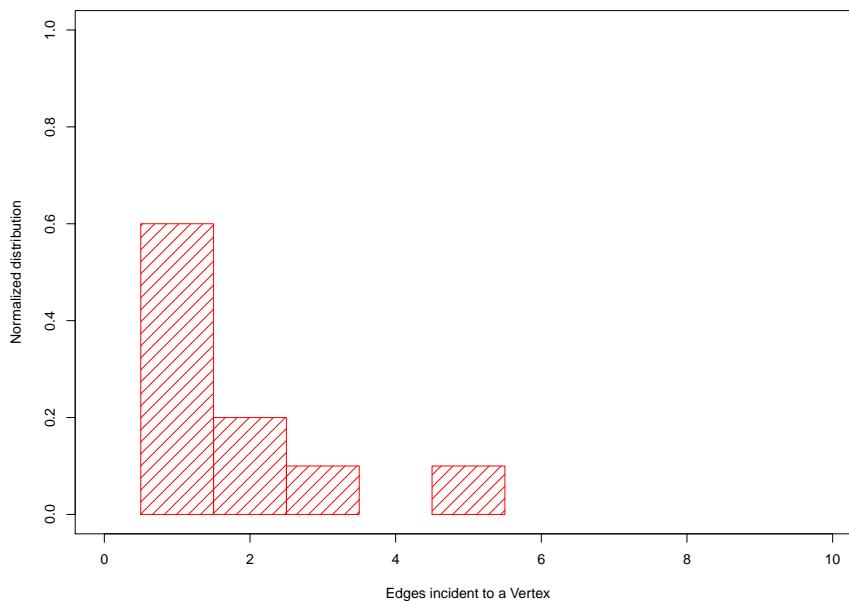


(b) Erdős-Rényi random graph degree distribution

Figure 134. Representative Erdős-Rényi random graph and associated degree distribution.

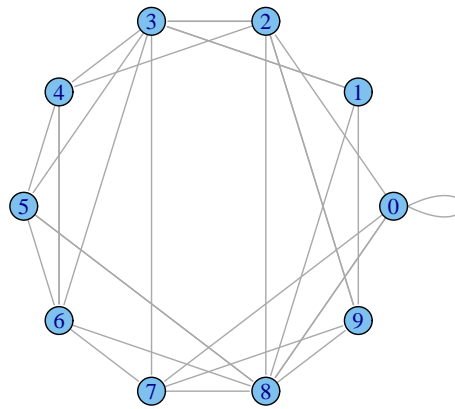


(a) Albert-Barabási scale free

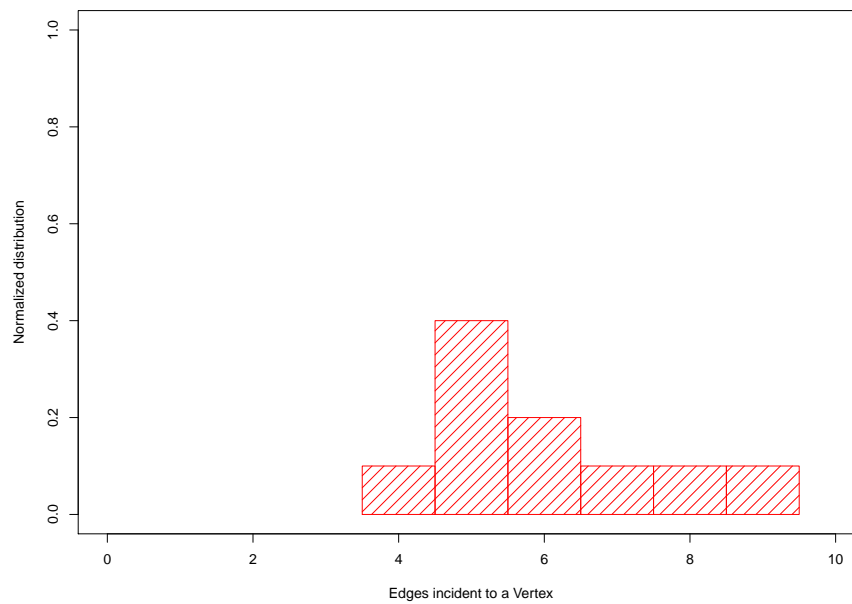


(b) Albert-Barabási scale free degree distribution

Figure 135. Representative Albert-Barabási scale free graph and associated degree distribution.

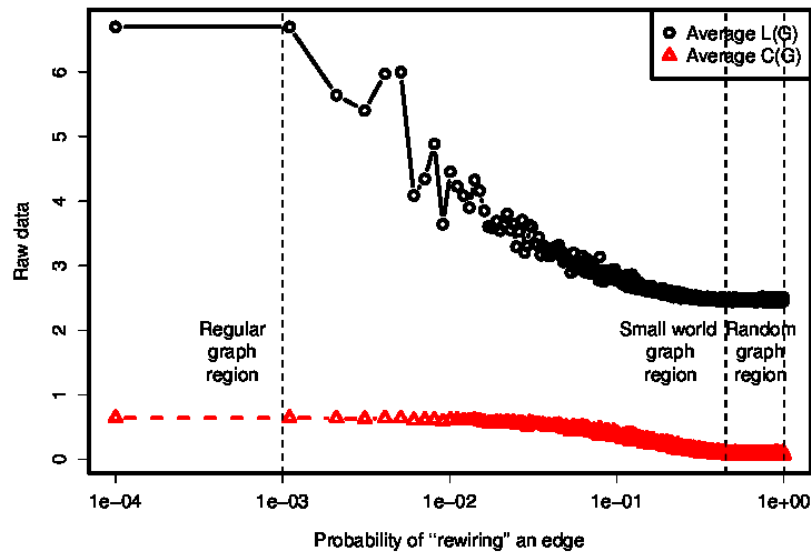


(a) Watts – Strogatz small-world graph

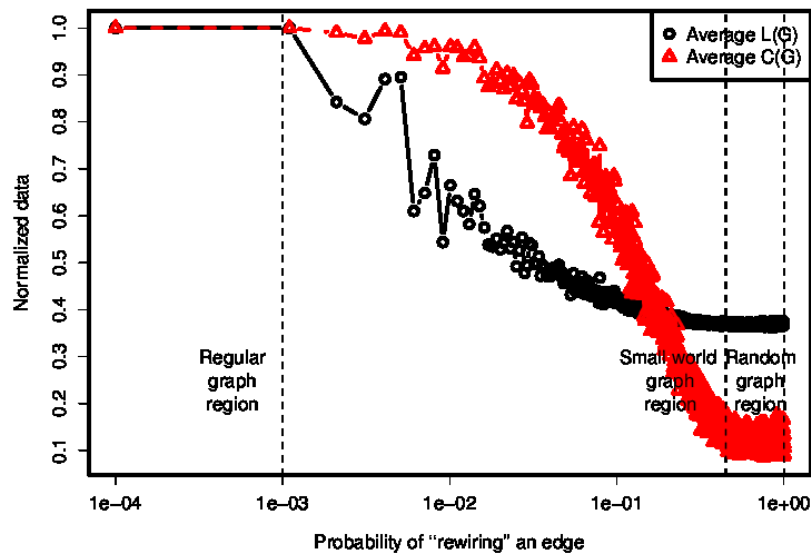


(b) Watts – Strogatz small-world graph degree distribution

Figure 136. Representative Watts – Strogatz small-world graph and associated degree distribution.



(a) Raw data



(b) Normalized data

Figure 137. A “small-world” graph exists along the continuum between a regular lattice and a random graph.

Newman [178] and Opsahl [179] define $C(G)$ as the total number of triangles in the graph divided by the total number of triples in the graph (Equation 122). This is the ratio of the means.

$$C(G)_{Average} = \frac{3 * \Sigma \text{NumberOfTrianglesInTheGraph}}{\text{NumberOfConnectedTriples}} \quad (122)$$

Newman [178] and Watts and Strogatz [43] define $C(G)$ as the summation of the ratio of triangles at each node divided by the number of triples that that node belongs to averaged over all nodes (Equation 123). This is the means of the ratios.

$$C(G)_{AverageLocal} = \frac{1}{n} \sum_i \frac{\text{NumberOfTrianglesConnectedToVertexI}}{\text{NumberOfTriplesCenteredAtVertexI}} \quad (123)$$

If the graph that a pure lattice, then $C(G)$ is purely dependent on the number of nodes (k) that each node is linked to (Equation 123). Equations 124 and 125 are special cases of equation 122 because the number of triangles and triples can be computed directly based on the number of connections (aka, degrees) is given by the value of k .

$$C(G)_{Lattice} = \frac{(3k - 3)}{2(2k - 1)} \quad (124)$$

If the graph is a lattice whose links have been *rewired* with some probability p , then $C(G)$ is initially dependent on the number of nodes (k) that each node is linked to and then the probability that an edge will be rewired (Equation 125).

$$C(G)_{LatticeRewired} = \frac{(3k - 3)}{(4k - 2)} * (1 - p)^3 \quad (125)$$

Figure 138 on the following page is an artificial undirected graph. It was chosen/created because it is small enough to be validated by hand and yet has enough interesting features to be useful when discussing the various $C(G)$ definitions. Table 53 on the next page lists the number of triangles and triples that are used by the the various $C(G)$ equations. The results of evaluating the sample graph are shown in Table 54 on page 402. Functions from the R `igraph` [207] and `sna` [180] packages were used for computing $C(G)$.

This is a quote from Newman comparing the $C(G)$ of structured and random graphs:

Table 53. Number of triangles and triples per node from the test graph.

Node	Num. of Triangles per Node	Num. of Triples per Node	$\frac{\Sigma \text{Triangles}}{\text{Triples}}$
1	1 — {2, 1, 3}	1 — {2, 1, 3}	$\frac{1}{1} = 1$
2	1 — {1, 2, 3}	1 — {1, 2, 3}	$\frac{1}{1} = 1$
3	1 — {1, 3, 2}	6 — {1, 3, 2}, {1,3,4}, {4,3,5}, {1,3,5}, {2,3,5}, {2,3,4}	$\frac{1}{6} = 0.166666667$
4	0	0	$\frac{0}{0} = \text{Not a number}$
5	0	0	$\frac{0}{0} = \text{Not a number}$

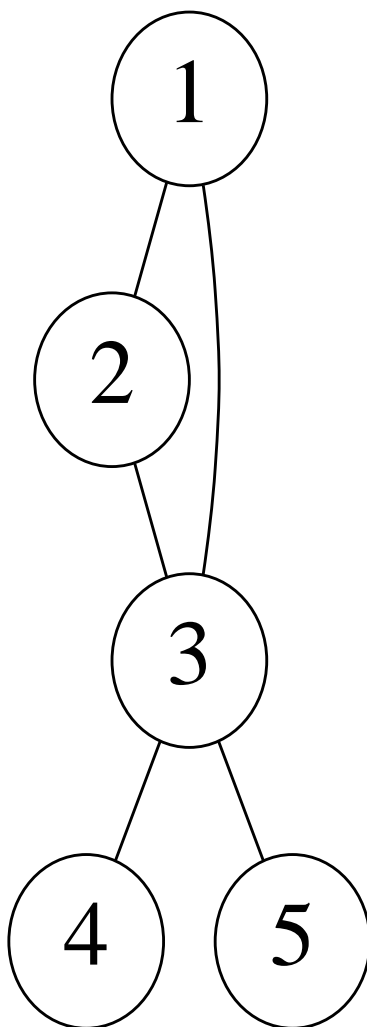


Figure 138. Sample undirected graph used to demonstrate effects of different Clustering Coefficient definitions. The graph is small enough to be tractable by hand and yet has enough different characteristics to be interesting.

Table 54. Results using different approaches for computing the clustering coefficients for the sample graph. The richness of the functions in the igraph package make it more interesting and germane to future analysis.

Package, function control	Underlying Clustering Coefficient equation	Numerical result	Notes
igraph, transitivity(type = ...)			
null	$C(G)_{Average}$	0.375	
undirected	$C(G)_{Average}$	0.375	Same as global
global	$C(G)_{Average}$	0.375	
globalundirected	$C(G)_{Average}$	0.375	Same as global
localundirected	C_i of $C(G)_{AverageLocal}$	{1, 1, 0.166667, NaN, NaN}	A vector of C_i values used to compute $C(G)_{AverageLocal}$
local	C_i of $C(G)_{AverageLocal}$	{1, 1, 0.166667, NaN, NaN}	A vector of C_i values used to compute $C(G)_{AverageLocal}$
average	$C(G)_{AverageTriples}$	0.7222	
localaverage	$C(G)_{AverageTriples}$	0.7222	
localaverageundirected	$C(G)_{AverageTriples}$	0.7222	
sna, gtrans(measure = ...)			
null	$C(G)_{Average}$	0.375	
weak	$C(G)_{Average}$	0.375	The transitive constraint corresponding to: $a \rightarrow b \rightarrow c \Rightarrow a \rightarrow c$
strong	$C(G)_{AverageLocal}$	0.4333	The transitive constraint corresponding to: $a \rightarrow b \rightarrow c \Leftrightarrow a \rightarrow c$
weakcensus	N/A	6	The number of transitive triads used in computing “weak” transitivity.
strongcensus	N/A	26	The number of transitive triads used in computing “strong” transitivity.

“In general, regardless of which definition of the clustering coefficient is used, the values tend to be considerably higher than for a random graph with a similar number of vertices and edges. Indeed, it is suspected that for many types of networks the probability that the friend of your friend is also your friend should tend to a non-zero limit as the network becomes large, so that $C = O(1)$ as $n \Rightarrow \infty$. On the random graph, by contrast, $C = O(n^{-\infty})$ for large n (either definition of C) and hence the real-world and random graph values can be expected to differ by a factor of order n .”

M. E. J. Newman [178]

In this context, Newman is referring to my $C(G)$ or $C(G)_{AverageLocal}$.

D.2.4 CENTRALITY

A centrality measurement is a way of quantifying the notion that some components of a graph are more important than others. Some centrality measurements are based purely on data that is available at the graph component level and is invariant with respect to the rest of the graph; these are called *local* centrality measurements. *Global* centrality measurements are dependent on the structure of the graph in toto. The difference between local and global knowledge is fundamentally one of degree using the idea of *k-neighborhood*. In the minimal case where $k = 1$, all knowledge is based on edges and vertices that are 1 edge away. In the maximal case where $k = D(G)$, all knowledge is based on global knowledge of the graph. Values of k from 1 and $D(G)$ reflecting increasing knowledge of G.

Betweenness centrality

Betweenness is a global centrality measurement. Betweenness is a measure of how many geodesic paths from any vertices $s, t \in V$ use either an edge [72] (Equation 126) or a vertex [72, 86, 200] (Equation 127 on the following page). Removal of a graph component based on its betweenness is a direct attack on the global structure of the graph.

$$c_B(e) = \sum_{s \neq t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}} \quad (126)$$

$$c_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (127)$$

Closeness centrality

Closeness is a global centrality measurement. Closeness quantifies the idea that a vertex has a shortest average geodesic distance when compared to all geodesic distances.

$$c_C(u) = \sum_{v \in V} d(u, v) \quad (128)$$

Degreeness centrality

Degreeness is a local centrality measurement. Degreeness is the number of edges that are incident to a vertex (Equation 129). Degreeness only makes sense for vertices. A vertex with a high degreeness is central to a local portion of the graph, but not to the graph in to-to.

$$c_D(v) = d(v) \quad (129)$$

D.2.5 DISCONNECTED GRAPH METRICS

Here we review a collection of characteristic metrics for disconnected graphs. In many cases the connected graph characteristic does not have meaning, or is not computable when the graph is disconnected.

Constrained average path length ($L(G) = \frac{1}{n(n-1)} \sum_{\substack{u \neq v \in V \\ 0 < d(u,v) < \infty}} d(u, v)$)

The average of all shortest path lengths between nodes u and v , given that there is a path between u and v . The lower an CAPL, the fewer edges on average there are between nodes.

$$L(G) = \frac{1}{n(n-1)} \sum_{\substack{u \neq v \in V \\ 0 < d(u,v) < \infty}} d(u, v) \quad (130)$$

Average inverse path length ($L(G)^{-1}$)

The inverse of the mean of all the shortest paths in the graph between all nodes u and v . Because the shortest path between vertices in two disconnected components is ∞ , the inverse is 0 and therefore is a valid value that does not cause the computation to fail. A larger AIPL means that the distance between

nodes is on average shorter [72].

$$L(G)^{-1} = \frac{1}{n(n-1)} \sum_{u \neq v \in V} \frac{1}{d(u, v)} \quad (131)$$

AIPL is also known as average inverse shortest path (AISP) [197] and average inverse shortest path length (AISPL) [198].

Equation 130 on the preceding page is a constrained APL as compared to a un-constrained APL (Equation 120 on page 394) that restricts the path lengths between nodes to those whose path length is not ∞ . Equation 131 at first appears to be dependent on a path length, but in fact, it is not. If a path does not exist between nodes u and v then, by definition, the path length is infinite ∞ .

D.3 USW TERMS

Here we list a collection of USW related terms and ideas used in this dissertation.

{WOset}

A WO's set of "friend" connections.

visitedSet

The set of WOs that the "wandering" WO visited prior to making its first connection.

toBeVisitedSet

The set of WOs that the "wandering" WO intends to visit prior to making its first connection.

visitedSet \cup toBeVisitedSet

The set of unique WOs that the wandering WO has discovered.

β

The threshold value against which a locally generated random number is compared to determine if the **NEW** node will be connected to the current **OLD** node.

γ

The percentage of the "visited" and "to be visited" lists that are used after the wandering node has found a home.

Active maintenance WO

The single WO in a “family” charged with maintaining the correct number of copies of the progenitor across unique hosts in a USW graph.

Candidate

A WO under consideration to become “connected.” As a WO “wanders” the USW graph, it discovers new WOs. Each newly visited WO is a candidate for the wandering WO to make a connection to prior to the decision to make the connection.

Connected

A WO is connected if it has “connections” to other WOs.

Connection

A logical connection between WOs. A connection is not necessarily an HTML navigational link.

Digital object (DO)

Any digital file.

Family

A collection of WOs that are preservation copies of the “progenitor” WO. A family consists of exactly one “active maintenance” WO and some limited number of “passive maintenance” WOs.

Friend

Any WO is connected to by another WO and is not a member of the second WO’s family.

Host

A computer that is connected to a TCP/IP network.

Introduced

Prior to a WO entering its “wandering” phase, it must be introduced to a WO already existent in the USW graph. This existent WO is “connected” and provides identities of other “connected” WOs.

Maximum number of preservation copies

The maximum number of preservation copies that the active maintainer will create for the family.

Minimum number of preservation copies

The lower bound number that the active maintainer will not go below when managing the number of preservation copies for the family.

Passive maintenance WO

Any “family” WO that detects the loss of a family member or a friend. The loss is communicated to the “active maintenance” WO for resolution.

Policy A

The policy used when selecting the initial and established WO that the “wandering” WO will be introduced to.

Policy B

The policy used when selecting all WOs after the initial WO that the “wandering” WO will explore.

Policy C

The policy used when selecting which WOs the no-longer “wandering” WO will make connections to.

Policy D

Not used.

Policy E

The policy used to decide how many preservation copies an active maintenance WO will make per preservation opportunity.

Progenitor

The original WO from which all “family” WOs are created. The ID of the progenitor is immutable.

REsource Map (REM)

“...describes an Aggregation. A Resource Map asserts which Aggregation it describes, and which resources are constituents of the Aggregation (the Aggregated Resources). In addition, a Resource Map can express relationships and types pertaining to the Aggregation, the Aggregated Resources, the Resource Map itself, and to resources related to them. Each Aggregation may be described by one or more Resource Maps, each of which must have exactly

one representation that is a serialization of the Resource Map according to a specific format. . . .” [133]

Wandering

A WO “wanders” through the USW graph prior to making its first connection. The WO learns about the USW graph, size of the graph, number of WOs that each WO in the graph is connected to, and other data. A WO will wander until one of these conditions is met:

1. the entire USW graph is explored, or
2. a locally generated random number exceeds β .

Web object (WO)

A *DO* that has been augmented with various USW specific metadata.

D.4 MISCELLANEOUS TERMS

Here we list a collection of terms that do not fall into the other categories.

Digital Object Identifier (DOI)

A DOI is a character string (a “digital identifier”) used to uniquely identify an object such as an electronic document. Metadata about the object is stored in association with the DOI name and this metadata may include a location, such as a URL, where the object can be found. The DOI for a document remains fixed over the lifetime of the document, whereas its location and other metadata may change. Referring to an online document by its DOI provides more stable linking than simply referring to it by its URL, because if its URL changes, the publisher need only update the metadata for the DOI to link to the new URL.

Globally Unique Identifier (GUID)

A GUID, is a unique reference number used as an identifier in computer software. The term GUID typically refers to various implementations of the universally unique identifier (UUID) standard.

Uniform Resource Identifier (URI)

A string of characters used to identify a name of a web resource. Such identification enables interaction with representations of the web resource over a

network (typically the World Wide Web) using specific protocols.

Uniform Resource Locator (URL)

A URL is a URI that, in addition to identifying a web resource, specifies the means of acting upon or obtaining the representation: providing both the primary access mechanism, and the network “location” For example, the URL `http://example.org/wiki/MainPage` refers to a resource identified as `/wiki/-MainPage` whose representation, in the form of HTML and related code, is obtainable via HyperText Transfer Protocol (http) from a network host whose domain name is `example.org`.

Uniform Resource Name (URN)

A URN is an Internet resource with a name that, unlike a URL, has persistent significance - that is, the owner of the URN can expect that someone else (or a program) will always be able to find the resource.

Universally Unique Identifier (UUID)

A UUID is an identifier standard used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE). The intent of UUIDs is to enable distributed systems to uniquely identify information without significant central coordination. In this context the word unique should be taken to mean “practically unique” rather than “guaranteed unique” Since the identifiers have a finite size, it is possible for two differing items to share the same identifier. The identifier size and generation process need to be selected so as to make this sufficiently improbable in practice. Anyone can create a UUID and use it to identify something with reasonable confidence that the same identifier will never be unintentionally created by anyone to identify something else. Information labeled with UUIDs can therefore be later combined into a single database without needing to resolve identifier (ID) conflicts.

APPENDIX E

USW MATHEMATICAL TERMS

Table 55 is a collection of mathematical symbols used when discussing the USW graph.

Table 55. USW related mathematical symbols.

Symbol	Explanation and expansion
$A_{\{C,D,E,V\},\{H,L\}}$	<p>The type of profile that an attacker would use against a graph. The first subscript is the metric that will be used (Closeness, Degree, Edge Betweenness, Vertex Betweenness). The second subscript is whether to use the High or Low value of the metric.</p> <p style="text-align: right;">No equation, a definition.</p>
c_{hard}	<p>The maximum number of preservation copies that the active maintainer will strive to create.</p> <p style="text-align: right;">No equation, a definition.</p>
c_{soft}	<p>The minimum number of preservation copies that the active maintainer will strive to create.</p> <p style="text-align: right;">No equation, a definition.</p>
$c_B(e)$	<p>The portion of paths that use a particular edge out of all paths [72].</p> <p style="text-align: right;">$c_B(e) = \sum_{s \neq t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}}$</p>

(Continued on the next page.)

Table 55. (Continued from the previous page.)

Symbol	Explanation and expansion
$c_B(E)$	The maximum $c_B(e)$ for the entire graph. $c_B(E) = \max(c_B(e) e \in E)$
$c_B(V)$	The maximum for the entire graph. $c_B(V) = \max(c_B(v) v \in V)$
$c_B(v)$	The portion of paths that use a particular vertex out of all paths [72, 86, 200]. $c_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$
$c_C(u)$	The closeness of a vertex to all other vertices. $c_C(u) = \frac{n-1}{\sum_{u \neq v \in V} d(u,v)}$
$C(G)_{AverageLocal}$	The clustering coefficient for a single vertex and its neighbors [178, 43]. $C(G)_{AverageLocal} = \frac{1}{n} \sum_i \frac{NumberOfTrianglesConnectedToVertexI}{NumberOfTriplesCenteredAtVertexI}$
$C(G)_{AverageTriples}$	The clustering coefficient for a single vertex and its neighbors. $C(G)_{AverageTriples} = \frac{1}{\sum_{i, C_i \neq 0}} \sum_i \frac{NumberOfTrianglesConnectedToVertexI}{NumberOfTriplesCenteredAtVertexI}$
$c(v)$	The clustering coefficient for a single vertex and its neighbors [201]. $c(v) = \frac{2 * \lambda(v)}{d(v)^2 - d(v)}$

(Continued on the next page.)

Table 55. (Continued from the previous page.)

Symbol	Explanation and expansion
$C(G)_{Average}$	The clustering coefficient for the entire graph [178, 179]. $C(G)_{Average} = \frac{3 * \Sigma \text{NumberOfTrianglesInTheGraph}}{\text{NumberOfConnectedTriples}}$
$C(G)_{Lattice}$	The clustering coefficient for a lattice graph [171]. $C(G)_{Lattice} = \frac{(3k-3)}{2(2k-1)}$
$C(G)$	The clustering coefficient for a Watts – Strogatz small-world graph [43]. $C(G) \sim \frac{3}{4}$
$C(G)$	The clustering coefficient for a random graph [43]. $C(G) \sim \frac{k}{n}$
$C(G)_{LatticeRewired}$	The clustering coefficient for a Watts – Strogatz small-world that is rewired with some probability p [171]. $C(G)_{LatticeRewired} = \frac{(3k-3)}{(4k-2)} * (1-p)^3$
$C(G)_{Rewired}$	(See $C(G)_{LatticeRewired}$.) $C(G)_{Rewired} = C(C_n^k)(1 - \frac{p}{2k})$
$Damage(G)$	How damaged a graph is compared to its previous state [172]. $Damage(G) = 1 - \frac{L(G_{fragmented})^{-1}}{L(G_{unfragmented})^{-1}}$
$D(G)$	The length of the maximal path in the graph [201]. $D(G) = \max\{d(u, v) : u, v \in V\}$

(Continued on the next page.)

Table 55. (Continued from the previous page.)

Symbol	Explanation and expansion
$d(u, v)$	The maximum distance between any two vertices for the entire graph [167]. $d(u, v) = E(P) , E(P) = \{u_0u_1, u_1u_2, \dots, v_{-1}v_0\}$
E	The foundational element in a $G(V, E)$. A E exists between two or more V . No equation, a definition.
$ E $	The number of edges in $G(V, E)$. This is also known as the <i>size</i> of the graph. The expansion is the number of undirected edges in a fully connected graph. $ E = \frac{n(n-1)}{2}$
$ E_{max} $	The maximum number of edges incident to a WO. No equation, a definition.
$\epsilon(G)$	The maximum $\epsilon(u)$ for the entire graph. $\epsilon(G) = \max\{\epsilon(u) : u \in V\}$
$\epsilon(u)$	The maximum distance between a particular vertex and all other vertices in the graph [199]. $\epsilon(u) = \max\{d(u, v) : v \in V\}$
$friendsToBe$	The set of WOs that will be friends of the current WO. No equation, a definition.

(Continued on the next page.)

Table 55. (Continued from the previous page.)

Symbol	Explanation and expansion
$G(V, E)$	The foundational definition of a graph. A composite structure composed of some number of vertices and possibly some edges. Edges must be between vertices in the graph. No equation, a definition.
h_{cap}	The upper limit of WOs that a host will allow. No equation, a definition.
h_{max}	The maximum number of hosts in the USW graph. No equation, a definition.
k	(See E .) No equation, a definition.
$\langle k \rangle$	Average degree for all vertices in the graph. $\langle k \rangle = \frac{\text{orderOfGraphLHS}}{\text{numberVerticesLHS}}$
$\langle k \rangle$	Average degree in a random graph [112]. $\langle k \rangle = \frac{2m}{n} = p(n-1) \approx pn$
$\langle k_{\text{---}} \rangle$	The average degree of a vertex. No equation, a definition.
$L(G)$	Average distance between all vertices in a connected graph. $L(G) = \frac{1}{n*(n-1)} * \sum_{i,j}^n d(v_i, v_j)$

(Continued on the next page.)

Table 55. (Continued from the previous page.)

Symbol	Explanation and expansion
$L(G)^{-1}$	Average of the inverse distances in a graph [72]. $L(G)^{-1} = \frac{1}{n(n-1)} \sum_{u \neq v \in V} \frac{1}{d(u,v)}$
$ LCC $	The size of the Largest Connected Component of the graph. No equation, a definition.
$\lambda(v)$	The number of number of triangles centered on a particular vertex v [201]. $\lambda(v) = \{\Delta \mid v \in V_{\Delta}\} $
m	(See $ E $.) $ E $
n	The number of vertices in $G(V, E)$. This is also known as the <i>order</i> of the graph. No equation, a definition.
n_{\max}	The maximum number of WOs (or vertices) in the USW graph. No equation, a definition.
<i>NewlyDiscoveredSet</i>	The set of discovered WOs. No equation, a definition.
$\rho(G)$	The ratio of the number of actual edges to the maximum possible number of edges [208]. $\rho(G) = \frac{m}{n}$

(Continued on the next page.)

Table 55. (Continued from the previous page.)

Symbol	Explanation and expansion
$\rho(G)$	(See $\rho(G)$ above [106].) $\rho(G) = \left(\frac{n^2}{2m} - 1\right)\left(1 - \frac{1}{n}\right)$
$\rho(G)$	(See $\rho(G)$ above [167].) $\rho(G) = \frac{m}{\binom{n}{2}}$
S_e	An event or activity that a WO completes. No equation, a definition.
S_t	The sequential event used to order events in the a USW simulation. No equation, a definition.
T_{step}	The number of S_t in a time bin. No equation, a definition.
T_{slice}	A single time across simulations. No equation, a definition.
<i>toBeVisitedSet</i>	The list of WOs that will be visited. No equation, a definition.
<i>toBeVisitedSet</i>	The set of WOs that will be visited. No equation, a definition.
V	A necessary and required component of a $G(V, E)$. No equation, a definition.

(Continued on the next page.)

Table 55. (Continued from the previous page.)

Symbol	Explanation and expansion
<i>visitedSet</i>	The list of WOs that have been visited. No equation, a definition.
<i>visitedSet</i>	The set of WOs have been visited. No equation, a definition.
{ <i>WOset</i> }	The set of WOs that friends of the current WO. No equation, a definition.

(Last page.)

APPENDIX F

DEGREE HISTOGRAMS

F.1 INTRODUCTION

A series of USW simulated graphs were created using default values for all parameters (see Section C.1 on page 368), except for:

1. USW graph order $n=500$
2. First WO selection:
 - (a) Always using the same first WO,
 - (b) Selecting a WO at random from the existing USW graph,
 - (c) Always using the last WO that was added to the USW graph.
3. Different ways to decide how many connections to make:
 - (a) $n * \gamma$
 - (b) $\max(1, \ln(n * \gamma))$
 - (c) $\max(1, \ln(n) * \gamma)$
 - (d) $\max(0, \ln(n * \gamma))$
 - (e) $\max(0, \ln(n) * \gamma)$
 - (f) $\max(1, \log_2(n * \gamma))$
 - (g) $\max(1, \log_2(n) * \gamma)$
 - (h) $\max(0, \log_2(n * \gamma))$
 - (i) $\max(0, \log_2(n) * \gamma)$
 - (j) $5 + \log_2(n * \gamma)$

The following sections contain degree histograms from the above parameter combinations.

F.2 HISTOGRAMS BASED RESULTING FROM USING THE SAME FIRST WO

A series of USW simulated graphs were created using default values for all parameters (see Section C.1 on page 368), except for:

1. USW graph order $n=500$
2. First WO selection:
 - (a) Always using the same first WO,
3. Different ways to decide how many connections to make:
 - (a) $n * \gamma$
 - (b) $\max(1, \ln(n * \gamma))$
 - (c) $\max(1, \ln(n) * \gamma)$
 - (d) $\max(0, \ln(n * \gamma))$
 - (e) $\max(0, \ln(n) * \gamma)$
 - (f) $\max(1, \log_2(n * \gamma))$
 - (g) $\max(1, \log_2(n) * \gamma)$
 - (h) $\max(0, \log_2(n * \gamma))$
 - (i) $\max(0, \log_2(n) * \gamma)$
 - (j) $5 + \log_2(n * \gamma)$

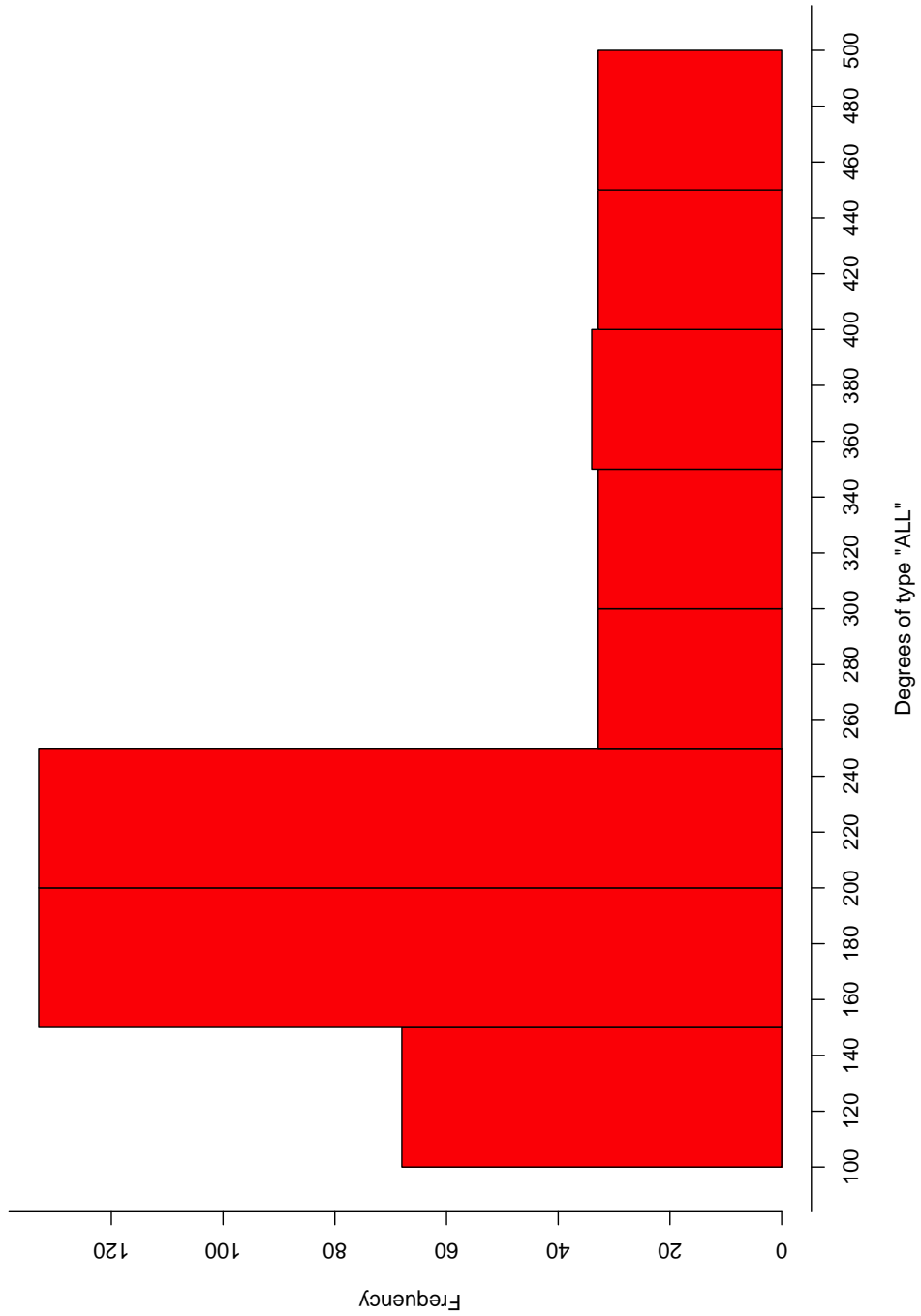


Figure 139. Degree histogram for $n * \gamma$, always using same first WO.

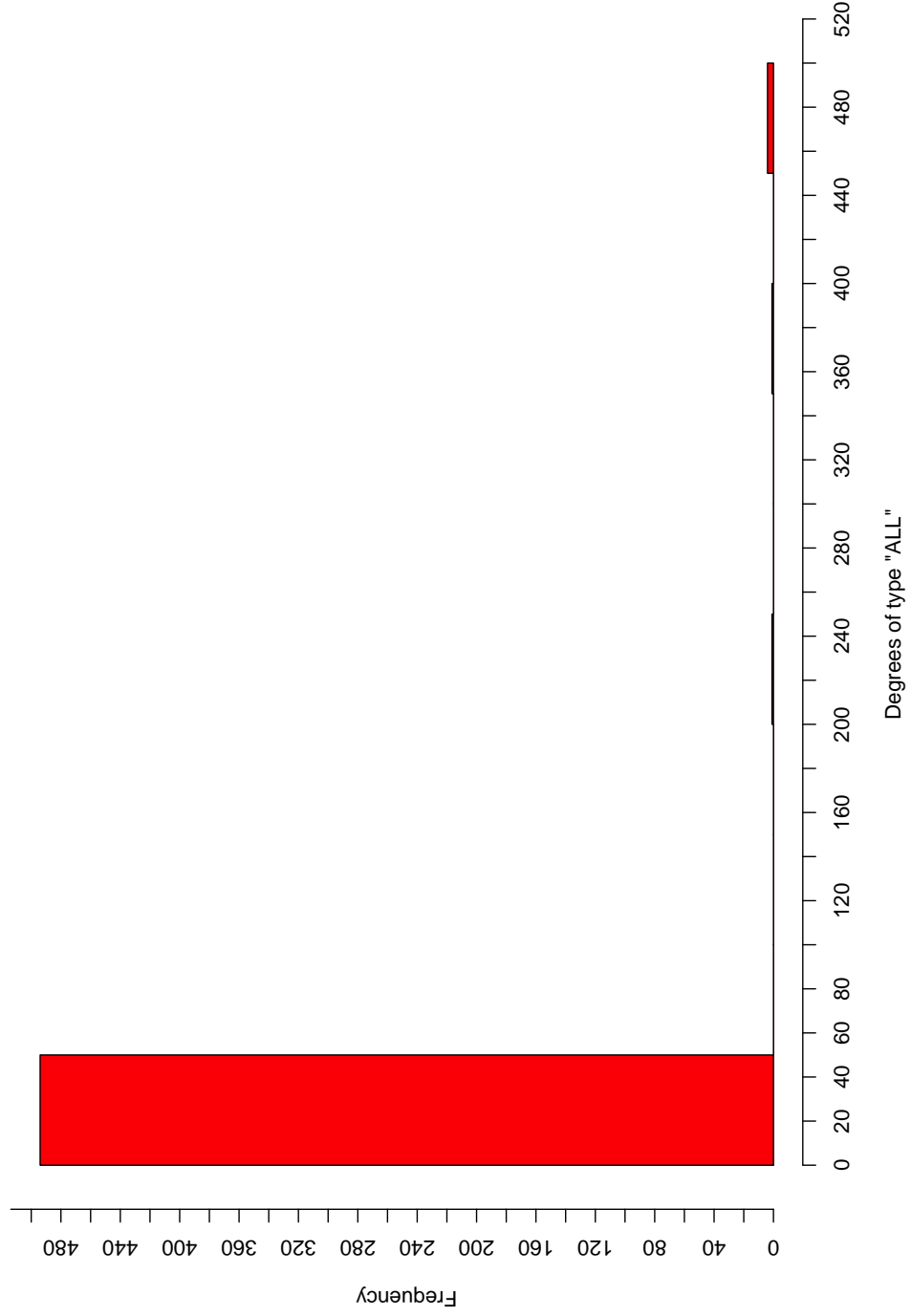


Figure 140. Degree histogram for $max(1, \ln(n * \gamma))$, always using same first WO.

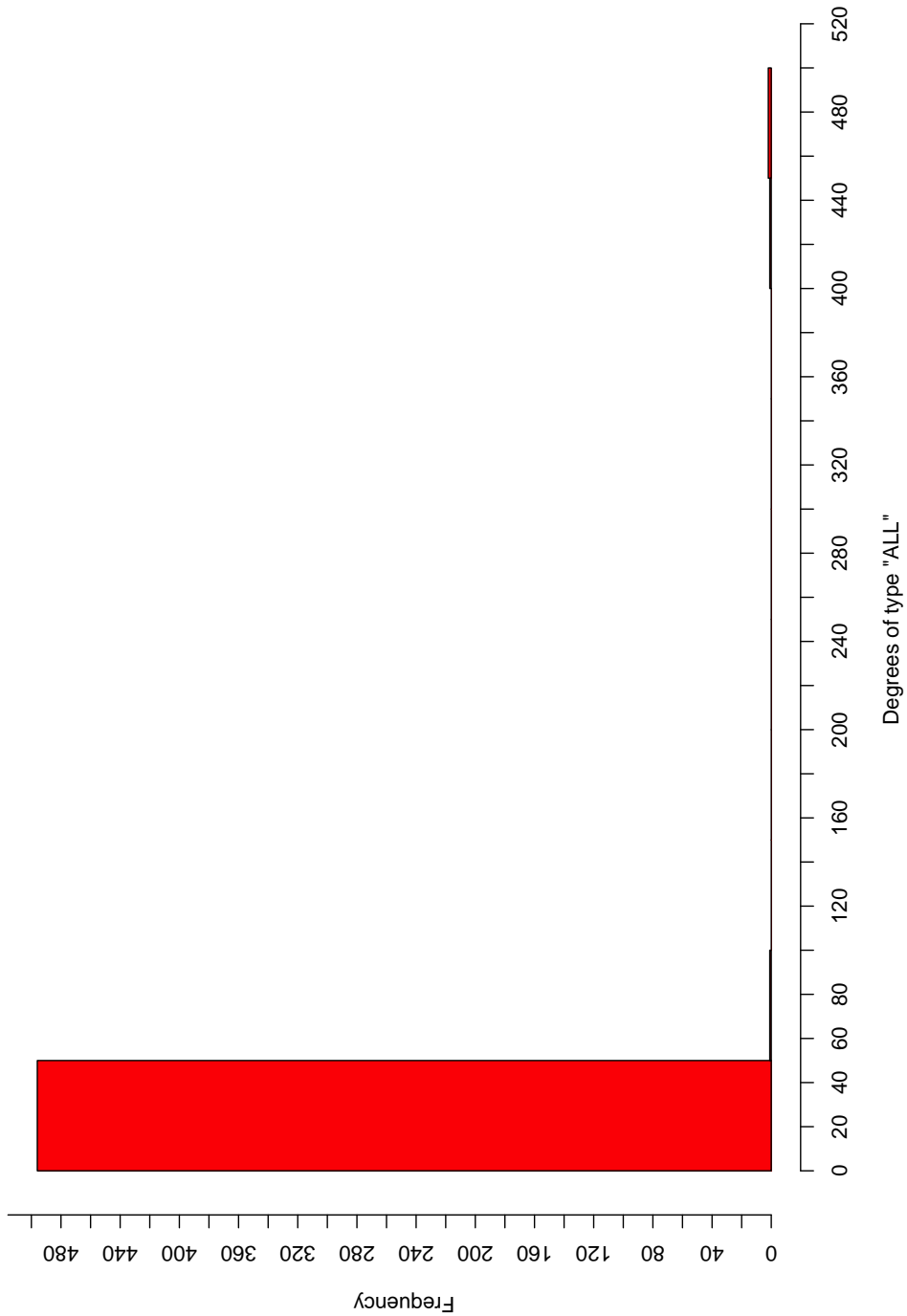


Figure 141. Degree histogram for $max(1, \ln(n) * \gamma)$, always using same first WO.

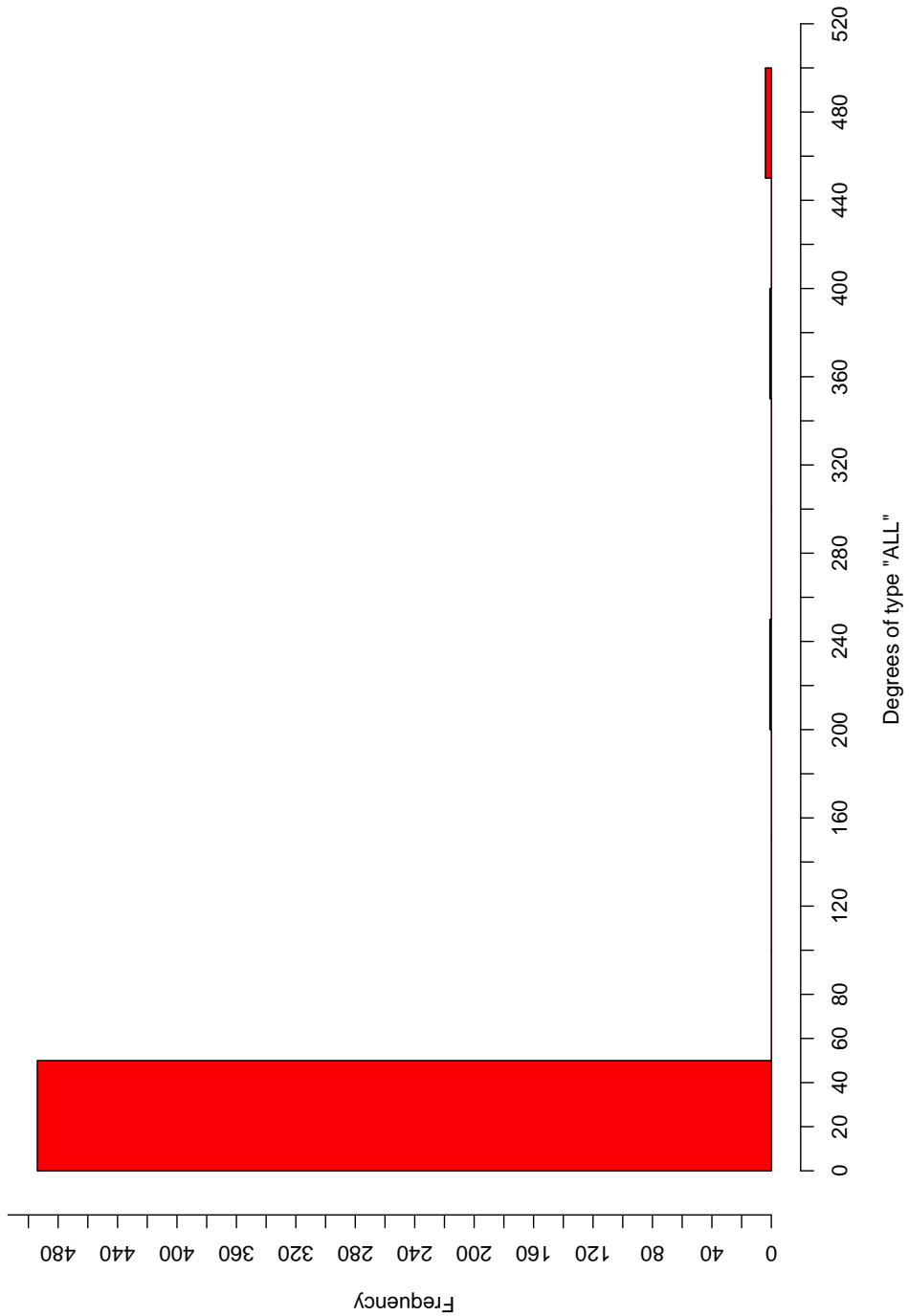


Figure 142. Degree histogram for $max(0, \ln(n * \gamma))$, always using same first WO.

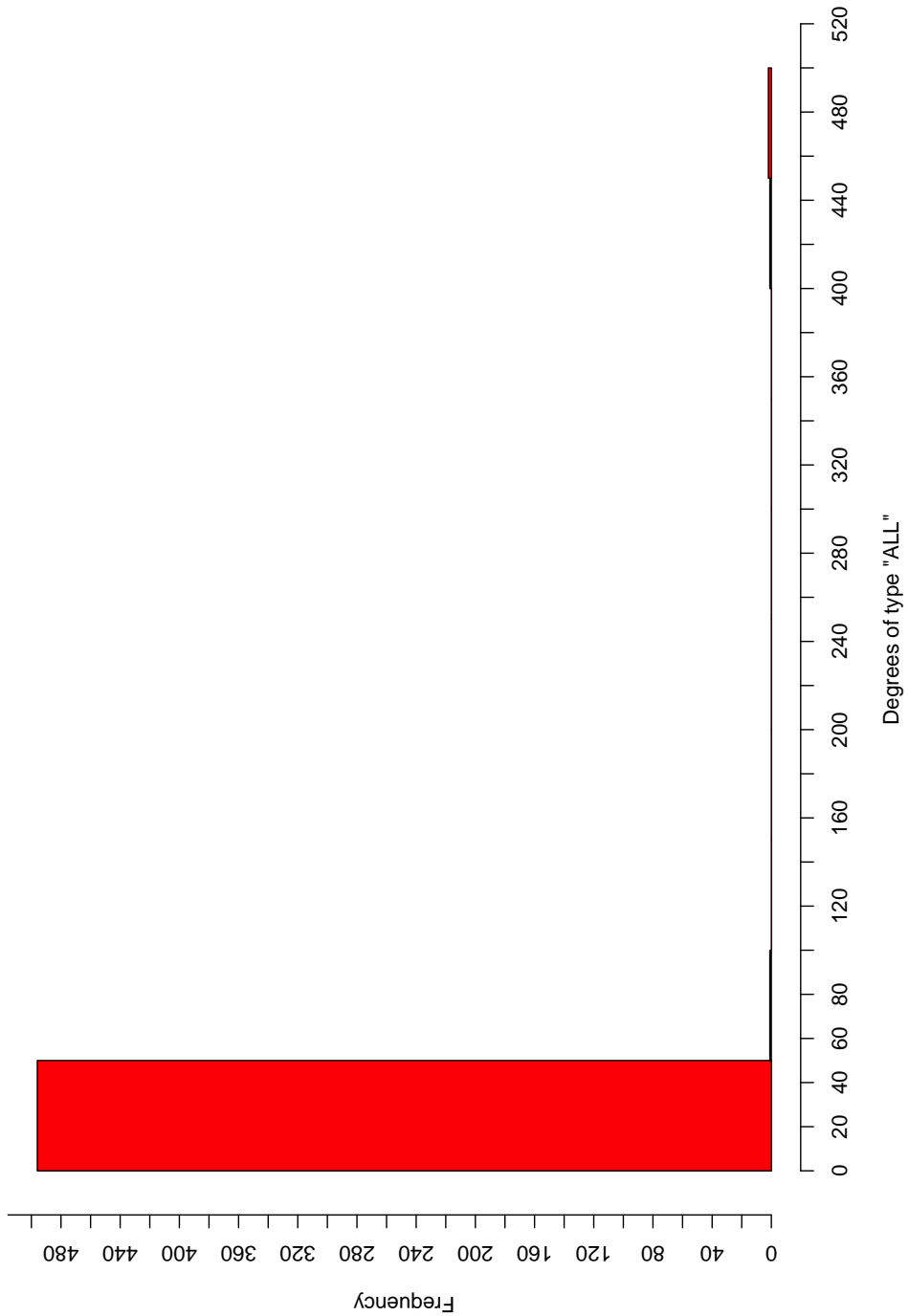


Figure 143. Degree histogram for $max(0, \ln(n) * \gamma)$, always using same first WO.

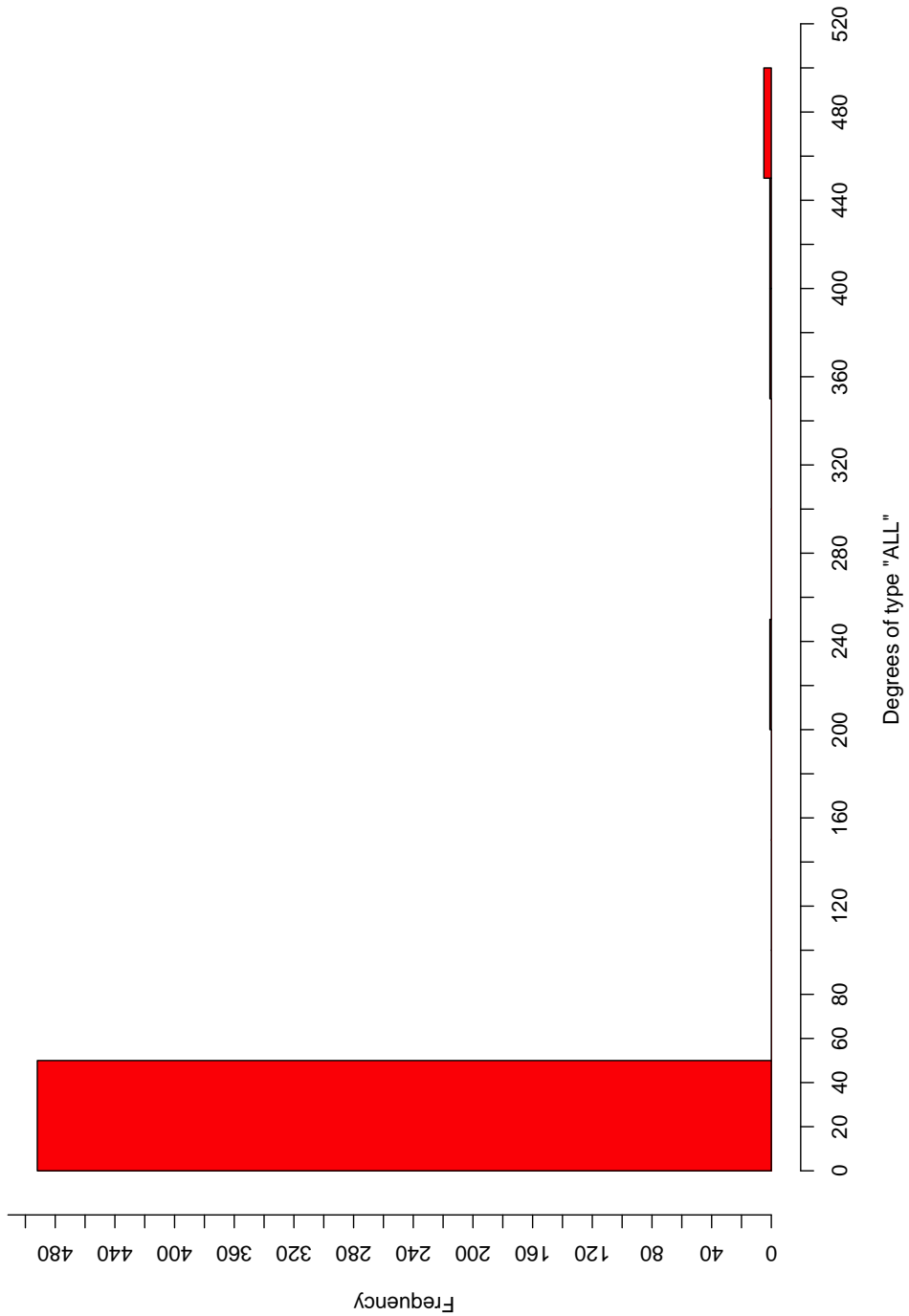


Figure 144. Degree histogram for $max(1, \log_2(n * \gamma))$, always using same first WO.

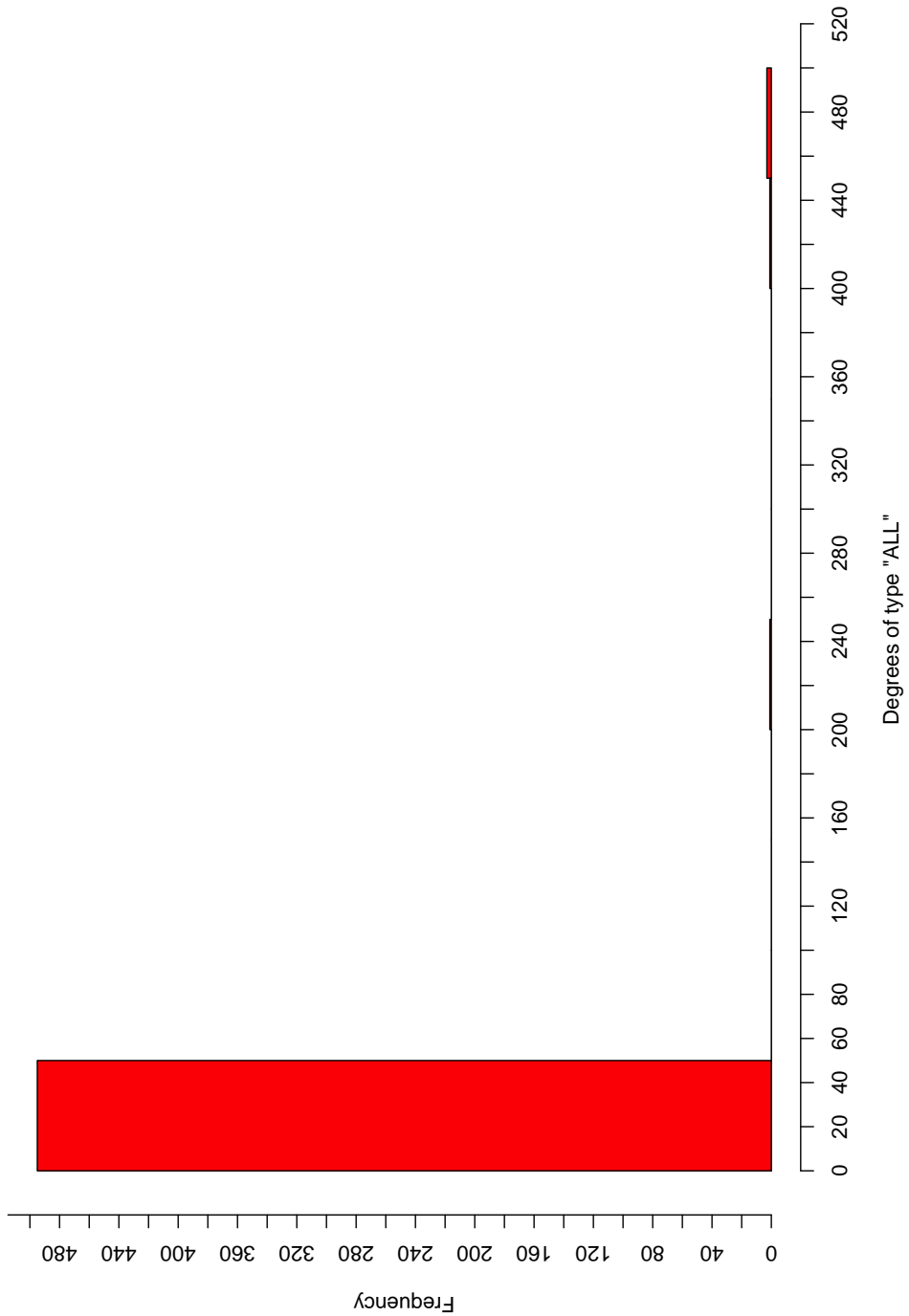


Figure 145. Degree histogram for $max(1, \log_2(n) * \gamma)$, always using same first WO.

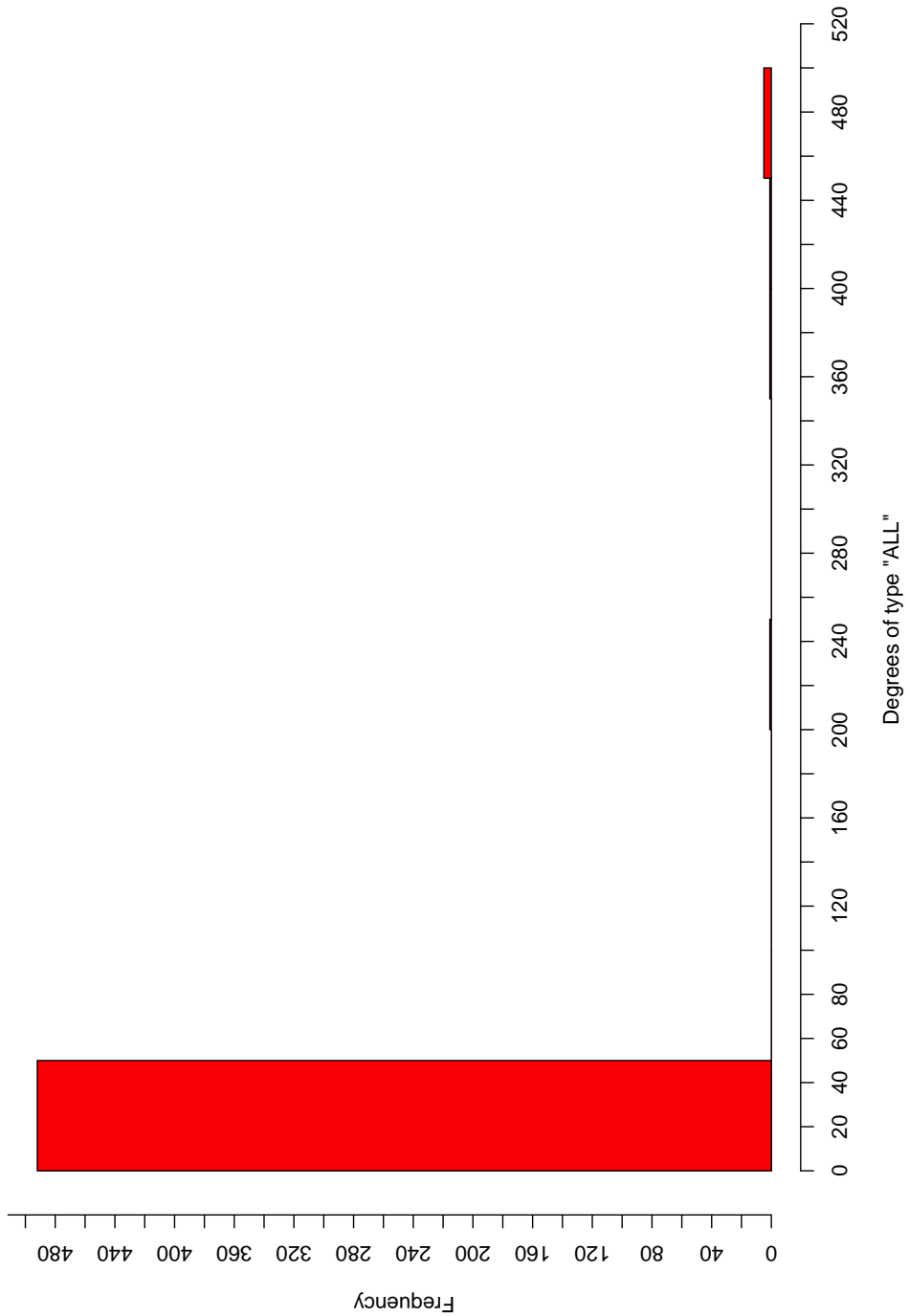


Figure 146. Degree histogram for $\max(0, \log_2(n * \gamma))$, always using same first WO.

