Eastern Michigan University
DigitalCommons@EMU

Master's Theses and Doctoral Dissertations

Master's Theses, and Doctoral Dissertations, and Graduate Capstone Projects

11-23-2009

A model to study cyber attack mechanics and denial-of-service exploits over the internet's router infrastructure using colored petri nets

Lawrence M. Healy

Follow this and additional works at: http://commons.emich.edu/theses

 Part of the Digital Communications and Networking Commons

Recommended Citation
Healy, Lawrence M., "A model to study cyber attack mechanics and denial-of-service exploits over the internet's router infrastructure using colored petri nets" (2009). *Master's Theses and Doctoral Dissertations*. 218.
http://commons.emich.edu/theses/218

This Open Access Dissertation is brought to you for free and open access by the Master's Theses, and Doctoral Dissertations, and Graduate Capstone Projects at DigitalCommons@EMU. It has been accepted for inclusion in Master's Theses and Doctoral Dissertations by an authorized administrator of DigitalCommons@EMU. For more information, please contact lib-ir@emich.edu.

A Model to Study Cyber Attack Mechanics and Denial-of-Service Exploits over the Internet's Router Infrastructure using Colored Petri Nets

Dissertation

Submitted to the College of Technology

Eastern Michigan University

By Lawrence M. Healy

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

Dissertation Committee:

Mary Brake, Ph.D., Dissertation Chair

Peter Stephenson, Ph.D., Norwich University

Elsa Poh, Ph.D.,

Ann Christiansen Remp, Ph.D.,

Andrew Ross, Ph.D.

November 23, 2009

© COPYRIGHT BY

LAWRENCE M. HEALY

2009

All Rights Reserved

ACKNOWLEDGEMENTS

First and most important, I thank my wife for her constant support during the peaks and valleys of this major life milestone. I can always count on her to keep me focused. Her patience and motivational support have been the foundation of my success. I would like to express my gratitude for the time and efforts of my dissertation committee. They have provided instrumental guidance during my transition into a researcher and scholarly writer. My research has benefited from the editorial support of Dr. Ann Remp and Dr. Mary Brake. I want to thank them both for helping me present my ideas in a concise and appropriate manner. Their patience and time have been instrumental in the presentation of my research. I also would like to thank Dr. Peter Stephenson for his mentorship. Often, his advice and forethought have enabled my thought processes in a fruitful direction. Peter has fed my intellectual curiosities with the seeds of ideas that have grown into this dissertation. I would also like to thank Dr. Andrew Ross and Dr. Elsa Poh, for undertaking the time-consuming task of reading, editing, and providing useful advice on ways to improve my research.

The process of earning a doctorate at my stage in life can be a delicate balancing act between family, financial, and research responsibilities. I would to thank my Mom for providing me with a foundation of self-confidence and self-worth required to attempt and succeed at this difficult task. In addition, I thank Norwich University Applied Research Institutes (NUARI), specifically Phil Susmann and Eric Braman, for enabling the mechanisms that have provided valuable financial support of this worthwhile academic endeavor. I appreciate their patience and support over the past few years.

ABSTRACT

The Internet's router infrastructure, a scale-free computer network, is vulnerable to targeted denial-of-service (DoS) attacks. Protecting this infrastructure's stability is a vital national interest because of the dependence of economic and national security transactions on the Internet. Current defensive countermeasures that rely on monitoring specific router traffic have been shown to be costly, inefficient, impractical, and reactive rather than anticipatory.

To address these issues, this research investigation considers a new paradigm that relies on the systemic changes that occur during a cyber attack, rather than individual router traffic anomalies. It has been hypothesized in the literature that systemic knowledge of cyber attack mechanics can be used to infer the existence of an exploit in its formative stages, before severe network degradation occurs. The study described here targeted DoS attacks against large-scale computer networks. To determine whether this new paradigm can be expressed though the study of subtle changes in the physical characteristics of the Internet's connectivity environment, this research developed a first of its kind Colored Petri Net (CPN) model of the United States AT&T router connectivity topology.

By simulating the systemic affects of a DoS attack over this infrastructure, the objectives of this research were to (1) determine whether it is possible to detect small subtle changes in the connectivity environment of the Internet's router connectivity infrastructure that occur during a cyber attack; and (2) if the first premise is valid, to ascertain the feasibility of using these changes as a means for (a) early infrastructure attack detection and (b) router infrastructure protection strategy development against these attacks.

Using CPN simulations, this study determined that systemic network changes can be detected in the early stages of a cyber attack. Specifically, this research has provided evidence

that using knowledge of the Internet's connectivity topology and its physical characteristics to protect the router infrastructure from targeted DoS attacks is feasible. In addition, it is plausible to use these techniques to detect targeted DoS attacks and may lead to new network security tools.

TABLE OF CONTENTS

List of Tables x

List of Figures xi

Chapter I: Introduction 1

Problem Statement 4

Research Statement 8

Research Contributions 9

Terminologies 10

Organization of the Dissertation 16

Chapter II: Foundations 17

Cyber Attacks 17

Complex Networks 22

Attack Vulnerabilities 26

Network Connectivity Stability 32

Topology Based Protection 35

Theoretical Rational Summary 36

Chapter III: Formal Definitions 39

Node State Data Structures 39

Topology Based Node Protection 48

Simulated Attack Definition 49

Computational Foundations 56

Chapter IV: Model and Simulation Design 62

Simulation Strategy 62

Foundational Assumptions 64
CPN Model and Simulation Language 65
General Research Simulation Design 72
Simulation Runs 78
Network Connectivity State Computations 85
Research CPN Model and Simulation Execution 88
Chapter V: Results 97
Model and Simulation Validation and Consistency 98
The Relationship between NCP and I 108
Node-pair and Information Transfer Correlations 109
Network Stability by Run Type 112
Attack Detection 120
Network Protection 122
Chapter VI: Discussion 124
Research Summary 124
Theoretical Implications of this Research 127
Practical Implications of this Research 130
Limitations 132
Chapter Summary 133
Chapter VII: Conclusions 134
Research Relevance 134
Attack Detection and Network Protection Application 135
Future Work 136

Summary 138

References 139

Appendix 152

A CPN Declarations 153

B CPN Function Code 156

C Simulation Run Data File Identification 174

D CPN Simulation Sub-Page Design Details 176

E Network Connectivity Parameter Results by Run Type 188

F Network Connectivity Parameter Results by Attack Class 191

G Network Connectivity Parameter Results by Attack Effect 197

H Run Type 2 Results for Network Stability and Node-Pair Type Counts 199

I Run Type 3 Results for Network Stability and Node-Pair Type Counts 203

J Information Transfer Results by Attack Class 207

K Node-pair Type 1-2 Counts Results by Attack Class 213

L Information Transfer Loss Results by Run Type 219

M Information Transfer Rate Loss Results by Run Type 222

N Node-pair type 1-2 Count Loss Results by Attack Class 223

O Node-pair Type 1-2 Counts Rate Loss by Run Type 226

P Attack Detection by Run Type 228

Q Attack Detection and Node-Pair Type 1-1 Counts Results by Run Type 231

R Attack Detection and Node-Pair Type 1-1 Counts Results for Fraction Changed by Run Type 237

S Attack Detection and Node-Pair Type 1-1 Counts Results for First 300 Seconds by Run Type 243

T Network Connectivity Parameter Efficiency Results by Attack Effect 246

U Information Transfer and Node-Pair Type Efficiency Results by Attack Effect 248

LIST OF TABLES

Table Page

3.1. Node state transition for Figure 3.1 40

3.2. Network connectivity terms used in the research computations 57

3.3. Pre-attack simulation baseline connectivity 59

4.1. Simulation runs denoting critical node removal proportions 75

4.2. Protection strategy 81

4.3. Overloaded orphan node profile in Figure 4.8 84

4.4. Main CPN page transitions and associated sub-pages 91

4.5. Summary of CPN places in Figure 4.11 93

5.1. Run type 1 simulations, entropy at total collapse at I = 0 107

5.2. Run type 1, potential attack markers for attack detection, node-pair counts having strong negative correlation with information transfer for the first 500 seconds 110

5.3. Run type 1, potential attack markers for network protection, node-pair counts having strong positive correlation with information transfer for the first 500 seconds 111

5.4. Attack detection percent variance, actual versus inferred 121

5.5. Network protection, equilibrium and network stability recovery for run type 4 123

6.1. Cyber attack mechanics hypothesis and this dissertation 128

LIST OF FIGURES

Figure Page

1.1. Information transfer (I) versus time curve showing descriptive research terms. Terms that depict network degradation are shown in a) and terms showing network recovery as a result of network protection are shown in b) 12

1.2. Node connectivity example 14

2.1. Error and attack tolerance study taken from (Crucitti et al. 2003b) 29

2.2. Error and attack tolerance over the Internet (scale-free), taken from (Albert, Jeong, and Barabasi 2000) 30

3.1. Node state diagram for the CPN simulations 41

3.2. Pre-attack network example of message path hops over small region 53

3.3. Attacked network example at some point after attack, message path change and fragmentation 54

3.4. Illustrative example of network fragmentation over time 55

3.5. Pre-attack network degree distribution 60

3.6. Pre-attack network rank/frequency plot (log-log scale) 60

4.1. Node-pairs and network connectivity model 63

4.2. Colored Petri Net example, before transition “A” fires 68

4.3. Colored Petri Net example, after transition “A” fires once 69

4.4. Colored Petri Net example, after transition “A” fires a second time 70

4.5. Summary of simulation process information flow 73

4.6. Summary of the node interactions during the simulation 77

4.7. Simulation execution strategy 79

4.8. Communication attack simulation example 83

4.9. POP backbone and access router architecture example 87

4.10. United States AT&T router backbone (layer 0) taken from (Spring, Mahajan, and Wetherall 2002) Image from: NASA's Visible Earth Project http://visibleearth.nasa.gov 88

4.11. CPN main page 90

4.12. CPN main page flow functionality 95

5.1. Run type 1, information transfer versus nodes removed fraction over time 100

5.2. Run type 1, connectivity parameter versus nodes removed fraction over time 101

5.3. Run type 1, fraction of all nodes removed versus simulation time 102

5.4. Run type 1, network connectivity parameter versus simulation time 104

5.5. Run type 1, avg. entropy versus avg. degree, combined runs, 50 sec. intervals 106

5.6. Run type 1, NCP and information transfer, 10 combined simulations 109

5.7. Run type 1, network stability, information transfer versus time 114

5.8. Run type 1, node-pair type 1-2 counts versus time 115

5.9. Run type 4, network stability, information transfer versus time 118

5.10. Run type 4, node-pair type 1-2 counts versus time 118

5.11. Attack detection, descriptive statistics over 40 simulation runs 122

I. INTRODUCTION

For the first time, Colored Petri Net (CPN) modeling and simulation techniques have been used to simulate targeted denial-of-service attacks over the Internet's router infrastructure. Cyber attacks were simulated against historic datasets collected over a specific time period using actual Internet router connectivity. The simulation was used to study changes in the Internet's connectivity state during a targeted denial-of-service (DoS) attack. Using scale-free network theory, this research sought to determine whether there is strong evidence that underlying network-wide *attack markers* exist. During the formative stages of a targeted denial-of-service exploit against a large-scale computer network these *attack markers* might be used to study cyber attack mechanics. *Attack markers*, as will be discussed, are subtle changes in Internet's connectivity during an attack.

Large-scale cyber attacks against the Internet's router infrastructure could lead to significant disruptions in global commerce as well as impede national security objectives. The Internet's router infrastructure is responsible for facilitating all communications over the Internet. A router is a special purpose computer on a network that is responsible for passing information between computers or other routers. Using routers, information is passed along a path from the original source computer to its ultimate destination computer. The Internet's router infrastructure is composed of thousands of connected routers that pass information around the world. The actual number of routers is dynamic because during normal Internet operations, routers are added and removed to adjust for changes in user demand, normal router failures, and maintenance periods.

Malicious attacks that impede robust and reliable Internet communications are becoming an ever-increasing problem. Over the last five years, computer vulnerabilities and

exploits have grown significantly and do not show any sign of slowing down. A 2008 FBI computer crime survey (Richardson 2008) reported that from 2004 to 2008, 45% to 55% of the organizations surveyed reported at least one occurrence of an unauthorized use of their computer systems. In addition, they found that over that same period, 25% of all security incidents were classified as denial-of-service attacks.

A denial-of-service attack generates large volumes of Internet messages over a short period of time. This traffic flood is aimed at one or many target destinations over a network. As a result of this onslaught of Internet traffic, the target(s) becomes overloaded and suffers severe service degradation. During a DoS attack, authorized users are denied access to computer systems (Douligeris and Mitrokotsa 2004; Olalekan 2008; Peng, Leckie, and Ramamohanarao 2007a). Denial-of-service attacks are a common tool used by Internet attackers to achieve malicious objectives (Cheol-Joo et al. 2007; Douligeris and Mitrokotsa 2004; Mirkovic and Reiher 2004; Olalekan 2008; Peng, Leckie, and Ramamohanarao 2007b; Richardson 2008) .

Two types of targets have been exploited by this mode of attack: enterprise and infrastructure. At the enterprise level, the target is usually specific commercial enterprise networks such as Microsoft. The primary objective is to severely disrupt communications within a few specific web sites on that network. At the infrastructure level, where the primary objective is service degradation over an entire large-scale network connecting many networks and web sites, the attacker seeks to disrupt normal information flow between the routers that facilitate network-wide communication. Prior to 2002, enterprise level networks were the primary targets for DOS attacks. Since 2002, the targets have also included the Internet's global router infrastructure (Cheung 2006; Dirk et al. 2004; Mizrak et al. 2006;

Olalekan 2008; Peng, Leckie, and Ramamohanarao 2007b). This shift signals a possible change in attacker motivations with potentially more far-reaching and dangerous impacts (Borchgrave et al. 2001; Mizrak et al. 2006).

Distinguishing between normal and malicious activity over the Internet is very difficult (Casey 2002; Casey 2004; Rattray 2001b). Denial-of-service attacks often involve multiple targets spread over a large-scale network. In its formative stages, a denial-of-service attack may appear in the network's system logs as normal but heavy Internet traffic. The DoS attack may not emerge as a threat to a network's availability until it has caused significant damage. One key reason for this detection latency is that cyber attack detection is a reactive process. It relies on the time-consuming and tedious examination of individual network router communication packets. This assessment is performed locally on a small portion of the network without a complete understanding of the potential broader network-wide ramifications of an attack. Often, the overall extent of the network-wide damage is not recognized until there is widespread and significant network connectivity instability.

A potentially more efficient technique for large-scale network-wide assessments is postulated by Stephenson and Prueitt (Stephenson 2006; Stephenson and Prueitt 2005). They argue that cyber attack mechanics over large-scale computer networks can be studied during the attack's formative stages through subtle changes in the network's environment. To prove the potential feasibility of their approach, the research in this dissertation developed a novel Colored Petri Net (CPN) model of the Internet's router connectivity and emulated targeted denial-of-service attacks against this simulated router infrastructure. The baseline data for this research simulation were extracted from Rocketfuel Internet maps collected by the University of Washington (Alderson et al. 2005; Rocketfuel: An ISP topology mapping

engine n.d.; Spring et al. 2004). Specifically, the simulation was based on a snapshot of the United States AT&T router backbone infrastructure consisting of 11,800 routers.

This research studied connectivity changes between individual adjacent routers during the denial-of-service attack simulation. At pre-determined time intervals during the simulated attack, the changes in individual router connectivity characteristics were aggregated into a network-wide connectivity state. Subtle changes in the overall network-wide connectivity states were studied for relevant patterns of behavior during the attack simulation. The techniques used in this research are a novel approach for studying networks under denial-of-service attack.

The simulations in this investigation have uncovered subtle fluctuations in the connectivity environment of the baseline data during the earliest stages of a simulated denial-of-service attack. The approach used in this research may lead to the development of Internet-wide security tools to prevent and anticipate DoS attacks against the Internet's router connectivity infrastructure.

A. Problem Statement

Over the past decade, scale-free networks have been discovered among biological, social, and technological communication systems (Alderson et al. 2005; Barabasi and Albert 2002; Boccaletti et al. 2006; Michalis, Petros, and Christos 1999). Empirical research has shown that the router connection activity of the Internet's infrastructure behaves as predicted by scale-free network theory (Barabasi and Albert 2002; Barabasi, Ravasz, and Vicsek 2001; Dorogovtsev and Mendes 2002; Gallos 2005; Michalis, Petros, and Christos 1999; Newman 2003). Empirical studies have shown that the Internet's communication infrastructure is

dependent upon a relatively few centralized, highly connected routers (critical nodes) that may be vulnerable to targeted denial-of-service attacks (Albert, Jeong, and Barabasi 2000; Barabasi and Albert 2002; Boccaletti et al. 2006; Cohen 2000, 2001; Crucitti, Latora, and Marchiori 2004; Gallos 2005; Latora and Marchiori 2005; Motter and Lai 2002; Olalekan 2008; Salla 2005; Sun et al. 2007; Wu et al. 2007).

In scale-free computer networks of over 4,000 nodes, systemically removing as little as 5 percent of the network's most connected routers, via targeted denial-of-service attacks, initiates a rapid cascading degradation of the network's global connectivity (Barabasi and Albert 2002; Cohen 2001; Gallos et al. 2006; Holme et al. 2002a; Lai, Motter, and Nishikawa 2004; Motter and Lai 2002; Wang et al. 2008). This cascaded degradation occurs as "… traffic is rerouted to bypass malfunctioning routers, eventually leading to an avalanche of overloads on other routers that are not equipped to handle extra traffic. The redistribution of the traffic can result in a congestion regime with a large drop in the performance" (Crucitti, Latora, and Marchiori 2004). Identifying cascaded denial-of-service attacks in their formative stages is a difficult task but is critical for impeding the cascading router failures (Cheetancheri et al. 2006; Cohen 2001; Dobson et al. 2007; Douligeris and Mitrokotsa 2004; Latora and Marchiori 2005; Lee et al. 2008; Lu et al. 2007; Motter 2004; Motter and Lai 2002; Peng, Leckie, and Ramamohanarao 2007a; Tsunoda et al. 2008; Wang et al. 2008).

The Internet's vulnerable router infrastructure along with identity concealment techniques (such as IP spoofing and packet redirection) and the extremely large volume of router-to-router communication messages have made identification of an attack on the Internet's core operations extremely difficult (Douligeris and Mitrokotsa 2004; Haggerty, Shi, and Merabti 2005; Lee et al. 2008; Mizrak et al. 2006; Mizrak, Savage, and Marzullo

2008; Peng, Leckie, and Ramamohanarao 2007a; Toby and Jun 2009; Tsunoda et al. 2008). Evidence of real world attacks dependent upon the application of cascading failures have not been cited in the literature. However, it is highly probable that an undetected attack that maliciously modifies the Internet's router tables could lead to extreme traffic congestion, sub-optimal message routing decisions and significantly lowered network throughput (Chakrabarti 2002; Cheol-Joo et al. 2007; Hussain, Heidemann, and Papadopoulos 2003; Markopoulou et al. 2008; Mirkovic and Reiher 2004; Olalekan 2008; Peng, Leckie, and Ramamohanarao 2007a).

Commercial transactions, the nation's critical physical infrastructures and national security have a profound dependence on the Internet's infrastructure. Individuals with hostile intent have ready access to easy-to-use malicious tools. Rapidly changing technology and the ever-increasing occurrence of system vulnerabilities present a significant threat to the nation's information infrastructure. The attackers may be malicious hackers, organized crime, terrorists or nation states (Adkins 2001; U. S. House 2005; U. S. Office of Science and Technology Policy 2006). Attack objectives may include (1) espionage against sensitive and poorly defended data in government and industry, (2) financial and identity fraud, (3) theft of financial or identity assets, (4) disruption of normal communications over Internet's router infrastructure, (5) coordinated physical and cyber attacks that hinder emergency response dependent upon Internet communications, (6) industrial process control of critical physical infrastructure such as the electrical grid, (7) terrorist targets such as chemical plants, (8) global financial transactions involving billions of dollars, and (9) offensive information warfare aimed at military targets connected to the Internet's infrastructure (U. S. Office of Science and Technology Policy 2006).

The Internet was designed to achieve openness in a research-driven environment. The openness of a collaborative research-driven environment is at odds with the objectives of most security audit functionalities. Investigation of network security incursions and network protective measures are severely inhibited by the Internet's inherent design assumptions, including (1) the user community would be trustworthy and not seek to obfuscate identity or manipulate the Internet's communication mechanisms for malicious purposes, (2) high-speed traffic and performance requirements were essential and therefore any attempts at significant tracking would be too costly in terms of system performance criteria, and (3) due to the large volume of packets in a relatively short timeframe, storage of packet information was not viable (Lipson 2002).

As previously discussed in this chapter, the Internet's router infrastructure is vulnerable to denial-of-service attacks. This vulnerability magnifies the aforementioned threat to the national information infrastructure. It is in the national interest to deter, uncover, and prosecute perpetrators. Tracking and investigating individual router communications attacks are costly, inefficient, and impractical (Casey 2002; Casey 2004; Mizrak, Savage, and Marzullo 2008; Rattray 2001a; Stephenson and Prueitt 2005). Solving the problem of attack detection, attribution, infrastructure protection, effective countermeasures, and attack retribution requires a new paradigm. This paradigm must rely on the systemic changes that occur during a cyber attack, not the individual changes (Mizrak et al. 2006; Overill 2007; Papadimitratos and Haas 2002; Stephenson 2006; Stephenson and Prueitt 2005). The problem addressed by this research is to determine whether this new paradigm can be expressed though the study of subtle changes in the degree distribution of the Internet's connectivity environment. Specifically, can these subtle changes be classified

as attack markers? Can these attack markers be used to detect and protect the Internet's router infrastructure from targeted denial-of-service attacks?

B. Research Statement

The first objective of this research was to develop a novel, first-of-its-kind Colored Petri Net (CPN) model of a large-scale regional (United States) Internet router connectivity topology. Using this model as the basis for the investigation of the systemic effects of targeted DoS attacks against the simulated Internet router infrastructure, the objectives of this research were to (1) determine whether it is possible to detect small subtle changes (*attack markers*) in the connectivity environment of the Internet's router connectivity infrastructure that occur during a cyber attack, and (2) if the first premise is valid, to ascertain the feasibility of using these changes as a means for (a) early infrastructure attack detection and (b) router infrastructure protection strategy development against these attacks. This investigation found strong evidence that *attack markers* exist and that they are quantifiable through common statistical characteristics of the network's connectivity topology.

C. Research Contributions

Basic Internet research will benefit from this novel model and simulation using Colored Petri Nets. The research simulation led to the formulation of the foundational justification for a unique approach to the study of denial-of-service cyber attacks. This dissertation has enhanced the body of knowledge in the application of attack modeling and simulation through the following contributions:

1. Provides evidence that *attack markers* that represent subtle changes in the Internet's router connectivity topology can be used as a means to uncover denial-of-service attacks against the Internet's router infrastructure.
2. Shows that the techniques presented in this research provide a feasible way to detect a cyber attack in its earliest stages.
3. Presents a plausible and scientifically sound approach for the study of cyber attack mechanics that will provide the basis for future development of practical Internet security tools.
4. Offers a unique quantifiable approach for illuminating *attack markers* as changes to the degree distribution during a targeted denial-of-service attack.
5. Enhances the scope of applications for Colored Petri Net (CPN) modeling and simulation of concurrent and complex network communications.
6. Develops the preliminary feasibility for future experimental research to empirically support Stephenson and Pruiett's theory of cyber attack mechanics (Stephenson 2006; Stephenson and Prueitt 2005).

D. Terminologies

The following definitions are necessary for understanding this dissertation. All node states will be formally defined in Chapter III of this dissertation. These and other terms will be further defined as needed later in this document.

adjacent node. An adjacent node is a node that has a direct communication link with another node (1-hop). It is often referred to as a neighbor node.

attack marker. An attack marker represents subtle changes in a scale-free network's connectivity that can be used to indicate a cyber attack's existence. Network connectivity corresponds to any physical characteristic of the network that facilitates inter-nodal communications. Here, network connectivity is represented by the number of node-pairs of type 1-1 and 1-2. The rationale for this selection will be presented in Chapter V. The terms *attack marker* and *changes in the number of node-pairs of types 1-1 and 1-2* will be used interchangeably in this document. Node-pair types are defined below.

critical threshold. As shown in Figure 1.1, the critical threshold represents the point in time during the simulation when the network connectivity stability rapidly degrades in a short period towards the terminal condition.

degree. A node's degree is the number of direct links between one node and all of its neighbor nodes. A node with "k" number of links is referred to as a k-degree node. The degree of a node is a rough measure of its connectivity. Figure 1.2 illustrates a simple network connectivity example. In Figure 1.2, the circles represent nodes, node "A" has a degree of 3 and node "B" has a degree of 4. A link between two nodes is represented by a line between the nodes. The term "degree" is used interchangeably with the term *node degree*.

degree distribution. The degree distribution of a network is the probability that a randomly selected node from a network is a k-degree node. The probability of k-degree node is $p(k) = \frac{n(k)}{n}$; where n is the number of nodes in the network and n(k) is the number of k-degree nodes. The distribution for all k-degree nodes in a network is one commonly used measurement for the network's overall connectivity.

equilibrium point and level. As shown in Figure 1.1, during the simulation the equilibrium point is encountered; at this point in time, the network's connectivity stability recovers to a constant equilibrium level for the remainder of the simulation's execution.

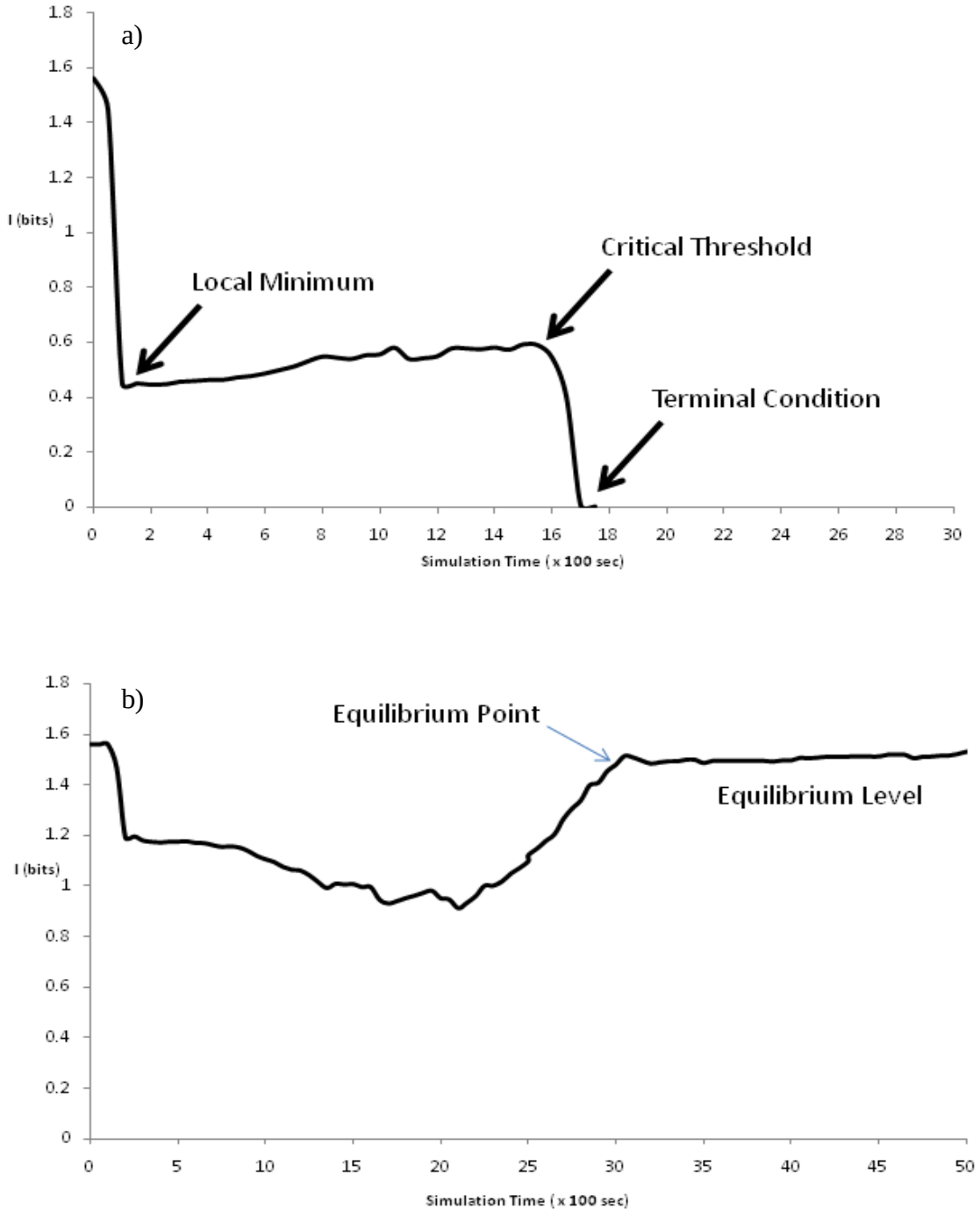


Figure 1.1. Information transfer (I) versus time curve showing descriptive research terms. Terms that depict network degradation are shown in A and terms showing network recovery as a result of network protection are shown in B.

mutual information transfer (I). Mutual information transfer is used in this research to measure relative network connectivity stability. Mutual information transfer (I) is the relative reduction in node connectivity uncertainty between two randomly selected nodes in the network (Cover and Thomas 2006). The reduction in uncertainty can be thought of as an increase in relative information (knowledge) transfer between any 2 randomly selected nodes. This increase in information leads to an increased likelihood that the two nodes will communicate as a node-pair. Node-pairs are defined below. From the network-wide perspective used in this research, mutual information transfer is the average reduction in uncertainty between 2 randomly selected nodes in the network at time t. Therefore, the average likelihood of connectivity between any two randomly selected nodes in the network ranges from very high (when I=2) to non-existent (when I = 0). The pre-attack network's information transfer value was 1.56, indicating that there was a relatively high likelihood for node-pair connectivity. The terms mutual information transfer and information transfer will be used interchangeably in this dissertation.

link. A link represents communication established with an adjacent node. One node may have multiple links. The number of links for a node is denoted as "k".

local minimum. As shown in Figure 1.1, this represents the point in time early in the simulation that the network connectivity stability has degraded to its lowest level before the critical threshold has been encountered.

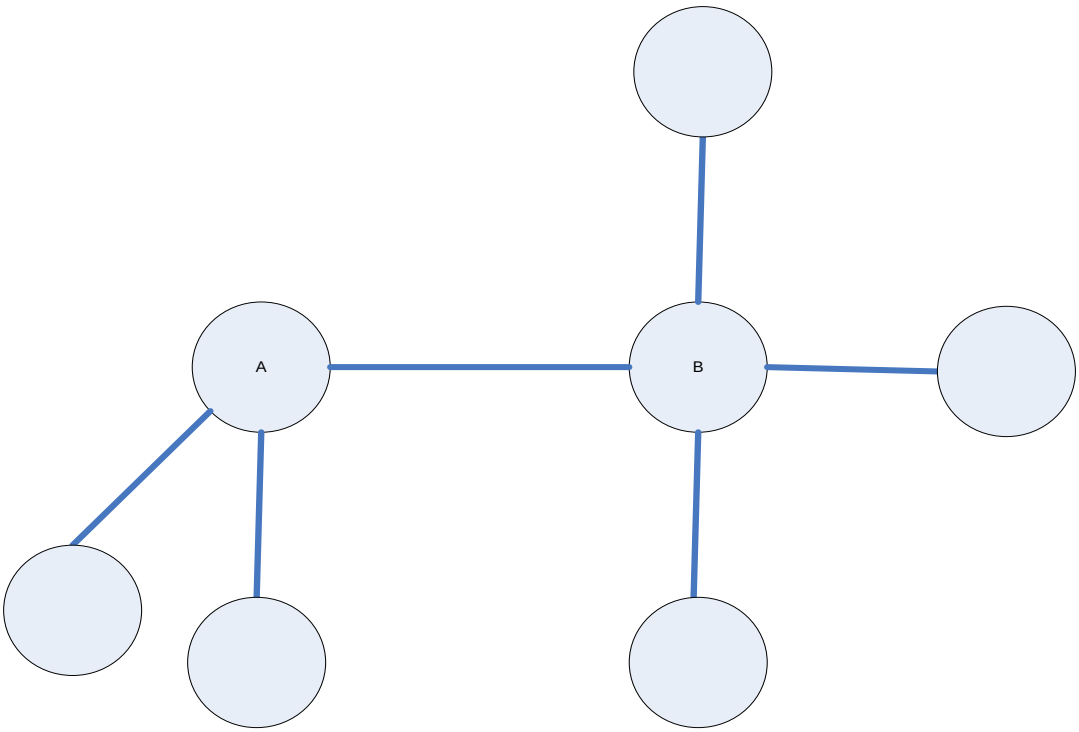


Figure 1.2. Node connectivity example

network connectivity parameter (NCP). This network connectivity stability measure is used to determine the relative extent of the overall network's fragmentation. The terms *network connectivity parameter* and *connectivity parameter* will be used interchangeably in this dissertation.

network connectivity stability. As defined by this research investigation, it is the ability for network-wide connectivity as measured by the mutual information transfer and network connectivity parameter. The terms *network connectivity stability* and *network stability* will be used interchangeably in this dissertation.

node. In this research, a node is an entity in a network that represents a router. The terms *node* and *router* will be used interchangeably in this dissertation. A network can have more than one node, and the network's size is represented by the number of nodes.

node degree. The number of neighbor node links for any a specific node. A node with k neighbor links is said to have a node degree of k.

node-pair type. For each simulation, routers were represented as nodes. Two adjacent (neighbor) routers as mapped through the router tables were depicted as node-pairs. All active node-pairs were classified into groups by determining the node degree of each node in the node-pair. These groups were called node-pair types. Each active node-pair was classified during the simulations at 50-second time intervals. All node-pair types were determined by combining the degree of each node in the pair. For example, if one node in a node-pair had a degree of 1 and the other node had a degree of 2, then this node-pair type was classified as type 1-2. The syntax for each node-pair type designation is (1) position 1 represents the node degree of the first node in the node-pair, (2) position 2 represents the node degree of the second node in the node-pair, and (3) position 1 in the node-pair type must always be less than or equal to position 2. Certain specific node-pair types were used as attack markers and will be presented in Chapter V, Section C.

power law degree distribution. Power laws are expressed in the form $y \propto x^a$; where x and y is the variables of interest, 'a' is constant, and $\propto$ indicates that the two variables have a proportional relationship. The power law polynomial relationship exhibits scale invariance; that is, the scaling coefficient (a) is constant. Another property of power law relationships is that a plot of log y versus log x (log-log plot) is linear. The slope of the resulting line for the log-log plot of $y = x^a$ is a constant value of 'a'. In terms of scale-free networks, the probability of any node in the network having k number of links is: $p(k) \propto k^{-\gamma}$; where γ is the constant scaling coefficient. It has been empirically determined that the scaling coefficient is between 2 and 3 for most "real world" networks (Barabasi and Albert 2002).

terminal condition. As shown in Figure 1.1, this represents the point in time during the simulation when the network-wide connectivity no longer exists. This occurs when the information transfer is less than zero and the network connectivity parameter is approximately 2.

E. Organization of the Dissertation

Foundational theories used by this research are discussed in Chapter II. Chapter III contains formalisms and terminologies used in this research. Chapter IV presents the CPN model design, its assumptions, and subsequent simulation details. Chapters V and VI discuss the results of the simulations and their relevance to real world router attacks. The conclusions are given in Chapter VII along with recommendations for further study.

CHAPTER II. FOUNDATIONS

This chapter will discuss the relevant background literature. The first section addresses cyber attacks. This is followed by a discussion of complex network theory. Then this chapter presents the literature support for the vulnerability of scale-free networks. Relevant network stability considerations are then addressed. This chapter concludes by covering network topology-based router protection strategies and summarizes the theoretical rationale used in this research.

A. Cyber Attacks

The background for the study of cyber attacks using the network's environment is presented in this section. This is followed by a discussion of attack modeling strategies. It concludes with a discourse on Internet infrastructure attack techniques.

1) Theory of Cyber Attack Mechanics.

Cyber attack mechanics have been studied extensively (Albert, Jeong, and Barabasi 2000; Chakrabarti 2002; Cohen 2001; Convery, Cook, and Franz 2004; Crucitti et al. 2004; Dirk et al. 2004; Douligeris and Mitrokotsa 2004; Gallos et al. 2006; Jung-Ying et al. 2008; Lai, Motter, and Nishikawa 2004; Lu et al. 2007; Mirkovic and Reiher 2004; Motter and Lai 2002; Peng, Leckie, and Ramamohanarao 2007a; Salla 2005; Shannon et al. 2006; Sun et al. 2007; Ziviani et al. 2007). The literature supports the proposition that the Internet's connectivity topology exhibits fractal (self-similar) properties relative to geographic population centers (Caldarelli, Marchetti, and Pietronero 2000; Chakraborty et al. 2004; Lakhina et al. 2002; Yook, Jeong, and Barabasi 2002). It may be plausible that the scale-free connectivity behaviors observed over the Internet are in some manner a manifestation of the

Internet's fractal connectivity topology (Stephenson 2006). One novel approach that may provide a foundational premise for anomaly detection over the Internet is known as the "Theory of Cyber Attack Mechanics" (Stephenson and Prueitt 2005). Using a unique set of foundational concepts and formalisms, Stephenson and Prueitt (2005) theorize that it may be possible to identify a cyber attack's origin by observing traffic disruptions in the Internet's fractal connectivity. As related to network communication connectivity and very large computer networks (such as the Internet), Stephenson and Prueitt (2005) argue that cyber attack detection using the network's environment is plausible (Stephenson 2006; Stephenson and Prueitt 2005).

During a cyber attack, the theory defines halting conditions as a set of specific network conditions that exist when the network's stability becomes totally degraded and network connectivity ceases. Stephenson and Prueitt (2005) propose that halting conditions may be observable through disruptions in the Internet's traffic mechanisms as identifiable events (attack markers). The "Theory of Cyber Attack Mechanics" leads Stephenson (2006) to postulate that the halting conditions brought on by the aforementioned disruptions in the Internet's fractal connectivity may be a result of an underlying violation of the Internet's preferential attachment linking rules and these violations may possibly be observable as event markers (attack markers). Empirical studies found in the literature support their hypothesis, suggesting that disruptions in normal Internet traffic patterns can be observed (Liljenstam et al. 2002; Yegneswaran, Barford, and Ullrich 2003).

Stephenson and Prueitt's (2005) hypothesis states that "The interaction between cyber attack space and fractal network space results in event markers that may anticipate the existence of a cyber attack. Because 'attack markers' are complex, they may result in

'halting conditions' within the network. These halting conditions can be represented formally and the source of the cyber attack may be deduced". The attack simulations in this research sought to uncover subtle variations in the network's connectivity patterns. These "ambiguities" were represented as changes in specific physical characteristics of the network's topology.

2) Attack Model Considerations.

A variety of attack modeling approaches can be found throughout recent literature (Albert, Jeong, and Barabasi 2000; Convery, Cook, and Franz 2004; Crucitti, Latora, and Marchiori 2004; Crucitti et al. 2004; Dirk et al. 2004; Gallos et al. 2006; Li et al. 2008; Motter 2004; Motter and Lai 2002; Olalekan 2008; Ole Martin Dahl and Wolthusen 2006; Wang and Rong 2009a, 2009b; Wang, Guan, and Lai 2009; Zhao et al. 2005). There are six fundamental attributes that should be considered when modeling cyber attacks: (1) consideration of the network's topology characteristics, (2) the attacker's system privileges, (3) the network's trust model, (4) the type of probable exploits, (5) the attack motivations, and (6) the attacker's specific knowledge of the target (Chakrabarti 2002; Jung-Ying et al. 2008; Olalekan 2008; Richardson 2008; Zhang et al. 2008).

When modeling Internet infrastructure attacks, it is commonly assumed that the attacker has knowledge of a network's topology (Gallos et al. 2006; Wu et al. 2007; Zhang et al. 2008). The attack scenarios used in the research simulations were based on scale-free network theory. The rationale for developing attack scenarios against scale-free computer network is that the literature indicates that the Internet's router connectivity topology behaves like a scale-free network. Network topologies and their relevance to this research will be discussed later in this chapter.

Specifically, this research studied changes in the network's degree distribution state and its effects on information-theoretic measures of network connectivity stability. Scale-free network theory and the foundations for the network stability measures will be discussed later in this chapter. This research assumes the attacker has sufficient knowledge to implement a denial-of-service attack against the Internet's most connected routers. This research did not consider system privileges, trust model ramifications, or attacker motivations.

3) *Internet Router Infrastructure Attacks.*

Denial-of-service attacks against the Internet's infrastructure focus on the malicious creation of message traffic congestion between routers. They also maliciously manipulate the algorithms that are used to determine efficient message paths. These conditions create communications havoc with the network's normal router traffic mechanisms, and this eventually leads to severe service degradation (Chakrabarti 2002; Mizrak et al. 2006). There are 3 types of infrastructure denial-of-service attacks (Chakrabarti 2002): 1) router table "poisoning," 2) router IP packet mistreatment, and 3) Domain Name System (DNS) hacking.

A denial-of-service attack that targets specific router regions might employ packet mistreatment and router table "poisoning" techniques to maliciously disrupt normal communications. Router tables are stored on each router and are used to identify potential paths between routers. They are essential to efficient path identification. Routers with maliciously manipulated router tables broadcast false message routing information. This leads to sub-optimal message routing and increased traffic congestion. Routing information and user data are transmitted throughout the Internet in containers known as IP packets. These packets secure the data's integrity and confidentiality. Packet mistreatment attacks

maliciously manipulate Internet packets. This technique can be used to disrupt normal traffic patterns.

The Internet's efficient communication is dependent upon the availability of 13 Domain Name System (DNS) root servers located around the world. These 13 DNS root servers form the Internet's backbone infrastructure. The Internet can sustain limited damage concurrently to a few of these root servers without experiencing major service degradation (Peng, Leckie, and Ramamohanarao 2007b). However, an attack that strategically cripples the DNS infrastructure might severely limit global Internet communication (Cheung 2006). In a DNS hacking attack, false entries are injected into the Domain Name System (DNS). This leads to counterfeit IP address translation that compromises the integrity of the Internet's web site authentication mechanisms. Attackers use domain hijacking techniques along with DNS attacks to create bogus web sites that masquerade as legitimate. They funnel large volumes of users to these bogus web sites. Placement of these sites in the same router region could potentially overwhelm normal router communication mechanisms.

One technique employed by an attacker to obfuscate the attack's source is IP spoofing (Daniels 2002; Daniels and Spafford 2000; Tang and Daniels 2005). This is accomplished using IP packet modifications. During a denial-of-service attack, attackers inject falsified return address information into all IP packets used in the attack. Large volumes of these packets are funneled to the target destination. Normally after initial communications are established, the destination host returns a confirming message back to the source. However, when IP spoofing techniques are used, these confirmation messages are invalid and the system generates large volumes of "host unreachable" messages. These "host unreachable" messages and other residual router messages generated during an attack are called

"backscatter" (Peng, Leckie, and Ramamohanarao 2007a; Shannon et al. 2006). The effect of a denial-of-service attack is magnified by the large volumes of "backscatter" messages sent over the Internet's router infrastructure. "Backscatter" will cause extreme message traffic congestion over the router infrastructure. The infrastructure may become an indirect victim of a denial-of-service attack originally targeted against an enterprise network (Douligeris and Mitrokotsa 2004; Haggerty, Shi, and Merabti 2005; Lee et al. 2008; Paxson 2001; Peng, Leckie, and Ramamohanarao 2007a; Shannon et al. 2006; Tsunoda et al. 2008).

B. Complex Networks

Complex networks have been discovered among biological, social, and technological communication systems (Albert and Barabasi 2002; Jamakovic, Uhlig, and Theisler 2007; Newman 2003). Scale-free networks are complex networks with heterogeneous node connectivity and power law degree distribution. The World Wide Web (Albert, Jeong, and Barabasi 1999; Ravi et al. 2000), metabolic pathways (Jeong et al. 2000), and many social networks (Aiello, Chung, and Lu 2000; Albert and Barabasi 2000; Barabasi and Albert 1999; Redner 1998) have been cited as having scale-free characteristic behaviors. Researchers have shown that the Internet is a scale-free network (Barabasi and Albert 2002; Boccaletti et al. 2006; Donnet and Friedman 2007; Hamed et al. 2008; Michalis, Petros, and Christos 1999; Newman 2003; Siganos et al. 2003; Yook, Jeong, and Barabasi 2002). The remainder of this section will cover complex network theory relevant to this research.

1) Network Connectivity.

A node's degree is the number of communication links established with adjacent nodes called neighbor nodes. It is a fundamental component used to study network

connectivity. Two approaches dominate the study of complex network connectivity. Prior to 2000, it was thought that large-scale complex networks were connected randomly. The early approach of Erdos and Renyi (ER); (1960) hypothesizes that complex network connectivity is governed by an exponential degree distribution. The ER model of network connectivity argued that each network node has an equal chance of forming a link with another node regardless of its degree (Erdos and Renyi 1960).

Since 2000, the "Theory of Evolving Networks" has been the prevailing conceptualization. This theory was originally postulated by Barabasi and Albert (BA) and has been supported by strong empirical evidence (Barabasi and Albert 2002; Boccaletti et al. 2006; Dorogovtsev and Mendes 2002). They hypothesized that the connectivity of complex networks is probabilistic and governed by power law degree distribution (Alderson et al. 2005; Barabasi and Albert 2002; Boccaletti et al. 2006; Costa et al. 2007; Dekker and Colbert 2008; Donnet and Friedman 2007; Michalis, Petros, and Christos 1999; Saffre et al. 2004). Barabasi and Albert (2002) argue that the likelihood of one node establishing a new link with a neighbor node is proportional to the degree of the neighbor node. Neighbor nodes with a higher degree will be more likely to establish new links. This mechanism is known as preferential attachment. Networks that exhibit power law degree distribution and follow preferential attachment mechanisms are known as scale-free networks (Alderson et al. 2005; Barabasi and Albert 2002; Costa et al. 2007; Newman 2003). Empirical studies have shown that the Internet's router connectivity topology exhibits scale-free network behaviors (Albert, Jeong, and Barabasi 2000; Barabasi and Albert 2002; Crucitti et al. 2003b; Motter et al. 2006; Newman 2003; Siganos et al. 2003). A scale-free network is defined as a complex network that follows power law degree distribution regardless of network size (Barabasi and

Albert 2002; Boccaletti et al. 2006; Costa et al. 2007; Michalis, Petros, and Christos 1999; Motter et al. 2006). Complex networks will be discussed in the next section. Power law networks have a large number of nodes with a few links and a few nodes with many links. In Figure 4 of Michalis, Petros, and Christos (1999), there is a depiction of power law degree distribution using live Internet data collected from a network of 3530 nodes with 6432 links in April 1998.

The power law degree distribution exhibited by the Internet's router infrastructure is attributed to its preferential attachment mechanisms. Preferential attachment has been studied extensively in the literature (Barabasi and Albert 2002; Boccaletti et al. 2006; Dorogovtsev and Mendes 2002; Motter et al. 2006; Newman 2003; Qin et al. 2008; Saffre et al. 2004; Sun et al. 2007; Wang et al. 2009; Zhang et al. 2008). Empirical studies have shown that the Internet's router communication mechanisms behave by preferential attachment rules (Barabasi, Ravasz, and Vicsek 2001; Saffre et al. 2004). In Figure 1 of Jeong, Neda, and Barabasi (2003), there is a depiction of preferential attachment probabilities in live Internet data collected from a network of 12,400 nodes with 13,445 links in 2000. Preferential attachment behavior leads to a greater probability that most nodes in the network will have a relatively few links that are connected to a small number of highly connected nodes.

As a result of preferential attachment, scale-free networks exhibit heterogeneous connectivity (Barabasi and Albert 2002). This connectivity topology has a few highly connected nodes that follow a power law degree distribution. The heterogeneous nature of the Internet's inter-nodal links is an essential characteristic of its robust and stable communications (Barabasi, Albert, and Jeong 2000; Boccaletti et al. 2006; Costa et al. 2007;

Criado et al. 2006; Crucitti et al. 2003a; Dekker and Colbert 2008; Demetrius and Manke 2005; Dorogovtsev and Mendes 2002; Hu and Wang 2008; Motter et al. 2006; Sanchirico and Fiorentino 2008; Wang et al. 2006; Wang et al. 2009; Zhang et al. 2008). The heterogeneity of a network can be quantified (1) directly using a heterogeneity index (Hu and Wang 2008) and (2) indirectly through measures such as its entropy (Demetrius and Manke 2005; Gudkov and Montealegre 2008; Wang et al. 2006), mutual information transfer (Leung and Chau 2007; Newman 2002; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004), and the network connectivity parameter (Cohen 2001). This research studies network connectivity using the Internet's infrastructure's heterogeneity as a measure of overall stability. Heterogeneity was measured through changes in the attacked network's mutual information transfer and the network connectivity parameter. Entropy was used to validate that the changes in the simulation reflected changes in heterogeneity.

2) Complex Networks and Emergence.

Researchers have shown that complex networks are governed by hidden mechanisms (Boschetti et al. 2005; Cassey 2004; Crutchfield 1994; Rosen 1985). They showed that these hidden mechanisms control macro-level network behaviors (such as changes in the degree distribution) through many small micro-level rules of behavior. The interactions of these mechanisms result in the emergence of patterns. These patterns are referred to as emergent behaviors. Emergence has been defined as "a process that leads to the appearance of structure not directly described by the defining constraints and instantaneous forces that control a system…" (Crutchfield 1994). Emergence can only be detected indirectly through subtle changes in a system.