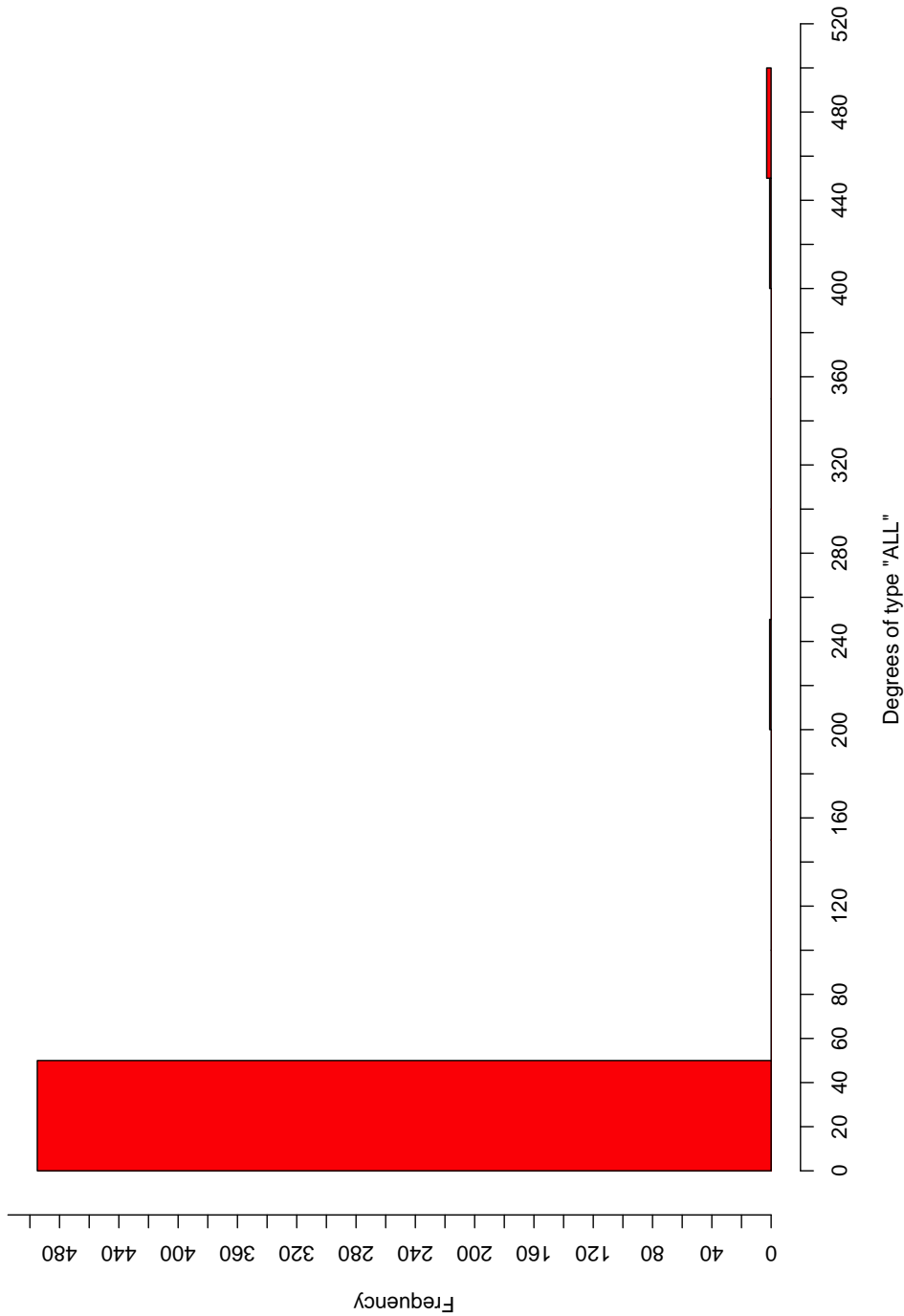


Figure 147. Degree histogram for $max(0, \log_2(n) * \gamma)$, always using same first WO.

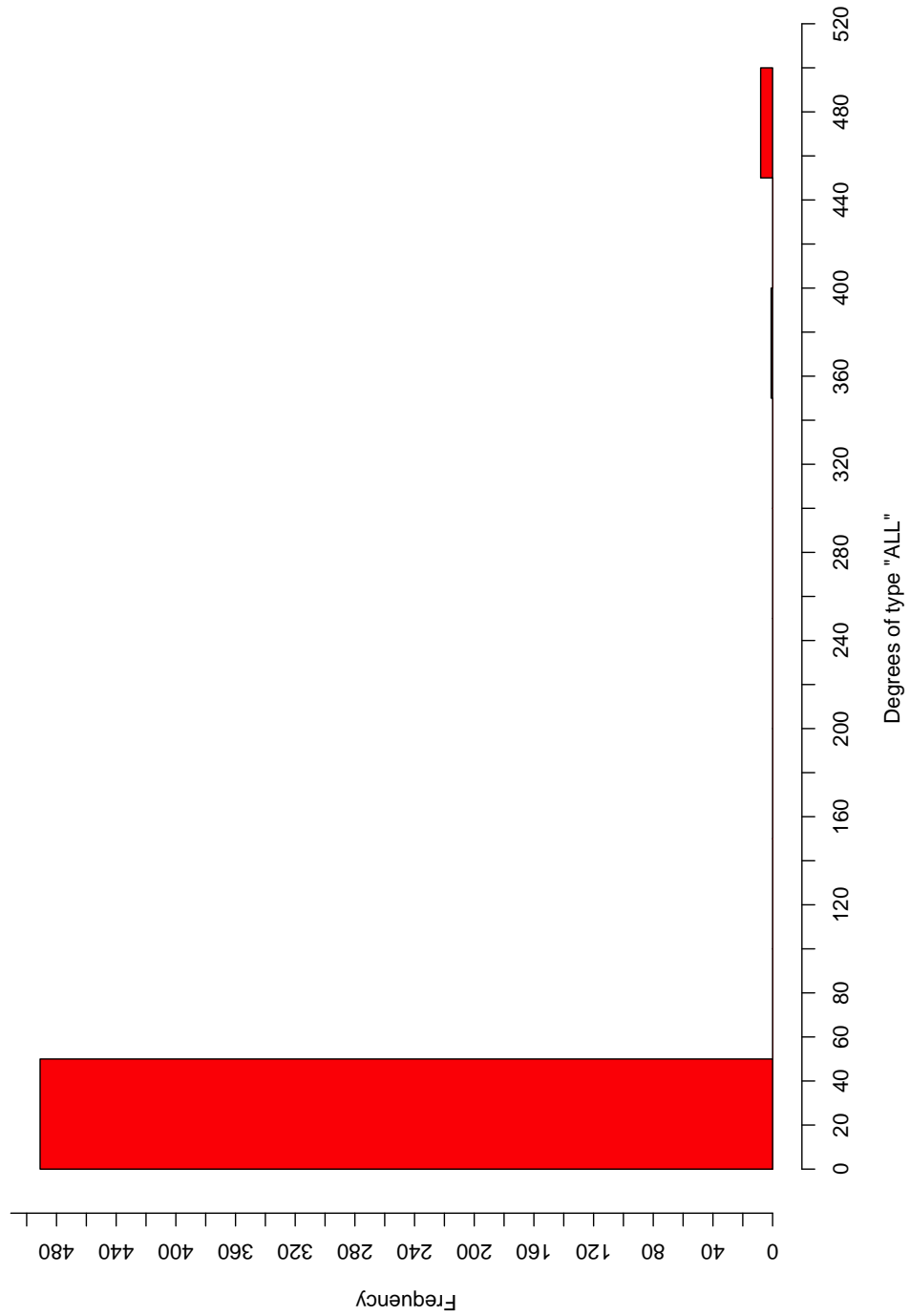


Figure 148. Degree histogram for $5 + \log_2(n * \gamma)$, always using same first WO.

F.3 HISTOGRAMS BASED RESULTING FROM USING A RANDOM FIRST WO

A series of USW simulated graphs were created using default values for all parameters (see Section C.1 on page 368), except for:

1. USW graph order $n=500$
2. First WO selection:
 - (a) Selecting a WO at random from the existing USW graph,
3. Different ways to decide how many connections to make:
 - (a) $n * \gamma$
 - (b) $\max(1, \ln(n * \gamma))$
 - (c) $\max(1, \ln(n) * \gamma)$
 - (d) $\max(0, \ln(n * \gamma))$
 - (e) $\max(0, \ln(n) * \gamma)$
 - (f) $\max(1, \log_2(n * \gamma))$
 - (g) $\max(1, \log_2(n) * \gamma)$
 - (h) $\max(0, \log_2(n * \gamma))$
 - (i) $\max(0, \log_2(n) * \gamma)$
 - (j) $5 + \log_2(n * \gamma)$

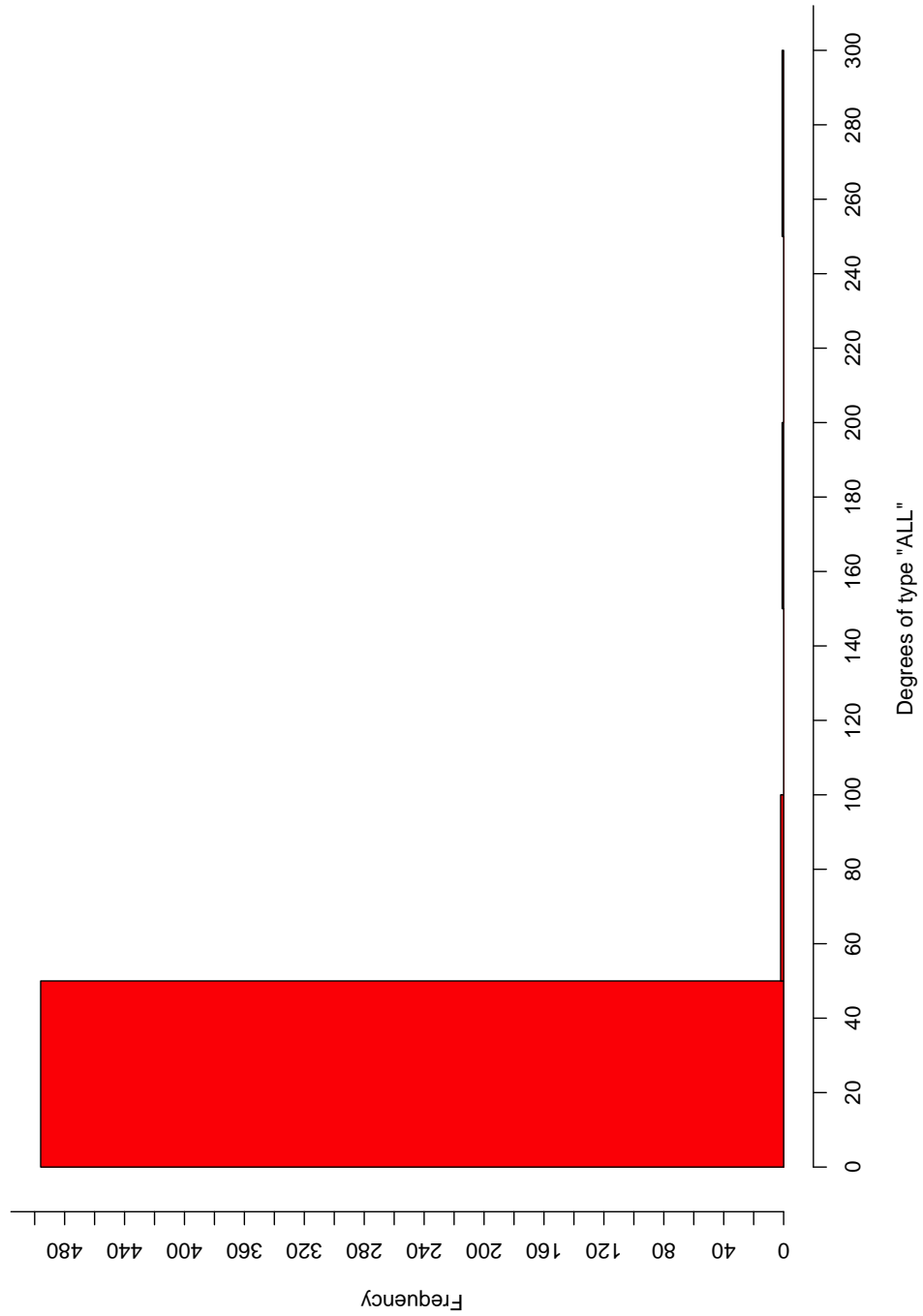


Figure 149. Degree histogram for $n * \gamma$, always using a random first WO.

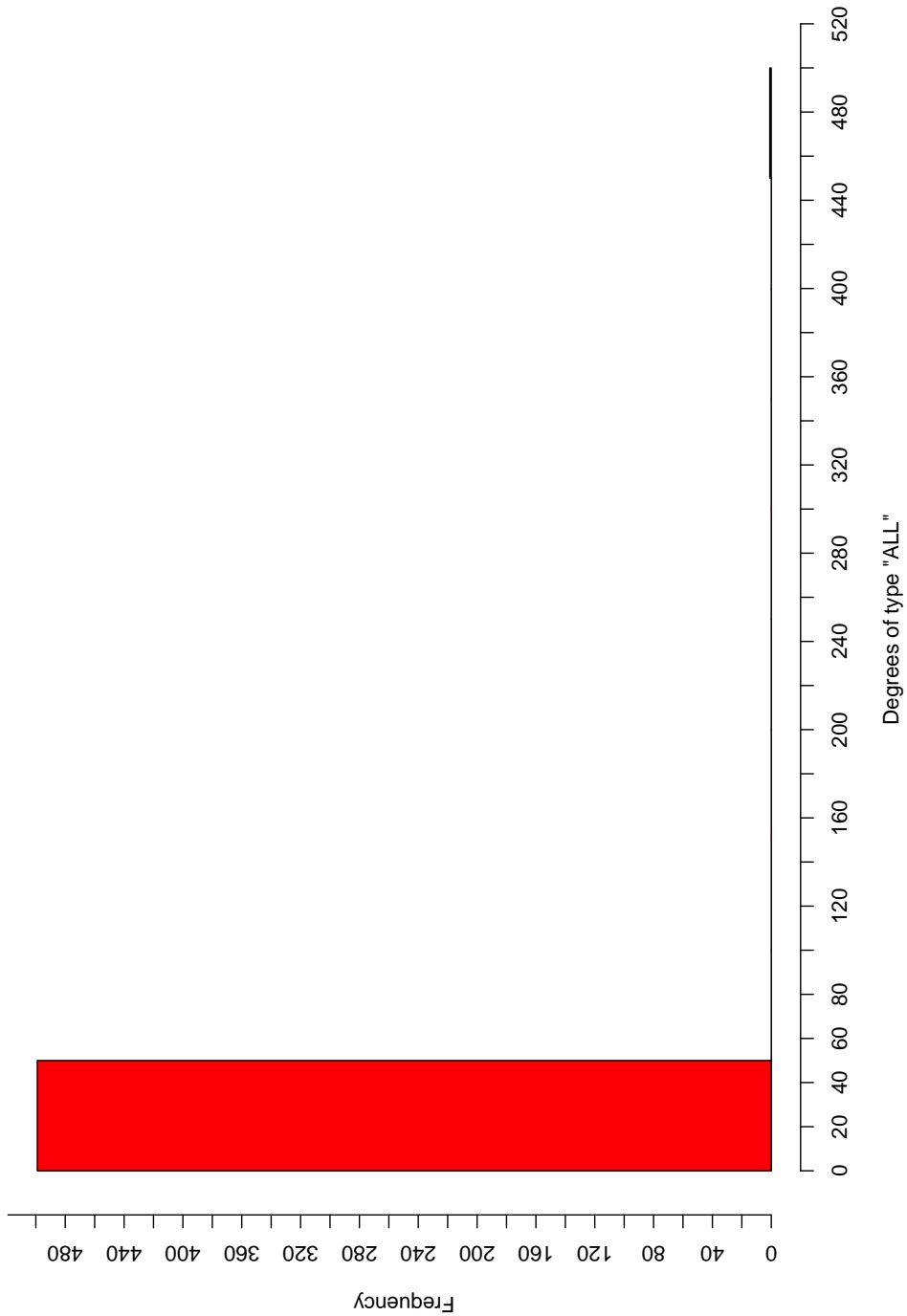


Figure 150. Degree histogram for $max(1, \ln(n * \gamma))$, always using a random first WO.

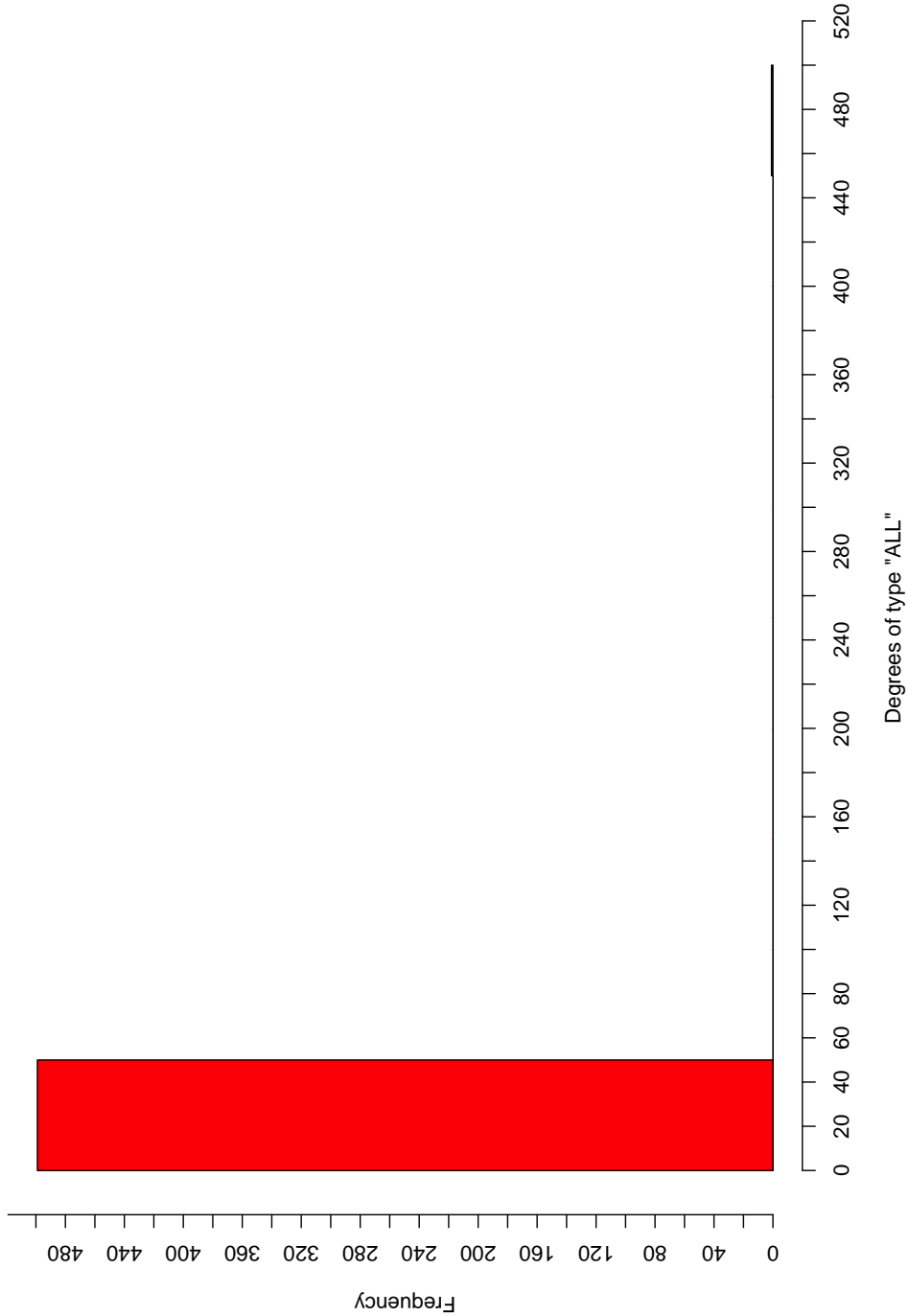


Figure 151. Degree histogram for $max(1, \ln(n) * \gamma)$, always using a random first WO.

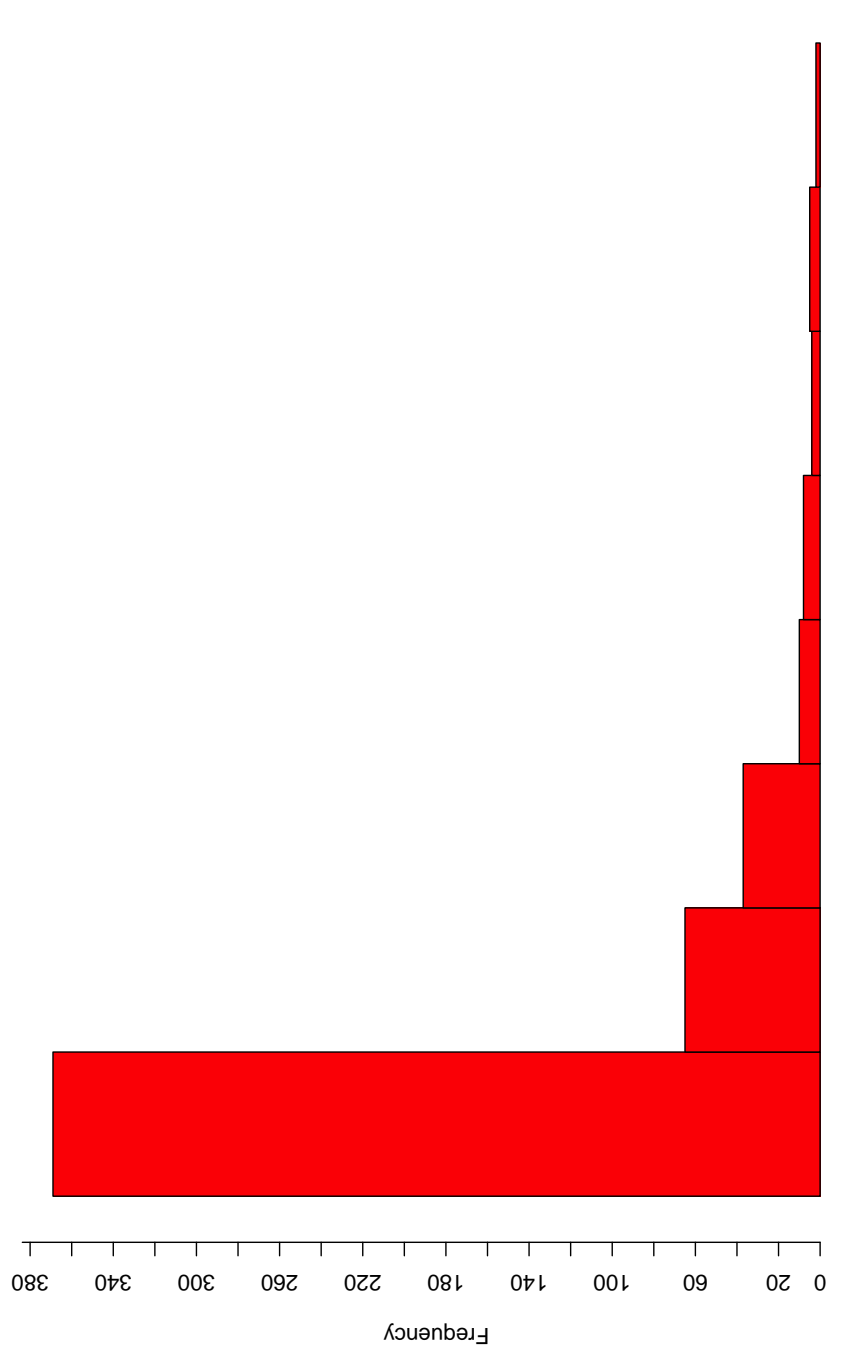


Figure 152. Degree histogram for $max(0, \ln(n * \gamma))$, always using a random first WO.

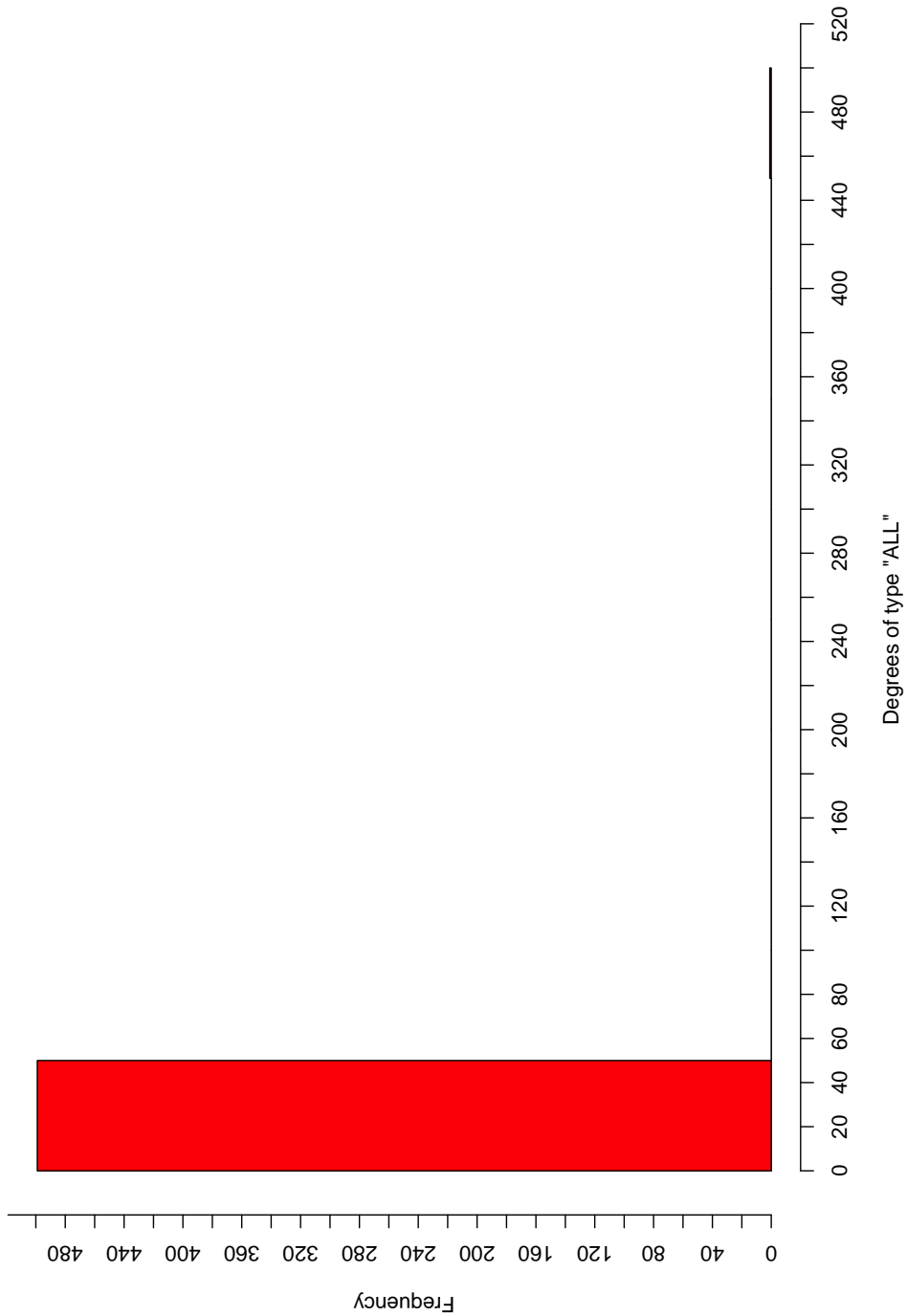


Figure 153. Degree histogram for $max(0, \ln(n) * \gamma)$, always using a random first WO.

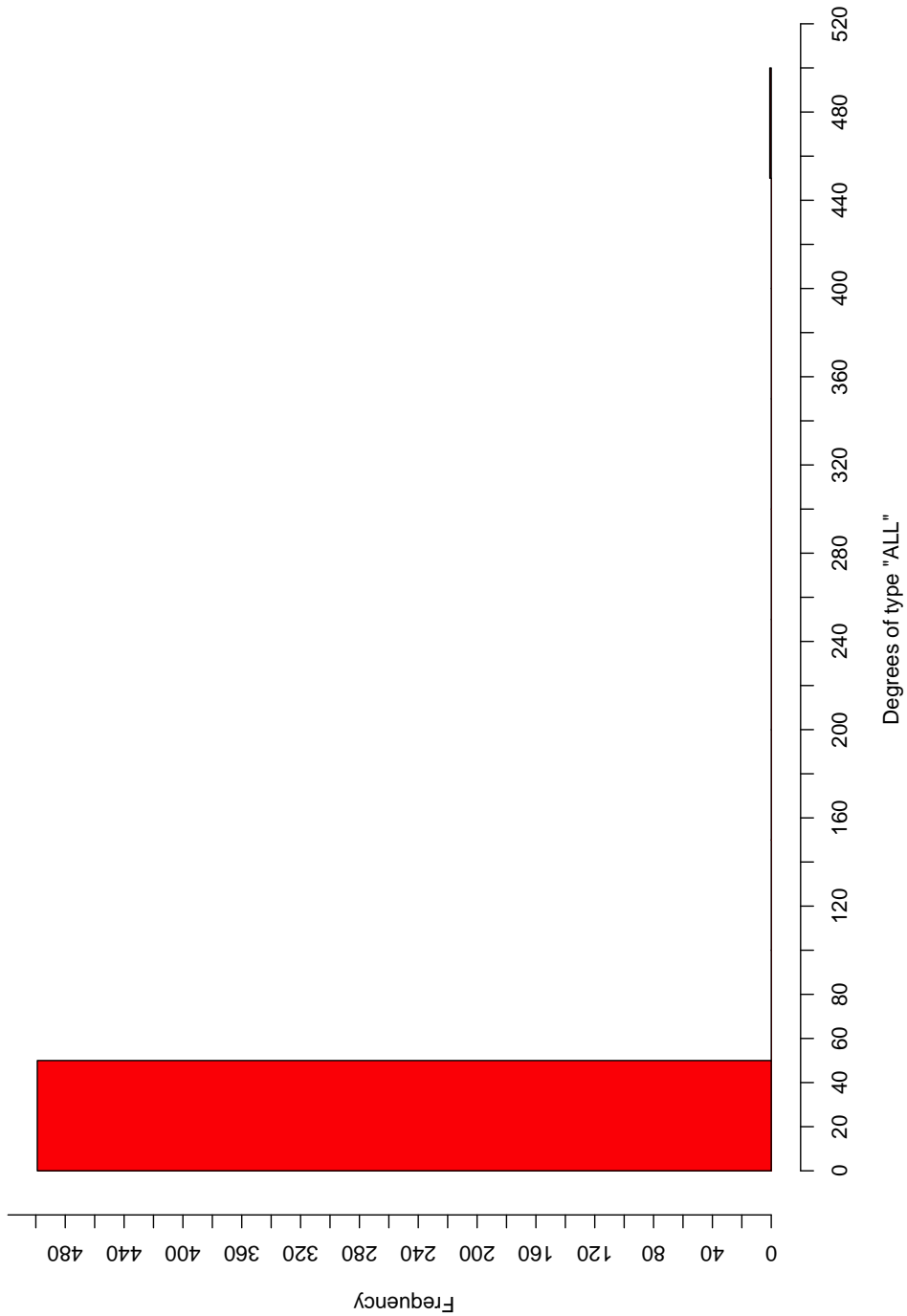


Figure 154. Degree histogram for $max(1, \log_2(n * \gamma))$, always using a random first WO.

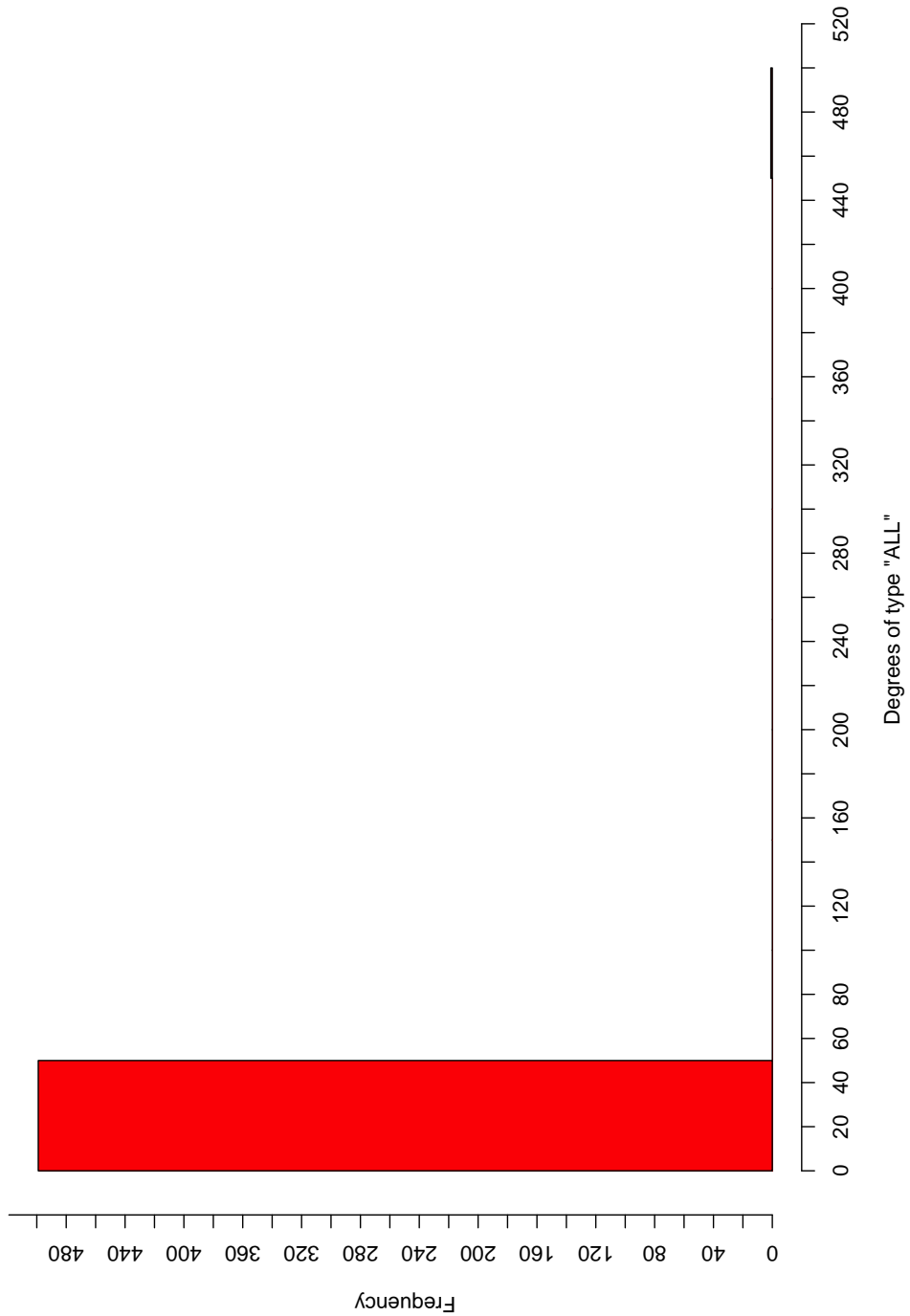


Figure 155. Degree histogram for $max(1, \log_2(n) * \gamma)$, always using a random first WO.

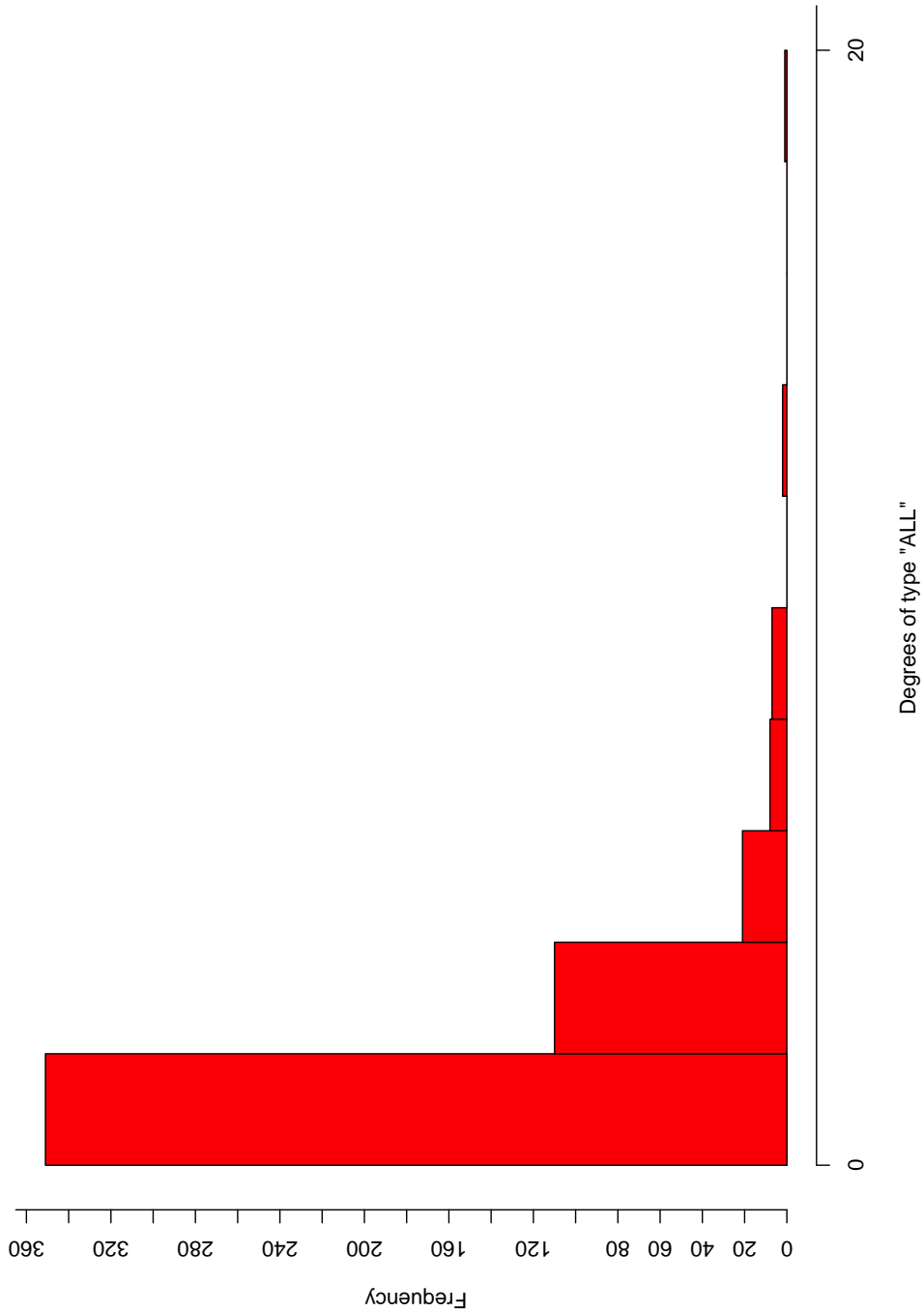


Figure 156. Degree histogram for $max(0, \log_2(n * \gamma))$, always using a random first WO.

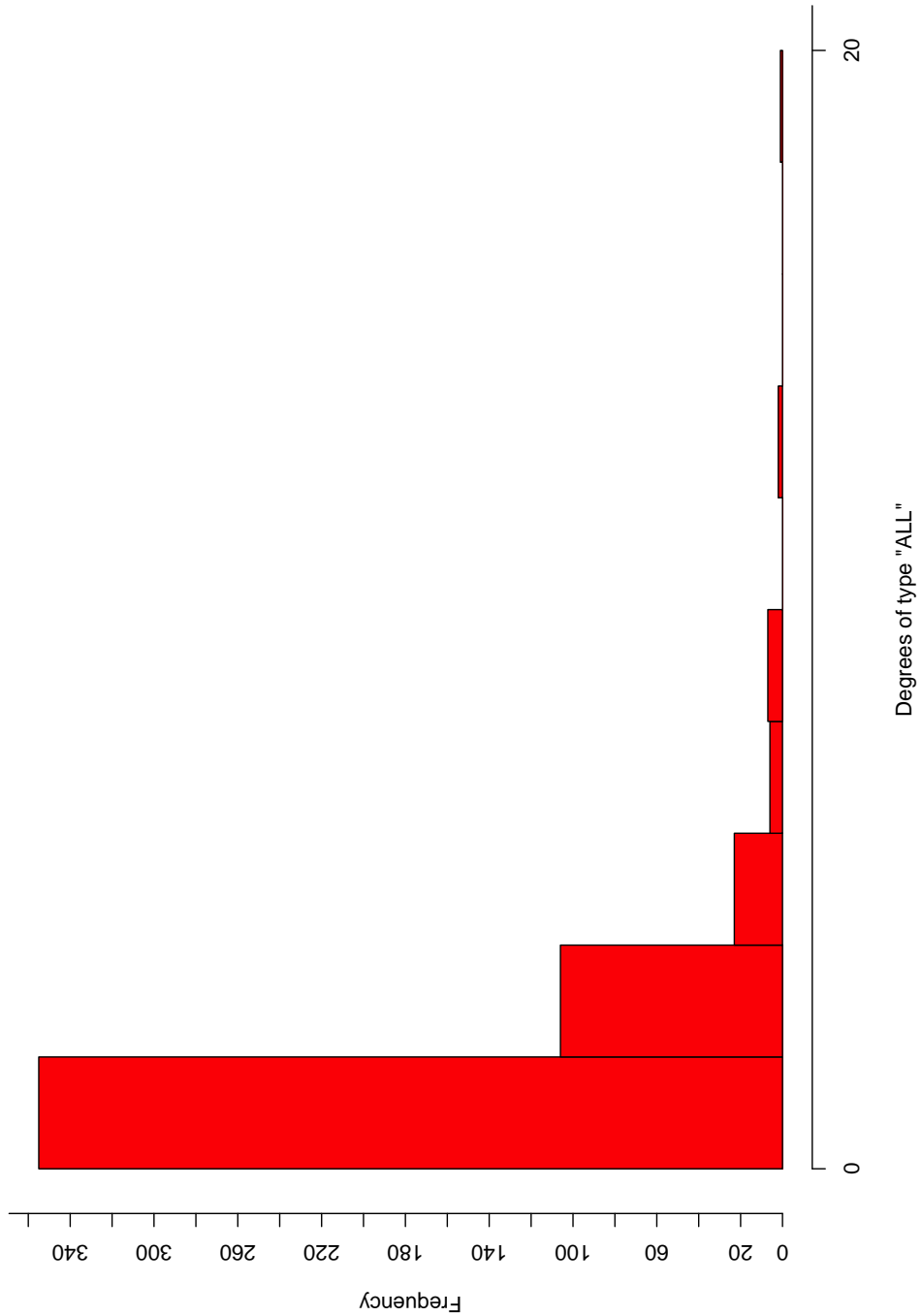


Figure 157. Degree histogram for $max(0, \log_2(n) * \gamma)$, always using a random first WO.

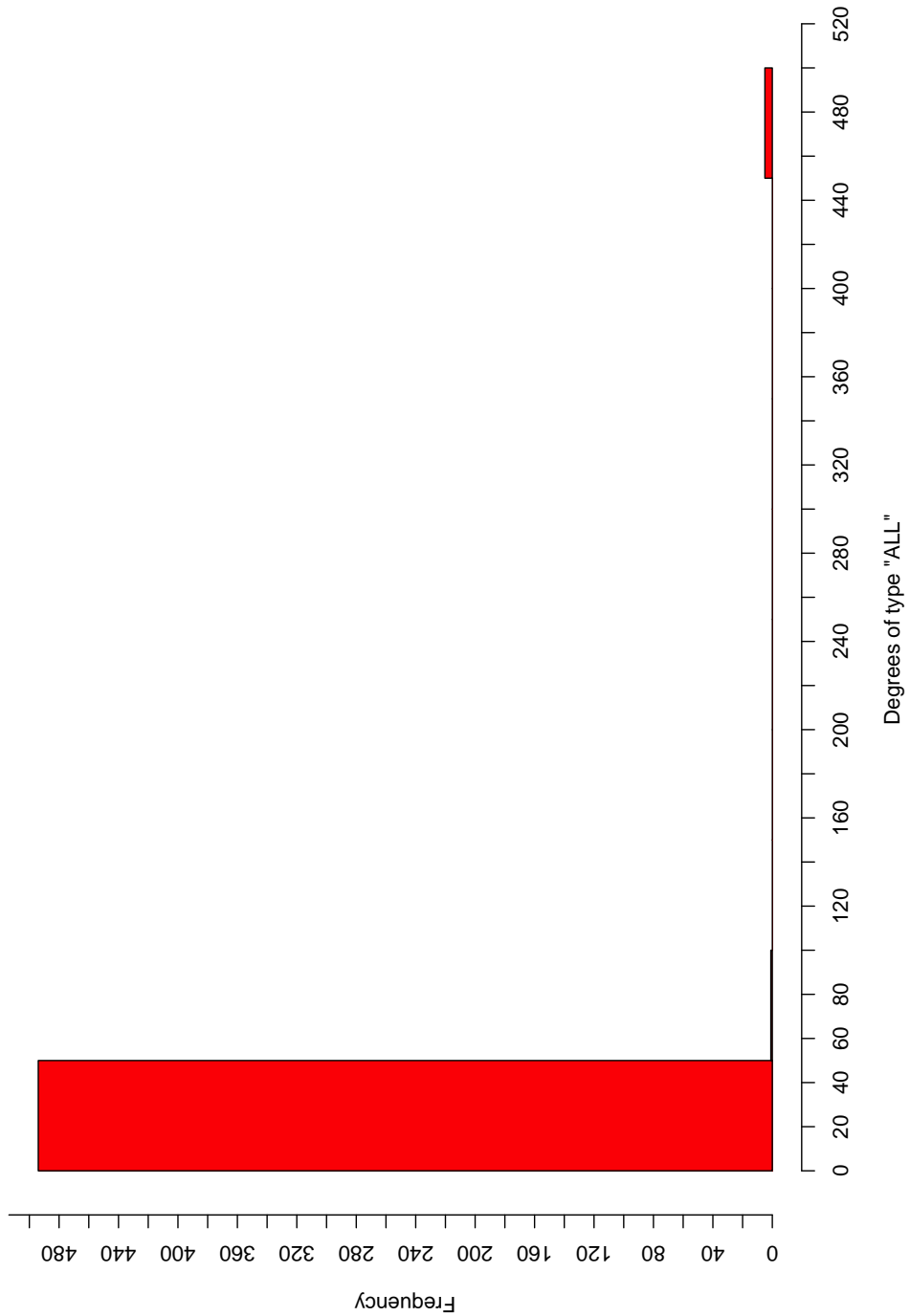


Figure 158. Degree histogram for $5 + \log_2(n * \gamma)$, always using a random first WO.

F.4 HISTOGRAMS BASED RESULTING FROM USING LAST ADDED WO

A series of USW simulated graphs were created using default values for all parameters (see Section C.1 on page 368), except for:

1. USW graph order $n=500$
2. First WO selection:
 - (a) Always using the last WO that was added to the USW graph.
3. Different ways to decide how many connections to make:
 - (a) $n * \gamma$
 - (b) $\max(1, \ln(n * \gamma))$
 - (c) $\max(1, \ln(n) * \gamma)$
 - (d) $\max(0, \ln(n * \gamma))$
 - (e) $\max(0, \ln(n) * \gamma)$
 - (f) $\max(1, \log_2(n * \gamma))$
 - (g) $\max(1, \log_2(n) * \gamma)$
 - (h) $\max(0, \log_2(n * \gamma))$
 - (i) $\max(0, \log_2(n) * \gamma)$
 - (j) $5 + \log_2(n * \gamma)$

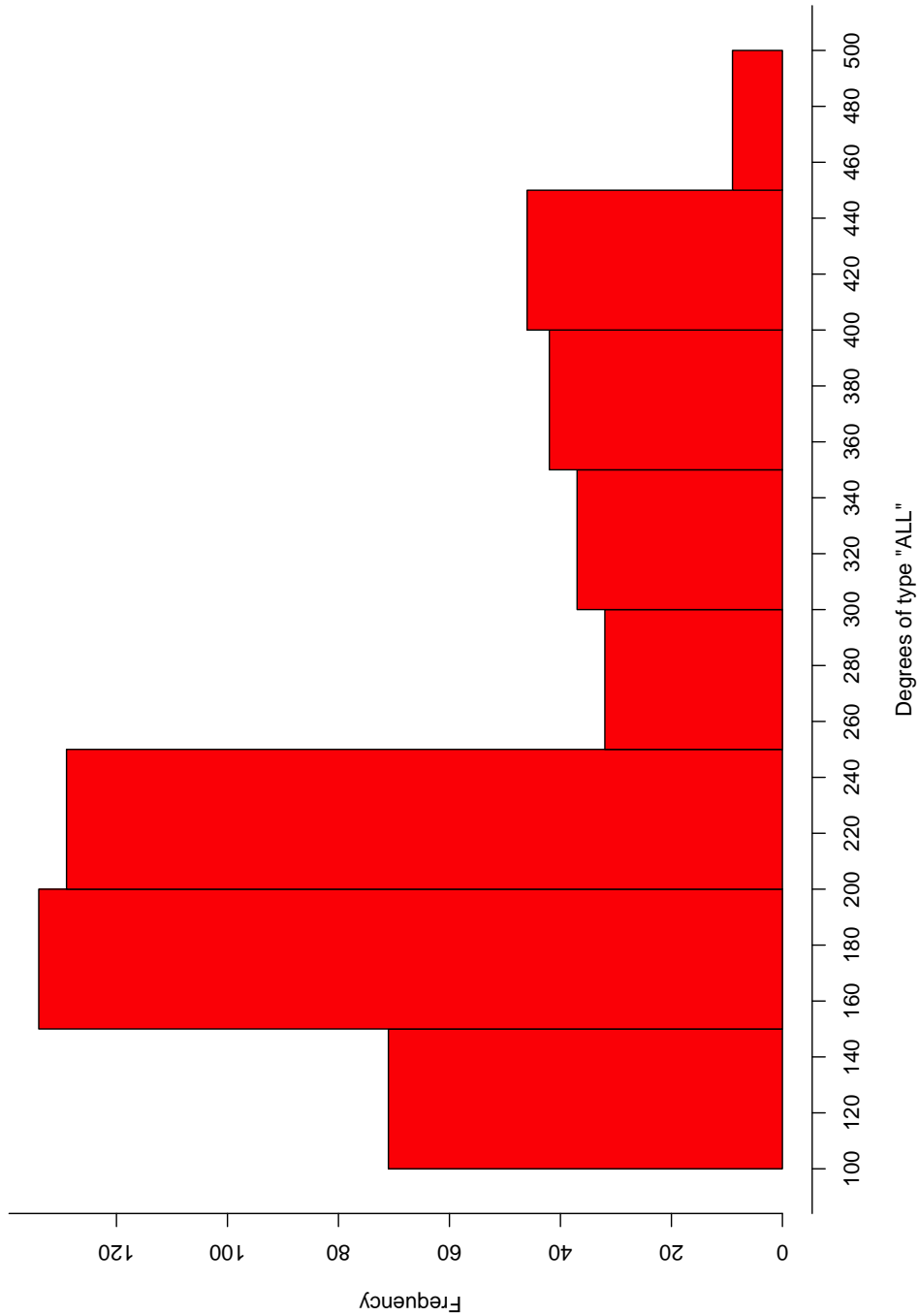


Figure 159. Degree histogram for $n * \gamma$, always using last added WO.

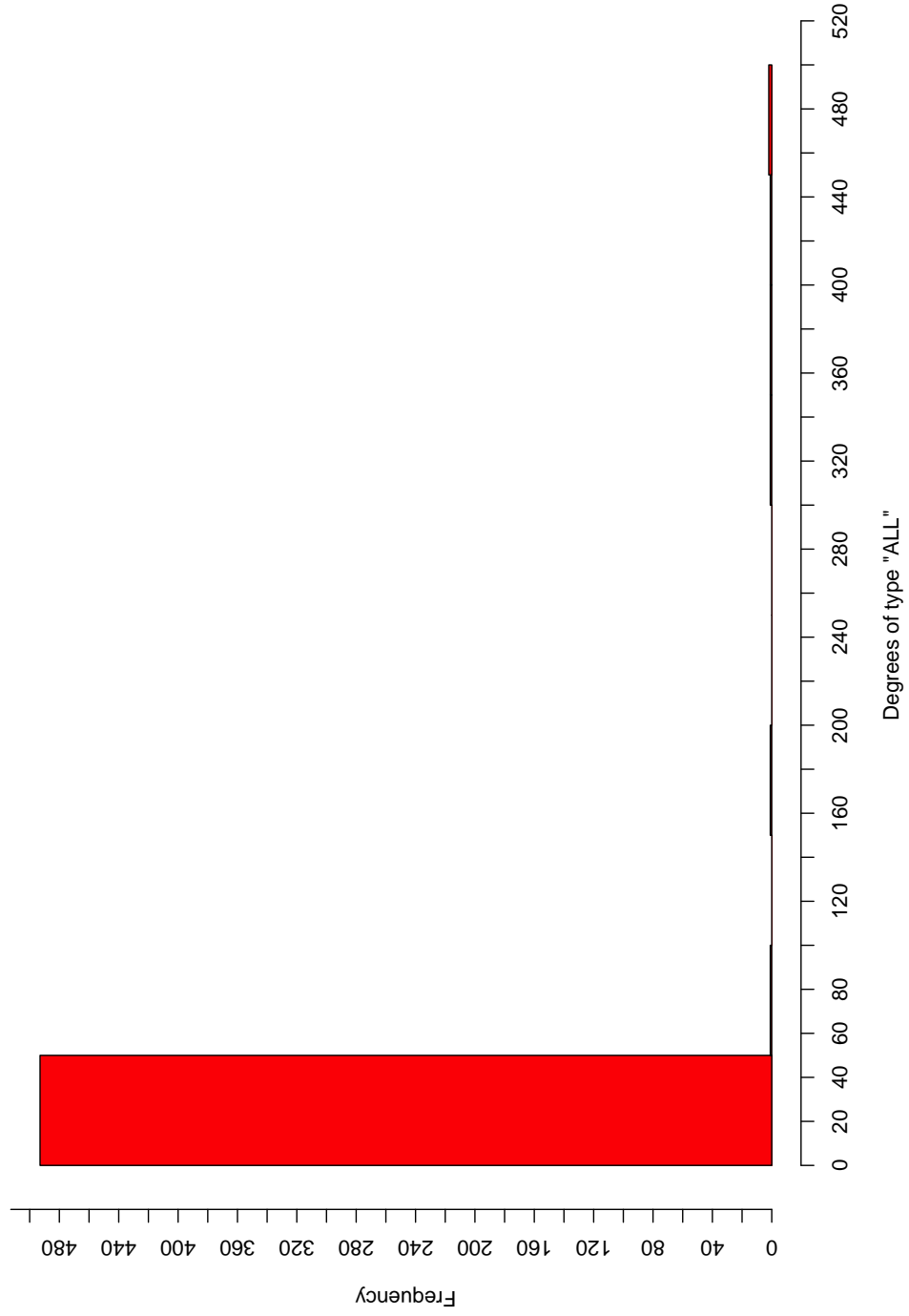


Figure 160. Degree histogram for $max(1, \ln(n * \gamma))$, always using using last added WO.

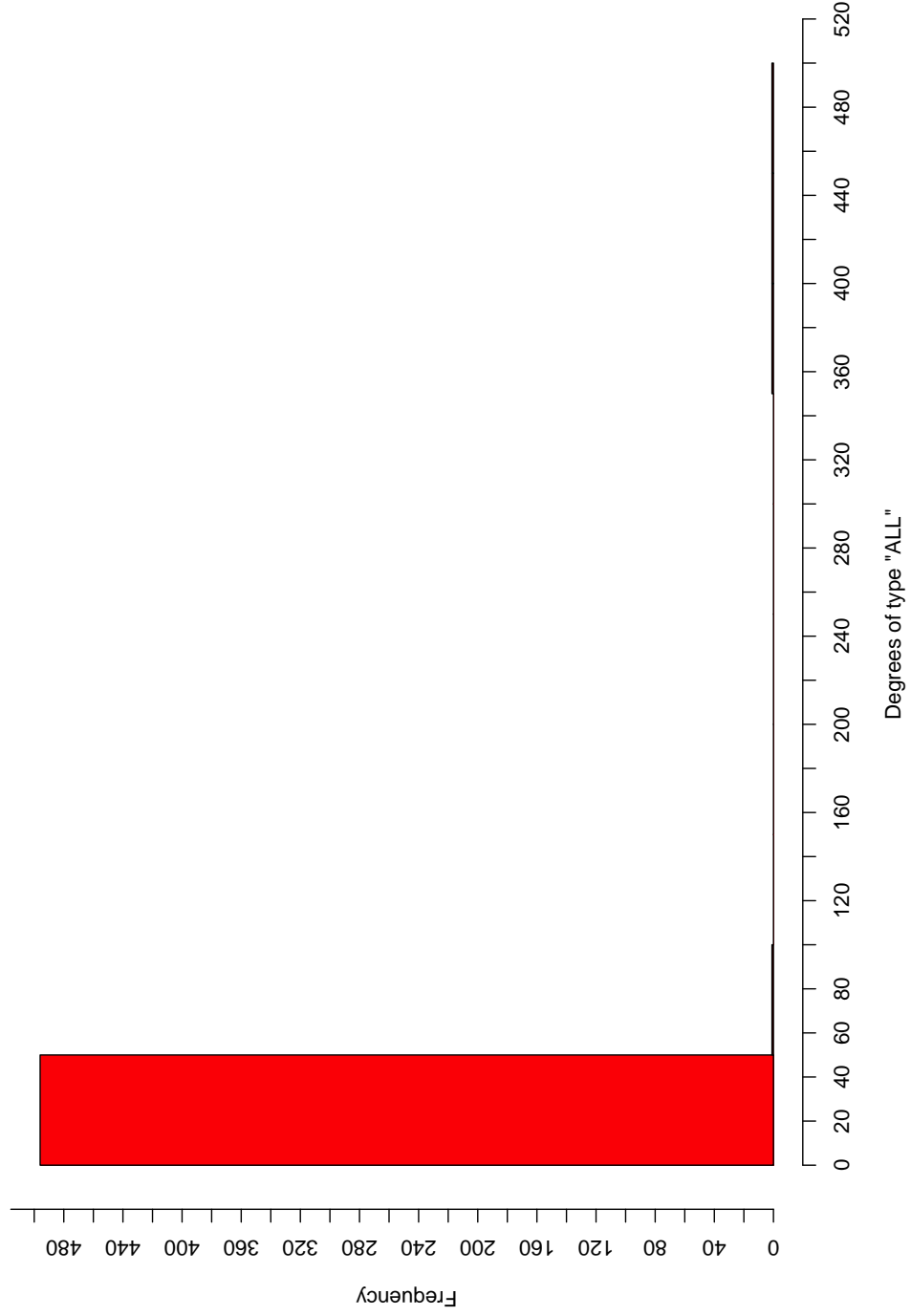


Figure 161. Degree histogram for $max(1, \ln(n) * \gamma)$, always using using last added WO.

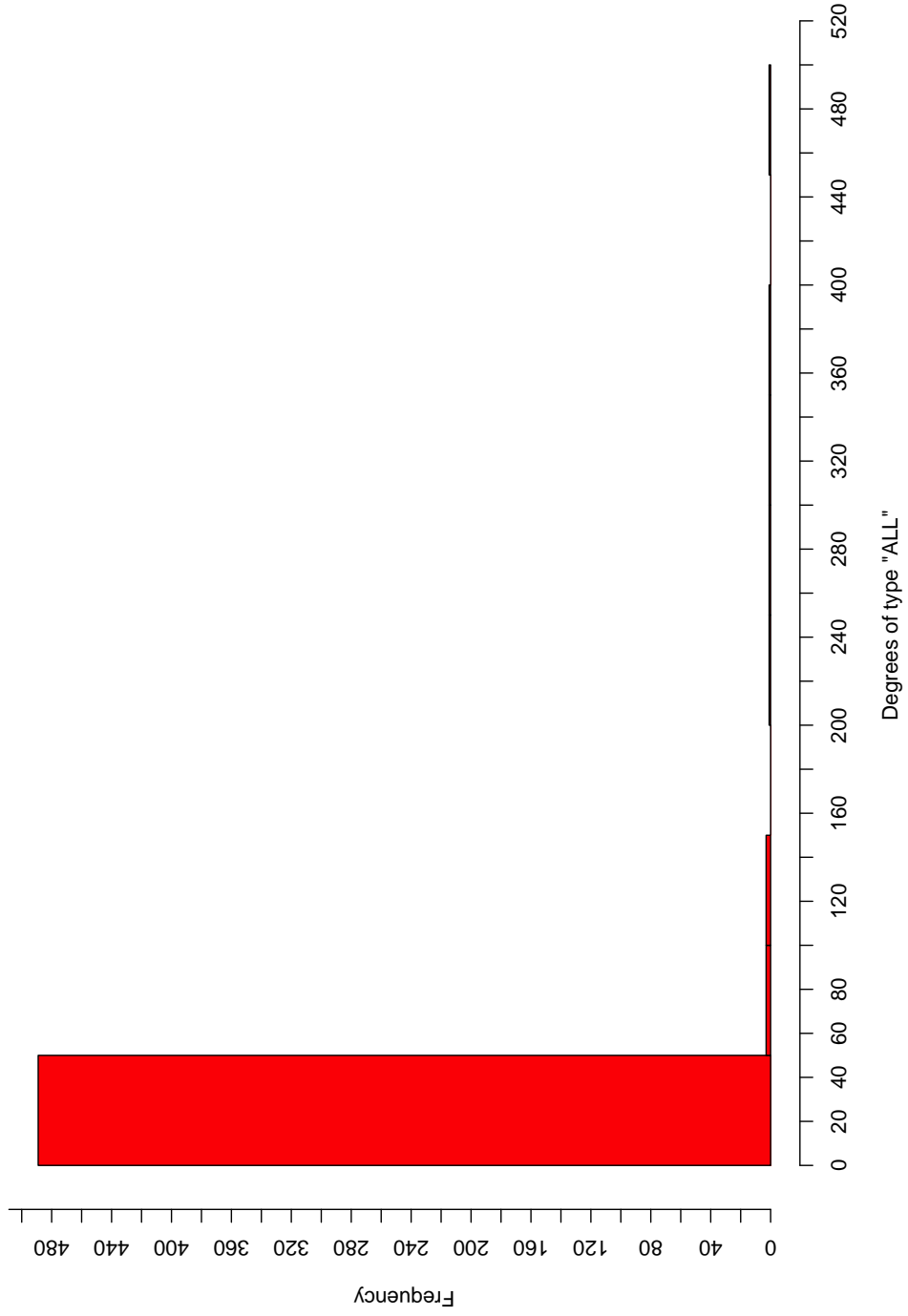


Figure 162. Degree histogram for $\max(0, \ln(n * \gamma))$, always using using last added WO.

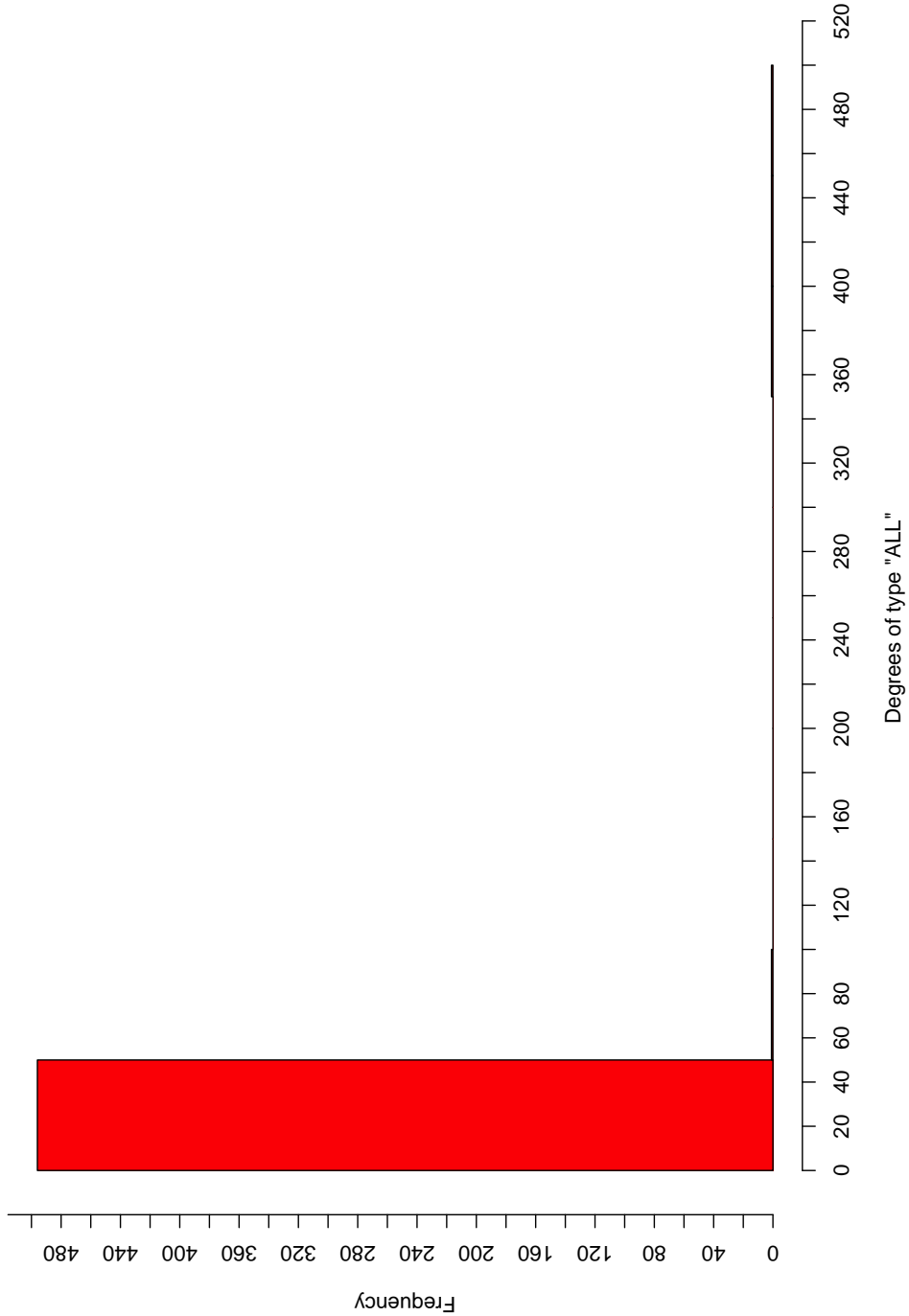


Figure 163. Degree histogram for $max(0, \ln(n) * \gamma)$, always using using last added WO.

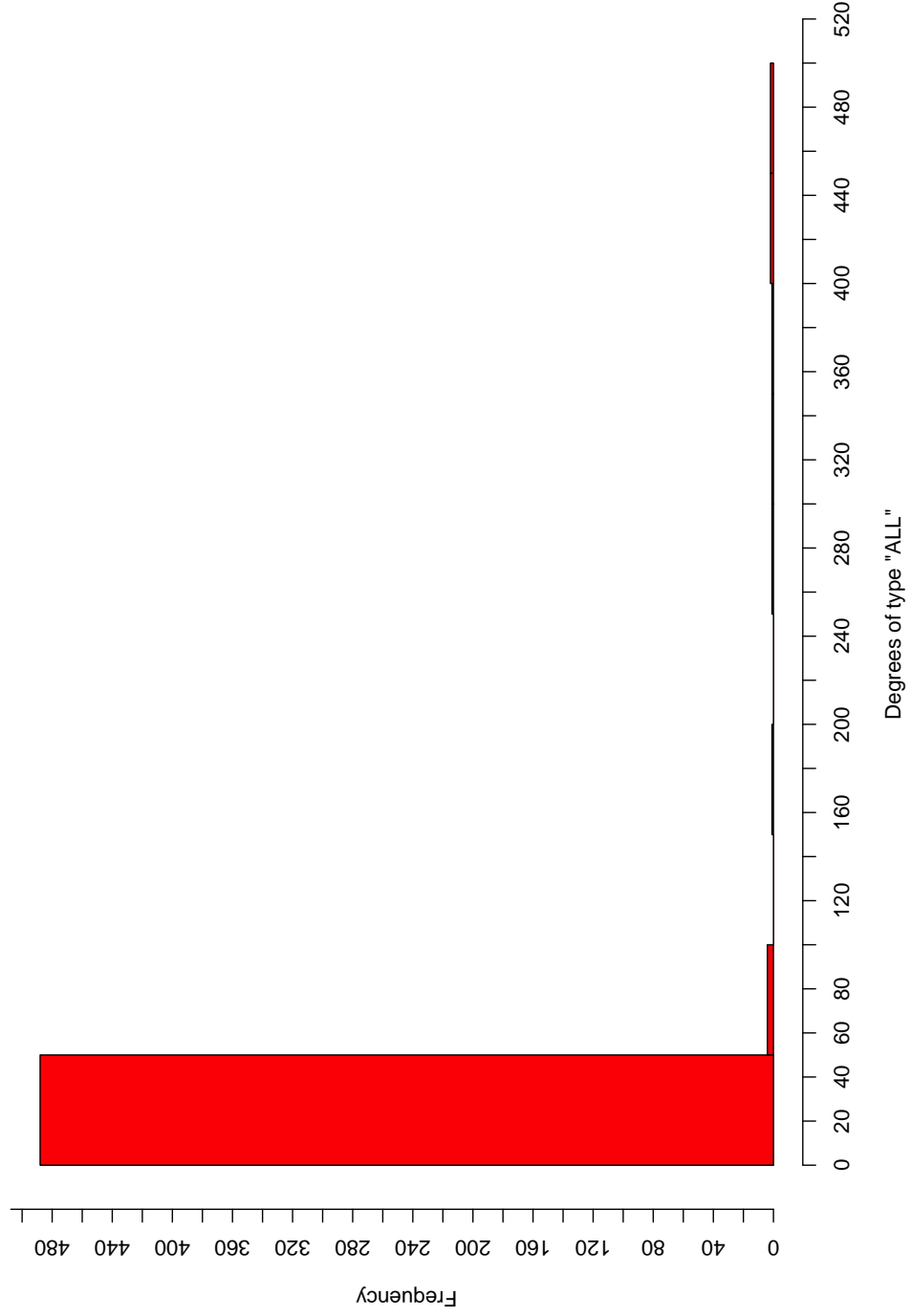


Figure 164. Degree histogram for $\max(1, \log_2(n * \gamma))$, always using using last added WO.

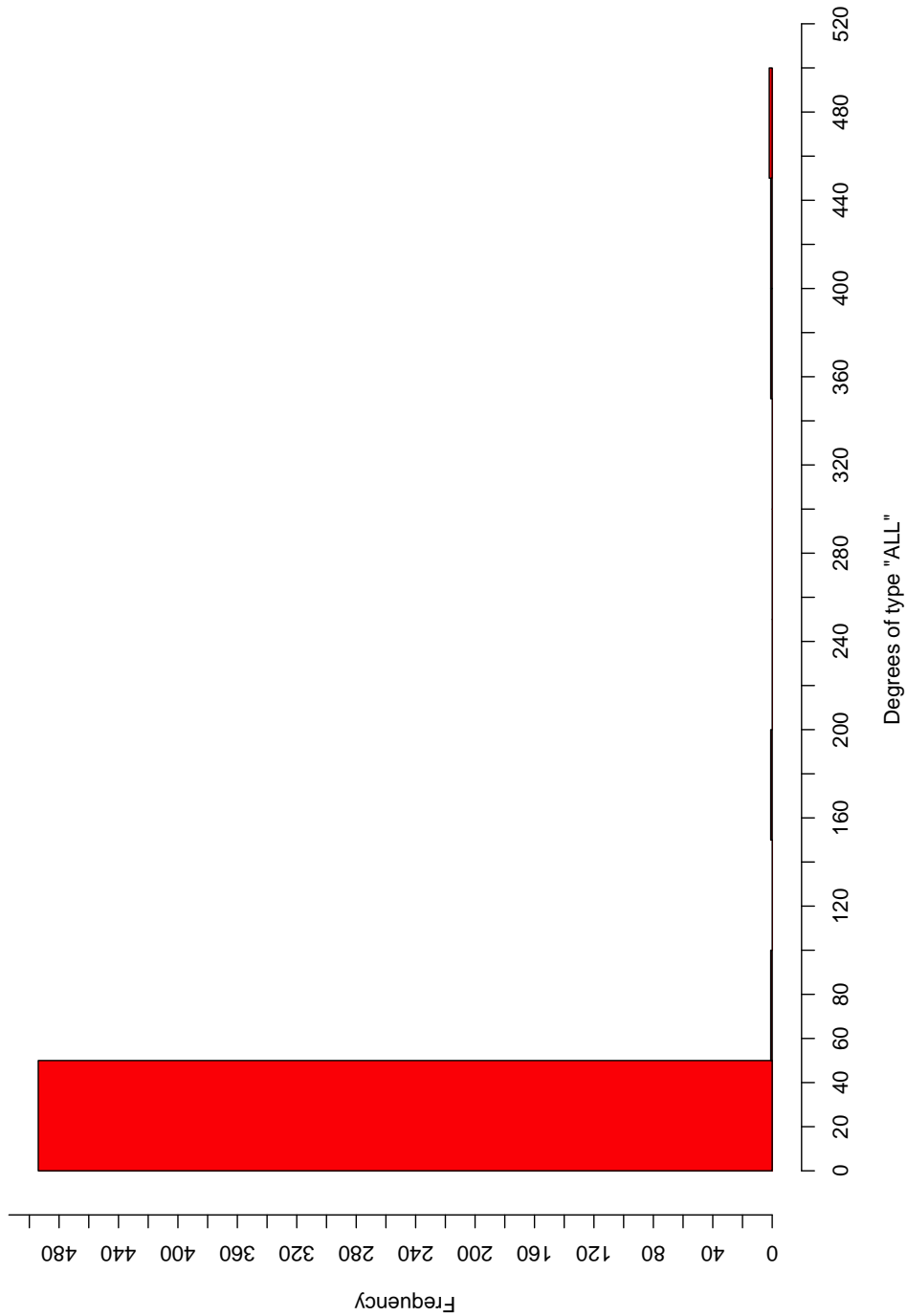


Figure 165. Degree histogram for $\max(1, \log_2(n) * \gamma)$, always using last added WO.

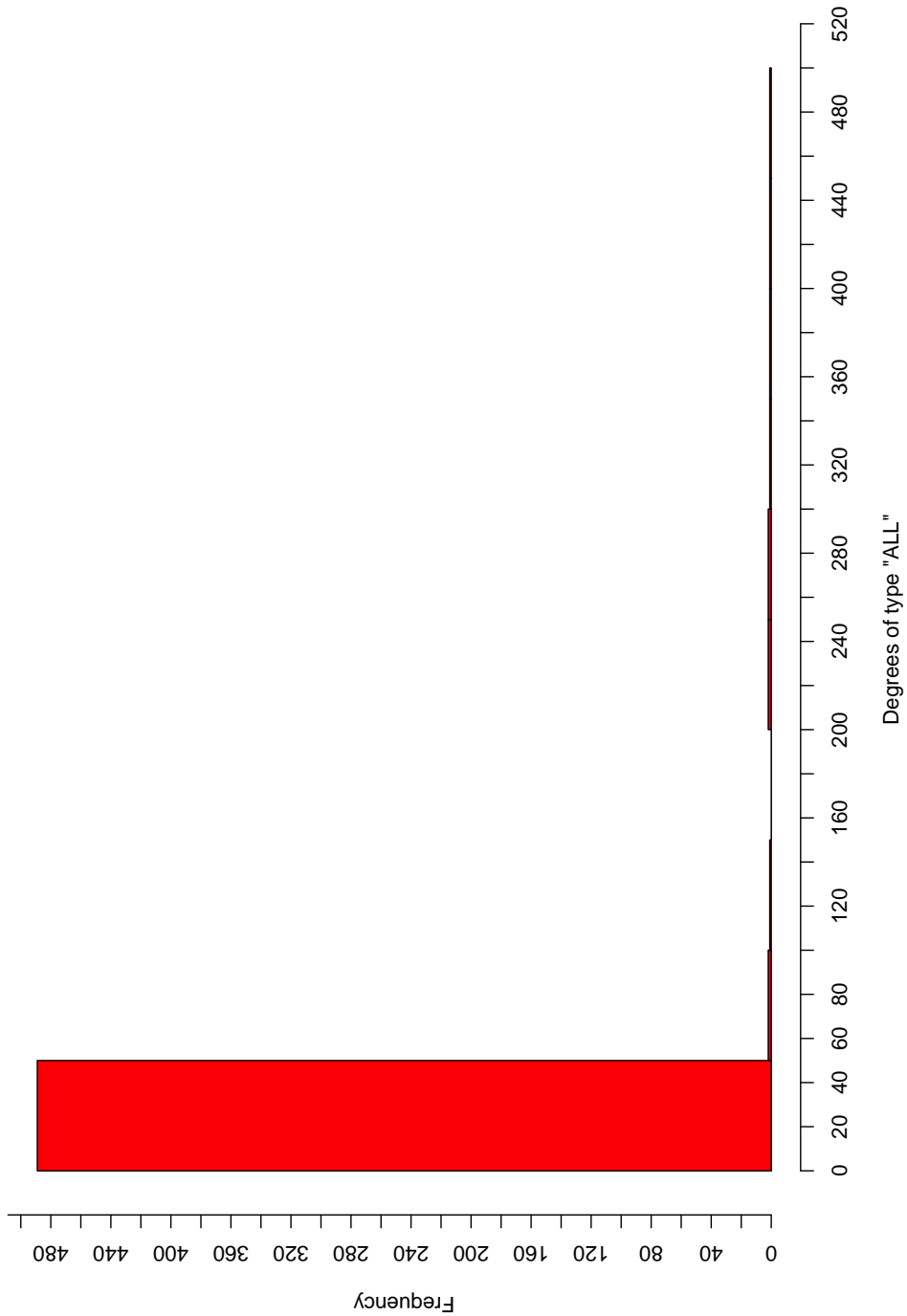


Figure 166. Degree histogram for $max(0, \log_2(n * \gamma))$, always using last added WO.

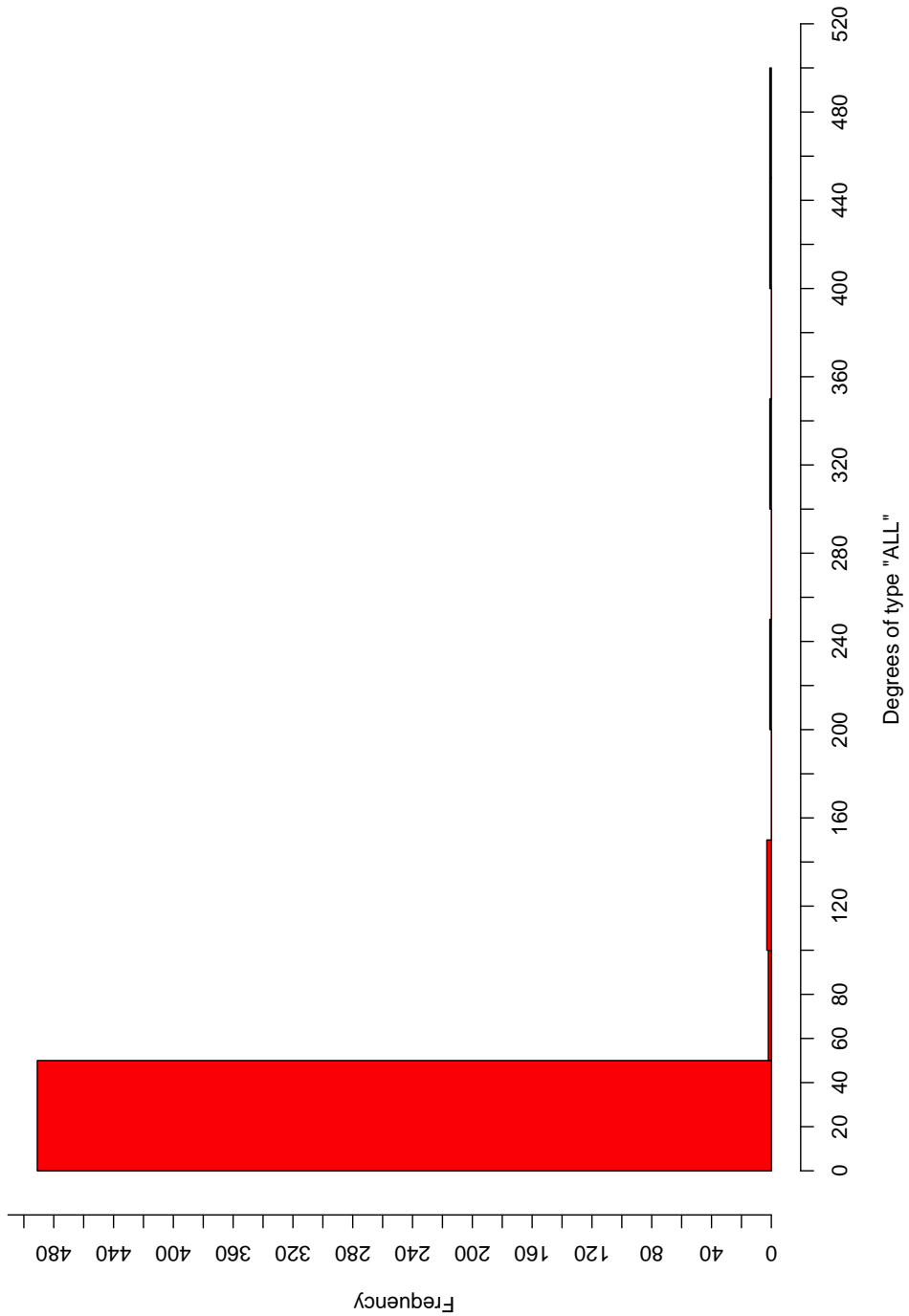


Figure 167. Degree histogram for $max(0, \log_2(n) * \gamma)$, always using last added WO.

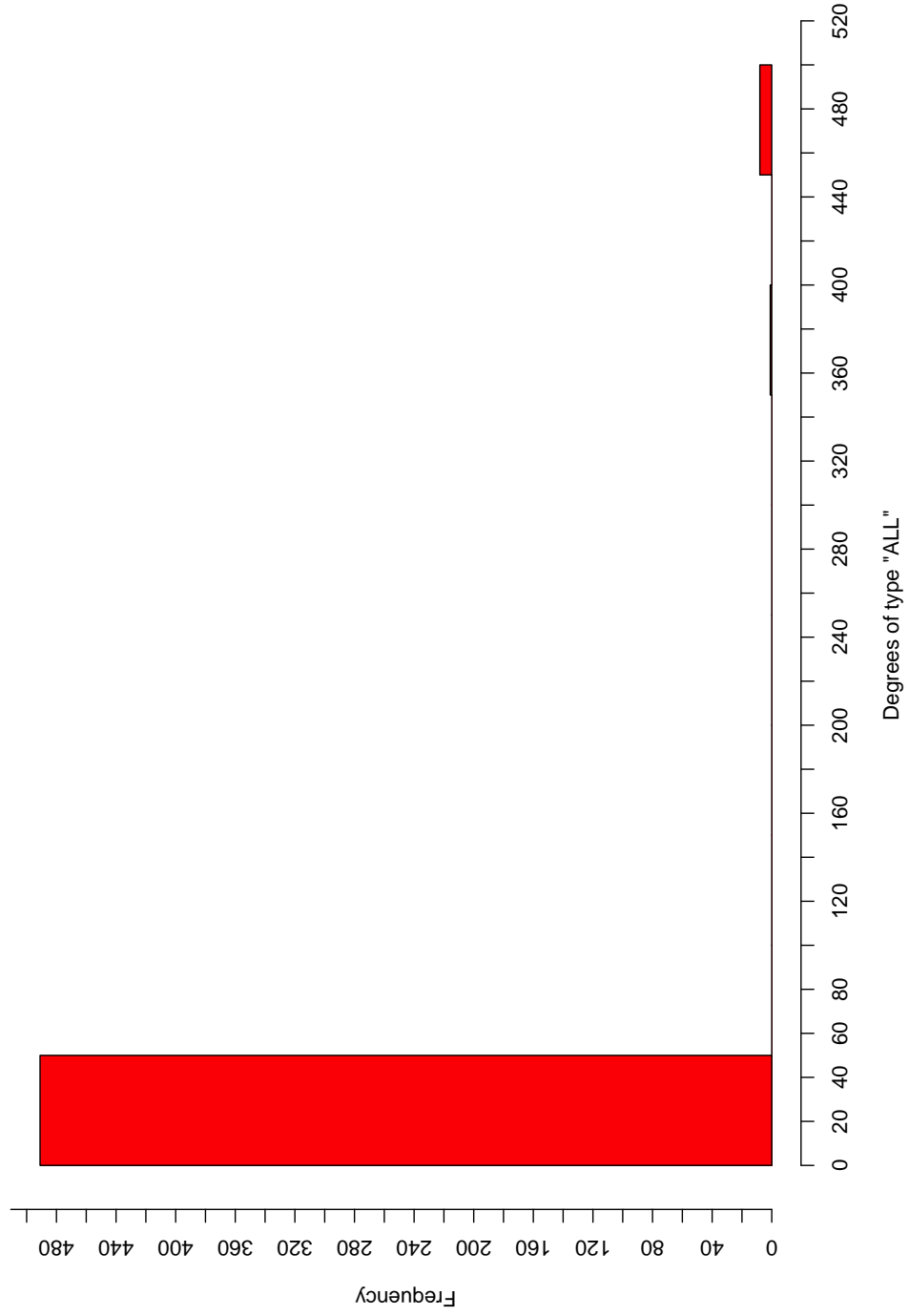


Figure 168. Degree histogram for $5 + \log_2(n * \gamma)$, always using using last added WO.

APPENDIX G

MESSAGE PROPAGATION FIGURES

We document the time the it takes for a message to propagate through a set of graphs based on different USW values for β and γ .