These subtle random changes are often referred to as system "noise" (Ale and Kub 2003; Boschetti et al. 2005; Crutchfield 1994; Lazaroff and Snowden 2006; Rosen 1985). Changes in the system "noise" and connectivity patterns generated by complex network communications under attack can be observed through changes in its statistical mechanics (Barabasi and Albert 2002; Boccaletti et al. 2006; Newman 2003). Specifically, this research applied changes in degree distribution mechanisms under attack to study emergent network connectivity patterns by distinguishing between random noise and systemic connectivity behaviors. This research studied these systemic variations in scale-free network connectivity during simulated denial-of-service attacks. The results of this examination may lead to a feasible technique to indirectly detect emergent behaviors during a cyber attack.

C. Attack Vulnerabilities

Scale-free computer networks (such as the Internet) are extremely robust to routine random errors yet are vulnerable to targeted attacks (Albert, Jeong, and Barabasi 2000; Crucitti et al. 2004; Sun et al. 2007). To study this behavior, simulations differentiate between cyber attacks and routine router errors using selective node (router) removal techniques (Albert, Jeong, and Barabasi 2000; Crucitti et al. 2004; Crucitti et al. 2003b; Holme et al. 2002b; Latora and Marchiori 2004b; Salla 2005; Sun et al. 2007). This section presents the foundations for the attack simulation methodology used in this research.

1) Error and Attack Tolerance.

Error and attack tolerance of scale-free networks has been studied extensively. (Dall'Asta et al. 2006; Gallos et al. 2006; Holme et al. 2002a; Lai, Motter, and Nishikawa 2004; Latora and Marchiori 2005; Moore, Ellison, and Linger 2001; Paxson 2001; Wang et al. 2008; Wu et al. 2007; Yu, Chen, and Zhou 2008; Zhao et al. 2005). Using the techniques found in these studies, this research simulated denial-of-service attacks. There are two relevant attack simulation techniques depicted in this literature, node removal and link removal (Dall'Asta et al. 2006; Gallos et al. 2006; Holme et al. 2002a; Lai, Motter, and Nishikawa 2004; Latora and Marchiori 2005; Moore, Ellison, and Linger 2001; Paxson 2001; Wang et al. 2008; Wu et al. 2007; Yu, Chen, and Zhou 2008; Zhao et al. 2005). A link's weight measures the relative volume of message traffic flowing over it. In a link removal approach, attacks are simulated through the removal of links based on their weight. (Dall'Asta et al. 2006; Furuya and Yakubo 2008; Huang and Li 2007; Leung and Chau 2007; Lopez 2007; Macdonald, Almaas, and Barabasi 2005; Wang and Rong 2009a, 2009b). In a node removal approach, nodes are removed in the attack simulation based on their degree (Dall'Asta et al. 2006; Gallos et al. 2006; Holme et al. 2002a; Lai, Motter, and Nishikawa 2004; Latora and Marchiori 2005; Moore, Ellison, and Linger 2001; Paxson 2001; Wang et al. 2008; Wu et al. 2007; Yu, Chen, and Zhou 2008; Zhao et al. 2005). Since the literature has indicated that node-removal techniques are commonly used to emulate denial-of-service attacks, this research applied selective node removal techniques to simulate denial-of-service attacks (Albert, Jeong, and Barabasi 2000; Crucitti et al. 2004; Holme et al. 2002a; Salla 2005; Sun et al. 2007; Wang et al. 2008).

The Internet's robust connectivity mechanisms seamlessly re-route traffic around disturbances without affecting normal efficient communication (Cicic 2008; Labovitz et al. 2001; Markopoulou et al. 2008; Wang et al. 2006). Random router errors occur frequently over the Internet (Cicic 2008; Labovitz et al. 2001; Markopoulou et al. 2008; Wang et al. 2006). The Internet was designed for efficient and robust communication in an error-prone environment:

> "From the early days of the Internet, the ability to tolerate loss of network components has been one of the key goals in its design. Internet routers include mechanisms that detect connectivity failures and topological changes, and convey this information to their routing protocols. The protocols distribute the change information network-wide, and the network gradually adopts the new routing paths and converges to the new stable routing state" (Cicic 2008).

Therefore, effective Internet attack detection studies must be able to distinguish between attack anomalies and routine router failures (Douligeris and Mitrokotsa 2004; Jung-Ying et al. 2008; Mizrak et al. 2006; Olalekan 2008; Shannon et al. 2006).

Figures 2.1 and 2.2 illustrate the effects on network connectivity in attack simulations that use a node removal strategy. Figure 2.1 was taken from Crucitti et al. (2003b) and Figure 2.2 was taken from Albert, Jeong, and Barabasi (2000) (annotations were added to these figures). Both figures depict attack simulations against scale-free networks. Figure 2.1 presents network efficiency measurements in 20 simulation runs, 10 using an exponential (ER) network and 10 using a scale-free (SF) network (Crucitti et al. 2003b).

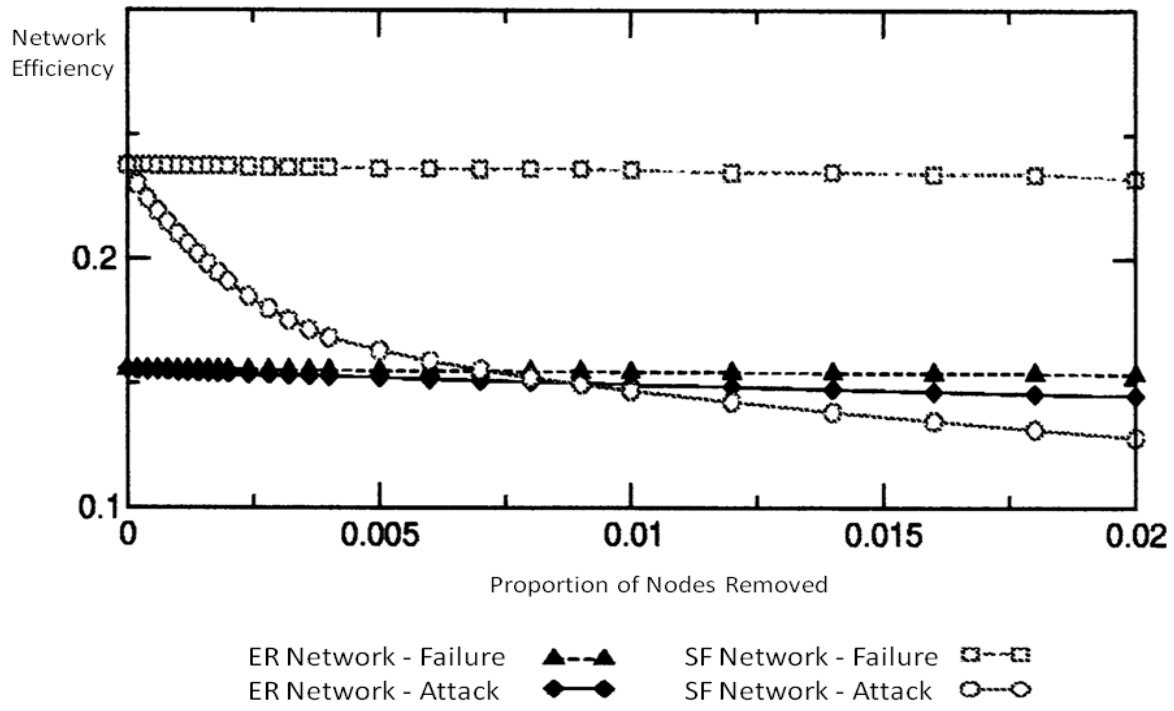


Figure 2.1. Error and attack tolerance study taken from Crucitti et al. (2003b)

Each simulation started with a virtual network of 5,000 nodes and 10,000 links (Crucitti et al. 2003b). Normal router errors are simulated as random removal of nodes regardless of their degree. Attacks are emulated as selective removal of nodes with the highest degrees. Starting at the node with the highest degree, the attack removal process removes nodes, one at a time, from the network simulation in decreasing node degree order. The proportion of the total number of nodes removed is plotted against changes in network efficiency measures. It shows that the ER networks are not sensitive to small node removal attacks. However, the scale-free network simulations show that selective node removal attacks and routine router error can be distinguished from normal router failures.

Figure 2.2 results were derived from experiments on an Internet router network of 6,029 nodes and 12,200 links. The node removal strategy used in Figure 2.2 was similar to the strategy used Figure 2.1. Figure 2.2 evaluates network connectivity using changes in network diameter during attack experiments over the Internet (Albert, Jeong, and Barabasi 2000). In Figure 2.2, circles represent attacks, and squares represent normal random failures. The network's diameter is the average shortest path for all possible paths in a network (Albert, Jeong, and Barabasi 2000). It is a common method for measuring a network's efficiency. The efficiency of the network decreases as the average shortest path increases. As presented in Figure 2.2, attacks can be differentiated from routine router error over the Internet's router infrastructure.

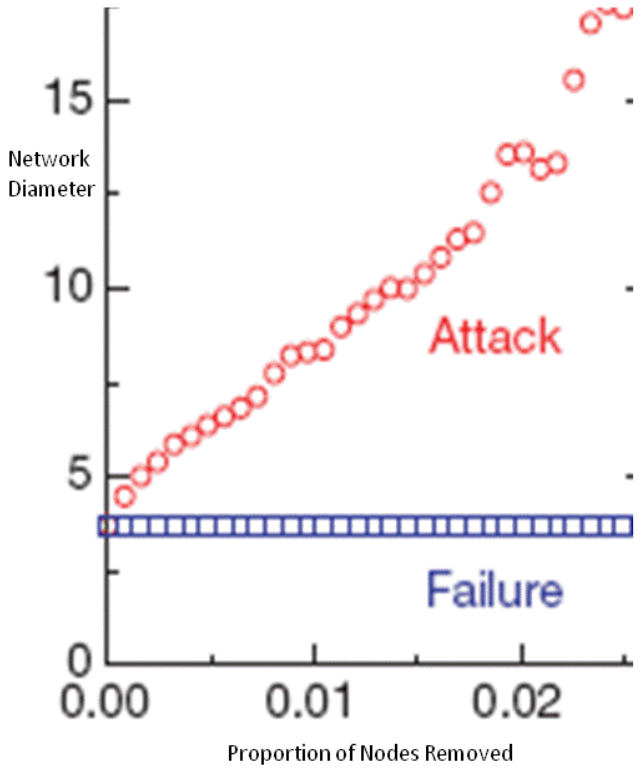


Figure 2.2. Error and attack tolerance over the Internet (scale-free), taken from Albert, Jeong, and Barabasi (2000)

For the random error simulations representing routine failures, Figure 2.1 (both ER and SF) and Figure 2.2 indicate that as the number of nodes removed increases the network efficiency remains in a stable steady state. Under attack, it has been shown that scale-free networks behave differently than ER networks. When the attacked nodes are removed in a scale-free network, both network efficiency measures shown in Figure 2.1 (for the SF network) and Figure 2.2 decrease dramatically. In scale-free networks, after as little as 2% of the nodes are removed, the network's efficiency was reduced by more than 50%. Both figures depict that denial-of-service attacks against scale-free networks are clearly distinguishable from normal router errors.

2) Cascaded Failures.

Cascaded failures induced by removing a small number of highly connected nodes from a scale-free network have been found to occur in complex technological, social, biological, and economic networks (Crucitti, Latora, and Marchiori 2004). For example, the electrical grid of the United States has been shown to be a scale-free network (Crucitti, Latora, and Marchiori 2004; Dobson et al. 2007). Two major electrical outages in 1996 and 2003 were the result of a single relatively small outage event that cascaded the electrical failures (Crucitti, Latora, and Marchiori 2004; Dobson et al. 2007). As previously discussed, scale-free networks subjected to node removal attacks suffer early and rapid degradation in network efficiency. Research studies show that an avalanche of cascading node failures is responsible for this rapid decline in network efficiency (Cohen 2000, 2001; Crucitti, Latora, and Marchiori 2004; Dobson et al. 2007; Huang, Lai, and Chen 2008; Huang and Li 2007; Lai, Motter, and Nishikawa 2004; Moreno, Gomez, and Pacheco 2002; Motter 2004; Motter and Lai 2002; Wang and Rong 2009a,2009b; Wu and Fang 2008; Xu and Wang 2005).

The cascading avalanche mechanism is well supported in the literature (Cohen 2000, 2001; Gallos 2005; Lai, Motter, and Nishikawa 2004; Lopez 2007; Motter and Lai 2002; Wang et al. 2008). It has also been shown that increasing the link capacity of the network's nodes slows down the avalanche's degradation of network performance (Crucitti, Latora, and Marchiori 2004; Lai, Motter, and Nishikawa 2004; Motter and Lai 2002). The research of this dissertation studied simulated DoS attacks by varying the speed and severity of the cascading avalanche of node failures using targeted denial-of-service attacks. Crucitti et al. (2004) describe the mechanism: "Cascading failures take place on the Internet, where traffic is rerouted to bypass malfunctioning routers, eventually leading to an avalanche of overloads on other routers that are not equipped to handle extra traffic. The redistribution of the traffic can result in a congestion regime with a large drop in the performance".

D. Network Connectivity Stability

As previously discussed in this chapter, network communication becomes irreversibly degraded when a "halting condition" is encountered (Stephenson and Prueitt 2005). During a cascaded failure, it has been observed that at some critical threshold ("halting condition") the network fragmentation rate dramatically increases and the network's connectivity is totally destroyed (Barabasi and Albert 2002; Cohen 2001; Gallos 2005; Huang and Li 2007; Moreno, Gomez, and Pacheco 2002; Motter 2004; Motter and Lai 2002; Wu et al. 2007; Wu and Fang 2008). Percolation theory studies an attacked network's stability as it fragments into increasingly smaller and isolated clusters (Cohen 2000, 2001; Dorogovtsev, Goltsev, and Mendes 2008; Gallos 2005; Lopez 2007; Pietsch 2006; Zhao et al. 2005). Percolation theory establishes the groundwork for further study of a network's physical statistics

undergoing a rapid decline in network stability. The research methods developed in this dissertation applied the basic tenets of percolation theory to study network stability.

Researchers have found a direct relationship between the network's stability and its heterogeneity (Wang et al. 2006). During a node removal attack, the network's connectivity loses its heterogeneous nature, and information flow between nodes is restricted (Albert, Jeong, and Barabasi 2000; Cohen 2001; Crucitti et al. 2004; Demetrius and Manke 2005; Salla 2005; Sun et al. 2007; Wang et al. 2006). This degradation of network stability can be quantified using (1) mutual information transfer (Lerner 2004; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004; Srivastav, Ray, and Gupta 2009), (2) global network efficiency (Criado et al. 2006; Crucitti et al. 2003a; Latora and Marchiori 2001, 2004a), (3) entropy (Gudkov and Montealegre 2008; Wang et al. 2006), (4) network connectivity parameter, average node degree, and neighbor node degree (Cohen 2000, 2001; Dorogovtsev and Mendes 2002; Gallos and Argyrakis 2007), (5) joint entropy (Boschetti et al. 2005; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004; Wang et al. 2006), (6) heterogeneity index (Hu and Wang 2008), and (7) assortativeness (Newman 2002; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). This research used mutual information transfer and the network connectivity parameter to monitor the network's fragmentation and loss of stability during the attack simulations. These two measures were used to determine the extent of network stability degradation. This provided a way to observe network stability degradation as a function of the heterogeneity changes catalyzed by the simulated DoS attacks. The remainder of this section describes these two measures of network stability.

1) Mutual Information Transfer.

Previous discourse indicates that communications over a complex network generate system "noise." Information theory studies the quantification of communications in a "noisy" environment (Piraveenan, Prokopenko, and Zomaya 2009). The uncertainty of Internet communications during a cyber attack can be studied using information theory (Gudkov and Montealegre 2008; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). Joint entropy is a measure of the average uncertainty of a network's inter-nodal linking mechanisms (Boschetti et al. 2005; Mahadevan et al. 2005). It has been shown to be a reliable measure of a network's link heterogeneity (Boschetti et al. 2005; Mahadevan et al. 2005). Mutual information transfer is derived from the network's joint entropy. It is a measure of the average uncertainty of information flow between 2 nodes (Boschetti et al. 2005; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004; Wang et al. 2006). Studies have shown that the mutual information transfer is a relatively unbiased statistic that accurately portrays the global inter-nodal information flow of a scale-free network (Boschetti et al. 2005; Newman 2002). Mutual information transfer quantifies the affects of network fragmentation during node removal attack (Boschetti et al. 2005; Demetrius and Manke 2005; Lopez 2007; Piraveenan, Prokopenko, and Zomaya 2008; Srivastav, Ray, and Gupta 2009).

2) Network Connectivity Parameter and Average Node Degree.

The network connectivity parameter (NCP) measures the physical extent of network fragmentation (Cohen 2000, 2001; Dorogovtsev and Mendes 2002; Gallos and Argyrakis 2007). While mutual information transfer measures the stability of the network's information flow, the network connectivity parameter measures the extent of the network's physical

fragmentation. As the parameter value approaches 2, the network's heterogeneity decreases. Generally, if the parameter value is less than 2, the network is totally fragmented and is no longer considered heterogeneous (Gallos and Argyrakis 2007). In addition to the NCP, network connectivity stability is measured through changes in the network's average node degree (Costa et al. 2007; Estrada, Higham, and Hatano 2009; Mahadevan et al. 2005; Wang et al. 2006). Both the average node degree and the NCP are used in this research to monitor network stability during the attack simulations.

E. Topology Based Protection

This section will present a few promising protection schemes that utilize topology knowledge of scale-free network connectivity. One study divided a large heterogeneous network into small clusters of nodes (Huang, Lai, and Chen 2008). Huang et al. (Huang, Lai, and Chen 2008) examined a protection scheme based on inter-cluster shortest paths and the "bridge" nodes between clusters. Huang et al. (2008) presented evidence that their techniques might be able to halt a cascaded avalanche of node failures. Another method examined changes to the average shortest path while individually removing select nodes from the network (Latora and Marchiori 2005). The removed nodes that caused the greatest damage to the shortest path were considered candidates for protection.

Wang and Rong (2008) studied the effects of changes to a node's link capacity to estimate a breakdown probability for each network node (Wang and Rong 2009a). They proposed protecting the nodes with the highest probability of breakdown. A low cost and counterintuitive finding in two different studies suggests that protecting the nodes with the

lowest degrees may reduce the cascaded degradation effects (Motter 2004; Wang and Rong 2008; Wang et al. 2008). Another study determined a critical threshold for the network connectivity parameter (Gallos and Argyrakis 2007). Gallos and Argyrakis (2007) proposed protecting nodes below this threshold. One research examination found that using a "reverse percolation" process to restore the power law properties of a network under attack was proposed as a reactive mechanism to maintain network stability (Rezaei et al. 2007). Rezaei et al. (2007) proposed that the network's connectivity robustness can be maintained during an attack by monitoring the degree distribution and adding links to compensate for nodes removed during the attack. Sekiyama and Araki (2007) investigate a similar network recovery approach. They examined manipulation of the network's topology during an attack to regenerate the network's connectivity infrastructure (Sekiyama and Araki 2007). It has also been found that hiding network topology information can be an effective protection technique (Gallos 2005; Gallos et al. 2007).

F. Theoretical Rationale Summary

This section summarizes the theoretical rationale for this study for the identification of *attack markers* has been presented in this chapter. As shown in the literature, assortativity is a summary measure of a network's link diversity (Newman 2002). The assortativity coefficient for complex networks has been found in the range of $-1 \leq r \leq 1$. A perfectly disassortative network ($r = -1$) signifies that all nodes are connected to nodes of a different degree. A perfectly assortative network ($r = 1$) indicates that all node connections are between 2 nodes of the same degree. Most networks are either predominant disassortative ($r<0$) or assortative ($r>0$) (Newman 2002). The literature has shown that technological scale-

free networks, such as the Internet's router infrastructure, tend to exhibit primarily disassortative behavior (Leung and Chau 2007; Newman 2002; Piraveenan, Prokopenko, and Zomaya 2008, 2009).

The literature indicates that as the assortativity approaches zero and becomes increasingly more positive, the network's heterogeneity and robustness decreases (Newman 2002; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). As the heterogeneous behavior diminishes, the network's linking mechanism tends to become more like an exponential (ER) network (Newman 2002; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). Previous discussion has shown that preferential attachment is a probabilistic mechanism that favors link establishment with highly connected nodes. Preferential attachment mechanisms are essential for the communication robustness found in scale-free networks. It follows that as the assortativity of a scale-free network approaches zero, its characteristic scale-free robustness also diminishes.

This research did not directly compute assortativity. However, researchers indicate that assortativity and mutual information transfer exhibit an inverse relationship (Newman 2002; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). They show that as a result of this relationship, assortativity can be observed indirectly through the mutual information transfer. This in turn can be used to study the simulated attack and the network's loss in heterogeneity. Entropy was also used to observe the simulated network's heterogeneity loss.

The theoretical rationale presented in this chapter has shown background support for this research investigation. This chapter has shown (1) the Theory of Cyber Attack Mechanics that argues that during cyber attacks, emergent patterns of connectivity can be

indirectly identified through *attack markers*; (2) studies of the Internet's router infrastructure vulnerabilities to targeted DoS attacks; (3) scale-free network theory; (4) error and attack studies against scale-free networks; (5) cascading node failures as a result of targeted DoS attacks; (6) mutual information transfer and the network connectivity parameter as a means to measure the simulated attack connectivity stability and heterogeneity; and (7) recent studies that seek to protect scale-free networks using knowledge of the physical characteristics of the network. In Chapter III, the formalisms used to develop the attack model and simulation developed in this research will be presented.

CHAPTER III. FORMAL DEFINITIONS

This chapter introduces the rudimentary formalisms developed to design the cyber attack model and simulation for this research. Chapter IV will cover the design derived from these formalisms. All simulation processes will also be discussed in Chapter V.

A. Node State Data Structures

The formal definitions developed in this chapter were derived from the node state transitions summarized in Figure 3.1 and Table 3.1. The node state diagram shown in Figure 3.1 is described by the state transition matrix in Table 3.1. As shown in Table 3.1, the 6-tuple (<AC, NR, TO, NL, OL, AN>) depicts a node's state as it is processed by the simulation. Each term and associated values in the 6-tuple are defined in Table 3.1. As depicted in Table 3.1, if the transition condition evaluates as true then the corresponding tuple position is recorded as a "1" value, otherwise the value is "0". Active nodes are represented as states NS_0, NS_1, NS_2 with a 6-tuple value of "1" in position "AC"; otherwise the value is "0". If all positions are "0" then the node state is a null-link. In Figure 3.1 each state is represented by a circle and the transition between states is depicted by a line between the circles. The direction of the transition is indicated by the arrows on the lines. Inside each node state circle of Figure 3.1 the corresponding 6-tuple values are shown. As depicted in Table 3.1, the value at the intersection of a row (transition conditional) and column (current state) indicates a new possible state. For example, a node in current state NS_0 will transition to state NS_1, if and only if NP-Release is true. All node state changes are triggered by the attack simulation clock ticks represented as $t \in \{0,1,2,3, \ldots\}$. The remainder of this chapter will formally define each node state and its corresponding data structures.

Table 3.1. Node state transitions for Figure 3.1

Condition	*Current Node State* → *Description*	*6-Tuple*	NS_0	NS_1	NS_2	NS_3	NS_4	NS_5
NP-RELEASE = T	Node is a member of a released node-pair	NR = 1	NS_1					
NP-RELEASE = F	Node is not a member of a released node-pair	NR = 0	NS_0					
TEMP-ORPHAN = T	Node is a temporary orphan	TO = 1		NS_2				
TEMP-ORPHAN = F	Node is not a temporary orphan	TO = 0		NS_0				
NEW-LINK = T	Node has an available neighbor	NL = 1			NS_5			
NEW-LINK = F	Node does not have an available neighbor	NL = 0			NS_3			
OVERLOAD = T	Node link load has exceeded its link capacity	OL = 1						NS_4
OVERLOAD = F	Node link load has not exceeded its link capacity	OL = 0						NS_0
ATTACKED = T	Node is a critical node that has been attacked	AN = 1	NS_5					
ATTACKED = F	Node is not a critical node that has been attacked	AN = 0	NS_0					

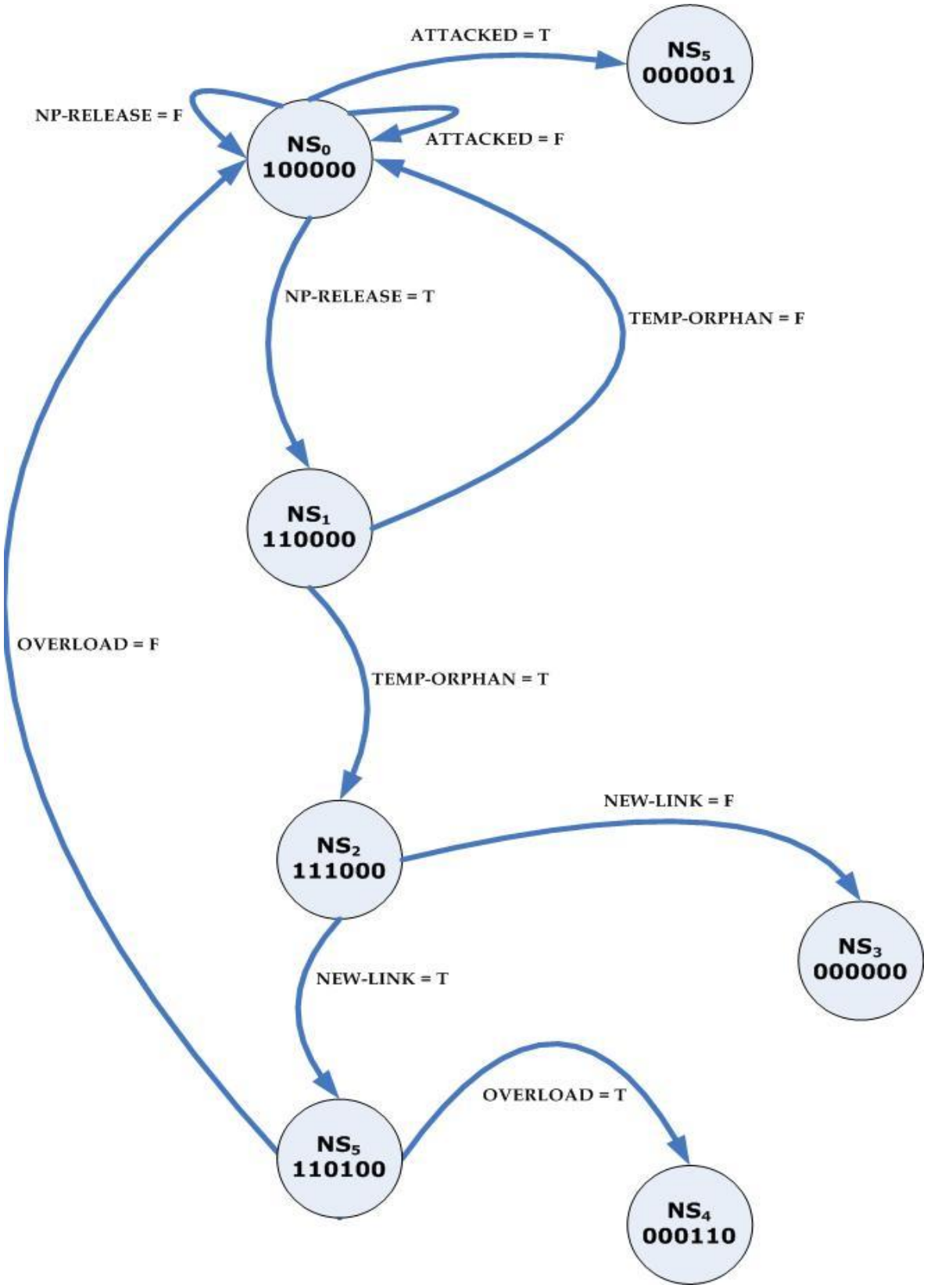


Figure 3.1. Node state diagram for the CPN simulations

1) Set of All Active Nodes.

At time t, a node is in the active state if it has not been removed from the simulation during an attack. Pre-attack active nodes are all nodes present at time $t = 0$.

$$N(t)^{Active} = \{n_i^{Active}(t) \mid i \in \{0,1,2,3,\dots,(M(t)^{Active} - 1)\}\}; \quad (1)$$

where $M(t)^{Active}$ is the number of active nodes at time t.

2) Set of All Node-Pairs.

At time t, the set of all node-pairs is the set of ordered pairs of two adjacent active nodes, (n_t^i, n_t^j), such that router (node) $n_j^{Active}(t)$ is a router table entry in router (node) $n_i^{Active}(t)$. A simulated communication attempt does not occur until the node-pair has been randomly evaluated in the simulation stream. Each node-pair was released into the evaluation using a linear congruential algorithm (Sedgewick 1983) and further randomized by the CPN application engine. The simulated communication attempt process will be described in the next chapter. A node-pair is defined as:

$$p_y(t)^{Active} = \{(n_i(t), n_j(t)) \mid n_i(t) \in N(t)^{Active},\ n_j(t) \in N(t)^{Active},\ i \neq j,\ n_j(t) \in A_i(t))\}; \quad (2)$$

$$P(t)^{Active} = \{p_y^{Active}(t) \mid y \in \{0,1,2,3,\dots,(Q(t)^{Active} - 1)\}\}; \quad (3)$$

where

$P(t)^{Active}$ is the set of active node-pairs, $(n_i(t), n_j(t))$;

$Q(t)^{Active}$ is the number of active node-pairs at time t;

$A_i(t)$ is the set of all neighbor nodes of node i.

The set of all neighbor (adjacent) nodes of node i is:

$$A_i(t) = \{a_k(t) \mid k \in \{0,1,2,3,\dots,(K_i(t) - 1)\}, a_k(t) \in N(t)^{Active}\}; \quad (4)$$

Where $K_i(t)$ is the number of neighbor nodes of active node i, $n_i(t)$, and referred to as the degree of node i, that is the number of links for node i.

3) Attack Class.

During the simulation critical nodes are randomly attacked. For each simulation run, an attack class was defined as the fraction of all nodes selected to be critical nodes. Critical nodes are nodes that will be eventually attacked (removed) during the simulation. For each simulation run there exists one and only one attack class. The attack classes used in this research were supported by previous error and attack studies (Albert, Jeong, and Barabasi 2000; Cohen 2001; Crucitti et al. 2004; Guillaume, Latapy, and Magnien 2005; Motter and Lai 2002; Salla 2005). The set of all attack classes is an ordered set such that:

$$C(t=0) = \{c_z | c \in \mathbb{R}, z \in \{(0,1,2,3,\dots\ (z_{max}\ -1)), z_{max}\ = 10\}\}; \quad (5)$$

where

z_{max} = maximum number of attack classes used in this research.

$c_0 = 0.05;$

$c_z = (c_{z-1} + c_0);$

$\forall z > 0, c_z > c_{z-1}$

4) Critical and Attacked Nodes.

Critical nodes for each simulation run are selected based on their degree. The set of all node degrees ($K(t=0)^{Ordered}$) is an ordered set of all unique active node degrees ($k_i(t=0)$). It is sorted from highest to lowest node degrees. Nodes corresponding to the top $c_z * 100$ percent are specified critical nodes ($n_w(t=0)^{Critical}$). The set of all critical nodes for attack class c_z is:

$$N_w(t=0)^{Critical\ (c_z)} = \left\{ n_w(t) \middle|\ w \in \{0,1,2,3,\dots,(W_{c_z}(t=0)-1)\} \right\} \quad (6)$$

Where $W_{c_z}(t=0)$ is the number of critical nodes.

From the set of critical nodes, attacked nodes were randomly selected at random simulation times. The set of all unordered node degrees is defined:

$$K(t) = \left\{K_i(t) \middle| \, i \in \{0,1,2,3,\dots,(M(t=0)^{Active} - 1)\}\right\} \quad (7)$$

The ordered set of all unique node degrees in the network at $t = 0$, sorted from high to low degree is:

$$K(t=0)^{Ordered} = \left\{K_i\,(t=0) \middle| \begin{matrix} i \in \{0,1,2,3,\dots\,(M(t)^{Active} - 1)\}, \\ (k_i > k_{i+1}) \end{matrix}\right\} \quad (8)$$

It follows that for all unique node degrees of attack class c_z in one simulated network:

1. The number of critical nodes is:

$$W_{c_z}(t=0) = (c_z * M(t=0)^{Active}) \quad (9)$$

2. The set of all of non-critical nodes is:

$$N_g(t=0)^{NC(c_z)} = \left\{n_g(t) \middle| \, g \in \{0,1,2,3,\dots,(U_{c_z}(t=0) - 1)\}\right\} \quad (10)$$

3. The number of all of non-critical nodes is:

$$U_{c_z}(t=0) = M(t=0)^{Active} - W_{c_z}(t=0) \quad (11)$$

4. The set of all attacked nodes at time t:

$$N(t)^A = \left\{n_a(t)^A \middle| \, a \in \{0,1,2,3,\dots,(M(t=0)^A - 1), n_a(t)^A \in N_w(t=0)^{Critical\ (c_z)}\right\} \quad (12)$$

where $M(t=0)^A$ is the number of nodes randomly designated for attack.

5) Set of all Temporary Orphan Nodes.

A communications attempt is emulated by randomly releasing node-pairs into the simulation stream for evaluation. During the node-pair evaluation process, one node in the node-pair may be classified as a temporary orphan node. This indicates that the communication attempt between the two nodes in the node-pair has failed. The criterion for

orphan node determination is defined below. The process designed to simulate communication attempts will be discussed in Chapter IV. Once a node has become a temporary orphan, the simulation will determine whether it can establish a new incremental link with an existing neighbor node. If no other active neighbor nodes exist, then the temporary orphan will become a permanent orphan. The permanent orphan has lost all ability to communicate with other nodes and it will be added to the set of removed nodes. Since no information flows to or from a permanent orphan, they do not contribute to the network-wide information transfer value. Given the node-pair, $(n_i(t), n_j(t))$, a temporary orphan, $n_r(t)^{TO}$ exists as a member of the set of all temporary orphan nodes:

$$if\ (n_i(t) \in N(t)^R) \wedge (n_j(t) \notin N(t)^R)) \text{ then } n_j(t)\ \in N(t)^{TO} \quad (13)$$

$$if\ \left(n_j(t) \in N(t)^R\right) \wedge (n_i(t) \notin N(t)^R)) \text{ then } n_i(t)\ \in N(t)^{TO} \quad (14)$$

Where $N(t)^R$ is the set of all removed node and will be defined in (22).

The set of all temporary orphans is: $N(t)^{TO} = \left\{n_r(t)^{TO} \middle| r \in \{0,1,2,3,\dots,(M(t)^{TO} - 1)\}\right\}$ (15)

Where $M(t)^{TO}$ is the number of temporary orphan nodes at time t.

6) Set of all Null-Link Orphan Nodes.

If a temporary orphan node exists, then the temporary orphan node recovery process will determine whether the orphan can establish a valid communication link with another neighbor node. If the temporary orphan node has at least one active neighbor node, then a new node-pair connection will be established using the preferential attachment mechanisms discussed in Chapter II. If an active neighbor node does exist, then a new node-pair is established and the orphan node is no longer an orphan and remains active. If there are no other active neighbor nodes, then it will be classified as a null-link orphan node. Null-link

orphans are permanent orphans. Given the existence of temporary orphan node, n_t^{TO}, then the set of null-link nodes is defined as:

$n_r(t)^{TO} \Rightarrow n_s(t)^{Null}$ if and only if there are no active neighbor nodes for the temporary orphan node $n_r(t)^{TO}$, then the set of neighbor nodes for the temporary orphan node r is:

$A_r(t) = \{\}$;

$$N(t)^{Null} = \left\{ n_s(t)^{Null} \middle| s \in \{0,1,2,3,\dots,(M(t)^{Null} - 1)\}, n_s(t)^{Null} \in N(t)^{O}, n_s(t)^{Null} \notin N(t)^{TO} \right\} \quad (16)$$

Where $M(t)^{Null}$ is the number of null-link orphan nodes at time t.

7) Set of all Overloaded Orphan Nodes.

When a temporary orphan node, $n_r(t)^{TO}$, has an active neighbor, a new incremental node-pair is established. This incremental link is denoted as $k_x(t)$ to node x, $n_x(t)$ in the new node-pair $(n_r(t)^{TO}, n_x(t))$. When the incremental node-pair $((n_t^{TO}, n_t^x))$ is established, the load on node x, $n_x(t)$, is incremented by one link. This additional load may exceed the node's link capacity. If the link capacity of node $n_x(t)$ is exceeded, then node $n_x(t)$ is transitioned to an overloaded orphan node, $n_f(t)^{OL}$.

The overloaded orphans $(n_f(t)^{OL})$ and null-link orphans $(n_s(t)^{Null})$ are unavailable for further communications and removed from the simulation. If the new incremental link results in an overloaded orphan, then $n_r(t)^{TO}$ remains a temporary orphan and continues to look for a valid link with one of its existing active neighbor nodes. When a valid link occurs, that is, the new incremental link does not result in an overloaded node, then $n_r(t)^{TO}$ is removed from the set of temporary orphans. The node transitions are described in the previously presented Table 3.1 and Figure 3.1. The overloaded node occurs as follows:

1. The pre-attack link load of node $n_x(t)$:$k_x^{Load\,(t=0)} = A_x(t=0)$ (17)

Where $A_x(t=0)$ is the number of pre-attack neighbor nodes of node x.

2. At time t, $N_r(t)^{NewLink\ (x)}$ is the set of all temporary orphan nodes that have established an incremental link to node $n_x(t)$ through the recovery process.

3. At time $t^{'}$, the additional link load due to the temporary orphan recovery process:

$$k_x^{Load\ (t^{'})} = (\ k_x^{Load\ (t)} + M_r(t)^{NewLink\ (x)}\ ); \tag{18}$$

Where $M_r(t)^{NewLink\ (x)}$ is the number of $N_r(t)^{NewLink\ (x)}$ nodes for the entire simulation that form a new incremental link with $n_x(t)$.

4. The capacity index (L) for all active nodes is dependent upon the tolerance parameter. The constant tolerance parameter (Motter 2004; Motter and Lai 2002; Wang and Rong 2009a, 2009b) used in this research was: $\delta = 0.1$:

$$L = 1.0 + \delta \tag{19}$$

5. Total link capacity for node n_t^x:

$$k_x^C = L * k_x^{Load\ (t=0)} \tag{20}$$

6. The set of overloaded orphan nodes that occur when a node's current load exceeds its link capacity:

$$N_f(t)^{OL} = \left\{ n_f(t)^{OL} \middle| f \in \{0,1,2,3,\dots,(M(t)^{OL}-1)\}, (\ k_x^{Load\ (t)} > \ k_x^C\ ) \right\}; \tag{21}$$

Where $M(t)^{OL}$ is the number of overloaded orphan nodes at time t.

8) Set of all Removed Nodes.

Null-link and overloaded orphan nodes are added to the set of removed nodes. The set of all removed nodes at time t represents all nodes that are unable to communicate with other nodes. The set of removed nodes is:

$$N(t)^R = N(t)^O + N(t)^A \tag{22}$$

$$N(t)^R = \left\{ n_f(t) \middle|\ f \in \{0,1,2,3,\dots,\left(M(t)^R - 1\right)\} \right\}; \tag{23}$$

Where $M(t)^R$ is the number of removed nodes at time t.

The set of all permanent orphan nodes is:

$$N(t)^O = N(t)^{Null} + N(t)^{OL} \quad (24)$$

Permanent orphans are unable to establish communications as a result of cascaded node failures.

B. Topology Based Node Protection

This research has shown that targeted protection of pre-attack nodes based on node-pair types is plausible. In the attack simulation, a protected node was a node that always remained active. These protected nodes were not subjected to communication breakdown during the simulated attacks. Protected nodes are defined as follows:

Given:

1. The set of protected node degrees:

   $$H_d(t)^{Protect} = \{h_d(t)^{Protect} \mid d \in \{0,1,2,3,\dots,(I_d(t)^{Protect} - 1)\}\}; \quad (25)$$

   Where $I_d(t)^{Protect}$ is the number of node degrees designated for protection.

2. The set of protected nodes:

   $$H(t)^{Protect} = \{h_p(t)^{Protect} \mid p \in \{0,1,2,3,\dots,(I(t)^{Protect} - 1)\}\}; \quad (26)$$

   Where $I(t)^{Protect}$ is the number of nodes designated for protection.

3. The set of *attack marker* node-pair types:

   $$V(t)^{AM} = \{(v_1(t), v_2(t)) \mid (v_1(t), v_2(t)) \in P(t)^{Active}\}; \quad (27)$$

4. Active node $n_i(t)$ has a node degree of $K_i(t)$ and node $n_j(t)$ has a node degree of $K_j(t)$; both nodes are members of node-pair $p_y^{Active}(t)$

It follows that at time t, if node i has a degree that is a member of the set of protected node degrees and node i is a member of an *attack marker* node-pair that was designed for protection, then node i is also protected. This was defined as:

If $(n_i(t) \in H_p(t)^{AM}) \wedge (K_i(t) \in H_d(t)^{Protect})$ then $n_i(t) \in H(t)^{Protect} \wedge (n_i(t)) \notin N(t)^R$; (28)

If node j has a degree that is a member of the set of protected node degrees and node j is a member of an *attack marker* node-pair that was designed for protection then node j is also protected. This was defined as:

If $(n_j(t) \in H_p(t)^{AM}) \wedge (K_j(t) \in H_d(t)^{Protect})$ then $n_j(t) \in H(t)^{Protect} \wedge (n_j(t)) \notin N(t)^R$; (29)

Node-pairs and active nodes were previously defined in Section A of this chapter. The selection of nodes to be protected, node-pair types, and attack markers will be discussed in Chapter IV.

C. Simulated Attack Definition

This research focused on DoS attacks. Stephenson and Prueitt define a taxonomy for DoS attacks using cyber attack primes as attack descriptors (2005). The attack tuple (DoS) described by Stephenson and Prueitt and relevant to this study was defined by their taxonomy as:

<DOS, User_Err_Slf_Protect, User_Err_Misuse_Avl_Resc, Power_Disrupt, Malicious_Code, Hack_Phys, Hack_Avl_Resc, Failure_DS_Comp, Dev_Flawed_Code, Component_Failure, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit>

For detailed explanations of each element, see Stephenson and Prueitt (2005). The remainder of this section will formally represent a DoS attack as simulated in this research.

1) Cyber Attack States.

An external stimulus (β) is any entity that seeks to induce a denial-of-service attack using the attack descriptors discussed above. An attack is defined as a network state change incurred as a result of an external stimulus applied to normal network states:

$$\beta \cdot \{ s_i \mid i \in \{0,1,2,\ldots,m\}\} \Rightarrow \{s_k^a \mid k \in \{0,1,2,\ldots,n\}\} \quad (30)$$

Given that $t' > t$ and $k' > k$ then an external stimulus (β) applied to a set of normal network states ($\{ s_i\}$) results in a set of new attacked states ($\{s_k^a\}$); $i = k = 0 \Leftrightarrow$ pre-attack network state. An attack state (s_k^a) is inferred when external stimulus is applied to a normal network state resulting in an anomalous new state: $\beta \cdot \ s_i \ \Rightarrow s_k^a$. By convention this research dissertation will represent $\beta \cdot s \ \Leftrightarrow \beta \cdot \{s\}$. The attack simulation process definitions are defined below. All terminologies below have been previously defined in this chapter.

2) Network State Transitions.

All node related terms were previously defined in Sections A and B of this chapter. This section will present the state transition details that were discovered during this study.

Transition 1: A new attack state results when an external stimulus is applied to the set of critical nodes.

$$\beta \cdot \ N_w(t=0)^{Critical\ (c_z)} \Rightarrow N(t')^A \quad (31)$$

$$N(t')^A \Rightarrow \ s_k^a \quad (32)$$

Transition 2: As a result of the attack state change in Transition 1:

1. The attacked nodes are removed from the set of active nodes.

$$s_k^a \Rightarrow N(t')^{Active} \text{; where } N(t')^{Active} = N(t)^{Active} - N(t)^{A} \quad (33)$$

2. Cascaded failures lead to overloaded nodes that are removed from the set of active nodes.

$$s_k^a \Rightarrow N(t')^{Active} \text{; where } N(t')^{Active} = N(t)^{Active} - N(t)^{OL} \quad (34)$$

3. Cascaded failures lead to null-link nodes that are removed from the set of active nodes.

$$s_k^a \Rightarrow N(t')^{Active} \text{; where } N(t')^{Active} = N(t)^{Active} - N(t)^{Null} \quad (35)$$

4. Nodes are permanently orphaned the set of active node-pairs is altered.

$$N(t')^{Active} \Rightarrow P(t')^{Active}; \quad (36)$$

$$P(t')^{Active} = P(t)^{Active} - (N(t')^{R} \subset P(t')^{Active}) \; ; \quad (37)$$

$$\left(n_i(t) \vee n_j(t)\right) \in (N(t')^{Null} + N(t')^{A} + N(t')^{OL}); \quad (38)$$

Transition 3: As a result of the network distortions depicted in Transition 2, the network's stability is altered. The network's stability is monitored during the attack simulation. The network's information transfer stability ($NetworkStability_k^I$) is a measure of the ability of any two random nodes to communicate. The network's NCP stability ($NetworkStability_k^{NCP}$) is the extent of the physical fragmentation of the network into increasingly more isolated clusters.

$$s_i \Rightarrow NetworkStability_i \quad (39)$$

$$\beta \cdot NetworkStability_i \Rightarrow NetworkStability_k \quad (40)$$

$$NetworkStability_k \Rightarrow NetworkStability_k^I + NetworkStability_k^{NCP} \quad (41)$$

Transition 4: Steep fluctuations in the network's stability as depicted in Transition 3 will eventually lead to a set of halting conditions. Execution of the simulation is halted when the

"halting condition" is encountered. This is the stability threshold at which network communications are completely degraded. The "halting condition" H is Boolean.

$$NetworkStability_k \Rightarrow H; \tag{42}$$

$$iff\ (NetworkStability_k^I < 0\ \wedge\ NetworkStability_k^{NCP} < 2) \tag{43}$$

3) Network Fragmentation.

Network fragmentation is one measure of the network's connectivity stability used in this research. As previously discussed in Chapter II, it is a relative measure of the physical extent of the network's breakup during the attack. As considered in this research, Figure 3.2 and Figure 3.3 depict an example connectivity fragmentation that occurs during an attack. The circles represent nodes, and the numbers in the circles represent the node's degree. The darker line shown indicates a specific message path. The path represented flows from its source, through two intermediate nodes and then to its final destination. The finer lines indicate links between neighbor nodes. These two figures represent a small area of network activity and a single message path from the much larger overall network.

Figure 3.2 represents a small portion of the network before the attack has commenced. It depicts normal operating conditions with no network fragments. Figure 3.3 depicts an attack on node B resulting in its removal from normal connectivity. When node B is removed, a new path is formed and all nodes previously connected to node B break off into smaller fragments. The fragment sizes vary and the number of fragments that occur is equal to the degree of node B. Some fragments may have only one node; these nodes were previously defined as orphan nodes. In Figure 3.3, the arrows point to the connectivity fragments. As a result of the attack, connectivity between the source and the destination requires an additional two intermediate nodes for completion.