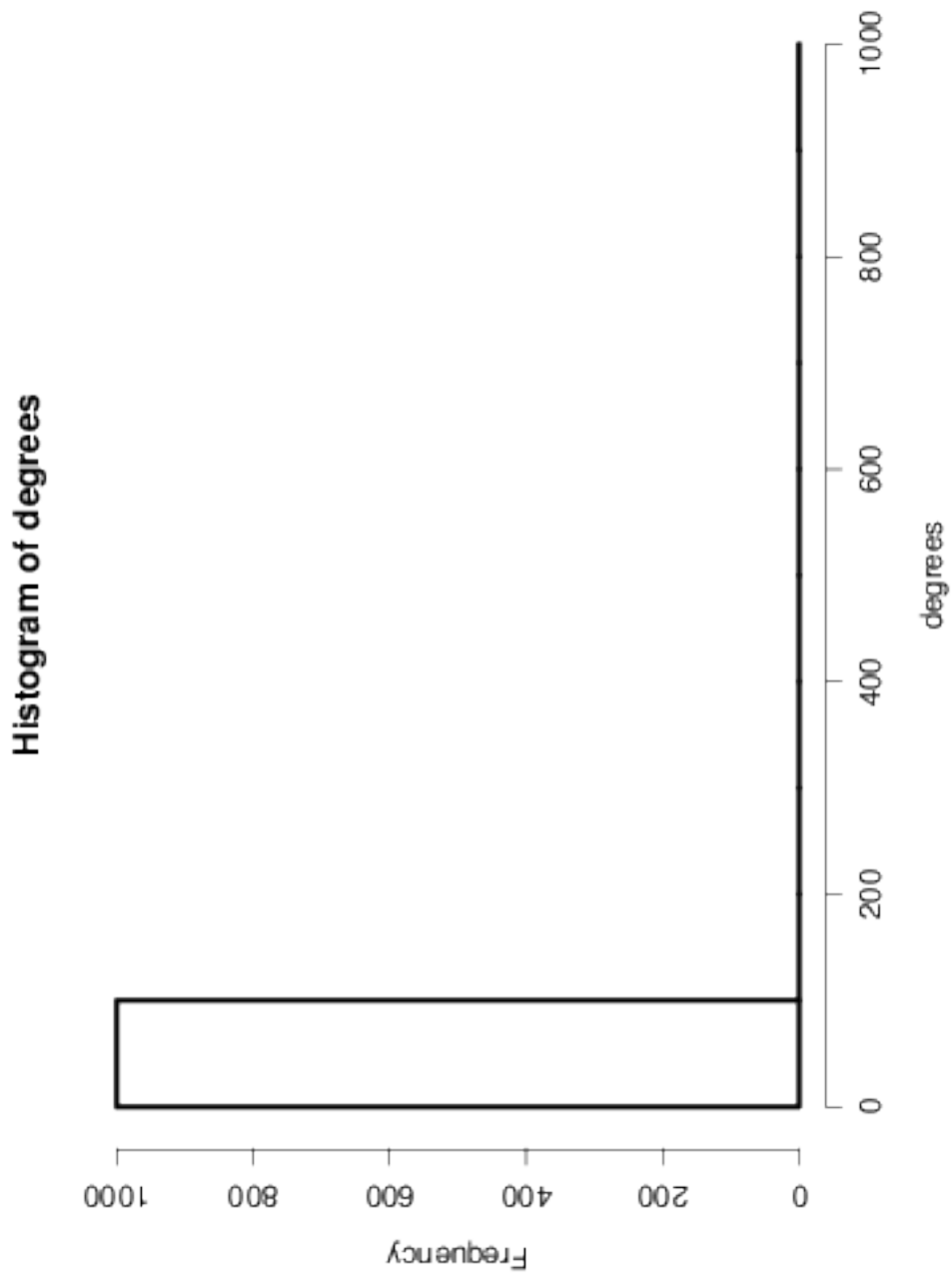


Figure 169. Degree distribution, size = 1000 $\beta = 0.0$ $\gamma = 0.0$

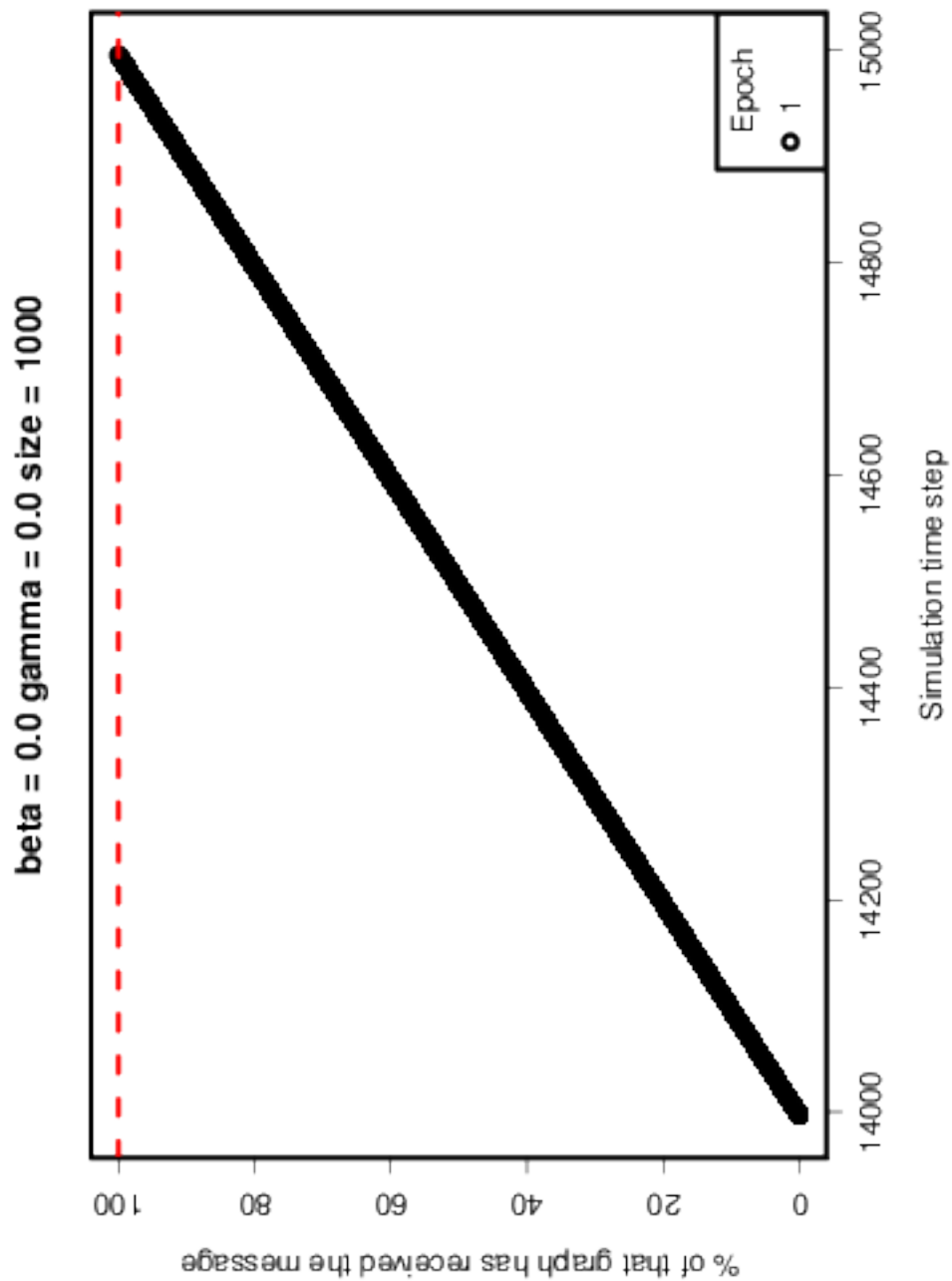


Figure 170. Sequential, size = 1000 $\beta = 0.0$ $\gamma = 0.0$

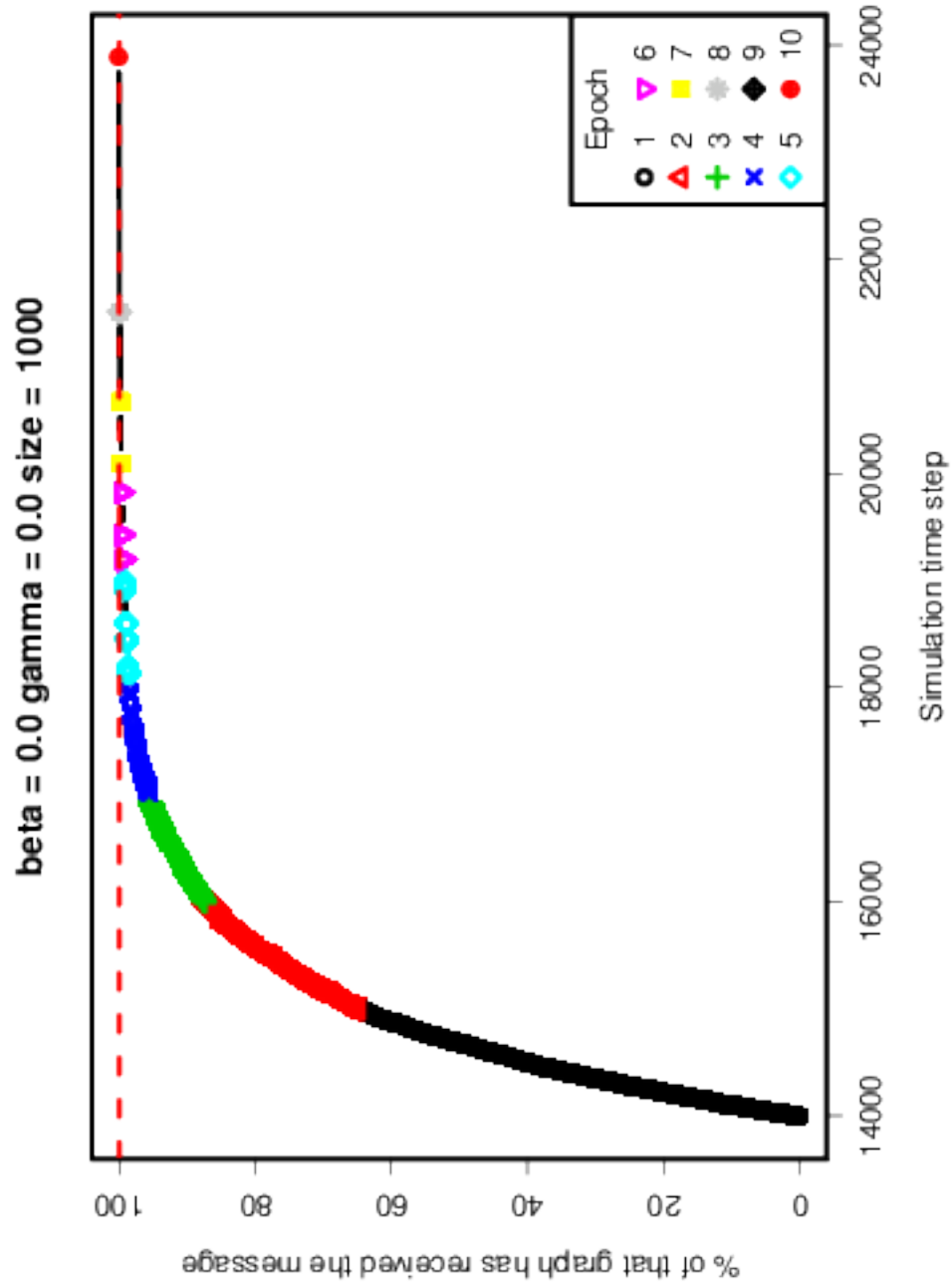


Figure 171. Random selection, size = 1000 $\beta = 0.0$ $\gamma = 0.0$

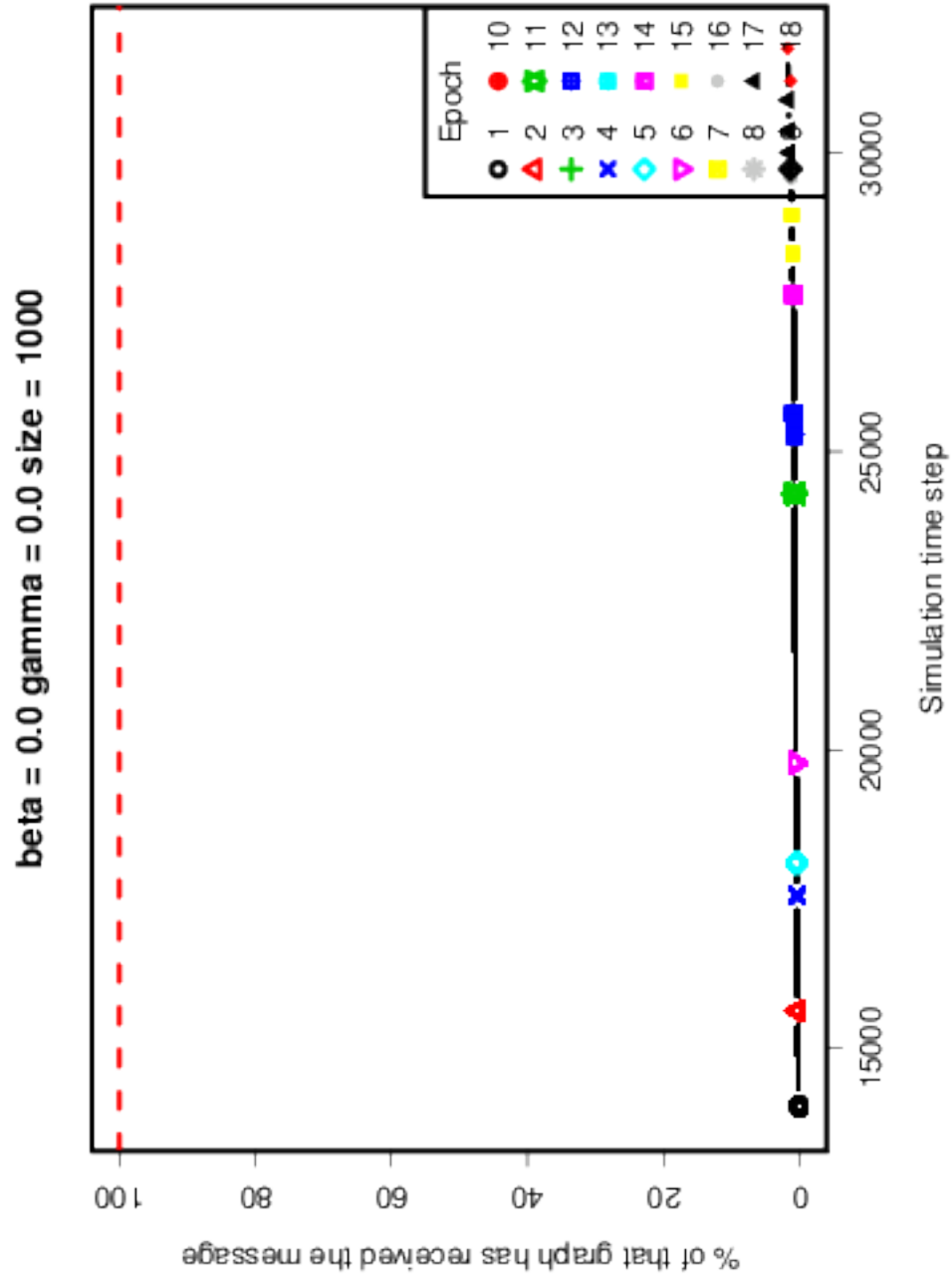


Figure 172. Degree biased selection, size = 1000 $\beta = 0.0$ $\gamma = 0.0$

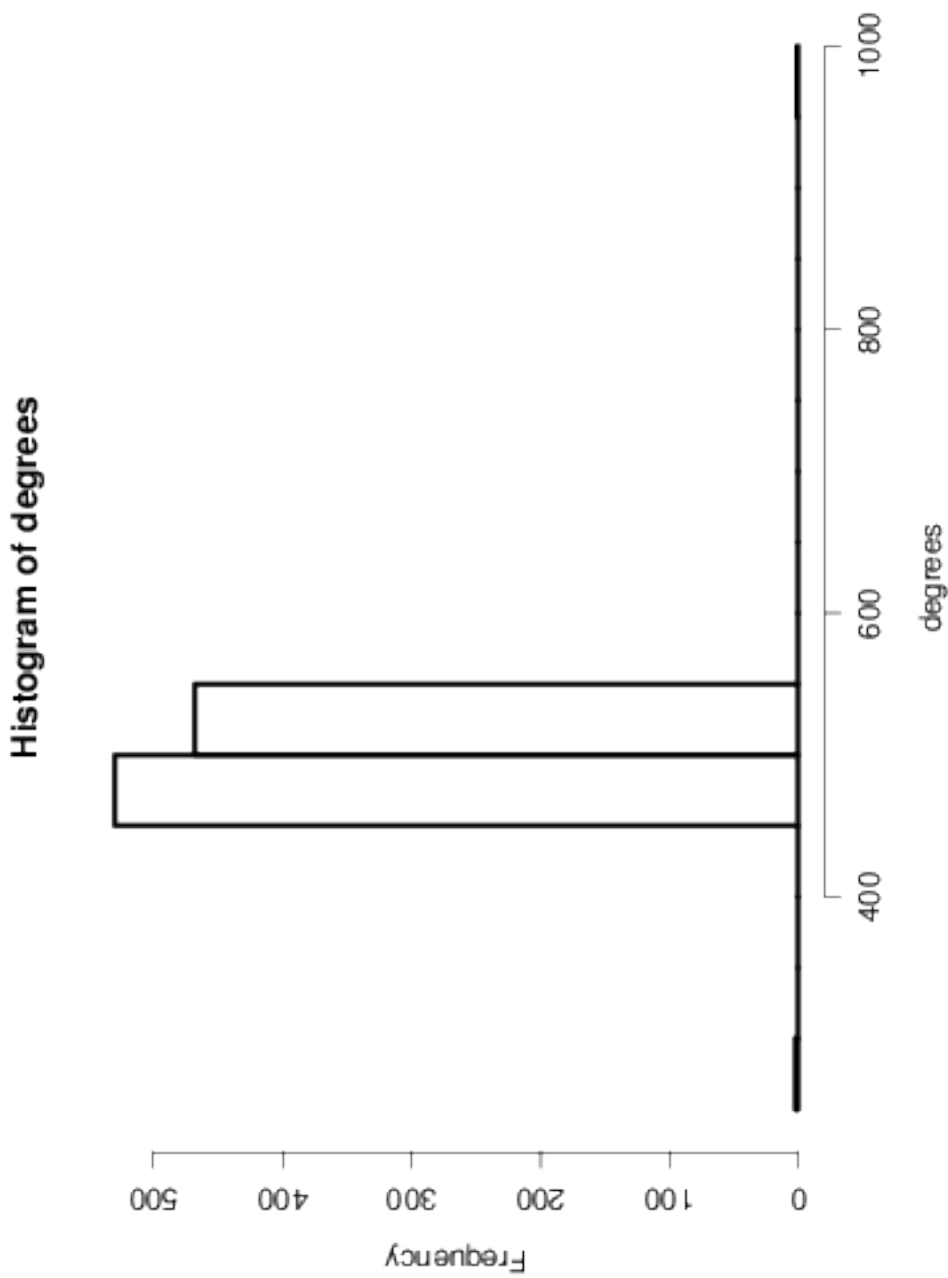


Figure 173. Degree distribution, size = 1000 $\beta = 0.0$ $\gamma = 0.5$

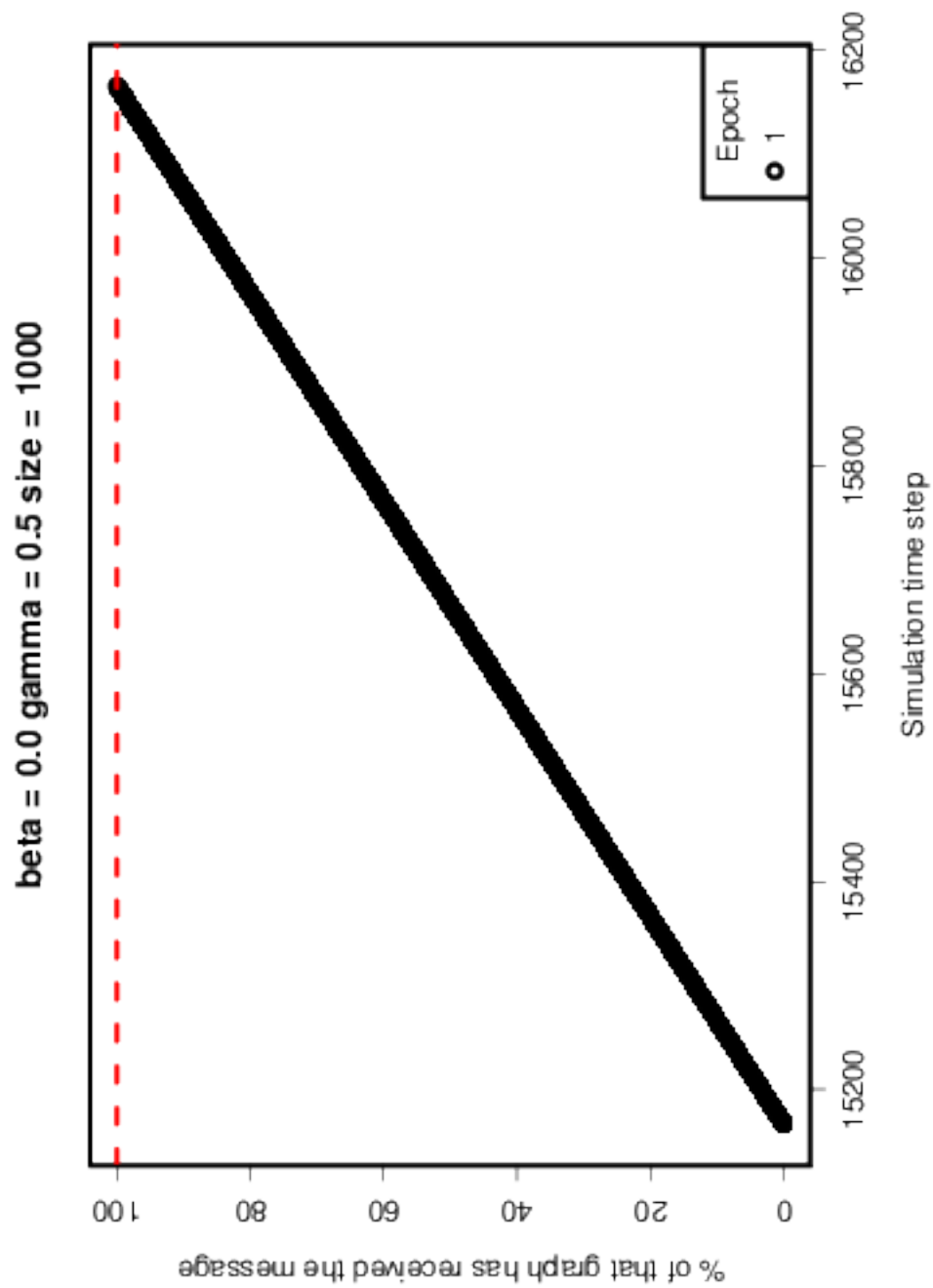


Figure 174. Sequential, size = 1000 $\beta = 0.0$ $\gamma = 0.5$

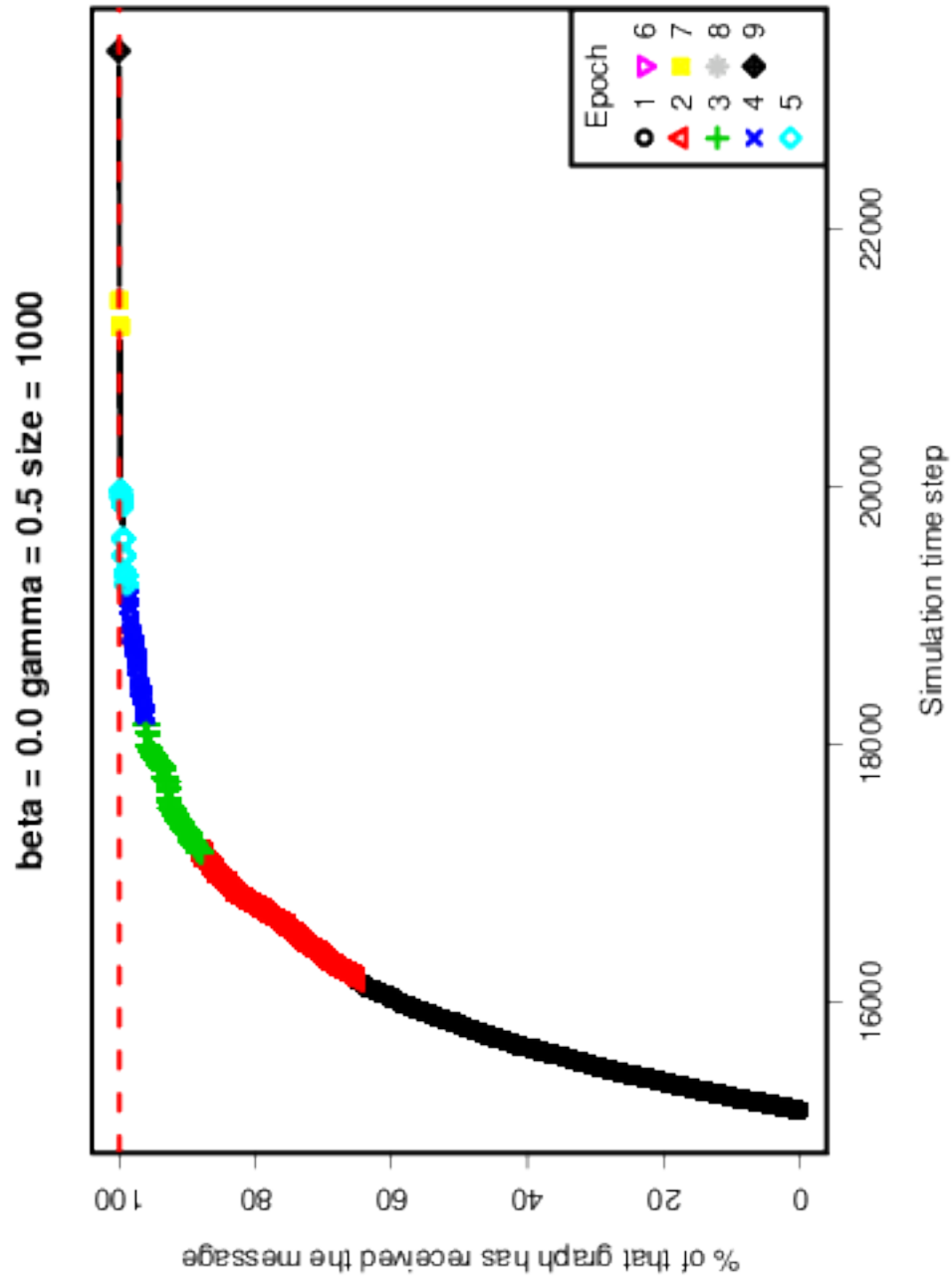


Figure 175. Random selection, size = 1000 $\beta = 0.0$ $\gamma = 0.5$

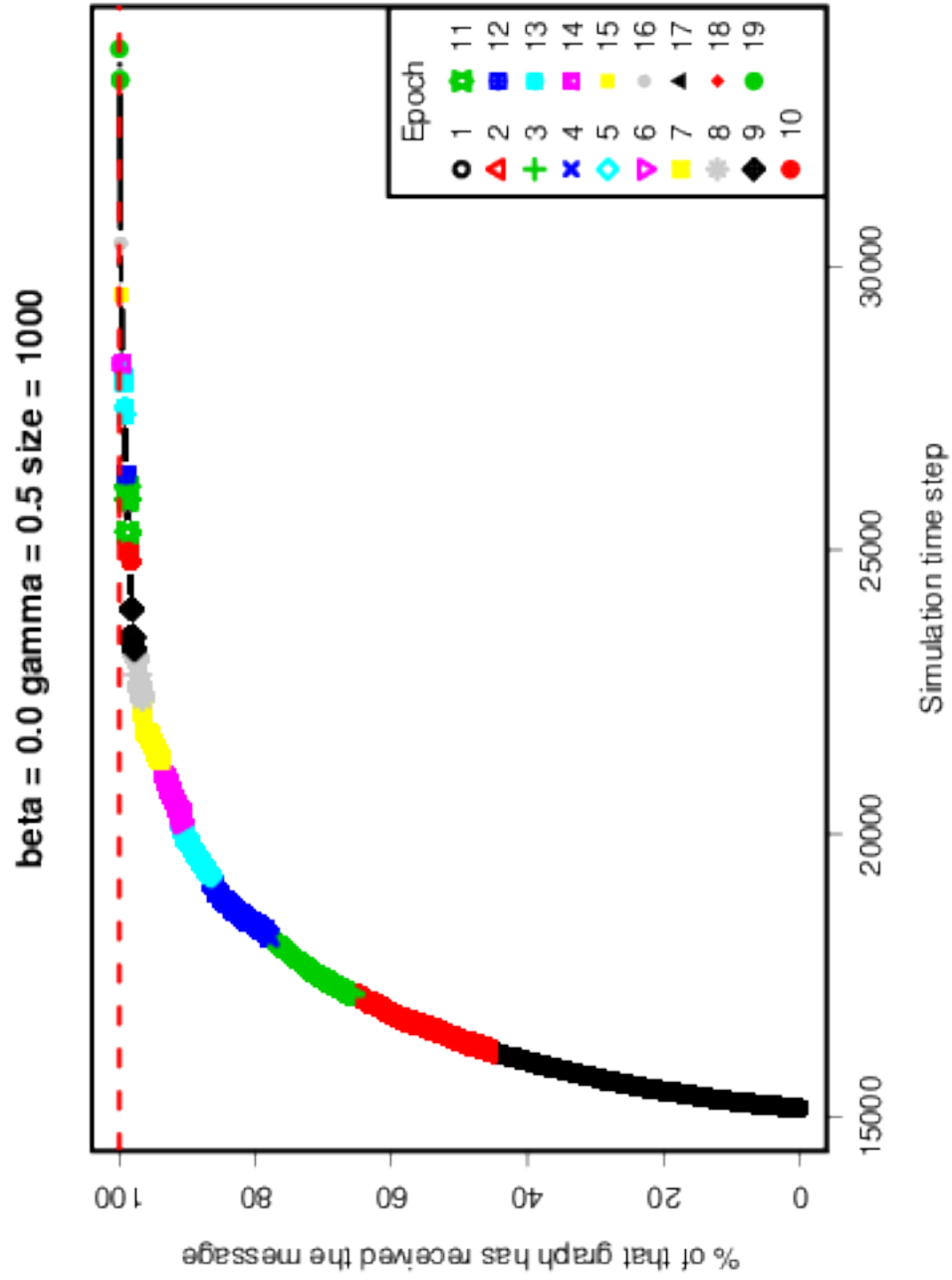


Figure 176. Degree biased selection, size = 1000 $\beta = 0.0$ $\gamma = 0.5$

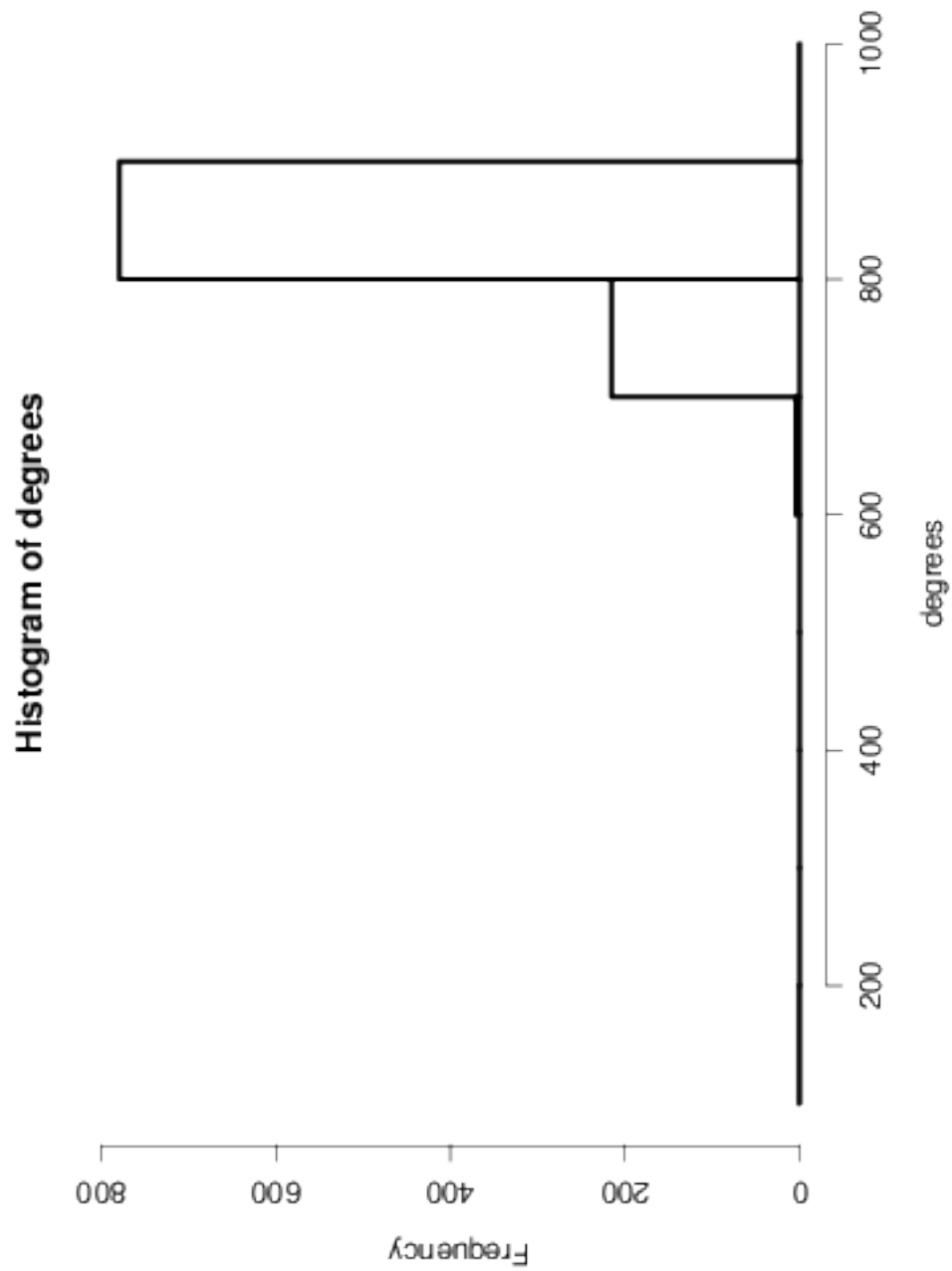


Figure 177. Degree distribution, size = 1000, $\beta = 0.0$, $\gamma = 1.0$

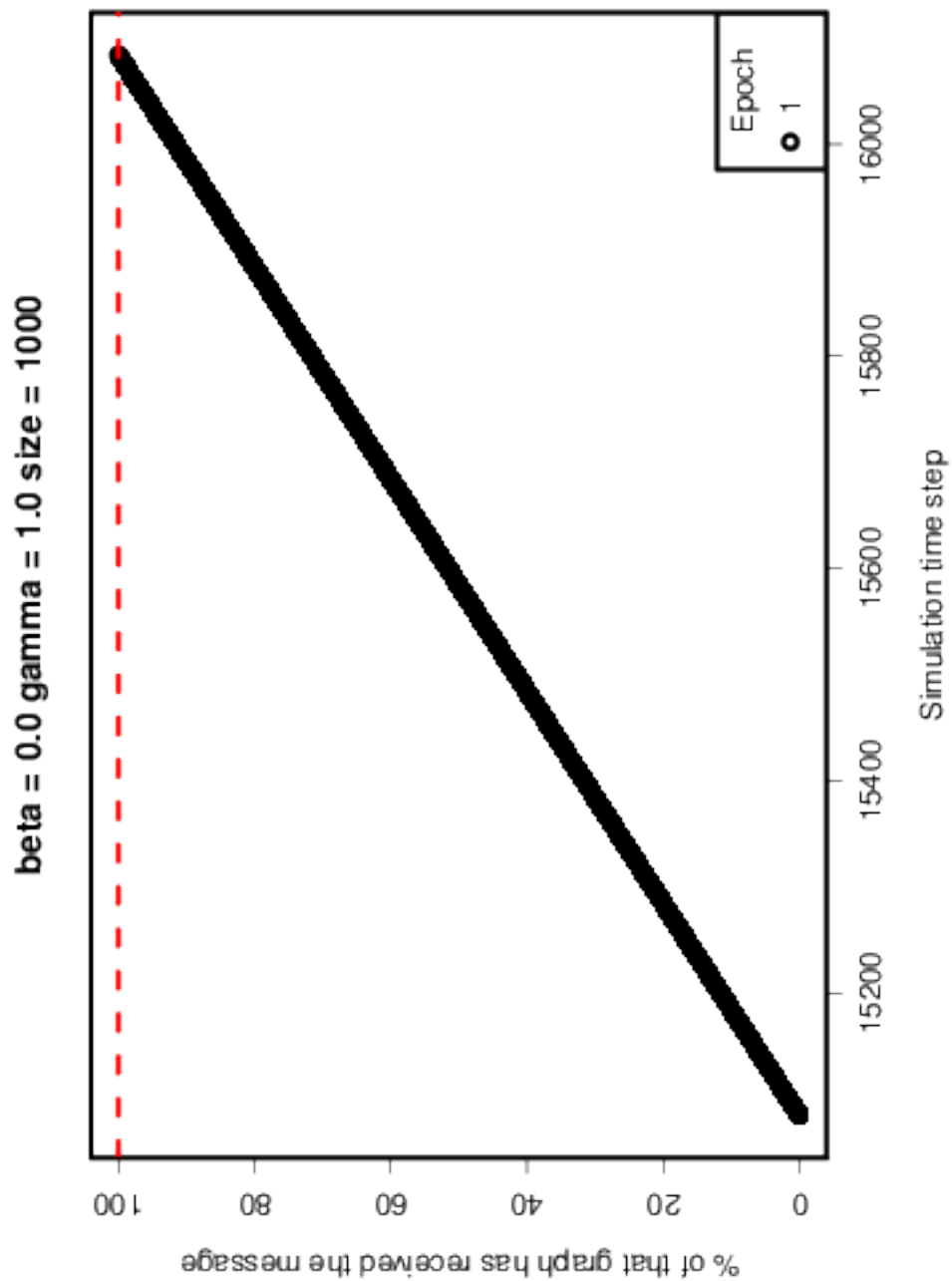


Figure 178. Sequential, size = 1000 $\beta = 0.0$ $\gamma = 1.0$

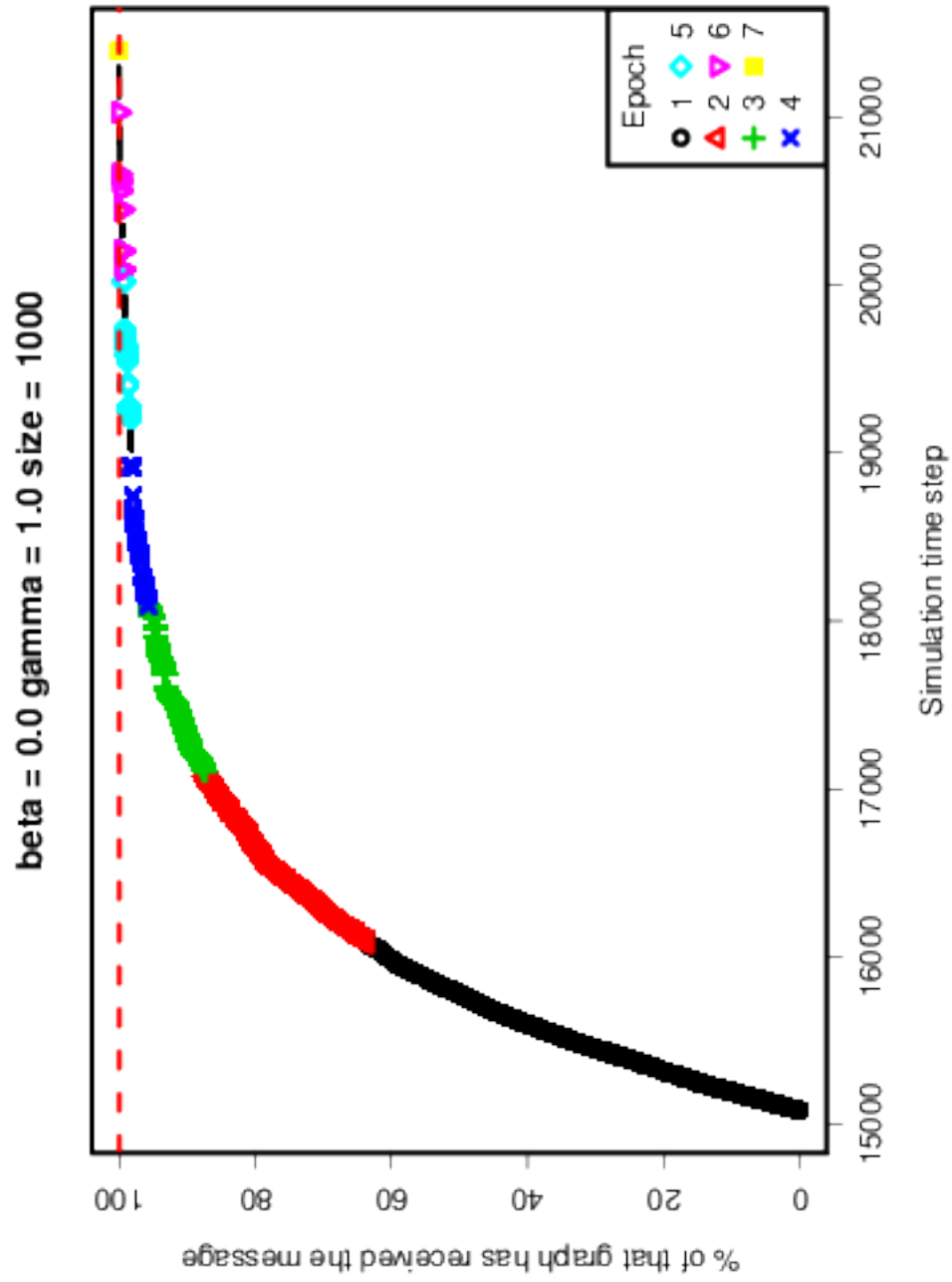


Figure 179. Random selection, size = 1000 $\beta = 0.0$ $\gamma = 1.0$

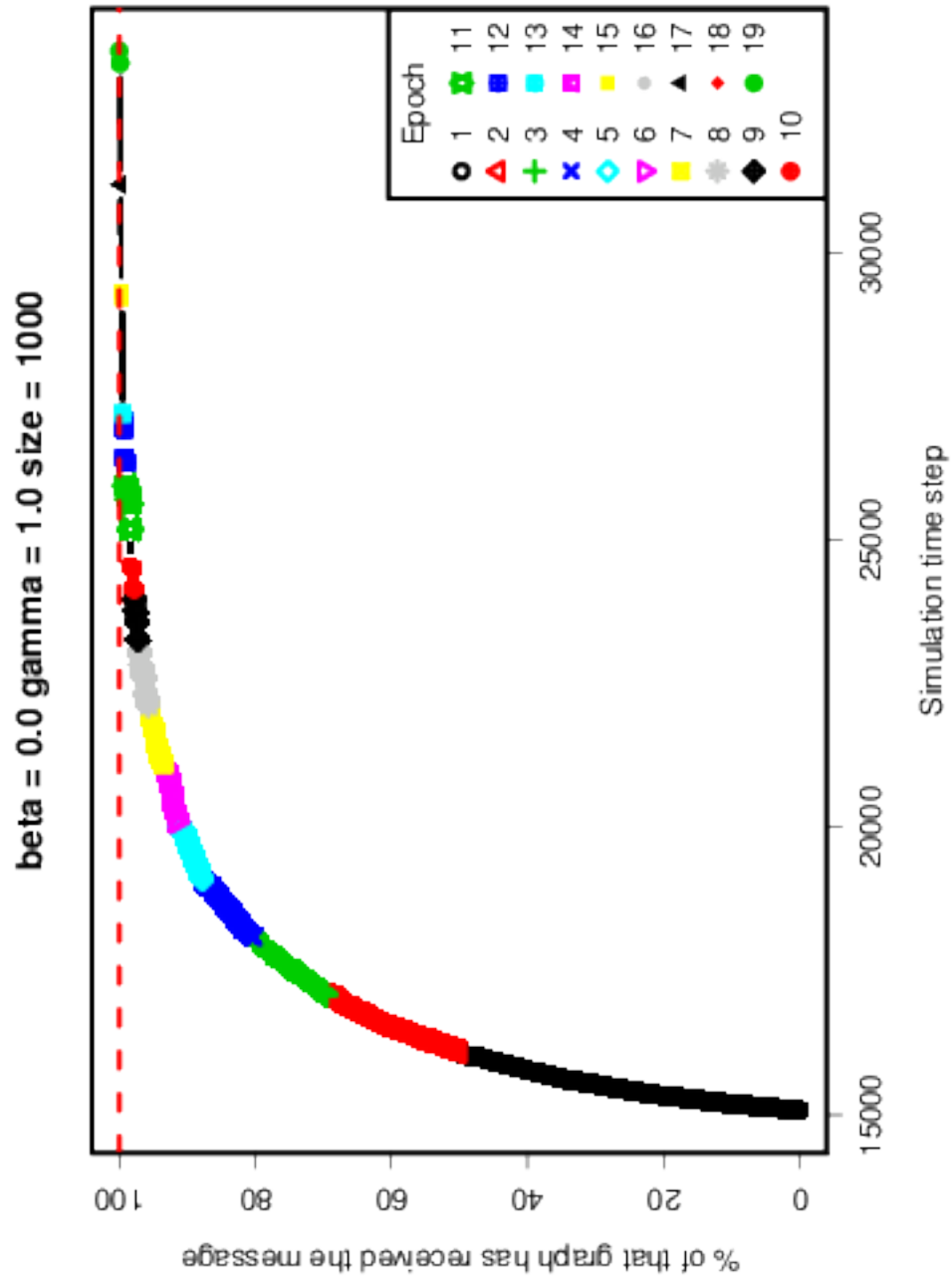


Figure 180. Degree biased selection, size = 1000 $\beta = 0.0$ $\gamma = 1.0$

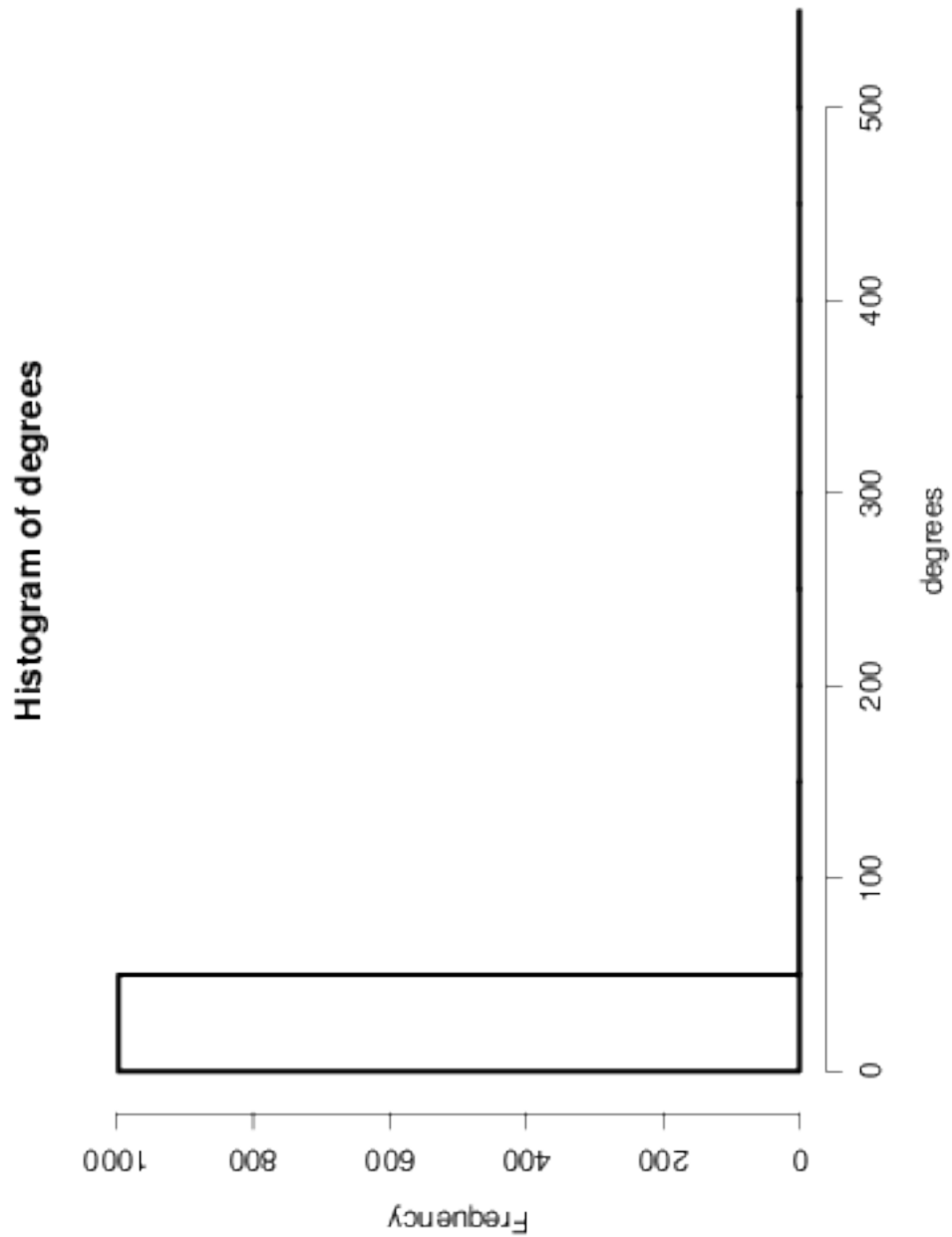


Figure 181. Degree distribution, size = 1000 $\beta = 0.5$ $\gamma = 0.0$

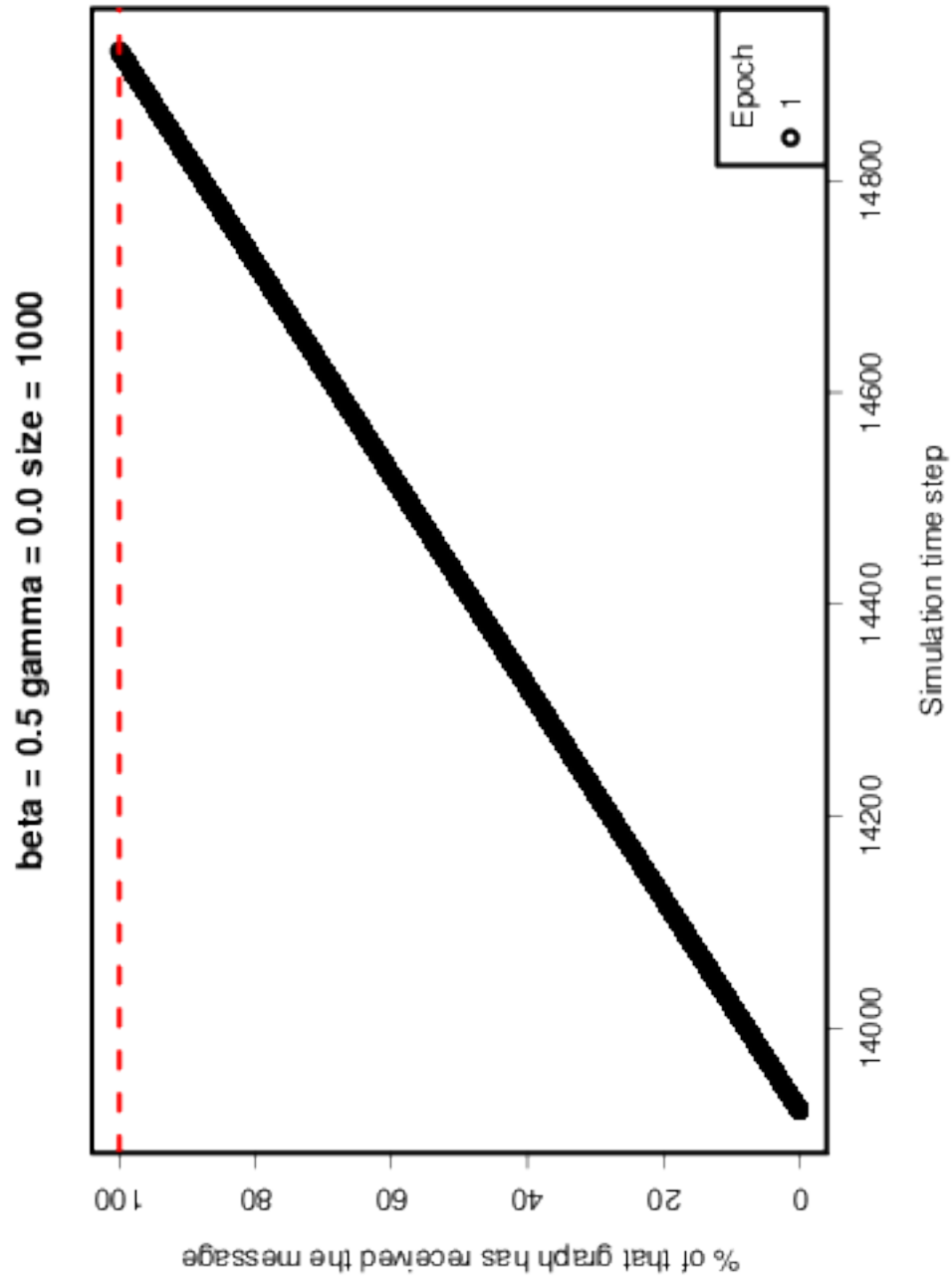


Figure 182. Sequential, size = 1000 $\beta = 0.5$ $\gamma = 0.0$

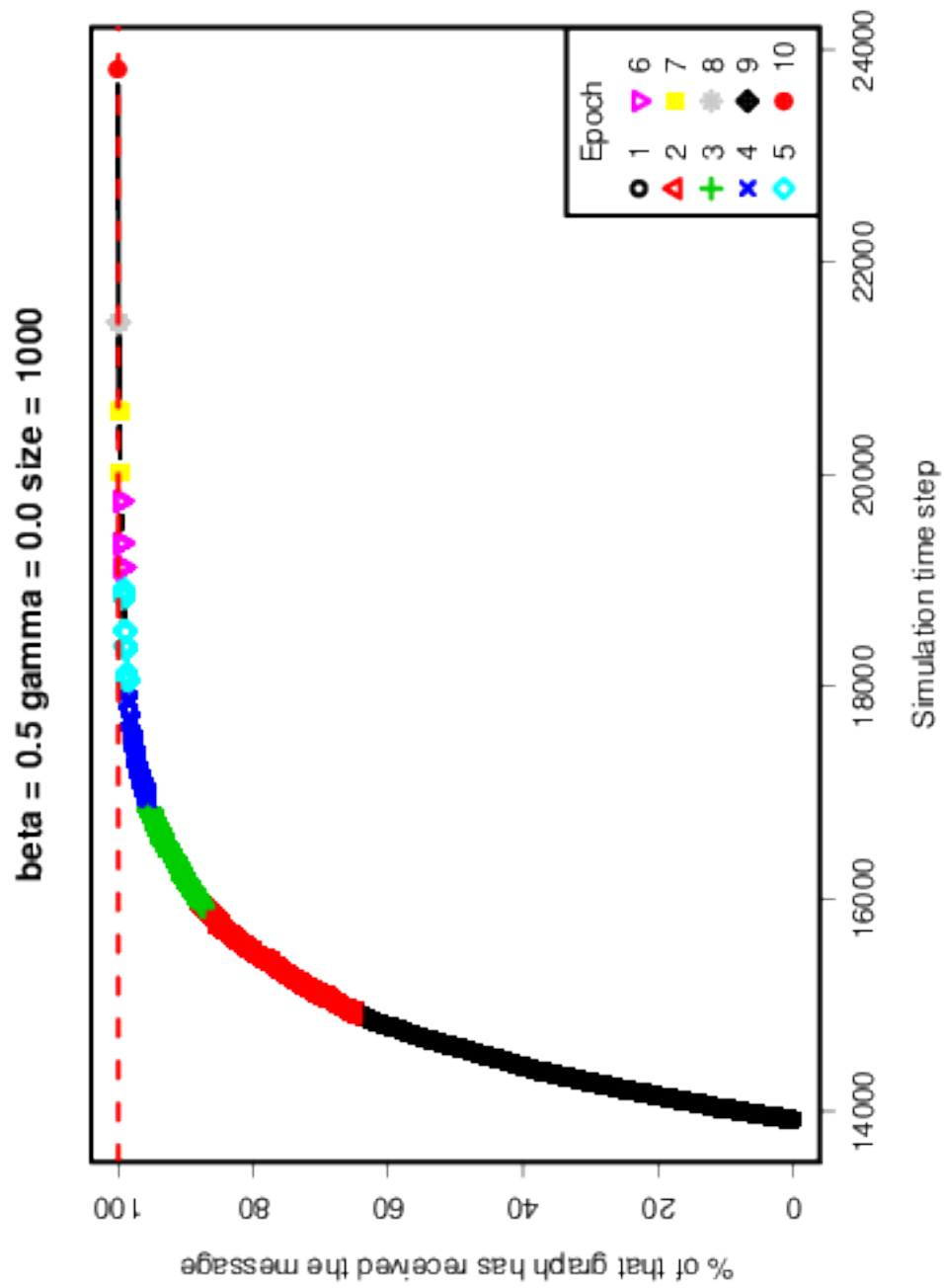


Figure 183. Random selection, size = 1000 $\beta = 0.5$ $\gamma = 0.0$

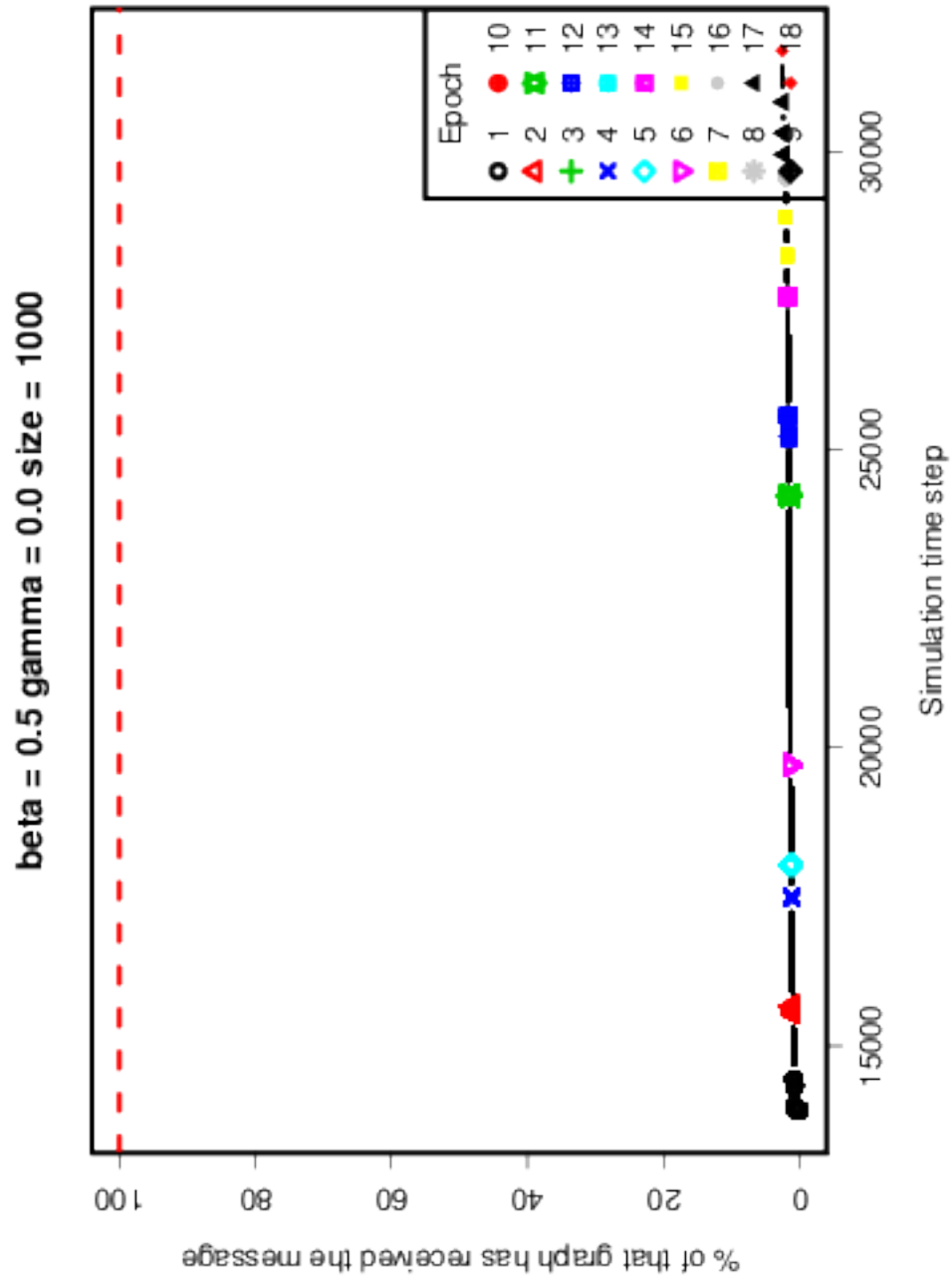


Figure 184. Degree biased selection, size = 1000 $\beta = 0.5$ $\gamma = 0.0$

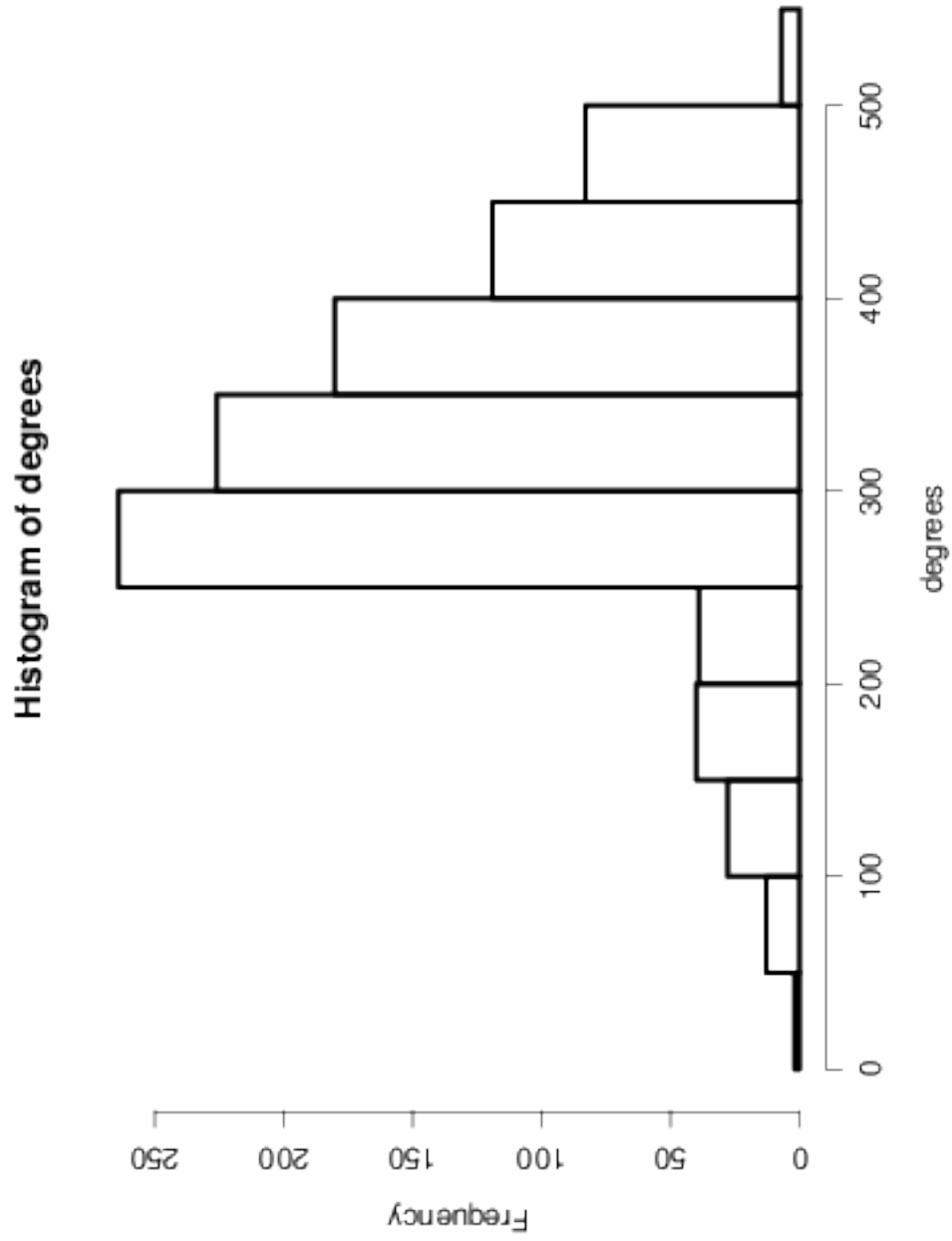


Figure 185. Degree distribution, size = 1000 $\beta = 0.5$ $\gamma = 0.5$

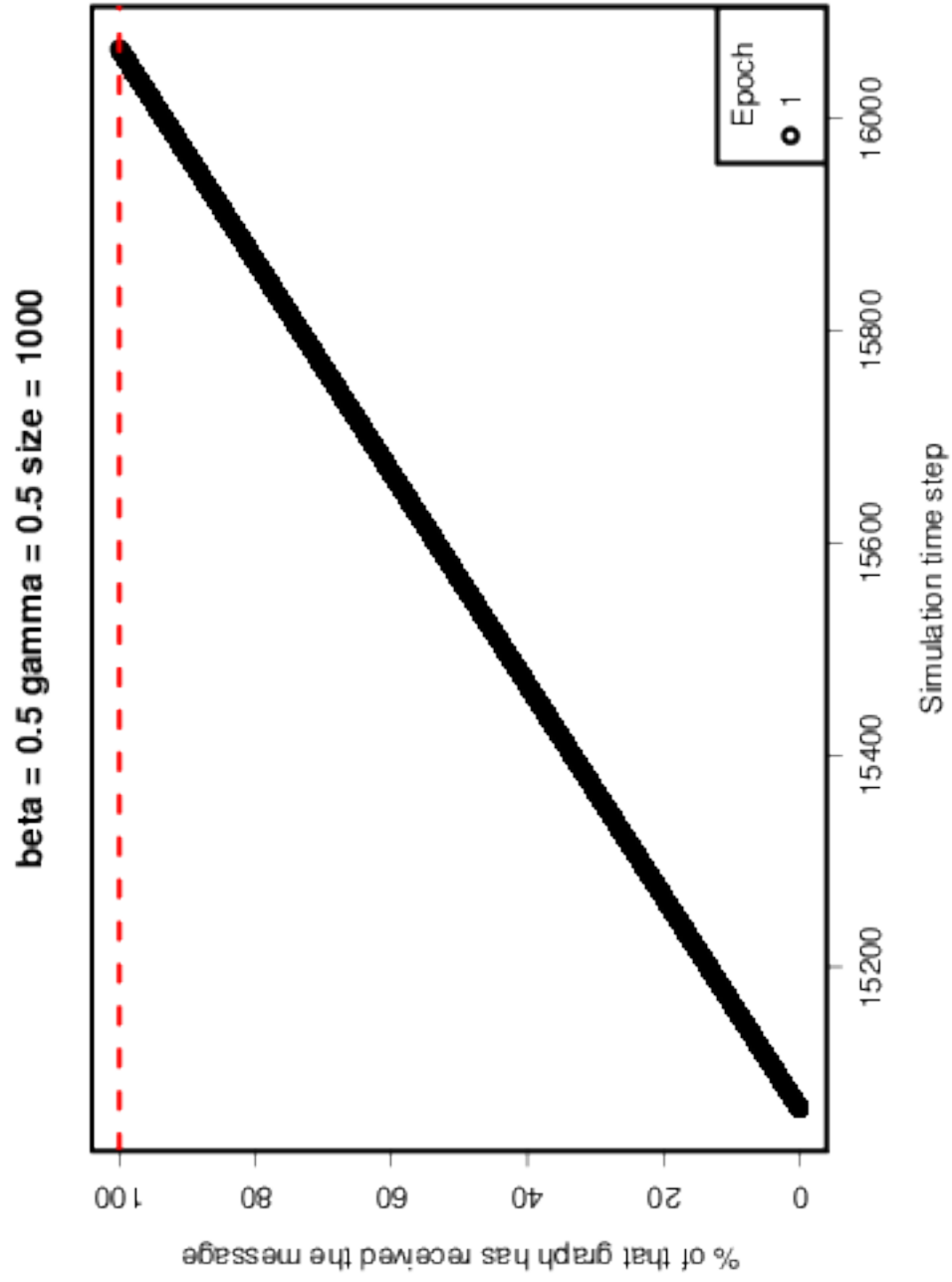


Figure 186. Sequential, size = 1000 $\beta = 0.5$ $\gamma = 0.5$

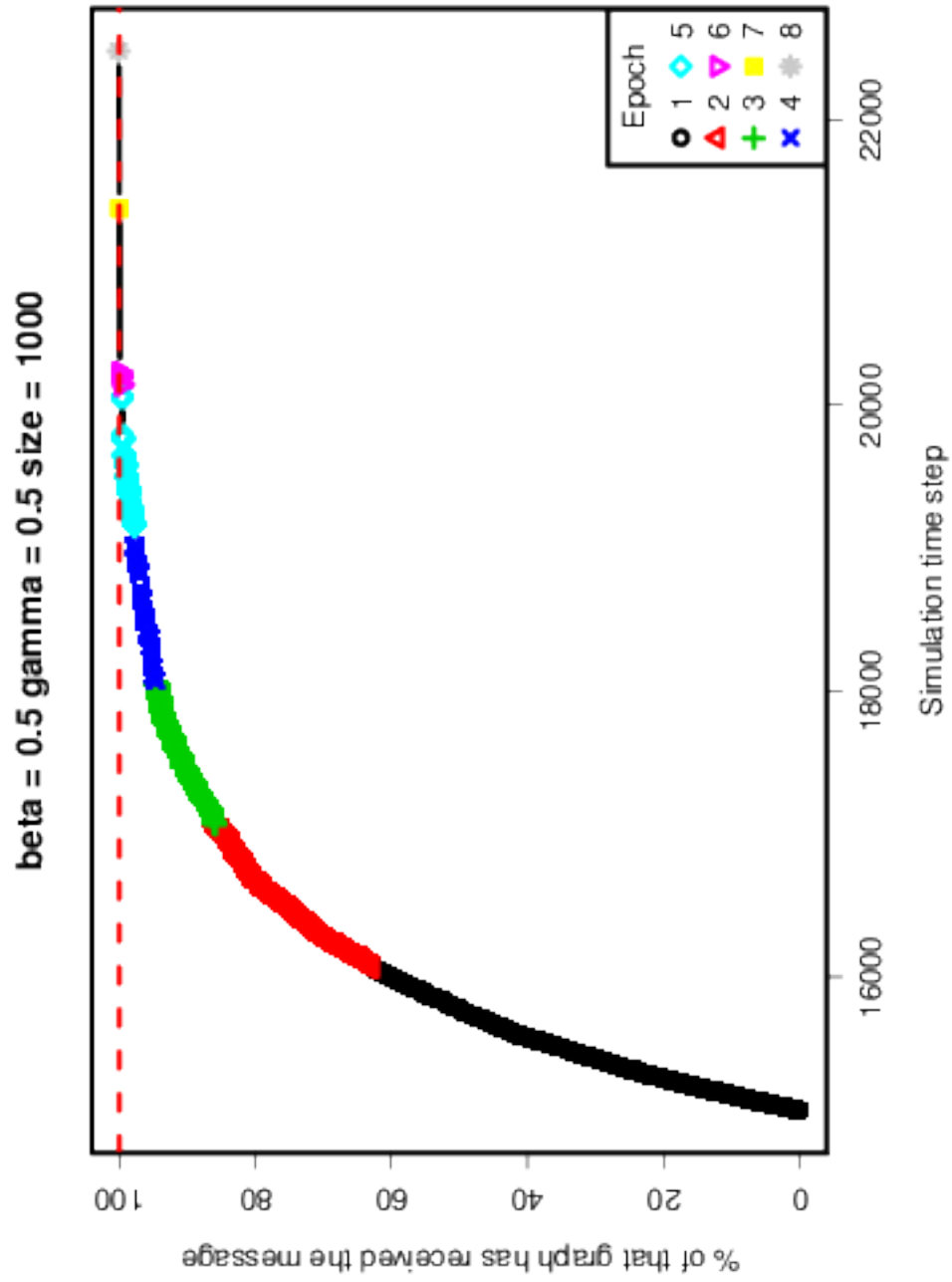


Figure 187. Random selection, size = 1000 $\beta = 0.5$ $\gamma = 0.5$

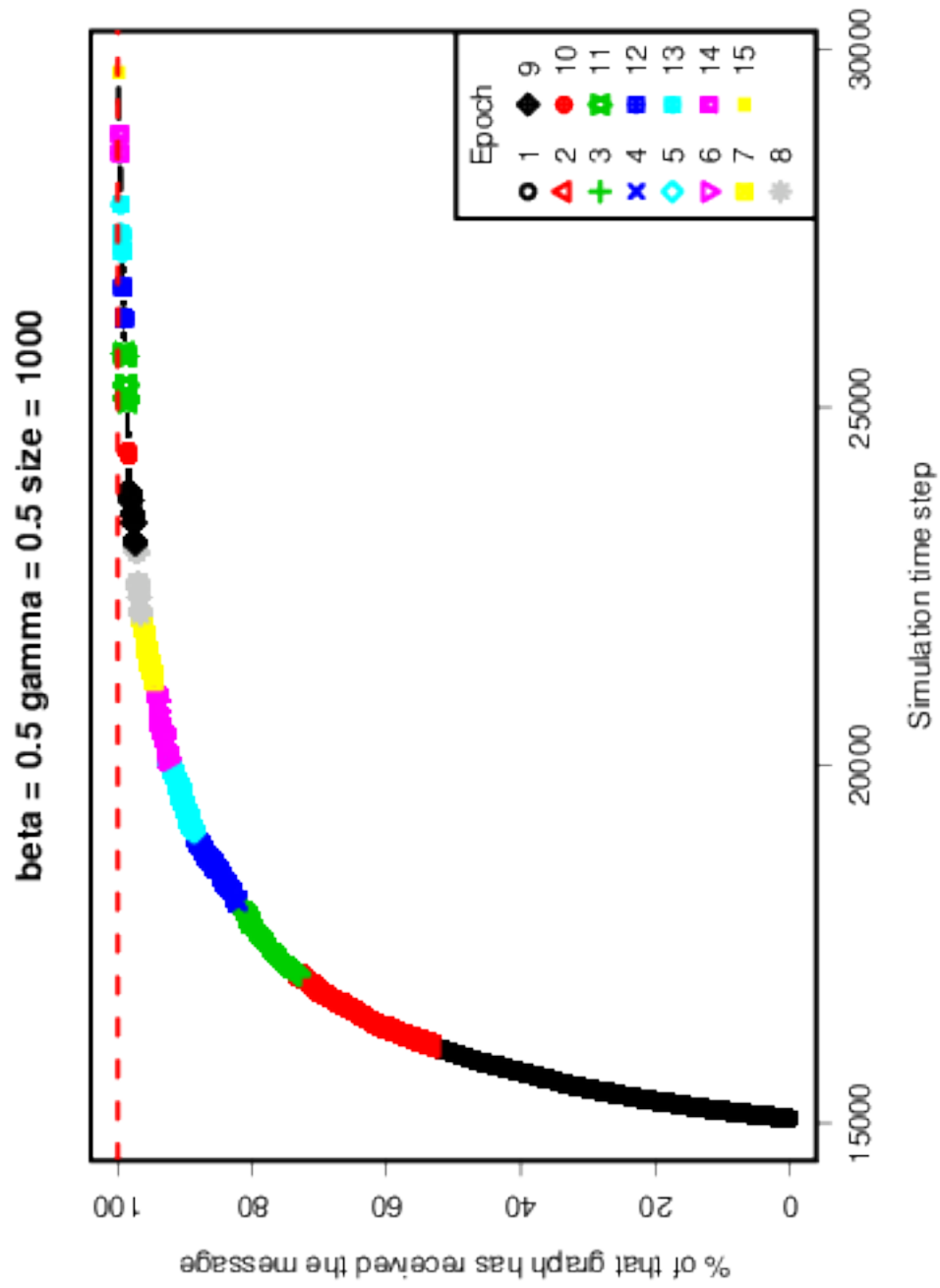


Figure 188. Degree biased selection, size = 1000 $\beta = 0.5$ $\gamma = 0.5$

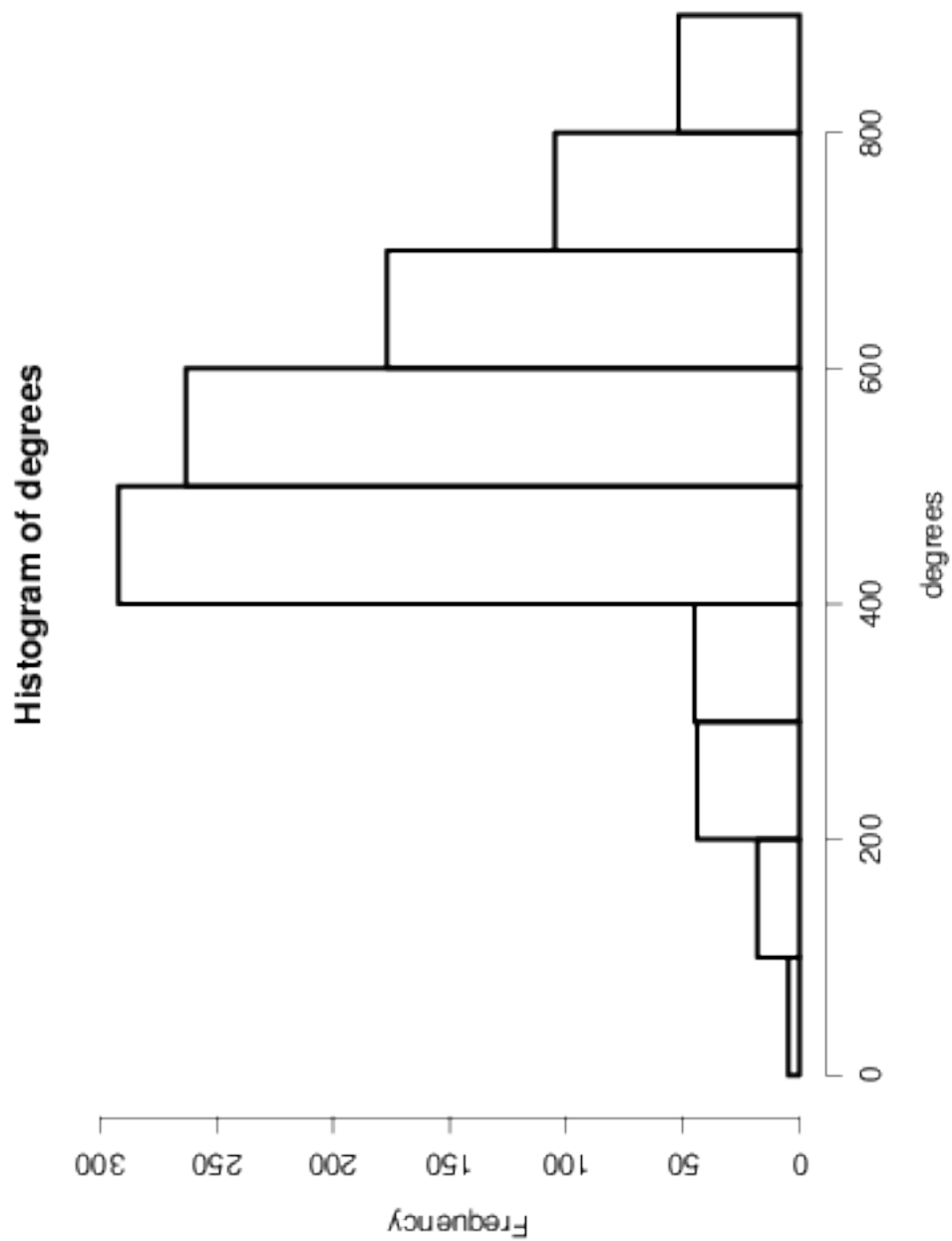


Figure 189. Degree distribution, size = 1000 $\beta = 0.5$ $\gamma = 1.0$

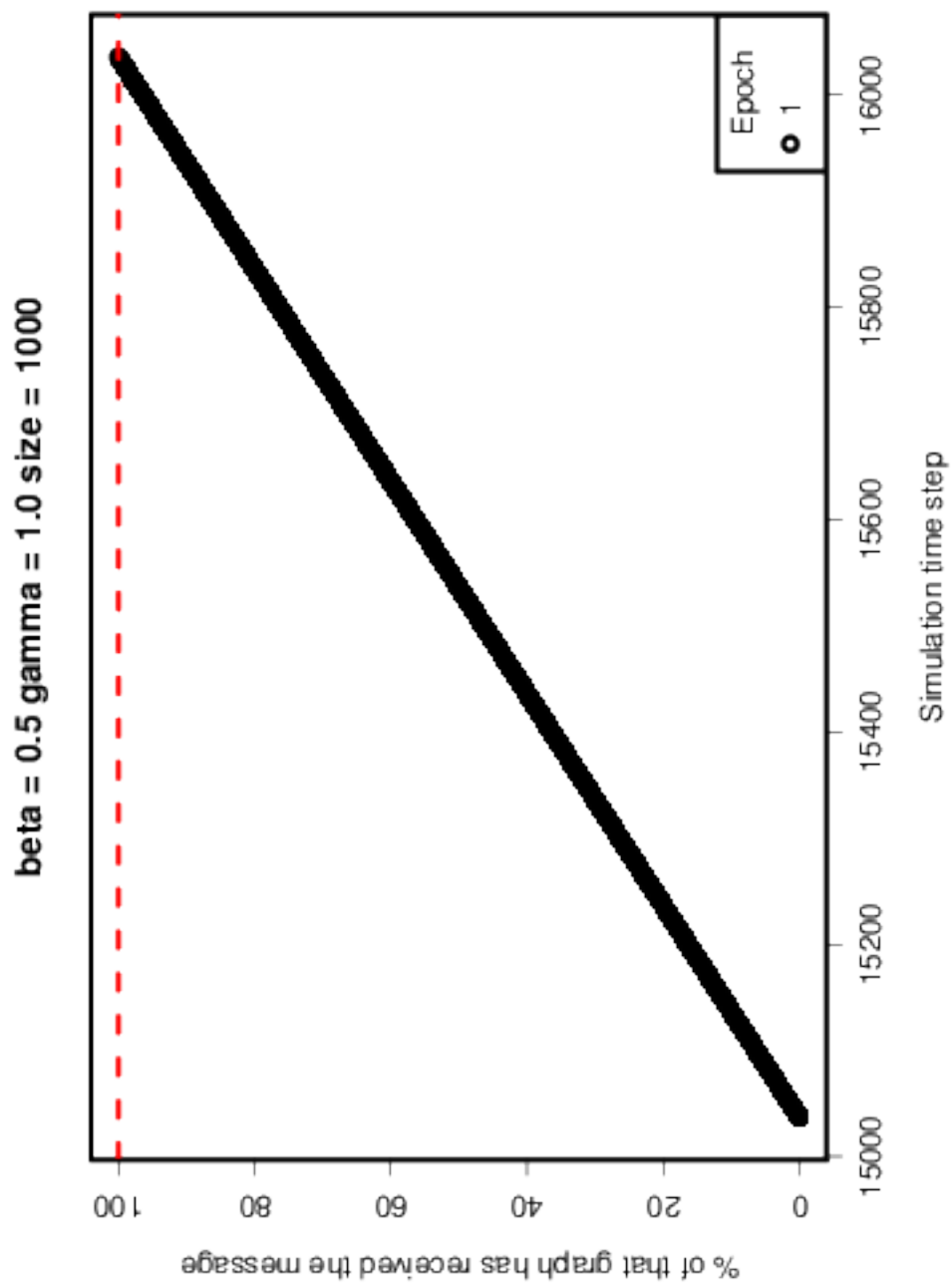


Figure 190. Sequential, size = 1000 $\beta = 0.5$ $\gamma = 1.0$

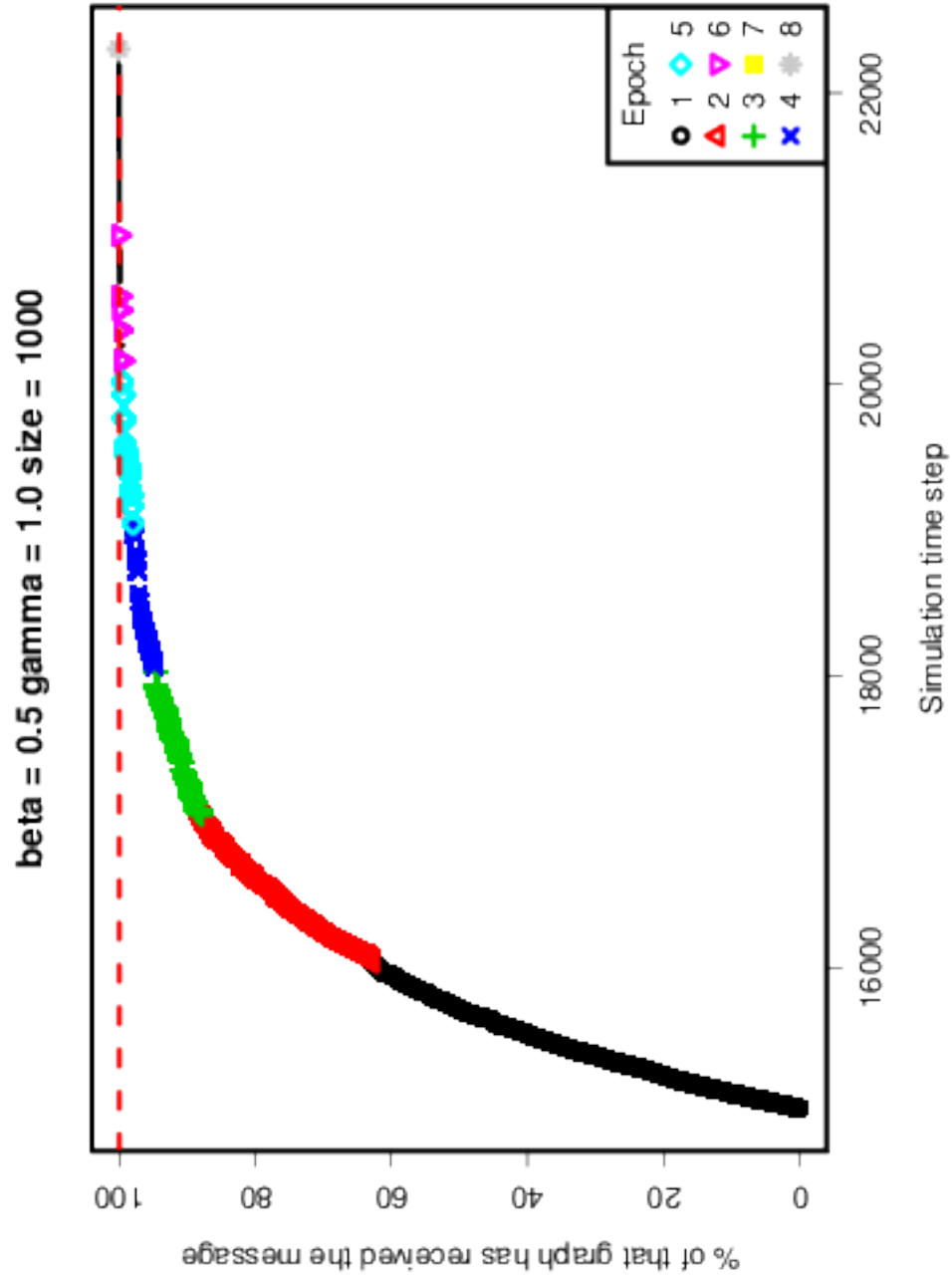


Figure 191. Random selection, size = 1000 $\beta = 0.5$ $\gamma = 1.0$

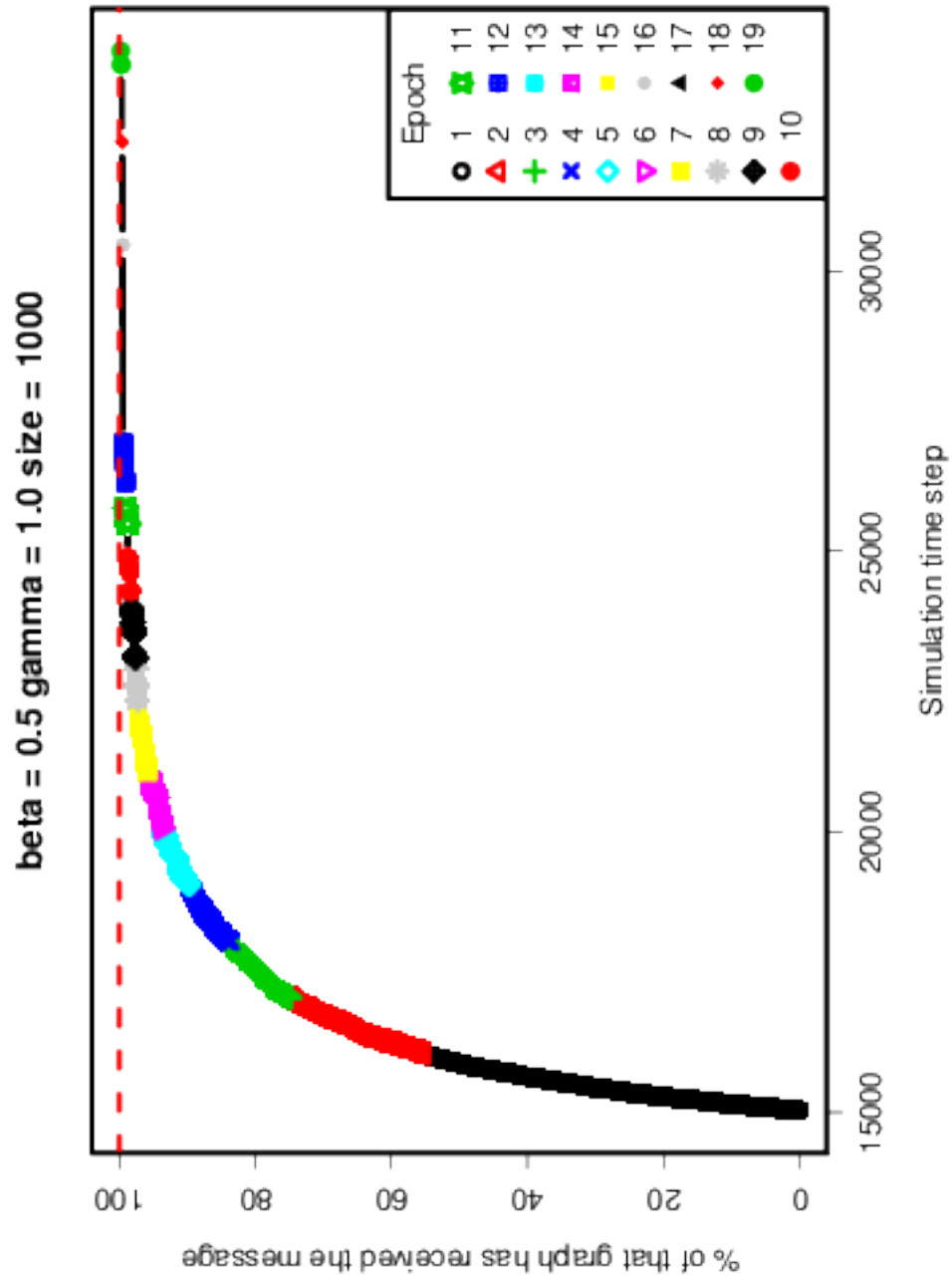


Figure 192. Degree biased selection, size = 1000 $\beta = 0.5$ $\gamma = 1.0$

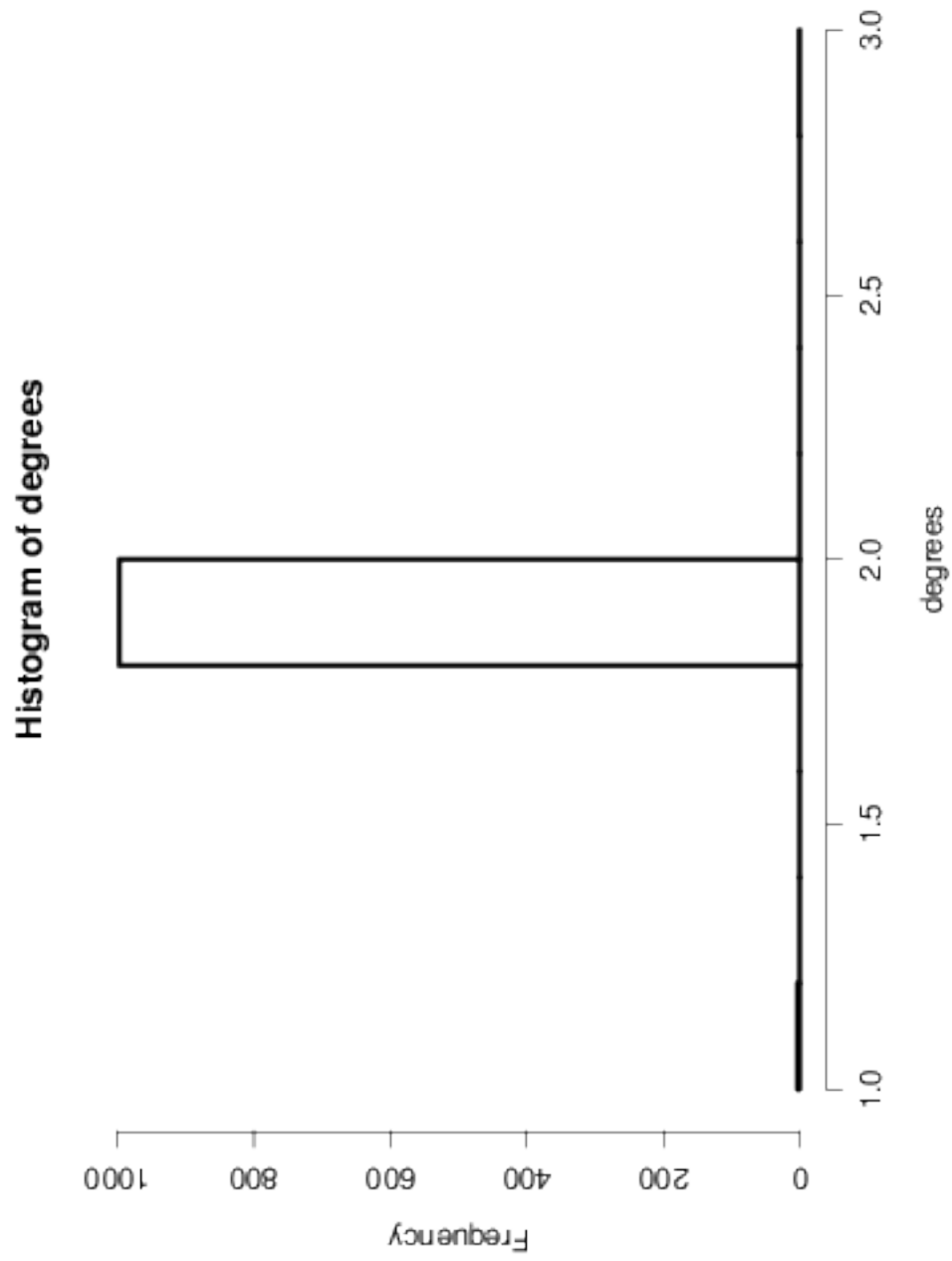


Figure 193. Degree distribution, size = 1000 $\beta = 1.0$ $\gamma = 0.0$

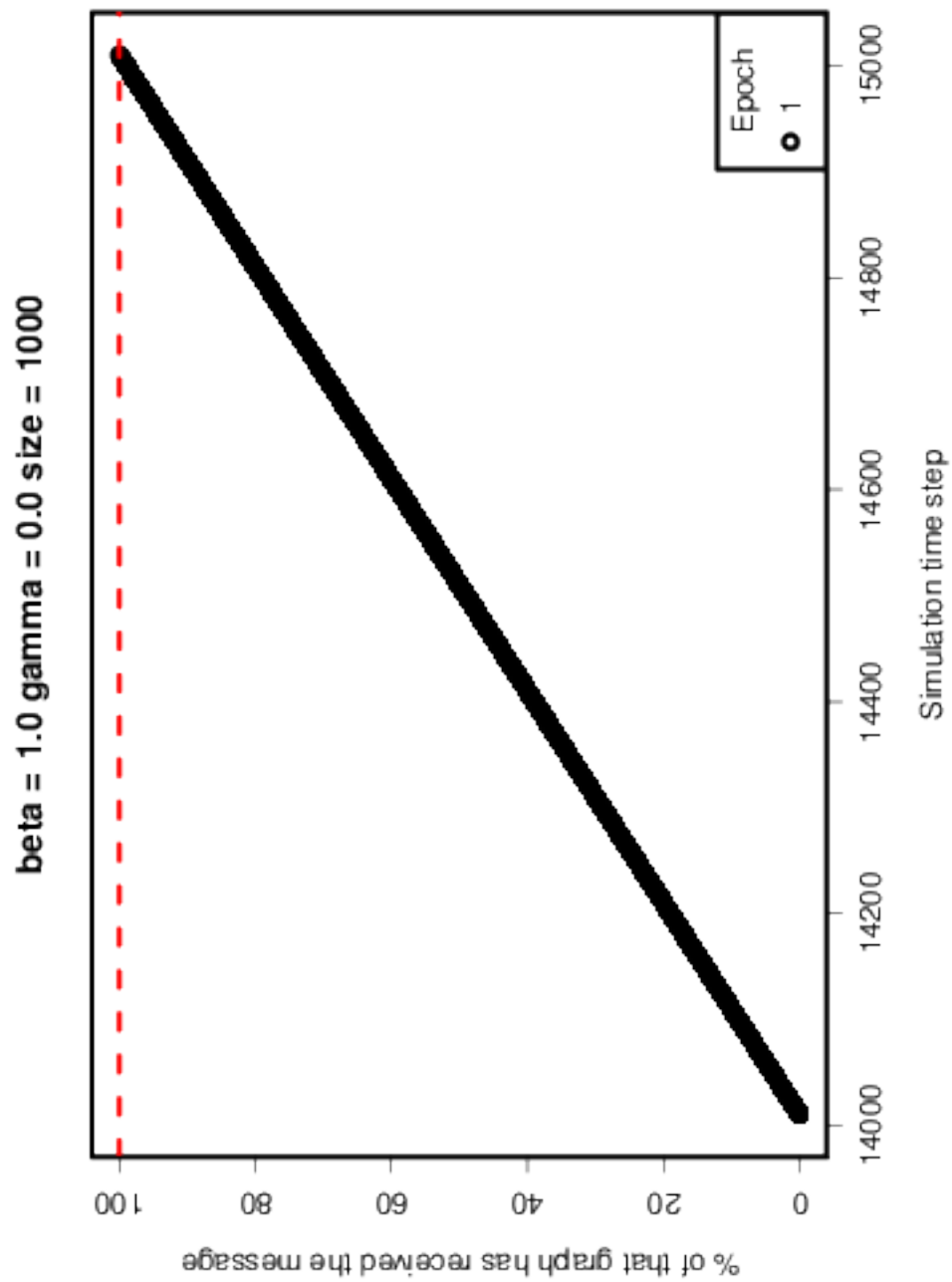


Figure 194. Sequential, size = 1000 $\beta = 1.0$ $\gamma = 0.0$

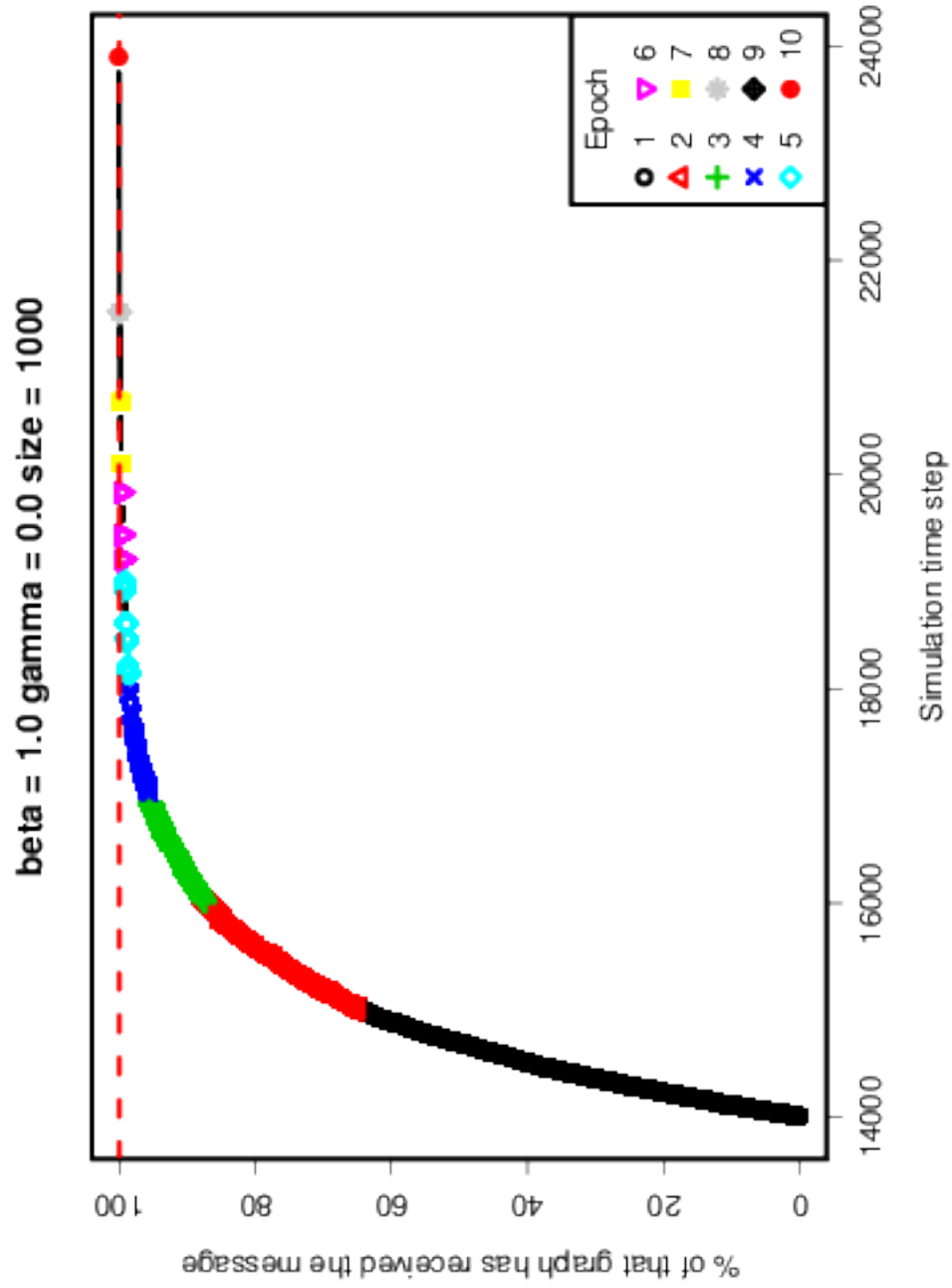


Figure 195. Random selection, size = 1000 $\beta = 1.0$ $\gamma = 0.0$

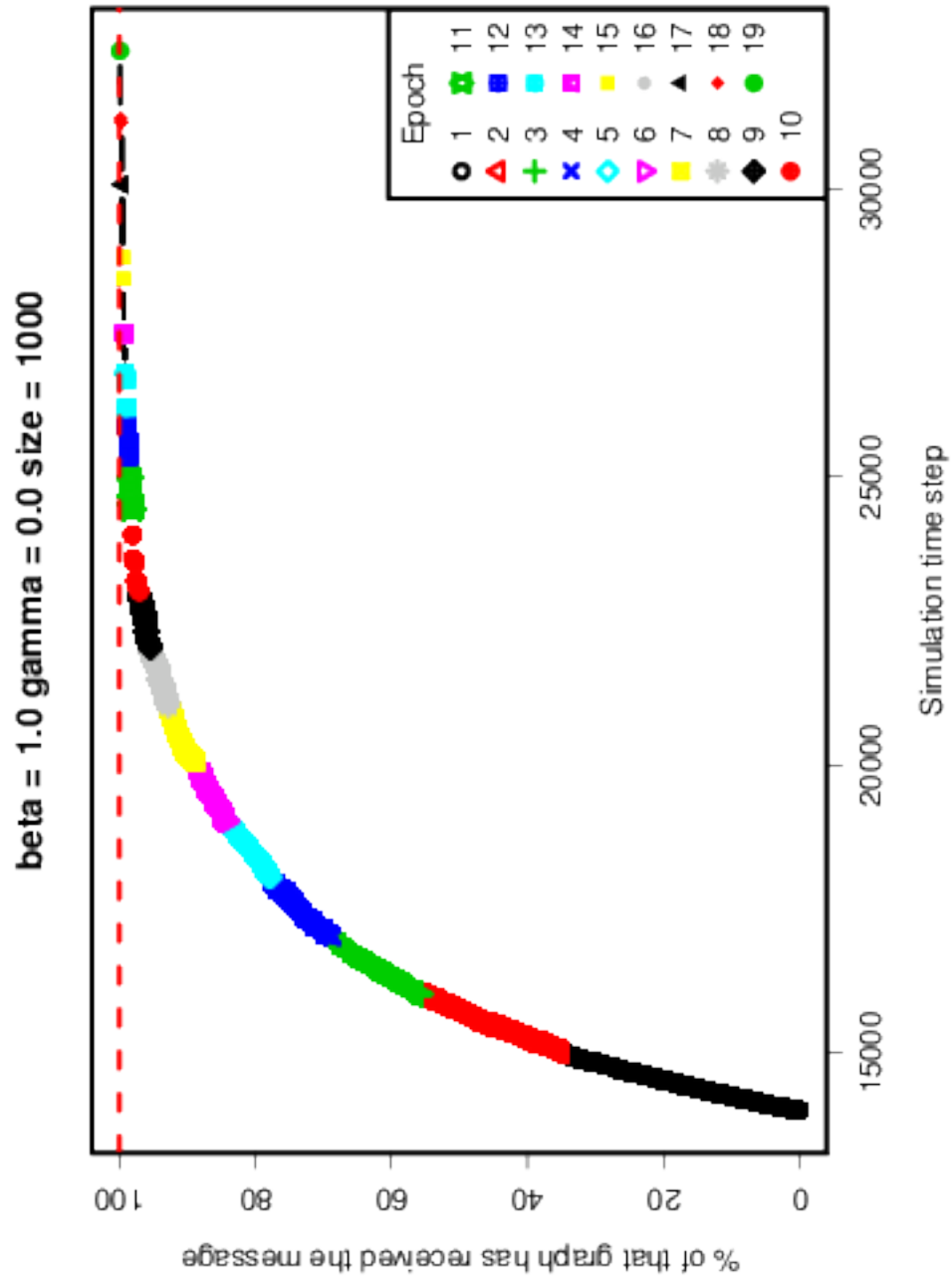


Figure 196. Degree biased selection, size = 1000 $\beta = 1.0$ $\gamma = 0.0$

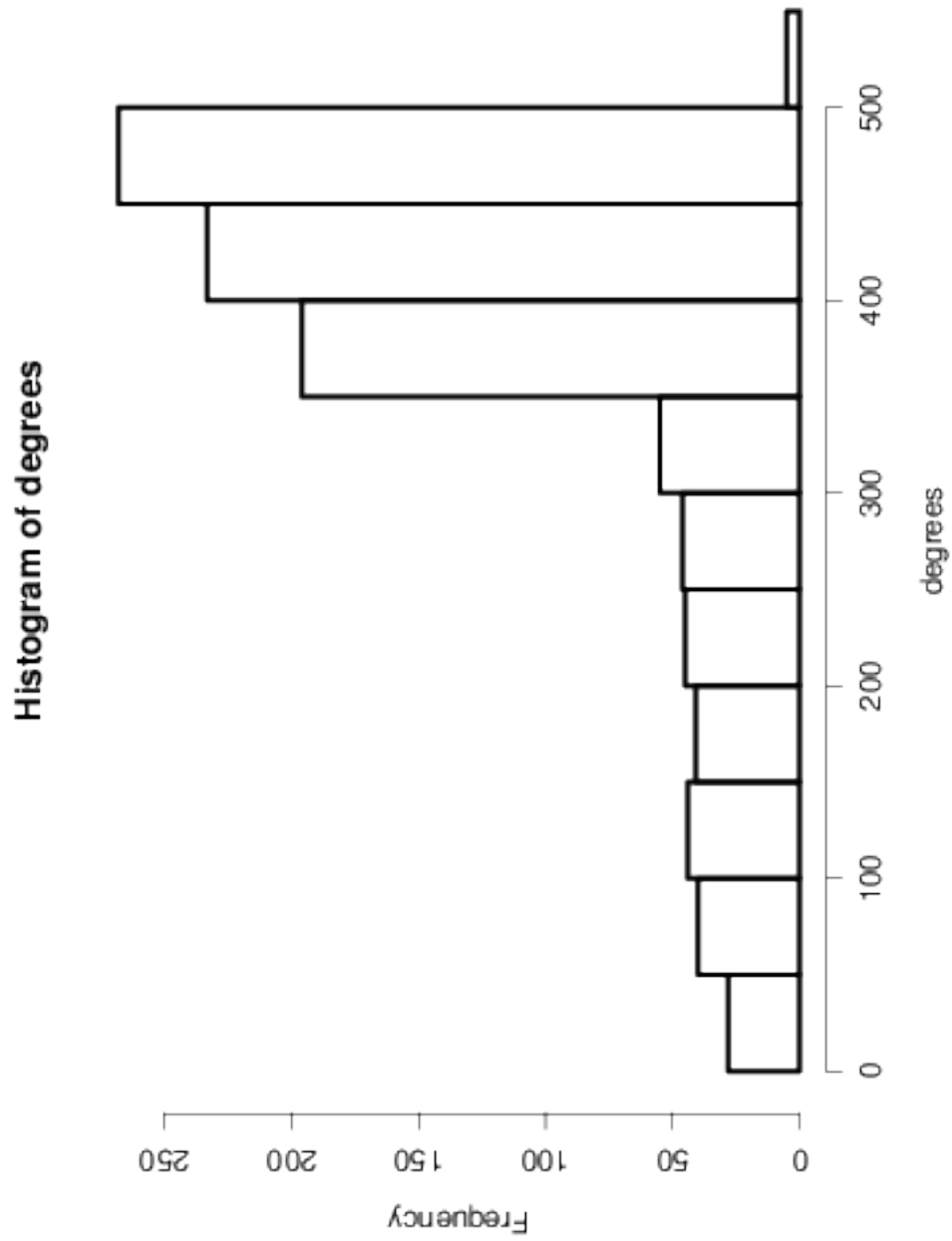


Figure 197. Degree distribution, size = 1000 $\beta = 1.0$ $\gamma = 0.5$

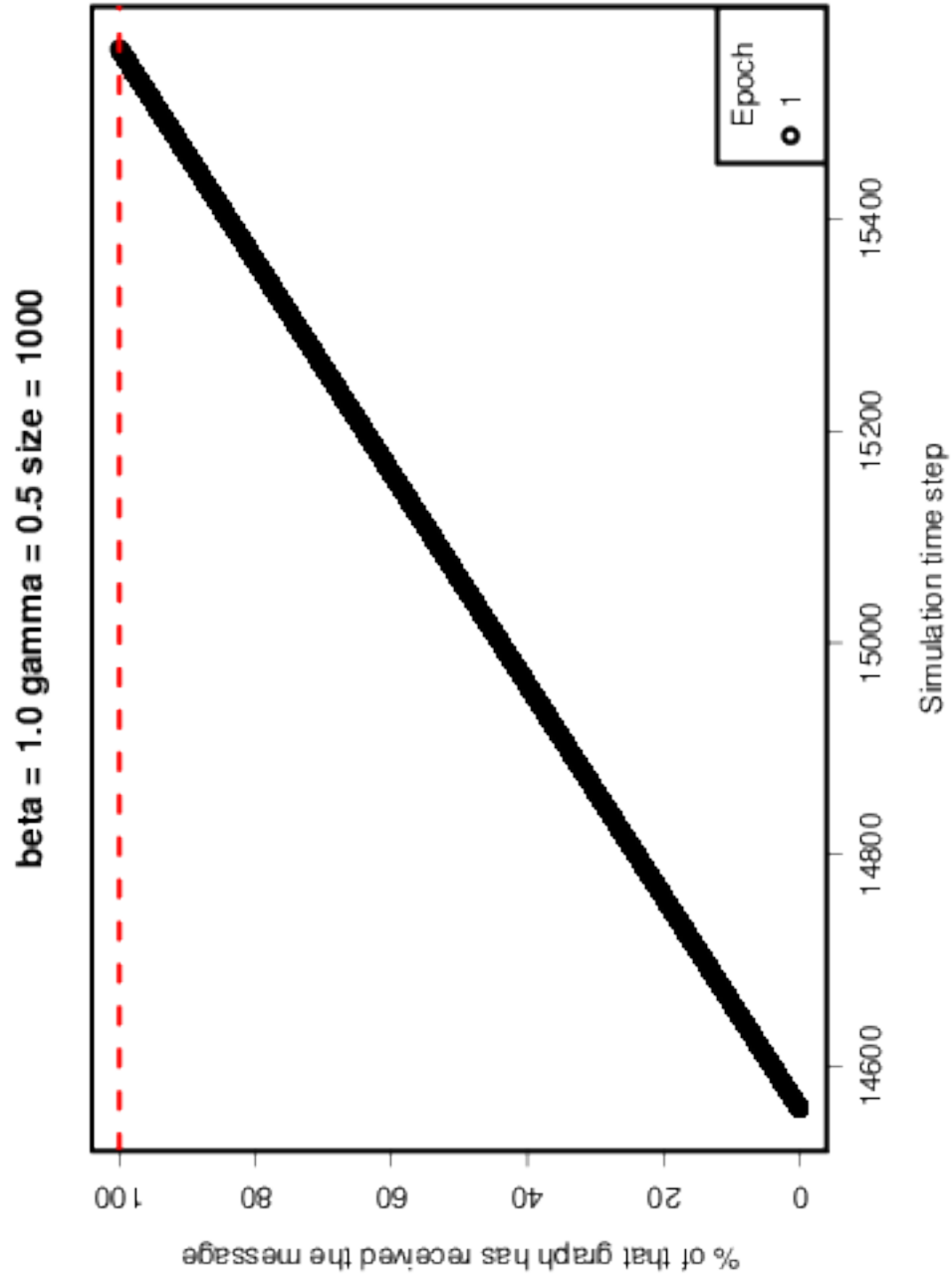


Figure 198. Sequential, size = 1000 $\beta = 1.0$ $\gamma = 0.5$

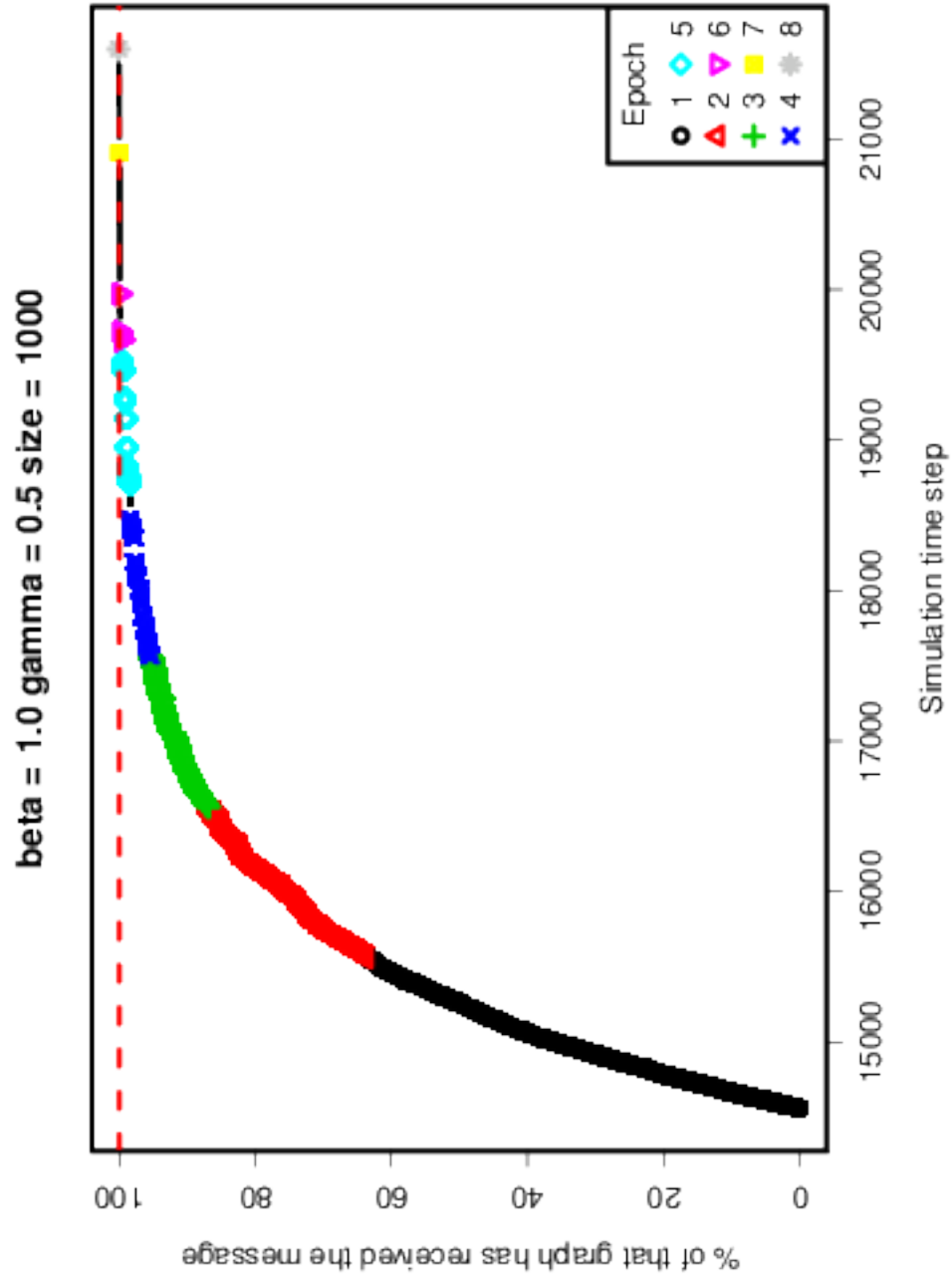


Figure 199. Random selection, size = 1000 $\beta = 1.0$ $\gamma = 0.5$

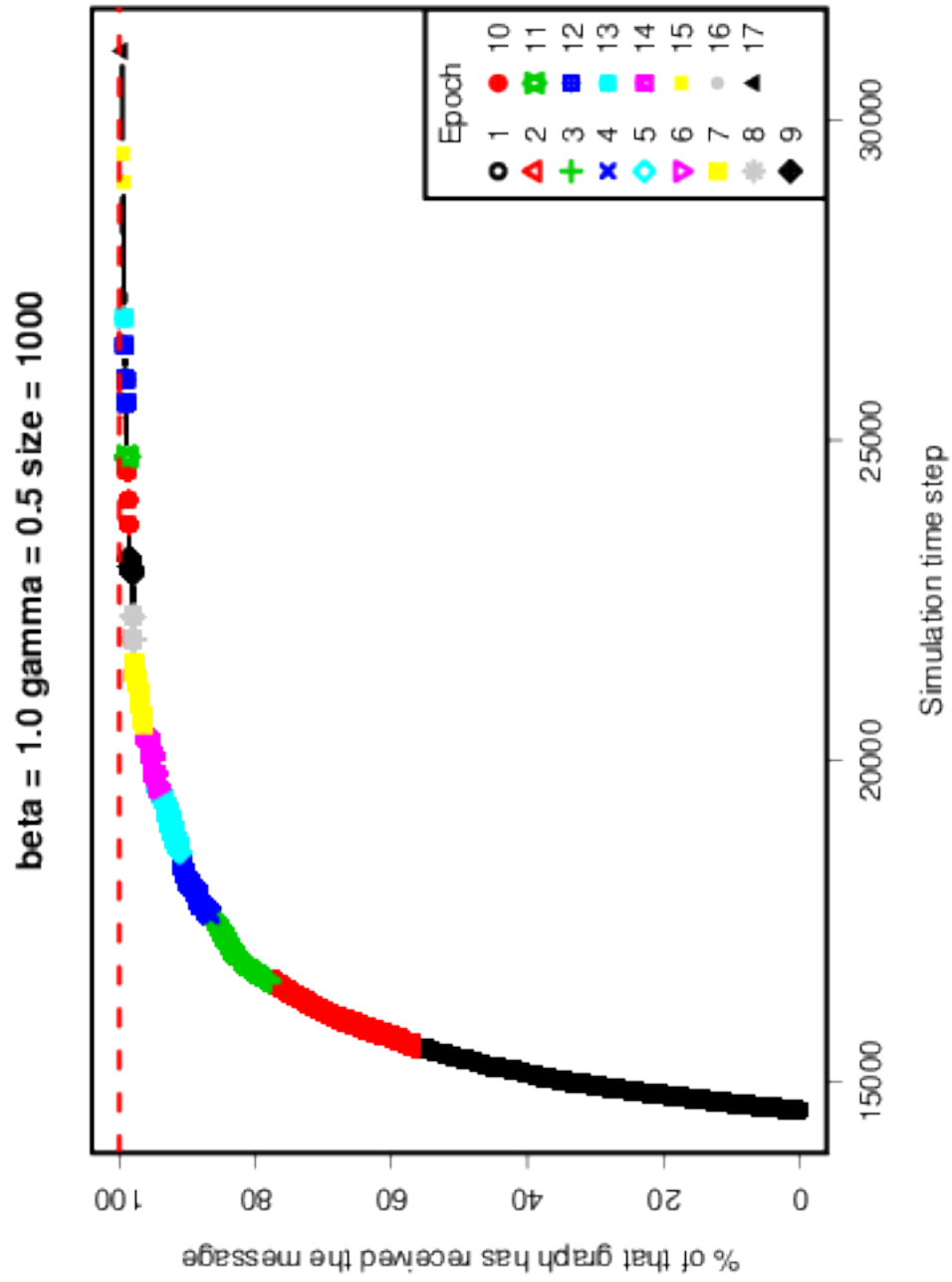


Figure 200. Degree biased selection, size = 1000 $\beta = 1.0$ $\gamma = 0.5$

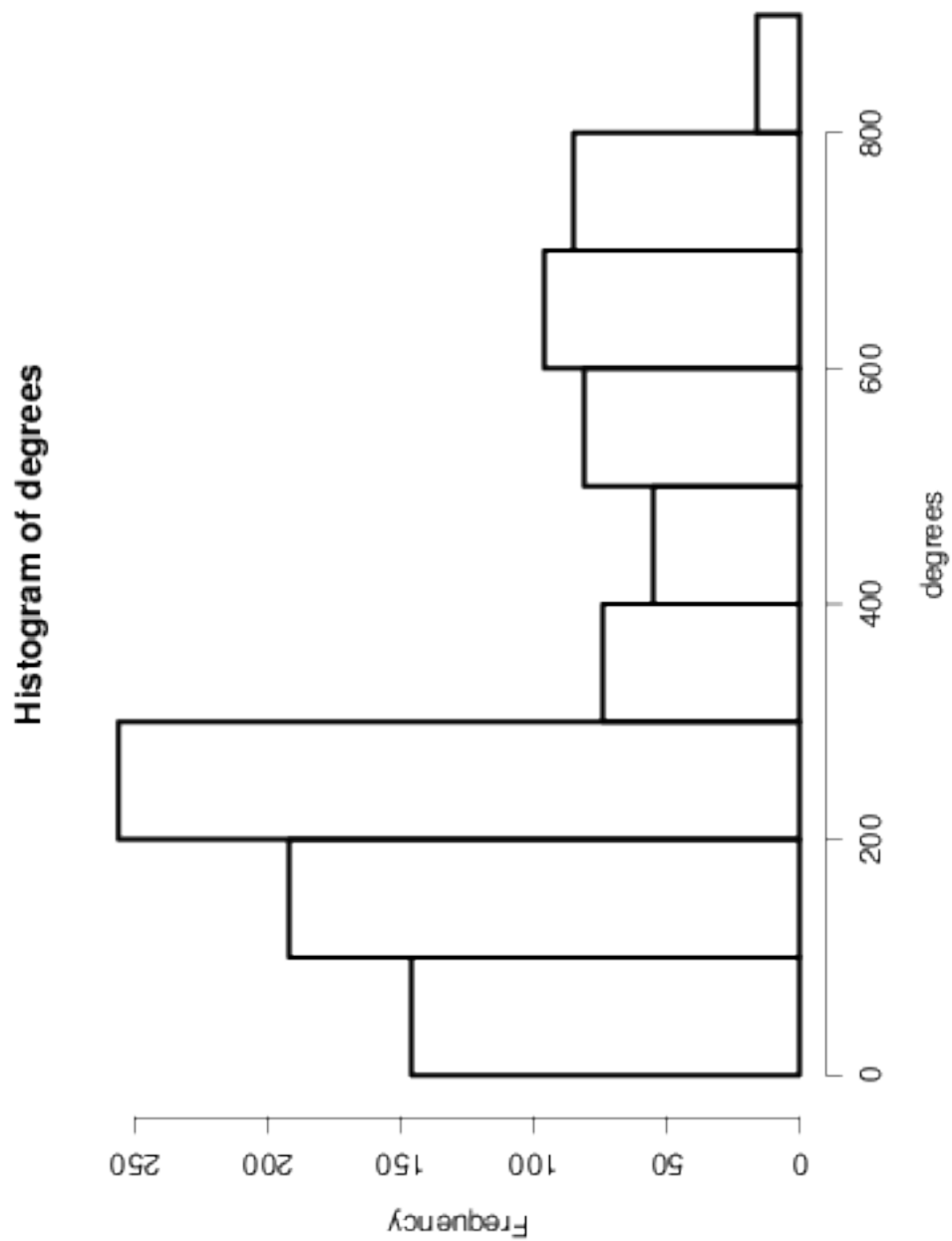


Figure 201. Degree distribution, size = 1000 $\beta = 1.0$ $\gamma = 1.0$

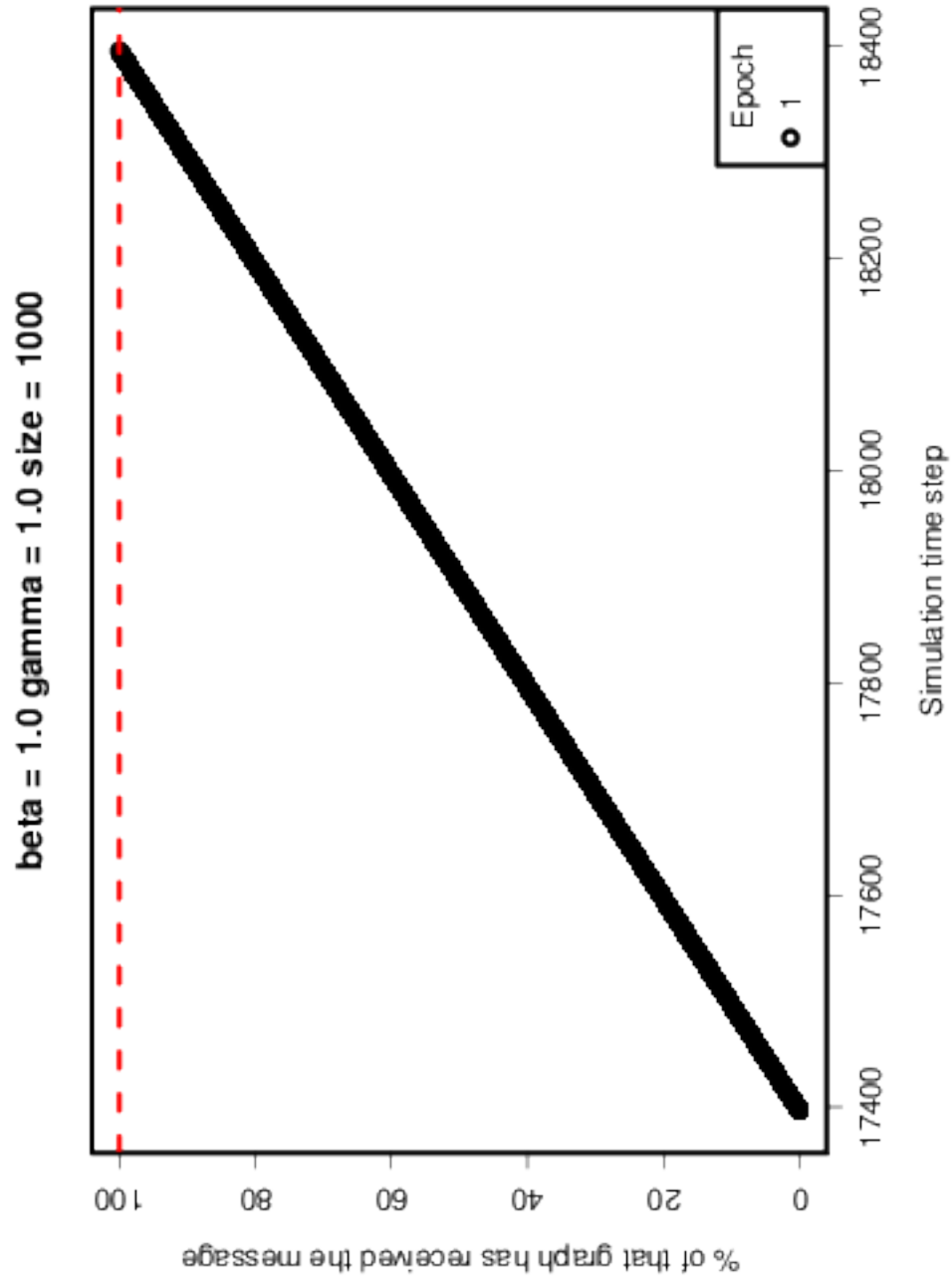


Figure 202. Sequential, size = 1000 $\beta = 1.0$ $\gamma = 1.0$

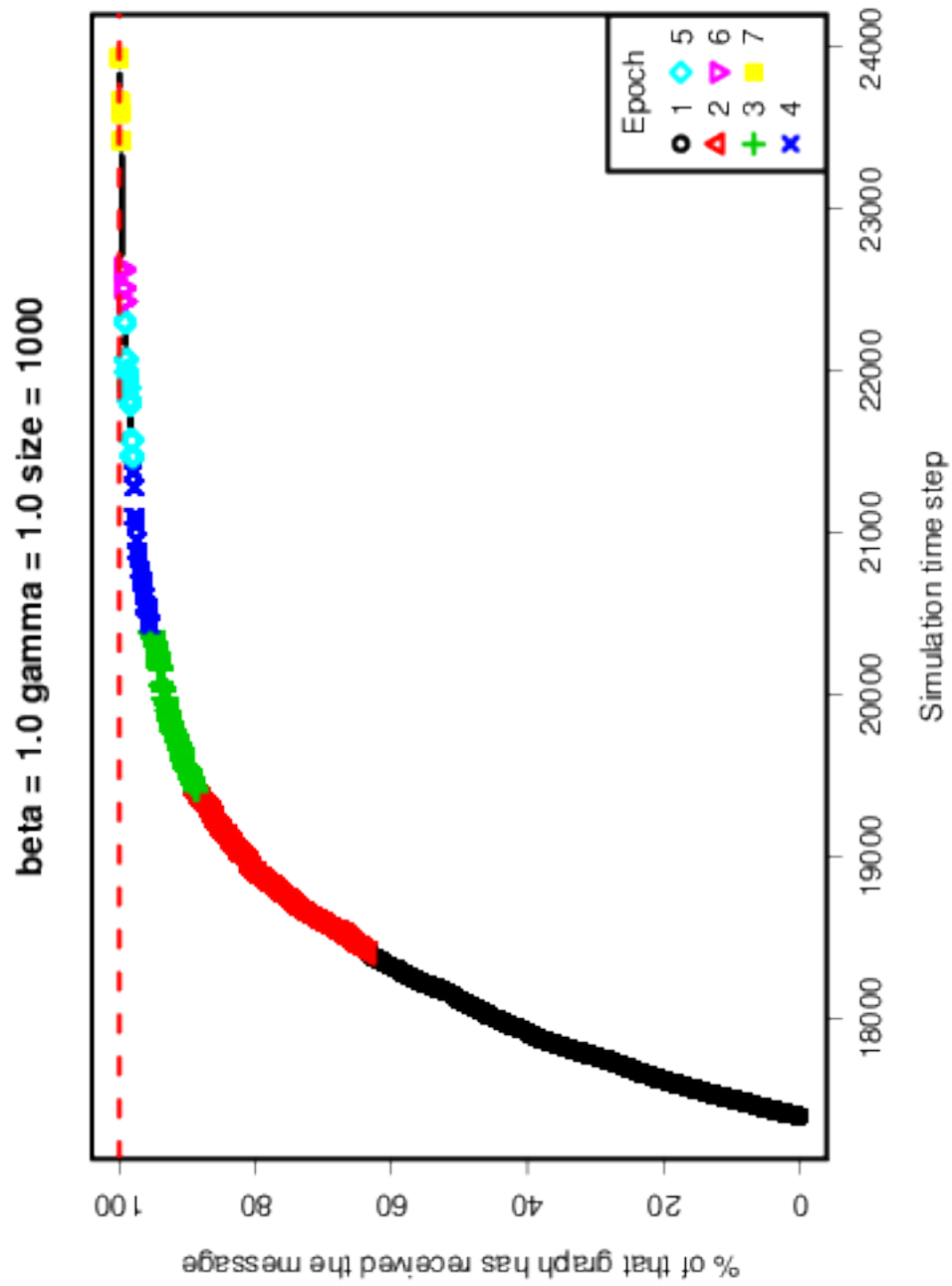


Figure 203. Random selection, size = 1000 $\beta = 1.0$ $\gamma = 1.0$

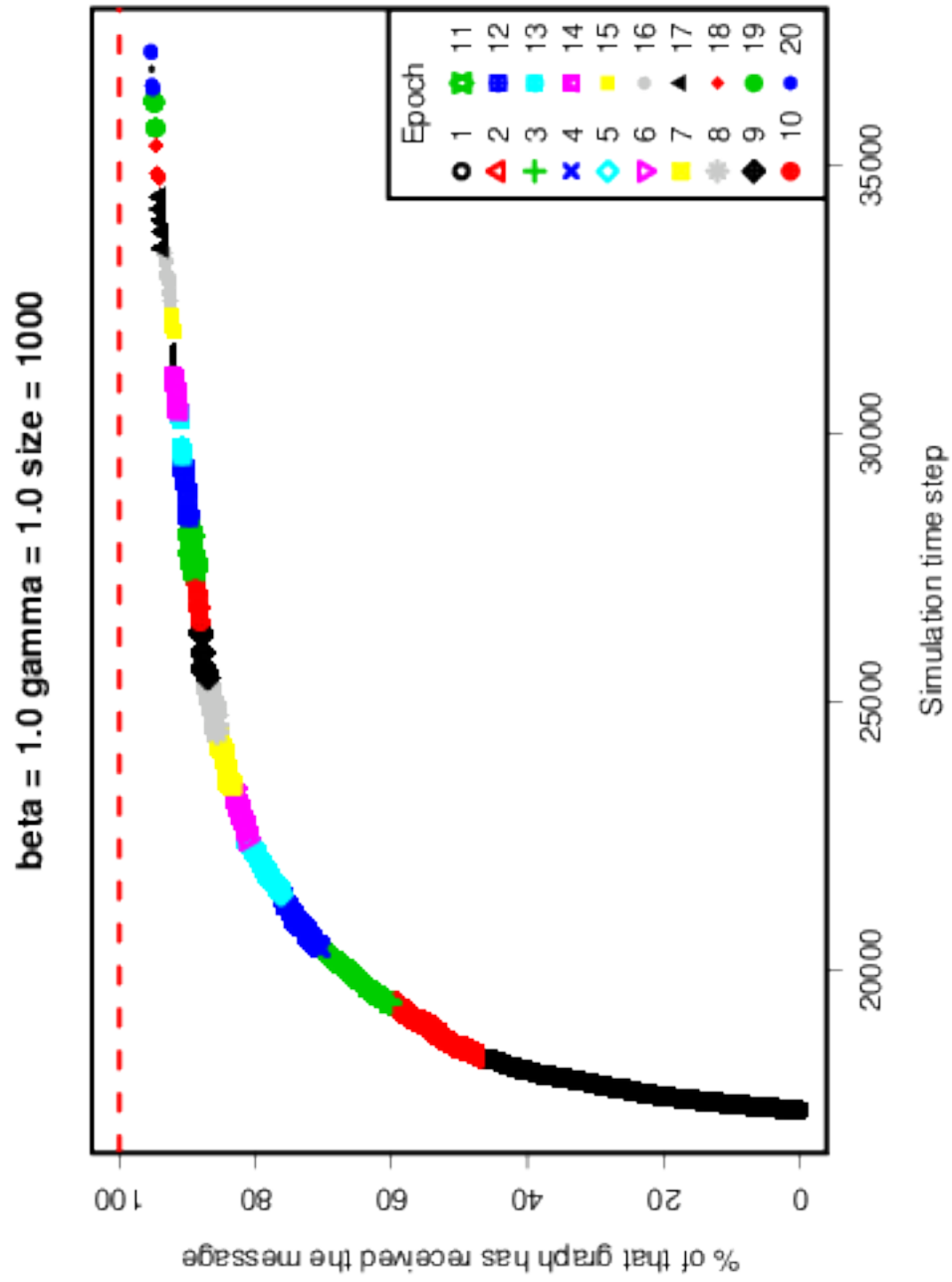


Figure 204. Degree biased selection, size = 1000 $\beta = 1.0$ $\gamma = 1.0$

APPENDIX H

USW DEGREE DISTRIBUTIONS

We examine and report the degree distributions for various USW graphs of order $L(G) = 100$. The graphs were created with different values for β and γ . Plots with midrange values of β and γ clearly show their random number origins.

0100-0.0000-1.0000-0.5000-1-3-5000.net

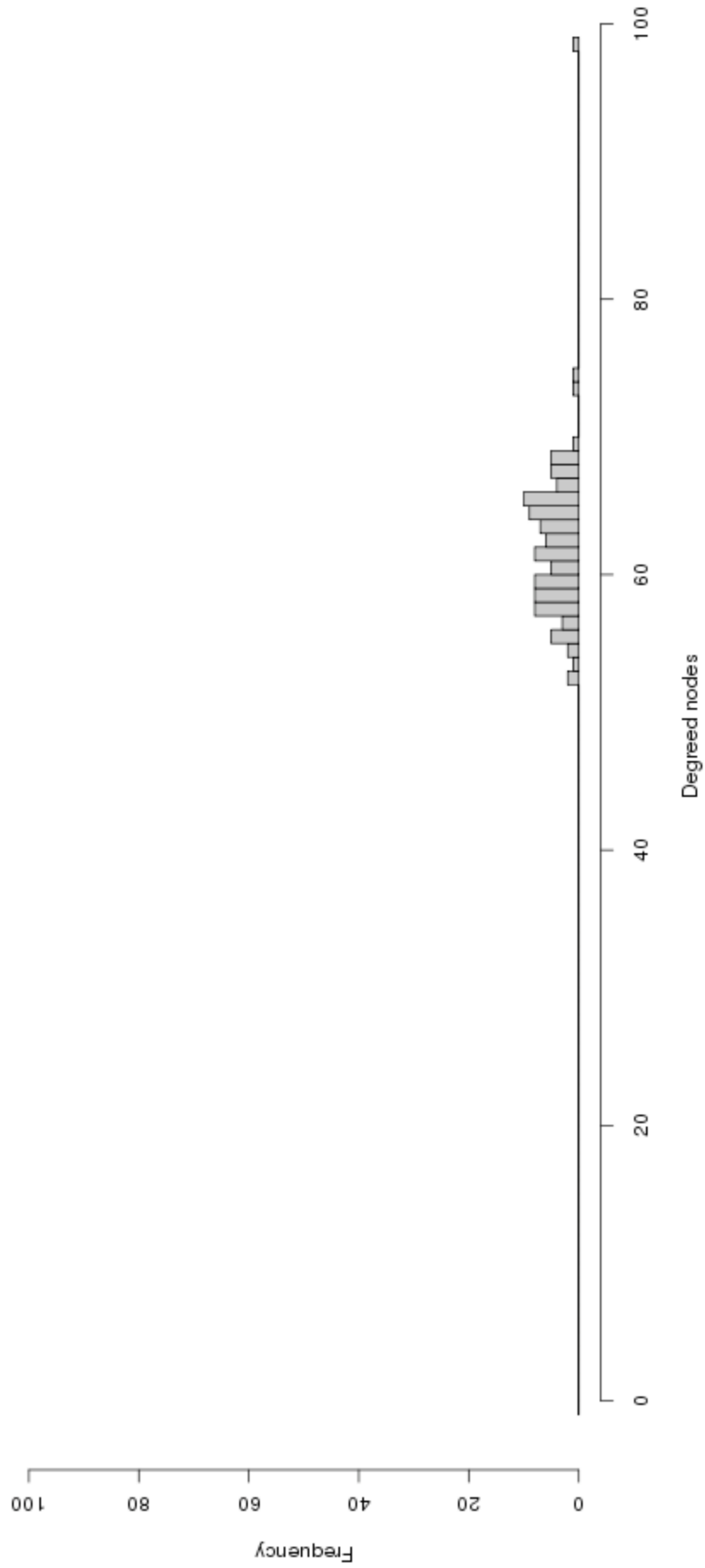


Figure 205. The degree distribution for $\beta = 0.0$ and $\gamma = 0.0$. Plots with midrange values of β and γ clearly show their random number origins.

0100-0.5000-1.0000-0.5000-1-3-5000.net

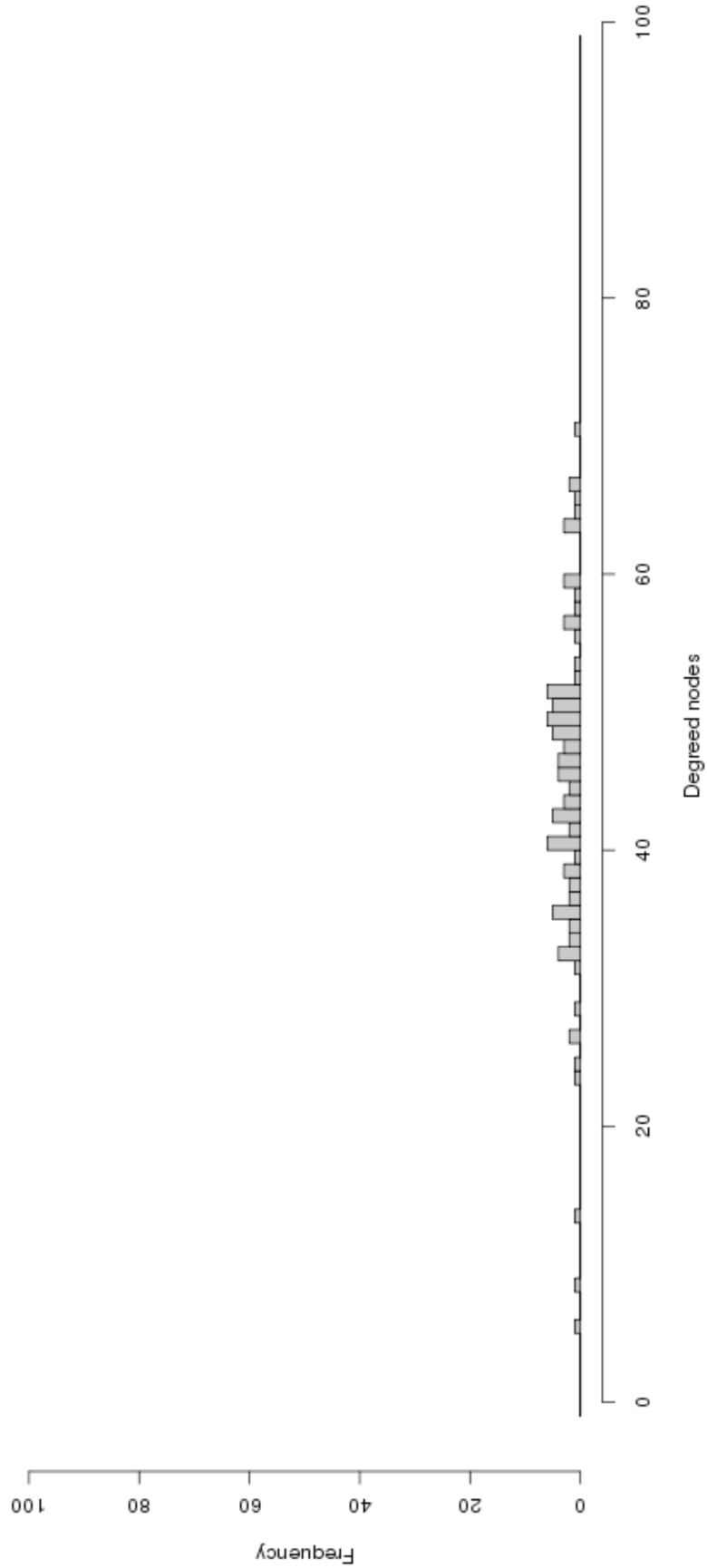


Figure 206. The degree distribution for $\beta = 0.5$ and $\gamma = 0.5$. Plots with midrange values of β and γ clearly show their random number origins.

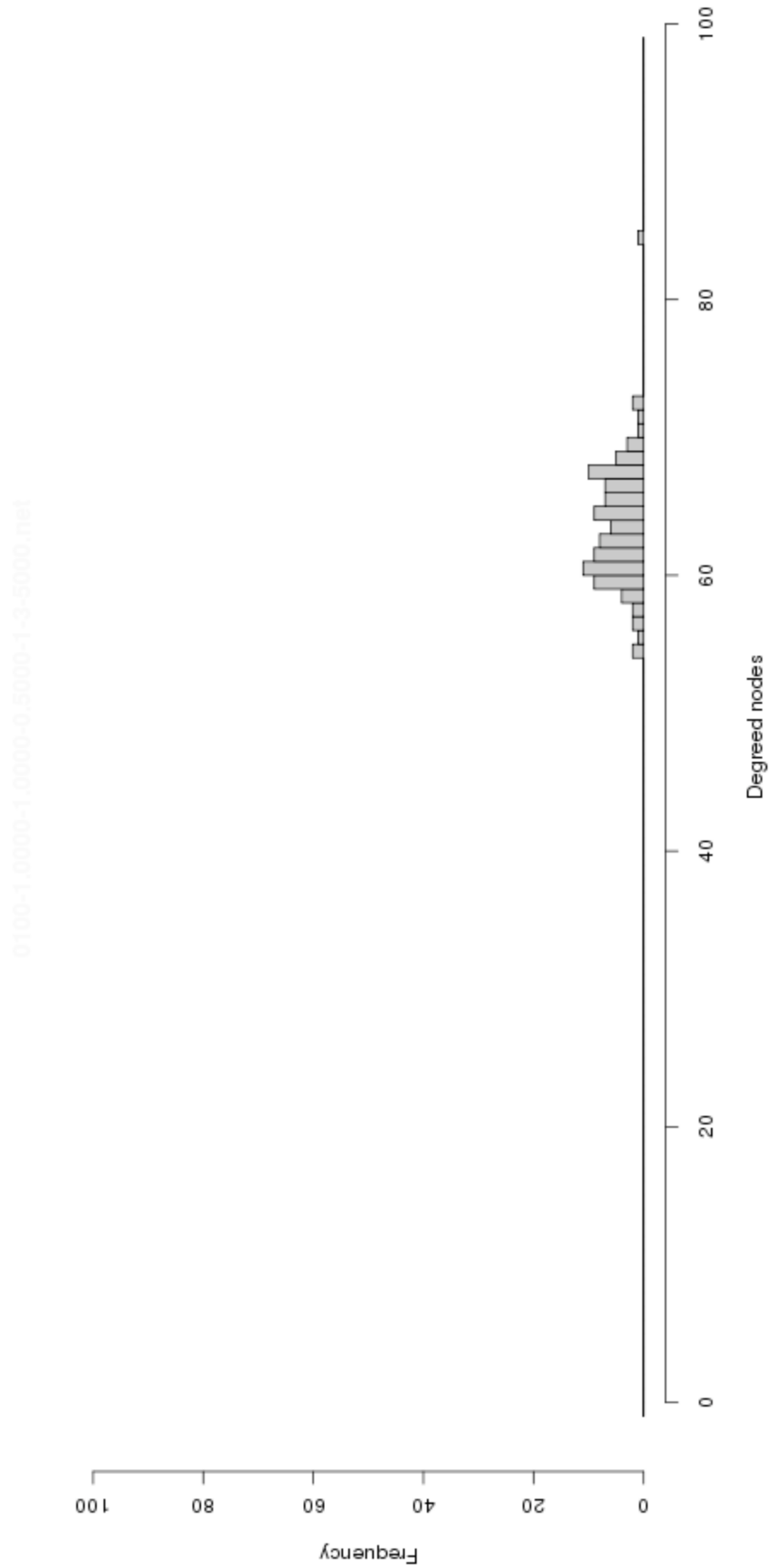


Figure 207. The degree distribution for $\beta = 1.0$ and $\gamma = 1.0$. Plots with midrange values of β and γ clearly show their random number origins.

0100-0.0000-0.5000-1-3-5000.net

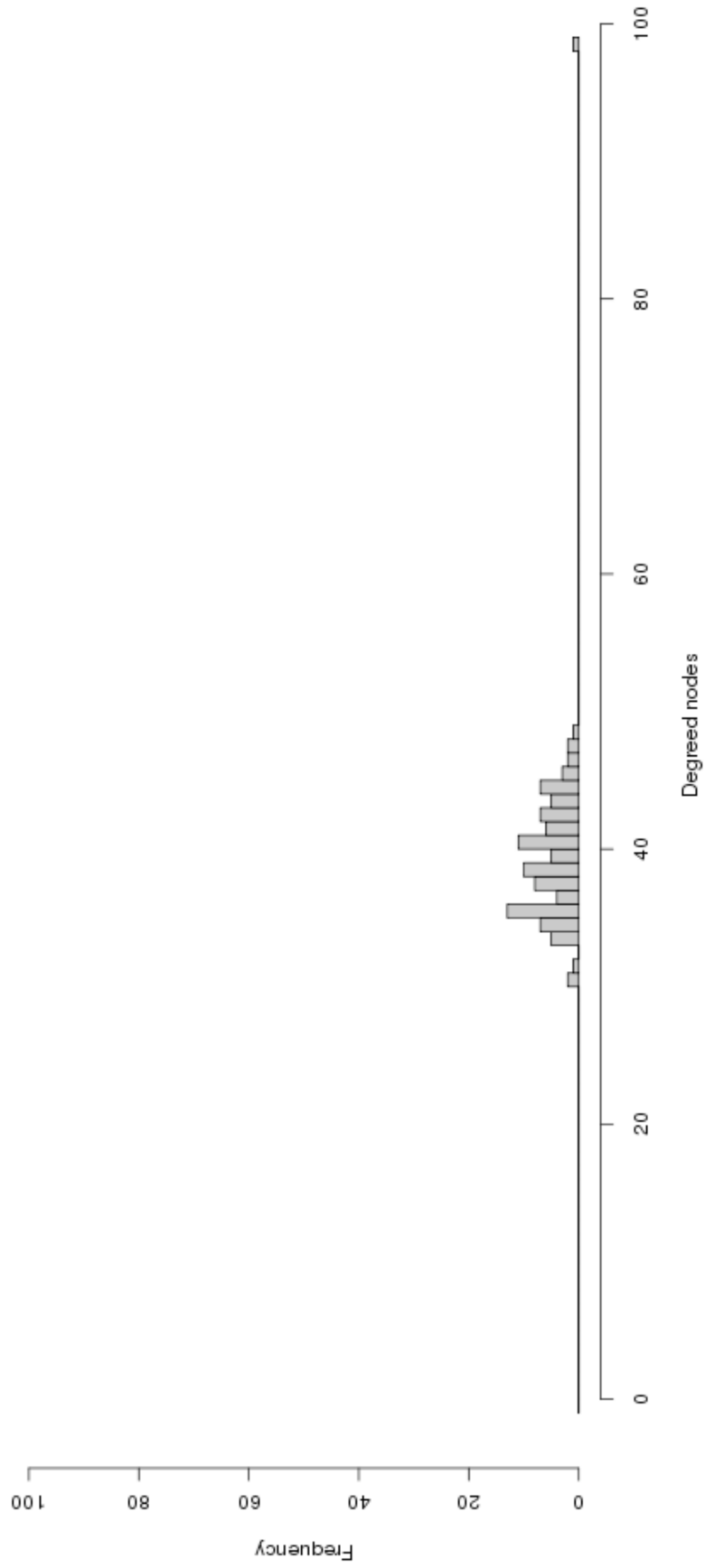


Figure 208. The degree distribution for $\beta = 0.0$ and $\gamma = 0.0$. Plots with midrange values of β and γ clearly show their random number origins.

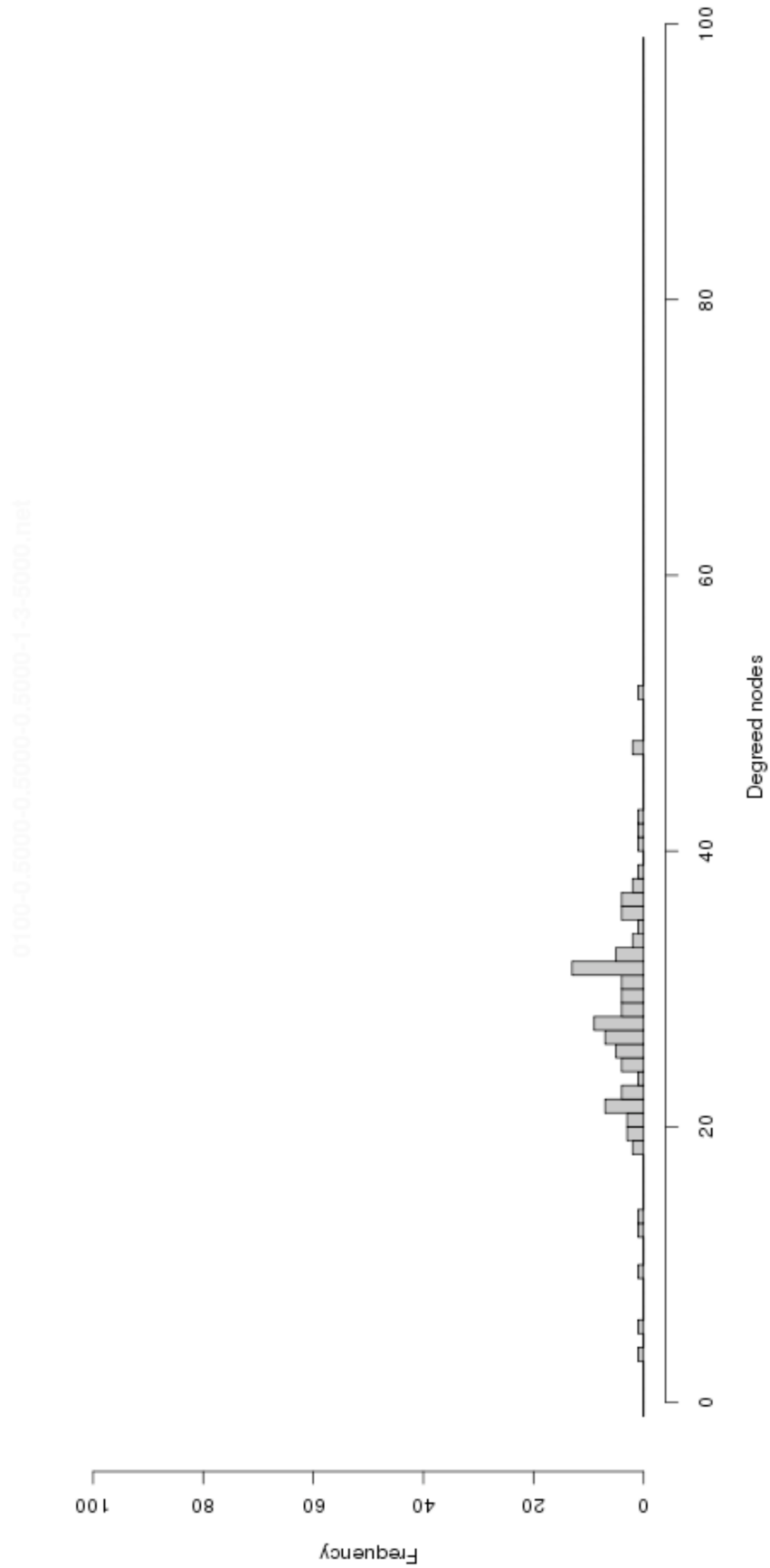


Figure 209. The degree distribution for $\beta = 0.5$ and $\gamma = 0.5$. Plots with midrange values of β and γ clearly show their random number origins.

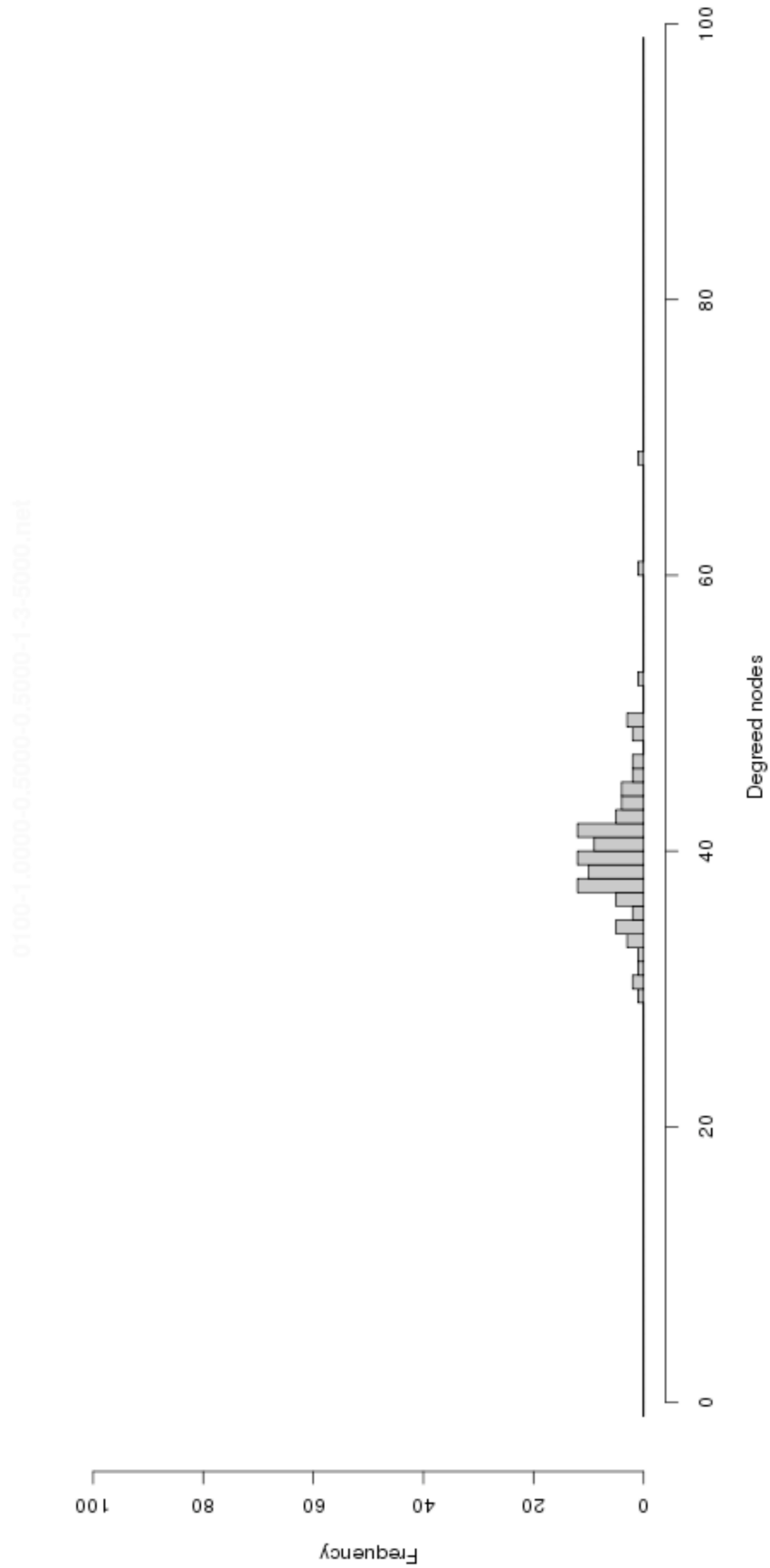


Figure 210. The degree distribution for $\beta = 1.0$ and $\gamma = 1.0$. Plots with midrange values of β and γ clearly show their random number origins.

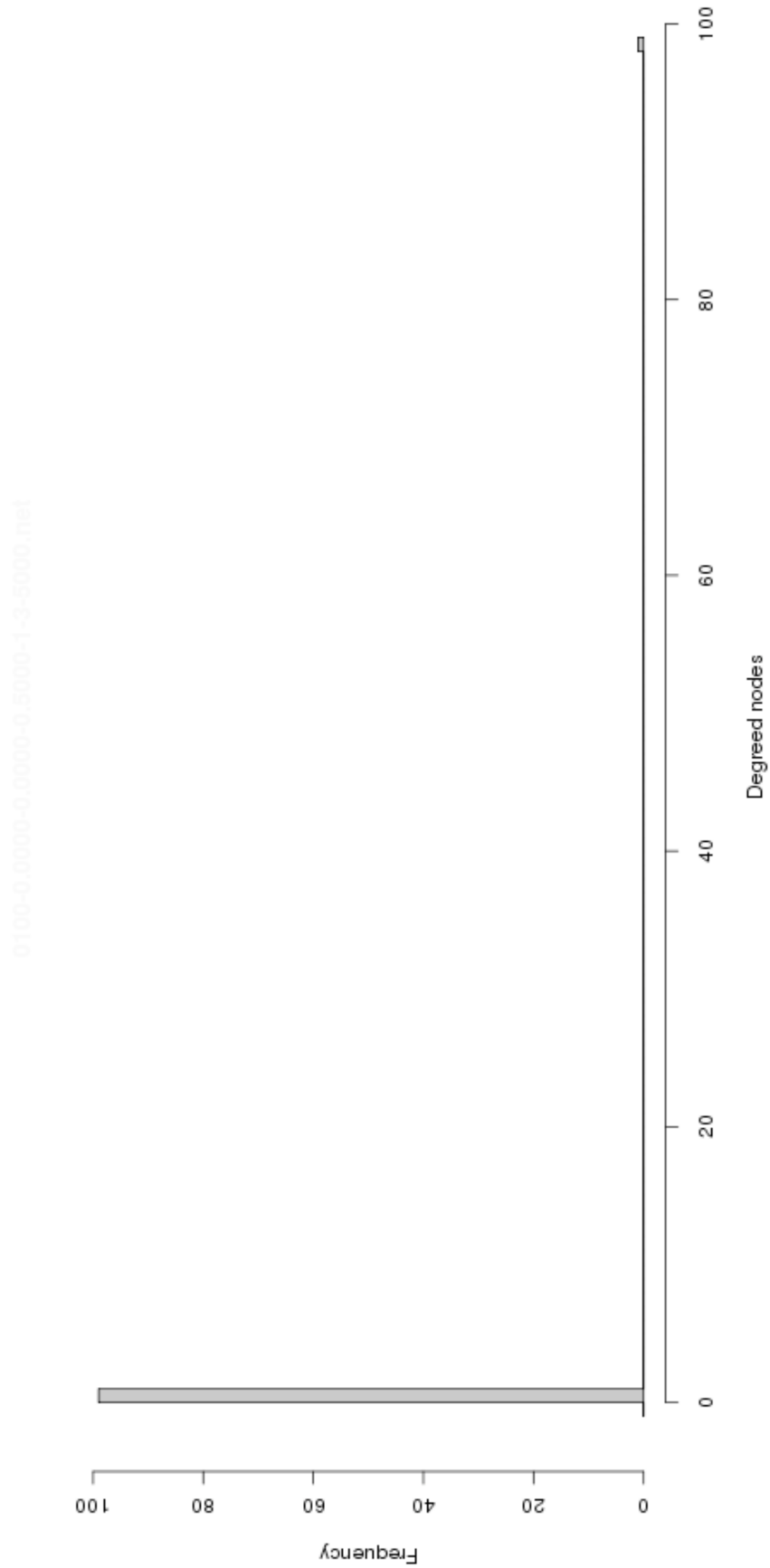


Figure 211. The degree distribution for $\beta = 0.0$ and $\gamma = 0.0$. Plots with midrange values of β and γ clearly show their random number origins.

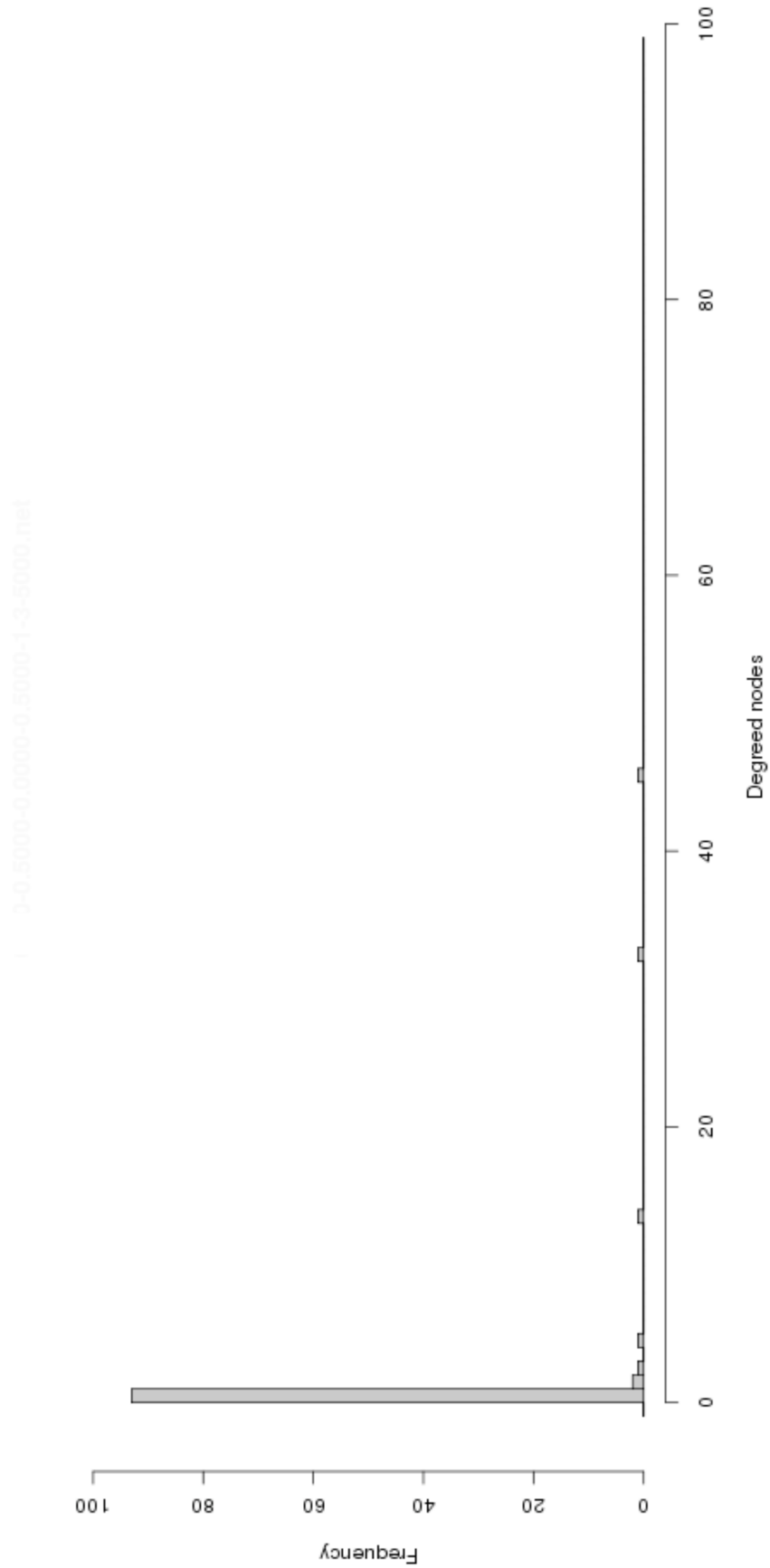


Figure 212. The degree distribution for $\beta = 0.5$ and $\gamma = 0.5$. Plots with midrange values of β and γ clearly show their random number origins.