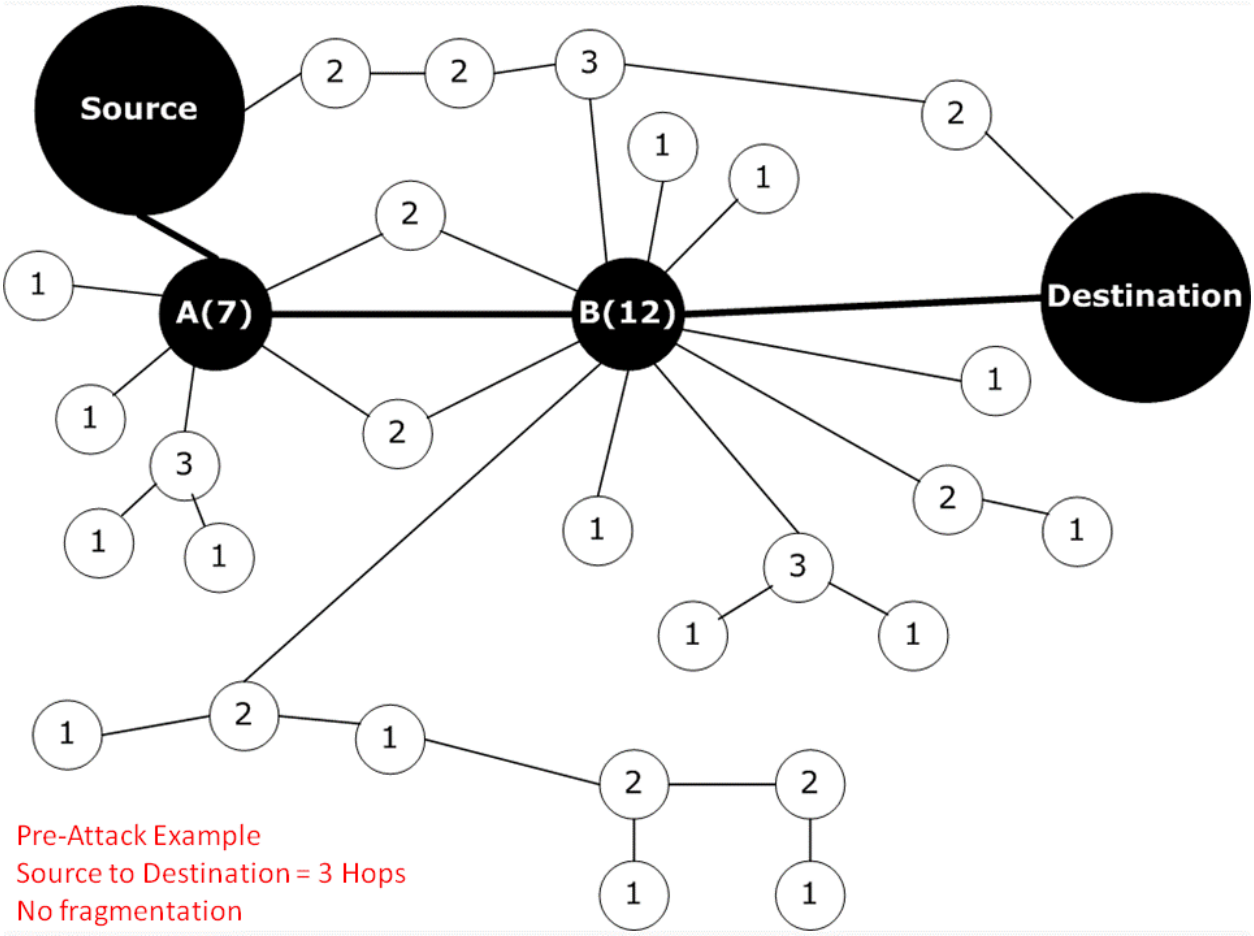


Figure 3.2. Pre-Attack network example of message path hops over small region

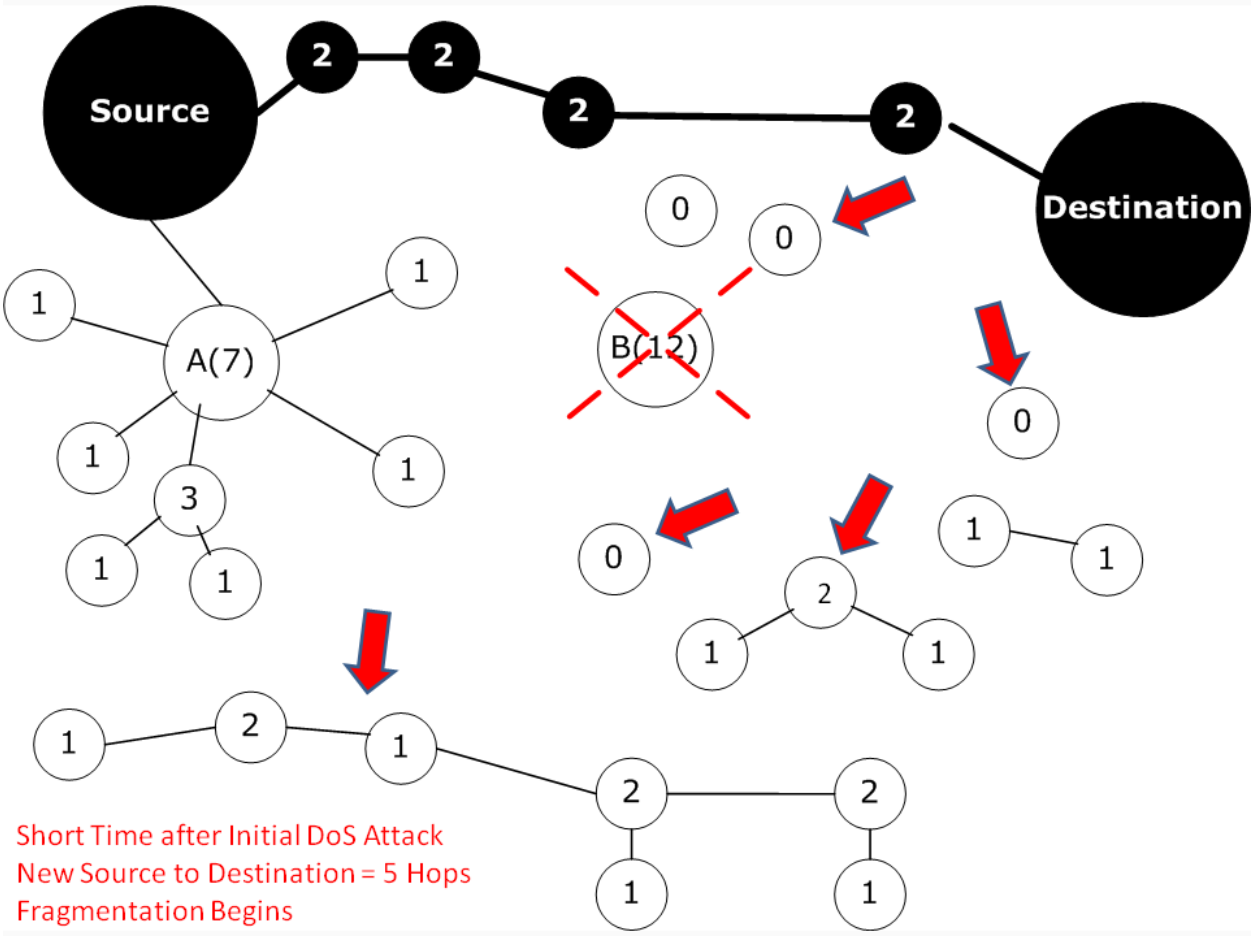


Figure 3.3. Attacked network at some point after attack, message path change and fragmentation

During the attack as the attacked nodes are removed, the cascading extent of the network-wide fragmentation dominates all connectivity patterns. Eventually, the network's stability is completely degraded. This progression is shown in Figure 3.4. Region A depicts the network before the attack without fragmentation. Region B reflects some intermediate snapshot over time and shows a partially degraded network. Region C represents the final state of the network after it has been totally degraded. This total connectivity stability degradation occurs because the network is predominant small isolated fragments of 2 nodes.

During this research, changes in the network's stability during the attack simulations as indicated by the network's fragmentation were measured in controlled intervals. The network stability changes were defined previously in this section during the discussion of Transitions 3 and 4.

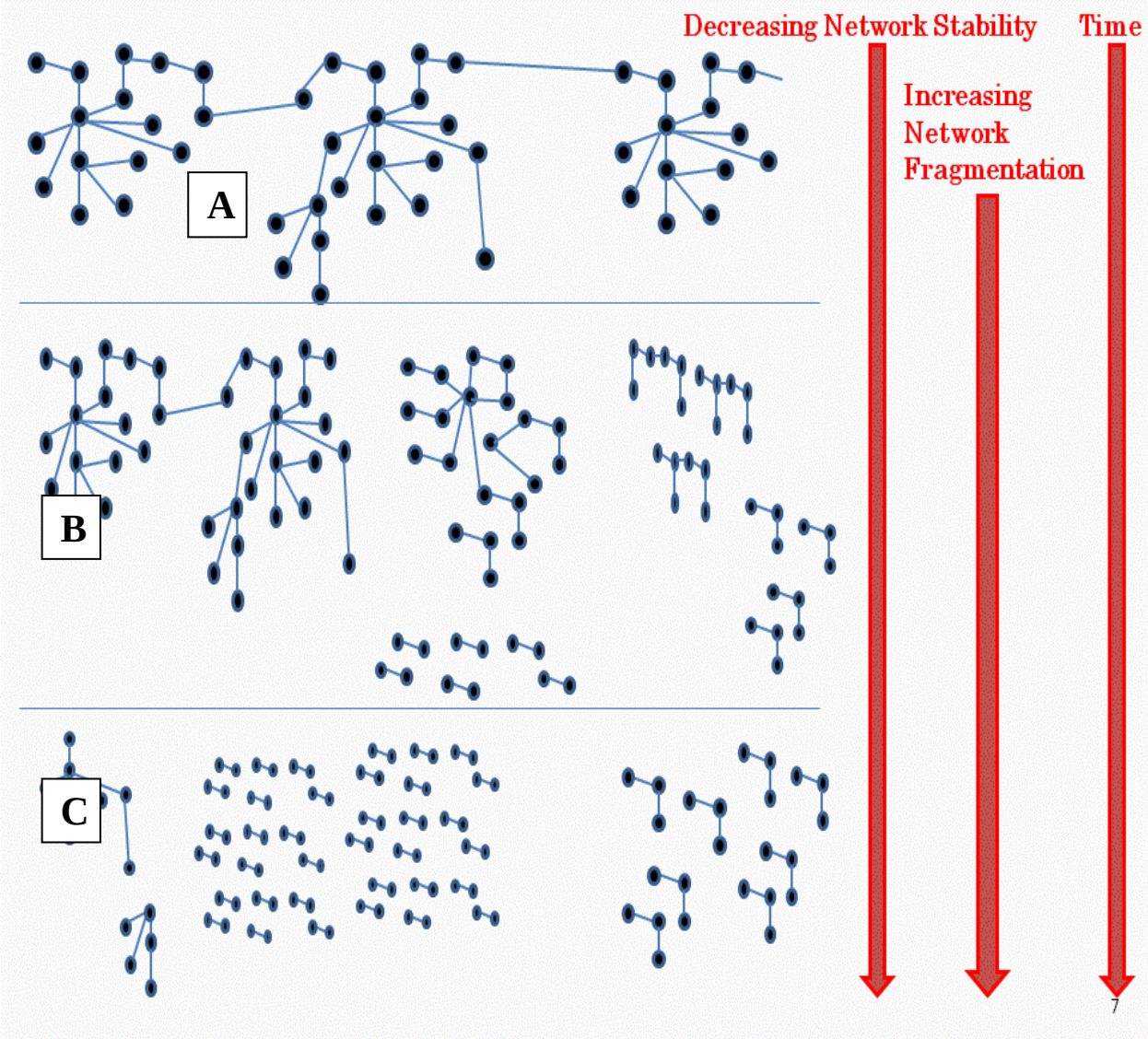


Figure 3.4. Illustrative example of network fragmentation over time

D. Computational Foundations

This section will discuss the foundational network connectivity definitions used in the research computations. This will be followed by a discussion of the pre-attack data and its descriptive statistics.

1) *Network Connectivity Stability Computations.*

The effects on network connectivity stability of the simulated denial-of-service (DoS) attack were measured primarily using two indicators, mutual information transfer and the network connectivity parameter. The network's mutual information transfer was used in this research to represent the loss of information transmission capabilities during an attack. The extent of the network's physical fragmentation was monitored using the network connectivity parameter. Along with the information entropy, these two measures represent the network's loss of heterogeneity during the simulated DoS attack. As previously discussed in Chapter II, the link heterogeneity of a scale-free network is representative of its connectivity robustness and stability. Table 3.3 presents the network connectivity terms used in the research computations.

Table 3.2. Network connectivity terms used in the research computations

Term	*Definition*
k	Node degree.
$m(k)$	Number of nodes with k-degree.
$m(k_1k_2)$	Total number of k_1-degree nodes linked to k_2-degree nodes.
m	Total number of active links for the network.
N	Total number of active nodes for the network.
k_{Max}	Maximum k-degree of the network.
$\mu(k_1k_2)$	Constant value, if $k_1 = k_2$ then 1 else 2; a weight assigned for joint degree computation.
$p(k)$	Probability that a node with k-degree node will establish a link; $p(k) = \frac{m(k)}{n}$
$p(k_1k_2)$	Probability that a node with k_1-degree node will establish a link with a k_2-degree node; $p(k_1k_2) = \mu(k_1k_2)\frac{m(k_1k_2)}{2m}$

All terms and their relevance in the equations below were previously discussed in Chapter II.

The computations used in this research are defined as follows:

Mutual Information Transfer (I), relative to its joint entropy (Boschetti et al. 2005; Schreiber 2000; Srivastav, Ray, and Gupta 2009),

$$I = \sum_{k_1=1}^{m} \sum_{k_2=1}^{m} p(k_1k_2) log_2 \frac{p(k_1k_2)}{p(k_1)p(k_2)} \quad (44)$$

Information Entropy (H) (Gudkov and Montealegre 2008)

$$H = -\sum_{k=1}^{N-1} p(k) \log_2 p(k) \quad (45)$$

Network connectivity parameter ($\mathbb{K}$) (Cohen 2001; Gallos and Argyrakis 2007)

$$\mathbb{K} = \frac{<k^2>}{<k>} \tag{46}$$

Average Node Degree ($\bar{k}\ or\ <k>$) (Mahadevan et al. 2005)

$$<k> = \frac{2m}{n} \tag{47}$$

Average Neighbor Node Degree ($\bar{k}_{nn}\ or\ <k_{nn}>$) (Mahadevan et al. 2005)

$$<k_{nn}> = \frac{\sum_{k=1}^{k_{Max}} k_{nn}}{m} \tag{48}$$

2) *Pre-Attack Network Connectivity.*

Table 3.3 depicts the pre-attack connectivity used in this study. Equations (44) through (48) were used for the computations found in Table 3.3. The pre-attack state ($t = 0$) depicted in Table 3.3 was the baseline condition for all simulated attacks. Previously cited literature indicates that a scale-free network's scaling degree exponent is between 2 and 3. Graphical analysis of its log-log plot for the degree (k) versus the frequency of each degree in the network is commonly used to determine the degree exponent of a power law network. Figure 3.5 and Figure 3.6 depict the degree distribution of the pre-attack network. The degree distribution plot depicted in Figure 3.5 was consistent with the literature. The degree exponent for the AT&T router infrastructure was found to be approximately 2.26 and therefore exhibits power law degree distribution behavior. The baseline router infrastructure used in this research represented a scale-free network. All values were consistent with the literature previously cited (Albert, Jeong, and Barabasi 2000; Barabasi and Albert 2002;

Crucitti et al. 2004; Demetrius and Manke 2005; Mahadevan et al. 2005; Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004; Wang et al. 2006).

Table 3.3. Pre-Attack simulation baseline connectivity

Term	*Definition*	*Pre-Attack Value*
m	Total links for the network	28,592
N	Total nodes for the network	11,800
k_{Max}	Maximum k-degree of the network	68
$\bar{k}$	Average node connectivity degree for the network	4.8
$\bar{k}_{nn}$	Average neighbor node connectivity degree for the network	15.8
$p(k)$	Probability that a node with k-degree node will establish a link: $p(k) = k^{-\alpha}$. See Figure 4.22 and Figure 4.23.	
α	Scale Coefficient (Alpha)	2.26
I	Mutual Information Transfer	1.56
H	Information Entropy	1.48
$\mathbb{K}$	Network Connectivity Parameter	15.8

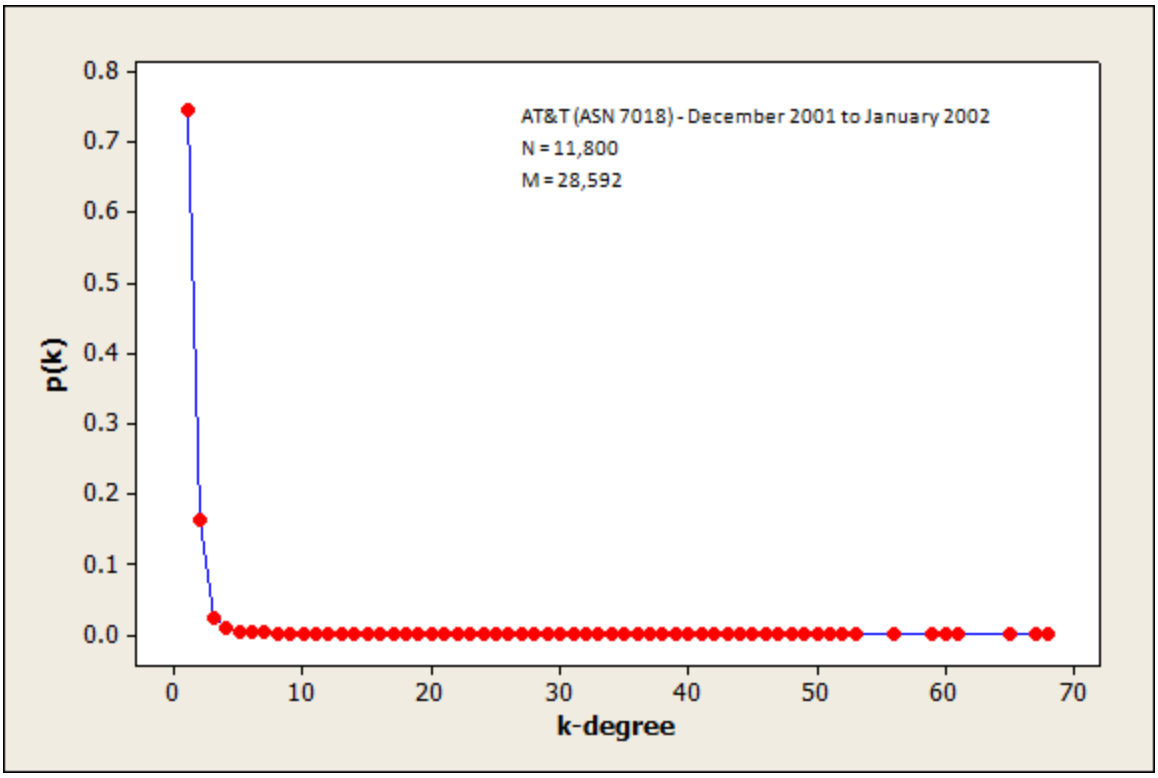


Figure 3.5. Pre-Attack network degree distribution

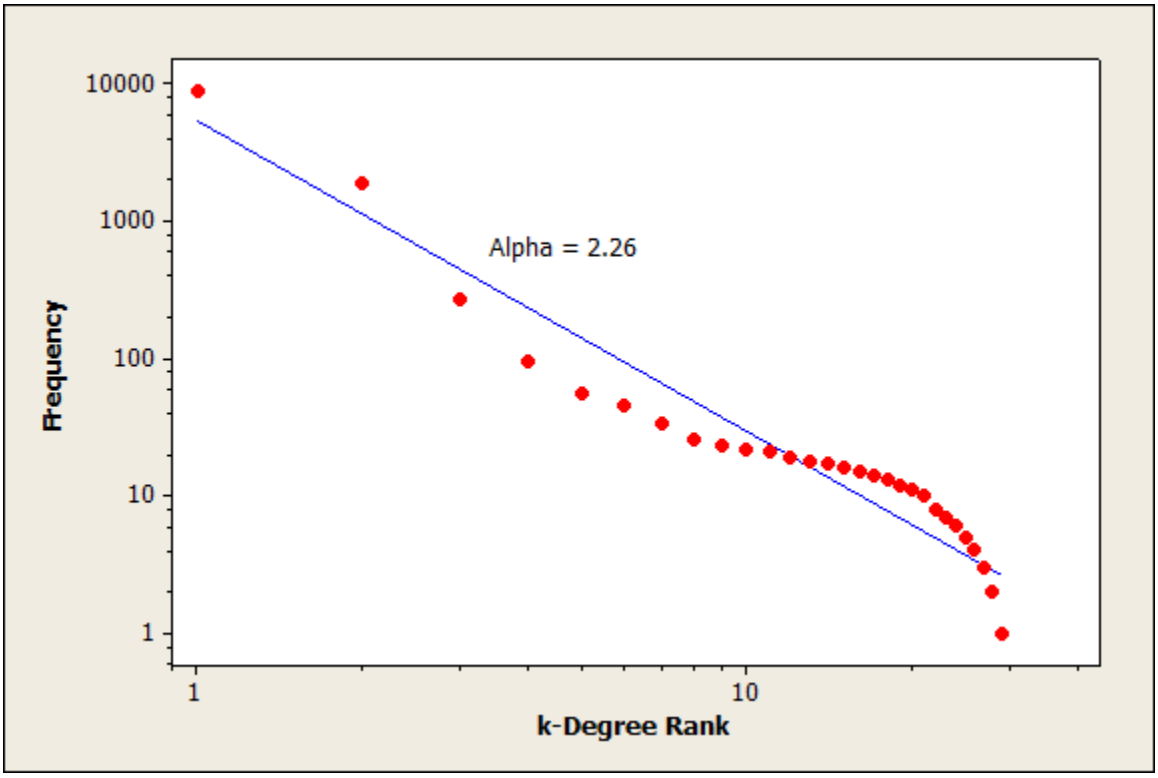


Figure 3.6. Pre-Attack network rank/frequency plot (log-log scale)

In this chapter, the formal definitions and foundational node representations were discussed. Chapter IV will now depict the cyber attack model and simulation designed for this research that was based on the definitions presented in this chapter.

CHAPTER IV. MODEL AND SIMULATION DESIGN

This chapter discusses the attack model and simulation used in this research. As previously discussed in Chapter I, it should be noted that routers are represented as nodes and the links between the nodes represent router communication adjacencies as defined in their router tables. Specific computer code, functions and declarations can be found in Appendix A and Appendix B.

A. Simulation Strategy

A robust, reusable, automated model has been developed that simulates complex scale-free computer network communication connectivity. The model simulates targeted denial-of-service attacks. These attacks were simulated through random removal of the network's most highly connected nodes. This research studied the resulting cascaded node failures and their effects on network connectivity. Network communications between routers was represented as node-pairs. Node-pair relationships under attack were studied.

The representation of network connectivity using node-pair relationships was foundational to this research. The rationale for this conceptualization is consistent with the basic algorithm for Internet message transmission. Over the Internet, a complete message path, from its initial source to its ultimate destination, consists of consecutive multiple hops between intermediate router pairs. As depicted in Figure 4.1, it is common to represent a complete path as the sum of each of its individual intermediate router-to-router ordered pairs (ordered because the path is one-way). The numbers shown in the circles represent router numbers. Source and destination shown in the figure represent the original source and the final destination. Figure 4.1 depicts 4 distinct communication paths in the following order:

(1) from the original message source to intermediate destination represented by router 1, (2) from the intermediate source represented by router 1 to the next intermediate destination represented by router 2, (3) from the intermediate source represented by router 2 to the next intermediate destination represented by router 3, and (4) from the intermediate source represented by router 3 to the final destination represented by router 4.

As a whole, the formation of these intermediate node-pairs over the entire message path reflects router communication relationships of the network. This research used these relationships to study targeted denial-of-service attacks over a scale-free computer network, specifically the Internet's router infrastructure.

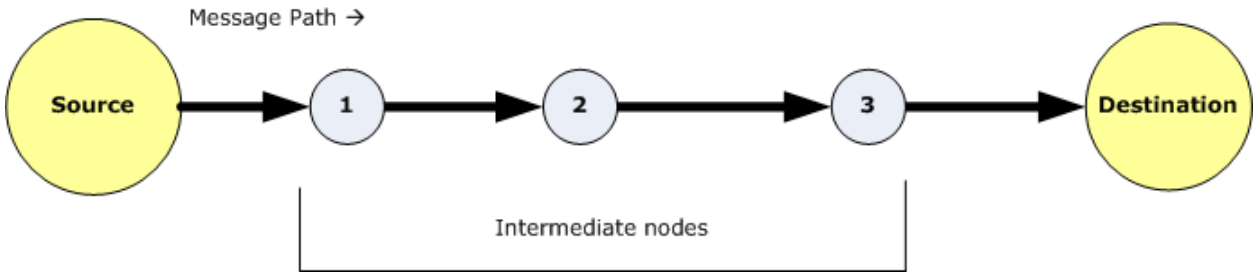


Figure 4.1. Node-pairs and network connectivity model

The model and simulation of network connectivity presented in this research is predicated upon an accepted modeling and simulation language known as Colored Petri Nets (CPN). The characteristic connectivity patterns of a network were emulated using node-pairs from a real network infrastructure. This research used the United States AT&T core infrastructure node-pairs to develop the simulation's pre-attack state. Attacks against this baseline were then simulated and anomalies in the distribution of these node-pairs were studied. Changes in the underlying communication characteristics of the network, such as its

mutual information transfer, were also studied. The next section in this chapter introduces the foundational assumptions used in this research to develop the model and simulation.

B. Foundational Assumptions

The following assumptions were made during the research simulations and all subsequent data analysis:

1. This research does not introduce attack motivation theories or discussion.
2. The simulations were limited to scale-free computer network connectivity mechanisms as related to preferential attachment and network traffic flow characteristic patterns.
3. The cyber attack was simulated in a virtual environment; there were no live Internet experiments performed in this research. However, real Internet router adjacency data were used to prime the pre-attack CPN simulations.
4. The CPN simulations were executed using the Microsoft Windows XP SP2 operating system and CPNTools version 2.2.0.
5. This research was limited to large-scale (regional backbone) router connectivity topologies and their degree distribution characteristic changes during a cyber attack.
6. This research assumes that the attacker has knowledge of the Internet's router infrastructure and has identified critical routers. This knowledge will not be modeled.
7. Ordered node-pairs represent network connectivity relationships.
8. Network stability was considered a relative function of its ability to transfer information and the extent of attack-induced physical connectivity fragmentation.
9. Network stability is a gradual degradation influenced by its node-pair connectivity behaviors and multiple violations of preferential attachment theory.

10. Node removal attack modeling is the only method used to simulate an attack; link-based attacks are not considered in this research.
11. All node-pairs are considered of equal weight (one link per node-pair). However, one node may have multiple physical links to the same node.

C. CPN Modeling and Simulation Language

The modeling and simulation language for this research was Colored Petri Nets (CPN) as realized through CPNTools (Kristensen, Christensen, and Jensen 1998). CPNs are used to model and simulate a wide variety of industrial strength applications through virtual representations (Kristensen, Christensen, and Jensen 1998). These include communication protocols, audio/visual systems, operating systems, hardware designs, embedded systems, software system designs and business process re-engineering. This research methodology represented a unique application of CPNs.

1) CPN Syntax.

CPN model development may be done in both a graphical and textual programming environment. The foundational building blocks of the CPN programming syntax are places, tokens, arcs, colors, transitions, markings and guards. A *place* represents an environment (such as a network router) and is assigned markings (token values) to portray the system state (configuration) of that *place. Tokens* are computer bit strings (such as variable values) that are transmitted across *arcs* (communication lines) to other places (routers). Communication between 2 places is facilitated by *transitions. Transitions* are enabled so that tokens may be passed between 2 *places* in the simulation. It is possible to transmit multiple *tokens* between

2 *places* sequentially or concurrently. *Colors* (programming declaration statements) provide the essential data types for the information stored on the *places*.

Syntactically, *places* must be connected to transitions. *Arcs* connect places and transitions and they allow state changes. The flow of *tokens* is controlled through *arc* conditionals and transition *guards*. As *transitions* are enabled they execute model code and control the information flow between *places*. For a contextual understanding, *tokens* will be referenced as being moved between places. However, the actual underlying simulation actions represented are state changes. When a *transition* between 2 places is enabled, the tokens bound to these places are altered. This reflects a state change for both *places*.

2) An Illustrative Simulation Example.

A simple illustrative example of a CPN simulation is shown in Figure 4.2, Figure 4.3, and Figure 4.4. In Figure 4.2, the "bindings" on place 1 consist of 2 tokens of the integer data type (color). As shown in Figure 4.2, for each simulation clock tick, variable *n* is bound to a randomly selected token passed from place 1. If the arc and "transition A" guard conditional statements are true, then "transition A" is enabled. The token is bound to variable *n* is passed to "transition A." If one or both conditional statements evaluates to false, then no token is passed. In this example, the value 5 is bound to variable *n* and is passed to "transition A."

As depicted in Figure 4.2, as the token passes through "transition A," if there is a functional code object associated with the transition, it will be executed using any available input tokens. Predicted upon the execution of the transition function, the tokens that are input to the transition may be altered. The function output is then placed on all outbound arcs from the transition. In this case, the input is token n with a value of 5 and the transition

computes n plus 10 for the output. The output of "transition A," shown in Figure 4.2, is bound to the token variable "a" and its value is 15. This token will be passed to place 2. After the first few clock ticks, place 1 will have no more bindings, and place 2 will be bound to 2 integer tokens with the values of 15 and 11 as computed by the function on "transition A." In this example it is assumed that the guard and arc conditional statements are true. Therefore, 15 is the result of the first clock ticks, and it is followed by an output of 11 with subsequent clock ticks. The results of the first 2 firings of "transition A" are depicted in Figure 4.3 and Figure 4.4.

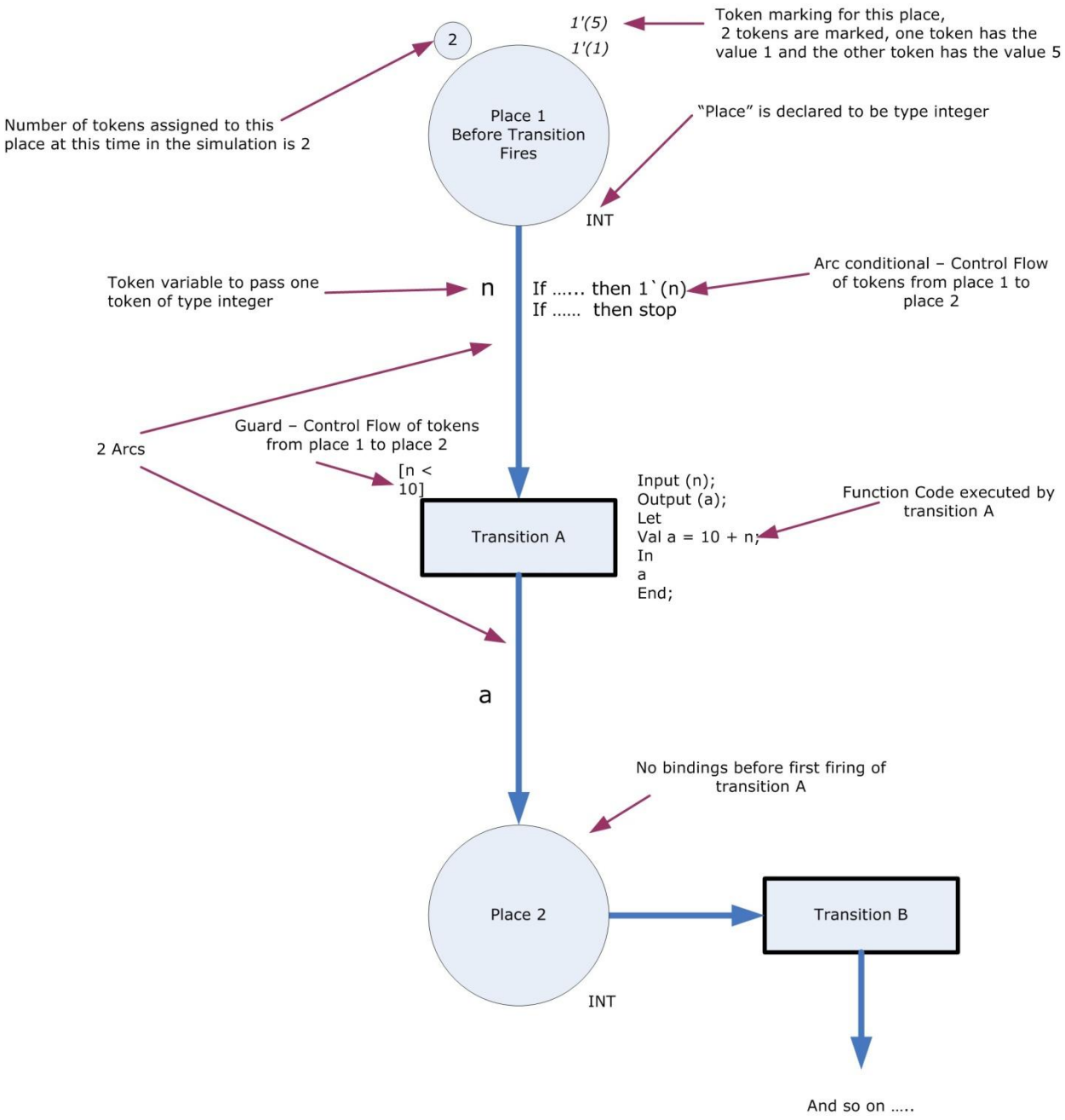


Figure 4.2. Colored Petri Net example, before transition “A” fires

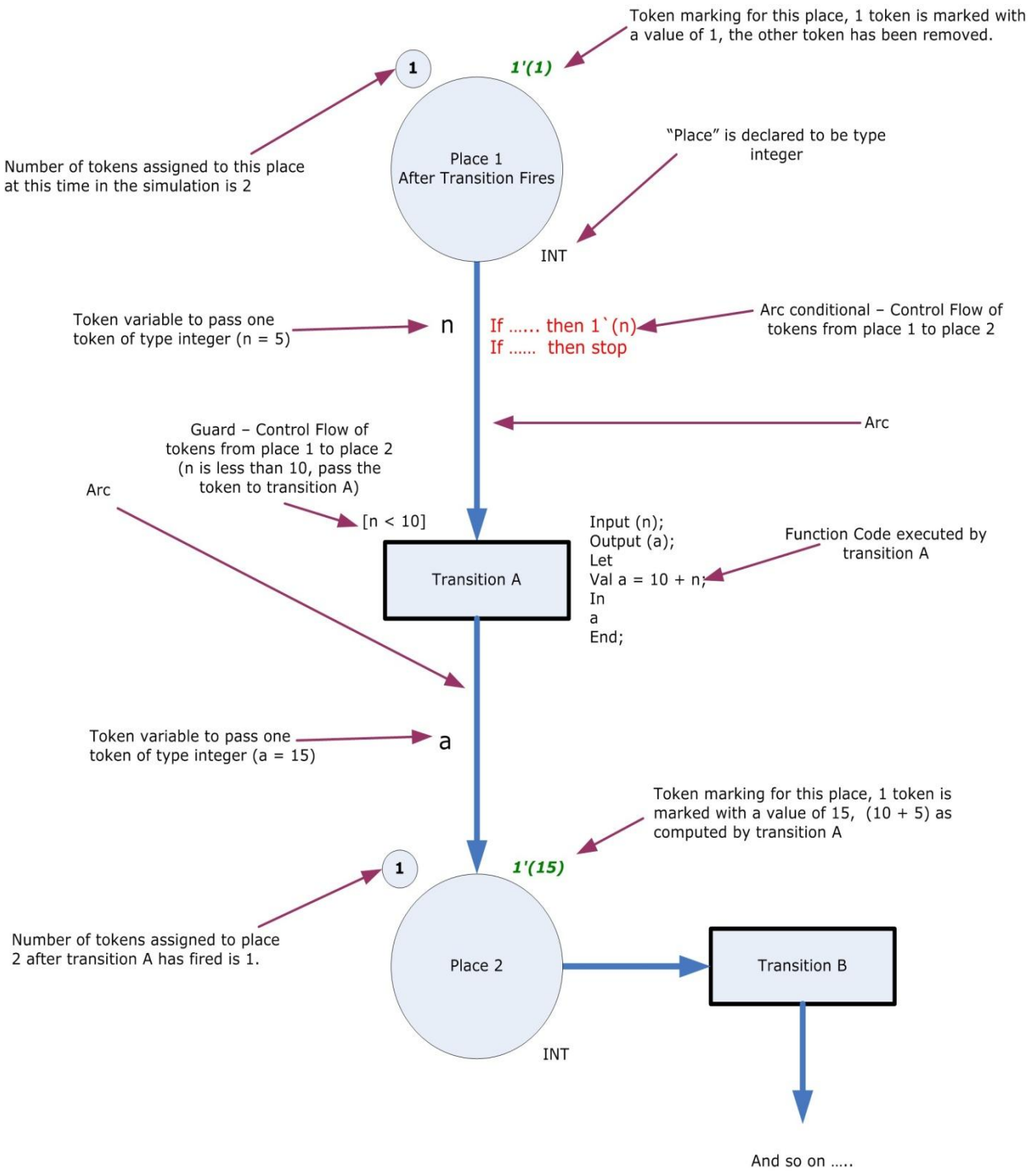


Figure 4.3. Colored Petri Net example, after transition “A” fires once

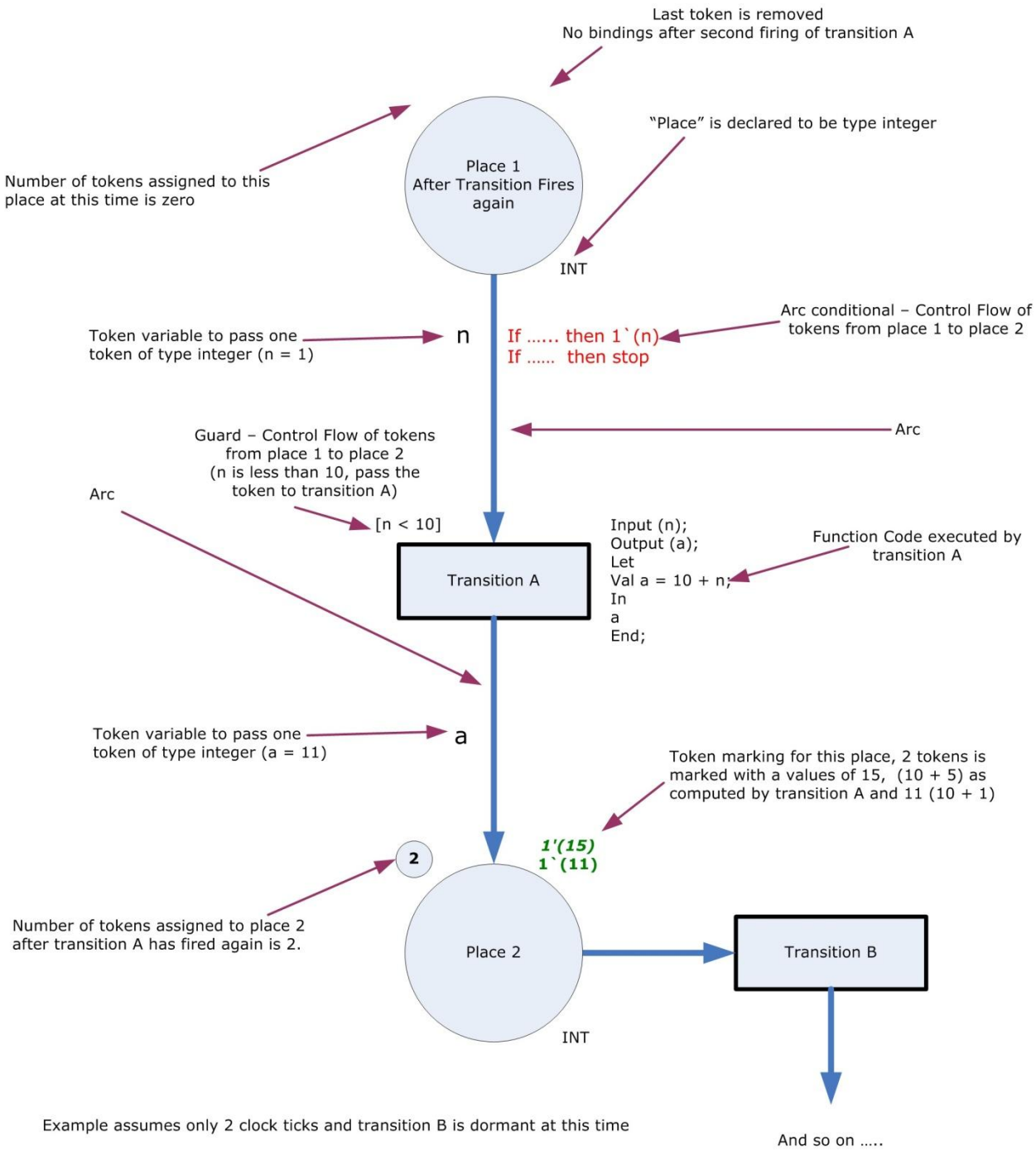


Figure 4.4. Colored Petri Net example, after transition "A" fires a second time

3) Justifications for Modeling Language Selection.

Jensen (Jensen 1997, 1998), an international authority on Colored Petri Nets, describes this modeling and simulation language. Kurt Jensen and his team at the University of Aarhus in Denmark are major contributors to the development and use of CPN models. They define the essence of the language as follows: "Colored Petri Nets provide a framework for the construction and analysis of distributed and concurrent systems" (Kristensen, Christensen, and Jensen 1998). Cyber attack interactions and router message transmissions over a scale-free computer network are complex, concurrent, and distributed. Therefore, CPN modeling is an ideally suited, mathematically proven virtual representation that may present these complex systems in a controllable and analytical context (Jensen 1994, 1997, 1998; Kristensen, Christensen, and Jensen 1998; Kristensen and Christensen 2004). Currently, there are many tools for design, development, and simulation of Color Petri Nets (Jensen 1994, 1997, 1998; Kristensen, Christensen, and Jensen 1998; Kristensen and Christensen 2004). Jensen's team has developed an intuitive tool for CPN modeling and simulation (the CPN Tool) that is used by more than 700 research organizations in over 70 different countries (Wang et al. 2008). Their CPN Tool will be the foundational instrument used for model development and simulation utilized in this research.

Advantages of CPN modeling and simulation as stated by the CPN group include (1) an intuitive modeling language that allows for the flexibility and power of modern programming languages as well as graphical representations of the model; (2) well-defined semantics leading to unambiguous models of system behavior; (3) a flexible modeling environment that can be used in a wide variety of complex industrial-strength applications; (4) a modeling language consisting of a few powerful programming primitives; (5) a model

that exhibits true concurrency, not interleaving; (6) timed and probabilistic simulation functionalities; and (7) a universally accepted formal analysis and verification of any derived model (Wang et al. 2008).

D. General Research Simulation Design

During the simulation, as node-pairs are "released" into the simulation stream, nodes are randomly "designated" for attack and undergo state changes as described in Chapter III. This section will summarize the simulation flow and node state changes.

1) Simulation Design Information Flow.

Figure 4.5 depicts a summary of the information flow for this research design. The pre-attack priming data were extracted from the Rocketfuel datasets (Spring et al. 2004). They will be discussed in Section E. The raw Rocketfuel data were formatted as CPN tokens by a series of offline Visual Basic routines specifically developed for this research. As discussed in Chapter III, the network's connectivity state was represented as a function of its node-pairings and statistical mechanics. As nodes were removed, the node-pairings changed and eventually led to cascading node failures. The network went through a series of intermediate node-pairings that influence network stability and its degree of fragmentation. As the attacked nodes were removed from the connectivity state, the output generated was an audit trail file that consisted of all changes to the network's node states. The network fragmentation and connectivity degradation continued until the halting conditions were encountered. The simulation was then terminated. These halting conditions were previously discussed in Chapter II, Section A.

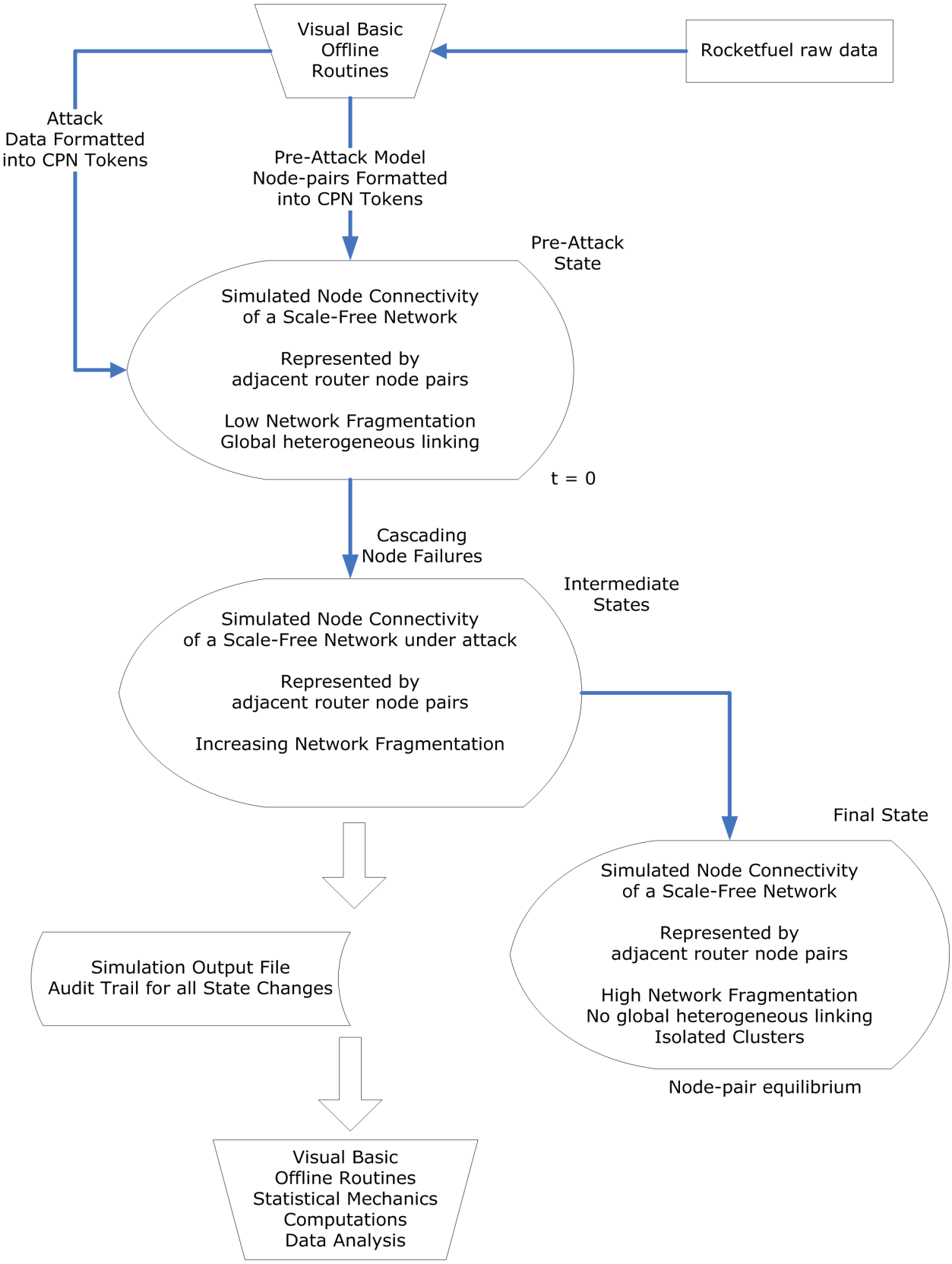


Figure 4.5. Summary of simulation process information flow

Through pre-simulation sensitivity analysis it was determined that 6 million CPN execution steps was sufficient to encounter halting conditions for all attack scenarios in this research. Each simulation run was implemented and halted after the same number of CPN execution steps (6 million). The audit trail of node state changes and the distribution of intermediate node-pairs were used as input to another set of offline Visual Basic routines. These routines were specifically designed for this research. The offline computations generated the relevant statistical mechanic data for research analysis.

2) Node Interactions.

The previous discourse presented the general flow of information in the simulation. The core processes of the simulation were driven by node state changes. Node state data were collected at pre-determined time intervals during the simulation. The intervals were determined using significant changes in network stability. For each unique attack scenario, from the set of critical nodes, attacked nodes were randomly "designated" by the simulation. Critical and attacked nodes were previously defined in Chapter III. The number of critical nodes does not exactly equal the top percent of all nodes as depicted in Table 4.1. Since multiple nodes have the same degree, the number of critical nodes will be greater than the percent of the total number of nodes. The total number of nodes in the pre-attack network was 11,801.

Table 4.1. Simulation runs classes denoting critical node removal proportions

Run Class	*Top xx percent of nodes by degree sorted in highest to lowest degree order (Critical Nodes)*	*Number of Critical Nodes*	*Tolerance Parameter*
RC0051	xx = .50%	60	.10
RC0101	xx = 1.0%	124	.10
RC0151	xx = 1.5%	179	.10
RC0201	xx = 2.0%	241	.10
RC0251	xx = 2.5%	304	.10
RC0301	xx = 3.0%	363	.10
RC0351	xx = 3.5%	421	.10
RC0401	xx = 4.0%	478	.10
RC0451	xx = 4.5%	544	.10
RC0501	xx = 5.0%	613	.10

As depicted in Table 4.1, each individual attack scenario was represented as a run class. Each attack scenario was initially applied against the same pre-attack network connectivity state baseline. As shown in Figure 4.6, the initial pre-attack connectivity state was altered as attacked nodes were removed from the simulation. This led to cascading node failures which, in turn, generated multiple intermediate connectivity states. Each intermediate state was the result of state changes of the network's nodes. During the simulation, one node in a released node-pair attempted communication with the other node in the pairing. If one of the 2 nodes were determined to be a temporary orphan, then the temporary orphan recovery process was engaged. This process either created a new communication node-pairing or it generated a permanent orphan. The temporary orphan recovery process was defined earlier in this dissertation in Chapter III, Section A.

As the number of permanent orphan nodes increased, the cascading affect was magnified. This led to increasingly degraded network stability. The stability was measured through its connectivity topology. The connectivity states were represented by the set of active node-pairs, and their stability was computed using their degree distribution characteristics. The network connectivity simulation continued to execute until the halting conditions were encountered in the final state.