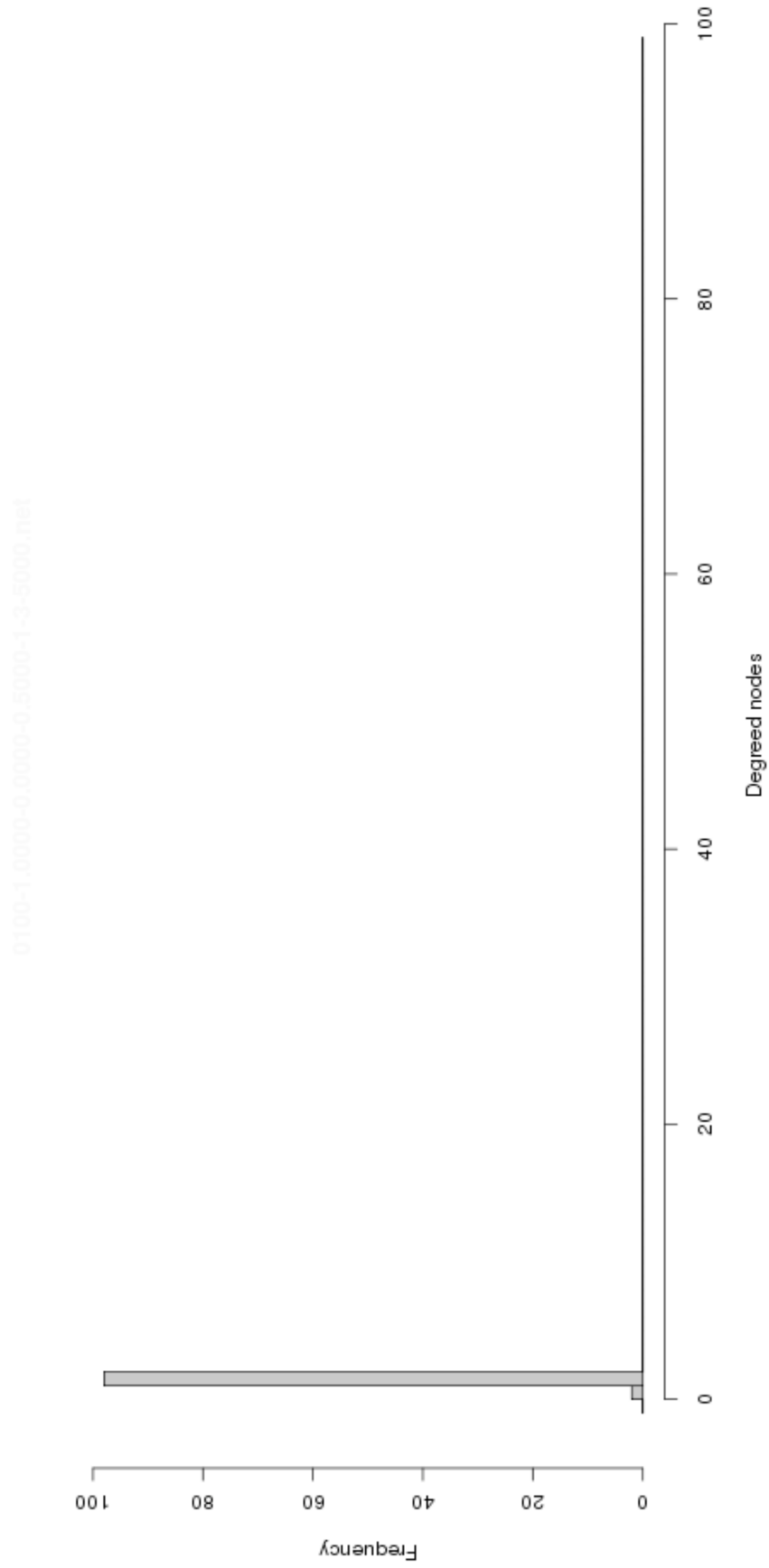


Figure 213. The degree distribution for $\beta = 1.0$ and $\gamma = 1.0$. Plots with midrange values of β and γ clearly show their random number origins.

APPENDIX I

USW DISCONNECTION

We examine and report the diameter $D(G)$ and the average path length $L(G)$ for various USW graphs of order $L(G) = 100$. The graphs were created with different values for β and γ . Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

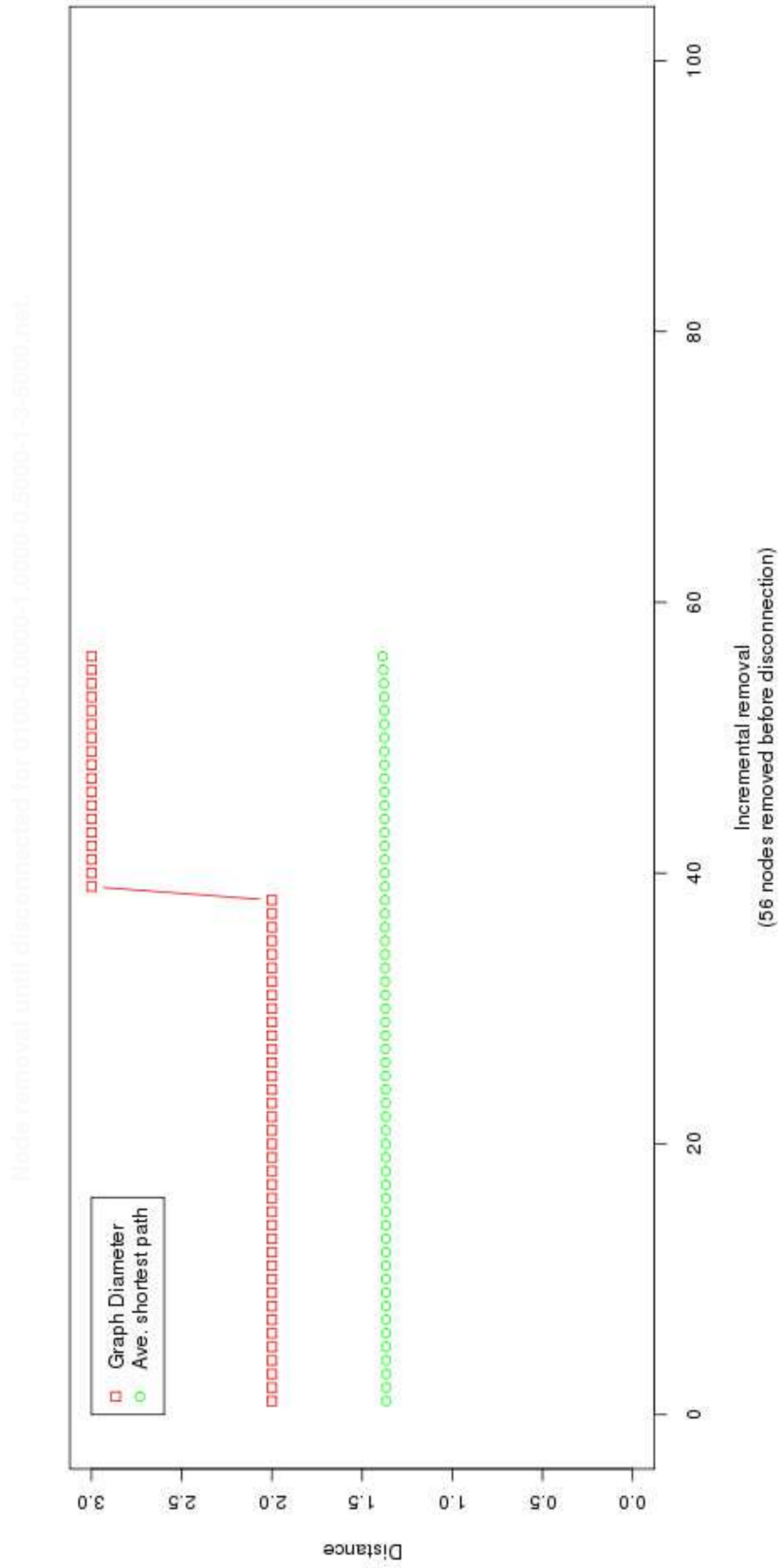


Figure 214. The disconnection plot for $\beta = 0.0$ and $\gamma = 0.0$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

Node removal until disconnected for 0100-0.5000-1.0000-1.5000-2.0000-2.5000-3.0000.net.

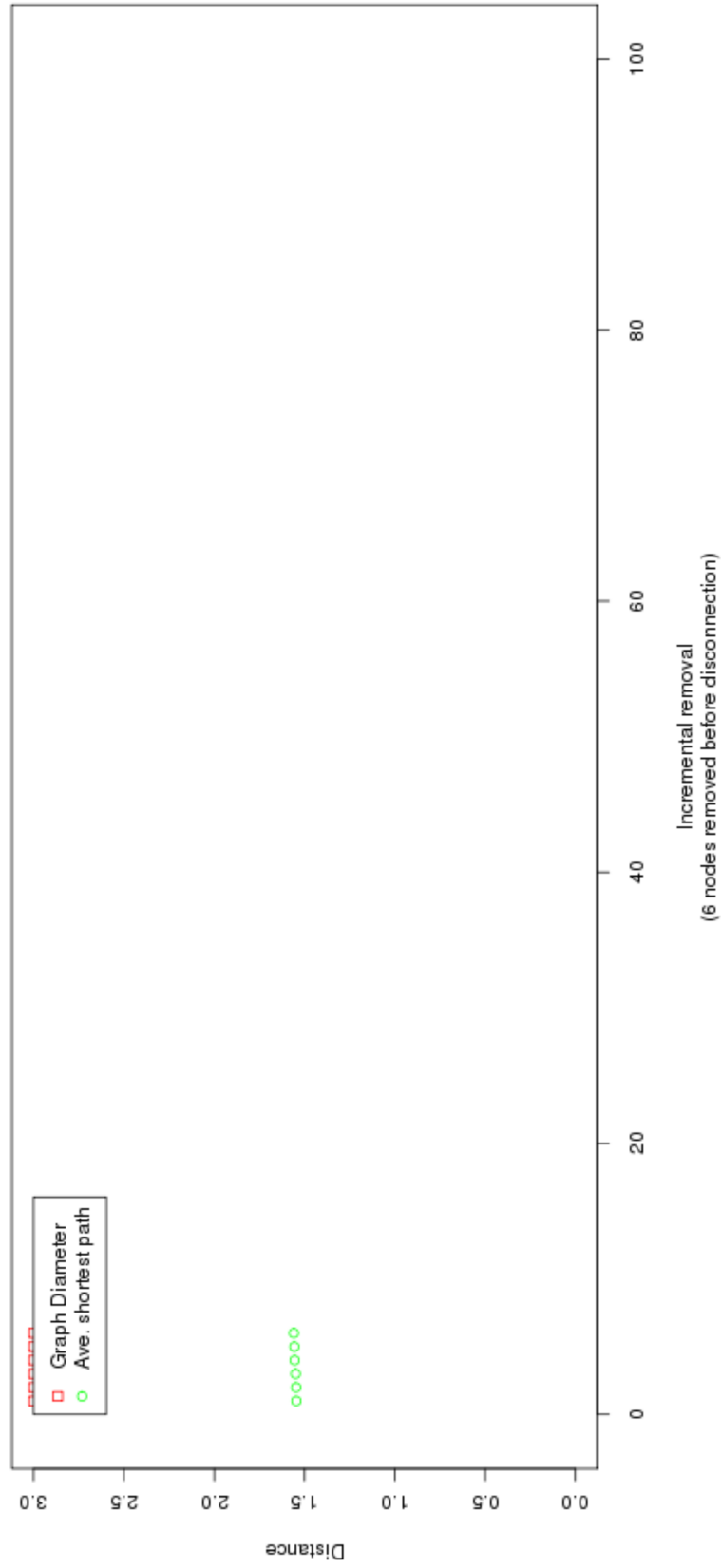


Figure 215. The disconnection plot for $\beta = 0.5$ and $\gamma = 0.5$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

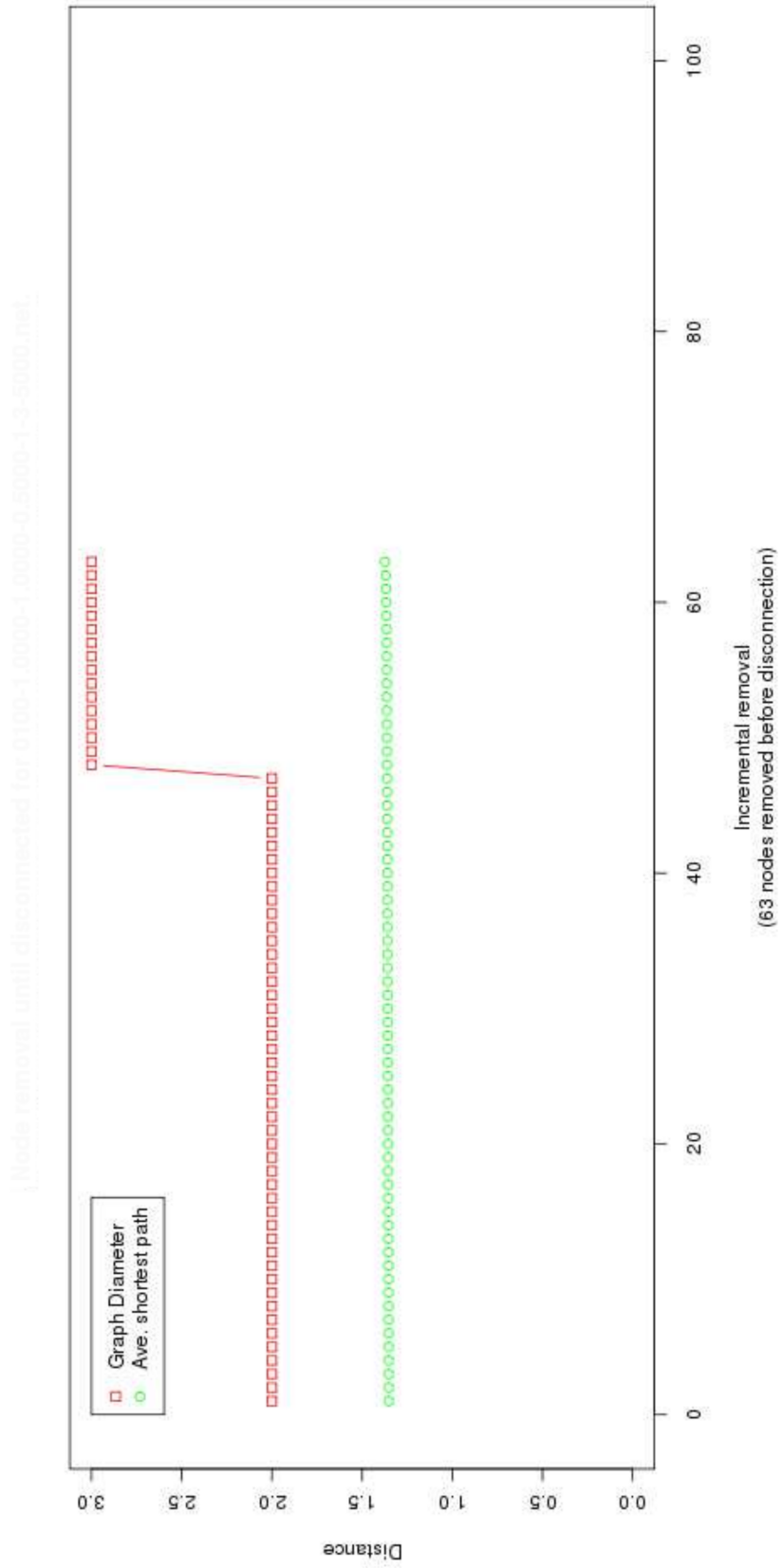


Figure 216. The disconnection plot for $\beta = 1.0$ and $\gamma = 1.0$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

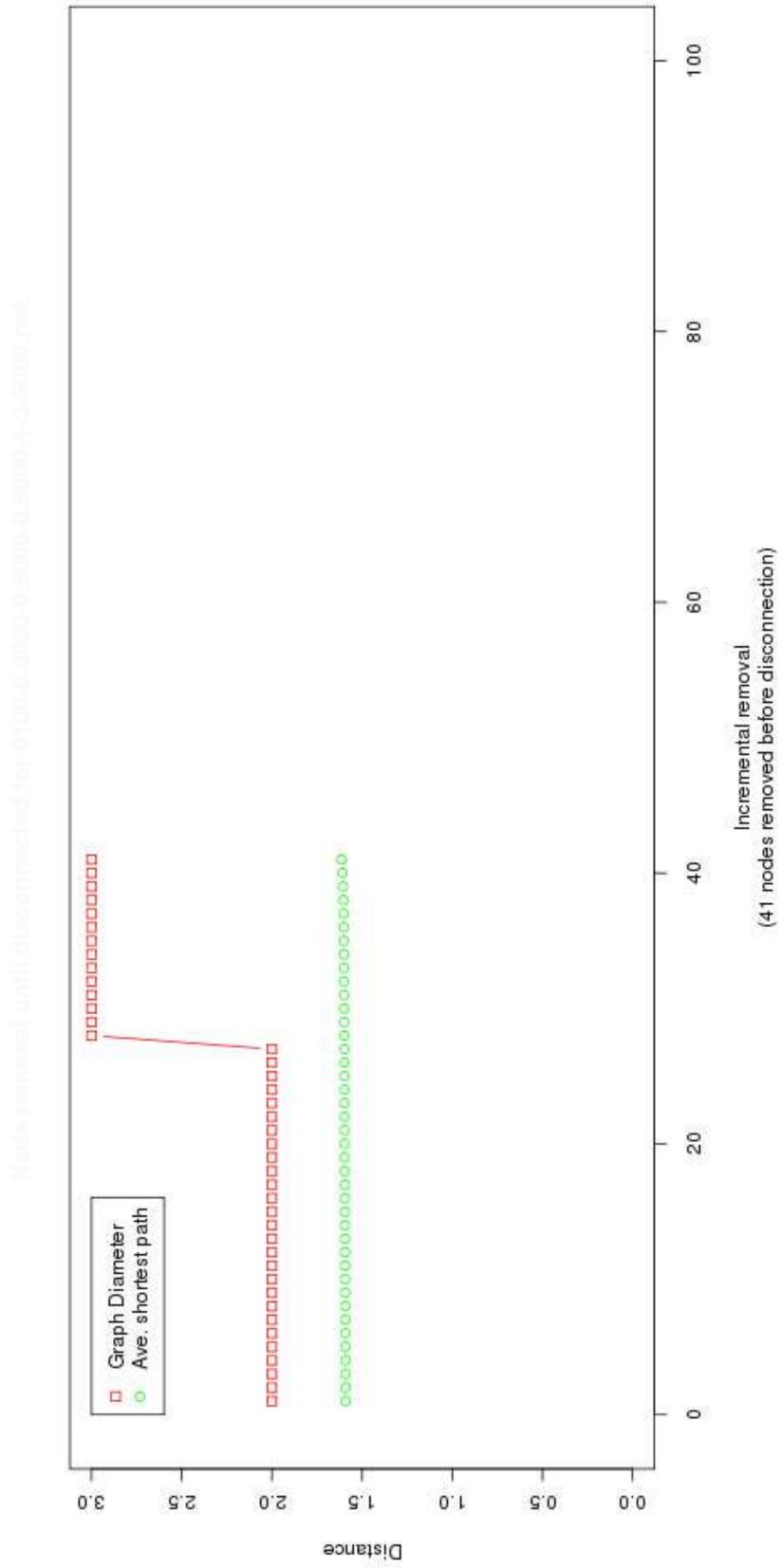


Figure 217. The disconnection plot for $\beta = 0.0$ and $\gamma = 0.0$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

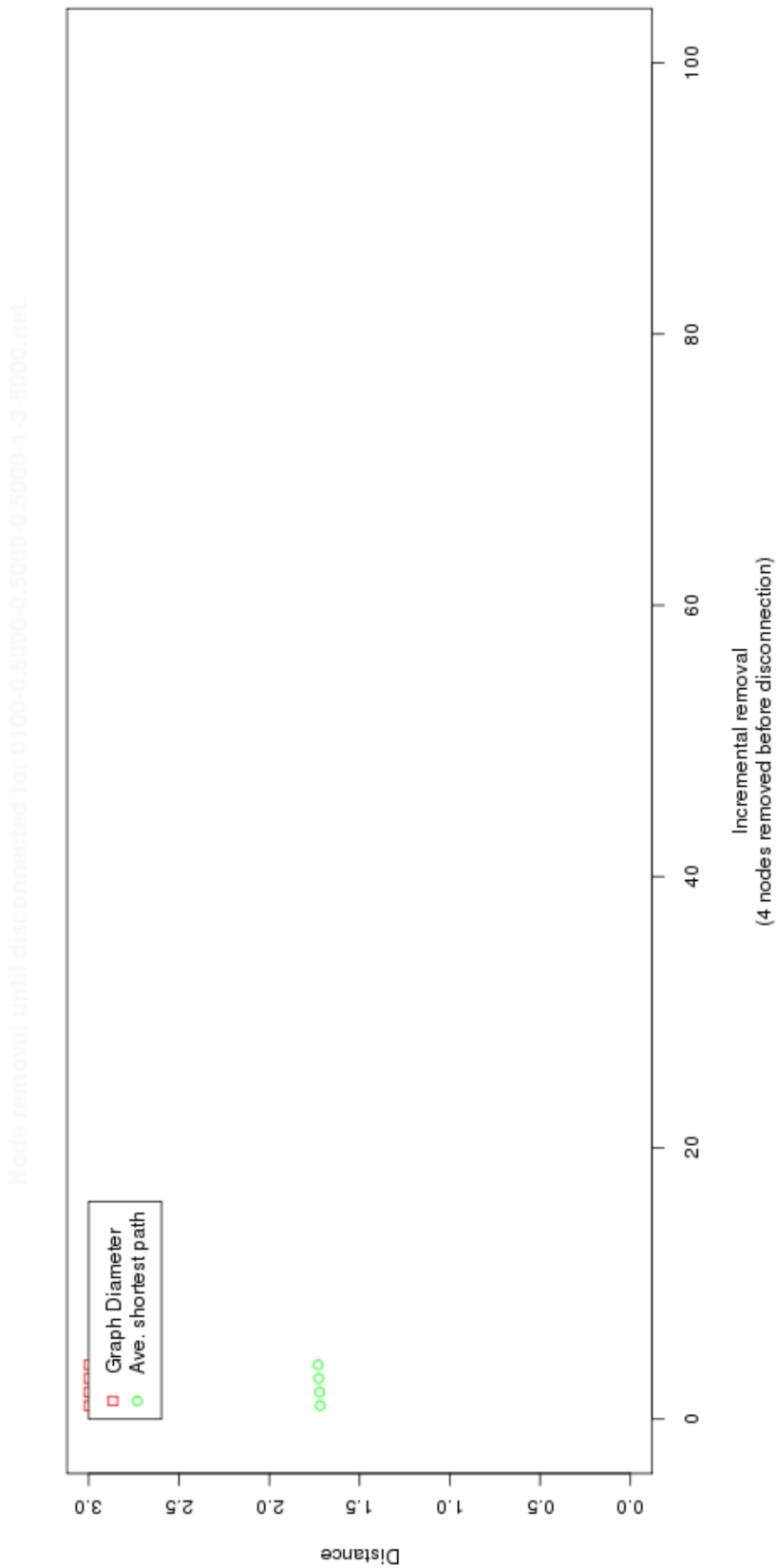


Figure 218. The disconnection plot for $\beta = 0.5$ and $\gamma = 0.5$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

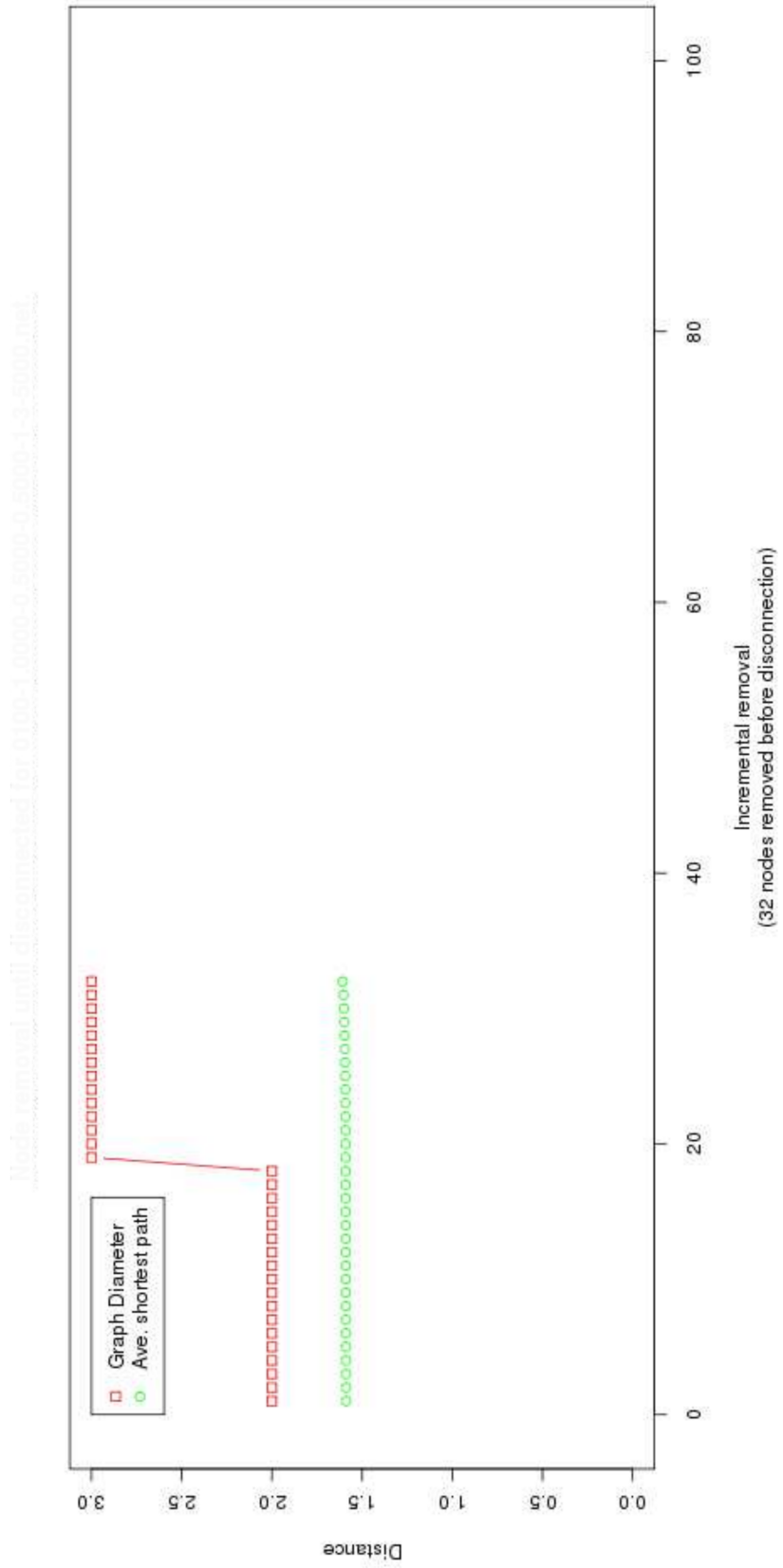


Figure 219. The disconnection plot for $\beta = 1.0$ and $\gamma = 1.0$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

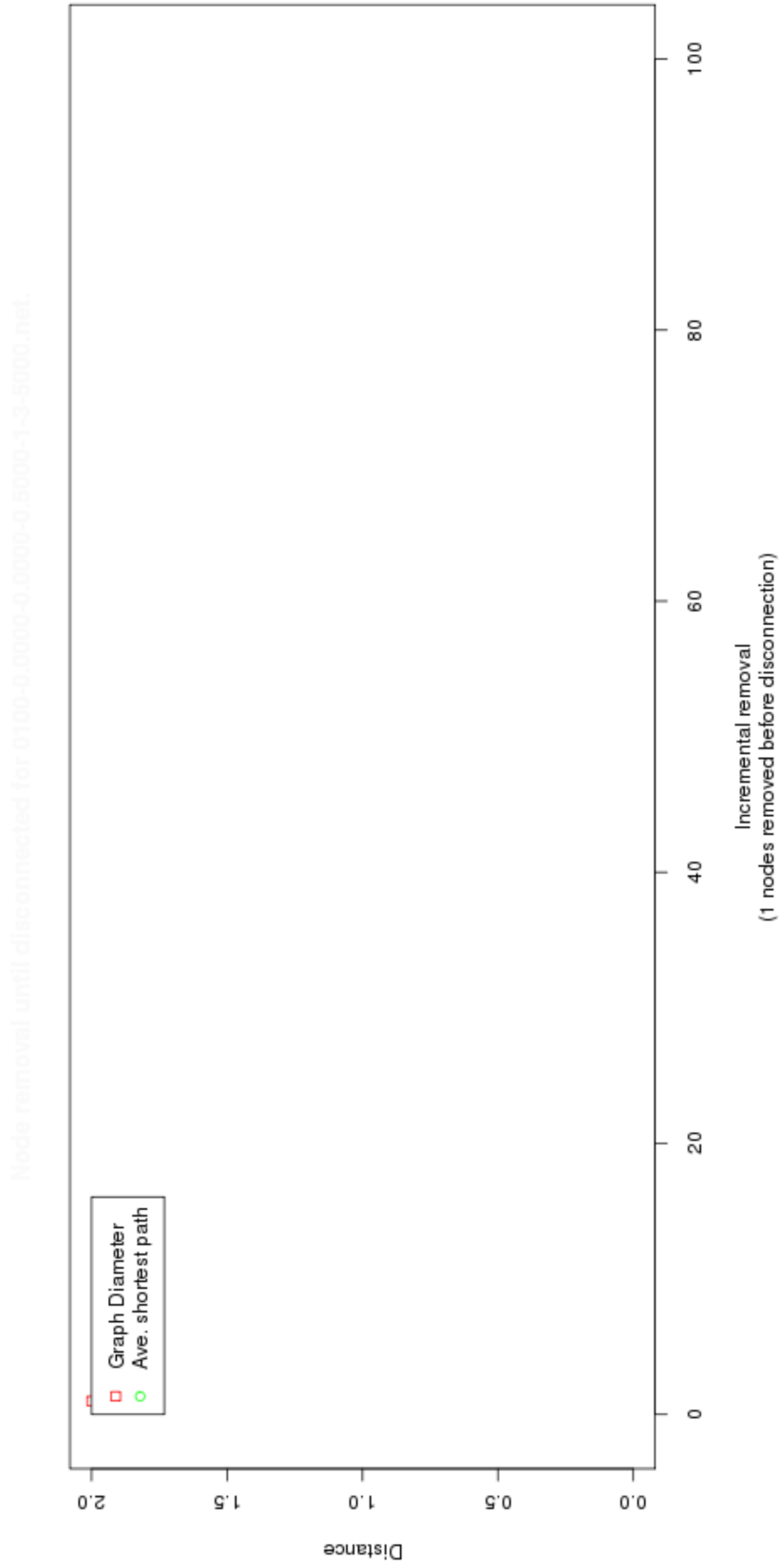


Figure 220. The disconnection plot for $\beta = 0.0$ and $\gamma = 0.0$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

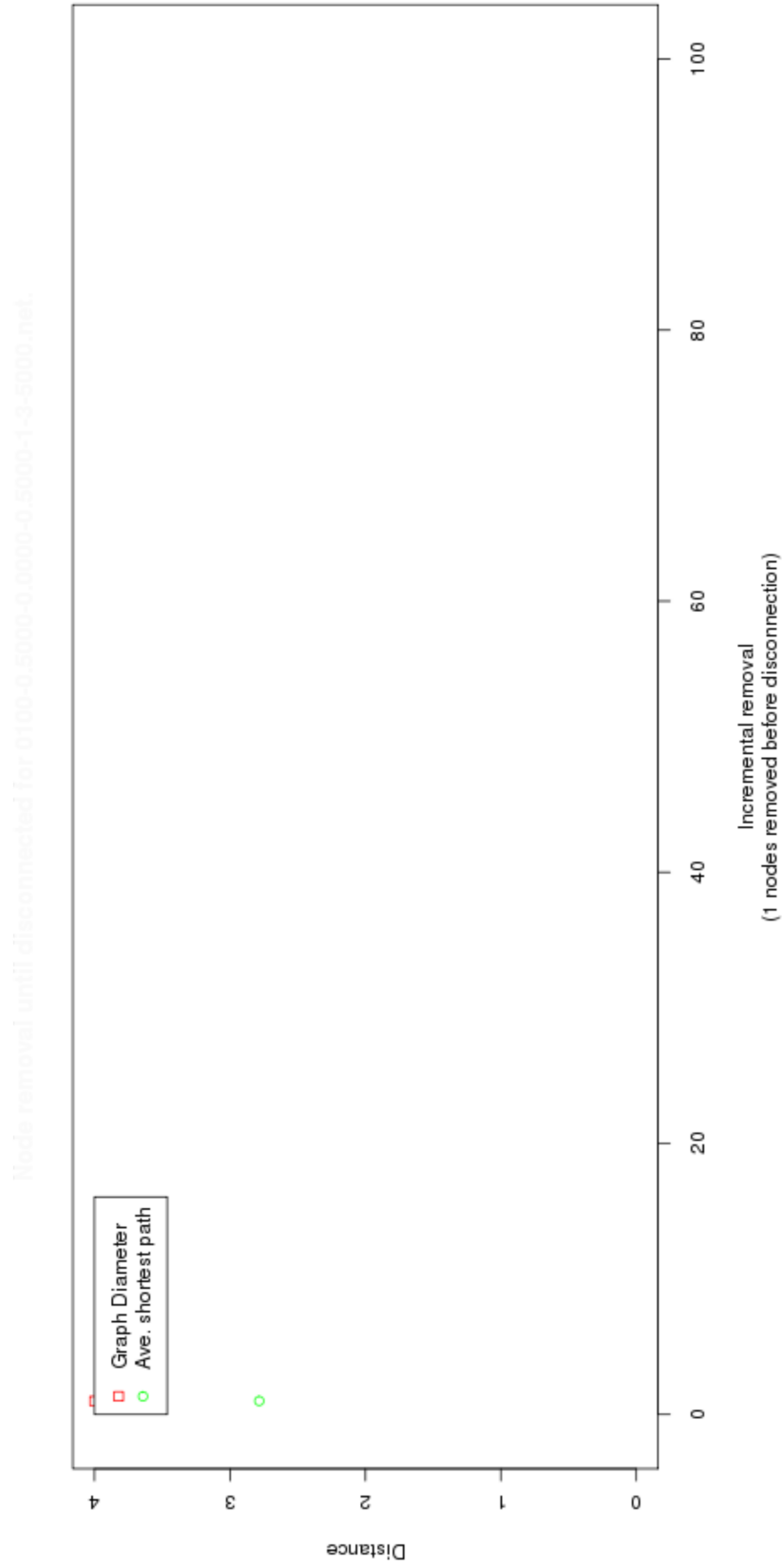


Figure 221. The disconnection plot for $\beta = 0.5$ and $\gamma = 0.5$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

Node removal until disconnected for 0100-1.0000-0.0000-0.5000-1-3-5000.net.

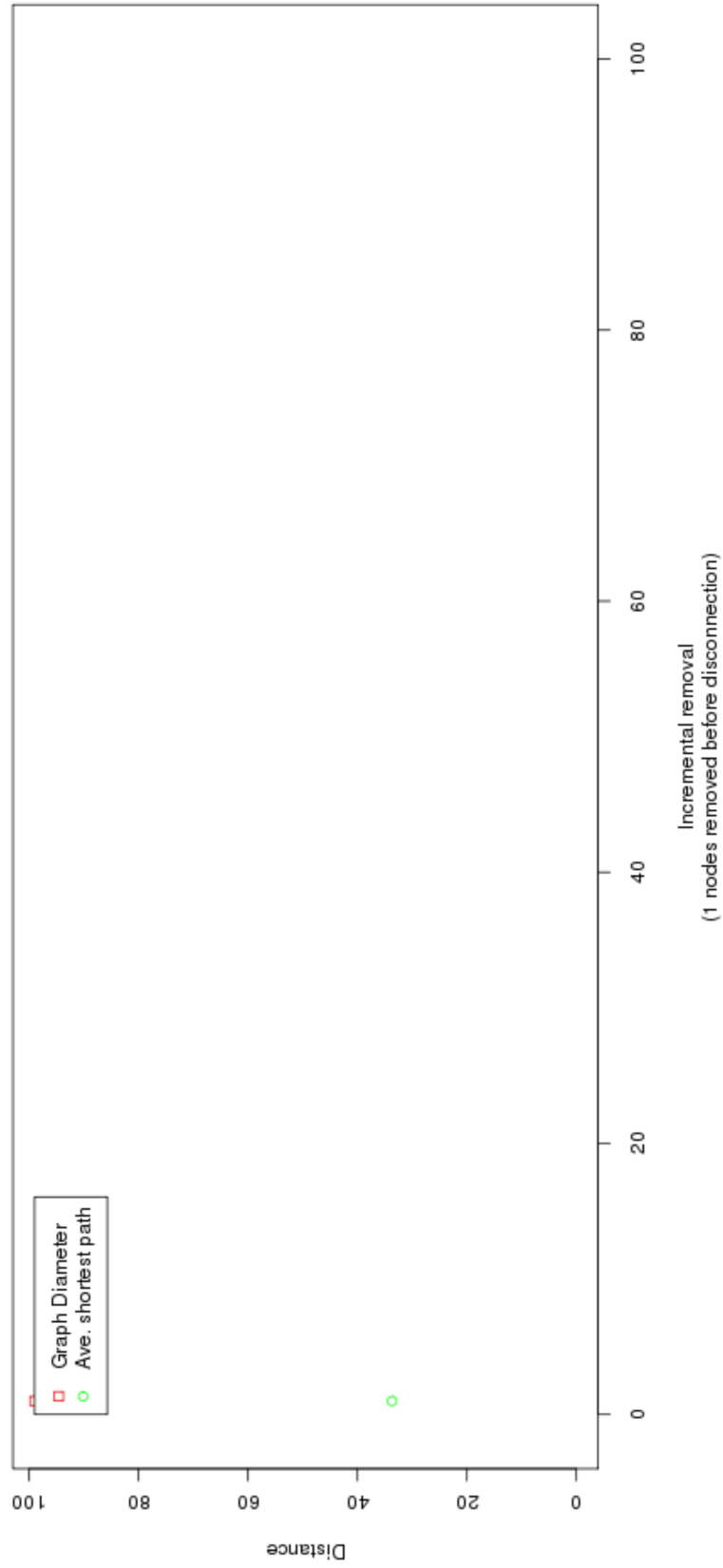


Figure 222. The disconnection plot for $\beta = 1.0$ and $\gamma = 1.0$. Plots with midrange values of β and γ clearly show their random number origins by being “short” relative to the length of similar Watts – Strogatz graphs.

APPENDIX J

USW PATH LENGTH HISTOGRAMS

We examine and report the path length $L(G)$ distributions for various USW graphs of order $L(G) = 10$ (Table 56 on the following page). The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

Table 56. The effect on $C(G)$ and $L(G)$ based on various values for β and γ . As γ grows to 1, the average path length becomes much shorter

β	γ	$C(G)$	$L(G)$
0.0	1.0	0.741	1
0.5	1.0	0.718	1
1.0	1.0	0.732	1
0.0	0.5	0.739	1
0.5	0.5	0.699	2
1.0	0.5	0.479	1
0.0	0.0	0.000	2
0.5	0.0	0.000	2
1.0	0.0	0.000	4

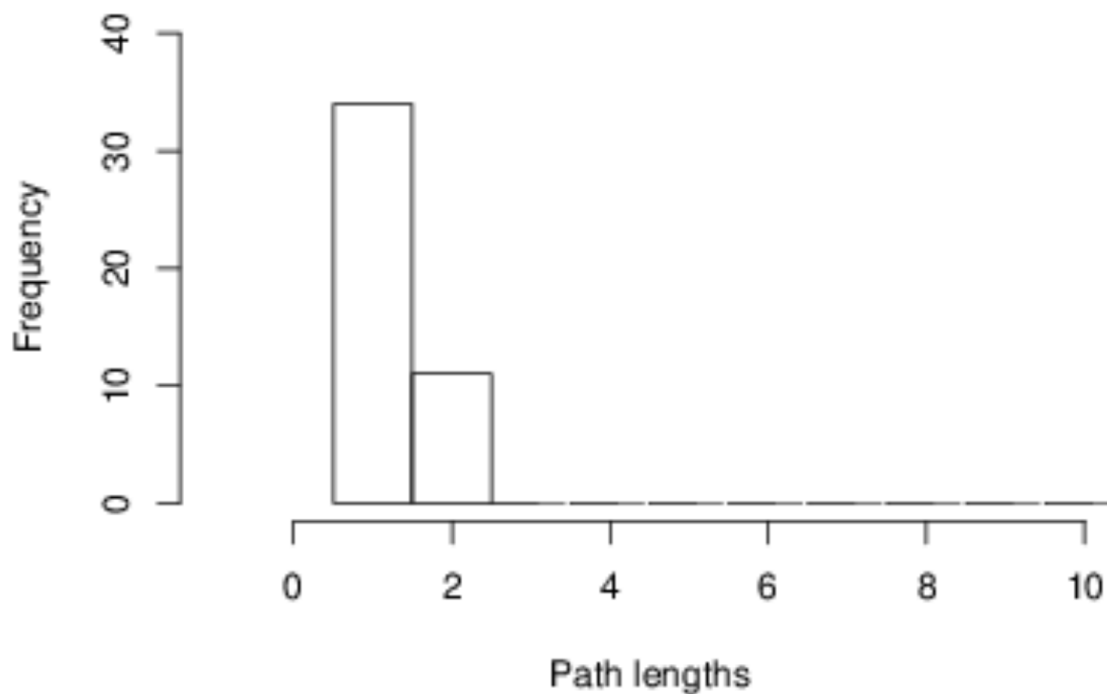


Figure 223. The path length histogram for $\beta = 0.0$ and $\gamma = 0.0$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

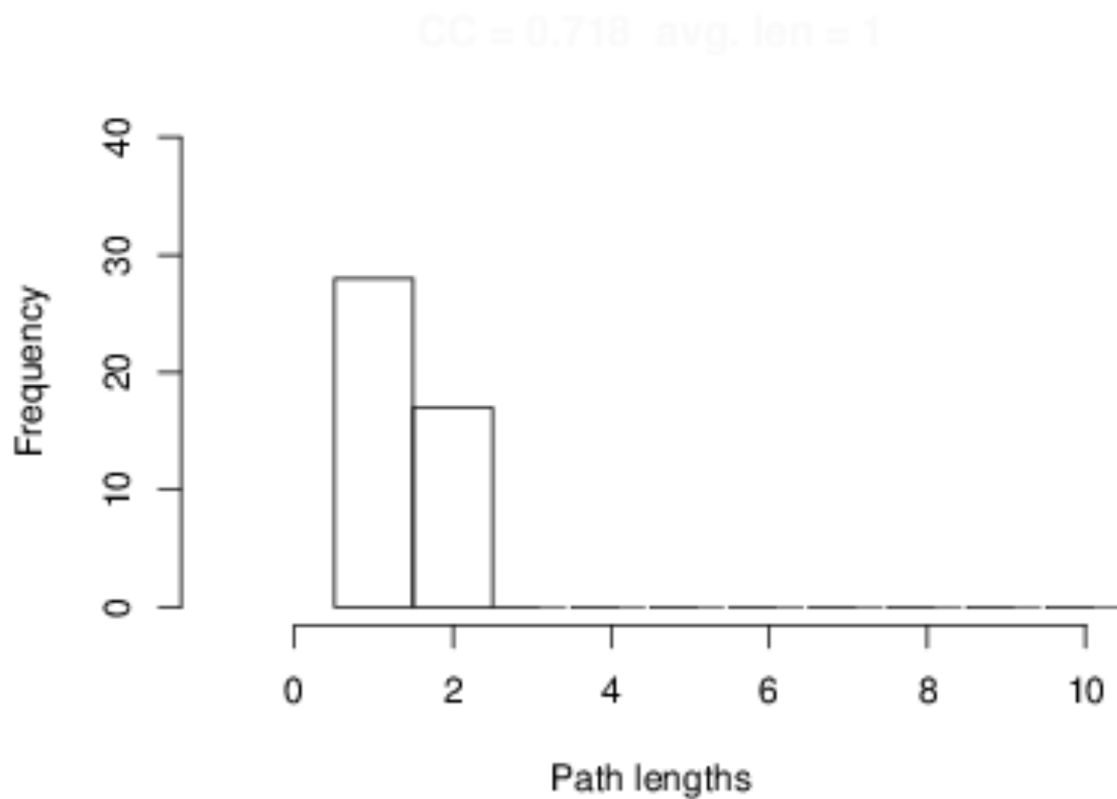


Figure 224. The path length histogram for $\beta = 0.5$ and $\gamma = 0.5$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

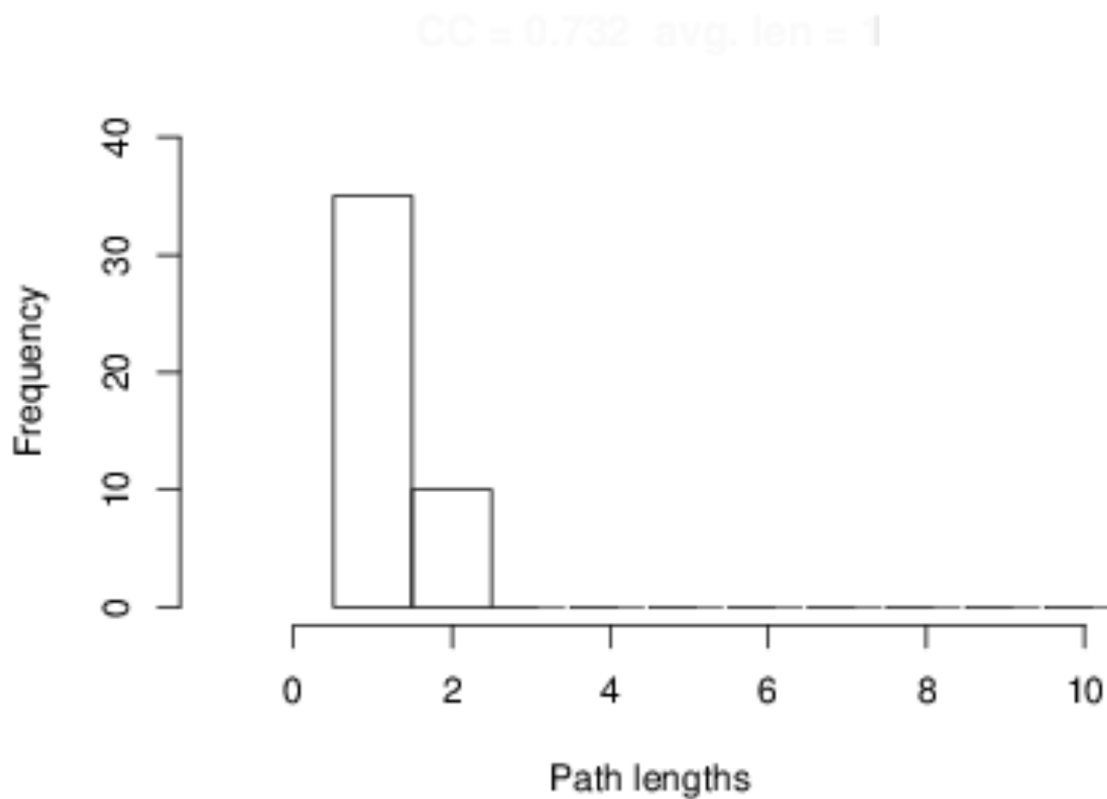


Figure 225. The path length histogram for $\beta = 1.0$ and $\gamma = 1.0$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

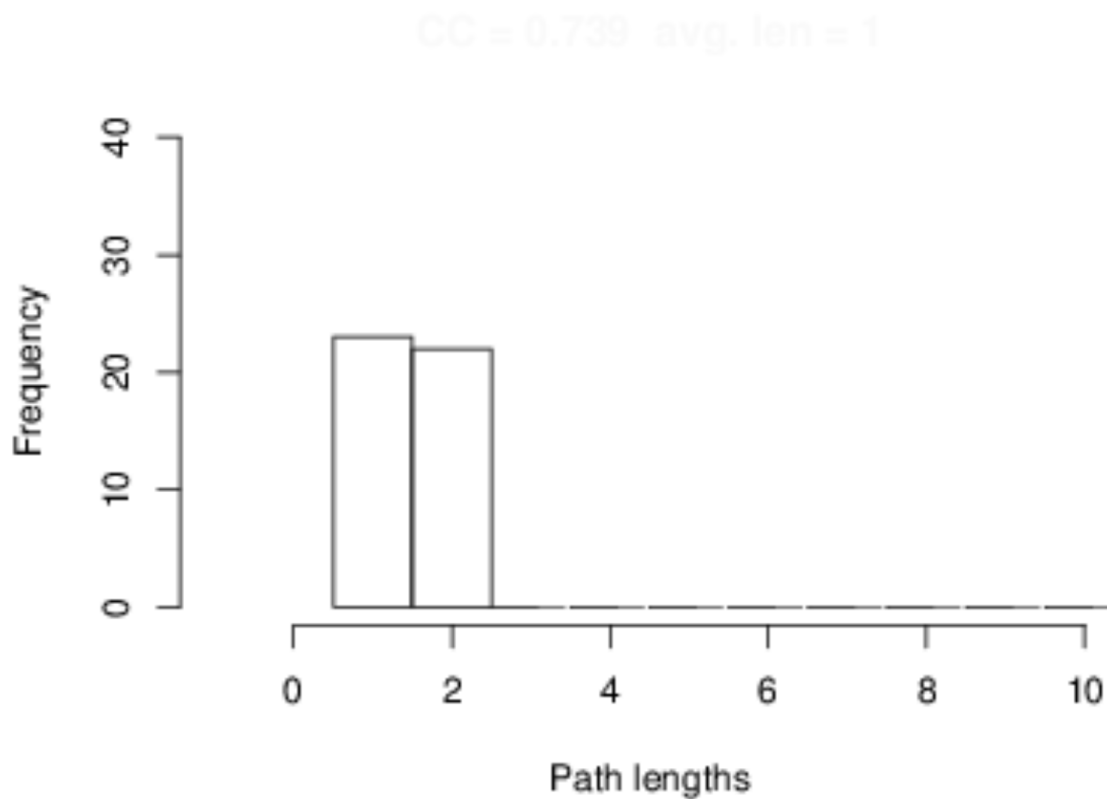


Figure 226. The path length histogram for $\beta = 0.0$ and $\gamma = 0.0$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

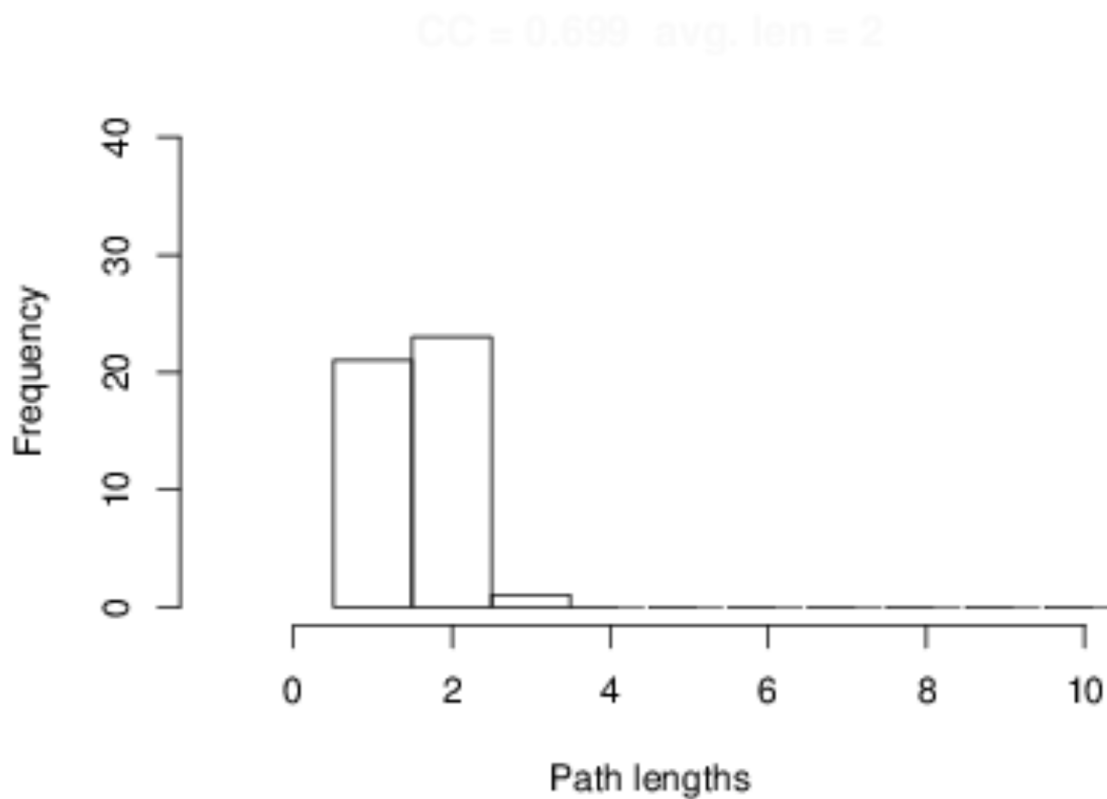


Figure 227. The path length histogram for $\beta = 0.5$ and $\gamma = 0.5$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

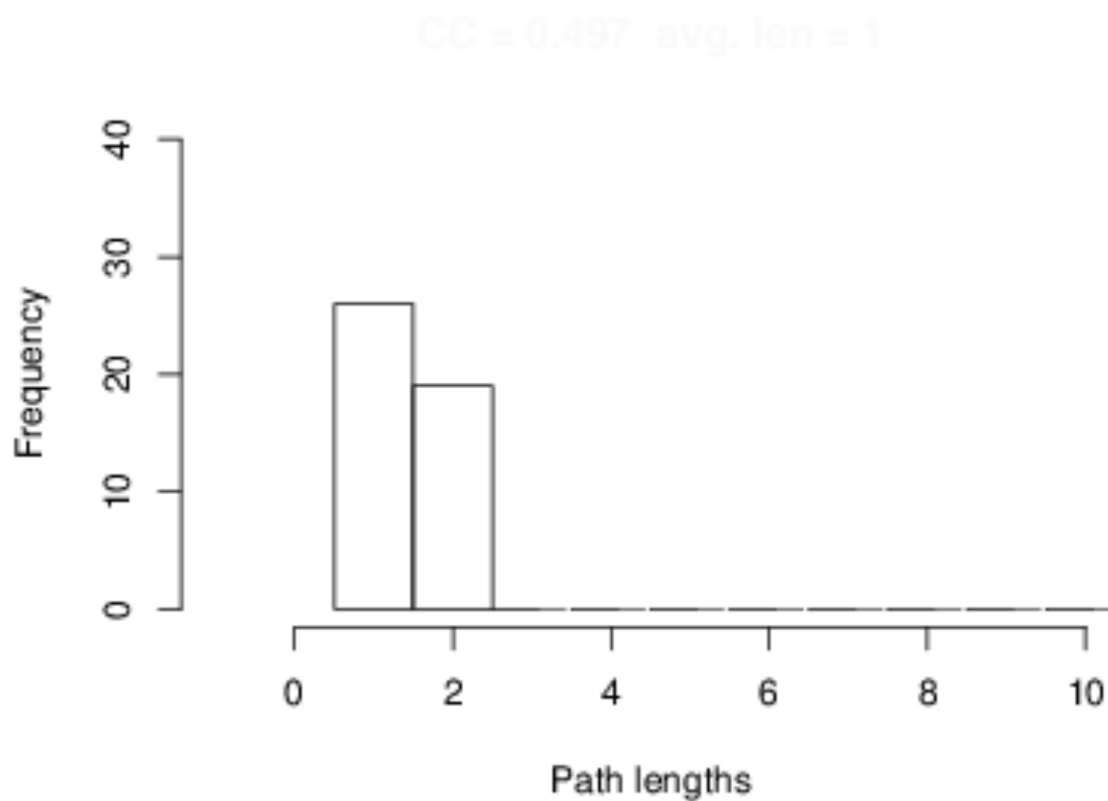


Figure 228. The path length histogram for $\beta = 1.0$ and $\gamma = 1.0$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

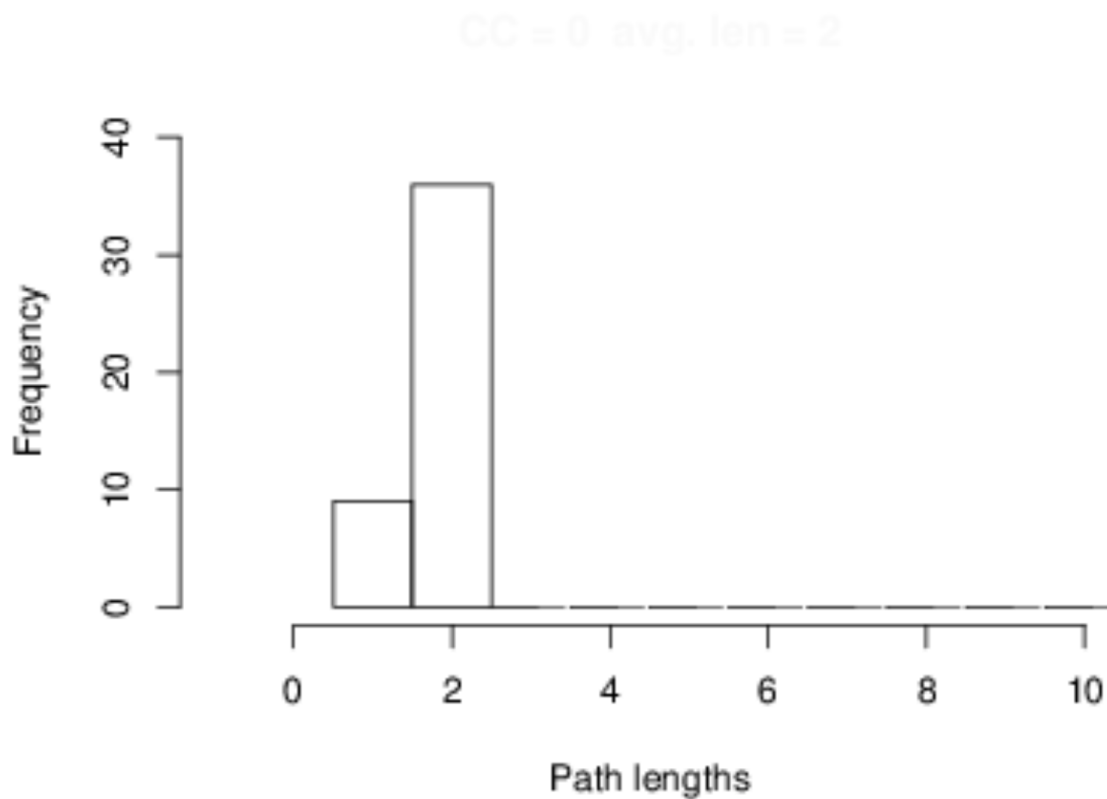


Figure 229. The path length histogram for $\beta = 0.0$ and $\gamma = 0.0$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

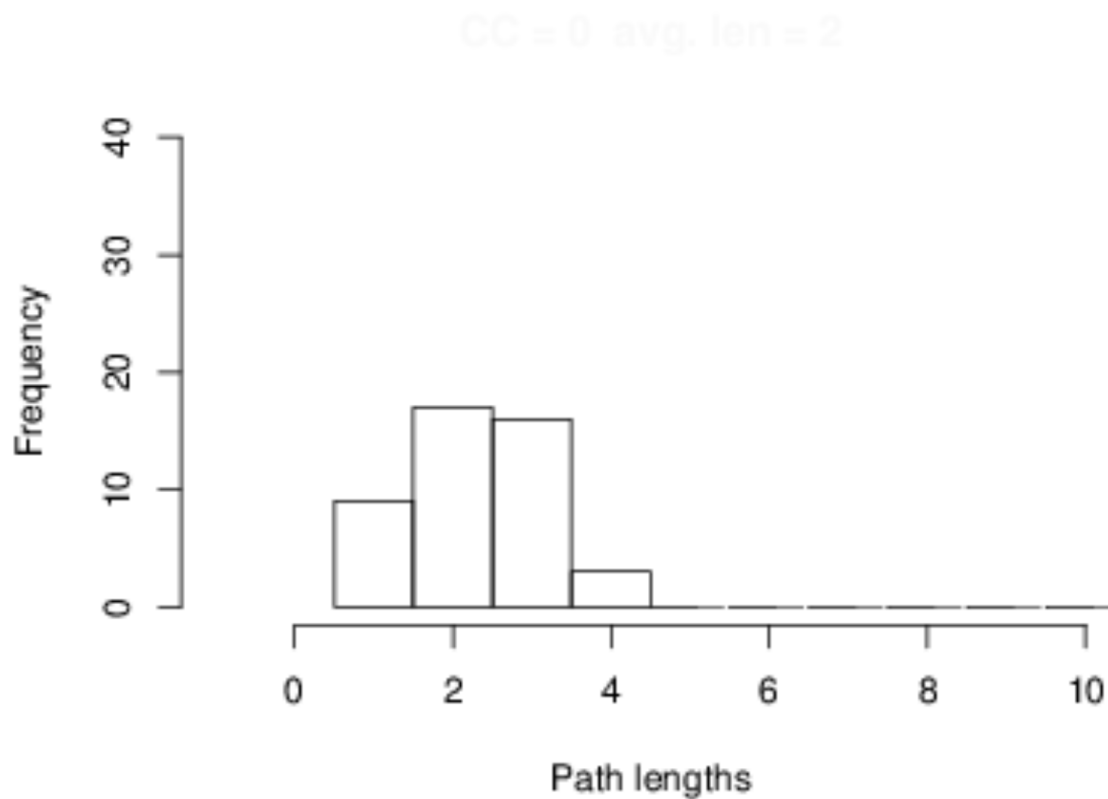


Figure 230. The path length histogram for $\beta = 0.5$ and $\gamma = 0.5$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

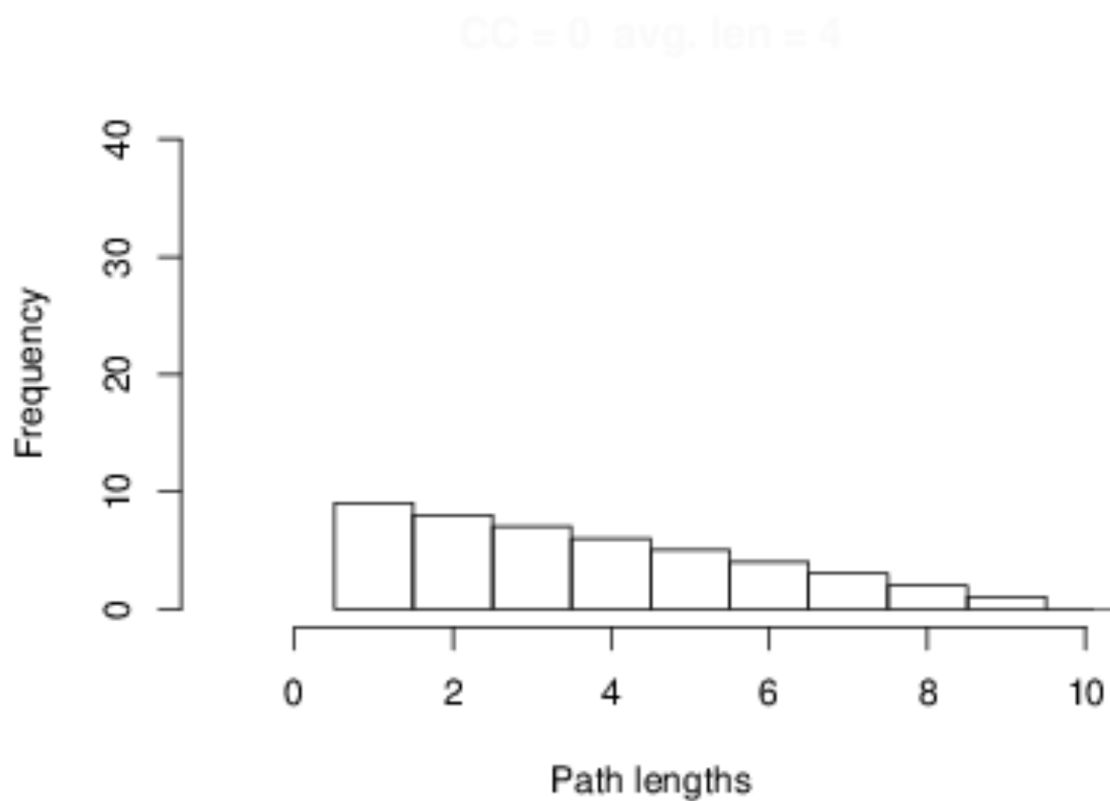


Figure 231. The path length histogram for $\beta = 1.0$ and $\gamma = 1.0$. shows the effect of varying β and γ on a USW graph of size 10. The axes are the same on all plots, so that a visual comparison of each histogram is facilitated. As γ grows to 1, the average path length becomes much shorter.

APPENDIX K

USW SYSTEM DESIGN CONSIDERATIONS

K.1 INTRODUCTION

Based on analysis of the preservation needs of the USW WOs and the assumed capacity of hosts to provide storage capacity for foreign WOs, we developed a set of equations that provide insight into USW preservation efficacy based on preservation policy.

K.2 DISCUSSION

The following set of equations relate to designing and predicting the preservation performance of the system:

$$h_{\min} = p + 1 \quad (132)$$

$$h_{\text{cap}} = \frac{|WO| * c_{\text{hard}}}{\min(h_{\min}, h)} \quad (133)$$

$$S_{\text{cap}} = h_{\text{cap}} * \min(|WO|, h) \quad (134)$$

$$h_{\max} = D \quad (135)$$

The absolute minimum number of hosts that need to be in the system in order for there to be any likelihood of preservation activity is based constrained by the fact that a preservation copy can not exist on the same host as the parent copy.

A WO's family members will be spread across a collection of hosts. A complete description of a WO's position in a family structure and the host that it is living on is given by $WO_{n,c,h}$. Where:

$$limits = \begin{cases} c_{\text{soft}} = \text{min. preservation copies} \\ c_{\text{hard}} = \text{max. preservation copies} \\ n_{\text{max}} = \text{max. WOs} \\ h_{\text{max}} = \text{max. hosts} \end{cases}$$

$$n, c, h \text{ constrained to: } \begin{cases} n = 1, \dots, n_{\text{max}} \\ c = 0, \dots, c_{\text{hard}} \\ h = 1, \dots, h_{\text{max}} \end{cases}$$

$$\text{subject to: } \begin{cases} (n, h) \text{ unique } \forall n \text{ and } \forall h \\ c = \begin{cases} 0 & \text{if parent WO} \\ > 0 & \text{otherwise} \end{cases} \\ 0 \leq c_{\text{soft}} \leq c_{\text{hard}} \end{cases}$$

There is a design consequence to this limitation (Equation 132 on the previous page).

The lower limit of the preservation capacity of each host is based on the total number of preservation copies that the system of WOs needs spread over all hosts in the system (Equation 133 on the preceding page). While this capacity will be enough to meet all the needs of the WOs, the WOs may not know about, or discover all hosts.

The total system capacity can be computed (Equation 134 on the previous page).

Figure 232 on the following page is a notational diagram of how the various combinations of hosts, WO preservation needs and host capacity relate. The x-axis shows a range of hosts from a small value to a large one, while the y-axis shows host preservation capacity ranging from small to large. There is a diagonal line that shows the total system capacity (Equation 134 on the previous page). System required capacity (Equation 133 on the preceding page) is shown as a horizontal line. The number of hosts in the system (Equation 132 on the previous page) is shown by the left-hand vertical line and the desired number of preservation copies is to the right of the host line.

In Figure 232 on the following page the red area is the region where the limited number of hosts preclude any likelihood of meeting the system's preservation needs. The green area represents enough system capacity, but there are too few hosts to

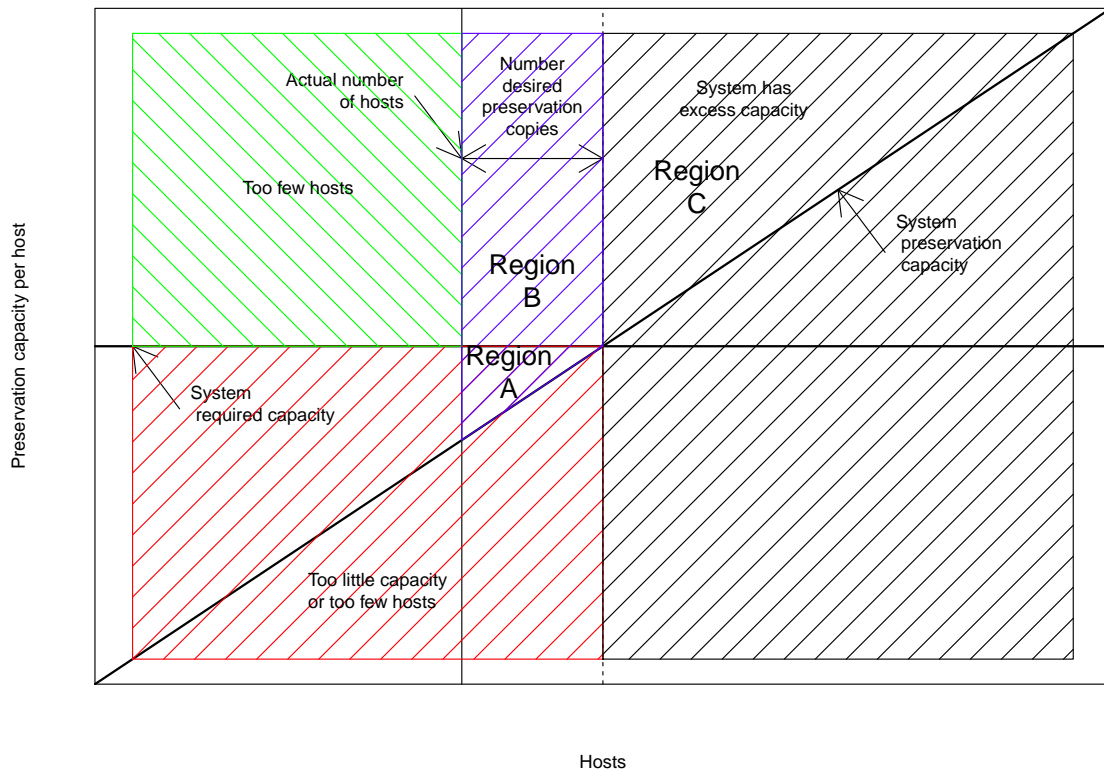


Figure 232. Notional host system design diagram relating host capacity to WO preservation needs.

meet the constraint that each WO family member be on a different host. The blue area is where preservation will occur and is subdivided into two regions, A and B. In Region A, there are there hosts and they have limited capacity, so will be unable to meet all the preservation needs of the system. While some WOs will meet their needs, some will not. In Region B, there is enough capacity and enough hosts to meet the system's needs. The closer the number of hosts is to the right hand boundary, the greater the number of WOs that will meet their needs. The black area (Region C) is a region of excess system capacity. Hosts in this region will never be discovered or used because no WOs live on them (Equation 135 on page 519).

K.3 PRESERVATION EFFECTIVENESS ANALYSIS

We have identified a limited number of relationships between c_{soft} , c_{hard} , and

h_{cap} . In all cases $c_{\text{soft}} \leq c_{\text{hard}}$, but the relationship of h_{cap} is less clean cut. We have identified, and named all possible relationships between c_{soft} , c_{hard} , and h_{cap} (Table 57).

Table 57. Named conditions for total system host preservation capacity h_{cap} in relation to total system c_{soft} and c_{hard} . We have taken the liberty to abuse the definitions of h_{cap} , c_{soft} and c_{hard} by interpreting them to apply to the total system, vice a single host or WO.

Name	Requirements
Famine	$h_{\text{cap}} < c_{\text{soft}} \leq c_{\text{hard}}$
Boundary Low	$h_{\text{cap}} = c_{\text{soft}} \leq c_{\text{hard}}$
Straddle	$c_{\text{soft}} \leq h_{\text{cap}} \leq c_{\text{hard}}$
Boundary High	$c_{\text{soft}} \leq c_{\text{hard}} = h_{\text{cap}}$
Feast	$c_{\text{soft}} \leq c_{\text{hard}} < h_{\text{cap}}$

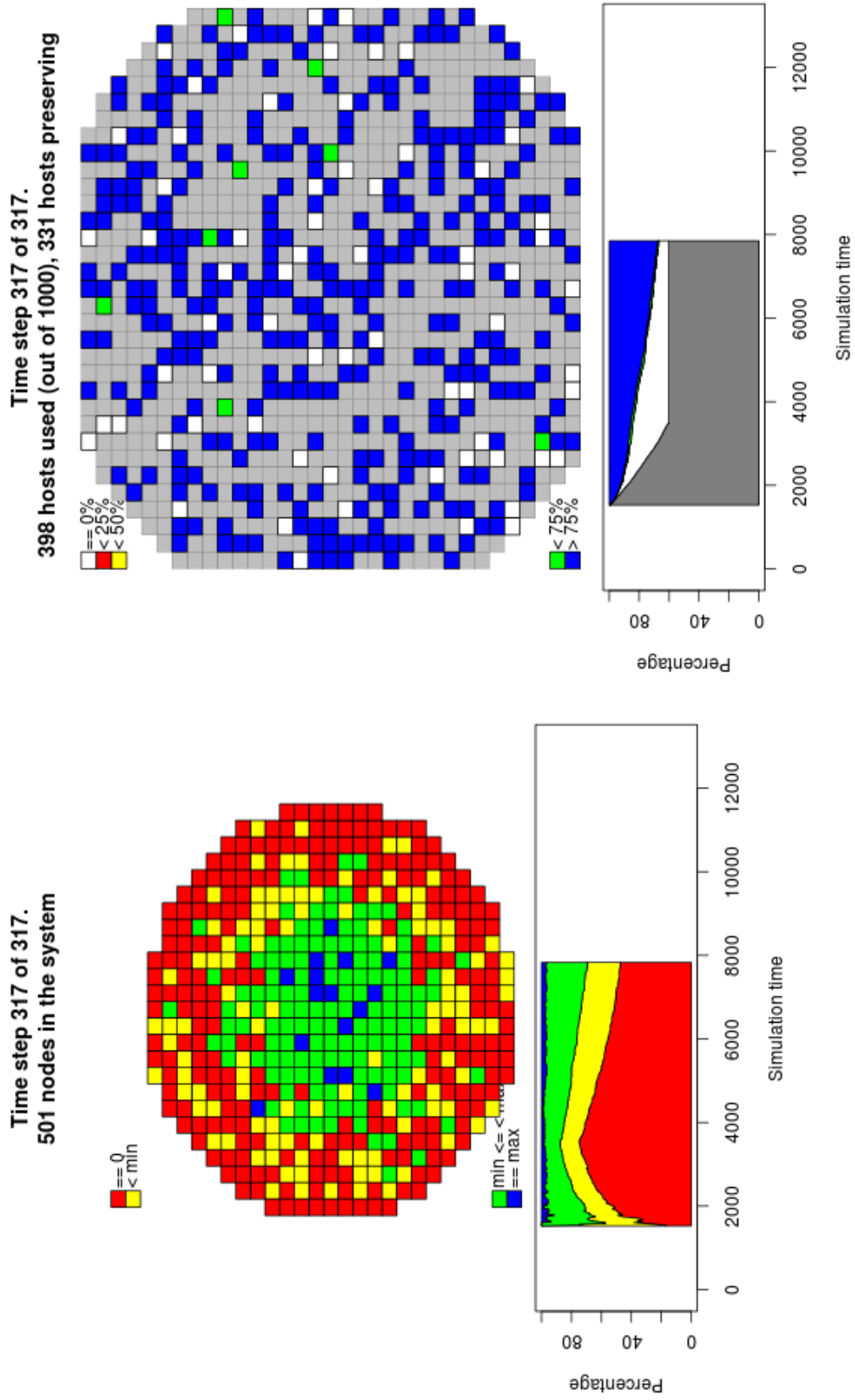


Figure 233. Effectiveness of least aggressive preservation policy in famine system capacity condition.

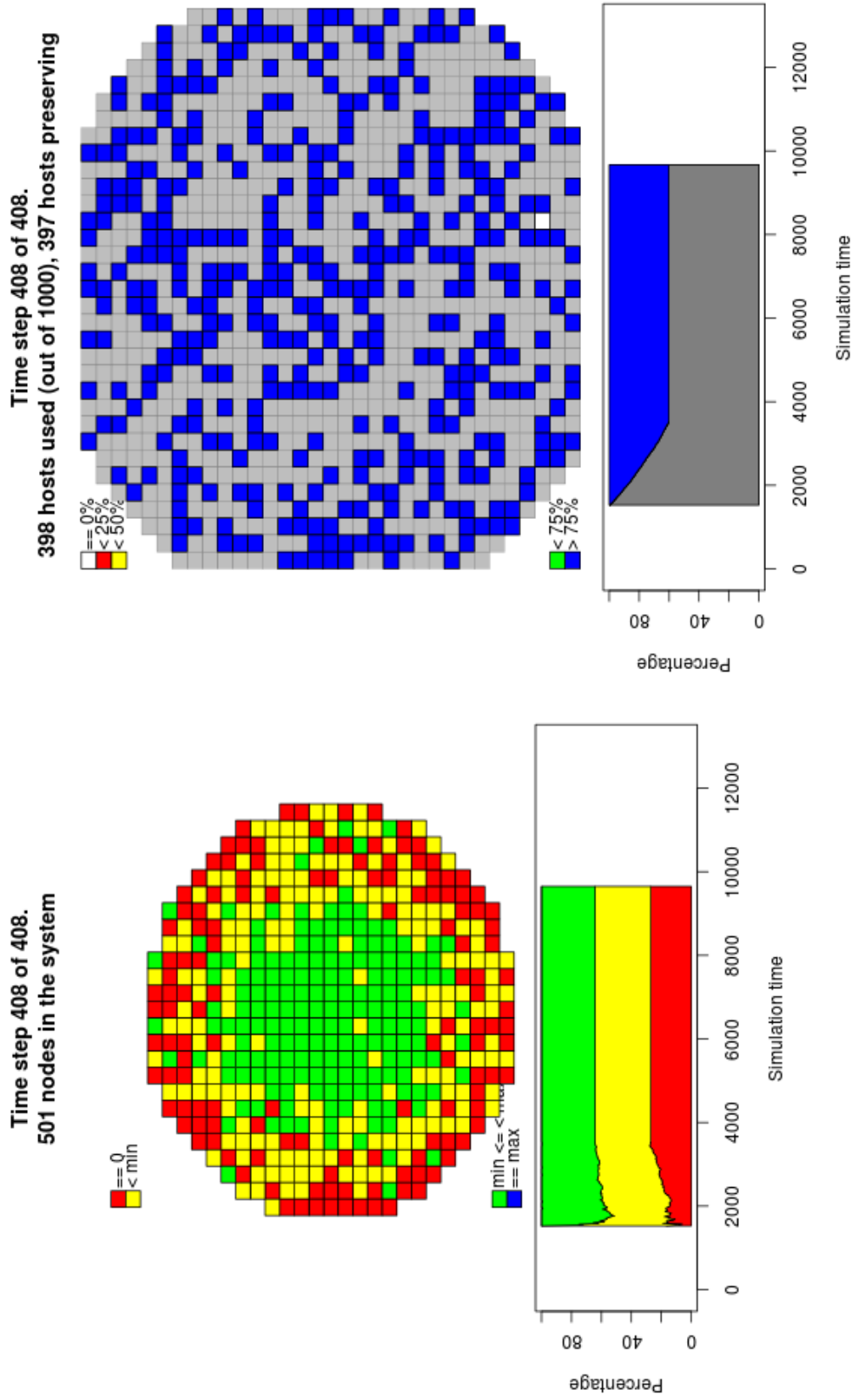


Figure 234. Effectiveness of moderately aggressive preservation policy in famine system capacity condition.

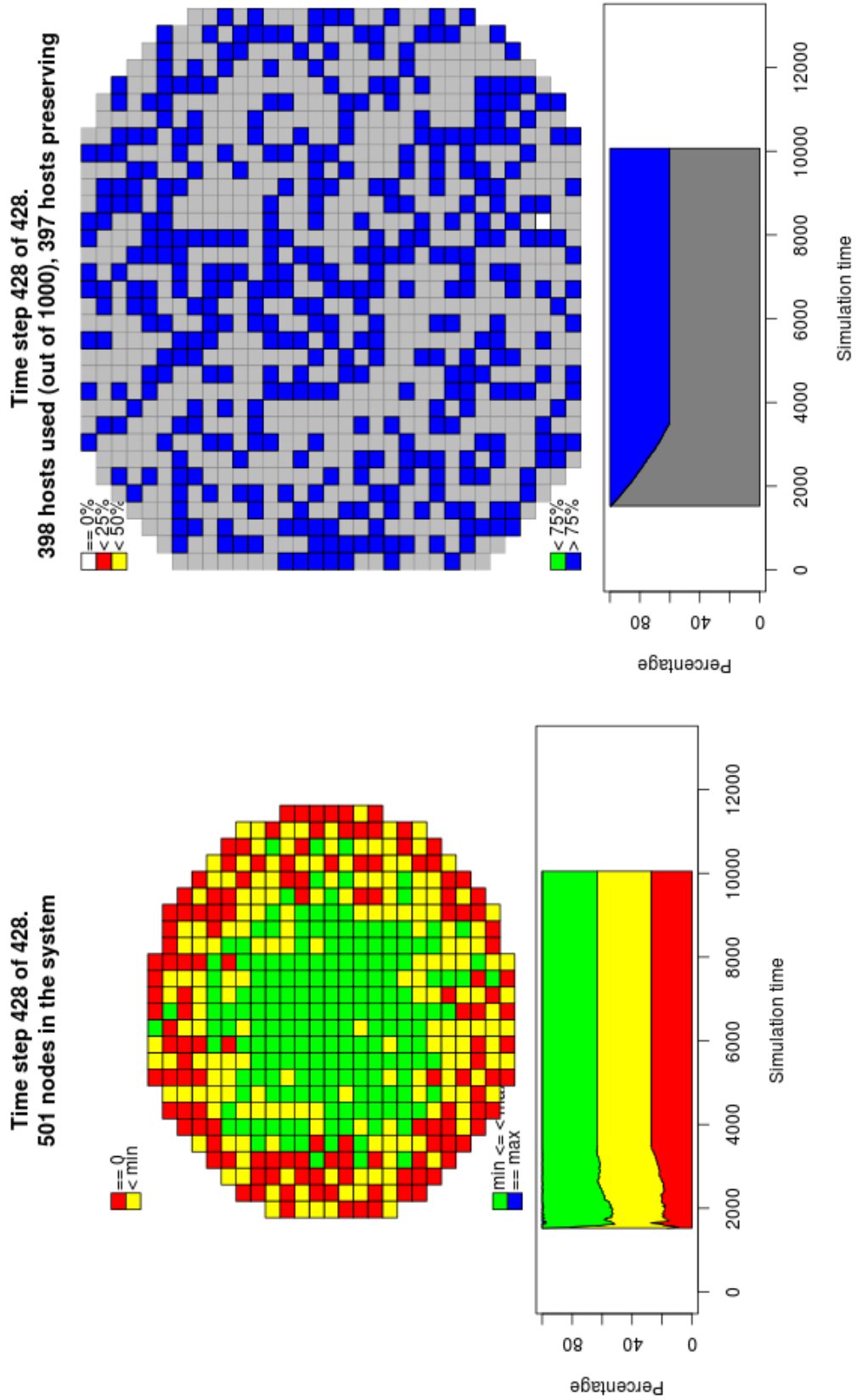


Figure 235. Effectiveness of most aggressive preservation policy in famine system capacity condition.

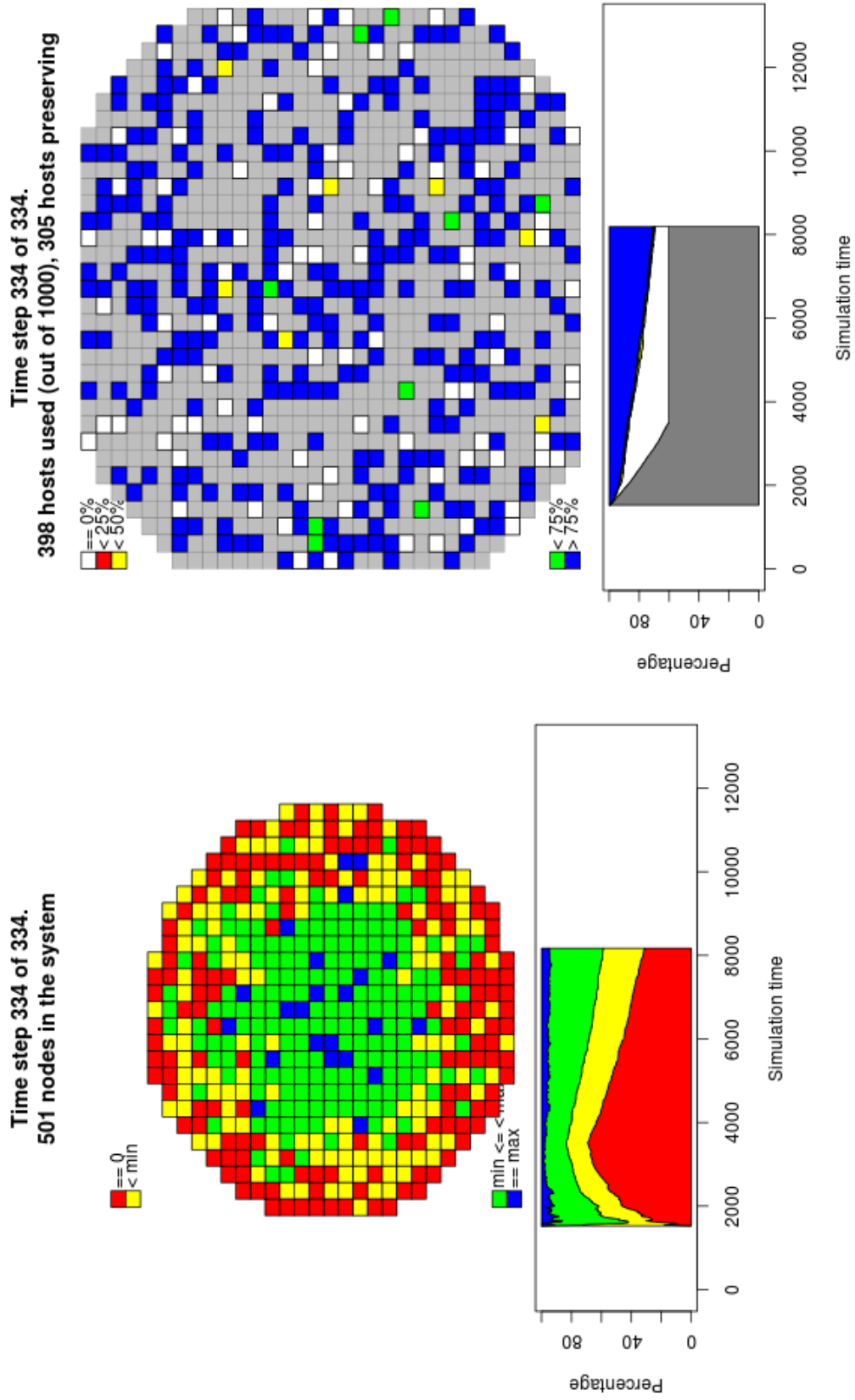


Figure 236. Effectiveness of least aggressive preservation policy in boundary-low system capacity condition.

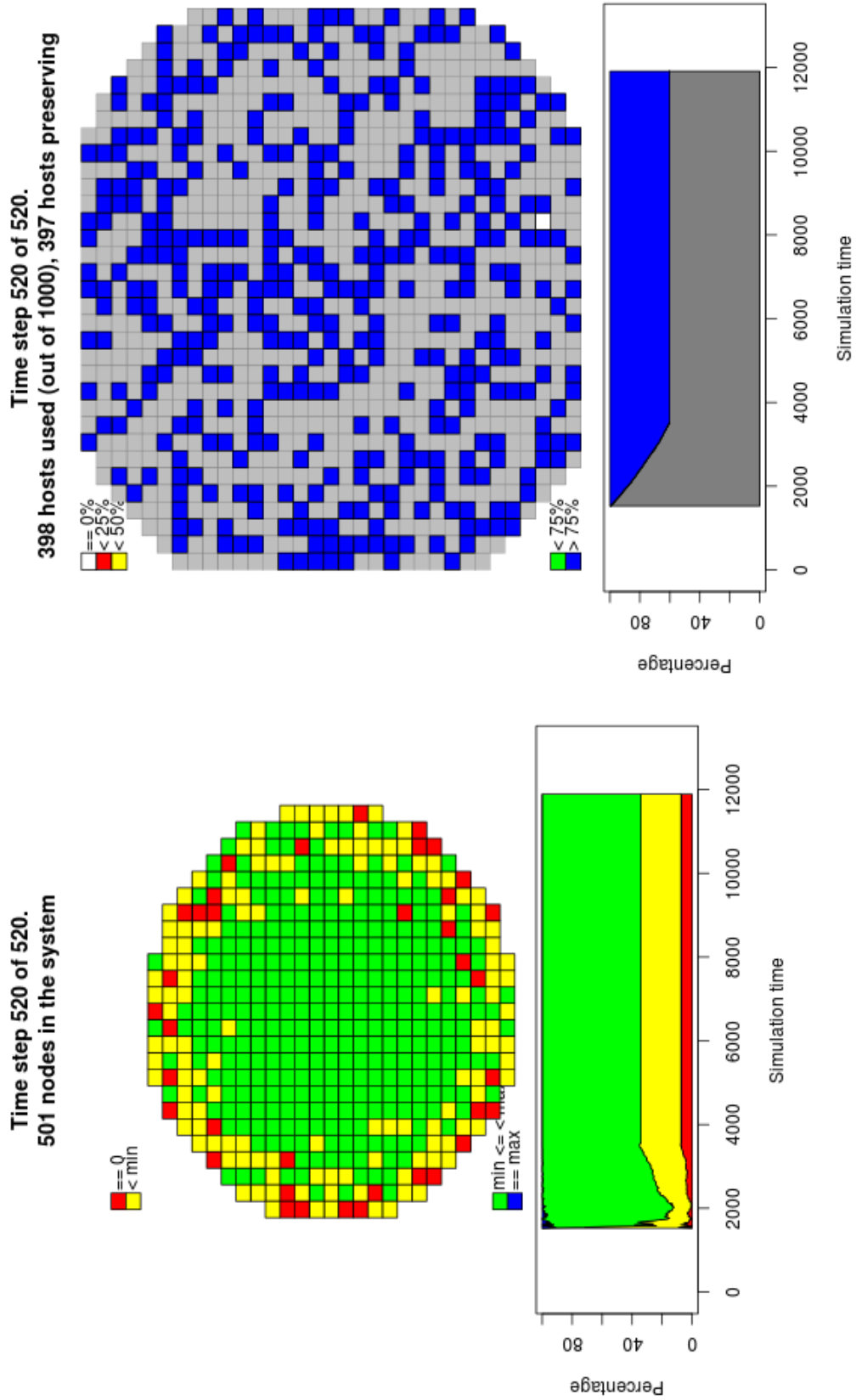


Figure 237. Effectiveness of moderately aggressive preservation policy in boundary-low system capacity condition.

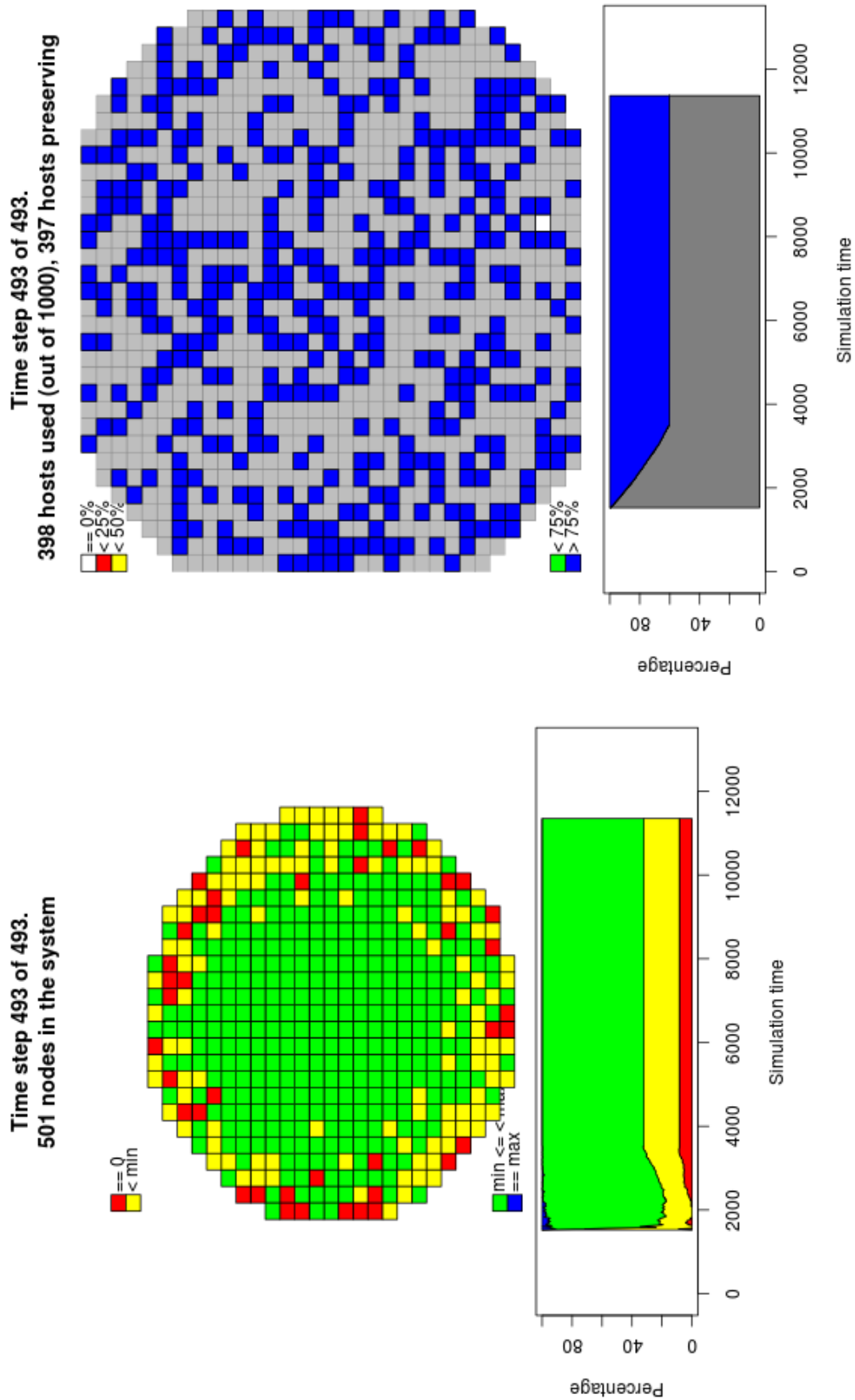


Figure 238. Effectiveness of most aggressive preservation policy in boundary-low system capacity condition.

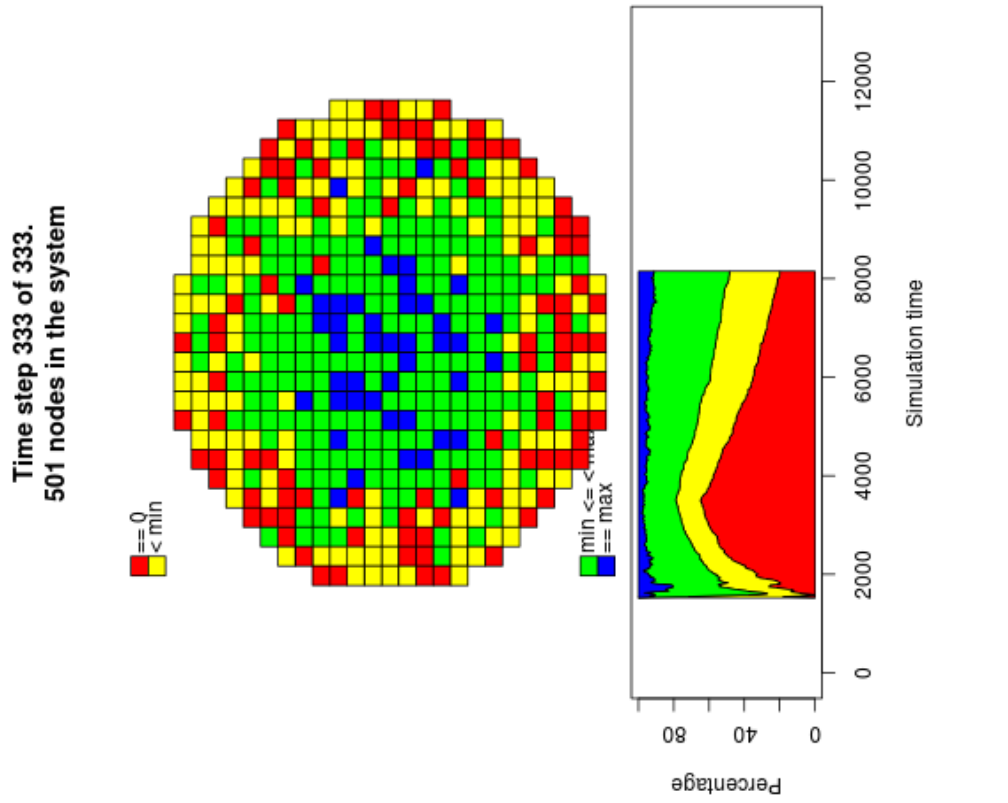
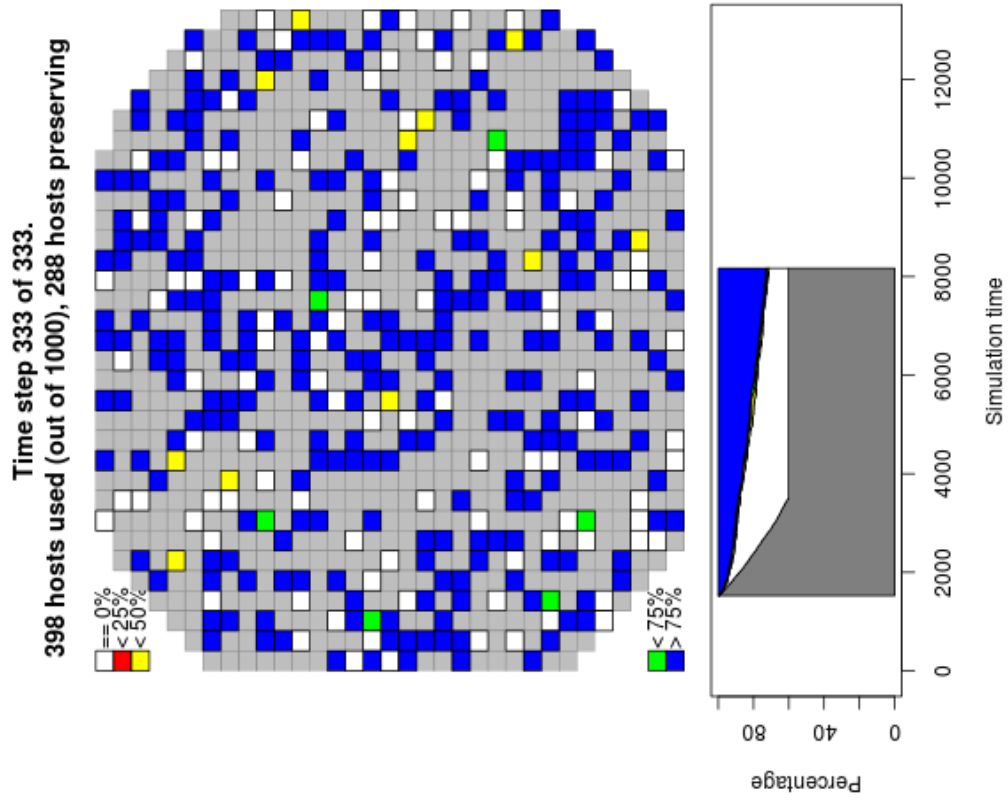


Figure 239. Effectiveness of least aggressive preservation policy in straddle system capacity condition.

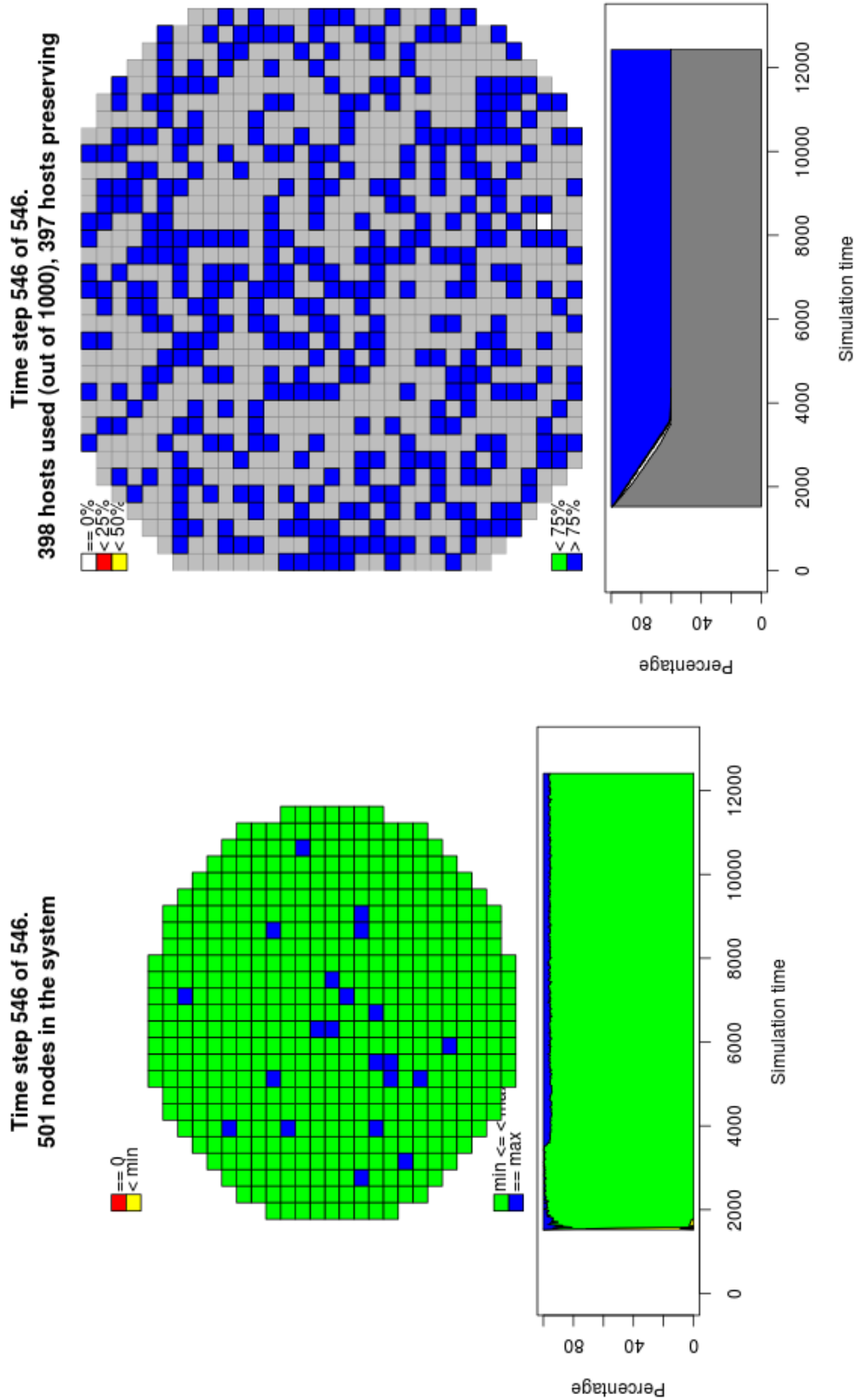


Figure 240. Effectiveness of moderately aggressive preservation policy in straddle system capacity condition.

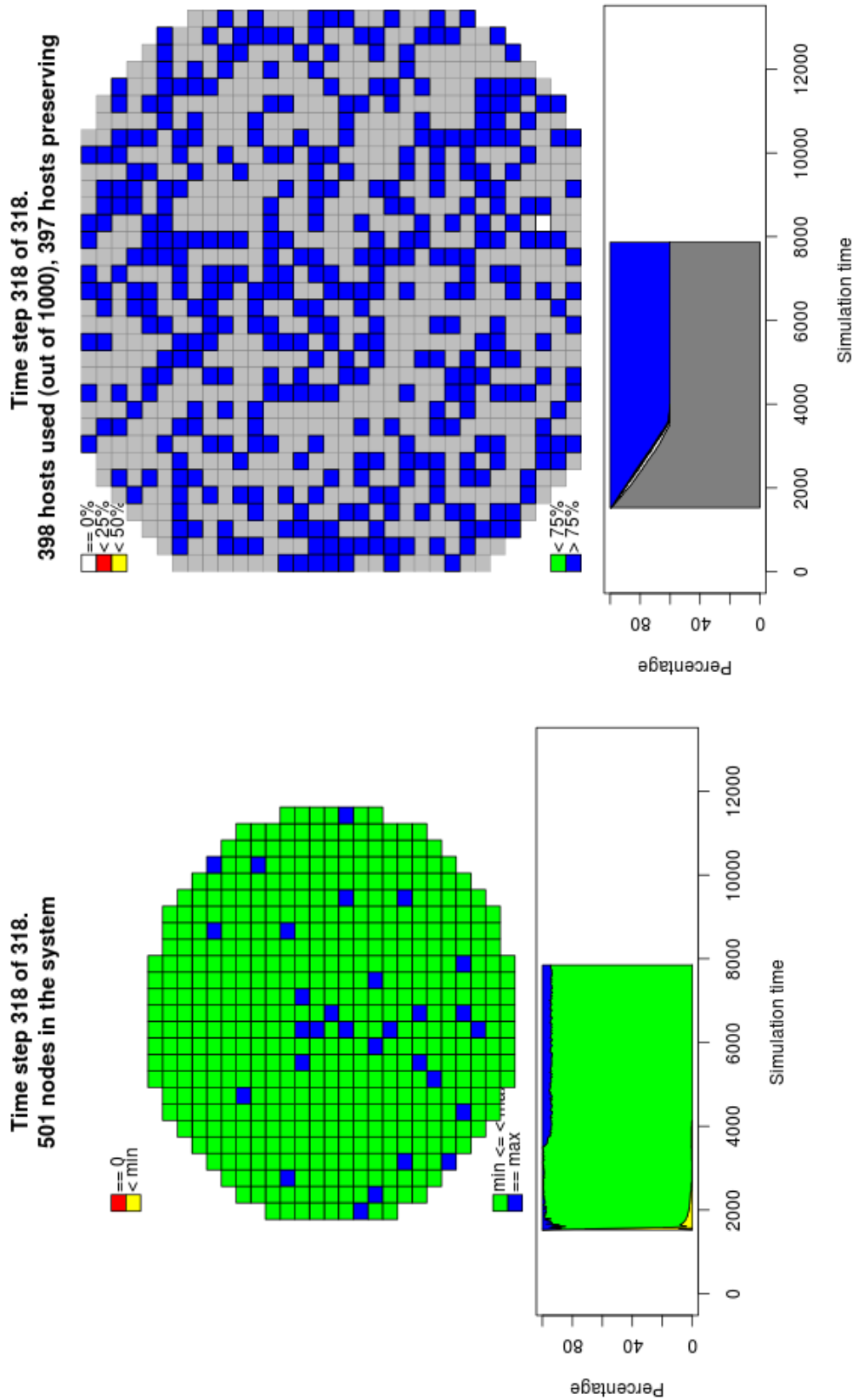


Figure 241. Effectiveness of most aggressive preservation policy in straddle system capacity condition.

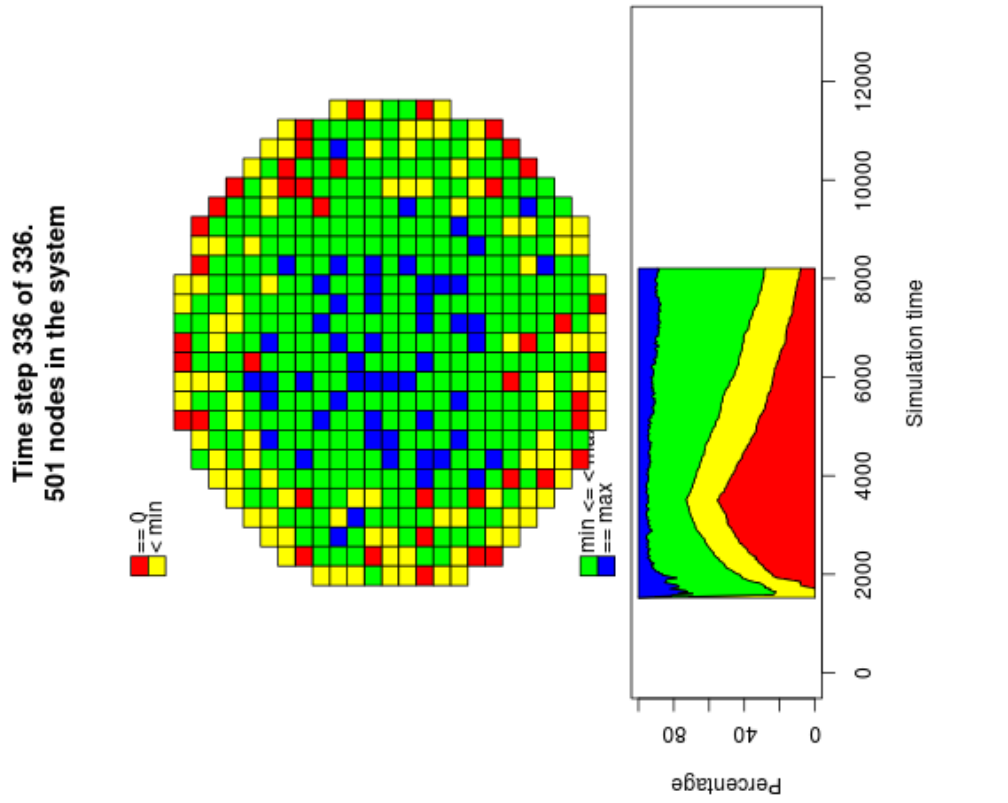
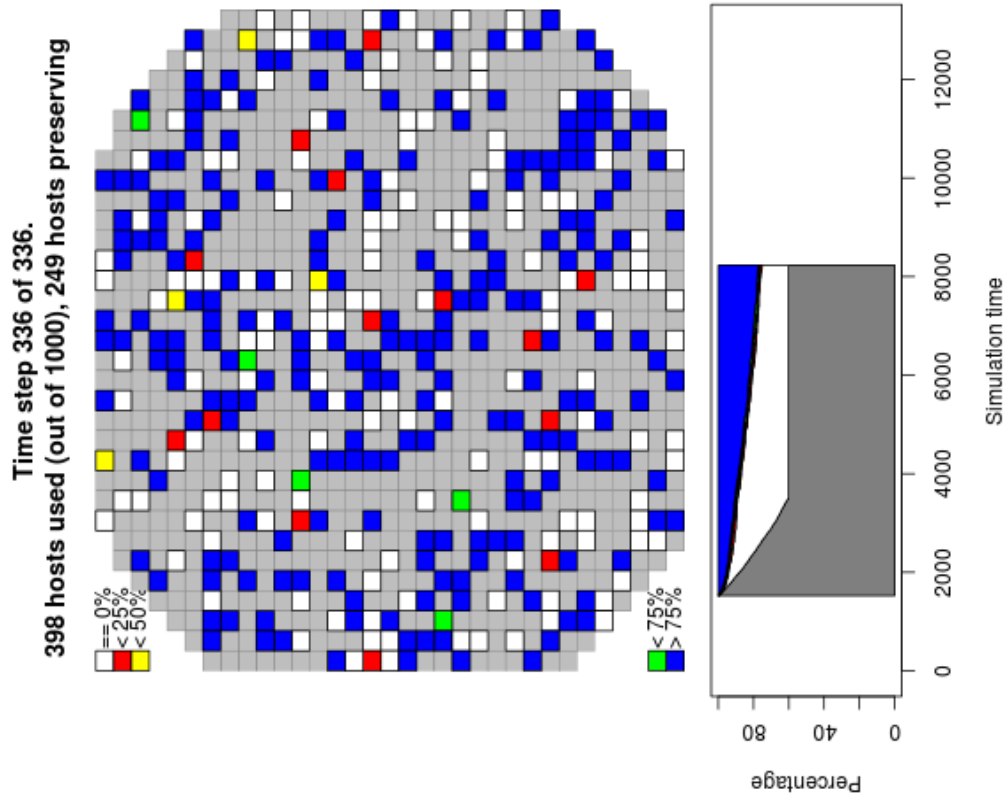


Figure 242. Effectiveness of least aggressive preservation policy in boundary-high system capacity condition.

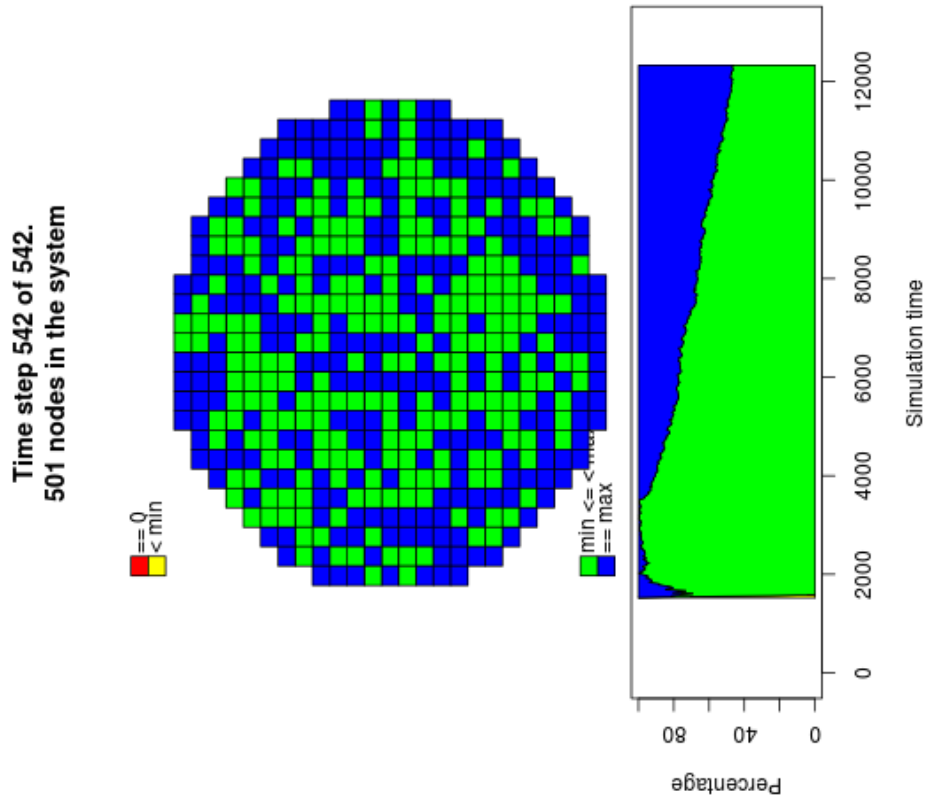
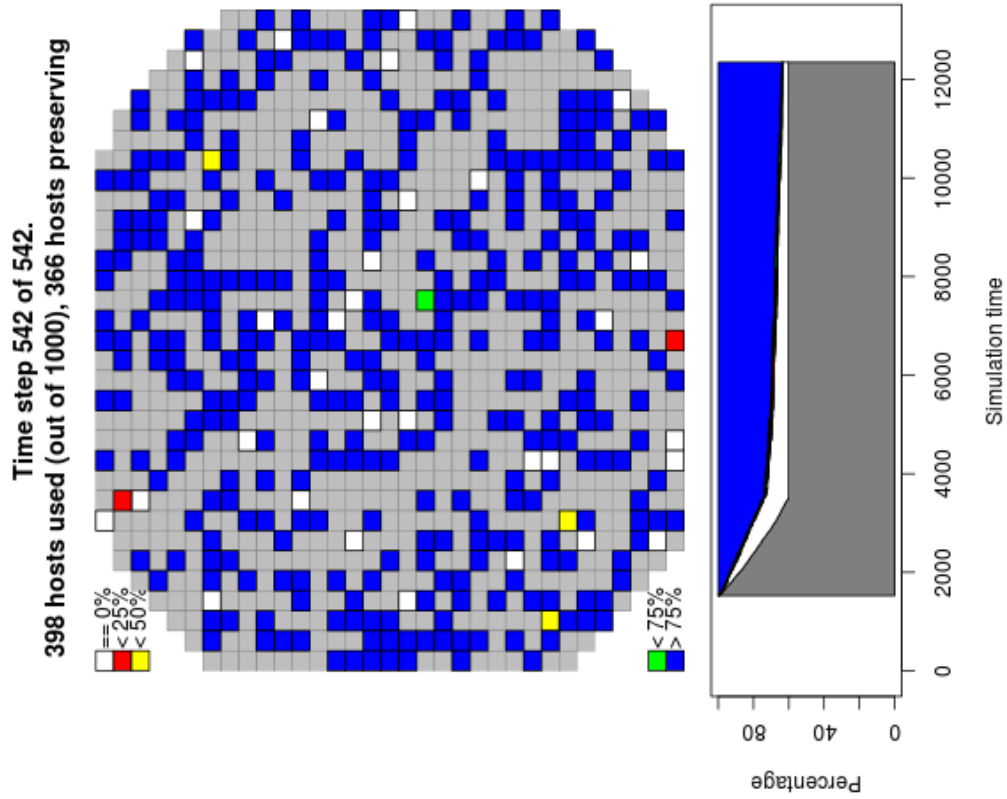


Figure 243. Effectiveness of moderately aggressive preservation policy in boundary-high system capacity condition.

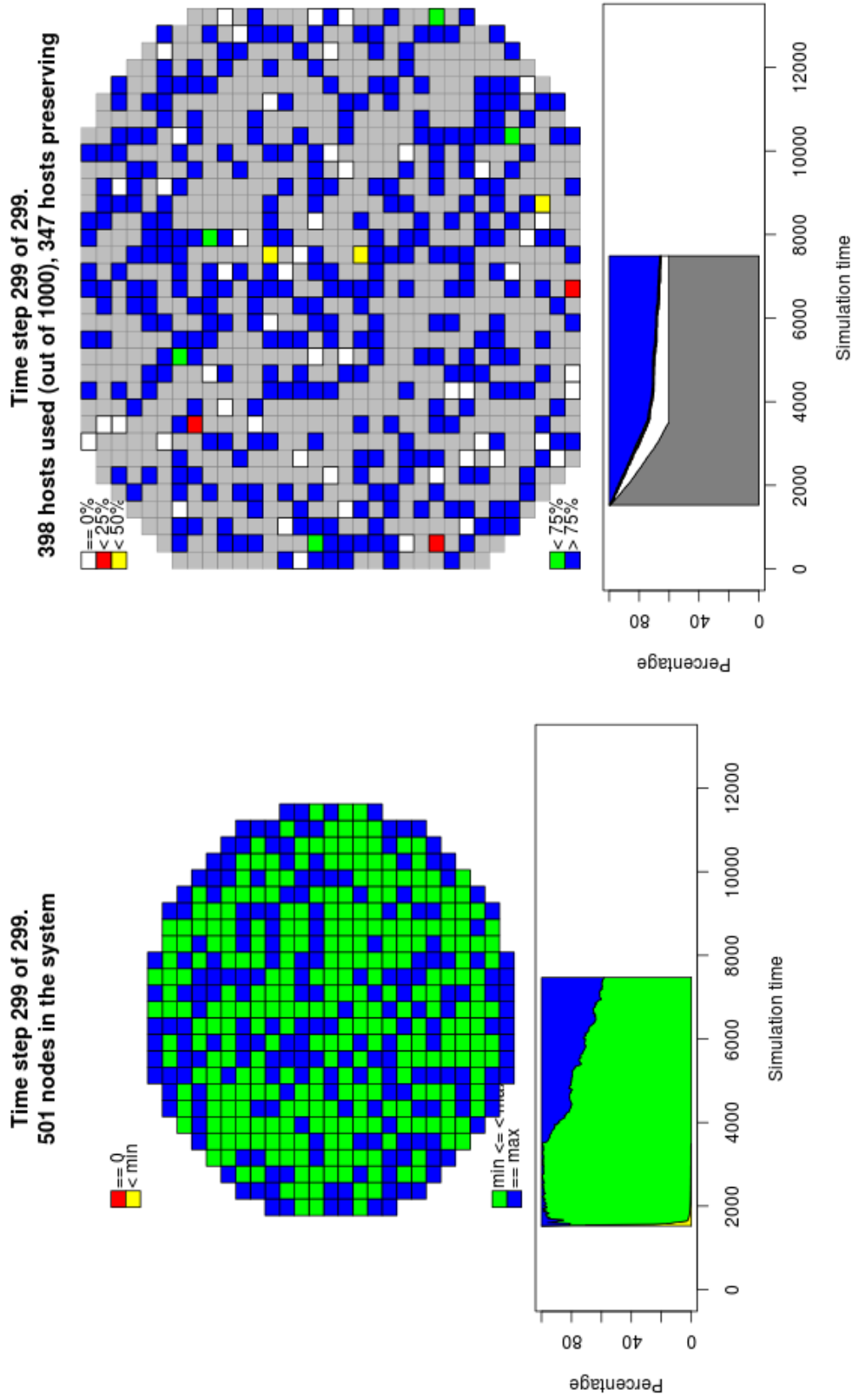


Figure 244. Effectiveness of most aggressive preservation policy in boundary-high system capacity condition.

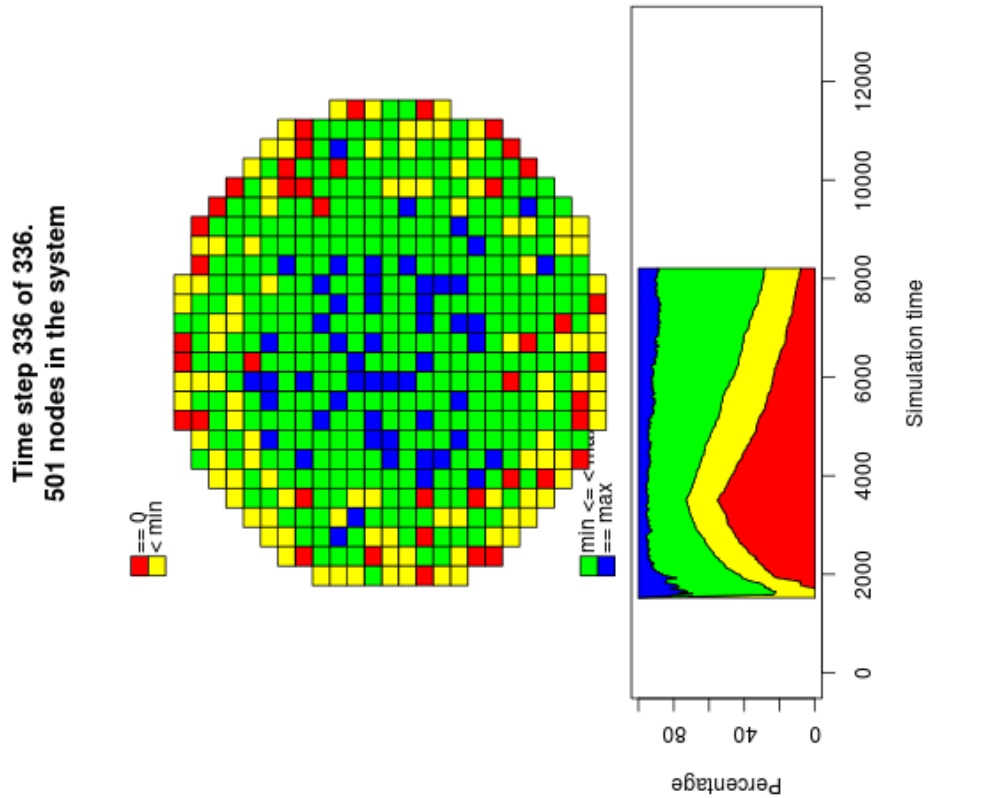
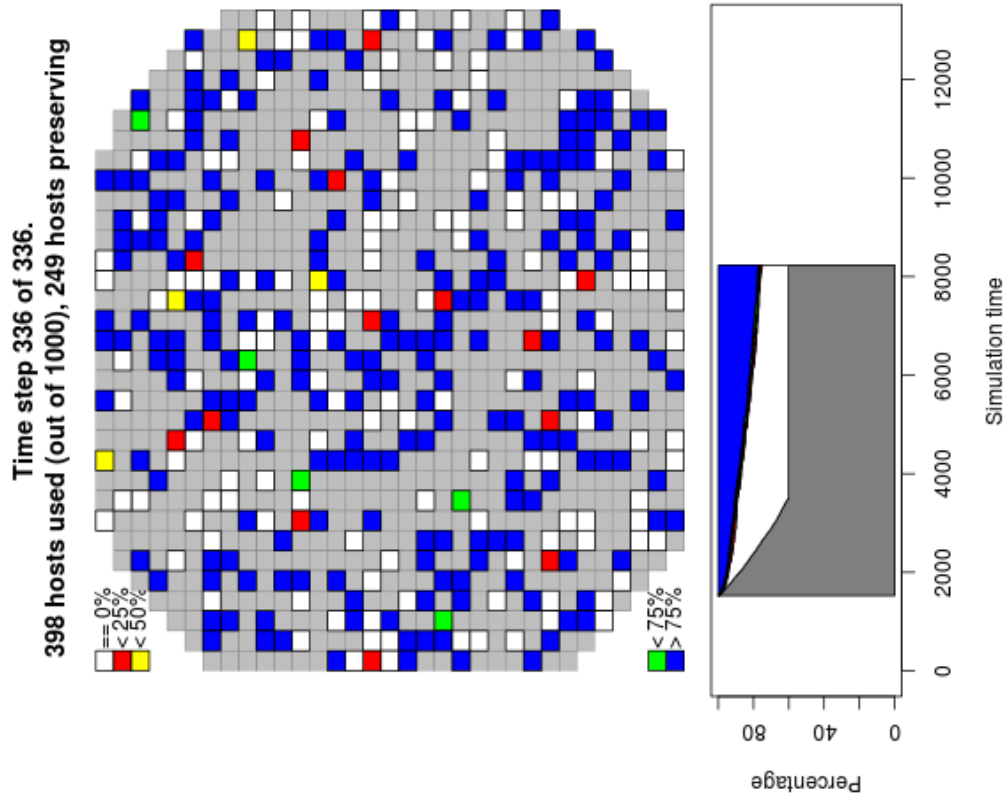


Figure 245. Effectiveness of least aggressive preservation policy in feast system capacity condition.