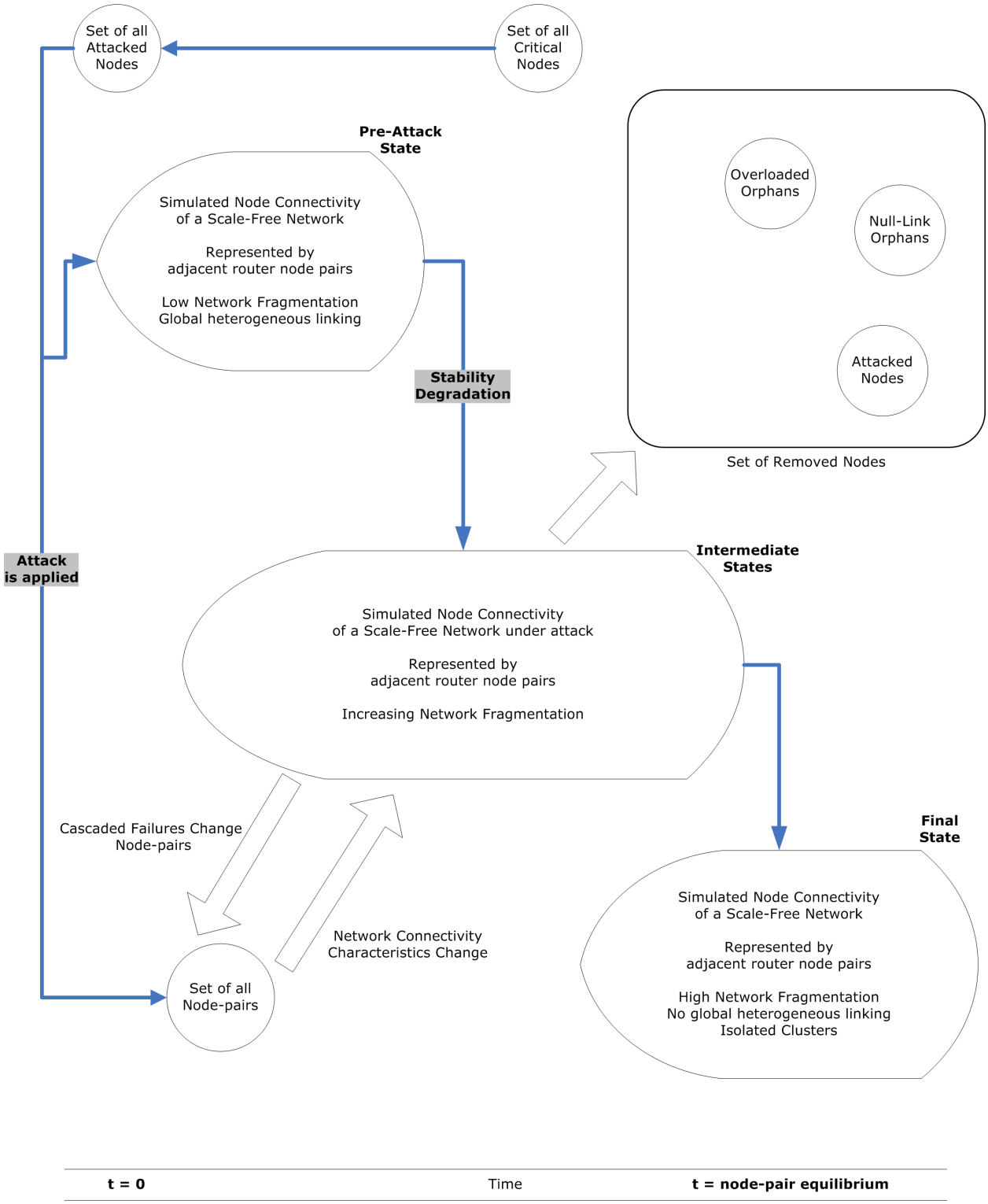


Figure 4.6. Summary of the node interactions during the simulation

E. Simulation Runs

1) General Strategy.

The overall process is summarized:

1. Extract Rocketfuel router adjacencies (node-pairs) for the pre-attack network connectivity state and use this to prime the CPN attack simulation.
2. Execute the simulation against a set of attack classes with varying degrees of severity.
3. Extract the simulation audit trail for each simulation run.
4. For each simulation run, compute the global network connectivity measures discussed in this chapter, such as mutual information transfer. This computation was performed offline.

Figure 4.7 depicts the simulation run architecture and the simulation run strategy used in this research. The Rocketfuel project extracted snapshots of selected autonomous systems data from the Internet (Rocketfuel: An ISP topology mapping engine n.d.; Spring et al. 2004; Spring, Mahajan, and Wetherall 2002). The extracted data consisted of backbone and access router pairs. The data collection methods used in this research will be discussed later in this chapter. The pre-attack state of this simulation was developed using one of the Rocketfuel datasets. Specifically, this research simulation focused on the United States AT&T backbone and access routers. Forty simulation runs representing 10 different attack scenarios were implemented. For each attack scenario, there was a corresponding run class. A run class represents an attack class and a protection class. The protection class for all runs is 1, representing a link capacity of 1.1 for all nodes. Link capacities were discussed in Chapter III. As shown previously in Table 4.1, the number of attacked nodes for each attack class ranged from 0.5% to 5.0% of the total number of network nodes.

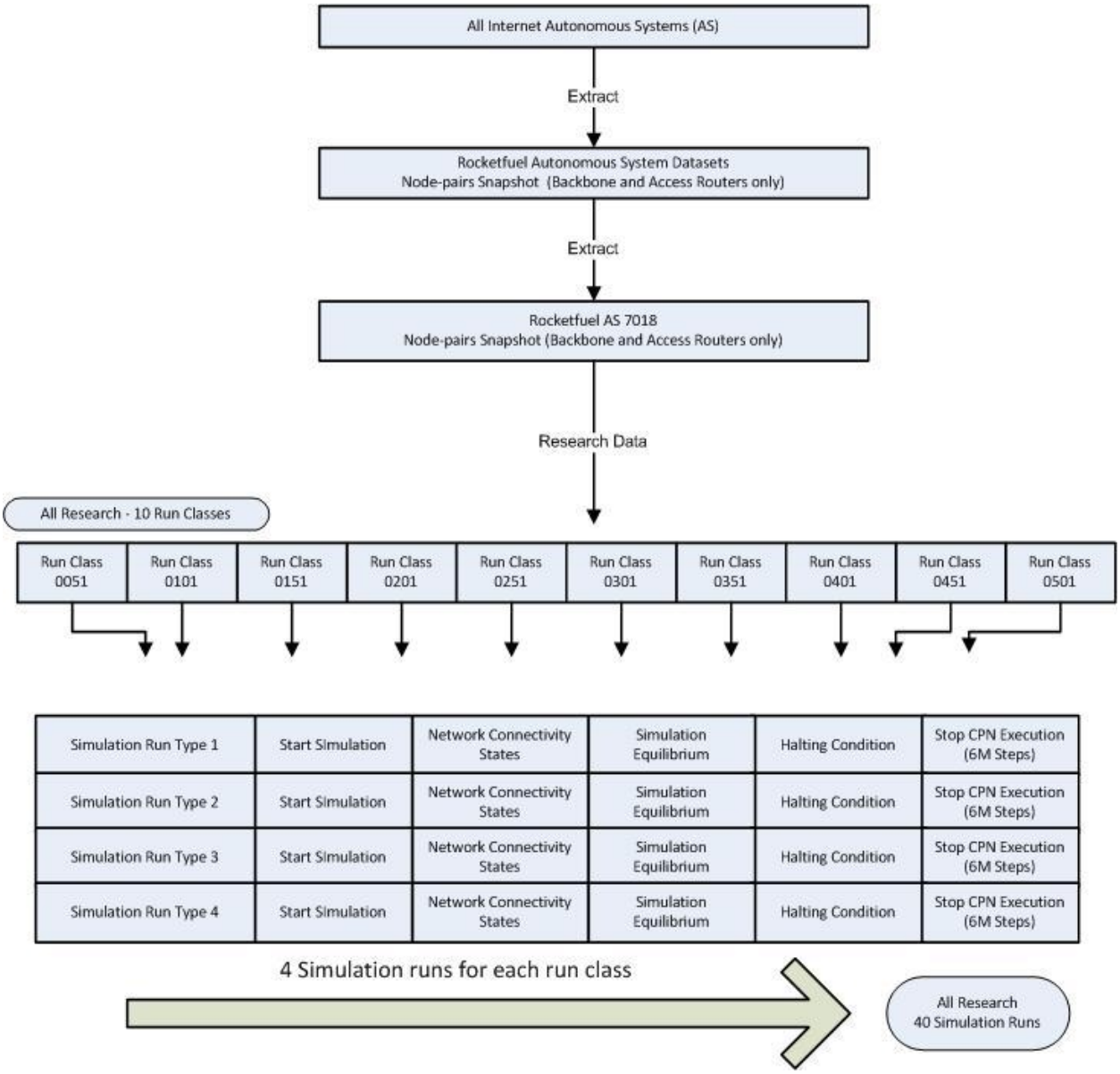


Figure 4.7. Simulation execution strategy

As shown in Figure 4.7, for each run class there were 4 simulation run types. Run type 1 represented the network simulation with no protection. Run type 1 simulation runs were executed first. Using these results, the protection strategies were developed. Table 4.2 depicts the nature of the nodes protected by each protection strategy. Node-1 is one node in an active node-pair, and node-2 represents the other node. For instance, run type 2 identifies

all node-pairs where node-1 is of degree 1 and node-2 is of degree 2. For run type 2, all pre-attack nodes with node-1 degree that are in a node-pair with a node-2 degree node were protected. A protected node cannot be removed from the network by a cascaded node failure or direct attack. Protection strategy development as an experimental treatment will be discussed in Chapter VI. As shown in Table 4.2, run type 1 represented the network without node protection, run type 2 is the network simulation using protection strategy 1, run type 3 protects the network simulation with protection strategy 2, and run type 4 implements protection strategy 3. Protection strategy 3 was studied as an extension of protection strategy 2. The purpose was to observe the effects of increasing the number of nodes from 1009 to 1257. Protection strategy 0 depicted the simulations without any additional protections and it was used as a baseline for the comparisons the other 3 protection strategies.

For example, as shown in Table 4.2, for all run type 4 simulations, node-pairs of type 1-2 found in the pre-attack network were protected by protecting one node in the node-pair. A node-pair type 1-2 consists of two nodes, one with a degree of 1 and the other with a degree of 2. The protected node for run type 4 was the node with a degree of 2 of all node-pairs of type 1-2. The protection strategy protected individual nodes in specific node-pair types. Only one of the nodes in the node-pair is protected. The protection determinations are made using the pre-attack network only. Node-pair types were defined in Chapter I.

The network connectivity states for all run classes, except 0451 and 0501, were computed at 50,000 ms (in real time) data collection intervals. Due to the severity of the attacks in run classes 0451 and 0501, the network degraded at a very rapid rate, leading to a limited volume of data points. Consequently, for these 2 run classes the data were collected

at 2,000 ms intervals. In Table 4.2, RC represented the set of all run classes: $RC = \{0051, 0101, 0151, \ldots 0501\}$ and * indicates that the column is not applicable.

Table 4.2. Protection strategy

Protection Strategy	*Run Classes (Total = 10)*	*Run Type*	*Degree of Node-1*	*Degree of Node-2*	*Protect*	*Number of Nodes to Protect*	*Protected % of all Pre-Attack Nodes*
0	RC	1	*	*	None	None	*
1	RC	2	1	2	Node-1	1009	8.5%
2	RC	3	1	2 and 3	Node-1	1257	10.7%
3	RC	4	1	2	Node-2	1009	8.5%

2) Simulated Attack, An Illustrative Example.

For clarification, Figure 4.8 presents a useful example. The following assumptions are in place for this example:

1. Node-B is a member of the set of removed nodes.
2. Node-A is an active node.
3. Node-X is an available neighbor node for node-A.
4. Node-X is selected using preferential attachment to communicate with node-A.
5. Node-X becomes overloaded after forming a new link with node-A.

As shown in Figure 4.8, the router table node-pairs are the set of all node-pairs $P(t)^{Active}$ prior to the attack at $t = 0$. It is static and assumes the router tables are not refreshed during the simulation. All released node-pairs are randomly selected from this set and released continuously throughout the simulation. A node-pair can be released more than

once. First, node-pair A-B is released; this represents a communication attempt based on the router tables between Node-A and its neighbor Node-B. The simulation stream processes determine whether Node-A and/or Node-B is a member of the set of removed nodes. The set of removed nodes is dynamically changed as nodes are orphaned and attacked. If Node-B is not a removed node, then the communication is successful and no other simulation actions are taken.

In this example, Node-B is a removed node. This leads to Node-A becoming a temporary orphan. The simulation stream process then selects a neighbor node of Node-A using preferential attachment mechanisms. The previously failed communication attempt will now be attempted using this newly selected node. In this example, Node-X is selected to facilitate the previously failed communication with Node-B. This leads to additional link load on Node-X. Before this communication is classified as successful, it must be determined whether additional load on Node-X has exceeded its link capacity. Link capacity was previously discussed in Chapter III. An example of this determination is depicted in Table 4.3.

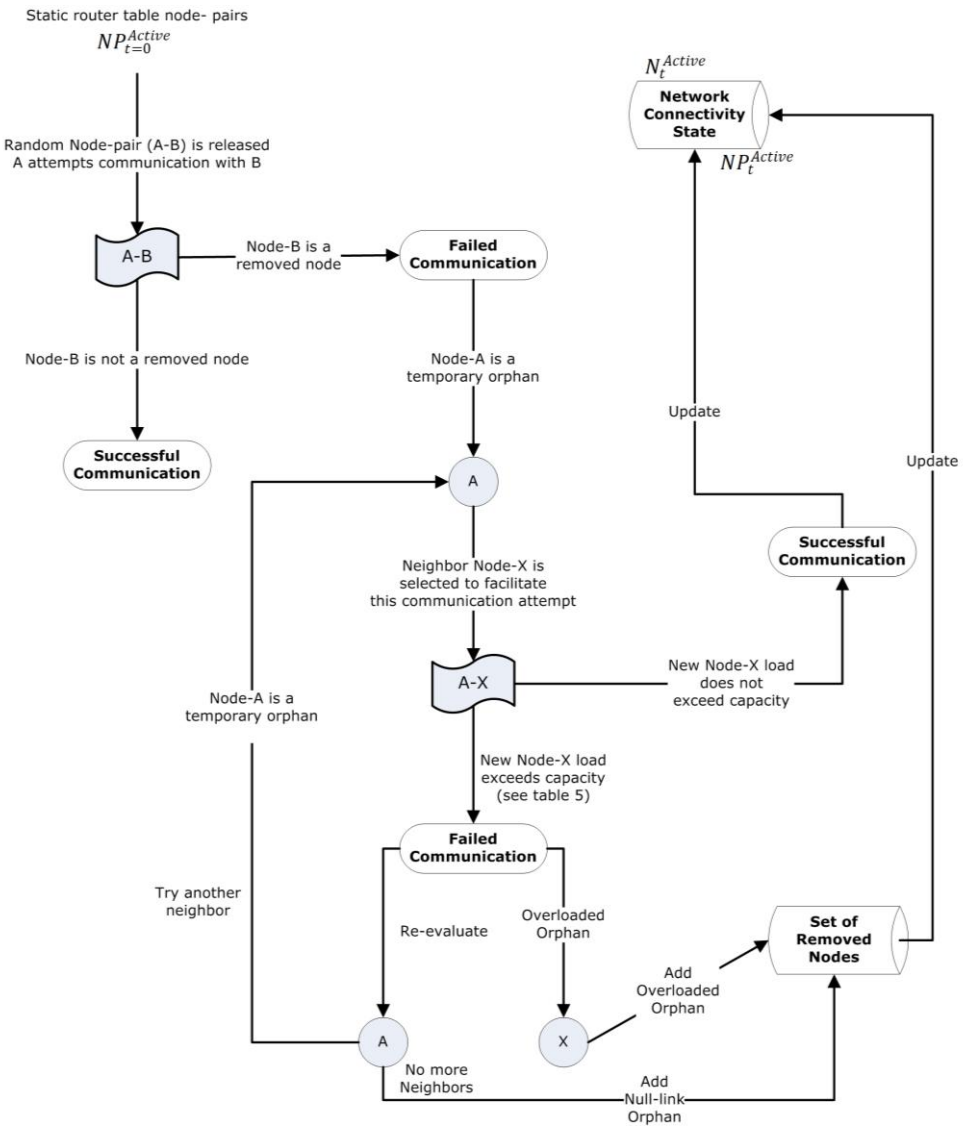


Figure 4.8. Communication attack simulation example

Table 4.3 depicts the connectivity profile of Node-X, before and after it was selected. If the link capacity is exceeded, Node-X is an overloaded permanent orphan, the network connectivity state is updated, and Node-X is added to the set of removed nodes. As a result, node-A is again a temporary orphan. It must now find another neighbor to complete the communication attempt. If all of its neighbors have been attacked or are permanent orphan

nodes, then Node-A will become a null-link permanent orphan. Then the network connectivity state is updated and node-A is added to the set of removed nodes.

Table 4.3. Overloaded orphan node profile in Figure 4.8

Simulation Action	*Node*	*Neighbor Node List*	*Degree*	*Capacity*
Before node-pair A-X (communication attempt)	A	C,X,W,C	4	6
	X	C,W,A,W,D	5	6
After node-pair A-X is incrementally added to X (overloaded orphan node X)	A	C,X,W,C	4	6
	X	C,W,A,W,D,A	6	6

If the link capacity of Node-X is not exceeded after the new communication with Node-A, then the communication is successful and the network connectivity state is updated. The network connectivity state is represented by the profiles of all active nodes and node-pairs, $P(t)^{Active}$ and $N(t)^{Active}$ at $t > 0$. All node connectivity states are stored in the CPN record structure called networkDB. A node connectivity profile stores a dynamic record for each node that includes its current degree and active neighbors and its current state. This process is repeated for all released node-pairs until the simulation halting condition is encountered. All node and node-pair states have been previously defined in Chapter III of this dissertation.

F. Network Connectivity State Computations

The audit trail recorded all node state changes during each simulation run over constant time intervals. This information was used to compute the network's temporal stability and degree characteristics. For each simulation run, the audit trail collected all node status (n_t^s) changes that denote a node's state. For each simulation run, the audit trail was used to represent the connectivity and stability characteristics of the attacked network over the life of the simulation. The status codes in the simulation were defined as:

1. Status = 0: node was considered active and a member of the set of active nodes ($N(t)^{Active}$).
2. Status = 1: node was considered an attacked node and a member of the set of attacked nodes ($N(t)^{A}$).
3. Status = 3: node was considered a Null-Link node and a member of the set of Null-Link nodes ($N(t)^{Null}$).
4. Status = 4: node was considered a overloaded node and a member of the set of overloaded nodes ($N(t)^{OL}$).

The audit trail output file recorded all node state changes and other node specific information. The information collected was node-id, node status, node neighbor list, time of change in milliseconds, and timestamp in user clock time hours and minutes. After each simulation run was halted, this information was fed to a series of Visual Basic computational routines. These routines were developed specifically for this research. At specific time intervals, these routines computed the network's connectivity state, the extent of the network's fragmentation, and its stability. Appendix C presents the all output files generated for each simulation run.

1) Pre-Attack Data Collection Methodology.

Traceroute is a well-known computer network research tool that traces network packets from their original source, through their intermediate paths and then to its ultimate destination. It has been found to be an appropriate technique for collecting network path data for studies of the Internet's topological characteristics (Leguay et al. 2007; Mahadevan et al. 2006; Mahadevan et al. 2005). Rocketfuel is a traceroute-based Internet topology mapping tool developed at the University of Washington (Alderson et al. 2005; Donnet and Friedman 2007; Liljenstam, Liu, and Nicol 2003; Spring et al. 2004; Spring, Mahajan, and Wetherall 2002). From December 2001 to January 2002, the Rocketfuel team collected large volumes of ISP-level traceroute path data representing the 10 distinguishable world-wide ISP router infrastructures. These data provided the requisite router adjacency relationships needed to model the pre-attack network state used in this research.

An illustrative example of the router connectivity architecture of the data collected by the Rocketfuel project team is depicted in Figure 4.9. Each Internet Service Provider (ISP, also known as an autonomous system [AS]) consists of multiple server regions referred to as points-of-presence (POP). Each POP connects user and enterprise level servers. Three POP regions are shown in Figure 4.9. Special purpose routers that connect the POP regions are known as backbone and access routers. These routers facilitate large volumes of Internet traffic within and between POPs. They also facilitate network traffic among the ISPs.

Figure 4.9 illustrates one ISP with 3 different points-of-presence depicted as POP-1, POP-2, and POP-3. As shown in the system architecture example in Figure 4.9, backbone routers (layer 0) connect each POP in the ISP. Access routers connect the enterprise (user) level routers to each other and to the backbone routers. The actual number of access routers

in a POP varies. Most inter-POP communication travels through the backbone routers, and most intra-POP traffic travels through the access routers. The pre-attack router adjacencies used in this research simulation represent the backbone and access router communications of the United States AT&T (ASN7018) ISP regional infrastructure. Figure 4.10 was taken from Spring, Mahajan, and Wetherall (2002) and depicts a descriptive rendering of the for layer 0 (backbone) routers in this infrastructure.

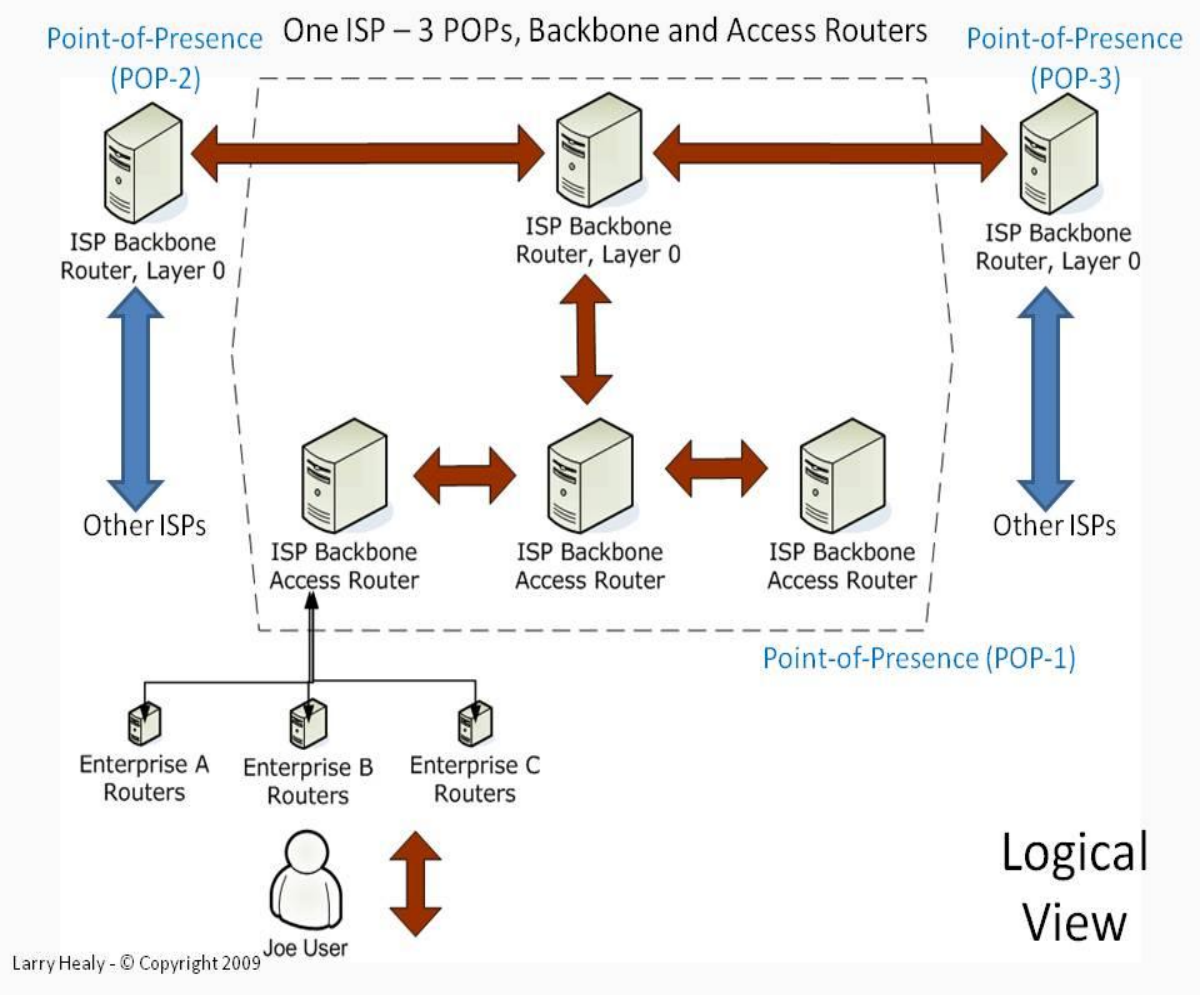


Figure 4.9. POP backbone and access router architecture example

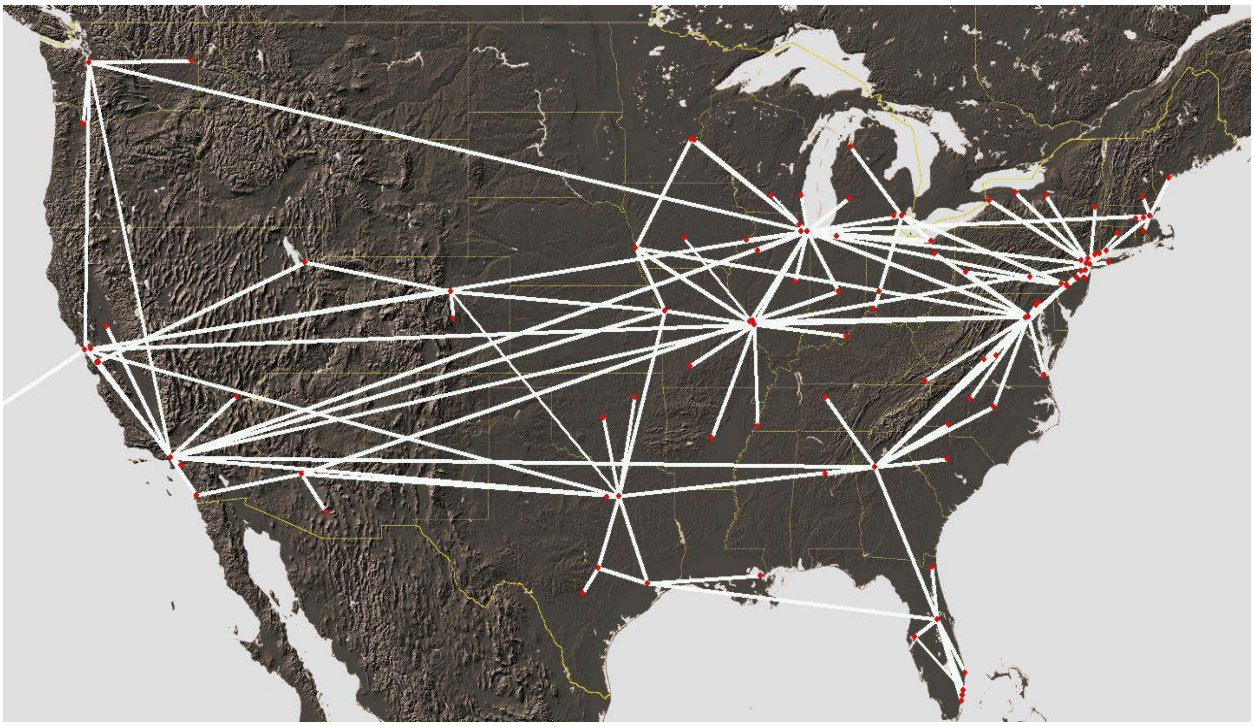


Figure 4.10. United States AT&T router backbone (layer 0) taken from (Spring, Mahajan, and Wetherall 2002), Image from: NASA's Visible Earth Project http://visibleearth.nasa.gov

G. Research CPN Model and Simulation Execution

The CPN syntax was introduced earlier in this chapter. The main CPN page of this simulation as depicted in Figure 4.11 represents the overall high level implementation summary. All sub-pages interact with the main page through the standard input/output ports provided in CPN Tools. The main page controls the overall flow of the simulation and the evaluation of each node-pair. For reference and clarity purposes, all transitions and places presented are assigned reference numbers. All CPN function code and color data declarations can be found in appendices A and B of this dissertation. This section presents the CPN main page and an overview of the model and simulation execution. The CPN sub-pages associated with the main page and a more detailed level of granularity can be found in Appendix D and Appendix E.

1) CPN Main Page.

As shown in Figure 4.11, transitions A1, A2, and A3 fire to initialize the CPN list tokens found on places 4, 2, 3 respectively. All data structures were loaded with the pre-attack node data. This occurs during the first few clock ticks of the simulation. The timing of all CPN actions was controlled by the CPN simulation engine. Place 1 triggered transition B to fire and transition B created a list of pre-attack network connectivity node records (networkDB). The list was stored on place 6. This list of node records was implemented using CPN record structures. During the simulation, the networkDB list was updated with the current node states. The networkDB was used in transition D1 to evaluate each currently released node-pair.

Transition C randomly designated critical nodes to be transformed into an attacked node from the critical nodes list on place 3. As previously discussed in Chapter II, attacked nodes are removed from all node communications, and they begin the cascaded failures of their children nodes. Transition C updates the network DB record for critical nodes randomly designated for attack. The transition also added the attacked node to the set of removed nodes on place 7. This process continued throughout the simulation until all the critical nodes were attacked. Node states and critical nodes were previously defined in Chapter III.

The CPN list token on place 6 represented a list of records where each record (tuple) represented one node's connectivity state at time. The node tuples were defined in Chapter III, Section A. Continuously, at time t, as the simulation proceeded, transition D2 was randomly fired to release node-pairs into the simulation stream. The release of node-pair tokens into the simulation stream represented a single communication attempt between two

adjacent routers. When a node-pair was released it was added to place 5. Eventually it triggered the random firing of transition D1.

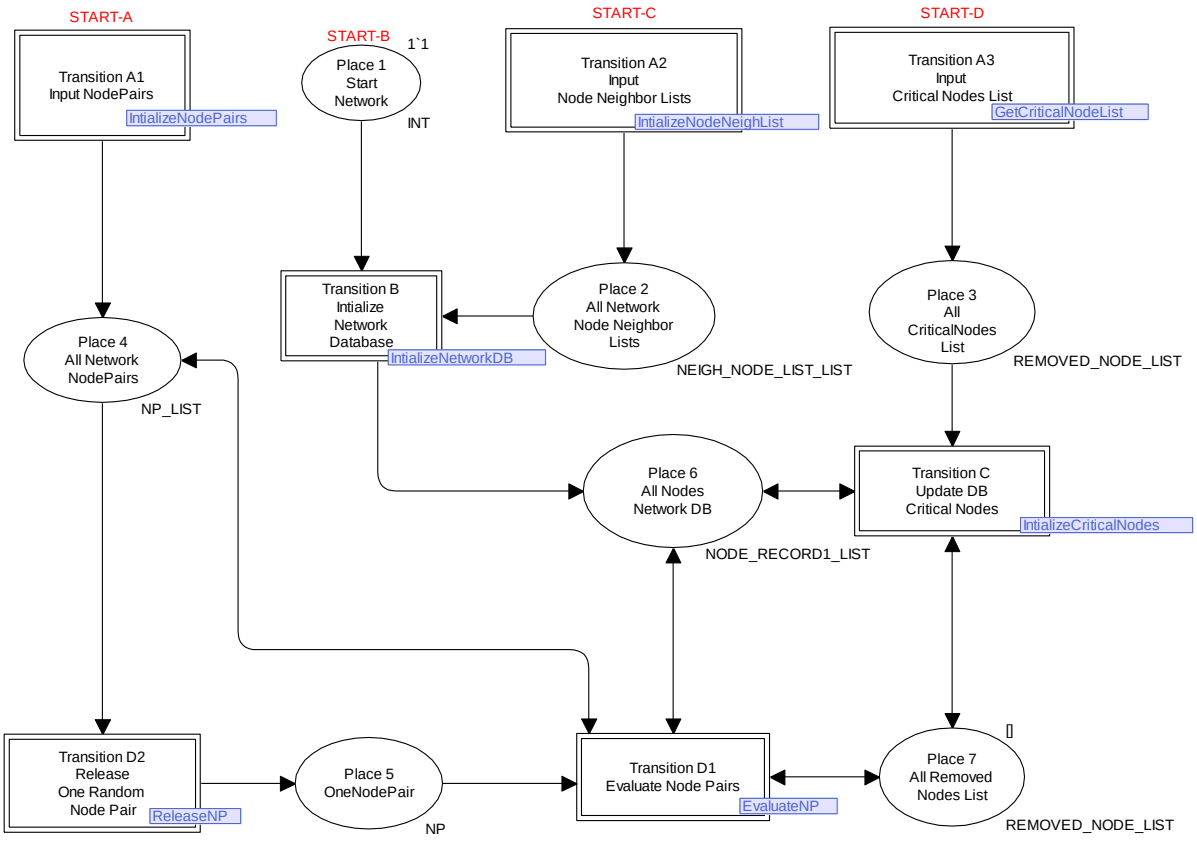


Figure 4.11. CPN main page

As node-pairs were available on place 5, each node-pair was triggered transition D1 to fire. It executed a set of functions that evaluated each released node-pair. This process was previously defined in Chapter III. As each transition in Figure 4.11 fired, the associated sub-page took control of simulation execution. Each sub-page associated with a specific transition rectangle is denoted on the transition in Figure 4.11. For example, when transition D2 is enabled, it will shift control to sub-page "ReleaseNP." Table 4.4 depicts the sub-pages that interact with each transition on the main page shown in Figure 4.11.

Table 4.4. Main CPN page transitions and associated sub-pages

Transition that Fires	*Control taken by sub-page*
A1	IntializeNodePairs
A2	IntializeNodeNeighList
A3	GetCriticalNodeList
B	IntializeNetworkDB
C	IntializeCriticalNodes
D1	EvaluateNP
D2	ReleaseNP

2) Core Simulation Evaluation Algorithm.

Node states for each node will vary over the life of the simulation run. At any given time, whenever a node-pair is released into the simulation stream, one of the two nodes may potentially be an orphan node. The simulation stream is a set of processes that are used to evaluate a node-pair and determine whether one of the nodes has been orphaned due to a cascaded failure of its partner. Network connectivity states change as the attack proceeds until the halting conditions are encountered. The halting conditions and network connectivity metrics was previously defined in Chapter II. The pseudo-code for the simulation is:

```
BEGIN       /* Start simulation */
Initialize node-pair set;            /* Transition A1 */
Initialize CPN data records;         /* Transition B */
Initialize node neighbor set;        /* Transition A2 */
Initialize Critical nodes set;       /* Transition A3 */

/*  Process 1 – Runs in parallel with Process 2*/
DO UNTIL Number of attack nodes = 0
    Randomly designate one attacked node from critical node list;   /* Transition
    C */
    Add to removed nodes set;          /* Place 7 */
LOOP

/*  Process 2 – Simulation Stream - Runs in parallel with Process 1*/
DO UNTIL halting condition
    Release one node-pair at random;        /* Transition D2 */
            /* Start Transition D1 */
            Evaluate node-pair against removed nodes set;
            If orphan node then
                    BEGIN
                            Update orphan set;
                            Add orphan to removed nodes set;
                            Update network connectivity state;
                    END
            else
                    do nothing;
            END IF
            /* End Transition D1 */
LOOP
END.
```

3) CPN Main Page Flow.

Table 4.5 depicts a summary of the CPN places in Figure 4.11. The tokens that move between places carry one of the following structures: (1) node-pairs, (2) CPN list of network-DB records representing the current node states, and (3) nodes.

Table 4.5. Summary of CPN places in Figure 4.11

Reference in Figure 4.11	*Description*
Place-1	Token used to start the simulation.
Place-2	A list of list structures holding the neighbor node relationships as described in detail in the appendix.
Place-3	The set of critical nodes, previously defined as $N_w(t = 0)^{Critical\ (C_z)}$.
Place-4	The set of all node-pairs, previously defined as $P(t)^{Active}$.
Place-5	One released node-pair for evaluation, previously defined as $(n_i(t), n_j(t))$.
Place-6	Current state of the network represented as a list of networkDB records, one per node. Defined as [<node-tuple1>, <node-tuple2>, <node-tuple3>,…. <node-tupleN>], where <node-tupleN> is defined as <node-id, node-status, [node-neighbor list], timestamp, simulation real time>. [..] denotes a CPN list.
Place-7	The set of all removed nodes, previously defined as $N(t)^R$.

Figure 4.12 depicts the functionality of the transition D1 denoted Figure 4.11. It is responsible for implementing process 2 in the previous paragraph. The token flow in this transition is controlled as follows:

1. As depicted by the annotation box labeled "A," as the node-pair was released into the simulation stream, if one of the nodes (for example, NODE2) was a member of the removed nodes set (Place 7), then the other node (NODE1) was transformed into a temporary orphan.

2. At annotation “A,” if neither node was a member of the set of removed nodes (Place 7) then the node-pair was returned to the set of all node-pairs (Place 4).
3. At annotation box labeled “B,” NODE1 attempted to establish a new link with an available neighbor node through preferential attachment mechanisms. In this figure, the new node is NODE3.
4. At annotation box labeled “C,” if link load of NODE3 did not exceed its link capacity then the new node-pair was added to the set of node-pairs (Place 4).
5. At annotation “C,” else if NODE3 link capacity was exceeded then the communication failed and NODE3 was transformed into an overloaded orphan. It was then added to the set of removed nodes (Place 7). NODE1 executed steps 2 and 3 above until a successful node-pair was established.
6. At annotation “B,” if no more neighbors existed for NODE1 in step 5 then NODE1 was transformed into a null-link orphan and it was added to the set of removed nodes (Place 7).

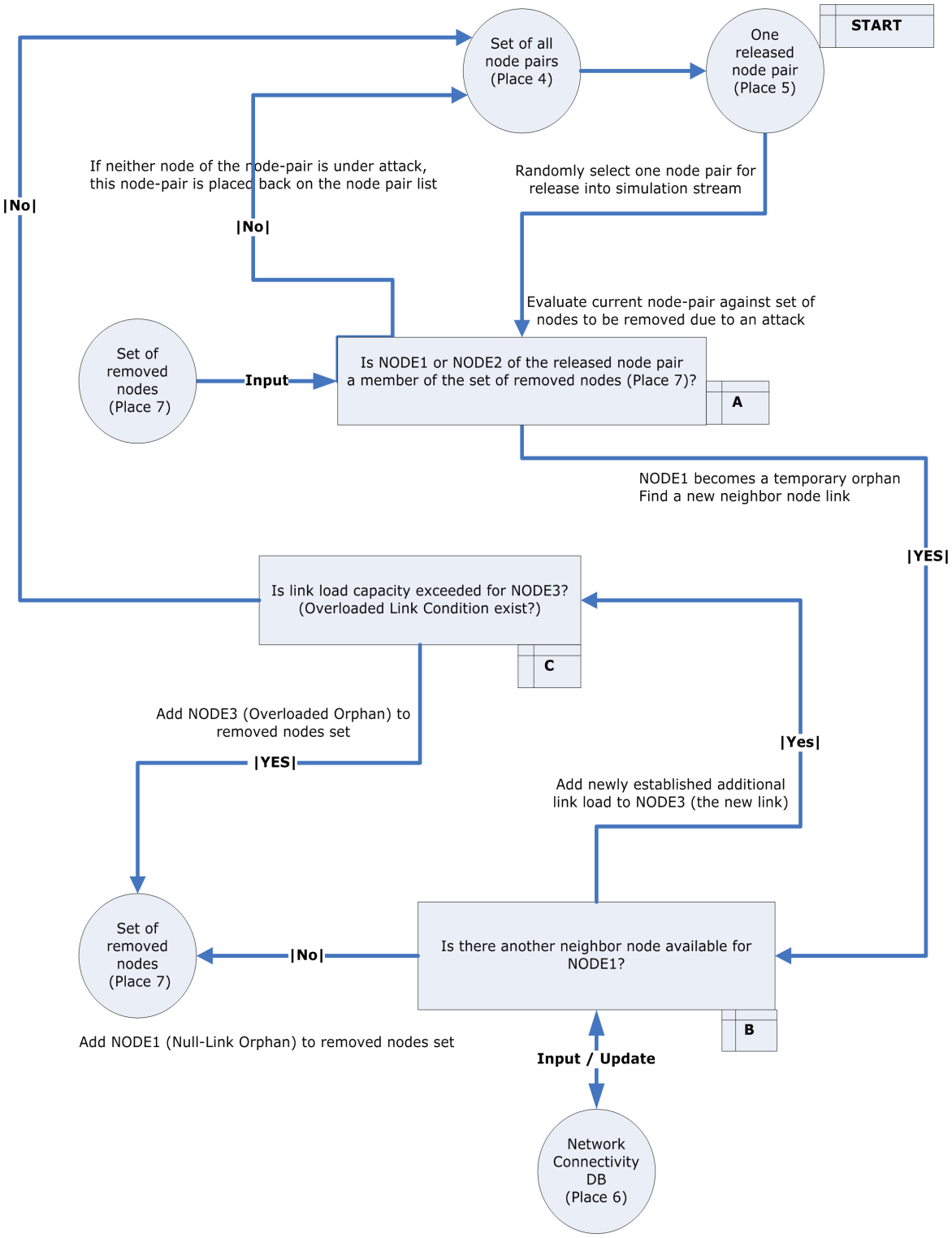


Figure 4.12. CPN main page flow functionality

All CPN data type declarations are available in Appendix A and CPN function code in Appendix B. This section has presented the top layer of the CPN model and simulation execution algorithms. Simulation code and detailed description of the CPN sub-pages discussed here can be found in Appendix A and Appendix B.

In this chapter the model and simulation design used in this research was introduced. This chapter also covered the research simulation run strategies, data collection methodology and the foundations. The results of the simulation runs defined in this chapter will now be presented in Chapter V.

CHAPTER V. RESULTS

This chapter will present the execution results of the 40 simulation runs performed in this research. For each of 4 run types, this research executed 10 simulations, one for every attack class. Table 4.1 defined the simulation's execution parameters. Table 4.2 defined the attack scenarios for the attack classes. The attack modeling and simulation techniques used in this section were previously discussed in Chapter IV. As discussed earlier in Chapter II, network stability was measured using information transfer (I) and the network connectivity parameter (NCP). The terms *terminal conditions*, *equilibrium* and *critical threshold* used later in this chapter were defined previously in Chapter I. Data were collected in 50-second time intervals to highlight significant change in network stability as it related to the research objectives. Simulation time represented real clock time.

As previously discussed in Section C of Chapter II, error and attack studies indicated that node removals targeted at a scale-free network's most connected nodes led to systemic degradation in network stability. The literature indicates that the extent of this degradation was dependent upon the attack's severity (Albert, Jeong, and Barabasi 2000; Crucitti et al. 2004; Guillaume, Latapy, and Magnien 2005; Lai, Motter, and Nishikawa 2004; Salla 2005; Wang et al. 2008). For this research, as shown in Table 4.1, attack severity was defined from least to most severe attack classes as follows: 0.5%, 1.0%, 1.5%, …, 5.0%. The first section in this chapter will show that the model and simulation were consistent with the relevant literature. The next section will establish the reliability of the network stability results. This chapter concludes by presenting data that support the foundational objectives of this research investigation.

A. Model and Simulation Validation and Consistency

The simulation design was validated using 10 run type 1 simulations. Each simulation collected data at 50-second real time intervals. This research investigation compared the results of these executions with earlier research studies of scale-free computer networks. This section will consider scale-free computer network: 1) error and attack studies, 2) fragmentation, 3) communication robustness, and 4) heterogeneous linking characteristics.

1) Attack Severity – Error and Attack Studies.

This section will validate run type 1 simulation results against foundational error and attack studies found in the literature. For each simulation run a critical threshold was encountered. After this critical threshold there was a sudden and rapid decrease in network stability. The existence of a critical threshold was supported in the scale-free computer network error and attack studies found in the literature (Barabasi and Albert 2002; Cohen 2000, 2001; Dorogovtsev and Mendes 2002; Lopez 2007). The supporting evidence for this assertion will be presented later in this chapter.

By varying the proportion of nodes removed during each simulation, this research was able to systemically reduce the network connectivity stability. These results are shown in Figure 5.1 and Figure 5.2 and were consistent with previous error and attack studies (Albert, Jeong, and Barabasi 2000; Crucitti et al. 2004; Guillaume, Latapy, and Magnien 2005; Lai, Motter, and Nishikawa 2004; Salla 2005; Wang et al. 2008). As shown in Figure 5.1, Figure 5.2, and Figure 5.3, the fraction of nodes removed was represented as: $f = \frac{N_t^R}{N_{t=0}^{Active}}$; where N_t^R and $N_{t=0}^{Active}$ were previously defined in Chapter III, Section A. As discussed earlier, N_t^R reflected both nodes removed directly by the targeted attack as well as

nodes that were subsequently orphaned by cascaded failures. For reader clarity, the dotted line in Figure 5.2 represents the previously cited NCP ≅ 2 cutoff.

Figure 5.1 and Figure 5.2 depicts changes in network stability relative to changes in the proportion of nodes removed during the attack simulations. As shown in Figure 5.1 and Figure 5.2, for each independent attack class the values of I and NCP decreased with an increase in the fraction of nodes removed. As the fraction of nodes removed increased, attack classes 0.5% through 4.0% established a local minimum at varying levels of I and NCP. This local minimum was influenced by the attack severity. The local minimums for each attack class shown in Figure 5.1 and Figure 5.2 decreased as the relative attack severity increased. As shown in Figure 5.1 and Figure 5.2, attack classes 4.5% and 5.0% did not establish a local minimum. Instead, both attack classes decreased at a relatively steady rate towards the terminal conditions. The terminal conditions for these two attack classes was achieved at $f \cong 0.70$.

Figure 5.3 depicts the change in the fraction of removed nodes for all 10 simulation runs. Independent of attack class, it shows that the fraction of nodes removed increases as a function of the simulation time. The rate and magnitude of this increase was influenced by attack class severity. The fraction of all nodes removed and the rate of removal increased as the attack class severity increased. As depicted in Figure 5.3, the rapid increase in the first 200 seconds was consistent with results that will be presented later in this chapter.

The local minimums established in Figure 5.1 and Figure 5.2 was consistent with the sudden rate change shown in region A on Figure 5.3. For attack classes 0.5% through 4.0% shown in Figure 5.1 and Figure 5.2, after the local minimum was established, I and NCP remained relatively constant with an increase in the fraction of nodes removed. This trend

continued until each attack class encountered a critical threshold at $f \cong 0.95$. At $f \cong 0.95$, I and NCP suddenly and rapidly decreased towards the terminal conditions. Terminal conditions were previously discussed in Chapter III. The critical thresholds found in Figure 5.1 and Figure 5.2 was consistent with the critical thresholds presented later in this chapter.

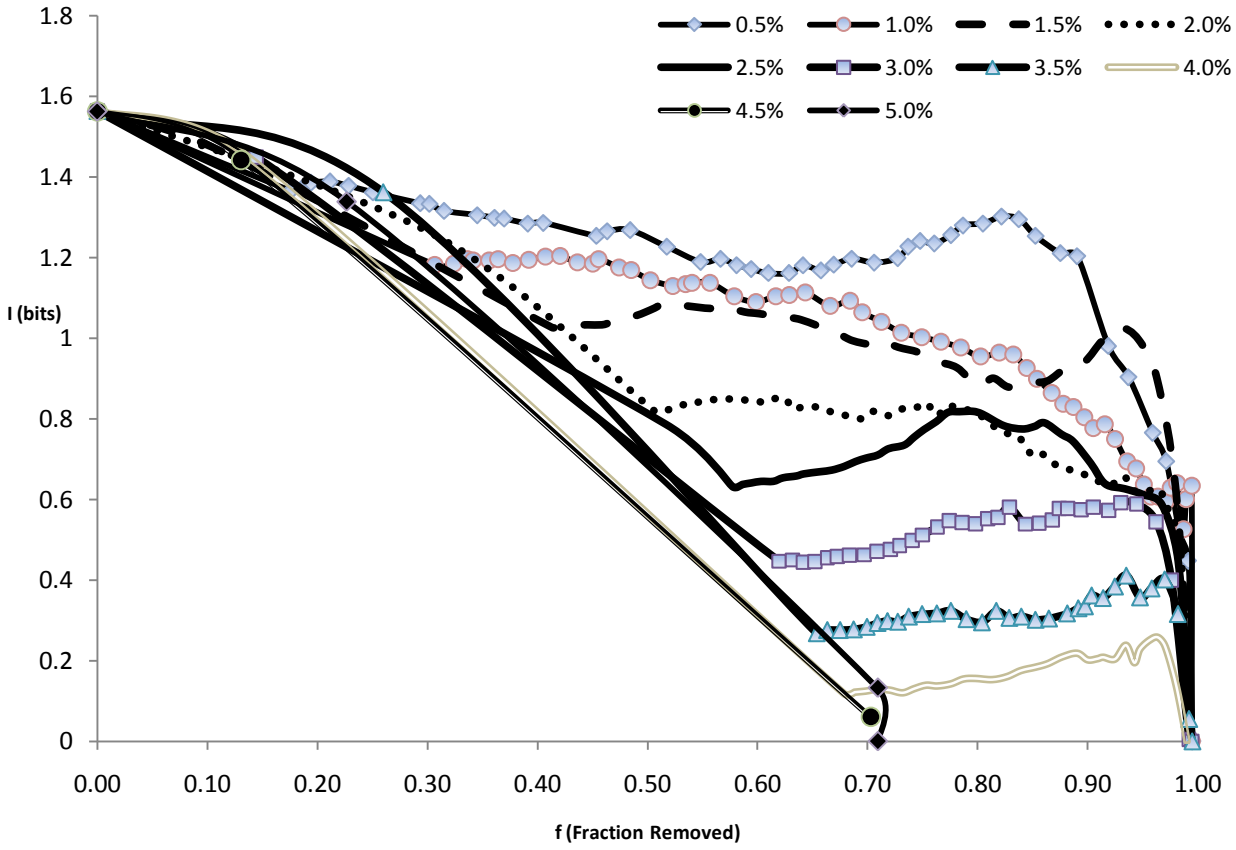


Figure 5.1. Run type 1, information transfer versus nodes removed fraction over time

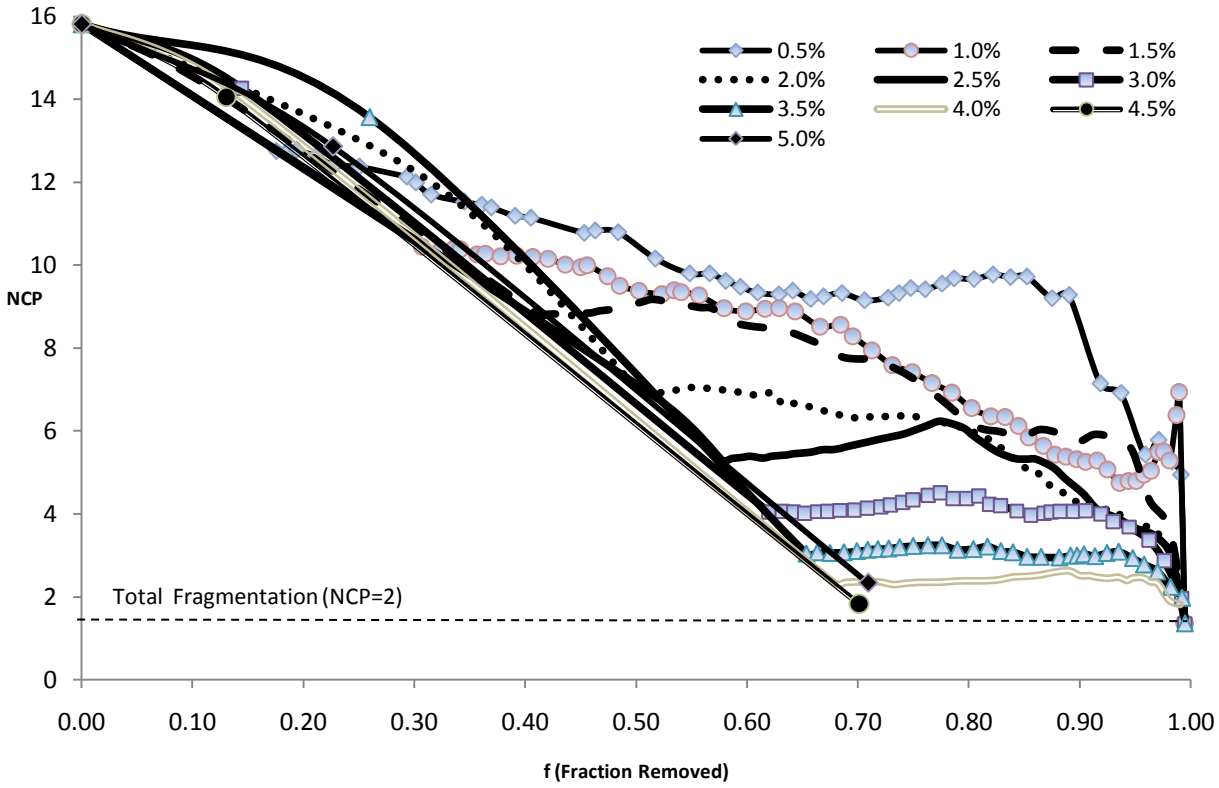


Figure 5.2. Run type 1, connectivity parameter versus nodes removed fraction over time

Over the 10 simulations studied, on average approximately 2.85% of the total removed nodes were nodes that were directly targeted for removal. As previously defined in Chapter III, the total number of removed nodes is the sum of all nodes permanently orphaned plus all nodes targeted for attack; $N(t)^R = N(t)^O + N(t)^A$. Therefore, the nodes removed in Figure 5.1, Figure 5.2, and Figure 5.3 primarily reflect network changes due to cascading node failures. Permanent orphans, attacked nodes, and cascaded failures were previously defined in Section A of Chapter III. Cascaded nodes were previously defined as permanent orphans ($N(t)^O$).

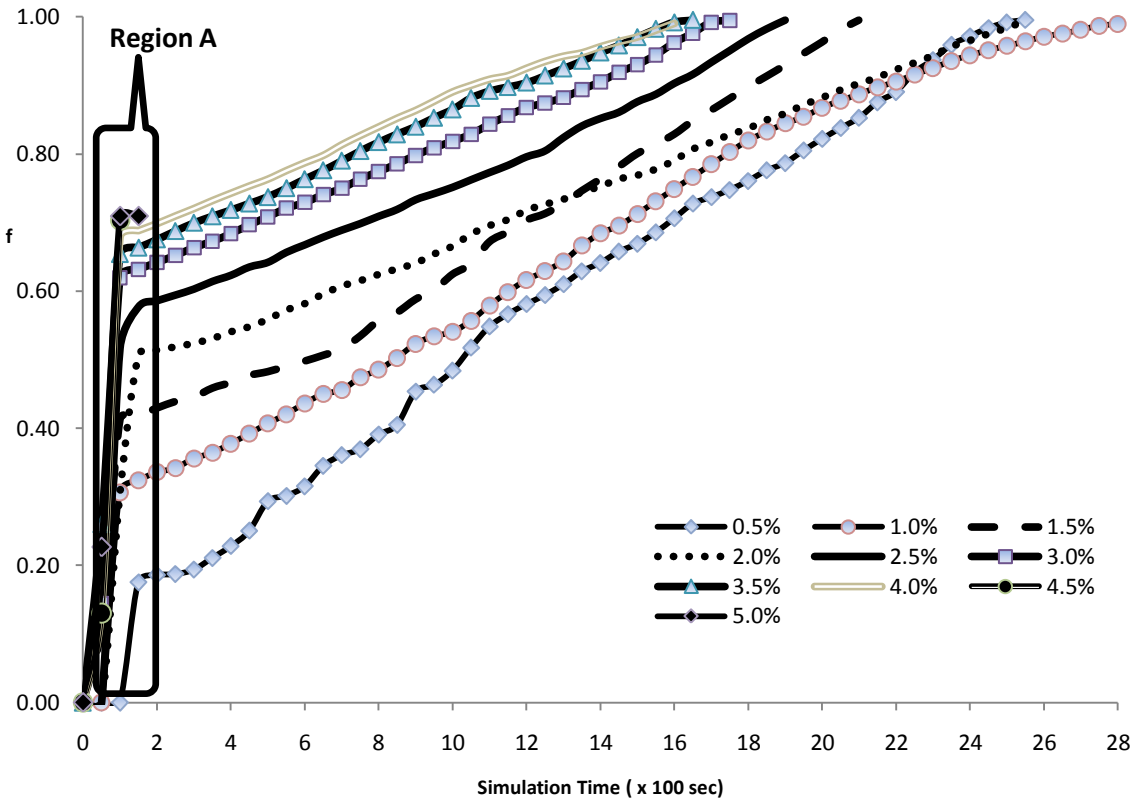


Figure 5.3. Run type 1, fraction of all nodes removed versus simulation time

The change in network stability shown in Figure 5.1, Figure 5.2, and Figure 5.3 were consistent with previous scale-free attack studies (Albert, Jeong, and Barabasi 2000; Crucitti et al. 2004; Guillaume, Latapy, and Magnien 2005a; Lai, Motter, and Nishikawa 2004; Salla 2005; Wang et al. 2008). These cascaded node failure trends were similar to those found by other researchers (Cohen 2001; Huang and Li 2007; Motter 2004; Motter and Lai 2002; Wu and Fang 2008). The next section will cover another characteristic of scale-free networks under attack, network fragmentation.

2) Network Fragmentation.

For the 10 run type 1 simulations executed, Figure 5.4 depicts evidence of increasing network fragmentation. As previously discussed in Chapter II, relative network

fragmentation can be measured using the network connectivity parameter (NCP). Higher NCP values indicate less fragmentation. Equation (46) in Chapter III defines the network connectivity parameter (NCP). Each attack class experienced similar rapid fragmentation in the first 150 seconds. Attack classes 4.5% and 5.0% rapidly and completely fragmented into isolated clusters in the first 150 seconds.

As shown in Figure 5.4, all run type 1 attack classes experienced a sudden decline in network fragmentation in the first 100 seconds. The information transfer for attack classes 4.5% and 5.0 % rapidly declined and met their terminal conditions in the first 150 seconds. The information transfer decrease in the first 100 seconds experienced by all attack classes was influenced by the relative attack severity. Over the first 100 seconds, the rate and magnitude of network fragmentation increased with an increase in attack severity.

Figure 5.4 depicts that attack classes 0.5% through 4.0% established a local minimum value at approximately 200 seconds. This local minimum value decreased with an increase in attack severity. After the local minimum was established, the network fragmentation level for each attack class remained relatively constant over time. This level was maintained until each attack class encountered its critical threshold. The attack severity influence on the critical threshold time for attack classes 0.5% through 2.0% varied. Attack classes 2.5% through 4.0% achieved their critical threshold earliest. The critical threshold time increased with an increase in attack severity for attack classes 2.5% through 4.0%. The most severe attack classes, 4.5% and 5.0%, did not project a critical threshold or local minimum.

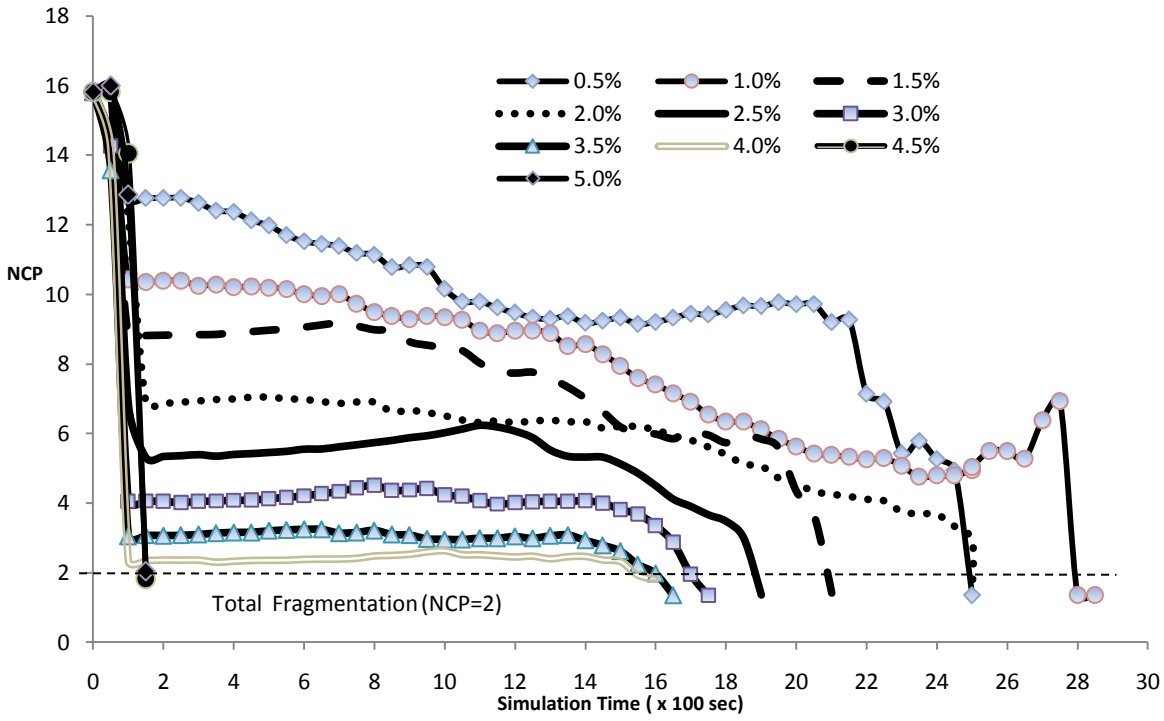


Figure 5.4. Run Type 1, network connectivity parameter vs simulation time

These results showed evidence of network fragmentation during the attack simulations. This discovery was consistent with the literature (Albert, Jeong, and Barabasi 2000; Barabasi and Albert 2002; Cohen 2000, 2001; Crucitti et al. 2004; Wang, Guan, and Lai 2009). The next section will cover the loss of heterogeneity that has been observed during a connectivity attack that results in scale-free network communication destabilization.

3) Loss of Heterogeneous Linking.

One way to study network stability is through its heterogeneous linking behaviors. It has been shown that as a scale-free network's heterogeneous connectivity decreases, its communication robustness also decreases (Crucitti, Latora, and Marchiori 2004; Demetrius and Manke 2005; Hu and Wang 2008; Sanchirico and Fiorentino 2008; Wang et al. 2006; Wu and Fang 2008). Loss of heterogeneity and robustness can be measured indirectly

through changes in information transfer and entropy. This research studied variations in the network's information transfer and information entropy during the simulated attacks. This section will present these results and their validity.

a) Mutual Information Transfer Loss.

As previously discussed, assortativity is an indirect measure of a network's heterogeneity. Scale-free and heterogeneous networks tend to be disassortative ($r < 0$) (Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). It has been shown that positive information transfer values indicate assortative connectivity ($r \geq 0$) (Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). It has also been shown in the literature that assortative connectivity is not heterogeneous and robust (Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). The results presented here showed that the information transfer decreased significantly towards zero for all of the 10 run type 1 simulations. During the simulated attacks, as the network connectivity changed the information transfer decreased towards zero. It has been shown in the literature that this behavior is indicative of a network with decreasing heterogeneity (Piraveenan, Prokopenko, and Zomaya 2009; Sole and Valverde 2004). Results trends that support this assertion will be presented later in this section.

b) Information Entropy Loss.

Information entropy was previously defined in Section D of Chapter III. Network heterogeneity can be measured through its information entropy (Demetrius and Manke 2005; Gudkov and Montealegre 2008; Wang et al. 2006). Researchers have found that the heterogeneity decreases with a decrease in entropy (Gudkov and Montealegre 2008; Wang et al. 2006). As shown in Figure 5.5, the combined data of these 10 simulations were used to

compute the average entropy and the average <k> for all run type 1 simulations. The data were evaluated in 50-second time intervals.

Figure 5.5 depicts the average <k> plotted against its corresponding average entropy for each run type 1 simulation. It shows a direct polynomial relationship between the simulated average <k> and its average entropy. The relationship was statistically significant at $\alpha = 0.01$. Table 5.1 depicts the change in entropy that was found at the terminal condition for all run type 1 simulations. As shown in Table 5.1, the average decline in entropy over the life of the simulation was 52.4%. Figure 5.5 and Table 5.1 were consistent with Figure 2 found in Wang et al. (Wang et al. 2006).

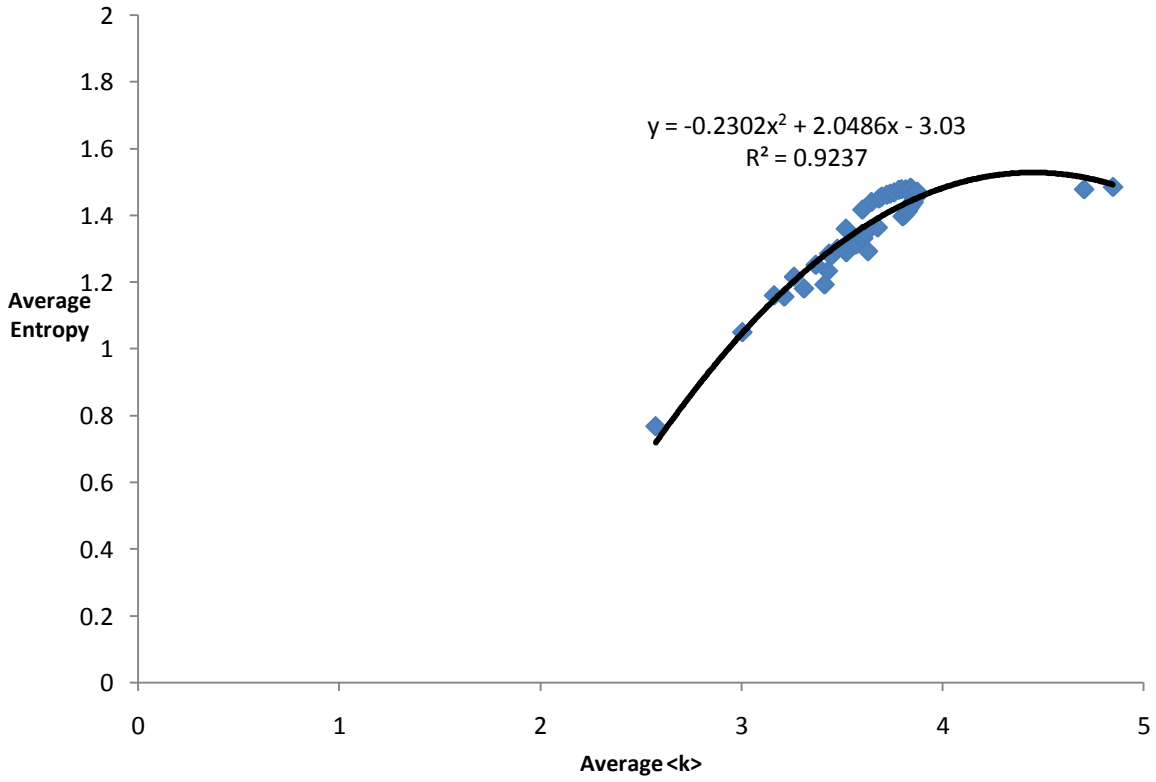


Figure 5.5. Run type 1, avg. entropy versus avg. degree, combined runs, 50-second intervals

Table 5.1. Run type 1 simulations, entropy at total collapse at I = 0

Attack Class	*Entropy at Terminal Condition (I = 0)*	*Percent Entropy Change from Pre-Attack Entropy of 1.48*
0.5%	0.60	-59.5%
1.0%	0.65	-56.1%
1.5%	0.70	-52.7%
2.0%	0.69	-53.4%
2.5%	0.60	-59.5%
3.0%	0.59	-60.1%
3.5%	0.60	-59.5%
4.0%	0.85	-42.6%
4.5%	0.95	-35.8%
5.0%	0.87	-41.2%
10 Run	0.71	-52.04%

As previously discussed in Section C of Chapter II, during a targeted node removal attack, the literature indicates that as the network fragments over time the characteristic scale-free heterogeneity diminishes. This can be reflected as a decrease in the network connectivity parameter and entropy. The combined results of all run type 1 simulations indicated 1) a Pearson correlation between NCP and the information entropy of 0.48 and 2) a Pearson correlation of -0.69 between the information entropy and simulation execution time. Both correlations were statistically significant at the $\alpha = 0.1$ level. Over the life of each simulation, information entropy decreased as the network fragmented. This relationship was consistent with the literature (Cohen 2001; Crucitti, Latora, and Marchiori 2004; Demetrius and Manke 2005; Gudkov and Montealegre 2008; Motter and Lai 2002; Wang et al. 2006) .

This research has provided evidence that the information entropy, information transfer, and the network connectivity parameter simulation data were consistent with the relevant scale-free network literature. The information entropy, information transfer, and network connectivity decreased over the life of each attack simulation. As previously

discussed in Chapter II, the literature indicates that these behaviors are indicative of a scale-free network undergoing a significant loss of heterogeneity, communication robustness and stability.

B. The Relationship between NCP and I

As previously discussed in Chapter III, information transfer (I) and the network connectivity parameter (NCP) were used in this research to monitor the network's connectivity stability. The relationship between the network connectivity parameter and information transfer is presented in Figure 5.6. This relationship exhibited a strong positive Pearson correlation of 0.90. The relationship was statistically significant at $\alpha = 0.1$. Since this research has uncovered a strong correlation between information transfer and the network connectivity parameter, for the remainder of this dissertation the discussion will be limited to information transfer only. Additional network connectivity parameter data can be found in Appendix E, Appendix F and Appendix G. The next section will present the foundational investigation of node-pair relationships with this network performance degradation.

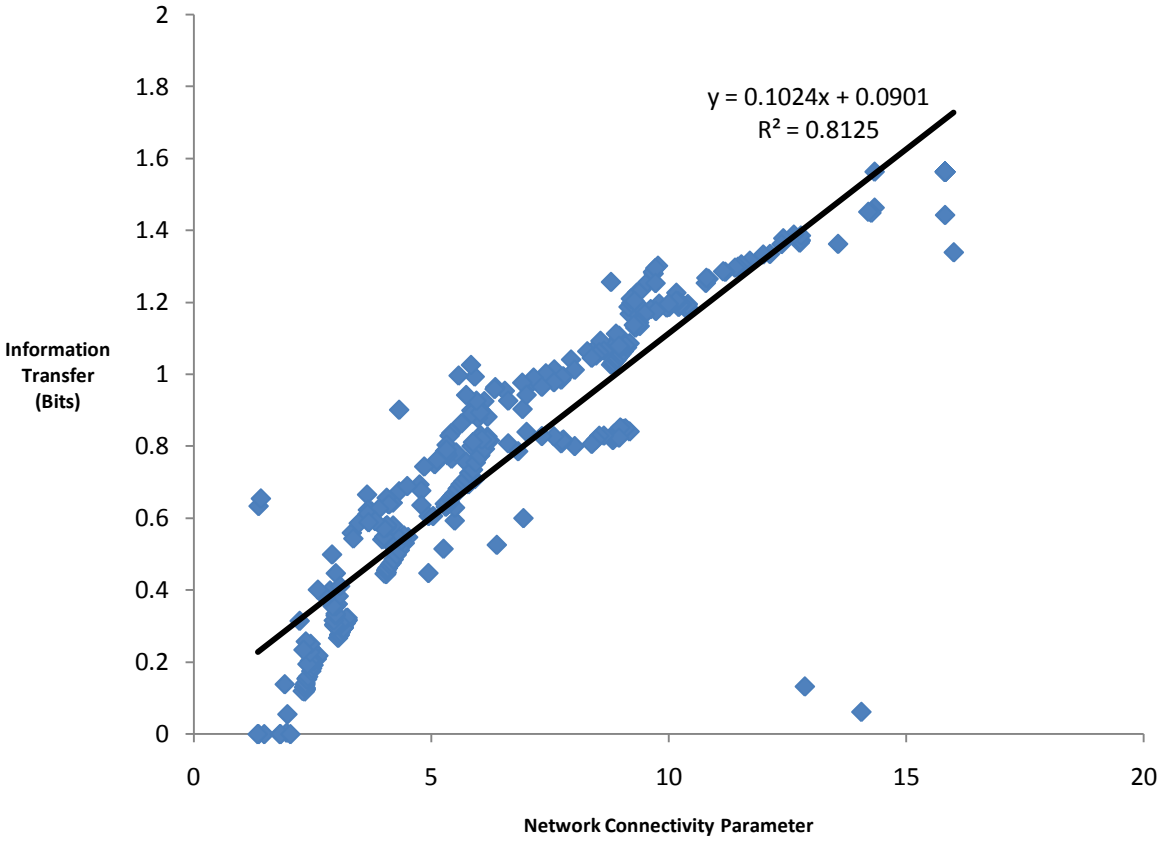


Figure 5.6. Run type 1, NCP and information transfer, 10 combined simulations

C. Node-pair and Information Transfer Correlations

The relationship between node-pair types and network stability will be presented in this Section. The combined results of 10 run type 1 simulations were utilized to compute Pearson correlations between information transfer and the number of node-pairs by type. The correlations were studied in 50-second time intervals over the first 500 seconds. This research considered the relative strength of these correlations as a means to select specific node-pair-types for further study. R^2 of 0.8 indicated that the linear fit was significant.

During the first 500 seconds of the combined run type 1 simulations, node-pair types that demonstrated potential to achieve the research objectives met these criteria: 1) a positive

or negative correlation greater than 0.8 and 2) the relationship was statistical significant at the α=0.1 level. This correlation cutoff provided a significantly strong enough correlation to distinguish emergent patterns in the node-pair type counts.

Node-pair types represent the state of the network at a specific time. Information transfer was previously shown in Chapter II to represent the stability of the network connectivity. As the information transfer decreases, the network-wide connectivity stability decreases, and the amount of information uncertainty between two randomly selected nodes increased. If the number of certain node-pairs is strongly correlated with changes in the information transfer, then the number of these node-pairs might indicate an attack's existence.

Table 5.2 and Table 5.3 depict all correlations found above the correlation cutoff. Both tables were sorted in decreasing order of correlation strength. Table 5.2 depicts node-pair type correlations with a negative correlation with the information transfer (I). Table 5.3 presents node-pair type correlations with a positive correlation with the information transfer.

Table 5.2. Run type 1, potential attack markers for attack detection, node-pair counts having strong negative correlation with information transfer for the first 500 seconds

Node-pair Type	*Degree of node1 in node-pair*	*Degree of node2 in node-pair*	*Counts having Strong Negative Pearson Correlation with I*
1-1	1	1	-0.902
4-4	4	4	-0.832
1-56	1	56	-0.821

Table 5.3. Run type 1, potential attack markers for network protection, node-pair counts having strong positive correlation with information transfer for the first 500 seconds

Node-pair Type	*Degree of node1 in node-pair*	*Degree of node2 in node-pair*	*Counts having Strong Positive Pearson Correlation with I*
4-54	4	54	0.988
4-55	4	55	0.988
4-40	4	40	0.987
4-31	4	31	0.965
4-51	4	51	0.963
4-48	4	48	0.957
2-17	2	17	0.928
2-2	2	2	0.927
3-60	3	60	0.923
1-2	1	2	0.913
3-18	3	18	0.911
3-21	3	21	0.909
2-7	2	7	0.905
3-17	3	17	0.902
2-15	2	25	0.890
2-24	2	24	0.887
1-17	1	17	0.869
2-22	2	22	0.857
1-3	1	3	0.845
3-29	3	29	0.842
3-65	3	65	0.838
4-21	4	21	0.802

This Pearson correlation analysis precipitated the selection of significant node-pair types for further analysis, specifically node-pair type 1-2. Chapter VI will discuss the selection criteria further. The correlation results were foundational in the development of the simulation specifications for run types 2, 3, and 4. Run types 2, 3, and 4 simulation

results will be addressed later in this chapter. The next section will present the relationship of the node-pair type 1-2 counts, attack class severity, and the protection schemes developed in this research.

D. Network Stability by Run Type

Variations in information transfer and the number of node-pairs of type 1-2 over the life of each simulation are covered by this section. The research results were derived from 40 simulation runs, 10 simulations for each of four run types. Run types were previously defined in Table 4.2. The data were collected in 50-second time intervals over the life of each simulation. The simulations used in this research were halted after 25,000 execution seconds. Node-pair type counts were a determinant used in this feasibility study. The next section will introduce the meaning of these node-pair types.

1) Node-pair Types.

This research classified all active node-pairs into groups by node degree composition. This classification was performed for each simulation at 50-second time intervals. These groups were called node-pair types. Each node-pair type was distinguished by a combination of the degree of each node in the pairing. For example, if one node in a node-pair had a degree of 1 and the other node had a degree of 2, then the node-pair type was classified as type 1-2. The syntax for each node-pair type designation is: 1) position 1 represents the node degree of the first node ($n_i(t)$) in the node-pair, 2) position 2 represents the node degree of the second node ($n_j(t)$) in the node-pair, and 3) position 1 in the node-pair type was always be greater than or equal to position 2. The first ($n_i(t)$) and second node ($n_j(t)$) in the node-pair were previously defined in Section A of Chapter III.

During the attack simulation, as node-pairs were destroyed and new node-pairs were established, the count of each node-pair type changed. These changes were monitored with changes in the overall network's connectivity stability. During each simulation, this research studied variations in the number of node-pairs of type 1-2 with changes in the information transfer.

The influence of attack severity on the information transfer data shown in Figure 5.7 and Figure 5.9 were consistent with the node-pair loss shown in Figure 5.8 and Figure 5.10, respectively. Therefore attack severity influence discourse found in this chapter will be limited to Figure 5.7 and Figure 5.9. The terminal conditions and the equilibrium points established in Figure 5.7 and Figure 5.9 were also consistent with the node-pair loss shown in Figure 5.8 and Figure 5.10, respectively. Therefore, future discourse on terminal condition and equilibrium point behaviors will be limited to Figure 5.7 and Figure 5.9.

Further data representing variations in the information transfer, network connectivity parameter and node-pair type 1-2 can be found in appendix. These data are available by run type and attack class. The attack simulation results presented in the remainder of this chapter will establish foundational evidence that the network's connectivity stability was related to the composition of its node-pairs.

Run type 2 and 3 simulation runs were determined to be transitional results leading to the discoveries found in the run type 4 results. Therefore, the results of run types 2 and 3 can be found in Appendix H and Appendix I. Run types 1 and 4 were found to present trends that merit further discourse. The remainder of this section will present the results for run type 1 and run type 4.

2) Run Type 1.

Ten run type 1 simulations are presented in this section. These simulations represented the affects of denial-of-service attacks against the pre-attack network. These simulations did not have additional node protection. For each attack class, Figure 5.7 represents changes in information transfer over the life of the simulation. Figure 5.8 depicts the corresponding change in the number of node-pairs of type 1-2 for the same simulations shown in Figure 5.7.

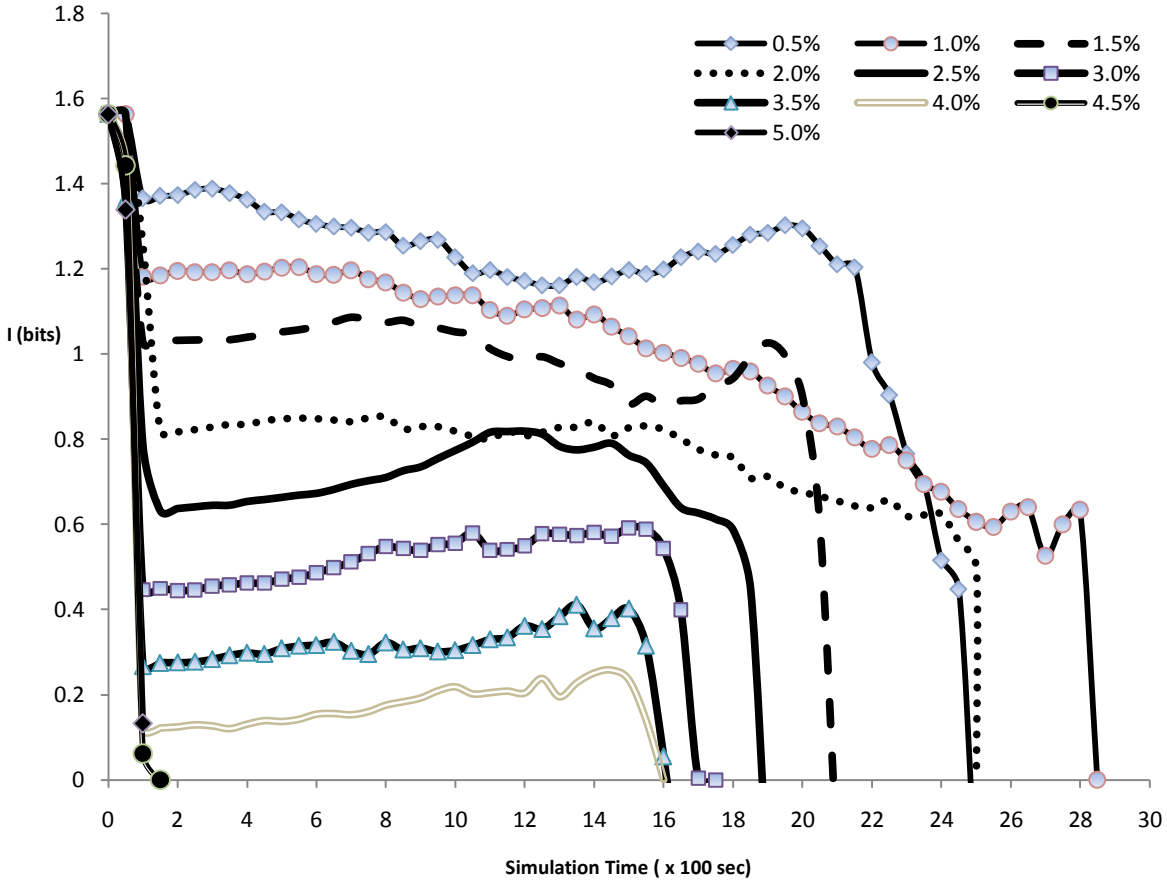


Figure 5.7. Run type 1, network stability, information transfer versus time

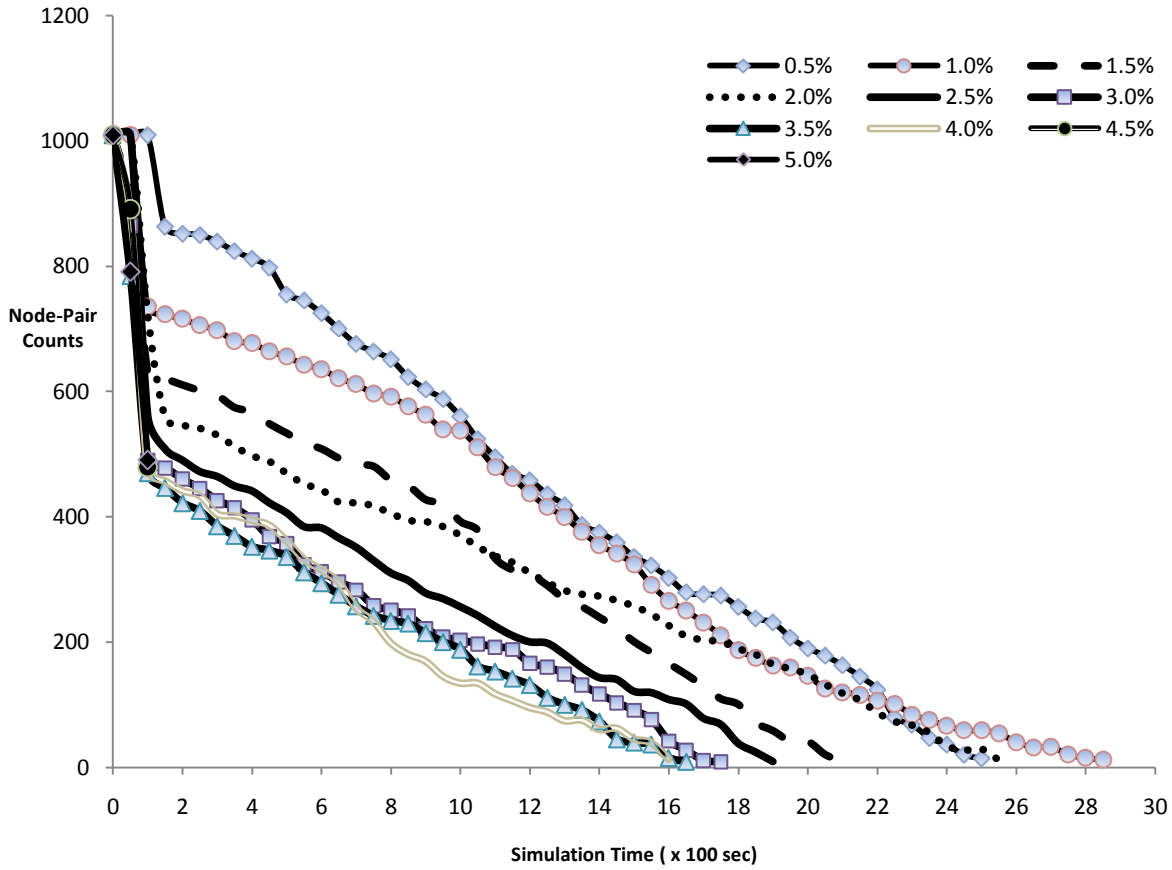


Figure 5.8. Run type 1, node-pair type 1-2 counts versus time

a) Behaviors in the First 200 Seconds.

As shown in Figure 5.7, all run type 1 attack classes experienced a sudden decline in information transfer in the first 100 seconds. The information transfer for attack classes 4.5% and 5.0 % rapidly declined and met their terminal conditions in the first 150 seconds. The information transfer decrease in the first 100 seconds experienced by all attack classes was influenced by the relative attack severity. Over the first 100 seconds, the rate and magnitude of the information transfer loss increased with an increase in attack severity. However, the rate and magnitude of the information transfer loss for attack class 2.0% was more evenly distributed over the first 200 seconds. Therefore, the rate of decline for attack class 2.0% was less pronounced in the first 100 seconds. In the first 100 seconds,

information transfer decreased for all attack classes by 12.7% to 96.2% from the simulation's pre-attack conditions. The information transfer loss increased during this period over a range of approximately 0.20 bits/sec to 1.50 bits/sec.

Along with the information transfer decreases shown in Figure 5.7 there was a corresponding decrease in the number of node-pairs of type 1-2. As shown in Figure 5.8, there was also sudden decline in the number of node-pairs of type 1-2 in the first 100 to 200 seconds. Over the first 100 to 200 seconds, the number of node-pairs of type 1-2 decreased for all attack classes by 0.0% to 52.6% from the simulation's pre-attack conditions. The node-pair count rate of decline during this period ranged from approximately 0 to 539 node-pairs/sec.

b) Behaviors after the First 200 Seconds.

As shown in Figure 5.7, attack classes 0.5% through 4.0% established a local minimum value at approximately 200 seconds. This local minimum value decreased with an increase in attack severity. After the local minimum was established, the information transfer level for each attack class remained relatively constant over time. For this time period, the information transfer loss stabilized at a rate of less than 0.03 bits/sec. This level was maintained until each attack class encountered its critical threshold.

During this period of slower information transfer decline shown in Figure 5.7 and Figure 5.8, there is a corresponding decrease in the number of node-pairs of type 1-2. After the first 200 seconds, the number of node-pairs for attack classes 0.5% through 4.0% declined over time at a significantly lower rate ranging from 23 to 38 node-pairs/sec. This slower rate of node-pair loss continued for each attack class until the terminal conditions were met.

c) Critical Threshold and Terminal Condition Behaviors.

For the information transfer data shown in Figure 5.7, the attack severity influence on the critical threshold time for attack classes 0.5% through 2.0% varied. Attack classes 2.5% through 4.0% achieved their critical threshold earliest. The critical threshold time increased with an increase in attack severity for attack classes 2.5% through 4.0%. The most severe attack classes, 4.5% and 5.0%, did not project a critical threshold. As shown in Figure 5.8, node-pair type 1-2 counts gradually approached the terminal conditions and did not present a critical threshold.

3) Run Type 4.

Ten run type 4 simulations are presented in this section. These simulations represented the effects of the denial-of-service attacks against the pre-attack network. These simulations represented protection strategy 3 that was previously presented in Table 4.2. For each attack class, Figure 5.9 represents changes in information transfer over the life of the simulation. Figure 5.10 depicts the corresponding change in the number of node-pairs of type 1-2 for the same simulations shown in Figure 5.9.

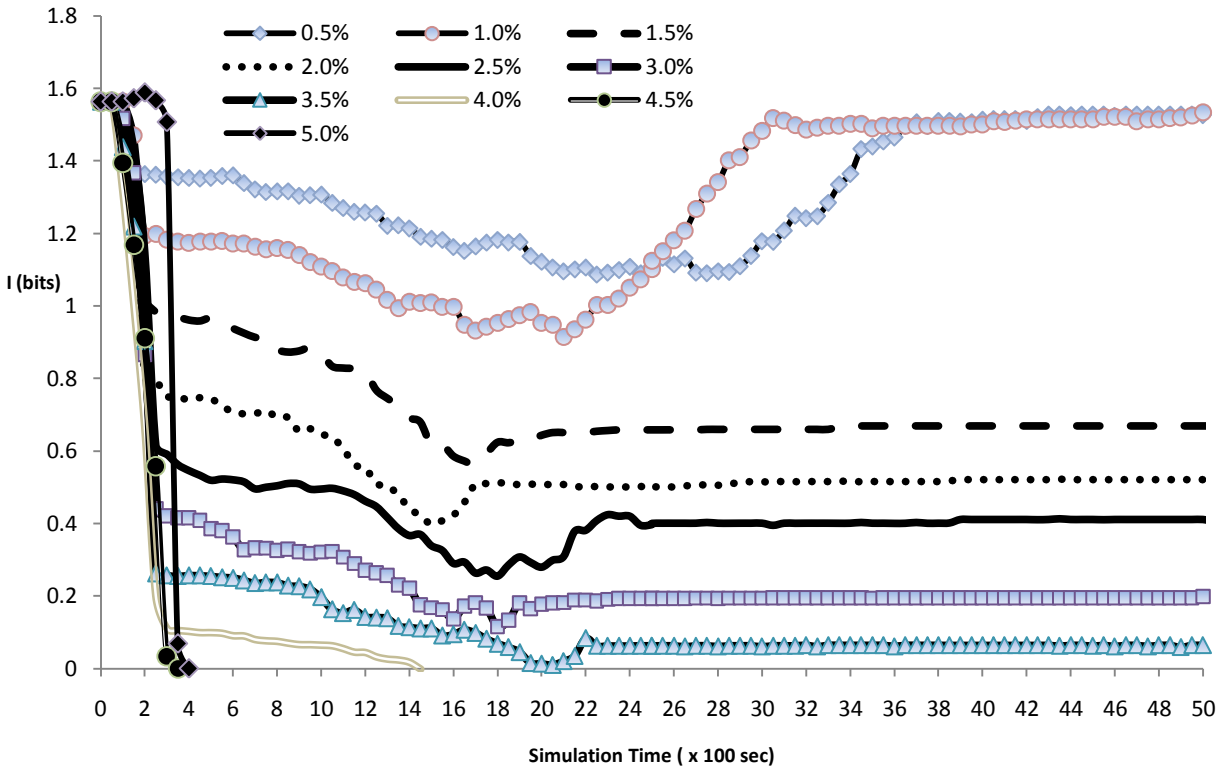


Figure 5.9. Run type 4, network stability, information transfer versus time

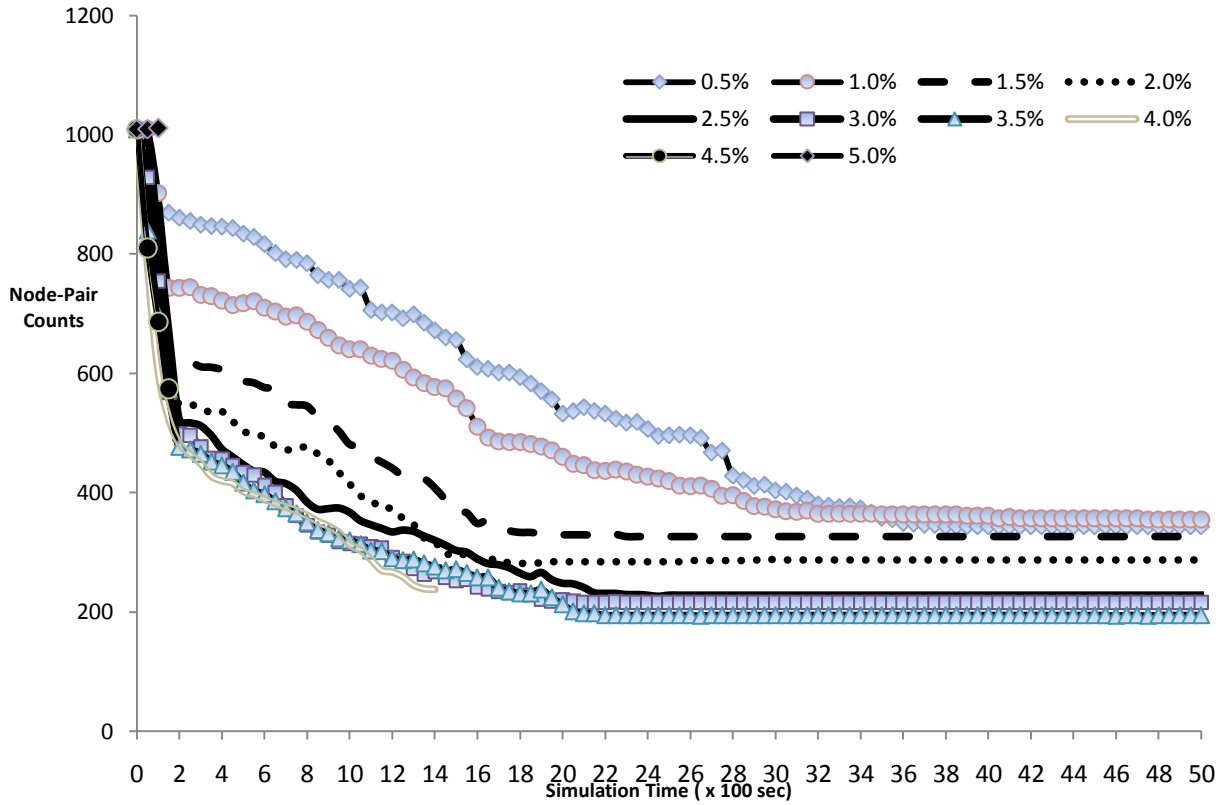


Figure 5.10. Run type 4, node-pair Type 1-2 counts versus time

a) Behaviors in the First 400 Seconds.

As shown in Figure 5.9, all run type 4 attack classes experienced a sudden decline in information transfer in the first 300 to 400 seconds. The information transfer for attack classes 4.5% and 5.0% rapidly declined and met their terminal conditions in the first 400 seconds. The information transfer decrease in the first 400 seconds experienced by all attack classes was influenced by the relative attack severity. Over the first 400 seconds, the magnitude of the information transfer loss increased with an increase in attack severity. In the first 400 seconds, information transfer decreased for all attack classes by 3.4% to 25.1% from the simulation's pre-attack conditions. The information transfer loss increased during this period over a range of approximately 0.06 bits/sec to 0.51 bits/sec.

Along with the information transfer decreases shown in Figure 5.9, there was a corresponding decrease in the number of node-pairs of type 1-2. As shown in Figure 5.10, there was also sudden decline in the number of node-pairs of type 1-2 in the first 400 seconds. Over the first 300 to 400 seconds, the number of node-pairs of type 1-2 decreased for all attack classes by 4.0% to 25.0% from the simulation's pre-attack conditions. The node-pair count rate of decline during this period ranged from approximately 40 to 206 node-pairs/sec. The loss of node-pairs by attack class varied. Attack severity did not influence the loss of node-pairs.

b) Behaviors after the First 400 seconds.

As shown in Figure 5.9, attack classes 0.5% through 4.0% established a local minimum value at approximately 400 seconds. This local minimum level decreased with an increase in attack severity. After the local minimum value was established, the information transfer decreased for each attack class at a slower rate less than 0.01 bits/sec. This slow rate

of decrease continued until each attack class encountered its equilibrium point. Attack classes 4.0% through 5.0% did not encounter an equilibrium point.

During this period of slower information transfer decline shown in Figure 5.9 and Figure 5.10, there was a corresponding decrease in the number of node-pairs of type 1-2. As shown in Figure 5.10, after the first 400 seconds, attack classes 0.5% through 4.0% declined over time at a significantly lower rate ranging from 8 to 19 node-pairs/sec. This slower rate of node-pair loss continued until each of these classes encountered its equilibrium point.

c) Equilibrium, Critical Threshold and Terminal Condition Behaviors.

Figures 5.9 and 5.10 did not exhibit a critical threshold. As shown in Figure 5.9, attack classes 0.5% through 3.5% encountered an equilibrium point after the initial 1,800 seconds. No equilibrium point occurred for attack class 4.0%, which met its terminal conditions around 1,400 seconds. Attack classes 4.5% and 5.0% did not encounter an equilibrium point; they met their terminal conditions in the first 400 seconds. The equilibrium point and level varied by attack class. The equilibrium levels attack classes 0.5% through 3.5% occurred in the range of approximately 200 to 400 node-pairs. The minimum node-pair count for attack class 4.0%, 4.5%, and 5.0% at their terminal conditions was 240, 480, and 485 node-pairs, respectively.

E. Attack Detection

During each simulation, this research inferred an attack's existence as the first time that the number of node-pairs of type 1-1 increased by more than 50%. To measure the accuracy of this detection, the inferred attack times were compared with the actual attack commencement times. The actual attack commencement time was recorded when the first

targeted node was removed during the simulation. Table 5.4 depicts the variance between the actual attack time and the inferred attack time for all 40 simulation runs by run type. This table summarizes results previously presented in Section D of this chapter. The differences are shown in Table 5.4 where the actual attack times were later than the projected attack times are denoted in parentheses. As previously defined in Table 4.2, each run type represented a different protection scheme.

Table 5.4. Attack detection percent variance, actual versus inferred

Attack Class	*Total Num of Attacked Nodes / Avg Degree*	*Run Type 1 Percent Variance*	*Run Type 2 Percent Variance*	*Run Type 3 Percent Variance*	*Run Type 4 Percent Variance*	*Average Percent Variance*
0.5%	60 / 47	1.9%	1.0%	(0.8%)	(2.8%)	(0.2%)
1.0%	124 / 41	1.6%	7.1%	(0.6%)	0.8%	2.2%
1.5%	179 / 38	11.1%	3.2%	(0.6%)	(2.0%)	2.9%
2.0%	241 / 35	1.4%	0.0%	1.5%	(2.4%)	0.1%
2.5%	303 / 31	5.3%	4.8%	0.0%	1.1%	2.8%
3.0%	363 / 29	7.1%	1.9%	(2.5%)	(3.8%)	0.7%
3.5%	421 / 26	(25.5%)	4.8%	(1.8%)	0.0%	(5.6%)
4.0%	478 / 24	2.3%	1.3%	(2.0%)	(5.7%)	(1.0%)
4.5%	544 / 22	11.1%	6.4%	20.7%	(7.4%)	7.7%
5.0%	613 / 20	9.8%	12.7%	0.0%	9.5%	8.0%
Average Percent Variance		2.6%	4.3%	1.4%	(1.3%)	1.8%

Figure 5.11 depicts the descriptive statistics for the average percent variance across all 40 simulation runs. Over the 40 simulation runs, the average detection time derived in this

research varied from the actual attack time by 1.75 %. Additional attack detection data can be found in appendices M through P.