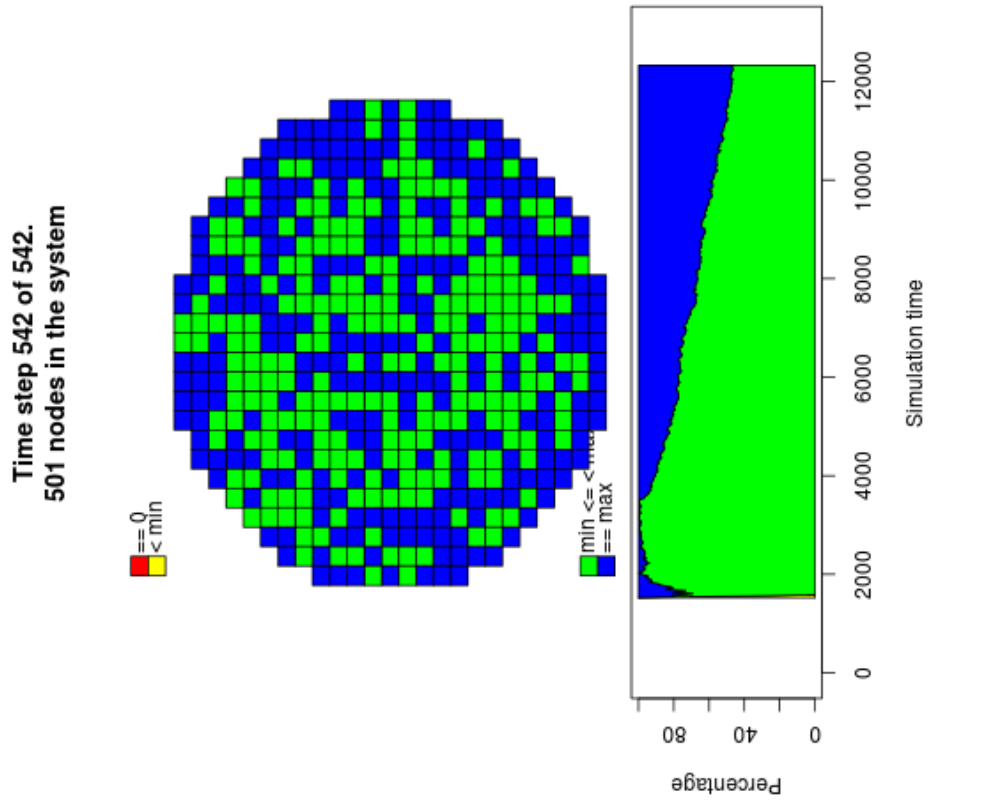
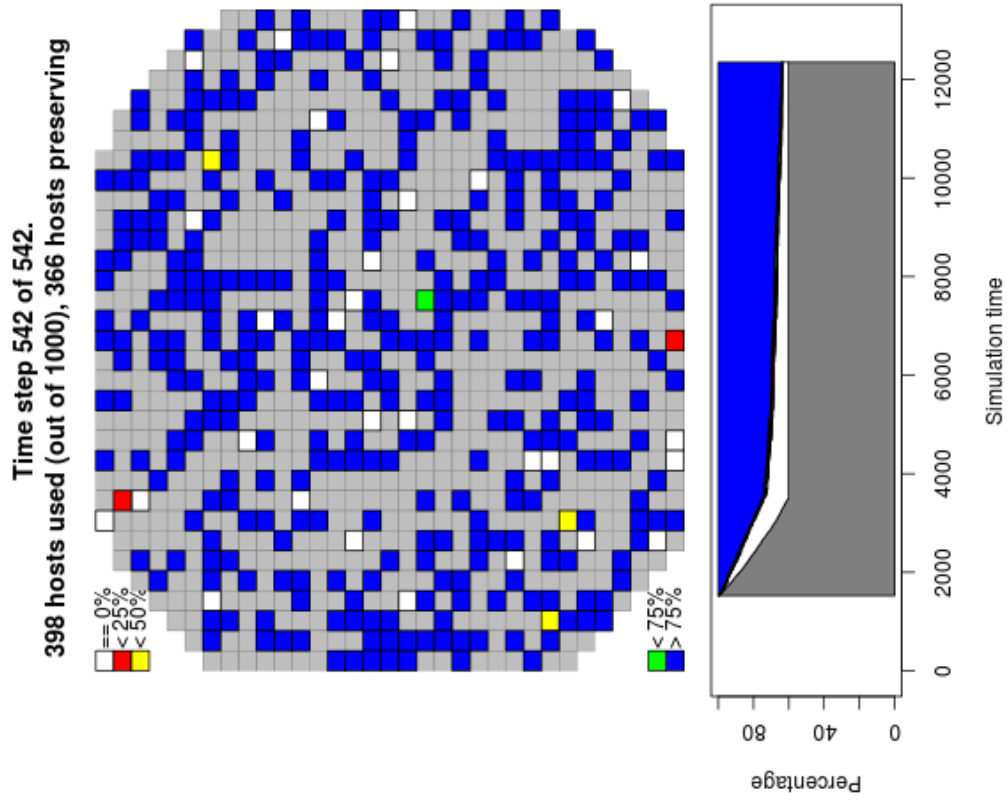


Figure 246. Effectiveness of moderately aggressive preservation policy in feast system capacity condition.

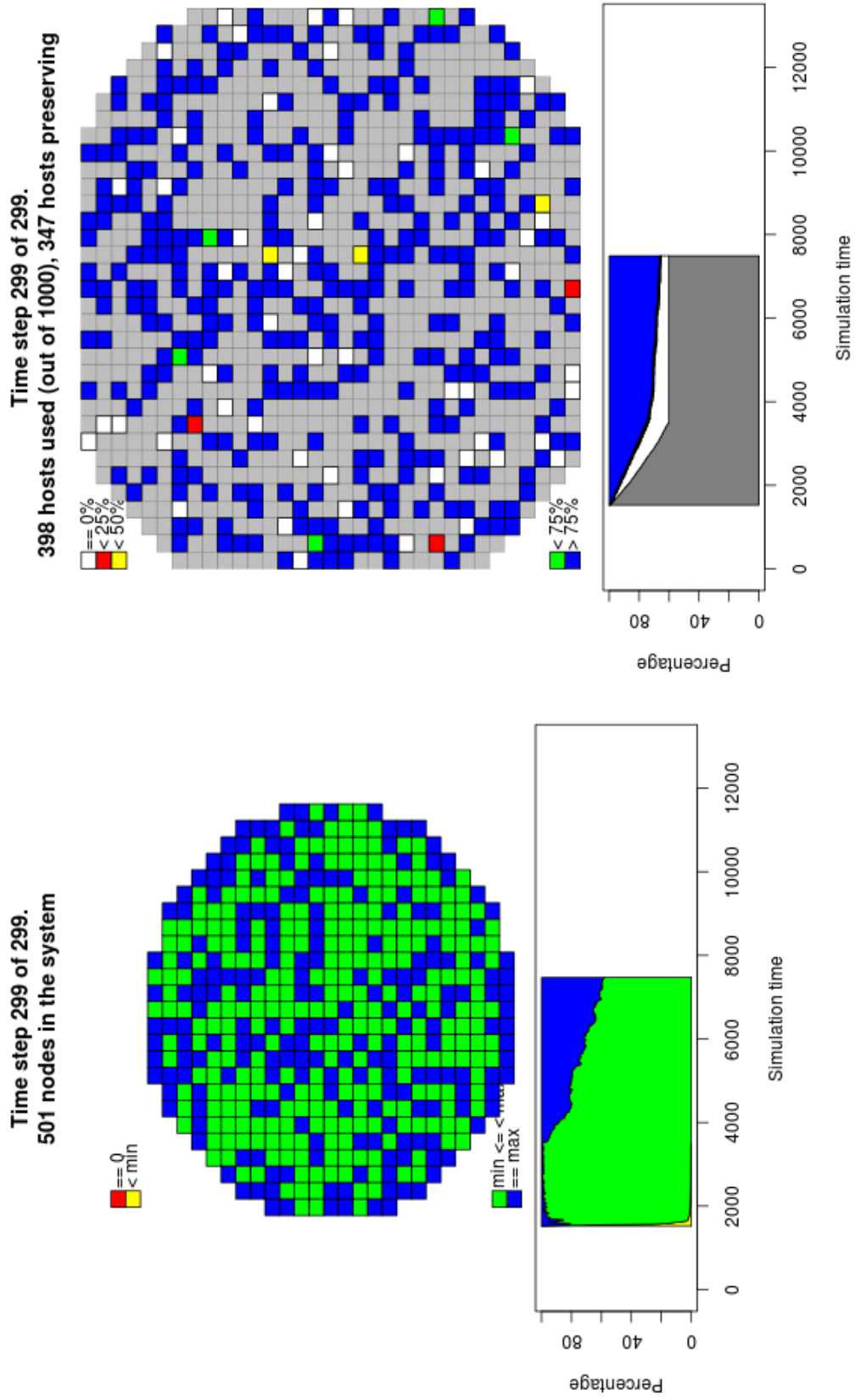


Figure 247. Effectiveness of most aggressive preservation policy in feast system capacity condition.

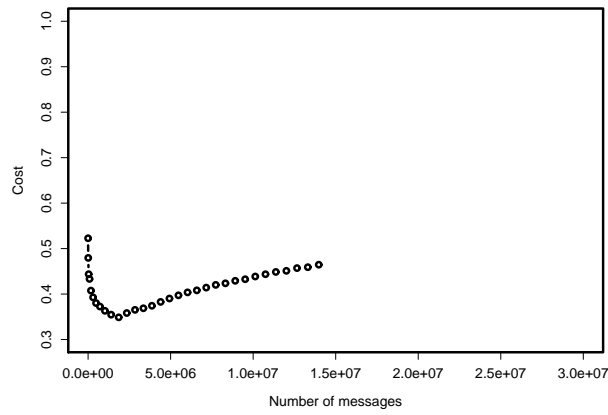
We evaluated the effectiveness of various preservation policies based on the named conditions (Figure 233 on page 523 through Figure 247 on the preceding page). Simulation control parameters were set so that the total preservation capacity of the hosts was varied to create the named conditions. For each of the named conditions, the preservation policy was varied and data was collected. In all cases, the least aggressive policy was the least successful at meeting the system's preservation goals. Under the Famine and Boundary Low conditions, it would be impossible to meet preservation goals, and the Moderately and Most aggressive policies arrived at approximately the same steady state situations in about the same length of time. Boundary High and Feast conditions have enough capacity for the system to meet its preservation needs.

K.4 COMMUNICATION COST ANALYSIS

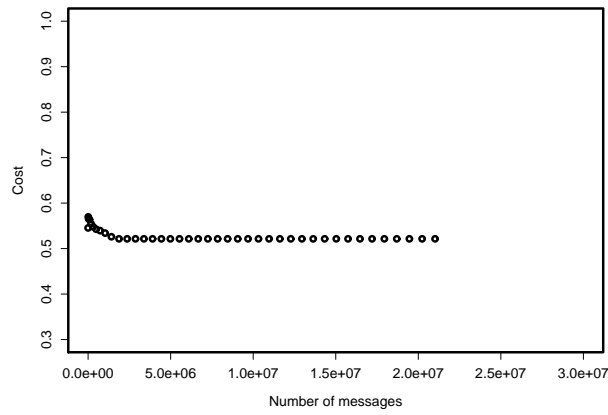
We next considered the number of messages exchanged in the USW system from inception to system stabilization under the named preservation conditions and preservation policies (Figure 248 on the next page through Figure 252 on page 543). In all cases, the Least aggressive policy was the least successful at meeting the system's preservation goals. Under the Famine and Boundary Low conditions, it would be impossible to meet preservation goals, and the Moderately and Most aggressive policies arrived at approximately the same steady state situations in about the same length of time. Boundary High and Feast conditions have enough capacity for the system to meet its preservation needs, and the Most aggressive policy operates about twice as efficiently as the Moderately aggressive policy. The Moderately aggressive policy took longer to stabilize, but in the end 18% more WOs reached their preservation goals and had a lower rate of messages sent per WO activation.

K.5 SUMMARY

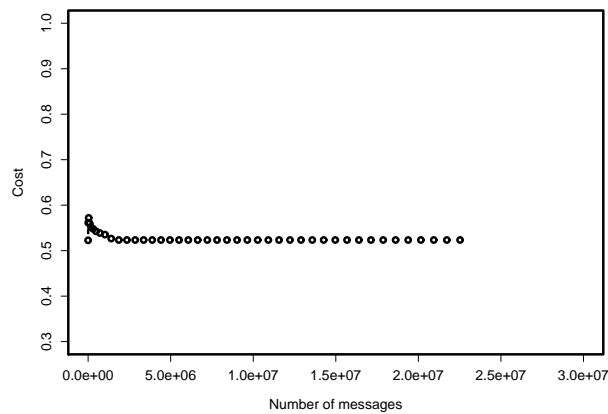
In Table 58 on page 544, we have taken the liberty to abuse the definitions of h_{cap} , c_{soft} and c_{hard} by interpreting them to apply to the total system, vice a single host or WO. In all cases, the Least aggressive policy was the least successful at meeting the system's preservation goals. Under the Famine and Boundary Low conditions, when it would be impossible to meet preservation goals, both the Moderately and Most aggressive policies arrived at approximately the same steady state situations after exchanging approximately the same number of messages. Straddle



(a) Least aggressive

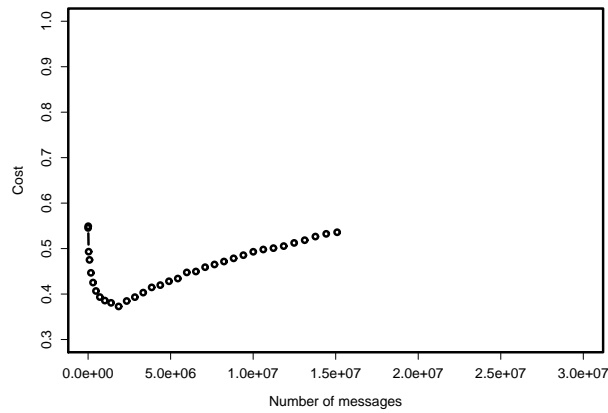


(b) Moderately aggressive

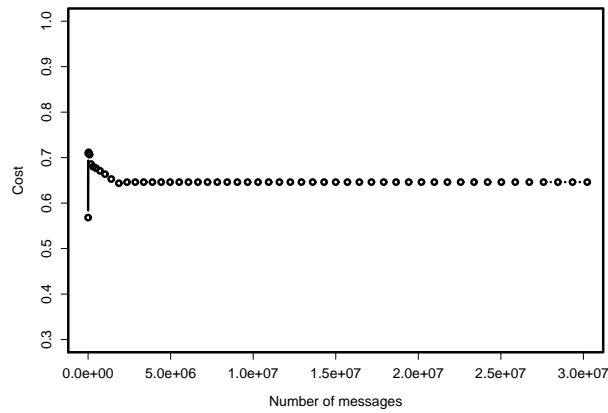


(c) Most aggressive

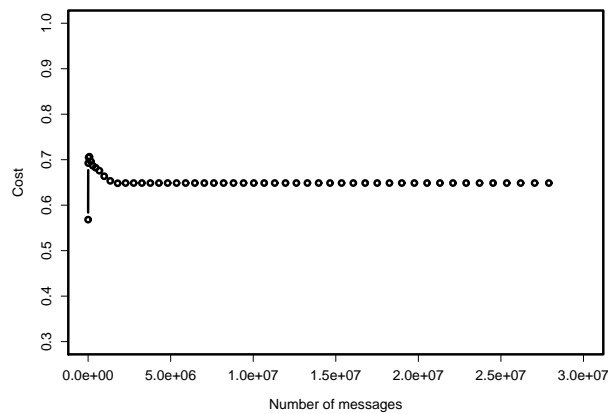
Figure 248. Number of messages exchanged based on preservation policies in famine system capacity condition.



(a) Least aggressive

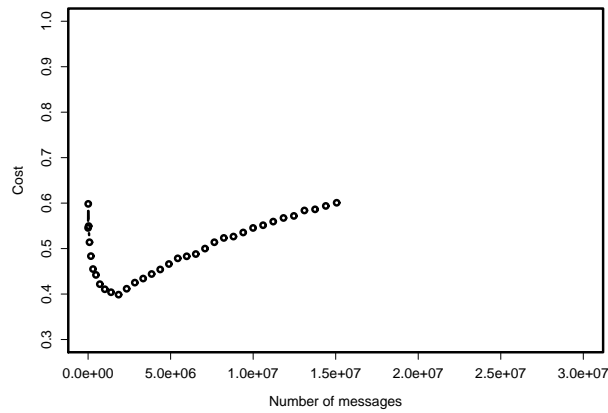


(b) Moderately aggressive

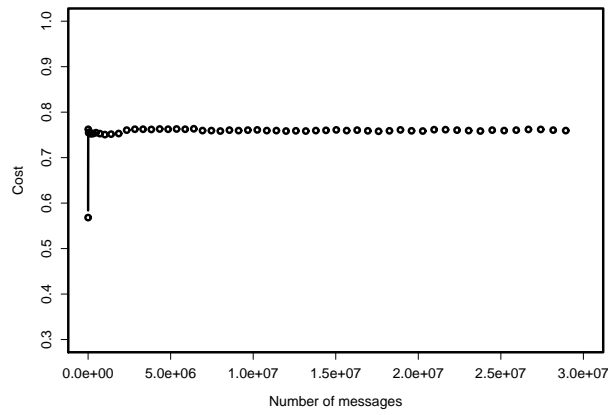


(c) Most aggressive

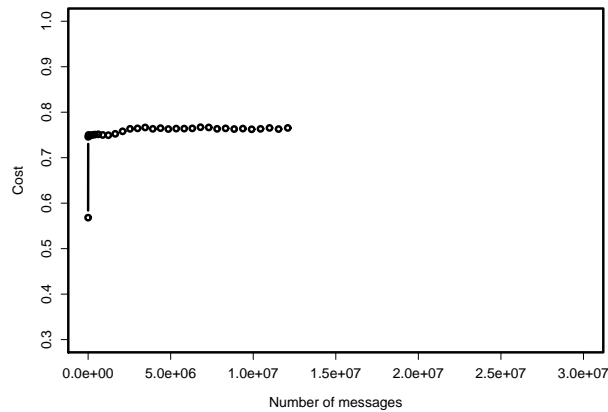
Figure 249. Number of messages exchanged based on preservation policies in boundary-low system capacity condition.



(a) Least aggressive

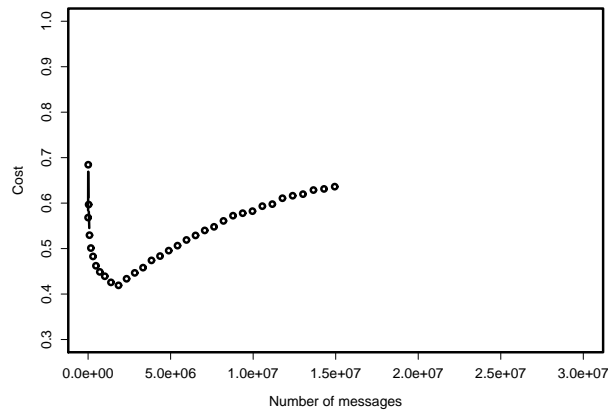


(b) Moderately aggressive

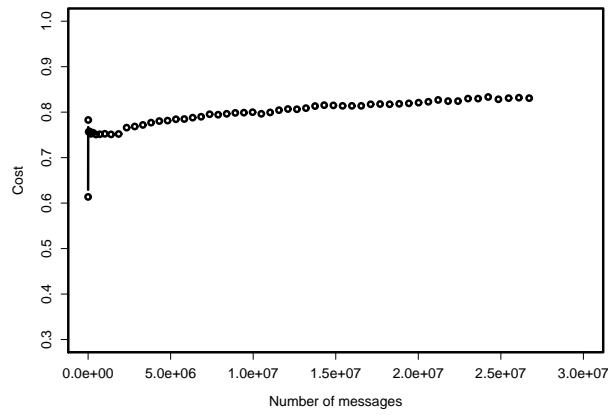


(c) Most aggressive

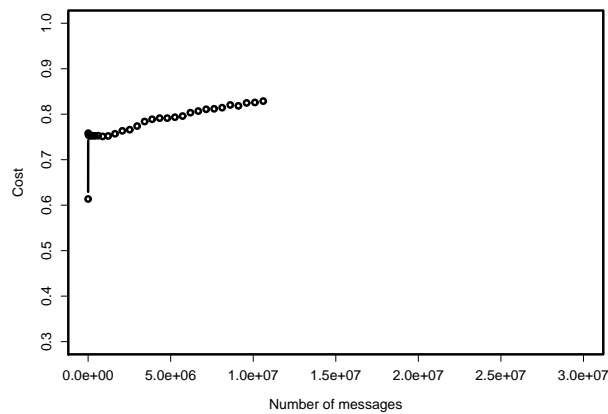
Figure 250. Number of messages exchanged based on preservation policies in straddle system capacity condition.



(a) Least aggressive

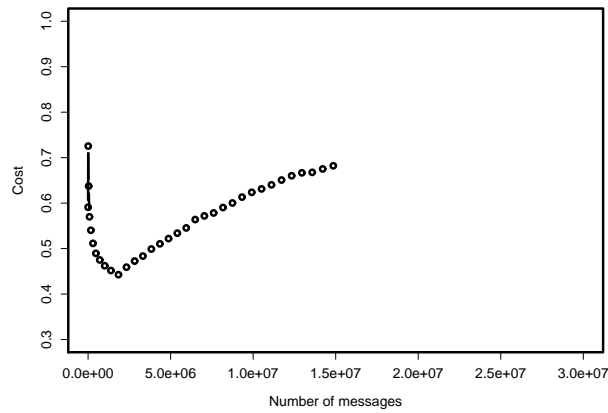


(b) Moderately aggressive

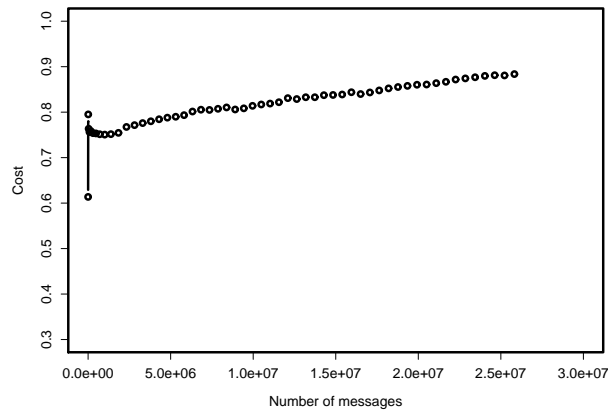


(c) Most aggressive

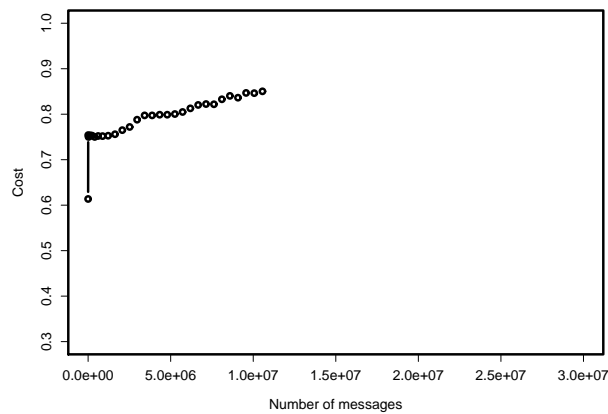
Figure 251. Number of messages exchanged based on preservation policies in boundary-high system capacity condition.



(a) Least aggressive



(b) Moderately aggressive



(c) Most aggressive

Figure 252. Number of messages exchanged based on preservation policies in feast system capacity condition.

Table 58. The effectiveness of various preservation policies based on named host capacity conditions.

Named host capacity	Preservation policy		
	Least aggressive	Moderately aggressive	Most aggressive
Famine $(c_{\text{soft}} < h_{\text{cap}} \leq c_{\text{hard}})$ Boundary Low $(h_{\text{cap}} = c_{\text{hard}} \leq c_{\text{soft}})$	Lowest percentage of WOs achieving preservation goal.	Equally marginally effective.	
Straddle $(h_{\text{cap}} \leq c_{\text{hard}} \leq c_{\text{soft}})$ Boundary High $(c_{\text{soft}} = h_{\text{cap}} \leq c_{\text{hard}})$ Feast $(h_{\text{cap}} < c_{\text{soft}} \leq c_{\text{hard}})$		The baseline against which others are measured. Takes longer to stabilize, but more WOs reach preservation goals.	Reaches steady state fastest with the most messages sent in the shortest period of time.

conditions would permit some WOs to achieve their goals, if the WOs were fortunate. Straddle results under Moderately and Most aggressive policies are comparable and the Most aggressive reaching steady state after exchanging about $\frac{1}{2}$ as many messages as the Moderately aggressive policy. Boundary High and Feast conditions have enough capacity for the system to meet its preservation needs, and the Most aggressive policy stabilizes the fastest, but the Moderately aggressive policy allows more WOs to reach their goals.

APPENDIX L

LIST OF ALGORITHMS

Alg.	Caption	Page
1	USW model.	288
2	ActiveMaintenance() function.	289
3	Creation() function.	290
4	Connecting() function.	291
5	Copy() function.	292
6	CopyNumberNeeded() function.	293
7	IsActiveMaintainer() function.	294
8	MessageMaintenance() function.	295
9	PassiveMaintenance() function.	296
10	SampleSize() function.	297
11	Wandering() function.	298
12	Wandering() detailed view.	300
13	WOSelectionProcess()	301
14	Evaluating the robustness and resiliency of graph.	302
15	removeHighestBetween() function.	303
16	reconstitute() function.	304
17	OrderedQueue() function.	305
18	Value() function.	306
19	EvaluateConnections() function.	308
20	Important() function.	309
21	Wiki rd() implementation.	309
22	Wiki out() implementation.	310
23	Gmail rd() implementation.	310
24	Gmail out() implementation.	310

APPENDIX M

LIST OF LISTINGS

Listing	Caption	Page
1	USW implementation instrumentation.	255
2	Sample Edit Service request.	260
3	Sample Edit Service request.	260
4	Sample Copy Service request.	261
5	Sample Copy Service REM.	261
6	Sample mailbox lines from a REM.	263
7	Current WO personal mailbox.	277
8	Future work WO personal mailboxes.	278
9	Sample Friend Request message.	314
10	Sample Patch, adding a friend location.	316
11	Complete Friend Request message.	318
12	Sample Friend Request message processing.	321
13	Sample Copy Request message.	324
14	Complete Copy Request message.	332
15	Corresponding Copy Request message.	333
16	Copy Request Message sent to receiving WO copy service.	341
17	Copy service returns copy's URI.	349
18	Copy URI is sent back to requesting WO.	350
19	Copy URI is retrieved by the originating WO.	352
20	Originating WO is updated with copy's URI.	354
21	Copy's URI is sent to all of the originating WO "family" members.	356
22	Family member retrieves new copy URI location.	358
23	New copy URI added to existing family member.	360
24	Updating the mailbox last time checked timestamp.	362

REFERENCES

- [1] Mary Baker, Kimberly Keeton, and Sean Martin. Why Traditional Storage Systems Dont Help Us Save Stuff Forever. In *Proc. 1st IEEE Workshop on Hot Topics in System Dependability*. Citeseer, 2005.
- [2] Jeff Rothenberg. Ensuring the Longevity of Digital Information. *Scientific American Magazine*, 272(1):42 – 47, 1995.
- [3] William Y. Arms. *Digital Libraries*. The MIT Press, December 1999.
- [4] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), 1965.
- [5] Intel Corporation. Microprocessor Quick Reference Guide. <http://www.intel.com/pressroom/kits/quickreffam.htm>, 2013.
- [6] Intel Corporation. Your Source for Intel Product Information. <http://ark.intel.com/>, 2013.
- [7] Jakob Nielsen. Niensens Law of Internet Bandwidth. *URL* <http://www.useit.com/alertbox/980405.html>, 1998.
- [8] Steve Gilheany. Projecting the Cost of Magnetic Disk Storage Over the Next 10 Years. *Archive Builders*, 22011v054:9 – 5, 2000.
- [9] Chip Walter. Kryder’s Law. *Scientific American*, 293(2):32 – 33, 2005.
- [10] David S. H. Rosenthal. Storage Will Be A Lot Less Free Than It Used To Be. *DSHR’s Blog* <http://blog.dshr.org/2012/10/storage-will-be-lot-less-free-than-it.html>, (October 1), 2013.
- [11] David S. H. Rosenthal. Dr. Pangloss and the Road-Maps. *DSHR’s Blog* <http://blog.dshr.org/2013/07/dr-pangloss-and-road-maps.html>, (October 1), 2013.
- [12] David S. H. Rosenthal. Storage Technology Update. *DSHR’s Blog* <http://blog.dshr.org/2012/06/storage-technology-update.html>, (June 4), 2012.

- [13] David S. H. Rosenthal. Moore, Kryder vs. SAW. *DShR's Blog* <http://blog.dshr.org/2013/04/moore-kryder-vs-saw.html>, (April 25), 2013.
- [14] John B. Horrigan. BroadBand Adoption and Use in America. *Federal Communications Commission Omnibus Broadband Initiative*, 2010.
- [15] George Gilder. Metcalfs Law and Legacy. *Forbes ASAP*, 27, 1993.
- [16] Melonie Heron, Donna L. Hoyert, Sherry L. Murphy, Jiaquan Xu, Kenneth D. Kochanek, and Betzaida Tejada-Vera. Deaths: Final Data for 2006. Technical Report 14, U.S. Department of Health and Human Services, Centers for Disease Control and Prevention, Nat. Center for Health Statistics, Nat. Vital Statistics System, 2009.
- [17] Ray C. Brown. Colleges that have Closed, Merged or Changed Names. http://www2.westminster-mo.edu/wc_users/homepages/staff/brownr/ClosedCollegeIndex.htm.
- [18] Betty H. Carter. Beyond Books and Buildings: North Carolina's History of Higher Education Digital Project. <http://www.lib.unc.edu/highered/list.html>.
- [19] Lawrence Biemiller. Antioch Rises From the Ashes. *The Chronicle of Higher Education*, April 2012.
- [20] Chris Edwards. Government Schemes Cost More than Promised. *Cato Institute*, (17):B1, September 2003.
- [21] United States General Accounting Office. Department of Energy: Opportunity to Improve Management of Major System Acquisitions. *Report to the Chairman, Committee on Governmental Affairs, U.S. Senate, GAO/RCED-97-17*, 1996.
- [22] Public Law 111-314. The National Aeronautics and Space Act. http://www.nasa.gov/offices/ogc/about/space_act1.html.
- [23] Public Law 101-501. Defense Base Closure and Realignment Commission. <http://www.brac.gov/charter.html>.

- [24] National Science Foundation. NSF Data Management Plan Requirements, 2013.
- [25] William Y. Arms. Preservation of Scientific Serials: Three Current Examples. *Journal of Electronic Publishing*, 5(2), 1999.
- [26] Catherine C. Marshall, Frank McCown, and Michael L. Nelson. Evaluating Personal Archiving Strategies for Internet-based Information. In *Proc. of the Information Systems and Technology (IS&T)*, 2007.
- [27] Matthew Humphries. ISP deletes all user media files every night. <http://www.geek.com/news/isp-deletes-all-user-media-files-every-night-566820/>.
- [28] Catherine C. Marshall, Frank McCown, and Michael L. Nelson. Why WebSites Are Lost (and How They're Sometimes Found). *Communications of the ACM*, 52(11):141 – 145, 2009.
- [29] Sara Yin. Flickr Permanently Deletes User's Account, 4,000 Photos by Accident. *PC Magazine*, February 2011.
- [30] Peng Hwa Ang and Berlinda Nadarajan. Censorship and the Internet: a Singapore perspective. *Communications of the ACM*, 39(6), 1996.
- [31] Internet Archive Team. Fire Update: Lost Many Cameras, 20 Boxes. No One Hurt. <http://blog.archive.org/2013/11/06/scanning-center-fire-please-help-rebuild/>.
- [32] Lee Hutchinson. Fire at Internet Archive destroys equipment and materials, but data safe. <http://arstechnica.com/business/2013/11/fire-at-internet-archive-destroys-equipment-and-materials-but-data-safe/>.
- [33] Ry Crozier. ISPs hit by Queensland flood crisis. <http://www.itnews.com.au/News/244293,isps-hit-by-queensland-flood-crisis.aspx>.
- [34] Peter Krogh. File format migration. <http://www.dpbestflow.org/node/386>.
- [35] Geoffrey Haward Martin. *Domesday Book: A complete translation*. Penguin USA, 2003.

- [36] John Goddard and Peter Armstrong. The 1986 Domesday Project. *Transactions of the Institute of British Geographers*, pages 290 – 295, 1986.
- [37] David Holdsworth and Paul Wheatley. Long-Term Stewardship of Globally-Distributed Representation Information. In *Mass Storage Systems and Technologies*, pages 17–29. NASA/IEEE, 2004.
- [38] Margaret Hedstrom. Professor saves 'domesday' for future generations. *SI@umich*, 20(Spring):1, 2003.
- [39] Stewart Granger. Digital Preservation and Emulation: from theory to practice. In *Proc. of the Int. Cultural Heritage Meeting (ICHIM) 2*, pages 289 – 296, 2001.
- [40] Craig W. Reynolds. Flocks, Herds and Schools: A Distributed Behavioral Model. *SIGGRAPH Computer Graphics*, 21(4):25 – 34, 1987.
- [41] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A Scalable Content-Addressable Network. In *Proc. of the 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 161 – 172, 2001.
- [42] Alfredo Pina, Eva Cerezo, and Francisco J. Serón. Computer animation: from avatars to unrestricted autonomous actors (A survey on replication and modelling mechanisms). *Computers and Graphics*, 24(2):297 – 311, 2000.
- [43] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small world' networks. *Nature*, 393:440 – 442, June 1998.
- [44] Stanley Milgram. The Small-World Problem. *Psychology Today*, 2(1):60 – 67, 1967.
- [45] Lav R. Varshney, Beth L. Chen, Eric Paniagua, David H. Hall, and Dmitri B. Chklovskii. Structural Properties of the *Caenorhabditis elegans* Neuronal Network. *PLoS computational biology*, 7(2), 2011.
- [46] Åke J. Holmgren. Using Graph Models to Analyze the Vulnerability of Electric Power Networks. *Risk Analysis*, 26(4):955 – 969, 2006.

- [47] Shinako Matsuyama and Takao Terano. Analyzing the ENRON Communication Network Using Agent-Based Simulation. *Journal of Networks*, 3(7), 2008.
- [48] Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65(2), 2002.
- [49] Daniel Burda and Frank Teuteberg. Towards an Understanding of Needs, Capabilities and Alignment Mechanisms in Digital Preservation: Results from an Explorative Case Study. In *Wirtschaftsinformatik*, page 50, 2013.
- [50] Simona Rabinovici-Cohen, John Marberg, and Kenneth Nagin. Preservation DataStores in the Cloud (PDS Cloud): Long Term Digital Preservation in the Cloud. Technical report, Technical Report H-0318, IBM Research-Haifa, 2013.
- [51] W3C Technical Architecture Group. Architecture of the World Wide Web, Volume One. Technical report, World Wide Web Consortium, 2004.
- [52] Robert Kahn and Robert Wilensky. A Framework for Distributed Digital Object Services. *Int. Journal on Digital Libraries*, 6(2):115 – 123, 2006.
- [53] Vicky Reich and David S. H. Rosenthal. LOCKSS: A Permanent Web Publishing and Access System. *D-Lib Magazine*, 7(6), 2001.
- [54] Frank McCown, Michael L. Nelson, and Herbert Van de Sompel. Everyone is a Curator: Human-Assisted Preservation for ORE Aggregations. *Proc. of the DigCCurr*, Jan 2009.
- [55] Herbert Van de Sompel, Sandy Payette, John Erickson, Carl Lagoze, and Simeon Warner. Rethinking Scholarly Communication. *D-Lib Magazine*, 10(9), 2004.
- [56] Carl Lagoze, Herbert Van de Sompel, Michael L. Nelson, Simeon Warner, Robert Sanderson, and Pete Johnston. Object Re-Use and Exchange: A Resource-Centric Approach. *arXiv preprint arXiv:0804.2273*, 2008.
- [57] Jeff Rothenberg. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation. A Report to the Council on Library*

- and Information Resources*. Council on Library and Information Resources, 1999.
- [58] Kenneth Thibodeau. Building the Archives of the Future: Advances in Preserving Electronic Records at the National Archives and Records Administration. *D-Lib Magazine*, 7(2), 2001.
- [59] J. R. van der Hoeven, R. J. van Diessen, and K. van der Meer. Development of a Universal Virtual Computer (UVC) for long-term preservation of digital objects. *Journal of Information Science*, 31(3):196, 2005.
- [60] Donald Waters and John Garrett. *Preserving Digital Information. Report of the Task Force on Archiving of Digital Information*. The Commission on Preservation and Access, 1996.
- [61] Mary Baker, Kimberly Keeton, and Sean Martin. Why Traditional Storage Systems Don't Help Us Save Stuff Forever. In *Proc. 1st IEEE Workshop on Hot Topics in System Dependability*, pages 2005 – 120, 2005.
- [62] CCSDS. Reference Model for an Open Archival Information System (OAIS). Technical report, Consultive Committee for Space Data Systems 650.0-M-2, Magenta Book, 2012.
- [63] Manish Parashar and Salim Hariri. *Autonomic Computing: Concepts, Infrastructure, and Applications*. CRC Press, 2006.
- [64] Christopher Hartman and Bedřich Beneš. Autonomous Boids. *Computer Animation and Virtual Worlds*, 17(3-4):199 – 206, 2006.
- [65] Nicholas Carriero and David Gelernter. Linda in Context. *Communications of the ACM*, 32(4):444 – 458, 1989.
- [66] Adriana Iamnitchi, Matei Ripeanu, and Ian Foster. Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations. In *Peer-to-Peer Systems*, volume 2429, pages 232 – 241, 2002.
- [67] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69, 2004.

- [68] Sawood Alam, Charles L. Cartledge, and Michael L. Nelson. HTTP Mailbox - Asynchronous RESTful Communication. Technical report, arXiv:1305.1992, Old Dominion University, Computer Science Department, Norfolk, VA, 2013.
- [69] Adilson E. Motter and Ying-Cheng Lai. Cascade-based attacks on complex networks. *Physical Review E*, 66(6), 2002.
- [70] Walid Najjar and Jean-Luc Gaudiot. Network Resilience: A Measure of Network Fault Folerance. *IEEE Transactions on Computers*, 39(2):174 – 181, 1990.
- [71] Amos Fiat and Jared Saia. Censorship Resistant Peer-to-Peer Content Addressable Networks. In *Proc. of the Thirteenth Annu. ACM-Society for Industrial and Applied Mathematics Symposium on Discrete Algorithms*, pages 94 – 103. Society for Industrial and Applied Mathematics, 2002.
- [72] Petter Holme, Beom Jun Kim, Chang No Yoon, and Seung Kee Han. Attack vulnerability of complex networks. *Physical Review E*, 65(5), 2002.
- [73] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and Attack Tolerance of Complex Networks. *Nature*, 406(6794):378 – 382, July 2000.
- [74] Y. Moreno, J. B. Gomez, and A. F. Pacheco. Instability of Scale-Free Networks Under Node-Breaking Avalanches. *Europhysics Letters*, 58(4):630 – 636, 2002.
- [75] Petter Holme and Beom Jun Kim. Vertex overload breakdown in evolving networks. *Physical Review E*, 65(6), 2002.
- [76] Paolo Crucitti, Vito Latora, Massimo Marchiori, and Andrea Rapisarda. Error and attack tolerance of complex networks. *Physica A: Statistical Mechanics and its Applications*, 340(1-3):388 – 394, 2004.
- [77] Thomas Petermann and Paolo De Los Rios. Exploration of Scale-Free Networks. *The European Physical Journal B*, 38(2):201 — 204, 2004.
- [78] Gabor Csárdi and Tamas Nepusz. The igraph Software Package for Complex Network Research. *InterJournal Complex Systems*, 1695, 2006.
- [79] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2011. R package version 0.999375-50.

- [80] Béla Bollobás and Oliver Riordan. Robustness and Vulnerability of Scale-Free Random Graphs. *Internet Mathematics*, 1(1):1 – 35, 2004.
- [81] R. Criado, J. Flores, B. Hernández-Bermejo, J. Pello, and M. Romance. Effective measurement of network vulnerability under random and intentional attacks. *Journal of Mathematical Modelling and Algorithms*, 4(3):307 – 316, 2005.
- [82] Gunnar W. Klau and René Weiskircher. Robustness and Resilience. *Network Analysis: Methodological Foundations*, pages 417 – 437, 2005.
- [83] Ulrik Brandes and Thomas Erlebach. *Network Analysis: Methodological Foundations*. Springer Verlag, 2005.
- [84] Hamilton Link, Randall A. LaViolette, Jared Saia, and Terran Lane. Parameters Affecting the Resilience of Scale-Free Networks to Random Failures. Technical report, University of New Mexico, 2005.
- [85] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Resilience of the Internet to Random Breakdowns. *Physical Review Letters*, 85(21):4626 – 4628, 2000.
- [86] Sergiu Netotea and Sándor Pongor. Evolution of robust and efficient system topologies. *Cellular Immunology*, 244(2):80 – 83, 2006.
- [87] Soo Kim, Heejo Lee, and Wan Yoen Lee. Improving Resiliency of Network Topology with Enhanced Evolving Strategies. In *Proc. of the Sixth IEEE Int. Conf. on Computer and Information Technology*. Citeseer, 2006.
- [88] Heejo Lee, Jong Kim, and Wan Leon Lee. Resiliency of Network Topologies under Path-Based Attacks. *IEICE Transactions on Communications*, 89(10):2878, 2006.
- [89] J. P. Kownacki. Robustness of planar random graphs to targeted attacks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008.
- [90] Duncan J. Watts. Networks, Dynamics, and the Small-World Phenomenon. *The American Journal of Sociology*, 105(2):493 – 527, 1999.

- [91] Luis A. Nunes Amaral, Antonio Scala, Marc Barthélemy, and H. Eugene Stanley. Classes of Behavior of Small-World Networks. *Statistical Mechanics*, Jan 2000.
- [92] Andrei Broder, Ravi Kumar, Frazin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph Structure in the Web. *Computer Networks*, 33(1-6):309 – 320, 2000.
- [93] Duncan S. Callaway, M. E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Network Robustness and Fragility: Percolation on Random Graphs. *Physical Review Letters*, 85(25), 2000.
- [94] Jon Kleinberg. The Small-World Phenomenon and Decentralized Search. *Society for Industrial and Applied Mathematics News*, 37(3):1 – 2, 2004.
- [95] Jon Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proc. of the 32nd ACM Symposium on Theory of Computing*, volume 32, pages 163 – 170, 2000.
- [96] Nisha Mathias and Venkatesh Gopal. Small-Worlds: How and why. *Disordered Systems and Neural Networks*, 2000.
- [97] M. E. J. Newman. Models of the Small World: A Review. *Journal of Statistical Physics*, 101:819, 2000.
- [98] M. E. J. Newman, Duncan J. Watts, and Steven H. Strogatz. Random Graph Models of Social Networks. *Proc. of the Nat. Academy of Sciences*, 99(Suppl 1), 2002.
- [99] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The Architecture of Complex Weighted Networks. *Proc. of the Nat. Academy of Sciences*, 101(11):3747 – 3752, 2004.
- [100] Ken Y. K. Hui, John C. S. Lui, and David K. Y. Yau. Small World Overlay P2P Networks. In *12th IEEE Int. Workshop on Quality of Service*, pages 201 – 210, 2004.
- [101] Anthony Bonato. A survey of models of the web graph. In *Combinatorial and Algorithmic Aspects of Networking*, pages 159 – 172. Springer, 2004.

- [102] Rama Cont and Emily Tanimura. Small World Graphs: Characterization and Alternative Constructions. *Advances in Applied Probability*, 40(4):939 – 965, 2008.
- [103] R. Duncan Luce and Albert D. Perry. A Method of Matrix Analysis of Group Structure. *Psychometrika*, 14(2):95 – 116, 1949.
- [104] Miroslav Fiedler. Algebraic Connectivity of Graphs. *Czechoslovak Mathematical Journal*, 23(2):298 – 305, 1973.
- [105] Bernardo A. Huberman and Lada A. Adamic. Growth dynamics of the World-Wide Web. *Nature*, 401(6749):131, 1999.
- [106] Milan Randić and Luz M. DeAlba. Dense Graphs and Sparse Matrices. *Journal of Chemical Information and Computer Science*, 37(6):1078 – 1081, 1997.
- [107] Albert-László Barabási. The physics of the Web. *Physics World*, 14(7):33 – 38, 2001.
- [108] Ulrik Brandes. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25(2):163 – 177, 2001.
- [109] Brian Cooper and Hector Garcia-Molina. Creating Trading Networks of Digital Archives. In *Proc. of the 1st ACM/IEEE-CS Joint Conf. on Digital Libraries*, pages 353 – 362. ACM, 2001.
- [110] K. I. Goh, B. Kahng, and D. Kim. Universal Behavior of Load Distribution in Scale-Free Networks. *Physical Review Letters*, 87(27), December 2001.
- [111] Filippo Menczer. Growing and navigating the small world web by local content. *Proc. of the Nat. Academy of Sciences*, 99(22):14014, 2002.
- [112] M. E. J. Newman. Random graphs as models of networks. *Wiley Online Library*, 2003.
- [113] Xiao Fan Wang and Guanrong Chen. Complex Networks: Small-World, Scale-Free and Beyond. *IEEE circuits and systems magazine*, 3(1):6 – 20, 2003.
- [114] M. E. J. Newman and M. Girvan. Finding and Evaluating Community Structure in Networks. *Physical Review E*, 69(2), 2004.

- [115] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and Identifying Communities in Networks. *Proc. of the Nat. Academy of Sciences*, 101(9):2658 – 2663, 2004.
- [116] Karl Aberer, Luc Onana Alima, Ali Ghodsi, Sarunas Girdzijauskas, Seif Haridi, and Manfred Hauswirth. The Essence of P2P: A Reference Architecture for Overlay Networks. In *Fifth IEEE Int. Conf. on Peer-to-Peer Computing, 2005. P2P 2005*, pages 11 – 20, 2005.
- [117] Stephen P. Borgatti. Centrality and Network Flow. *Social Networks*, 27(1):55 – 71, 2005.
- [118] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs Over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177 – 187, 2005.
- [119] Scott White and Padhraic Smyth. A Spectral Clustering Approach to Finding Communities in Graphs. In *Society for Industrial and Applied Mathematics Data Mining Conf.* Citeseer, 2005.
- [120] Stephen P. Borgatti and Marin G. Everett. A Graph-theoretic perspective on centrality. *Social networks*, 28(4):466 – 484, 2006.
- [121] M. E. J. Newman. Modularity and Community Structure in Networks. *Proc. of the Nat. Academy of Sciences*, 103(23), 2006.
- [122] M. E. J. Newman, Albert-László Barabási, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.
- [123] Felix Halim, Yongzheng Wu, and Roland H. C. Yap. Small World Networks as (Semi)-Structured Overlay Networks. *Decentralized Self Management for Grids, P2P, and User Communities*, page 25, 2008.
- [124] Martin Rosvall and Carl T. Bergstrom. Maps of Random Walks on Complex Networks Reveal Community Structure. *Proc. of the Nat. Academy of Sciences*, 105(4):1118, 2008.
- [125] Katharina A. Zweig and Karin Zimmermann. Wanderer between the worlds-Self-Organized Network Stability in Attack and Random Failure Scenarios. In

- Proc. of the 2008 2nd IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems*, pages 309 – 318. IEEE Computer Society, 2008.
- [126] M. Karsai, M. Kivelä, R. K. Pan, K. Kaski, J. Kertész, A. L. Barabási, and J. Saramäki. Small But Slow World: How Network Topology and Burstiness Slow Down Spreading. *Physical Review E*, 83, 2011.
- [127] David Bearman. Archival Methods. *Archives and Museum Informatics*, 3(1), 1989.
- [128] William Y. Arms. Key Concepts in the Architecture of the Digital Library. *D-Lib Magazine*, 1(1), 1995.
- [129] Michael Lesk. Preserving Digital Objects: Recurrent Needs and Challenges. In *Proc. of the 1995 Nat. Preservation Office (NPO) Conf.*, pages 27 – 30, 1995.
- [130] Ron Daniel and Carl Lagoze. Extending the Warwick Framework: From Metadata Containers to Active Digital Objects. *D-Lib Magazine*, 3(11), 1997.
- [131] Andrew V. Goldberg and Peter N. Yianilos. Towards an Archival Intermemory. In *Proc. of the Advances in Digital Libraries Conf.*, pages 147 – 156, 1998.
- [132] Jari Urpalainen. An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors. *RFC*, (5261), 2008.
- [133] Carl Lagoze, Herbert Van de Sompel, Pete Johnston, Michael Nelson, Robert Sanderson, and Simeon Warner. ORE User Guide - Resource Map Implementation in Atom. Technical report, Open Archives Initiative, 2004.
- [134] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. Extensible Markup Language (XML). *World Wide Web Consortium Recommendation REC-xml-19980210*, August 2006.
- [135] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. RFC 2616, Hypertext Transfer Protocol—http/1.1, 1999. URL <http://www.rfc.net/rfc2616.html>, 1999.

- [136] Andrew Waugh, Ross Wilkinson, Bredan Hills, and Jon Dell’Oro. Preserving Digital Information Forever. In *Proc. of the 5th ACM Conf. on Digital Libraries*, pages 175 – 184. ACM, 2000.
- [137] Michael L. Nelson and Kurt Maly. Smart Objects and Open Archives. *D-Lib Magazine*, 7(2), 2001.
- [138] Michael L. Nelson and B. Danette Allen. Object Persistence and Availability in Digital Libraries. *D-Lib magazine*, 8(1), 2002.
- [139] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41 – 50, 2003.
- [140] Carolyn E. Lipscomb. Librarian supply and demand. *Journal of the Medical Library Association*, 91(1):7, 2003.
- [141] John Markwell and David W. Brooks. Broken Links: Just How Rapidly Do Science Education Hyperlinks Go Extinct? URL: http://www-class.unl.edu/biochem/url/broken_links.html, 18, 2003.
- [142] Norman Paskin. On Making and Identifying a Copy. *D-Lib Magazine*, 9(1), 2003.
- [143] Jane Hunter and Sharmin Choudhury. A Semi-Automated Digital Preservation System Based on Semantic Web Services. In *Proc. of the 4th ACM/IEEE-CS Joint Conf. on Digital Libraries*, pages 269 – 278, 2004.
- [144] Richard Anderson, Hannah Frost, Nancy Hoebelheinrich, and Keith Johnson. The AIHT at Stanford University: Automated Preservation Assessment of Heterogeneous Digital Collections. *D-Lib Magazine*, 11(12), 2005.
- [145] Clay Shirky. AIHT: Conceptual Issues from Practical Tests. *D-Lib Magazine*, 11(12), 2005.
- [146] Petros Maniatis, Mema Roussopoulos, T. J. Giuli, David S. H. Rosenthal, and Mary Baker. The LOCKSS Peer-to-Peer Digital Preservation System. *ACM Transactions on Computer Systems*, 23(1):2 – 50, 2005.
- [147] Frank McCown, Sheffan Chan, Michael L. Nelson, and Johan Bollen. The Availability and Persistence of Web References in D-Lib Magazine. In *Proc. of the 5th Int. Web Archiving Workshop and Digital Preservation*, 2005.

- [148] Michael L. Nelson, Johan Bollen, Giridhar Manepalli, and Rabia Haq. Archive Ingest and Handling Test. *D-Lib Magazine*, 11(12), 2005.
- [149] David S. H. Rosenthal, Thomas S. Robertson, Tom Lipkis, Vicky Reich, and Seth Morabito. Requirements for Digital Preservation Systems: A Bottom-Up Approach. *D-Lib Magazine*, 11(11), 2005.
- [150] Clay Shirky. Library of Congress Archive Ingest and Handling Test (AIHT) Final Report. *Report by the Nat. Digital Information Infrastructure and Preservation Program*, 2005.
- [151] Reagan Moore, Arcot Rajasekar, Michael Wan, Wayne Schroeder, and Richard Marciano. On Building Trusted Digital Preservation Repositories. In *5th e-Science All Hands Meeting*, 2006.
- [152] Julie Allinson, Sebastien François, and Stuart Lewis. SWORD: Simple Web-service Offering Repository Deposit. *Ariadne*, 54, April 2008.
- [153] Carl Lagoze, Herbert Van de Sompel, Pete Johnston, Michael L. Nelson, Robert Sanderson, and Simeon Warner. Object Re-Use and Exchange: A Resource-Centric Approach. *arXiv:0804.2273*, 2008.
- [154] Caryn Wojcik, Richard Marciano, Chien-Yi Hou, Reagan Moore, and Mark Conrad. The Perspectives of Digital Curators on Building Distributed Repositories. Technical report, University of North Carolina, 2007.
- [155] Committee on Government Operations. Taking a Byte Out of History: The Archival Presentation of Federal Computer Records. *House of Representatives Committee on Government Operations*, 1990.
- [156] Clifford A. Lynch. Institutional Repositories: Essential Infrastructure for Scholarship in the Digital Age, February 2003.
- [157] Rachel Heery and Sheila Anderson. Digital Repositories Review. Technical Report 23566, Joint Information Systems Committee, 2005.
- [158] Rachel Heery and Andy Powell. Digital Repositories Roadmap: looking forward. Technical report, University of Bath, 2006.

- [159] Robert K. Merton. *On the Shoulders of Giants: The Post-Italianate Edition*. University of Chicago Press, 1965.
- [160] Van Nguyen and Chip Martel. Analyzing and characterizing small-world graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 311 – 320, 2005.
- [161] Konstantin Klemm and Víctor M. Eguíluz. Growing Scale-Free Networks with Small-World Behavior. *Physical Review E*, 65(5), 2002.
- [162] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-Free Characteristics of Random Networks: The Topology of the World Wide Web. *Physica A*, 281(1):69 – 77, June 2000.
- [163] Philippe Duchon, Nicolas Hanusse, Emmanuelle Lebhar, and Nicolas Schabanel. Towards Small World Emergence. In *ACM Symposium on Parallelism in Algorithms and Architectures*, pages 225 – 232, 2006.
- [164] Béla Bollobás, Oliver Riordan, Joel Spencer, and Gabor Tusnády. The Degree Sequence of a Scale-Free Random Graph Process. *Random Structures and Algorithms*, 18(3):279 – 290, 2001.
- [165] Bruno Gaume and Fabien Mathieu. From Random Graph to Small World by Wandering. Technical Report 6489, Unité de recherche INRIA Rocquencourt, 2008.
- [166] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall Upper Saddle River, NJ, 2001.
- [167] Béla Bollobás. *Modern Graph Theory*. Springer Verlag, 1998.
- [168] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:96 – 115, 1927.
- [169] Frank Kammer and Hanjo Täubig. Connectivity. *Network Analysis, Lecture Notes in Computer Science*, pages 143 – 177, 2005.
- [170] Réka Albert and Albert-László Barabási. Statistical Mechanics of Complex Networks. *Reviews of Modern Physics*, 74(1):47, 2002.

- [171] A. Barrat and M. Weigt. On the Properties of Small-World Network Models. *The European Physical Journal B - Condensed Matter*, 13(3):547 – 560, January 2000.
- [172] Charles L. Cartledge and Michael L. Nelson. Connectivity Damage to a Graph by the Removal of an Edge or Vertex. Technical report, arXiv 1103.3075, Old Dominion University, Computer Science Department, Norfolk, VA, 2011.
- [173] Kenneth P. Birman, Mark Hayden, Ozgur Ozkasap, Zehn Xiao, Mihai Budiu, and Yaron Minsky. Bimodal Multicast. *ACM Transactions on Computer Systems*, 17(2):41 – 88, 1999.
- [174] Satoshi Ikeda, Izumi Kubo, and Masafumi Yamashita. The hitting and cover times of random walks on finite graphs using local degree information. *Theoretical Computer Science*, 410(1):94 – 100, 2009.
- [175] David Gelernter and Nicholas Carriero. Coordination Languages and their Significance. *Communications of the ACM*, 35(2):97 – 107, 1992.
- [176] P. Ciancarini, R. Gorrieri, and G. Zavattaro. Towards a calculus for generative communication. *Formal Methods for Open Object-Based Distributed Systems*, 1:283, 1997.
- [177] Irwin Miller and John E. Freund. *Probability and Statistics for Engineers*. Prentice-Hall Englewood Cliffs, NJ, 1977.
- [178] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167 – 256, 2003.
- [179] Tore Opsahl and Pietro Panzarasa. Clustering in Weighted Networks. *Social Networks*, 31(2):155 – 163, 2009.
- [180] Carter Butts. SNA: Tools for Social Network Analysis. URL <http://erzuli.ss.uci.edu/R.stuff>, 2009.
- [181] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz-open source graph drawing tools. In *Graph Drawing*, pages 483 – 484. Springer, 2002.

- [182] Vladimir Batagelj and Andrej Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47 – 57, 1998.
- [183] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, January 1995.
- [184] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Emergence of scaling in random networks. *Science*, 286(5439):509 – 512, October 1999.
- [185] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The Massachusetts Institute of Technology, 1990.
- [186] Michael Nielsen. How to crawl a quarter billion webpages in 40 hours. <http://www.michaelnielsen.org/ddi/how-to-crawl-a-quarter-billion-webpages-in-40-hours/>.
- [187] Sawood Alam, Charles L. Cartledge, and Michael L. Nelson. Support for Various HTTP Methods on the Web. Technical report, arXiv:1405.2330, Old Dominion University, Computer Science Department, Norfolk, VA, 2014.
- [188] Scott L. Feld. Why Your Friends Have More Friends Than You Do. *American Journal of Sociology*, 96(6), 1991.
- [189] Sawood Alam. HTTP Mailbox - Asynchronous RESTful Communication. Master’s thesis, Old Dominion University, Norfolk, VA, 2013.
- [190] Herbert Van de Sompel, Robert Sanderson, Martin Klein, Michael Nelson L., Bernhard Haslhofer, Simeon Warner, and Carl Lagoze. A Perspective on Resource Synchronization. *D-Lib Magazine*, 18(9):3, 2012.
- [191] Jeffrey K. Hollingsworth and Ethan L. Miller. Using Content-Derived Names for Caching and Software Distribution. Technical Report CS-TR-3671 UMIACS-TR-96-55, University of Maryland Institute for Advanced Computer Studies, 1998.
- [192] Jon Crowcroft, Steven Hand, Richard Mortier, Timothy Roscoe, and Andrew Warfield. Plutarch: An Argument for Network Pluralism. In *ACM SIGCOMM Computer Communication Review*, volume 33, pages 258 – 266. ACM, 2003.

- [193] Space Data and Information Transfer Systems – Open Archival Information System (OAIS) – Reference Model. ISO 14721:2012, 2012.
- [194] Brian Lavoie. Meeting the challenges of digital preservation: The OAIS reference model. *OCLC Newsletter*, 243:26–30, 2000.
- [195] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–HTTP/1.1. Request for Comment (RFC) 2616, June, 1999.
- [196] Neil Beagrie and Maggie Jones. Preservation Management of Digital Materials: The Handbook, 2007.
- [197] Alina Beygelzimer, Geoffrey Grinstein, Ralph Linsker, and Irina Rish. Improving Network Robustness by Edge Modification. *Physica A: Statistical Mechanics and its Applications*, 357(3-4):593 – 612, 2005.
- [198] Abhinay Venuturumilli and Ali Minai. Obtaining Robust Wireless Sensor Networks Through Self-organization of Heterogeneous Connectivity. In *Proc. of the 6th Int. Conf. on Complex Systems*. Citeseer, 2006.
- [199] Dirk Koschützki, Katharina Lehmann, Leon Peeters, Stefan Richter, Dagmar Tenfelde-Podehl, and Oliver Zlotowski. Centrality Indices. *Network Analysis, Lecture Notes in Computer Science*, pages 16 – 61, 2005.
- [200] Marc Barthélemy. Betweenness Centrality in Large Complex Networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):163 – 168, 2004.
- [201] Michael Brinkmeier and Thomas Schank. Network Statistics. *Network Analysis, Lecture Notes in Computer Science*, 3418:293 – 317, 2005.
- [202] Lie-Quan Lee, Andrew Lumsdaine, and Jeremy G. Siek. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Professional, 2002.
- [203] Chris Godsil and Gordon Royle. *Algebraic Graph Theory*. Springer New York, 2001.

- [204] J. H. Kwakkel, M. J. P. Mens, A. de Jong, J. A. Wardekker, W. A. H. Thissen, and J. P. van der Sluijs. *Uncertainty Terminology: Version 1.0*. National Research Programme Knowledge for Climate, 2011.
- [205] Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. *Random Graphs. Inter-science Series in Discrete Mathematics and Optimization*. John Wiley and Sons, 2000.
- [206] Nadine Baumann and Sebastian Stiller. Network Models. *Network Analysis, Lecture Notes in Computer Science*, 3418:341 – 341, 2005.
- [207] Gabor Csárdi. igraph: Routines for network analysis R package. *URL <http://cneurocv.s.rmki.kfki.hu/igraph>*, 2009.
- [208] Lukasz Kowali. Approximation Scheme for Lowest Outdegree Orientation and Graph Density Measures. *Algorithms and Computation*, pages 557–566, 2006.

VITA

Charles L. Cartledge
Department of computer science
Old Dominion University
Norfolk, VA 23529

Charles Lane Cartledge was born in Los Angeles, California on 2 March 1952. He was raised in central Alaska, graduating from high school in Fairbanks, Alaska in 1970, the University of Alaska with an AEET in 1972, Oregon Institute of Technology with a BEET in 1974 and Old Dominion University with a MS in CS in 2007. He has worked as a US government contractor since 1977 focusing on state-of-the-art defense technology for the Department of Defense, United States and foreign allied nations. Mr. Cartledges responsibilities have included the design of sonars, combat and data link systems and the enhancement and maintenance of world-class production and scheduling systems. He owns a proven record of delivering results through the use of technology and creative problem solving, as well as meeting challenges head on. Overseeing project management, timelines, schedules and assignments; calculating and managing the engineering budget; providing assistance and advice to middle and upper management in addition to internal and external customers. Mr. Cartledge attributes his success to his supportive wife, Mary, their son, Charles Lane and to his 31 year career in the United States Navy. Mentoring and identifying entry level personnel's potential; directing design and deployment of high-level defense projects; providing quality training to others; and helping solve critical system problems are some of his career highlights. He retired as a CAPT from the United States Navy. He is a Grand Croix in the Sovereign Military Order of the Temple of Jerusalem Knight Templar, a former Boy Scouts of America Scout Master, and a former board member of Vets House (a transitional house for US Veterans in need).