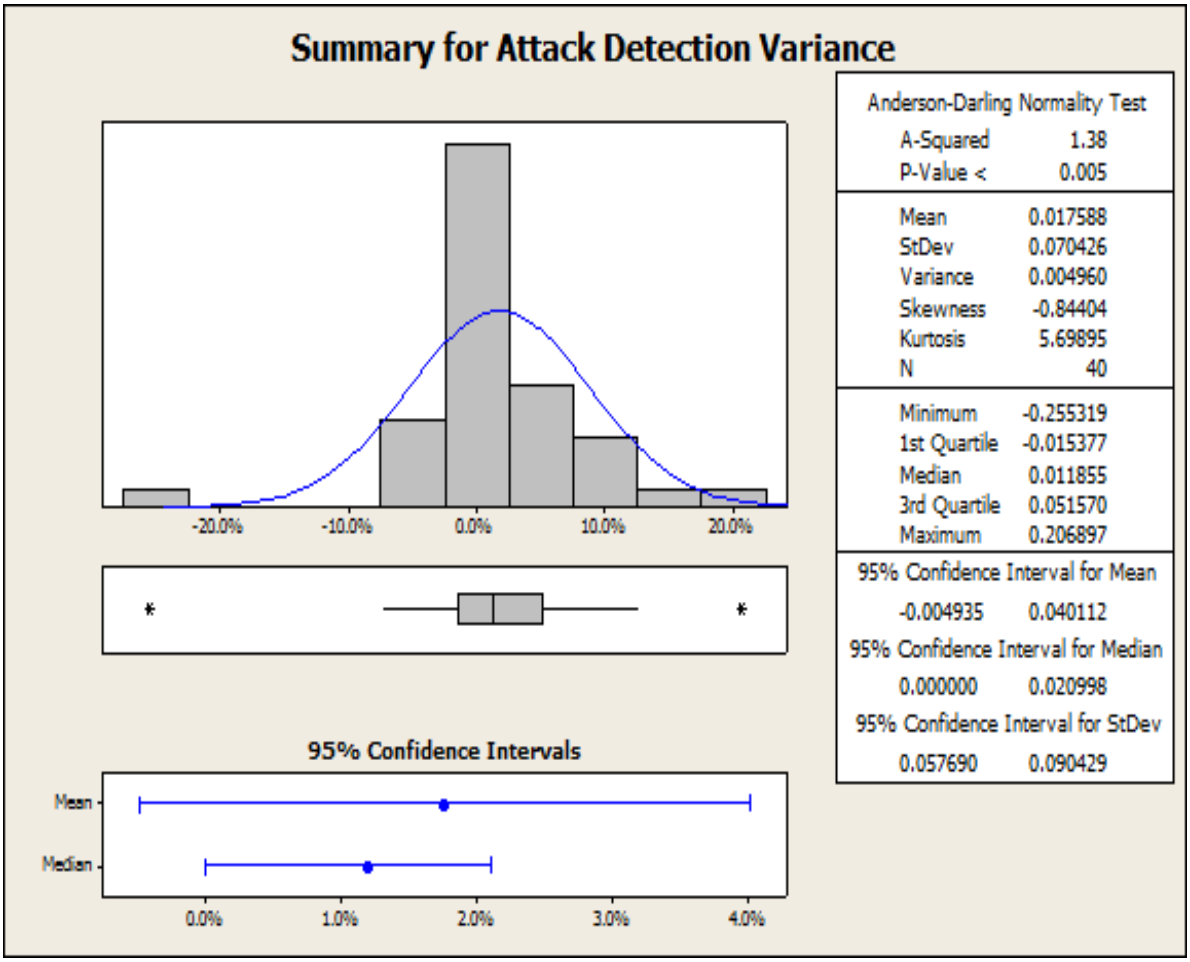


Figure 5.11. Attack detection, descriptive statistics over 40 simulation runs

F. Network Protection

For attack classes 0.5% through 3.5%, the results depicted in Chapter V indicate that after the early rapid network stability degradation, there was a recovery in network stability. The level of recovery was influenced by the attack class severity. After the recovery, each of these attack classes established a relatively constant equilibrium level at this recovery level for the remainder of the simulation. The recovery levels were most pronounced and at

higher information transfer levels for the run type 4 simulations. Table 5.5 depicts the percent of the original information transfer recovered at the equilibrium level. This table summarizes results previously presented in Section D of this chapter. The 20 simulations of run types 2 and 3 exhibited similar equilibrium behaviors but not as pronounced as run type 4. The average equilibrium effects for all run types can be found in the appendix.

Table 5.5. Network protection, equilibrium and stability recovery for run type 4

Attack Class	*Information Transfer Recovery at the Equilibrium level*
0.5% and 1.0%	Almost 100%
1.5%	Approx. 44%
2.0%	Approx. 31%
2.5%	Approx. 25%
3.0%	Approx. 13%
3.5%	Approx. 6%
4.0%, 4.5% and 5.0%	0%

This chapter has shown that changes in network stability during the attack simulations resulted in corresponding variations in the number of node-pairs. The results depicted in Figure 5.8 and Figure 5.10 were consistent with the behaviors observed in Figures 5.7 and Figure 5.9, respectively. The results indicated that attack severity and run type influenced node-pair and network stability behaviors during the simulated attacks. The next chapter will discuss the theoretical and practical implications of the research results presented in this section.

CHAPTER VI. DISCUSSION

This chapter will summarize the previously presented rationale and methods of this research. This will be followed by a discussion of the theoretical and practical implications of this research investigation. A discourse on research limitations will conclude this chapter.

A. Research Summary

For the first time, Colored Petri Net (CPN) modeling and simulation techniques have been used in this research to simulate targeted denial-of-service attacks over the Internet's router infrastructure. This research developed a cyber attack model and simulation using Colored Petri Nets with actual Internet router connectivity data. The simulation was used to study changes in the Internet's connectivity state during a targeted denial-of-service attack. Using scale-free network theory, this research sought to determine whether there is strong evidence that underlying network-wide connectivity changes (*attack markers*) that occur during the formative stages of a massive targeted denial-of-service attack against large-scale computer networks can be used to study cyber attack mechanics.

As presented earlier in this dissertation, the CPN model and simulation developed specifically for this research investigation was used to 1) determine whether it is possible to detect small subtle changes (*attack markers*) in the connectivity environment of the Internet's router connectivity infrastructure that occur during a cyber attack, and 2) if the first premise is valid, to ascertain the feasibility of using these changes as a means for a) early infrastructure attack detection and b) router infrastructure protection strategy development against these attacks.

Previous studies have shown that the Internet's router infrastructure is vulnerable to targeted denial-of-service attacks against the network's most connected routers. Severe

damage to the Internet's router infrastructure could lead to significant disruptions in global commerce as well as impede national security objectives. The availability of easy-to-use malicious attack tools and significant potential gain for an attacker has increased the frequency of all cyber attacks. Early attack detection is critical to avoid catastrophic network stability degradation. In addition, cost effective and reliable means to protect the Internet's router infrastructure from malicious attempts to impede critical global transactions is a national priority (Richardson 2008; U. S. House 2005).

Previous discourse has shown that current cyber attack detection techniques rely on the tedious and time-consuming examination of individual network router communication packets. This leads to a reactive process that detects an attack only after significant network degradation has occurred. In addition, most network protection schemes are dependent upon router traffic analysis for deployment of defensive countermeasures. This dependence also leads to a reactive defense of the network's connectivity.

Tracking and investigating individual router communications is costly, inefficient, and impractical. As previously discussed in Chapter I, to avoid the problems associated with the large volumes of complex network traffic, attack detection and network protection must be environmentally-based. To achieve early attack detection and efficient network protection requires a new paradigm. For attack detection purposes, this paradigm must rely on systemic changes to the underlying physical characteristics of the router's infrastructure during the formative stages of an attack. Understanding these changes may also provide a means to protect the Internet's core router infrastructure from catastrophic network stability degradation. Attack detection and network protection will be discussed later in this chapter.

1) Methods Summary.

As previously discussed in Chapter IV, the attack simulations used data from an actual large scale Internet router infrastructure to simulate router connectivity. Using a trace-route based protocol, the University of Washington's Rocketfuel project extracted snapshots of the Internet's core router infrastructure (Alderson et al. 2005; Rocketfuel: An ISP topology mapping engine n.d.; Spring et al. 2004) at the autonomous system level. These data represented router adjacencies present over a time period ranging from December 2001 to January 2002. The research simulation utilized these data as its initial pre-attack state. Specifically, this research used the United States AT&T (ASN 7018) backbone and access router datasets extracted by the Rocketfuel project team. The pre-attack network represented 11,800 routers connected by 28,592 links.

As discussed earlier, the research model and attack simulation results were consistent with earlier studies of scale-free computer networks found in the literature. To emulate denial-of-service attacks, the simulations employed a targeted node removal attack strategy (Albert, Jeong, and Barabasi 2000; Crucitti et al. 2004; Salla 2005; Sun et al. 2007). During each simulation, the network's most connected nodes were randomly removed. Over specific time intervals, the changes in the simulated network's underlying physical characteristics and stability were studied. As previously discussed in Chapter II, network stability was measured using two network characteristics, mutual information transfer, and the network connectivity parameter.

This research executed 40 simulation runs. Each execution represented a different attack scenario. For every attack scenario there were 4 different protection schemes emulated. Table 4.1 defines the simulation's execution parameters. Table 4.2 defines the

attack scenarios. Using a Pearson correlation analysis, this investigation focused on the relationship of information transfer and the number of node-pairs by type. Node-pair types were previously defined in Chapter I.

For select simulations, the analysis was performed in 50-second time intervals. This correlation analysis led to the selection of significant node-pair types. The number of node-pairs identified as significant types exhibited a correlation greater than 0.8 at a 99.9% statistical confidence level. During the attack simulations, as node-pairs were destroyed and new node-pairs were established, the count of each node-pair type changed. During each simulation, changes in the number of node-pairs of type 1-1 and 1-2 were correlated with variation in the network's overall connectivity stability. These variations represented critical changes in the physical characteristics of the network's connectivity during an attack. The next section will present the significant theoretical implications of this research investigation.

B. Theoretical Implications for this Research

The theoretical implications of this research with regard to the Theory of Cyber Attack Mechanics will now be addressed (Stephenson and Prueitt 2005). In Chapter II, this dissertation previously discussed the relevance of the cyber attack mechanics hypothesis developed by Stephenson and Prueitt (2005). The formal representation of a cyber attack as postulated by Stephenson and Prueitt (2005) is: $a \cdot \{e_f \Rightarrow \Delta\, D_{f(e)}\} \Rightarrow \xi$, where:

a is an attack, that is an ordered threat-vulnerability pair.

ξ is an event marker.

e_f is an element of a fractal set describing Internet based router traffic.

$\Delta\, D_{f(e)}$ is the change in the fractal dimension

Table 6.1. Cyber attack mechanics hypothesis and this dissertation

Term	*Theory of Cyber Attack Mechanics*	*Research in this dissertation*
a	Ordered threat-vulnerability pair.	Threat: easy availability of targeted denial-of-service techniques. Vulnerability: nature of scale-free computer network connectivity.
e_f	Element describing Internet based router traffic.	Network connectivity stability and fragmentation.
$\Delta\ D_{f(e)}$	Change in e_f.	Variance in the expected behaviors of scale-free router connectivity
ξ	An event (attack) marker.	Connectivity "noise" as depicted through changes in the network's characteristic statistical mechanics.

The work detailed in this research can be applied to the cyber attack mechanics hypothesis described as follows: The Internet's router connectivity can be represented as a finite state machine. During a targeted denial-of-service attack, finite network stability states present during the breakdown of normal communications characterize its relative connectivity fragmentation. The network's connectivity topology can be formally represented through its node states. Two adjacent node states represent a node-pair state. Subtle changes in node-pair states can formally represent variations in the network's topology. Changes in the node-pair states present during a denial-of-service attack produce emergent patterns and residual connectivity "noise." It is possible to distinguish the "noise" from the emergent patterns through characteristic changes in the underlying network degree distribution that represent *attack markers*. These attack markers indirectly represent variation in the network's emergent connectivity patterns. During the formative stages of the attack, discrete attack

markers may provide a means to sense an attack's existence before significant stability degradation has occurred. In addition, by preserving discrete attack markers it should be possible to protect the network's connectivity topology against denial-of-service attacks.

The cyber attack mechanics theory also postulates that scale-free networks under attack will encounter halting conditions. The halting conditions represent a sudden rapid and complete degradation in network connectivity. The critical threshold presented in the results chapter of this dissertation has confirmed the existence of these halting conditions. Another important aspect of this theory is the existence of *attack markers*. *Attack markers* have been previously defined in Chapter I. Node-pair type behaviors relative to information transfer as studied in this research indicated that *attack markers* exist.

This section has shown that the results previously presented support the Theory of Cyber Attack Mechanics as a scientifically sound hypothesis. The next section in this chapter will discuss potential practical applications of this research.

C. Practical Implications of this Research

The necessity for new techniques that utilize changes in the Internet's environment to thwart malicious attacks against the Internet's router connectivity infrastructure was previously discussed in Chapter I. This research has shown that using the network's environment to detect denial-of-service attacks as well as protecting the network is feasible. This research found two *attack markers* represented as changes in the number of node-pairs of type 1-1 and 1-2. This section will first address how node-pair type 1-1 counts were used to detect the existence of a denial-of-service attack against the network. The discussion will then focus on using node-pairs of type 1-2 as a means to protect network connectivity stability against a denial-of-service attack.

1) Using Node-pair Counts to Detect Denial-of-Service Attacks.

As previously discussed in Chapter I, identifying a practical means to detect a denial-of-service attack in its formative stages has useful applications. The underlying physical characteristics of an attacked network could be used as an attack indicator. The underlying physical characteristics were represented in this research as the number of node-pairs of type 1-1. As a result of the correlation analysis presented in Chapter V, changes in the number of node-pairs of type 1-1 were considered *attack markers.* The changes in the number of node-pairs of type 1-1 were strongly correlated with network stability degradation. In the first 150 seconds of each simulation, the rapid increase in the number of node-pairs of type 1-1 represented network fragmentation during the formative stages of an attack because they represented isolated node-pairs.

Table 5.4 suggests that the network's environment can be used for attack detection. Current detection methods assume that the attack would be detected at the critical threshold,

when the network is rapidly degrading and no communication is possible. The detection method used in this research indicates an attack's existence much earlier than the critical threshold. The detection techniques described here led to an approximately 90% improvement over the detection time using critical threshold as an indicator of attack.

Based on these results, it may be possible to develop a new application that monitors the number of node-pairs of type 1-1 present in the network over time. An automated sensor might be employed to signify an attack's existence using the techniques described in this section. Further potential applications of these research findings will be discussed in Chapter VII. The next section will present the practical implications of using a different node-pair type to protect the network from a targeted denial-of service attack.

2) Using Node-pair-type Counts for Network Protection.

As previously discussed in Chapter I, the Internet's router infrastructure defense would benefit from a new security tool that utilizes a systemic paradigm that can protect the network's connectivity. By manipulating the number of node-pairs of specific types, this investigation has uncovered a potential technique to protect the network's connectivity. This research investigation selected node-pair type 1-2 for further study of potential protection effects. The reasoning for this selection was 1) it is assumed that routers with fewer connections are less expensive and less complex to protect than routers with a large number of connections, 2) the literature indicates that protecting the outer low degree nodes in a scale-free network may be advantageous for the network's overall stability, 3) the consistent data trends previously shown in Figure 5.7 and Figure 5.8, and 4) the correlation of this node-pair type with information transfer met the previously defined research criteria for *attack marker* selection.

The evidence of network protection using node-pair type manipulation found for run types 2, 3, and 4 indicates that it is feasible to protect the network using these techniques. However, further study is required to determine the optimum *attack markers*. This section has shown that changes in the network's connectivity stability are accompanied by corresponding changes in the number of node-pairs of type 1-2. The changes in the node-pair type count were consistent over the life of the simulation. As shown in Table 5.5, simulations using protection strategy 3 (run type 4) as defined earlier suggest that it is possible to protect the network's connectivity stability by manipulating the node-pair type counts. Therefore, this research has shown that using the network's environment as a means to protect the network is scientifically plausible and merits further study.

D. Limitations

This research was a feasibility study that sought to determine whether the network's connectivity environment could be utilized for attack detection and network protection. The theoretical potential for attack attribution discussed in the theory of cyber attack mechanics was not addressed. The feasibility of the techniques proposed was benchmarked against a static pre-attack network. A dynamic pre-attack network undergoing normal router failures during the simulation may also be a useful benchmark. As shown earlier, the Internet's robust connectivity allows normal network operations, even with the occurrence of regular, random, and routine router failures. A study of simulation behaviors using a dynamic pre-attack network as the benchmark might uncover trends not found in this research investigation.

This study only considered changes in the number of node-pair of types 1-1 and 1-2 as *attack markers*. The correlation analysis produced other node-pair types that exhibited a

relatively strong correlation with network stability. It is possible that other node-pair types may provide further evidence to support or refute the conclusions of this investigation. In addition, this study only considered denial-of-service attacks on large scale router infrastructures and may not be applicable to significantly smaller networks. Further investigation is needed to determine if this method could be applied to other types of exploits such as worm or virus propagation attacks. However, it is expected that regardless of the attack genre, the attack router infrastructure will likely exhibit systemic router failures.

E. Chapter Summary

The CPN model and simulation developed for this research behaved in a consistent manner with relevant scale-free network connectivity studies. This model and simulation has provided a unique methodology for further study of scale-free network connectivity attacks. This chapter has discussed the practical and theoretical implications of this research including the feasibility of using the Internet router node-pair types to detect attacks and protect the network from denial-of-service attacks. The research limitations presented may provide foundation for further study. Research conclusions and potential future work to expand the discoveries of this research investigation will be addressed in the next chapter.

CHAPTER VII. CONCLUSIONS

This chapter will first discuss the conclusions reached by this research investigation and then address future work.

A. Research Relevance

As previously discussed in Chapter I, the Internet's router infrastructure, a scale-free computer network, is vulnerable to targeted denial-of-service attacks. Current attack detection techniques and countermeasures have been shown to be costly and inefficient (Casey 2002; Casey 2004; Mizrak et al. 2006; Rattray 2001a; Stephenson 2006; Stephenson and Prueitt 2005). Attack detection before the network has suffered substantial degradation would greatly enhance the security of military and economic transactions.

This research investigation developed a Colored Petri Net model and simulation that emulated changes in the Internet's core router infrastructure connectivity during a targeted denial-of-service attack. From these simulations, *attack markers* were discovered that identified the critical indicators of DoS attacks. These results support the feasibility of using knowledge of a network's underlying physical connectivity environment for defensive purposes. In summary this research concludes that:

1. The unique CPN model and its simulation results were consistent with the scale-free network literature.
2. It is plausible to use changes in a scale-free computer network's underlying physical characteristics to study attack detection and network protection.
3. Subtle changes in the number of node-pairs of type 1-1 and 1-2 represent a physical characteristic that can be used as *attack markers*.

4. The Theory of Cyber Attack Mechanics is a scientifically sound premise.
5. Specific *attack marker* node-pair types can be used to detect attacks, while others can be used to protect a network's connectivity.
6. The protective effects of certain *attack markers* were strongly influenced by the attack's severity.
7. The model and simulation developed in this research has provided a prototype for studying cyber attacks.
8. A systemic approach for identifying cyber attacks based on the network's environment may provide the foundation for the development of new network security tools.

B. Attack Detection and Network Protection Application

This study has provided confirmation that monitoring a network's environment can be used for defensive purposes. It has presented a unique paradigm for the development of future network security applications. This section will describe a potential network security application that might be derived from this research.

Autonomous agents are "a group of free-running processes which can act independently of each other and the global controls" (Crosbie and Spafford 1995). These agents are goal-oriented systems that can be defined for single purposes while maintaining a small environmental footprint. The techniques proposed in this research might lead to a network-wide defense strategy by strategically placing autonomous agents throughout the network. Real time node-pair information collected by these autonomous agents could determine the location of potential *attack marker* node-pairs and dynamically provide a list of all node-pairs in its domain to a centralized command center.

After the centralized command center has identified *attack marker* node-pairs and sent the information to the autonomous agents, the agents would periodically send the command center the number of each *attack marker* node-pairs. The number of *attack marker* node-pairs could be analyzed by the command center, and if it appears an attack is forming then the command center could notify the affected autonomous agents. One possible quick countermeasure that could be taken by these autonomous agents might be to increase the link capacity of the node-pairs identified as *attack markers.* This increase in link capacity would protect these node-pairs from attack-induced destruction. After the threat has subsided, the command center could order the autonomous agents to remove the additional capacity.

Advantages of this attack detection and protection methodology are: (1) it is cost efficient because only nodes that have been flagged will increase their capacity; (2) it is resource efficient because the only small amounts of information must be transmitted to a few select routers; (3) it is specific; only *attack marker* node-pairs are updated; (4) the time-interval of the data transmissions can be fine-tuned as needed; and (5) it is flexible because autonomous agent designations can be dynamic and changed to optimize agent deployments.

C. Future Work

This study can lead to several avenues of work. Expansion of this feasibility study might lead to the development of new cyber attack tools. These tools would be proactive and systemic, not reactive and haphazard. While this research validated two potential *attack marker* node-pair types, there may be other more accurate *attack marker* node-pair types. Further study of the other 22 node-pair types identified by the correlation analysis might lead to more efficient *attack marker* node-pair types. In addition, the collection and analysis

protocol for the identification of *attack marker* node-pair types was manually intensive. Automation of this analysis would greatly enhance the usability of this technique.

To further evaluate the variability and validity of the research conclusions, the research simulations discussed in Chapters V and VI should be replicated. This research simulation analyzed a large scale-free computer network. Large networks may behave differently than smaller networks. Future work is needed to determine whether the *attack marker* selection methods discovered in this research are scalable to smaller networks. Additional studies using other Internet router datasets, such as those found at the CAIDA website (www.caida.org), might advance this study's contribution to the cyber attack literature. This research dissertation focused on scale-free network connectivity theory, but it might be possible to base the simulations developed in this research on other network theories, such as the Erdos and Renyi (ER) (1960) exponential network.

Further investigation using the research results of this study might lead to the development of an information warfare offensive application. Since protection of the aforementioned node-pair types has been shown to protect network stability, it may be reasonable to assume that a DoS attack against these node-pair types might significantly degrade an adversary's network connectivity stability.

Instead of comparing the attacked networks against a static pre-attack network, further study benchmarking the attack simulation results against a dynamic pre-attack network undergoing normal router failures might yield additional useful information. A next phase of this research would validate the simulation results against a virtual Internet router lab. By implementing the same protocols used in this investigation against the virtual router lab, the results from the two settings could be compared. If the virtual lab testing results

were consistent with the CPN simulations, then these research findings would further validate the results presented here. Additional validation using an existing Internet security and event management tool over an actual Internet region may also be possible.

D. Summary

As discussed earlier in this dissertation, detecting cyber attacks and protecting networks using individual router anomalies is costly, inefficient, and impractical. This study presented a new paradigm that focused on systemic environmental network changes that occur during a cyber attack. The techniques proposed by this investigation are unique and have not been cited in the literature. This research has provided evidence that using knowledge of the Internet's connectivity topology and its physical characteristics to protect the router infrastructure from targeted DoS attacks is a scientifically sound premise. In addition, this research has also shown that it is plausible that these techniques could be used to detect targeted DoS attacks and may lead to new network security tools.

REFERENCES

Adkins, Bonnie N. 2001. *The spectrum of cyber conflict from hacking to information warfare: What is law enforcement's role?* U. S. Air Command and Staff College, Air University.

Aiello, William, Fan Chung, and Linyuan Lu. 2000. A random graph model for massive graphs. Paper presented at Proceedings of the Thirty-Second Annual ACM symposium on Theory of computing, Portland, Oregon, United States.

Albert, R. and A. L. Barabasi. 2000. Topology of evolving networks: Local events and universality. *Physical Review Letters* 85, no. 24: 5234-5237.

_____. 2002. Statistical mechanics of complex networks. *Reviews of Modern Physics* 74, no. 1: 47-97.

Albert, R., H. Jeong, and A. L. Barabasi. 1999. Diameter of the world-wide web. *Nature* 401, no. 6749: 130-131.

_____. 2000. Error and attack tolerance of complex networks. *Nature* 406, no. 6794: 378-382.

Alderson, D., L. Li, W. Willinger, and J. C. Doyle. 2005. Understanding internet topology: Principles, models, and validation. *IEEE-ACM Transactions on Networking* 13, no. 6: 1205-1218.

Ale and Kub. 2003. Toward a formalization of emergence. *Artif. Life* 9, no. 1: 41-65.

Barabasi, A. L. and R. Albert. 1999. Emergence of scaling in random networks. *Science* 286, no. 5439: 509-512.

_____. 2002. Statistical mechanics of complex networks. *Reviews of Modern Physics* 74, no. 1: 47.

Barabasi, A. L., R. Albert, and H. Jeong. 2000. Scale-free characteristics of random networks: The topology of the world-wide web. *Physica A* 281, no. 1-4: 69-77.

Barabasi, A. L., Erzsebet Ravasz, and Tamas Vicsek. 2001. Deterministic scale-free networks. *Physica A: Statistical Mechanics and its Applications* 299, no. 3-4: 559-564.

Boccaletti, S., V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang. 2006. Complex networks: Structure and dynamics. *Physics Reports-Review Section of Physics Letters* 424, no. 4-5: 175-308.

Borchgrave, A., F.J. Cilluffo, S.L. Cardash, and M.M. Ledgerwood. 2001. *Cyber threats and information security meeting the 21st century challenge*. Csis report - executive summary of four working group reports on homeland defense. Washington, D.C.: Center for Strategic and International Studies.

Boschetti, F., M. Prokopenko, I. Macreadie, and A. M. Grisogono. 2005. Defining and detecting emergence in complex networks. In *Knowledge-based intelligent information and engineering systems, pt 4, proceedings*, 3684:573-580.

Caldarelli, G, R Marchetti, and L Pietronero. 2000. The fractal properties of internet. *Europhysics letters* 52: 386-992.

Casey, E. 2002. Error, uncertainty, and loss in digital evidence. *International Journal of Digital Evidence* 1, no. 2. http://www.utica.edu/academic/institutes/ecii/ijde/articles.cfm accessed Date Accessed)|.

Casey, Eoghan. 2004. Network traffic as a source of evidence: Tool strengths, weaknesses, and future needs. *Digital Investigation* 1, no. 1: 28-43.

Cassey, Lee. 2004. Emergence and universal computation. *Metroeconomica* 55, no. 2-3: 219-238.

Chakrabarti, A., Manimaran,G. 2002. Internet infrastructure security: A taxonomy. *IEEE Network* 16, no. 6: 13-21.

Chakraborty, D., A. Ashir, T. Suganuma, G. Mansfield Keeni, T. K. Roy, and N. Shiratori. 2004. Self-similar and fractal nature of internet traffic. *International Journal of Network Management* 14: 119-129.

Cheetancheri, Senthilkumar G., John Mark Agosta, Denver H. Dash, Karl N. Levitt, Jeff Rowe, and Eve M. Schooler. 2006. A distributed host-based worm detection system. Paper presented at Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense, Pisa, Italy.

Cheol-Joo, Chae, Lee Seoung-Hyeon, Lee Jae-Seung, and Lee Jae-Kwang. 2007. A study of defense ddos attacks using ip traceback. Paper presented at Proceedings of the The 2007 International Conference on Intelligent Pervasive Computing.

Cheung, S. 2006. Denial of service against the domain name system. *IEEE Security & Privacy* 4, no. 1: 40-45.

Cicic, T. 2008. On basic properties of fault-tolerant multi-topology routing. *Computer Networks* 52, no. 18: 3325-3341.

Cohen, R., Erez, K., ben-Avraham, D., & Havlin, S. 2000. Resilience of the internet to random breakdowns. *Physical Review Letters* 85, no. 21: 4626-4628.

_____. 2001. Breakdown of the internet under intentional attack. *Physical Review Letters* 86, no. 16: 3682-3685.

Convery, S., D. Cook, and M. Franz. 2004. An attack tree for the border gateway protocol. *IETF Draft - Internet-Draft draft-ietf-rpsec-bgpattack-00*, no. http://tools.ietf.org/html/draft-ietf-rpsec-bgpattack-00 accessed Date Accessed)|.

Costa, L. D., F. A. Rodrigues, G. Travieso, and P. R. V. Boas. 2007. Characterization of complex networks: A survey of measurements. *Advances in Physics* 56, no. 1: 167-242.

Cover, T. M. and Joy A. Thomas, eds. 2006. *Elements of information theory*. Hoboken, N.J.: Wiley-Interscience.

Criado, R., A. G. del Amo, B. Hernandez-Bermejo, and M. Romance. 2006. New results on computable efficiency and its stability for complex networks:59-74.

Crosbie, Mark and Gene Spafford. 1995. Defending a computer system using autonomous agents. In *Proceedings of the 18th National Information Systems Security Conference*. Baltimore, MD: Department of Computer Sciences, Purdue University.

Crucitti, P., V. Latora, and M. Marchiori. 2004. Model for cascading failures in complex networks. *Physical Review E* 69, no. 4.

Crucitti, P., V. Latora, M. Marchiori, and A. Rapisarda. 2003a. Efficiency of scale-free networks: Error and attack tolerance. *Physica a-Statistical Mechanics and Its Applications* 320: 622-642.

_____. 2004. Error and attack tolerance of complex networks. *Physica a-Statistical Mechanics and Its Applications* 340, no. 1-3: 388-394.

Crucitti, Paolo, Vito Latora, Massimo Marchiori, and Andrea Rapisarda. 2003b. Efficiency of scale-free networks: Error and attack tolerance. *Physica A: Statistical Mechanics and its Applications* 320: 622-642.

Crutchfield, James P. 1994. The calculi of emergence: Computation, dynamics and induction. *Physica D: Nonlinear Phenomena* 75, no. 1-3: 11-54.

Dall'Asta, L., A. Barrat, M. Barthelemy, and A. Vespignani. 2006. Vulnerability of weighted networks. *Journal of Statistical Mechanics-Theory and Experiment*: 12.

Daniels, Thomas E. 2002. Reference models for the concealment and observation of origin identity in store-and-forward networks. *DAI* 64, no. 09B: 157.

Daniels, Thomas E. and Eugene H. Spafford. 2000. Network traffic tracking systems: Folly in the large? In *Proceedings of the 2000 workshop on New security paradigms*. Ballycotton, County Cork, Ireland: ACM Press.

Dekker, Anthony H. and Bernard Colbert. 2008. Scale-free networks and robustness of critical infrastructure networks. *Complexity International* 12.

Demetrius, L. and T. Manke. 2005. Robustness and network evolution - an entropic principle. *Physica a-Statistical Mechanics and Its Applications* 346, no. 3-4: 682-696.

Dirk, Ourston, Matzner Sara, Stump William, and Hopkins Bryan. 2004. Coordinated internet attacks: Responding to attack complexity. *J. Comput. Secur.* 12, no. 2: 165-190.

Dobson, I., B. A. Carreras, V. E. Lynch, and D. E. Newman. 2007. Complex systems analysis of series of blackouts: Cascading failure, critical points, and self-organization. *Chaos* 17, no. 2: 13.

Donnet, B. and T. Friedman. 2007. Internet topology discovery: A survey. *Communications Surveys & Tutorials, IEEE* 9, no. 4: 56-69.

Dorogovtsev, S. N., A. V. Goltsev, and J. F. F. Mendes. 2008. Critical phenomena in complex networks. *Reviews of Modern Physics* 80, no. 4: 1275-1335.

Dorogovtsev, S. N. and J. F. F. Mendes. 2002. Evolution of networks. *Advances in Physics* 51, no. 4: 1079-1187.

Douligeris, C. and A. Mitrokotsa. 2004. Ddos attacks and defense mechanisms: Classification and state-of-the-art. *Computer Networks* 44, no. 5: 643-666.

Erdos, P. and A. Renyi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Science* 5: 17-61.

Estrada, E., D. J. Higham, and N. Hatano. 2009. Communicability betweenness in complex networks. *Physica a-Statistical Mechanics and Its Applications* 388, no. 5: 764-774.

Furuya, S. and K. Yakubo. 2008. Generalized strength of weighted scale-free networks. *Physical Review E* 78, no. 6.

Gallos, L. K. and P. Argyrakis. 2007. Scale-free networks resistant to intentional attacks. *Epl* 80, no. 5.

Gallos, L. K., R. Cohen, F. Lijeros, P. Argyrakis, A. Bunde, and S. Havlin. 2006. Attack strategies on complex networks, ed. V. N. Alexandrov, G. D. VanAlbada, P. M. A. Sloot and J. Dongarra:1048-1055: Springer-Verlag Berlin.

Gallos, L. K., Cohen, R., Argyrakis, P., Bunde, A., & Havlin, S. 2005. Stability and topology of scale-free networks under attack and defense strategies. *Physical Review Letters* 94, no. 18: 188701-188704.

Gallos, L. K., F. Liljeros, P. Argyrakis, A. Bunde, and S. Havlin. 2007. Improving immunization strategies. *Physical Review E* 75, no. 4.

Gudkov, V. and V. Montealegre. 2008. Analysis of networks using generalized mutual entropies. *Physica a-Statistical Mechanics and Its Applications* 387, no. 11: 2620-2630.

Guillaume, J. L., M. Latapy, and C. Magnien. 2005. Comparison of failures and attacks on random and scale-free networks. In *Principles of distributed systems*, 3544:186-196: Springer Berlin / Heidelberg.

Haggerty, J., Q. Shi, and M. Merabti. 2005. Early detection and prevention of denial-of-service attacks: A novel mechanism with propagated traced-back attack blocking. *Ieee Journal on Selected Areas in Communications* 23, no. 10: 1994-2002.

Hamed, Haddadi, Uhlig Steve, Moore Andrew, Mortier Richard, and Rio Miguel. 2008. Modeling internet topology dynamics. *SIGCOMM Computer Commuications. Rev.* 38, no. 2: 65-68.

Holme, P., B. J. Kim, C. N. Yoon, and S. K. Han. 2002a. Attack vulnerability of complex networks. *Physical Review E* 65, no. 5: 14.

_____. 2002b. Attack vulnerability of complex networks. *Physical Review E* 65, no. 5.

Hu, H. B. and X. F. Wang. 2008. Unified index to quantifying heterogeneity of complex networks. *Physica a-Statistical Mechanics and Its Applications* 387, no. 14: 3769-3780.

Huang, L., Y. C. Lai, and G. R. Chen. 2008. Understanding and preventing cascading breakdown in complex clustered networks. *Physical Review E* 78, no. 3: 5.

Huang, W. and C. G. Li. 2007. Cascading breakdown on weighted scale-free complex networks. *Progress of Theoretical Physics* 118, no. 1: 15-24.

Hussain, Alefiya, John Heidemann, and Christos Papadopoulos. 2003. A framework for classifying denial of service attacks. In *Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications*. Karlsruhe, Germany: ACM Press.

Jamakovic, A., S. Uhlig, and I. Theisler. 2007. On the relationships between topological metrics in real-world networks. In *European Conference on Complex Systems*. Dresden, Germany.

Jensen, K. 1994. An introduction to the theoretical aspects of coloured petri nets. In *A decade of concurrency*, ed. W.-P. de Roever J.W. de Bakker, G. Rozenberg, 803:230-272. Enschede: Springer-Verlag.

_____. 1997. *Coloured petri nets: Basic concepts, analysis methods and practical use*. Monographs in theoretical computer science: Springer-Verlag.

_____. 1998. An introduction to the practical use of coloured petri nets. In *Lectures on petri nets ii: Applications, lecture notes in computer science*, ed. W. Reisig and G. Rozenberg, 1492:237-292: Springer-Verlag

Jeong, H., Z. Neda, and A. L. Barabasi. 2003. Measuring preferential attachment in evolving networks. *Europhysics Letters* 61, no. 4: 567-572.

Jeong, H., B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabasi. 2000. The large-scale organization of metabolic networks. *Nature* 407, no. 6804: 651-654.

Jung-Ying, Lai, Wu Jain-Shing, Chen Shih-Jen, Wu Chia-Huan, and Yang Chung-Huang. 2008. Designing a taxonomy of web attacks. Paper presented at Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology - Volume 00.

Kristensen, L.M., S. Christensen, and K. Jensen. 1998. The practitioner's guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer* 2: 98-132.

Kristensen, Lars Michael and Soren Christensen. 2004. Implementing coloured petri nets using a functional programming language. *Higher Order Symbol. Comput.* 17, no. 3: 207-243.

Labovitz, C., A. Ahuja, A. Bose, and F. Jahanian. 2001. Delayed internet routing convergence. *Ieee-Acm Transactions on Networking* 9, no. 3: 293-306.

Lai, Ying-Cheng, Adilson E. Motter, and Takashi Nishikawa. 2004. Attacks and cascades in complex networks. In *Complex Networks*:299-310: Springer Berlin / Heidelberg.

Lakhina, Anukool, John W. Byers, Mark Crovella, and Ibrahim Matta. 2002. On the geographic location of internet resources. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*:249-250. Marseille, France: ACM Press.

Latora, V. and M. Marchiori. 2001. Efficient behavior of small-world networks. *Physical Review Letters* 87, no. 19.

_____. 2004a. How the science of complex networks can help developing strategies against terrorism. *Chaos Solitons & Fractals* 20, no. 1: 69-75.

_____. 2005. Vulnerability and protection of infrastructure networks. *Physical Review E* 71, no. 1: 4.

Latora, Vito and Massimo Marchiori. 2004b. How the science of complex networks can help developing strategies against terrorism. *Chaos, Solitons & Fractals* 20, no. 1: 69-75.

Lazaroff, Mark and David Snowden. 2006. Anticipatory models for counter-terrorism. In *Emergent information technologies and enabling policies for counter-terrorism*, ed. Robert L. Popp and John Yen:51-73. Hoboken, New Jersey: Wiley-Interscience.

Lee, Keunsoo, Juhyun Kim, Ki Hoon Kwon, Younggoo Han, and Sehun Kim. 2008. Ddos attack detection method using cluster analysis. *Expert Systems with Applications* 34, no. 3: 1659-1665.

Leguay, J., M. Latapy, T. Friedman, and K. Salamatian. 2007. Describing and simulating internet routes. *Computer Networks* 51, no. 8: 2067.

Lerner, V. S. 2004. Introduction to information systems theory: Concepts, formalism and applications. *International Journal of Systems Science* 35, no. 7: 405-424.

Leung, C. C. and H. F. Chau. 2007. Weighted assortative and disassortative networks model. *Physica A: Statistical Mechanics and its Applications* 378, no. 2: 591-602.

Li, P., B. H. Wang, H. Sun, P. Gao, and T. Zhou. 2008. A limited resource model of fault-tolerant capability against cascading failure of complex network. *European Physical Journal B* 62, no. 1: 101-104.

Liljenstam, M., Y. Yuan, B. J. Premore, and D. Nicol. 2002. A mixed abstraction level simulation model of large-scale internet worm infestations. In *10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*:109.

Liljenstam, Michael, Jason Liu, and David Nicol. 2003. Simulation of large scale networks ii: Development of an internet backbone topology for large-scale network simulations. In *WSC '03: Proceedings of the 35th conference on Winter simulation*:694-702. New Orleans, Louisiana: Winter Simulation Conference.

Lipson, Howard F. 2002. *Tracking and tracing cyber-attacks: Technical challenges and global policy issues*. Carnegie Melon Software Engineering Institute, CERT Coordination Center.

Lopez, E., Parshani, R., Cohen, R., Carmi, S., & Havlin, S. 2007. Limited path percolation in complex networks. *Physical Review Letters* 99, no. 18: 188701-4.

Lu, K., D. Wu, H. Fan, S. Todorovic, and A. Nucci. 2007. Robust and efficient detection of ddos attacks for large-scale internet. *Computer Networks* 51, no. 18: 5036-5056.

Macdonald, P. J., E. Almaas, and A. L. Barabasi. 2005. Minimum spanning trees of weighted scale-free networks. *Europhysics Letters* 72, no. 2: 308-314.

Mahadevan, Priya, Dmitri Krioukov, Marina Fomenkov, Xenofontas Dimitropoulos, k c claffy, and Amin Vahdat. 2006. The internet as-level topology: Three data sources and one definitive metric. *SIGCOMM Comput. Commun. Rev.* 36, no. 1: 17-26.

Mahadevan, Priya, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Xenofontas Dimitropoulos, kc claffy, and Amin Vahdat. 2005. *Lessons from three views of the internet topology: Technical report* San Diego Supercomputer Center, University of California, San Diego: Cooperative Association for Internet Data Analysis - CAIDA.

Markopoulou, A., G. Iannaccone, S. Bhattacharyya, C. N. Chuah, Y. Ganjali, and C. Diot. 2008. Characterization of failures in an operational ip backbone network. *Ieee-Acm Transactions on Networking* 16, no. 4: 749-762.

Michalis, Faloutsos, Faloutsos Petros, and Faloutsos Christos. 1999. On power-law relationships of the internet topology. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*:251-262. Cambridge, Massachusetts, United States: ACM Press.

Mirkovic, Jelena and Peter Reiher. 2004. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.* 34, no. 2: 39-53.

Mizrak, A. T., Y. C. Cheng, K. Marzullo, and S. Savage. 2006. Detecting and isolating malicious routers:230-244: Ieee Computer Soc.

Mizrak, A. T., S. Savage, and K. Marzullo. 2008. Detecting compromised routers via pocket forwarding behavior. *IEEE Network* 22, no. 2: 34-39.

Moore, Andrew P., Robert J. Ellison, and Richard C. Linger. 2001. *Attack modeling for information security and survivability*. Pittsburgh, PA: Carnegie Mellon University Software Engineering Institute.

Moreno, Y., J. B. Gomez, and A. F. Pacheco. 2002. Instability of scale-free networks under node-breaking avalanches. *Europhysics Letters* 58, no. 4: 630-636.

Motter, A. E. 2004. Cascade control and defense in complex networks. *Physical Review Letters* 93, no. 9: 4.

Motter, A. E. and Y. C. Lai. 2002. Cascade-based attacks on complex networks. *Physical Review E* 66, no. 6.

Motter, A. E., M. A. Matias, J. Kurths, and E. Ott. 2006. Dynamics on complex networks and applications. *Physica D-Nonlinear Phenomena* 224, no. 1-2: VII-VIII.

Newman, M. E. J. 2002. Assortative mixing in networks. *Physical Review Letters* 89, no. 20.

_____. 2003. The structure and function of complex networks. *Siam Review* 45, no. 2: 167-256.

Olalekan, Adeyinka. 2008. Internet attack methods and internet security technology. Paper presented at Proceedings of the 2008 Second Asia International Conference on Modelling \& Simulation (AMS).

Ole Martin Dahl and Stephen D. Wolthusen. 2006. Modeling and execution of complex attack scenarios using interval timed colored petri nets Paper presented at Fourth IEEE International Workshop on Information Assurance (IWIA'06).

Overill, Richard E. 2007. Computational immunology and anomaly detection. *Information Security Technical Report* 12, no. 4: 188-191.

Papadimitratos, P. and Z. J. Haas. 2002. Securing the internet routing infrastructure. *Ieee Communications Magazine* 40, no. 10: 60-68.

Paxson, V. 2001. An analysis of using reflectors for distributed denial-of-service attacks. *Computer Communication Review* 31, no. 3: 38-47.

Peng, T., C. Leckie, and K. Ramamohanarao. 2007a. Survey of network-based defense mechanisms countering the dos and ddos problems. *Acm Computing Surveys* 39, no. 1: 42.

Peng, Tao, Christopher Leckie, and Kotagiri Ramamohanarao. 2007b. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.* 39, no. 1: 3.

Pietsch, W. 2006. Derivation of the percolation threshold for the network model of barabasi and albert. *Physical Review E* 73, no. 6: 7.

Piraveenan, M., M. Prokopenko, and A. Y. Zomaya. 2008. Local assortativeness in scale-free networks. *Epl* 84, no. 2: 6.

_____. 2009. Assortativeness and information in scale-free networks. *European Physical Journal B* 67, no. 3: 291-300.

Qin, Q., Z. P. Wang, F. Zhang, and P. Y. Xu. 2008. Evolving scale-free network model. *International Journal of Modern Physics B* 22, no. 13: 2138-2148.

Rattray, Gregory J. 2001a. The cyber threat. In *The terrorism threat and u.S. Government response: Operational and organizational factors* 79-119. US Air Force Academy: USAF Institute for National Security Studies

_____. 2001b. The cyber threat:79-119. US Air Force Academy: USAF Institute for National Security Studies

Ravi, Kumar, Raghavan Prabhakar, Rajagopalan Sridhar, D. Sivakumar, Tompkins Andrew, and Upfal Eli. 2000. The web as a graph. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*:1-10. Dallas, Texas, United States: ACM Press.

Redner, S. 1998. How popular is your paper? An empirical study of the citation distribution. *European Physical Journal B* 4, no. 2: 131-134.

Rezaei, B. A., N. Sarshar, V. P. Roychowdhury, and P. O. Boykin. 2007. Disaster management in power-law networks: Recovery from and protection against intentional attacks. *Physica a-Statistical Mechanics and Its Applications* 381: 497-514.

Richardson, R. 2008. 2008 CSI/FBI computer crime and security survey. no. February 10, 2006. http://www.gocsi.com/ (accessed March, 2008).

Rocketfuel: An ISP topology mapping engine. http://www.cs.washington.edu/research/networking/rocketfuel/ (accessed December, 2008.

Rosen, R. 1985. *Anticipatory systems - philosophical, mathematical and methodological foundations* Pergamon Press.

Saffre, F., H. Jovanovic, C. Hoile, and S. Nicolas. 2004. Scale-free topology for pervasive networks. *BT Technology Journal* 22, no. 3: 200-208.

Salla, Vamsi. 2005. Error and attack tolerance of complex real networks. *MAI* 44, no. 04: 90.

Sanchirico, A. and M. Fiorentino. 2008. Scale-free networks as entropy competition. *Physical Review E* 78, no. 4.

Schreiber, T. 2000. Measuring information transfer. *Physical Review Letters* 85, no. 2: 461-464.

Sedgewick, Robert. 1983. *Algorithms*. Edited by Michael A. Harrison. Addison-wesley in computer science. Reading, Massachusetts: Addison-Wesley

Sekiyama, Kosuke and Hirohisa Araki. 2007. Network topology reconfiguration against targeted and random attack. In *Self-organizing systems*, 4725/2007:119-130: Springer Berlin / Heidelberg.

Shannon, C., D. Moore, D. J. Brown, G. M. Voelker, and S. Savage. 2006. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.* 24, no. 2: 115-139.

Siganos, Georgos, Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. 2003. Power laws and the as-level internet topology. *IEEE/ACM Trans. Netw.* 11, no. 4: 514-524.

Sole, Ricard V. and Sergi Valverde. 2004. Information theory of complex networks: On evolution and architectural constraints. In *Lecture notes in physics: Complex networks*, ed. E. Ben-Naim, H. Frauenfelder and Z. Toroczkai, 650:189-207. Berlin, Germany: Springer-Verlag.

Spring, N., R. Mahajan, D. Wetherall, and T. Anderson. 2004. Measuring ISP topologies with rocketfuel. *Networking, IEEE/ACM Transactions on* 12, no. 1: 2-16.

Spring, Neil, Ratul Mahajan, and David Wetherall. 2002. Measuring ISP topologies with rocketfuel. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, 32:133-145: ACM Press.

Srivastav, A., A. Ray, and S. Gupta. 2009. An information-theoretic measure for anomaly detection in complex dynamical systems. *Mechanical Systems and Signal Processing* 23, no. 2: 358-371.

Stephenson, P. 2006. Towards improving attribution confidence in cyber attacks. *Journal of Cyber Conflict Studies* 1, no. 1: 48-54.

Stephenson, P. R. and P. S. Prueitt. 2005. Towards a theory of cyber attack mechanics. Paper presented at IFIP wg 11.9 Digital Forensics, First Annual Conference, Orlando, Florida.

Sun, S., Z. X. Liu, Z. Q. Chen, and Z. Z. Yuan. 2007. Error and attack tolerance of evolving networks with local preferential attachment. *Physica a-Statistical Mechanics and Its Applications* 373: 851-860.

Tang, Yongping and Thomas E. Daniels. 2005. A simple framework for distributed forensics. Paper presented at Proceedings of the Second International Workshop on Security in Distributed Computing Systems (SDCS) (ICDCSW'05) - Volume 02.

Toby, Ehrenkranz and Li Jun. 2009. On the state of ip spoofing defense. *ACM Trans. Internet Technol.* 9, no. 2: 1-29.

Tsunoda, Hiroshi, Kohei Ohta, Atsunori Yamamoto, Nirwan Ansari, Yuji Waizumi, and Yoshiaki Nemoto. 2008. Detecting drdos attacks by a simple response packet confirmation mechanism. *Computer Communications* 31, no. 14: 3299-3306.

U.S. Congress. House. House Armed Services Committee. 2005. H*earing on "Cyber Security, Information Assurance and Information Security", Statement of Eugene h. Spafford*.

U. S. Office of Science and Technology Policy. National Science and Technology Council. 2006. *Federal plan for Cyber Security and Information Assurance Research and Development*. Report by the Interagency Working Group on Cyber Security and Information Assurance.

Wang, B., H. W. Tang, C. H. Guo, and Z. L. Xiu. 2006. Entropy optimization of scale-free networks' robustness to random failures. *Physica a-Statistical Mechanics and Its Applications* 363, no. 2: 591-596.

Wang, F., Z. Q. M. Mao, J. Wang, L. X. Gao, and R. Bush. 2006. A measurement study on the impact of routing events on end-to-end internet path performance:375-386: Assoc Computing Machinery.

Wang, J. W. and L. L. Rong. 2008. Effect attack on scale-free networks due to cascading failures. *Chinese Physics Letters* 25, no. 10: 3826-3829.

_____. 2009a. Edge-based-attack induced cascading failures on scale-free networks. *Physica a-Statistical Mechanics and Its Applications* 388, no. 8: 1731-1737.

_____. 2009b. A model for cascading failures in scale-free networks with a breakdown probability. *Physica a-Statistical Mechanics and Its Applications* 388, no. 7: 1289-1298.

Wang, J. W., L. L. Rong, L. Zhang, and Z. Z. Zhang. 2008. Attack vulnerability of scale-free networks due to cascading failures. *Physica a-Statistical Mechanics and Its Applications* 387, no. 26: 6671-6678.

Wang, L. N., J. L. Guo, H. X. Yang, and T. Zhou. 2009. Local preferential attachment model for hierarchical networks. *Physica a-Statistical Mechanics and Its Applications* 388, no. 8: 1713-1720.

Wang, X. A., S. G. Guan, and C. H. Lai. 2009. Protecting infrastructure networks from cost-based attacks. *New Journal of Physics* 11: 9.

Wu, J., H. Z. Deng, Y. J. Tan, Y. Li, and D. Z. Zhu. 2007. Attack vulnerability of complex networks based on local information. *Modern Physics Letters B* 21, no. 16: 1007-1014.

Wu, J., H. Z. Deng, Y. J. Tan, and D. Z. Zhu. 2007. Vulnerability of complex networks under intentional attack with incomplete information. *Journal of Physics a-Mathematical and Theoretical* 40, no. 11: 2665-2671.

Wu, Z. H. and H. J. Fang. 2008. Cascading failures of complex networks based on two-step degree. *Chinese Physics Letters* 25, no. 10: 3822-3825.

Xu, J. and X. F. Wang. 2005. Cascading failures in scale-free coupled map lattices. *Physica a-Statistical Mechanics and Its Applications* 349, no. 3-4: 685-692.

Yegneswaran, Vinod, Paul Barford, and Johannes Ullrich. 2003. Internet intrusions: Global characteristics and prevalence. In *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*:138-147. San Diego, CA, USA: ACM Press.

Yook, Soon-Hyung, Hawoong Jeong, and Albert-Laszlo Barabasi. 2002. Modeling the internet's large-scale topology. *Proceedings of the National Academy of Sciences of the United States of America* 99, no. 21: 13382-13386.

Yu, M., W. D. Chen, and X. Y. Zhou. 2008. Adaptive detection of syn flooding attacks at source-end networks. *Chinese Journal of Electronics* 17, no. 1: 141-144.

Zhang, Shaojun, Jianhua Li, Xiuzhen Chen, and Lei Fan. 2008. Building network attack graph for alert causal correlation. *Computers & Security* 27, no. 5-6: 188-196.

Zhang, Z. Z., S. G. Zhou, T. Zou, and G. S. Chen. 2008. Fractal scale-free networks resistant to disease spread. *Journal of Statistical Mechanics-Theory and Experiment*: 11.

Zhao, L., K. H. Park, Y. C. Lai, and N. Ye. 2005. Tolerance of scale-free networks against attack-induced cascades. *Physical Review E* 72, no. 2: 4.

Ziviani, A., A. T. A. Gomes, M. L. Monsores, and P. S. S. Rodrigues. 2007. Network anomaly detection using nonextensive entropy. *Ieee Communications Letters* 11, no. 12: 1034-1036.

APPENDIX

Appendix A

CPN Declarations

```
(* Base Declarations *)
    colset INT = int declare output_col;
    colset BOOL = bool;
    colset E = with e;
    colset LIST = list INT;
    colset NODE = INT declare ms;
    colset ORPHAN_NODE = NODE declare ms;
    colset NEIGH_NODE = INT declare ms;
    colset ATTACKED_NODE = NODE declare ms;
    colset TEMP_ORPHAN_NODE = NODE declare ms;
    colset PERM_ORPHAN_NODE = NODE declare ms;
    colset NODE_TYPE = INT declare ms;
    colset NP = product NODE * NODE declare ms;
    colset NP_LIST = list NP declare ms;
    colset STATUS_INT = int with 0..4;
    colset NP_ORPHAN_STATUS = INT declare ms;
    colset NODE_STATUS = STATUS_INT declare ms;
    colset NODE_REMOVED_STATUS = BOOL declare ms;
    colset REMOVED_NODE = NODE declare ms;
    colset REMOVED_NODE_LIST = list REMOVED_NODE declare ms;
    colset NEIGH_NODE_LIST = list NEIGH_NODE declare ms;
    colset NEIGH_NODE_LIST_LIST = list NEIGH_NODE_LIST declare ms;
    colset NODE_THRESHOLD = INT declare ms;
    colset NODE_DEGREE = INT declare ms;
    colset LINK_PROB_OUT_OF_100 = INT declare ms;
    colset NEIGH_NODE_PROB =
      product NEIGH_NODE * LINK_PROB_OUT_OF_100 declare ms;
    colset NEIGH_NODE_PROB_LIST = list NEIGH_NODE_PROB declare ms;
    colset NODE_RECORD1 =
          record id:NODE * neigh:NEIGH_NODE_LIST * degree:NODE_DEGREE *
          threshold:NODE_THRESHOLD * status:NODE_STATUS *  ts:INT
          declare output_col,ms;
    colset NODE_RECORD1_LIST = list NODE_RECORD1 declare output_col;
    colset NODE_RECORD2 =
         record oid:NODE * odegree:NODE_DEGREE *
         nnDegreeProbList:NEIGH_NODE_PROB_LIST * nnDegreeTotal:INT *
         newLink:NEIGH_NODE declare ms;
    colset NODE_RECORD2_LIST = list NODE_RECORD2 declare ms;
```

```
(* Product Declarations with Multiple Colorsets *)
    colset NODExNPxREMOVED_NODE_LIST =
          product NODE * NP *      REMOVED_NODE_LIST declare ms;
    colset NPxNP_LIST = product NP * NP_LIST declare ms;
    colset ORPHAN_NODExNODE_STATUS =
          product ORPHAN_NODE * NODE_STATUS declare ms;
    colset NP_STATUSxNPxNODE =
           product NP_ORPHAN_STATUS * NP * NODE declare ms;
    colset NPxNP_ORPHAN_STATUS = product NP * NP_ORPHAN_STATUS declare ms;
    colset NPxNODE_REMOVED_STATUS =
          product NP * NODE_REMOVED_STATUS declare ms;
    colset NODE_RECORD1xNODE_RECORD1_LIST =
          product NODE_RECORD1 * NODE_RECORD1_LIST declare ms;
    colset NODExNODE_RECORD1_LIST =
          product NODE * NODE_RECORD1_LIST declare ms;
    colset ORPHAN_NODExNODE_STATUSxNODE_RECORD1_LIST =
          product ORPHAN_NODE * NODE_STATUS * NODE_RECORD1_LIST declare
ms;
    colset NODE_RECORD1xNODE_RECORD1_LISTxNODE_RECORD2 =
          product NODE_RECORD1 * NODE_RECORD1_LIST *
         NODE_RECORD2 declare ms;
    colset REMOVED_NODExNODE_RECORD1xNODE_RECORD1_LIST =
           product REMOVED_NODE * NODE_RECORD1 * NODE_RECORD1_LIST
          declare ms;
    colset REMOVED_NODExNODE_RECORD1_LIST =
          product REMOVED_NODE * NODE_RECORD1_LIST declare ms;
    colset REMOVED_NODExNODE_RECORD1_LISTxREMOVED_NODE_LIST  =
          product REMOVED_NODE * NODE_RECORD1_LIST *
          REMOVED_NODE_LIST   declare ms;
    colset REMOVED_NODExREMOVED_NODE_LIST =
          product REMOVED_NODE * REMOVED_NODE_LIST declare ms;
    colset NODE_RECORD2xNODE_RECORD1_LIST =
           product NODE_RECORD2 * NODE_RECORD1_LIST declare ms;
    colset NODE_RECORD2xNODE = product NODE_RECORD2 * NODE declare ms;
    colset INTxINT = product INT * INT declare ms;
    colset NODExBOOL = product NODE * BOOL declare ms;
    colset NODE_RECORD1_LISTxTEMP_ORPHAN_NODE =
          product NODE_RECORD1_LIST  * TEMP_ORPHAN_NODE declare ms;
    colset TEMP_ORPHAN_NODExNODExNODE_RECORD1_LIST =
           product TEMP_ORPHAN_NODE * NODE * NODE_RECORD1_LIST declare ms;

(* Variables *)
    var nplist1,nplist2,nplist3,nplist4,nplist5,nplist6:NP_LIST;
    var neighNodeListList1, neighNodeListList2: NEIGH_NODE_LIST_LIST;
    var rlist1,rlist2,rlist3,rlist4,rlist5,rlist6,rlist7,rlist8,rlist9:REMOVED_NODE_LIST;
```

```
    var nodeRecord1List1,nodeRecord1List2,nodeRecord1List3,nodeRecord1List4,
      nodeRecord1List5,nodeRecord1List6:NODE_RECORD1_LIST;
    var nodeRecord2List1,nodeRecord2List2,nodeRecord2List3,nodeRecord2List4,
      nodeRecord2List5:NODE_RECORD2_LIST;
    var b1,b2,b3,b4,criticalNodeCount,timeStamp:INT;
    var n1,n2:NODE;
    var n3_n4:BOOL;
    var tempOrphanNode1,tempOrphanNode2,tempOrphanNode3,
      tempOrphanNode4:TEMP_ORPHAN_NODE;
    var
permOrphanNode1,permOrphanNode2,permOrphanNode3:PERM_ORPHAN_NODE;
    var node,node1,node2,node3,node4:NODE;
    var nodeRemovedStatus1,nodeRemovedStatus2:NODE_REMOVED_STATUS;
    var npStatus1,npStatus2:NP_ORPHAN_STATUS;
    var nodePair1,nodePair2:NP;
    var nodeStatus1,nodeStatus2:NODE_STATUS;
    var evalNodePair1,evalNodePair2:NP;
    var removedNode,removedNode1,removedNode2,removedNode3:REMOVED_NODE;
    var
nodeRecord1,nodeRecord2,nodeRecord3,nodeRecord4,nodeRecord5:NODE_RECORD1;
    var nodeRecord2_1,nodeRecord2_2,nodeRecord2_3,nodeRecord2_4:NODE_RECORD2;

(* Global Variables *)
    globref packets = empty: NEIGH_NODE_LIST ms;
    globref packets1 = empty: NP ms;
    globref outfile = TextIO.stdOut;
    globref packetsCrit = empty: REMOVED_NODE ms;
    globref packetsCritCount = empty: INT ms;
    globref rt = Timer.startRealTimer();
    val PROTECT_LIST = [];
    val NODE_THRESHOLD = 0.1;
    val NODE_THRESHOLD_PROTECT = 5.0;
    val M = 100000000;
    val M1 = 10000;
    val B = 31415821;
    val RANDOM_SEED = 7789;
(* Exceptions *)
    exception notFound of int;
```

Appendix B

CPN Function Code

```
(* Functions *)

 fun getTarget (_,nil) = raise notFound(1)
    | getTarget (target, recordList:NODE_RECORD1_LIST) =
            let
            val b = hd recordList;
            val key = #id(b);
            in
            if (key = target) andalso (List.null recordList = false)
                then b
                else (getTarget (target,tl recordList))
            end;

fun getTarget2 (_,nil) = raise notFound(1)
    | getTarget2 (target, recordList:NODE_RECORD2_LIST) =
            let
            val b = hd recordList;
            val key = #oid(b);
            in
            if (key = target) andalso (List.null recordList = false)
                then b
                else (getTarget2 (target,tl recordList))
            end;

fun mult(p:INT,q:INT):INT =
            let
               val p1 = p div M1;
               val p0 = p mod M1;
               val q1 = q div M1;
               val q0 = q mod M1;
            in
              (((p0*q1+p1*q0) mod M1) * M1 + p0 * q0) mod M
            end;

 fun randomInt(r:INT):INT =
```

```
            let
               val a = (mult(RANDOM_SEED,B)+1) mod M;
            in
               ((a div M1)*r) div M1  (* generate random number between 0 and r-1 *)
            end;

fun simulationTimer (rt) =
             let
               val currentTime = Time.toMilliseconds(Timer.checkRealTimer rt);
               val IntTimeSeconds = Int.fromLarge currentTime;
             in
               IntTimeSeconds
            end;

fun currentTimeSimulation ():INTxINT =
             let
               val currentTime = Date.fromTimeLocal(Time.now());
               val currentHour = Date.hour(currentTime);
               val currentMinute = Date.minute(currentTime);
             in
               (currentHour,currentMinute)
            end;

fun idFound ({id,...}:NODE_RECORD1) = id=10;

fun getPackets()  = (!packets);

fun getPackets1()  = (!packets1);

fun getPacketsCritical() = (!packetsCrit);

fun getPacketsCritCount() = (!packetsCritCount);

fun getCritical() =
            let
              val infileCrit=TextIO.openIn("criticalNodesTokens.txt");
              val message2 = REMOVED_NODE.input_ms(infileCrit);
            in
              packetsCrit := message2;
              TextIO.closeIn(infileCrit);
              ()
            end handle _ => ();

fun getInputList() =
            let
              val infile=TextIO.openIn("nodeNeighborListTokens.txt");
```

```
            val message = NEIGH_NODE_LIST.input_ms(infile);
          in
            packets := message;
            TextIO.closeIn(infile);
            ()
          end handle _ => ();

fun getInputNPList() =
          let
              val infile=TextIO.openIn("nodePairTokens.txt");
              val message1 = NP.input_ms(infile);
          in
             packets1 := message1;
            TextIO.closeIn(infile);
            ()
          end handle _ => ();

fun getCriticalCount() =
          let
            val infile=TextIO.openIn("criticalNodeCount.txt");
            val message2 = INT.input_ms(infile);
          in
            packetsCritCount := message2;
            TextIO.closeIn(infile);
            ()
          end handle _ => ();

fun releaseOneNodePair (nplist):NPxNP_LIST =
          let
           val listLength = List.length(nplist);
           val randomIndex = randomInt(listLength);
           val npair = List.nth(nplist, randomIndex);
           val newList = rm npair nplist;
          in
           (npair,newList)
          end;

fun releaseOneCriticalNode (rlist):REMOVED_NODExREMOVED_NODE_LIST =
          let
            val listLength = List.length(rlist);
            val randomIndex = randomInt(listLength);
            val remNode = List.nth(rlist, randomIndex);
            val newList = rm remNode rlist;
          in
            (remNode,newList)
          end;
```

```
fun updateRecordList2 (recordList2,record2:NODE_RECORD2) =
                let
                   fun newMember (recordList2,record2):BOOL =  (mem recordList2
        record2);
                   fun intializeR2 (recordList2,record2) =
                   let
                      val target = #oid(record2);
                      val r2 = getTarget2
        (target:ORPHAN_NODE,recordList2):NODE_RECORD2 handle notFound(1) =>
        record2;
                 in
                    if newMember(recordList2,record2)= true then
        {oid=0,odegree=0,nnDegreeProbList= [],nnDegreeTotal=0,newLink=0}
                 else r2
                 end;
                val recordListMinusOldRecord = rm
        (intializeR2(recordList2,record2):NODE_RECORD2)  recordList2;
             in
                [intializeR2 (recordList2,record2)] ^^ recordListMinusOldRecord
            end;

fun updateStatus (r,status) = NODE_RECORD1.set_status r status;

fun updateDegree(r,degree) = NODE_RECORD1.set_degree r degree;

fun updateTimeStamp (r,timestamp) = NODE_RECORD1.set_ts r timestamp;

fun determineCascade (nodeRecord:NODE_RECORD1):BOOL =
            let
              val currentThreshold = #threshold(nodeRecord);
              val currentDegree = #degree(nodeRecord);
            in
              if (currentDegree >= currentThreshold) then true else false
            end;

fun determine_3_4(currentRecord:NODE_RECORD1):NODE_RECORD1 =
          let
            val currentNode = #id(currentRecord);
            val currentDegree = #degree(currentRecord);
            val currentThreshold = #threshold(currentRecord);
           in
            (
              if currentDegree = 0
              then
                updateStatus(currentRecord,3)
```

```
            else
              if (currentDegree >= currentThreshold)
              then
                updateStatus(currentRecord,4)
              else currentRecord
          )
        end;

fun outputNodeRecord (outfile,nodeRecord3:NODE_RECORD1) =
          let
              val (currentHour,currentMinute) = currentTimeSimulation();
          in
            (
                  NODE.output(outfile,#id(nodeRecord3));
                  TextIO.output(outfile,"\t");
                  NODE_STATUS.output(outfile,#status(nodeRecord3));
                  TextIO.output(outfile,"\t");
                  NODE_DEGREE.output(outfile,#degree(nodeRecord3));
                  TextIO.output(outfile,"\t");
                  NODE_THRESHOLD.output(outfile,#threshold(nodeRecord3));
                  TextIO.output(outfile,"\t");
                  INT.output(outfile,#ts(nodeRecord3));
                  TextIO.output(outfile,"\t");
                  INT.output(outfile,currentHour);
                  TextIO.output(outfile,"\t");
                  INT.output(outfile,currentMinute);

                  TextIO.output(outfile,"\t");
                  NEIGH_NODE_LIST.output(outfile,#neigh(nodeRecord3));

                  TextIO.output(outfile,"\n")

            )
          end;

fun updateNeighList
(outfile,neighborNodeRecord,newNeighList,newNeighDegree):NODE_RECORD1 =
          let
            val neighborListBefore = #neigh(neighborNodeRecord);
            val neighborNodeRecord1= NODE_RECORD1.set_neigh neighborNodeRecord
newNeighList;
            val neighborNodeRecord2= NODE_RECORD1.set_degree
neighborNodeRecord1 newNeighDegree;
            val currentNode = #id(neighborNodeRecord2);
            val neighborNodeRecord3= determine_3_4(neighborNodeRecord2);
```

```
            val neighborNodeRecord4= updateTimeStamp
(neighborNodeRecord3,simulationTimer(!rt));
            val NeighNodeStatus = #status(neighborNodeRecord4);
            val neighborListAfter = #neigh(neighborNodeRecord4);
          in
                (
                  if NeighNodeStatus = 3 orelse NeighNodeStatus = 4
                  then
                      (
                        outputNodeRecord(outfile,neighborNodeRecord4);
                        neighborNodeRecord4
                       )
                  else  if neighborListAfter <> neighborListBefore
                        then
                        (
                          outputNodeRecord(outfile,neighborNodeRecord4);
                          neighborNodeRecord4
                         )
                        else neighborNodeRecord4
                 )
          end;

fun intializeOutputNodeRecord (outfile,nodeRecord3:NODE_RECORD1) =
          let
              val (currentHour,currentMinute) = currentTimeSimulation();
          in

          (
                    NODE.output(outfile,#id(nodeRecord3));
                    TextIO.output(outfile,"\t");
                    NODE_STATUS.output(outfile,#status(nodeRecord3));
                    TextIO.output(outfile,"\t");
                    NODE_DEGREE.output(outfile,#degree(nodeRecord3));
                    TextIO.output(outfile,"\t");
                    NODE_THRESHOLD.output(outfile,#threshold(nodeRecord3));
                    TextIO.output(outfile,"\t");
                    INT.output(outfile,#ts(nodeRecord3));
                    TextIO.output(outfile,"\t");
                    INT.output(outfile,currentHour);
                    TextIO.output(outfile,"\t");
                    INT.output(outfile,currentMinute);

                    TextIO.output(outfile,"\t");
                    NEIGH_NODE_LIST.output(outfile,#neigh(nodeRecord3));

                    TextIO.output(outfile,"\n")
```

```
            )
        end;


fun closeFile(outfile) =  TextIO.closeOut(outfile);

fun checkForProtection(node):BOOL =
        if mem PROTECT_LIST node then true else false;

fun removeNodeFromNNList
(outfile,target:NEIGH_NODE,recordList:NODE_RECORD1_LIST):NODE_RECORD1 =
        let
           fun protectNode(neighNode:NODE,neighNodeList) =
           if
            checkForProtection(neighNode)
           then
            neighNodeList
           else
            rmall target (neighNodeList);

           val neighborNodeRecord = (hd recordList):NODE_RECORD1;
           val neighNode = #id(neighborNodeRecord);
           val oldNeighList = #neigh(neighborNodeRecord);
           val neighList = protectNode(neighNode,oldNeighList);
           val neighDegree = length neighList;
           val neighNodeStatus = #status(neighborNodeRecord);
        in
           (
                  (* only process neighbor list of active nodes *)
            if neighNodeStatus = 0
            then
                  updateNeighList (outfile,neighborNodeRecord,neighList,neighDegree)
            else
                  neighborNodeRecord  (* no change *)
           )
        end;

fun traverseRecordList (_,_,nil) = nil
    |  traverseRecordList (outfile,target,recordList) =
         [removeNodeFromNNList(outfile,target,recordList)] ^^ traverseRecordList
(outfile,target,(tl recordList));

fun computeNodeDegreeThreshold (node:NODE,nodeDegree:INT):INT =
       let
```

```
          val nodeDegreeReal = Real.fromInt nodeDegree;
          val nodeDegreeThresholdReal = nodeDegreeReal * NODE_THRESHOLD;
          val nodeDegreeThreshold = Real.ceil nodeDegreeThresholdReal + nodeDegree;

          val nodeThresholdProtect =
                              NODE_THRESHOLD +     NODE_THRESHOLD_PROTECT;
          val nodeDegreeThresholdRealProtect = nodeDegreeReal * nodeThresholdProtect;
          val nodeDegreeThresholdProtect = Real.ceil nodeDegreeThresholdRealProtect +
nodeDegree;
        in
         (
          if checkForProtection(node) = true
         then
            nodeDegreeThresholdProtect
         else
         nodeDegreeThreshold
        )
        end;

fun computeNodeProfileRecordStep1 (outfile,nlist) =
            let
              val nodeDegree = length (tl nlist);
              val node = hd nlist;
              val r1 = {
                  id = (hd nlist):NODE,
                 neigh = (tl nlist):NEIGH_NODE_LIST,
                 degree = nodeDegree:NODE_DEGREE,
                 threshold = computeNodeDegreeThreshold(node,nodeDegree),
                 status = 0,ts=0
                  };
            in
               intializeOutputNodeRecord(outfile,r1);
               r1
            end;

fun  updateNodeProfileRecordStep1A (outfile,recordList) =
            let
              fun checkNull (recordList) =
               if List.null recordList = false then
               [computeNodeProfileRecordStep1 (outfile, hd recordList)]
               else [];
            in
              if List.null recordList = true then []
             else
               computeNodeProfileRecordStep1 (outfile,hd recordList) ::
         (updateNodeProfileRecordStep1A (outfile,tl recordList))
```

```
            end;

fun updateNodeProfileRecordThreshold {id,neigh,degree,threshold,status,ts} =
    {id = id,neigh=neigh,degree=degree,threshold = computeNodeDegreeThreshold(degree),
    status=status,ts=0};

 fun insertRecord (newRecord:NODE_RECORD1,recordList:NODE_RECORD1_LIST ) =
     ins_new recordList newRecord;

fun removeRecord (target,rs) =
    (*Find and remove the first instance of the target
    Assumes that this function is used with the insertRecord function, the
    insertRecord function adds the NEW record to the end of the recordList, thus the old re
    record is the first occurence of target in the record list*)
            let
                        val r = getTarget (target,rs):NODE_RECORD1;
            in
                        rm r rs
           end;

fun traverseAndUpdateStatus (nil) = nil
    |       traverseAndUpdateStatus (recordList) =
            let
              val targetRecord = hd recordList;
              val neighList = #neigh(targetRecord);
            in
             (
              if List.null neighList = true then
                        [updateStatus (targetRecord,3)] ^^ traverseAndUpdateStatus (tl
        recordList)
              else
                        [targetRecord] ^^ traverseAndUpdateStatus (tl recordList)
            )
             end;

fun recordUpdateForStatus_1
(outfile,s,removedNodeList,recordList:NODE_RECORD1_LIST):REMOVED_NODExNO
DE_RECORD1_LISTxREMOVED_NODE_LIST  =
           let
                val (removedNode,removedNodeListMinus) =
                   releaseOneCriticalNode (removedNodeList);
                val targetRecord = getTarget(removedNode,recordList):NODE_RECORD1;
                val targetRecord1 = updateStatus(targetRecord,s);
                val targetRecord2 = updateTimeStamp (targetRecord1,simulationTimer(!rt));
                val recordListMinus = removeRecord(removedNode, recordList);
                val newRecordList = insertRecord(targetRecord2,recordListMinus);
```

```
                val updatedNeighLists =
traverseRecordList(outfile,removedNode,newRecordList)
            in
              (
                outputNodeRecord(outfile,targetRecord2);
                (removedNode,updatedNeighLists,removedNodeListMinus)
              )
            end;
fun updateNodeRecord
(s:INT,rs:NODE_RECORD1_LIST,TPorphanNode:ORPHAN_NODE):NODE_RECORD1
=
            let
                 val updatedRecord1= getTarget
(TPorphanNode:ORPHAN_NODE,rs):NODE_RECORD1
                 val updatedRecord2 = updateStatus (updatedRecord1,s);
                 val updatedRecord3 = updateTimeStamp
(updatedRecord2,simulationTimer(!rt));
                 val theNeighList = #neigh(updatedRecord3);
                 val d = length theNeighList;
                 val updatedRecord = updateDegree (updatedRecord3,d);
             in
                  updatedRecord
            end;

fun updateRemovedNodeAndDeleteFromNeighLists
(outfile,targetRecord:NODE_RECORD1,recordList:NODE_RECORD1_LIST) =
                 let
                   val targetNode = #id(targetRecord);
                   val nodeStatus = #status(targetRecord);
                 in
                   (
                    if nodeStatus = 3 orelse nodeStatus = 4
                    then
                       (
                         outputNodeRecord(outfile,targetRecord);
                         closeFile(outfile);
                         traverseRecordList(outfile,targetNode,recordList)
                       )
                    else
                        recordList
                   )
                 end;

fun checkFor3_4(currentNode:NODE,recordListBefore:NODE_RECORD1_LIST):BOOL =
                  let
```

```
                val currentRecord = getTarget
        (currentNode:NODE,recordListBefore):NODE_RECORD1;
                val currentDegree = #degree(currentRecord);
                val currentThreshold = #threshold(currentRecord);
            in
              (
                if currentDegree = 0
                then
                  true
                else
                  if (currentDegree >= currentThreshold)
                  then
                    true
                  else false
              )
            end;

fun updateDB_Input_NR
(outfile,s:INT,rs:NODE_RECORD1_LIST,newRecord:NODE_RECORD1):NODE_RECOR
D1_LIST=
        let
            val newRecordNode = #id(newRecord);
            val newRecord = updateNodeRecord(s,rs,newRecordNode);
            val newRecordStatus = #status(newRecord);
            val newRecord1 = determine_3_4(newRecord);
            val recordListMinus = removeRecord(newRecordNode, rs);
            val newRecordList = insertRecord(newRecord1,recordListMinus);
            val newRecordList1 = updateRemovedNodeAndDeleteFromNeighLists
(outfile,newRecord1,newRecordList);
        in
            newRecordList1
        end;

fun updateDB_RecordList
(recordList:NODE_RECORD1_LIST,newRecord:NODE_RECORD1):
                                                        NODE_RECORD1_LIST=
        let
            val newRecordNode = #id(newRecord);
            val recordListMinus = removeRecord(newRecordNode, recordList);
            val newRecordList = insertRecord(newRecord,recordListMinus);
        in
            newRecordList
        end;
```

```
fun
processNewLink(outfile,recordListBeforeUpdate:NODE_RECORD1_LIST,r2:NODE_REC
ORD2):NODE_RECORD1_LIST =
            let
                  val newLinkNode = #newLink(r2);
                  val orphanNode = #oid(r2);
                  val targetNodeRecord = getTarget
(newLinkNode:NEIGH_NODE,recordListBeforeUpdate):NODE_RECORD1

                  val oldNeighList = #neigh(targetNodeRecord);
                  val newNeighList = ins oldNeighList orphanNode;
                  val newNeighDegree = #degree(targetNodeRecord) + 1;

                  val updatedRecord = updateNeighList
(outfile,targetNodeRecord,newNeighList,newNeighDegree);
                  val updatedRecordList1 =
updateDB_RecordList(recordListBeforeUpdate,updatedRecord);
            in
                  updatedRecordList1
            end;

fun updateStatusRecord (target,rs,nodeStatus) =
      let
         val r = getTarget (target,rs):NODE_RECORD1;
      in
           updateStatus (r,nodeStatus)
      end;

fun  updateNodeRecordThreshold (recordList) =
    if List.null recordList = true then [] else
    [updateNodeProfileRecordThreshold (hd recordList)] ^^ updateNodeRecordThreshold(tl
recordList);

fun nnLinkProb (nnDegree:INT,nnDegreeTotal:INT):INT =
              let
              val nnDegreeReal = Real.fromInt nnDegree;
              val nnDegreeTotalReal = Real.fromInt nnDegreeTotal;
              val nnProbReal = (nnDegreeReal / nnDegreeTotalReal) * 100.00;
              val nnProb = Real.trunc nnProbReal;
              in
              nnProb
              end;

fun nnDegreeTotal (nil,_):INT = 0
      |    nnDegreeTotal (nnList:NEIGH_NODE_LIST,rs:NODE_RECORD1_LIST):INT =
             let
```

```
                val nn = hd nnList;
                val r = getTarget (nn:NEIGH_NODE,rs):NODE_RECORD1;
                val nnD = #degree(r);
            in
            if List.null nnList = false then
             nnD + nnDegreeTotal (tl
nnList:NEIGH_NODE_LIST,rs:NODE_RECORD1_LIST)
            else 0
            end;

fun createNNProbList (nil,_,_) = nil
      |   createNNProbList (nnList,rs,nnDTotal) =
            let
                val nn = hd nnList;
                val r = getTarget (nn:NEIGH_NODE,rs):NODE_RECORD1
                val nnD = #degree(r);
                val nnProb = nnLinkProb (nnD,nnDTotal);
                val neighNodeStatus = #status(r);
            in
                if neighNodeStatus =2 orelse neighNodeStatus = 0
                then
                   [(nn,nnProb)] ^^ createNNProbList (tl nnList,rs,nnDTotal)
                else
                  createNNProbList (tl nnList,rs,nnDTotal)
            end;

fun  newLink2
(orphanNode,orphanNodeDegree,nnList:NEIGH_NODE_LIST,rs:NODE_RECORD1_LIST)
=
            let
                val nnDTotal = nnDegreeTotal (nnList,rs);
                val nnProbList = createNNProbList (nnList,rs,nnDTotal);
            in
           {oid = orphanNode,odegree = orphanNodeDegree, nnDegreeProbList = nnProbList,
nnDegreeTotal = nnDTotal, newLink = 0}
            end;
    fun newLink1 (orphanNode:TEMP_ORPHAN_NODE,rs:NODE_RECORD1_LIST) =
            let
                val r = getTarget
(orphanNode:TEMP_ORPHAN_NODE,rs):NODE_RECORD1;
                val orphan = #id(r);
                val nnList = #neigh(r);
                val orphanDegree = #degree(r);
            in
               newLink2 (orphan,orphanDegree,nnList,rs)
            end;
```

```
fun buildNewLinkList2 (elementNode,elementCount,count) =
            let
              val count = count + 1;
            in
              (
               if (count > elementCount)
               then
                  nil
               else
                  [elementNode] ^^ buildNewLinkList2(elementNode,elementCount,count)
              )
            end;



fun buildNewLinkList1 (nil,_) = nil
      | buildNewLinkList1
(elementList:NEIGH_NODE_PROB_LIST,currentNewLinkList:LIST) =
         let
           val elementNodeData = hd elementList;
           val elementNode = #1(elementNodeData);
           val elementCount = #2(elementNodeData);
         in
          buildNewLinkList2(elementNode,elementCount,0) ^^ buildNewLinkList1(tl
elementList,currentNewLinkList)
         end;


fun updateNewLink (r,newLinkNode) =
              let
              val r1 = NODE_RECORD2.set_newLink r newLinkNode;
              in
              r1
    end;

fun selectNewLink (r:NODE_RECORD2):NODE_RECORD2xNODE =
        let
          val elementList = #nnDegreeProbList(r);
          val possibleNewLinksList = buildNewLinkList1(elementList,[]);
          val newLinkNode = List.nth(possibleNewLinksList, discrete(0,
List.length(possibleNewLinksList) - 1))
        in
          (updateNewLink (r,newLinkNode),newLinkNode)
        end;
```

```
fun attackedNodePair(node,removedNodesList) = (mem removedNodesList node)

fun getTempOrphan (npair:NP,status):TEMP_ORPHAN_NODE =
    if (status = 1) then #1(npair)
    else #2(npair);
    fun buildNPList (nplist,npair:NP):NP_LIST =
    nplist ^^ [npair];

fun buildRemList (node1,node2,remList,recordList):REMOVED_NODE_LIST =
              let
                val targetRecord1 = getTarget(node1:NODE,recordList):NODE_RECORD1;
                val nodeStatus1 = #status(targetRecord1);
                val targetRecord2 = getTarget(node2:NODE,recordList):NODE_RECORD1;
                val nodeStatus2 = #status(targetRecord2);
              in
                (
                 (
                 if (nodeStatus1 = 3) orelse (nodeStatus1 = 4) orelse (nodeStatus1 = 1)
                then
                   ins_new remList node1
                else
                  remList
                 );
                      (
                if (nodeStatus2 = 3) orelse (nodeStatus2 = 4) orelse (nodeStatus2 = 1)
                then
                   ins_new remList node2
                else
                  remList

                 )
                )
              end;

fun makeRemovedNodeList2
(node,m:REMOVED_NODE_LIST):REMOVED_NODE_LIST = ins_new m node;
    fun createUpdateRemovedNodeList (nil,_) = nil
     | createUpdateRemovedNodeList
(recordList1:NODE_RECORD1_LIST,updateRemovedNodeList) =
         let
            val record1 = (hd recordList1);
         in
           if (#status(record1) = 3) orelse (#status(record1) = 4) then
             [#id(record1)] ^^ updateRemovedNodeList ^^ createUpdateRemovedNodeList (tl
recordList1,updateRemovedNodeList)
           else
```

```
        createUpdateRemovedNodeList (tl recordList1,updateRemovedNodeList)
      end;

fun recordUpdateForStatus_4
(outfile,s,node,recordList:NODE_RECORD1_LIST):NODE_RECORD1_LIST  =
        let
            val targetRecord = getTarget(node,recordList):NODE_RECORD1;
            val targetRecord1 = updateStatus(targetRecord,s);
            val targetRecord2 = updateTimeStamp (targetRecord1,simulationTimer(!rt));
            val recordListMinus = removeRecord(node, recordList);
            val newRecordList = insertRecord(targetRecord2,recordListMinus);
            val updatedNeighLists = traverseRecordList(outfile,node,newRecordList) (*
remove from neighbor lists *)
        in
          (
            outputNodeRecord(outfile,targetRecord2);
            updatedNeighLists
          )
        end;

fun checkForNewLinkCascade(outfile,s,node,recordList:NODE_RECORD1_LIST):
                                                              NODE_RECORD1_LIST =
         let

          in
           (
           if checkFor3_4(node,recordList) = true    (* returns updated record *)
           then
             recordUpdateForStatus_4 (outfile,s,node,recordList)  (* Cascade as result of
new link *)
           else
             recordList   (* no change *)
           )
         end;

fun processAndSelectNewLink
(outfile,orphanNode:TEMP_ORPHAN_NODE,recordListBefore:NODE_RECORD1_LIST):
   TEMP_ORPHAN_NODExNODExNODE_RECORD1_LIST =
            let
               fun checkNewLink
      (outfile,recordListBefore,updatedNR2:NODE_RECORD2):NODE_RECORD1_LIST
      =
                let
                val newLinkNode = #newLink(updatedNR2);
                val orphanNode = #oid(updatedNR2);
                in
```

```
                    if newLinkNode = orphanNode
                    then
                      recordListBefore
                    else
                      processNewLink(outfile,recordListBefore,updatedNR2)
                    end;

fun newLinkSelection(orphanNode,recordListBefore):
                      TEMP_ORPHAN_NODExNODExNODE_RECORD1_LIST=
            let
               val nr2_1 = newLink1(orphanNode,recordListBefore);
               val (updatedNR2,newLinkNode) = selectNewLink(nr2_1)  handle Discrete =>
                  ({oid = orphanNode,odegree = 0, nnDegreeProbList = [], nnDegreeTotal = 0,
newLink = orphanNode},orphanNode);



               val updatedList = checkNewLink (outfile,recordListBefore,updatedNR2);
               val updatedList1 =
checkForNewLinkCascade(outfile,4,newLinkNode,updatedList );   (* check for cascade as
result of newLink *)
            in
                 (orphanNode,newLinkNode,updatedList1)
            end;
      in
           (
            if   checkFor3_4(orphanNode,recordListBefore) = true
            then
              (orphanNode,orphanNode,recordListBefore)  (* No Change *)
            else
              newLinkSelection(orphanNode,recordListBefore)     (* RecordList1 with new
link added due to being selected *)
            )
     end;

fun updateDB_Input_N
(s:INT,rs:NODE_RECORD1_LIST,nodeForUpdate):NODE_RECORD1_LIST =
           let
                val newRecord = updateNodeRecord(s,rs,nodeForUpdate);
                val newRecordNode = #id(newRecord);
                val newRecordStatus = #status(newRecord);
                val recordListMinus = removeRecord(newRecordNode, rs);
                val newRecordList = insertRecord(newRecord,recordListMinus);
           in
                newRecordList
           end;
```

```
fun node3_4(node1,recordList):BOOL =
    let
      val targetRecord = getTarget(node1:NODE,recordList):NODE_RECORD1;
      val nodeStatus = #status(targetRecord);
    in
      (
      if (nodeStatus = 3) orelse (nodeStatus = 4) orelse (nodeStatus = 1)
      then (true)
      else (false)
      )
    end;

fun buildRemList1 (node,remList):REMOVED_NODE_LIST = ins_new remList node;
```

Appendix C

Simulation Run Data File Identification

Each CPN simulation run generates the following files:

File name: RC***xxxy***_6M_7018_Run***z***_SimResults.txt, one file per simulation run.

File-id = 1: identifies file for reference purposes

Description: CPN simulation run output, audit trail of all changes that occur in one CPN simulation run.

Data Format: One record per node.

Column data at time t: Node-id, status, degree, capacity, timestamp(ms), real clock hour, real clock time (hour and minute), list of all neighbor nodes.

Filename Descriptors for file-id number 1 (as defined in Chapters III and IV):
Run Class: *xxx* = (000, 005, 010, 015, 020, ...,050) representing attack classes (in order): (pre-attack, 0.5%, 1.0%, 1.5%, …, 5.0%).

Additional Capacity: *y* = 1 indicates each node's capacity in simulation was 1.1 *pre-attack node degree.

6M
Each simulation run was executed for 6 million CPN execution steps.

7018
Pre-Attack Autonomous System Number (ASN) for Rocketfuel router adjacency data.

Run Type: *z* = 01, 02, 03, 04.

In addition to the intermediate "processing files" generated through execution of the offline Microsoft Visual Basic routines that were developed specifically for this research, other significant data files associated with each simulation run included:

File-id: 2. File name: NetworkProfile.txt

Description: Collected at each simulation time interval, represents all current network connectivity states at time t.
Storage interval: One file for each simulation run.

Data format: One record per time interval and node-pair type.
Column data at time t: simulation time(ms), Information transfer, k1Count, k2Count, pk1, pk2, p(k1k2),file-id descriptors as defined for file-id number 1 above.

File-id: 3. File name: ActiveNodes.txt
Description: Active nodes at time t as defined in Chapter III (status = 0).
Storage interval: One file for each time interval in each simulation run.
Data format: One record per node-id.
Column data: file-id descriptors, node-id, status, degree, simulation timestamp(ms), node neighbor list at time t.

File-id: 4. File name: ActiveNodesWithNeighbors.txt
Description: Active nodes with neighbors at time t as defined in Chapter III (status = 0).
Storage interval: One file for each time interval in each simulation run.
Data format: One record per node-id and neighbor node-pair.
Column data: file-id descriptors, node-id, neighbor node-id, node degree, simulation timestamp(ms).

File-id: 5. File name: OrphanNodeProfile.txt
Description: Orphan nodes at time t as defined in Chapter III (status = 0).
Storage interval: One file for each time interval in each simulation run.
Data format: One record per node-id.
Column data: file-id descriptors, orphan node-id, status, pre-attack degree, oTime (time the node became orphaned).

File-id: 6. File name: OrphanNodeNeighProfile.txt
Description: Active nodes with neighbors at time t as defined in Chapter III (status = 0). Storage interval: One file for each time interval in each simulation run.
Data format: One record per node-id and neighbor node-pair.
Column data: file-id descriptors, orphan node-id, pre-attack orphan neighbor node-id, pre-attack node degree, oTime (time the node became orphaned).

Appendix D

CPN Simulation Sub-Page Design Details

CPN Sub-page Details

Each I/O port on the CPN main page discussed in the previous section communicates with a CPN sub-page. These sub-pages execute the core functionality of the simulation. This section presents the details of each sub-page.

CPN Sub-Pages - IntializeNodePairs, IntializeNodeNeighList

As shown in Figure U.1 (IntializeNodePairs), place 1 triggers transition A to fire, transition A then reads a text file containing the pre-attack node pairs in CPN token format. Transition A creates a one CPN list (denoted as […]) of node pair tuples ($[< n_i, n_j >]$) representing the pre-attack router adjacencies; where n_i is one node of the node pair and n_j represents the other. Each neighbor node list in Figure D.2 (IntializeNodeNeighList) corresponds to one node's adjacencies. Place 1 triggers transition A to retrieve from a text file a set of lists (each list is denoted as […]) stored in CPN token format. Each list contains a network node with all other elements in the list representing its neighbor nodes as follows: $[< n_i, nn_{k(i)} >]$. Where node $i = \{0,1,2, \ldots., N\}$; N is the total number of pre-attack nodes, n_i is the node key for that list and $nn_{k(i)}$ represents a neighbor node of i. Each node $k(i)$ is the k^{th} neighbor node of node i with degree $k = \{1,2,3, \ldots. K\}$ and where K is the degree of a

specific node. Transition A creates a CPN list of these node-neighbor lists as formatted as follows:

$$[\ [< n_i, nn_{k(i)} >, < n_i, nn_{k+1(i)} >, \ldots., < n_i, nn_{k+K(i)} >\ ],$$

$$[< n_{i+1}, nn_{k(i+1)} >, < n_{i+1}, nn_{k+1(i+1)} >, \ldots., < n_{i+1}, nn_{K(i+1)} >\ ], \ldots..,$$

$$[< n_i, nn_{k(i)} >, < n_i, nn_{k+1(i)} >, \ldots., < n_N + N, nn_{k+K(N)} >\ ]\ ].$$

An illustrative example of the list of neighbor lists on place 2 is as follows:

If place 1 contained 2 node lists:

Node 1 with a degree of 3 and neighbor nodes, 6, 8, 77 and

Node 4 with a degree of 2 and neighbor nodes 55, 43

Then

CPN list found on place 2 created by transition A would be represented as:

[[1,6,8,77], [2,55,l43].].

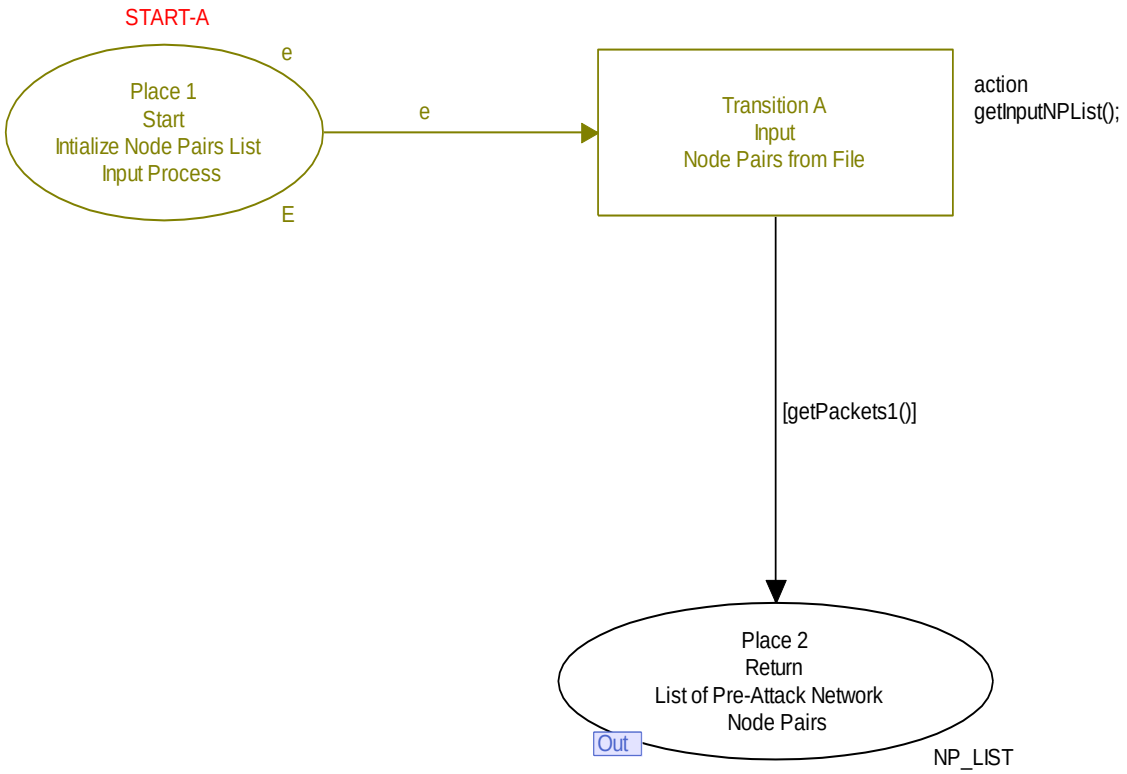


Figure D.1 CPN IntializeNodePairs page

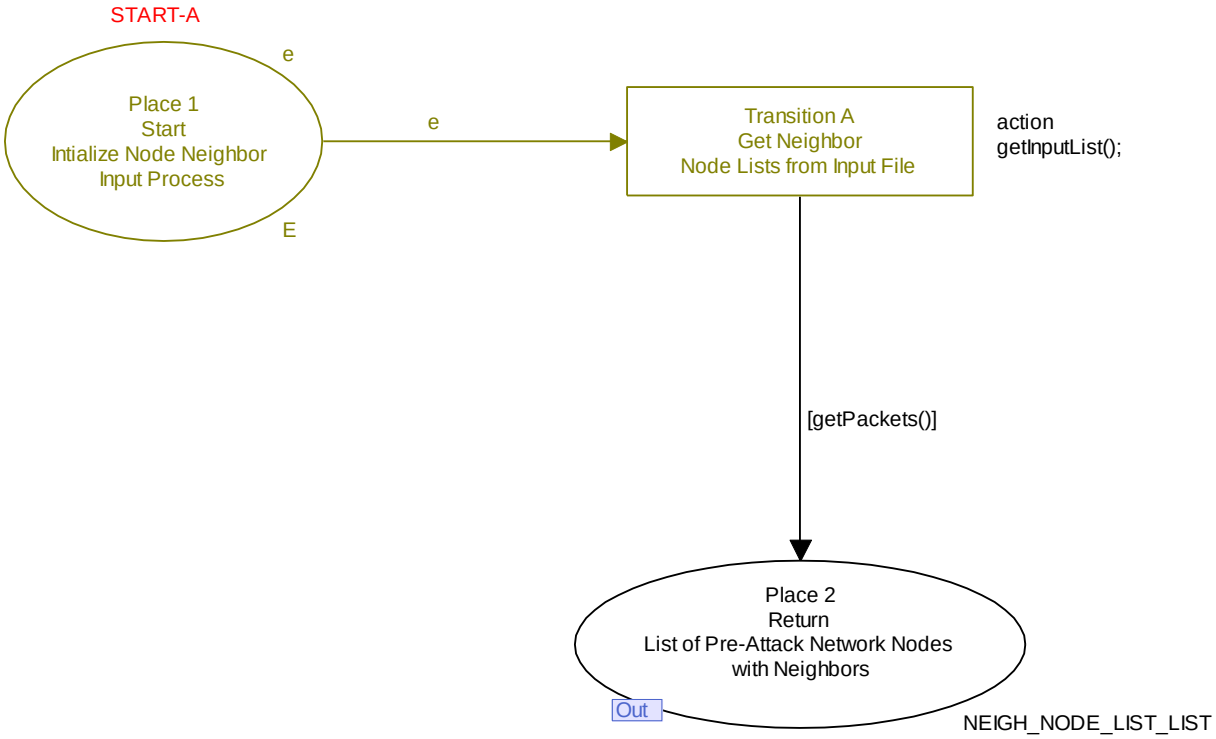


Figure D.2. CPN IntializeNodeNeighList page

CPN Sub-Pages - GetCriticalNodeList, IntializeNetworkDB

Transition A shown in Figure D.3 (GetCriticalNodeList) is triggered by place 1 and it will input critical node tokens from a text file stored in CPN token format and create a list of critical nodes on place 2 in the following format: [$n_{critical\ (i)}$, $n_{critical\ (i+1)}$, …..,$n_{critical\ (N(critical\ ))}$]; where $N(critical)$ is the number of critical nodes as previously defined for each run class. In Figure D.4 (IntializeNetworkDB), when there are tokens available on place 1A and 1B, transition A will fire and it creates the simulation output file(NetworkDB_NodeStatus). This audit trail text file will be updated continuously by the simulation output function (to be defined later in this chapter) by adding an incremental tuple to the file whenever there is a change to a node's status, degree or neighbor nodes (ie. it records all changes to the networkDB node records).

After the file has been initially created by transition A, it passes the token with the list of all pre-attack nodes and the associated neighbors to place 2. Place 2 serves as temporary storage location, and eventually the tokens are passed to transition B. Transition B will write the initial pre-attack node data (status = 0, timestamp = 0ms) to the output file. Transition B will also create the initial pre-attack CPN record structure for each node and store it in the network DB and return it to the main page via place 3. The network DB data structure is a list of cpn records with one record for each node. The sum of all the networkDB records represents the current global connectivity state of the network during the attack. The networkDB node records will be used to implement node state changes and it is continually updated during the simulation.

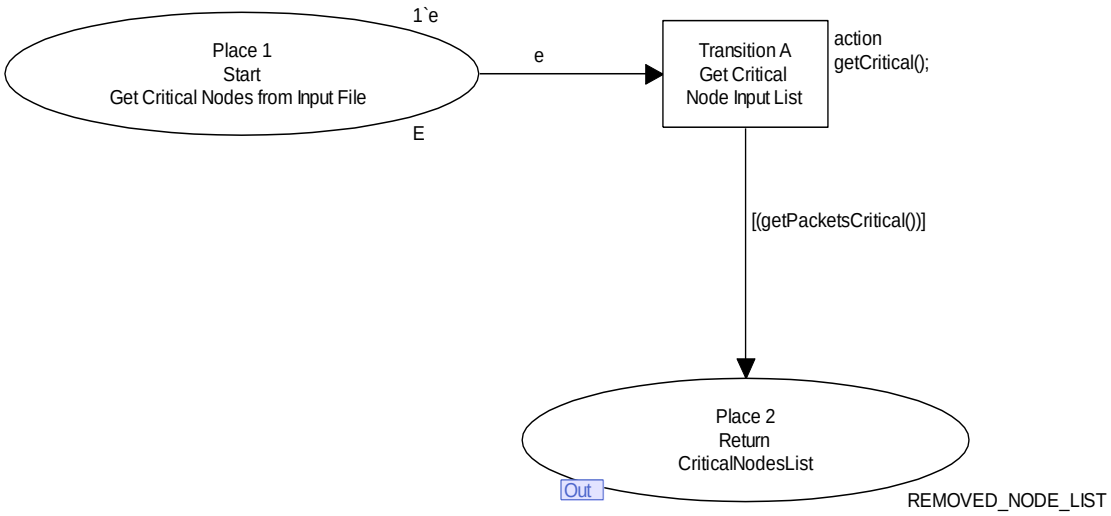


Figure D.3. CPN GetCriticalNodeList page

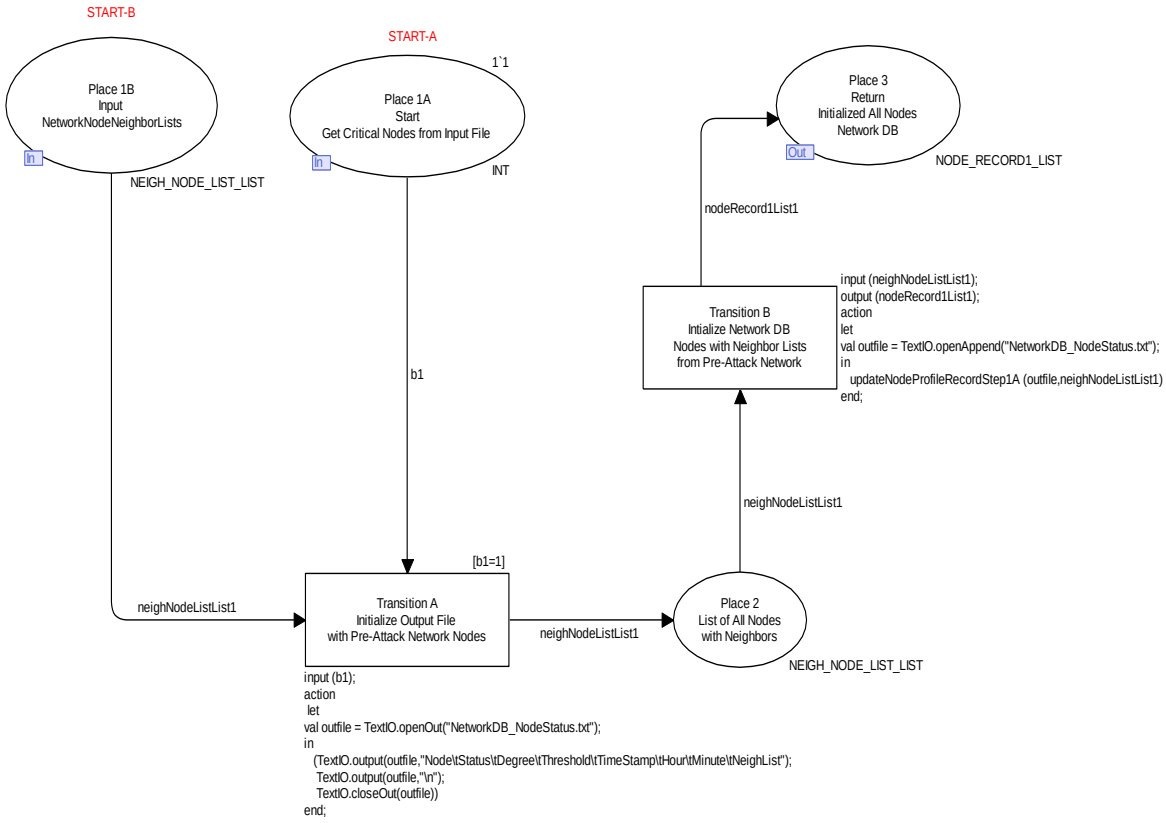


Figure D.4. CPN IntializeNetworkDB page

CPN Sub-Pages - IntializeCriticalNodes, ReleaseNP

The simulated attack is shown in Figure D.5 (IntializeCriticalNodes). Place 1A (critical Nodes List) and place 1B (current list of network DB connectivity records) triggers transition A to fire creating an attacked node, randomly selected from the list of critical nodes. Transition A also updates the network DB connectivity record for the newly designated attacked node. As previously defined, an attacked node is removed from the simulation and all communications with that node are halted. Transition B then adds the chosen attacked node to the removed nodes list (place 1C). The updated removed nodes list and network DB records are returned to the main page and are globally available to the simulation. The removed nodes list in place 1C will be evaluated against the node-pair released in Figure D.6 (ReleaseNP) by the EvaluateNP page depicted in Figure D.6. Place 1 shown in Figure D.6 represents the current list of all node-pairs available to the simulation. Randomly and continuously as determined by the CPN simulation engine, place 1 will trigger transition A to fire and randomly select one node pair from the node pair list and return it to the main page via place 3 for further processing.

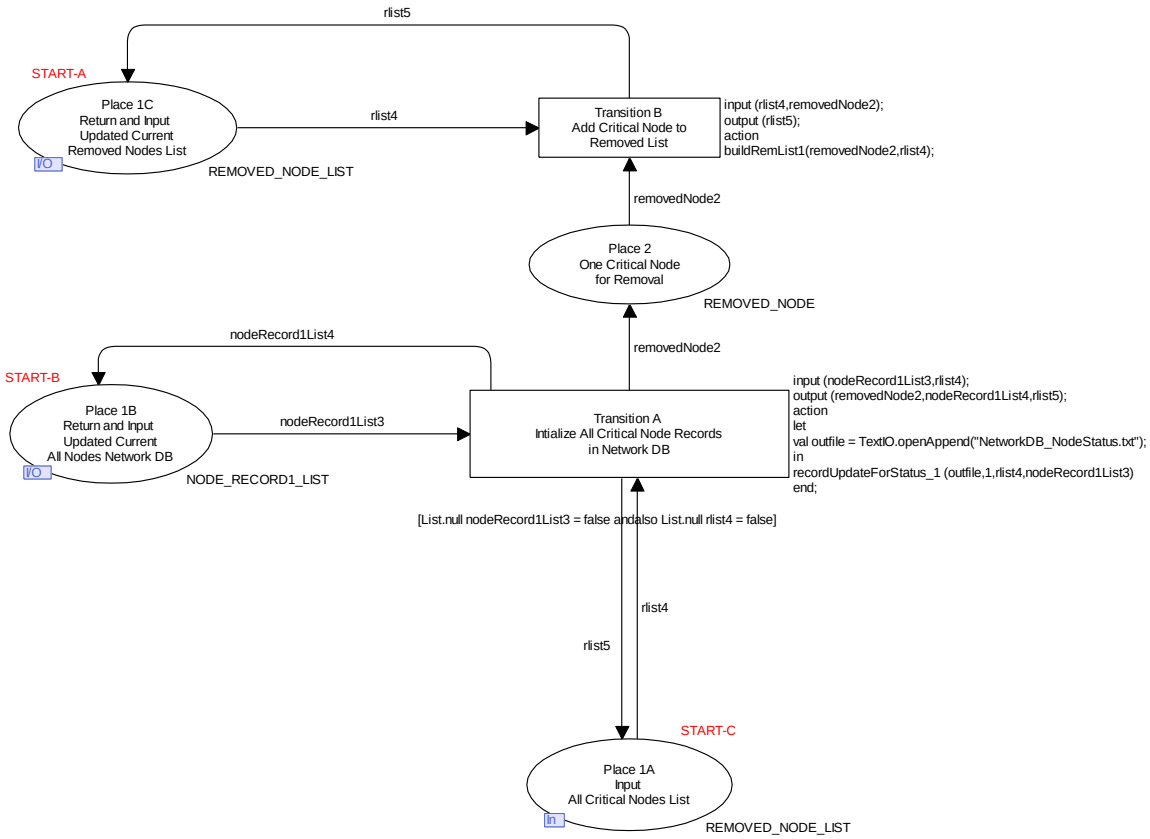


Figure D.5. CPN IntializeCriticalNodes page

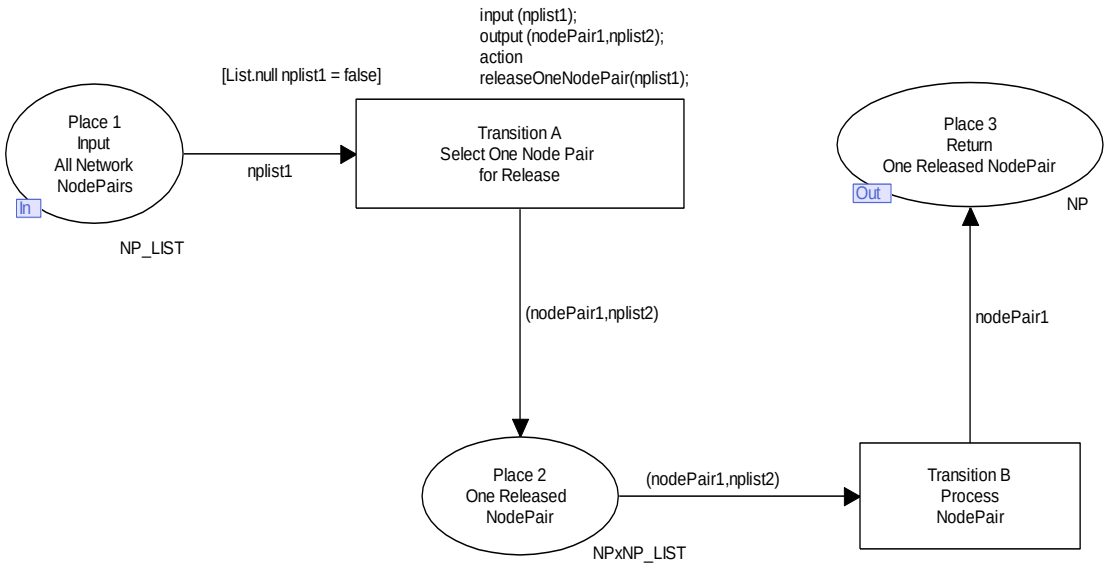


Figure D.6. CPN ReleaseNP page.

CPN Sub-Page - EvaluateNP

The CPN page for the EvaluateNP sub page is depicted in Figure D.7. This sub-page executes the core attack simulation processes. Due to the complexity of this page Figures D.8 and D.9 are provided for further clarification. As mentioned earlier, the release of node pairs represents an attempt by one router to communicate with one of its neighbors. The "EvaluateNP" sub page continuously evaluates the node pairs released (communication attempts) and determines whether the communication is successful. As shown in Figure D.7, one released node pair inputted is passed through transition F1 for evaluation. This node pair is added to place 12. Place 12 and the current set of removed nodes (place 9) triggers transition A. Transition A divides the node pair into 2 nodes, one node is added to place 1A and the other node is added to place 1B. Each node is subsequently evaluated as depicted in Figure D.8.

Annotation "A" in Figure D.8 projects that each node is evaluated separately using the Boolean functions, attackedNodePair1 and attackedNodePair2. The attackedNodePair1 function returns true if node1 of the node pair is a member of the removed nodes list (place 9) and attackedNodePair2 returns true if node 2 is a member of the removed nodes list. After transition C has passed the results of each Boolean expression, the conditional statement on the arc to place 3 is evaluated.

Reading from left to right, with T being true and F being false, Figure D.8 (annotation "A") indicates that if attackedNodePair1 and attackedNodePair2 for both nodes is true then the token associated with that node pair is assigned a nodeStatus1 value of 3 indicating that both nodes have been removed from the simulation and the node pair should be discarded (to trash bin, place 4). If either of the nodes is evaluated to be true, then the other node is

designated a temporary orphan and passed to place 6 for further processing. This leads to the node pair being discarded and implementation of the temporary orphan recovery process as introduced in the formal definitions chapter of this dissertation. And finally, if both nodes are not members of the current set of removed nodes list (place 9) then the node pair is added back to the list of active node pairs (place 5) and no further processing of that node pair occurs until it is re-released at random and evaluated again against the dynamic list of removed nodes. Transition E2 of Figure D.8 is the CPN implementation of the temporary orphan recovery process mechanism and it is the focus of Figure D.9.

Once the node has been determined to be a temporary orphan and is placed on place 6 as previously shown in Figure D.7, the temporary orphan recovery process must ascertain the disposition of this node. At this juncture, temporary orphans will not be processed further if they are members of the PROTECTED_LIST. This is a list of nodes that have been selected to be protected against the attack. Therefore protected nodes are not candidates for removal and transition E1 will fire for all nodes on the protected list. If the node is not on the PROTECTED_LIST then it is passed to transition E2 for further evaluation and potential removal from the simulated network connectivity. As the implementation of transition E2 (Figure D.9) continues, the CPN functions will evaluate the temporary orphans neighbor nodes, and attempt to establish a new link through preferential attachment. After processing the new link and any subsequent overloaded or null-link orphans, the CPN functions will update the networkDB and return the updated networkDB and removed nodes list to the main page.

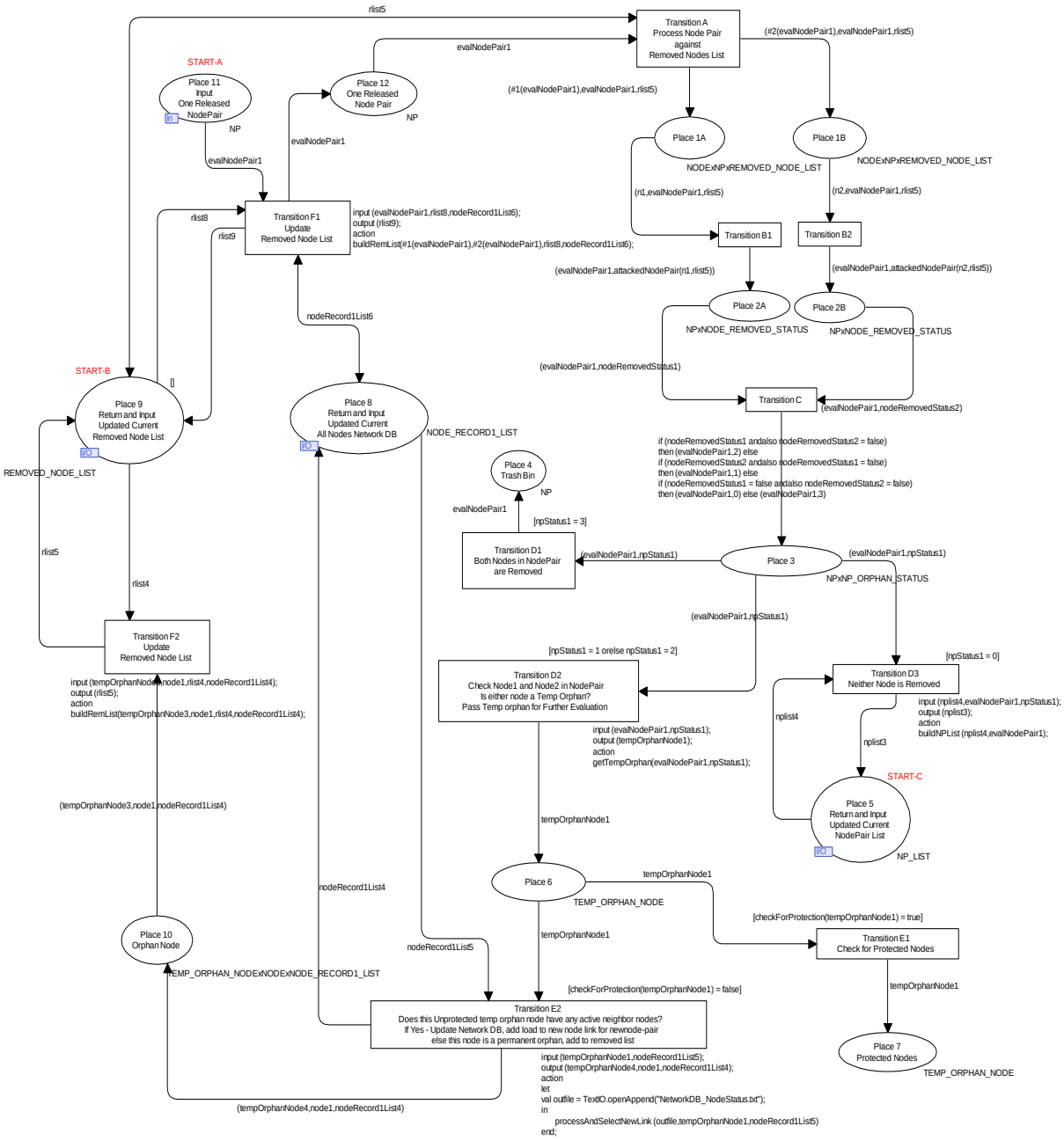


Figure D.7. CPN EvaluateNP page

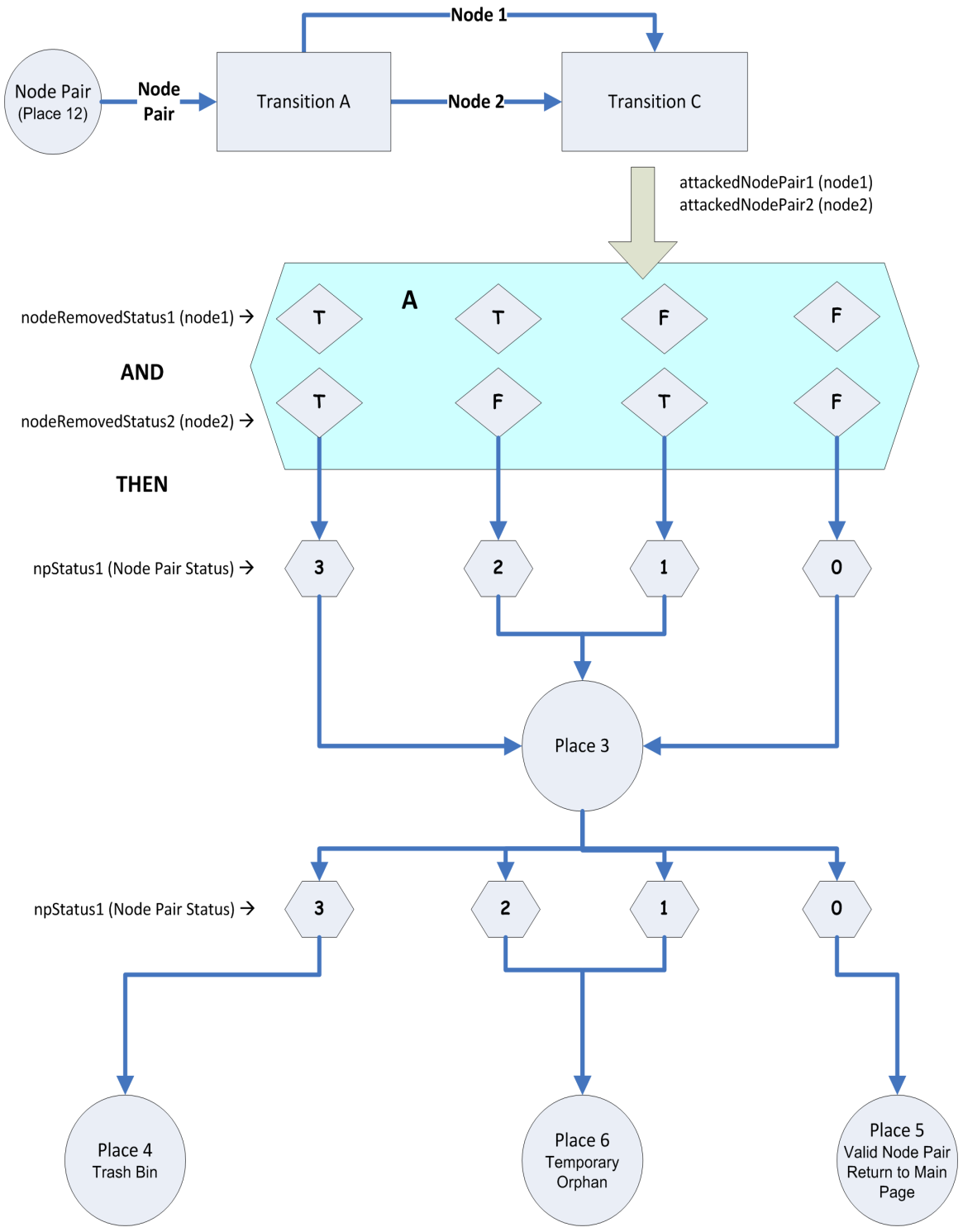


Figure D.8. Evaluate NP processing summary

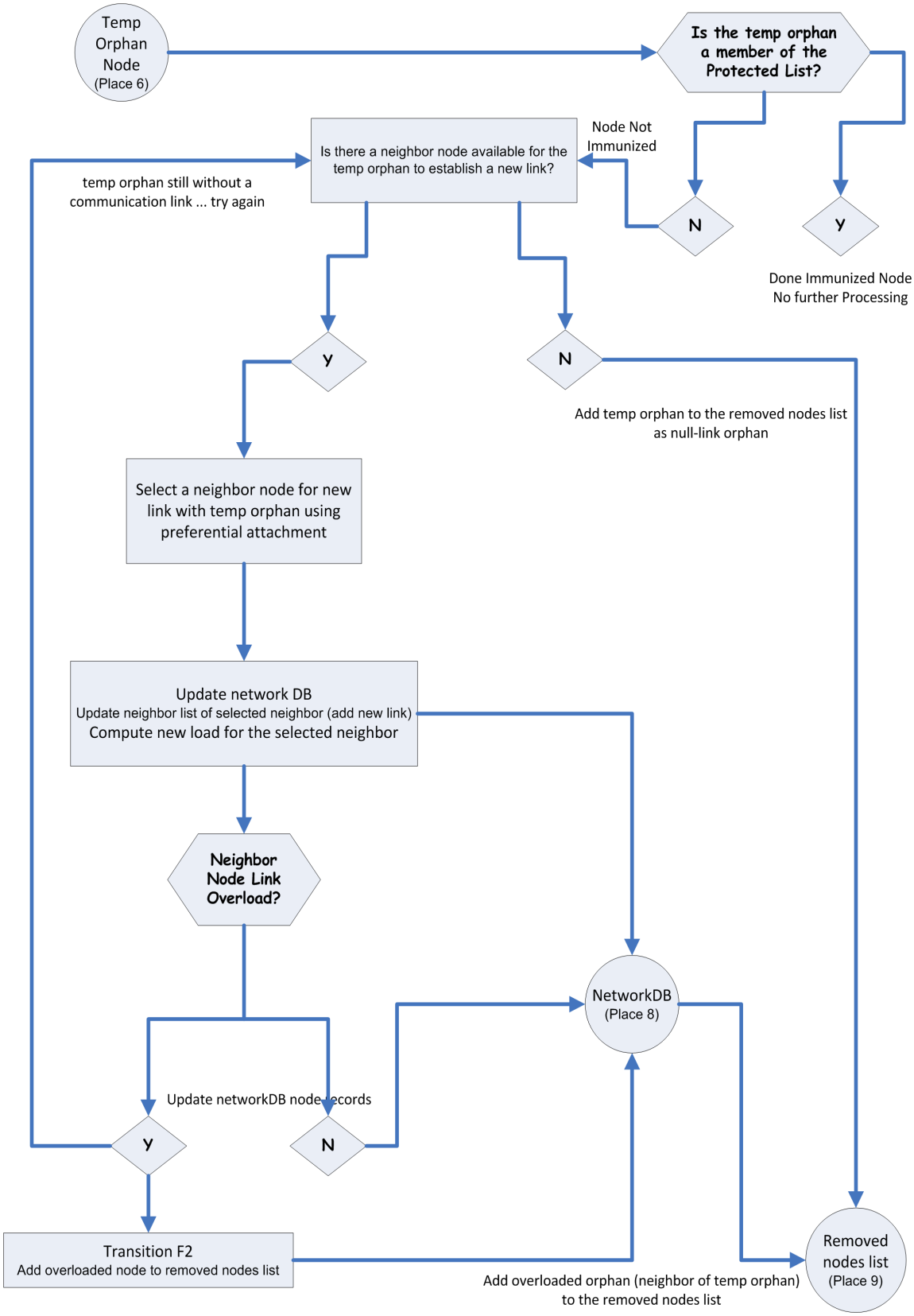
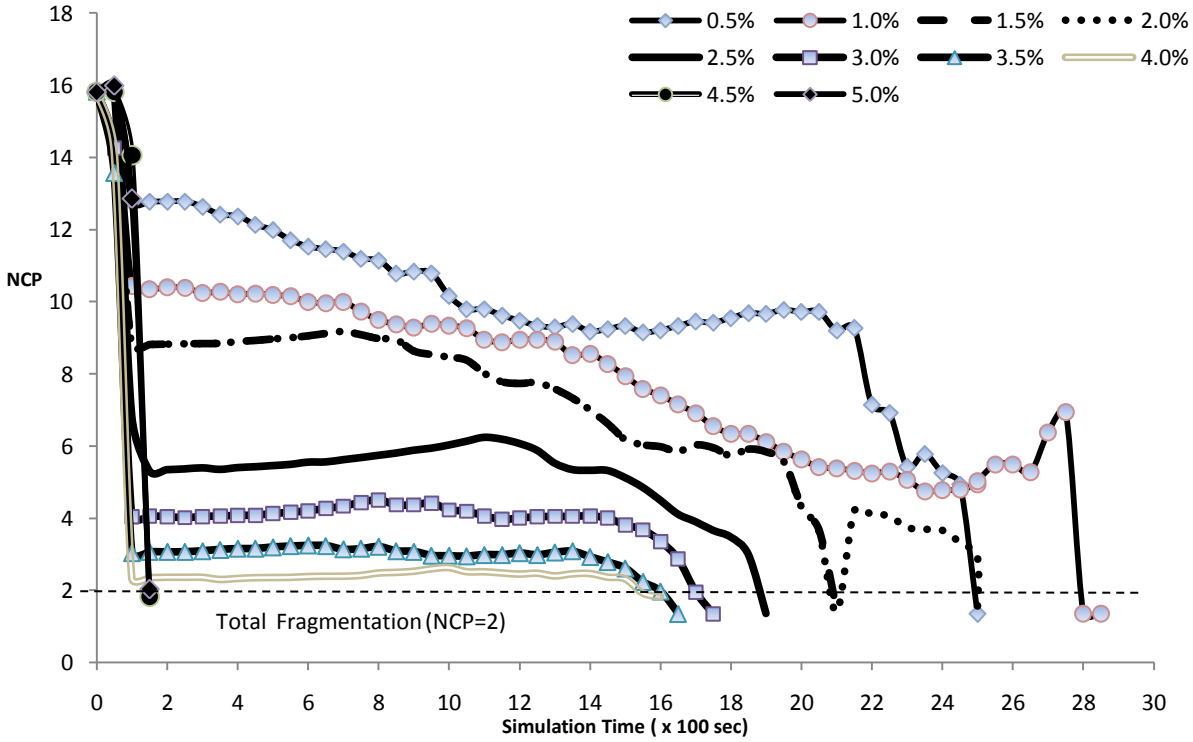


Figure D.9.CPN Transition E2 Processing Logic

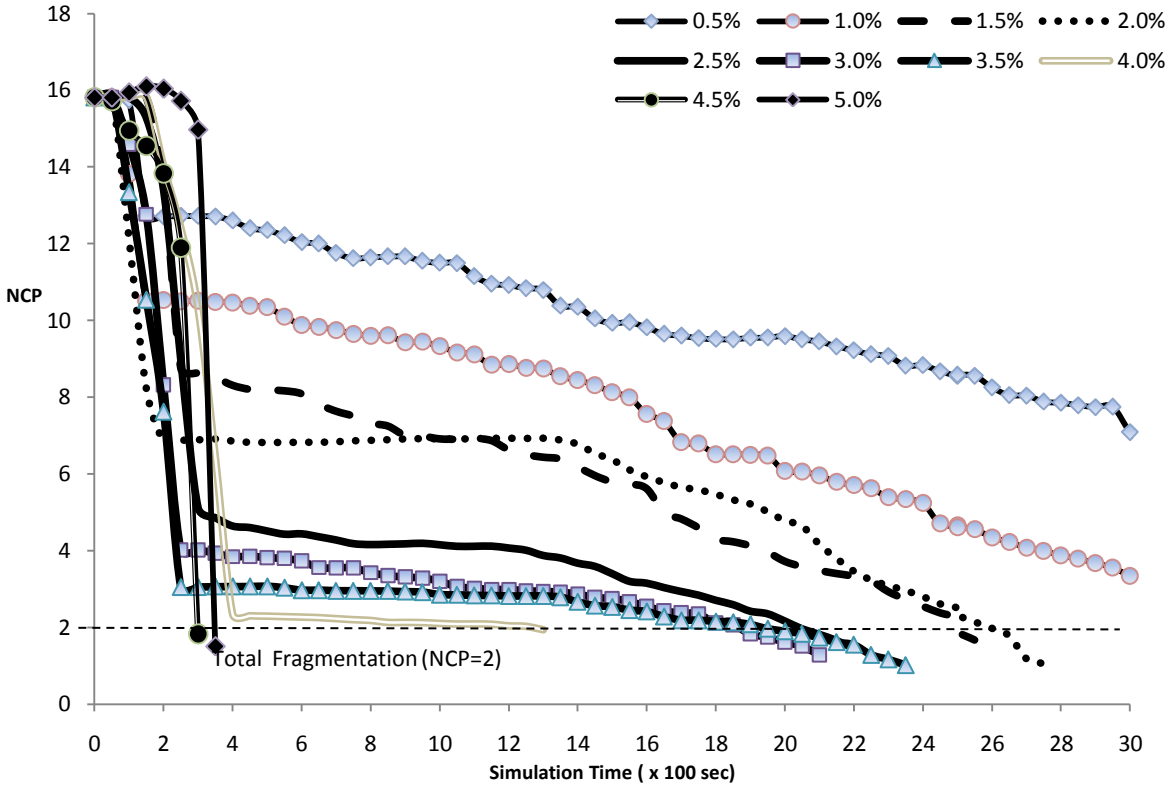
Appendix E

Network Connectivity Parameter Results by Run Type

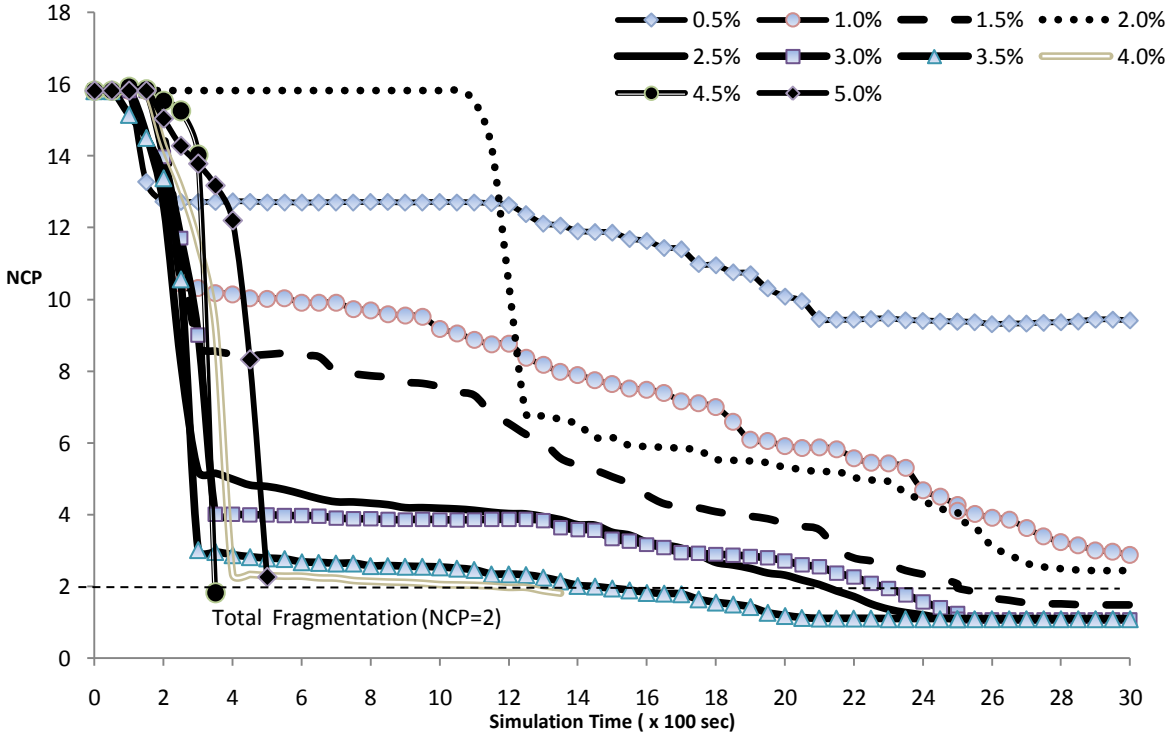
Run Type 1



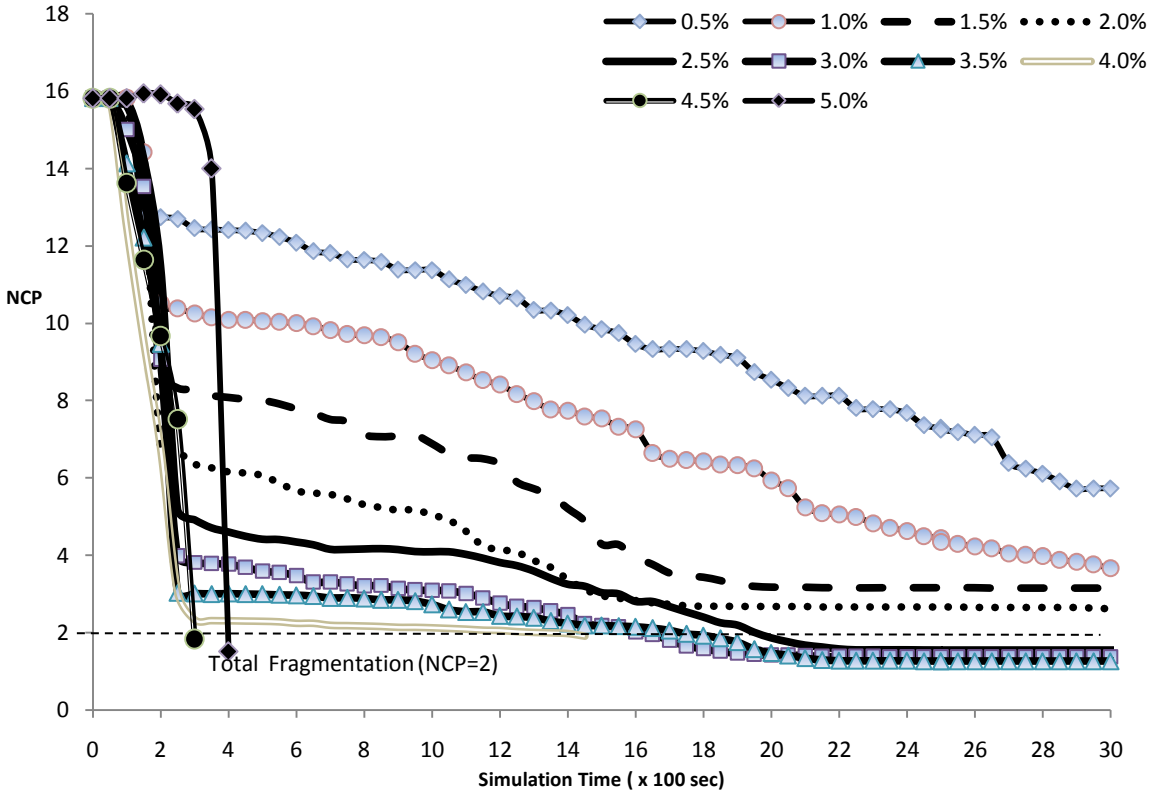
Run Type 2



Run Type 3



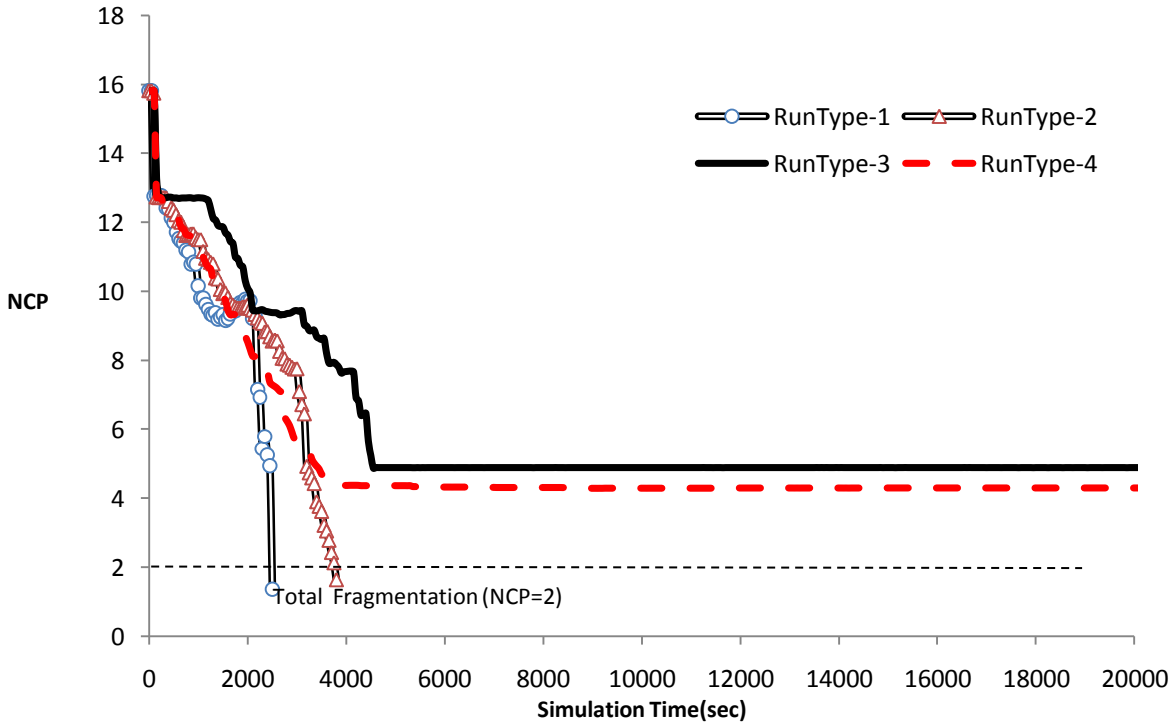
Run Type 4


0.5%
1.0%
1.5%
2.0%
2.5%
3.0%
3.5%
4.0%
4.5%
5.0%
NCP
Total Fragmentation (NCP=2)
Simulation Time (x 100 sec)
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30

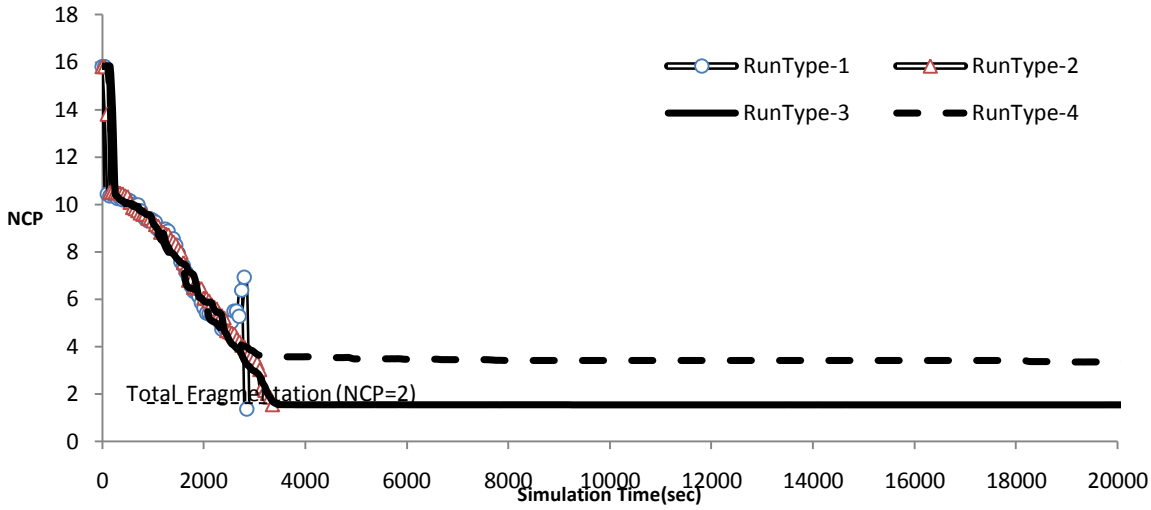
Appendix F

Network Connectivity Parameter Results by Attack Class

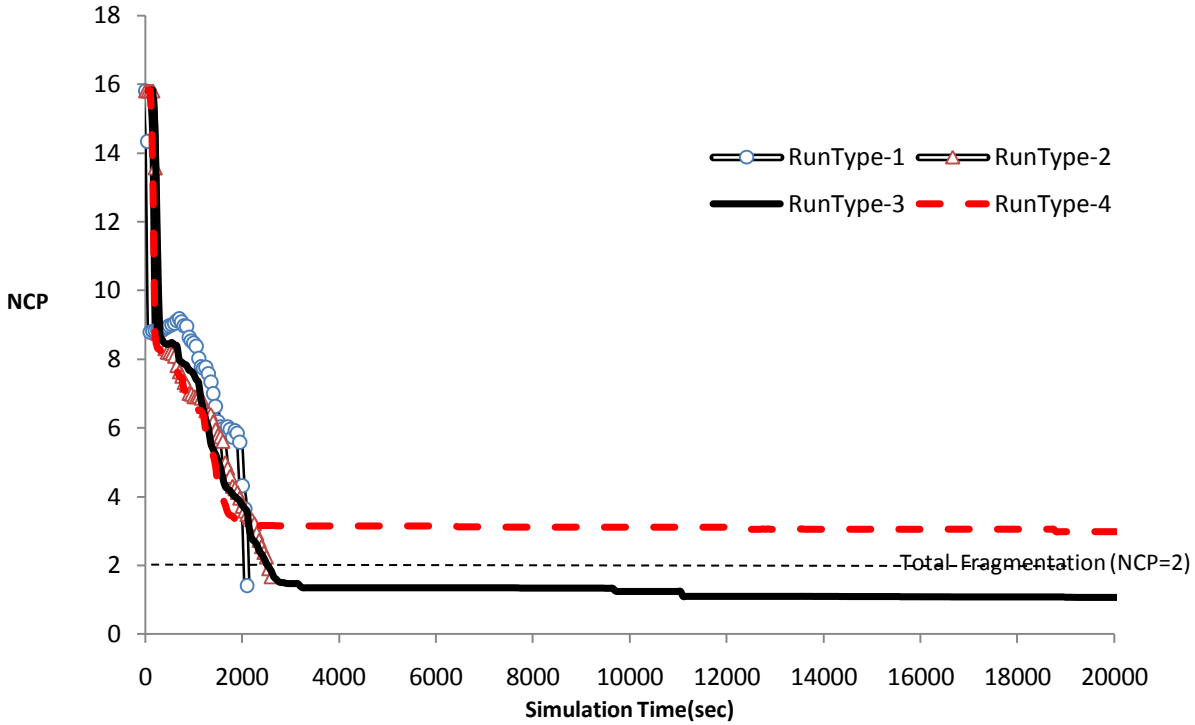
Attack Class 0.5%



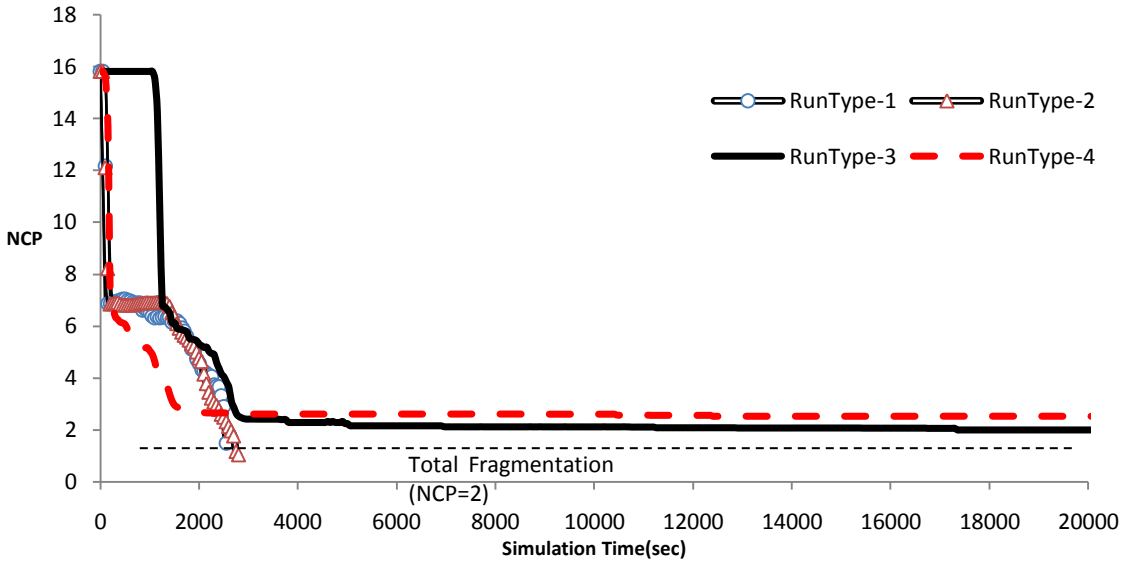
Attack Class 1.0%



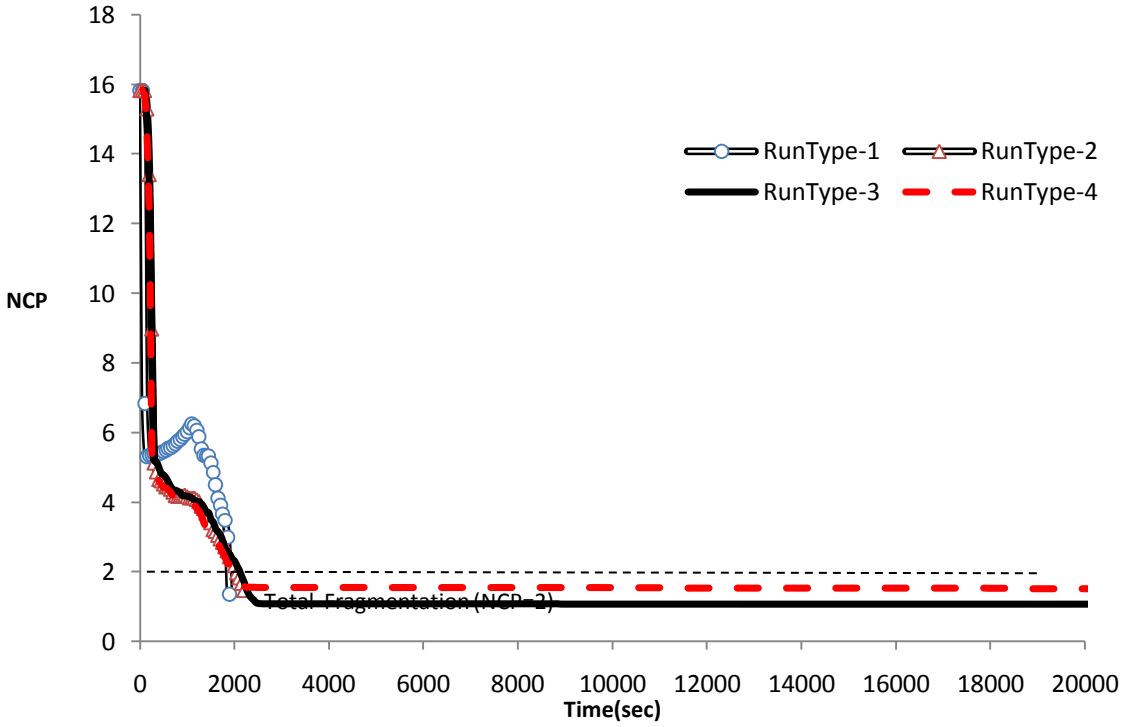
Attack Class 1.5%



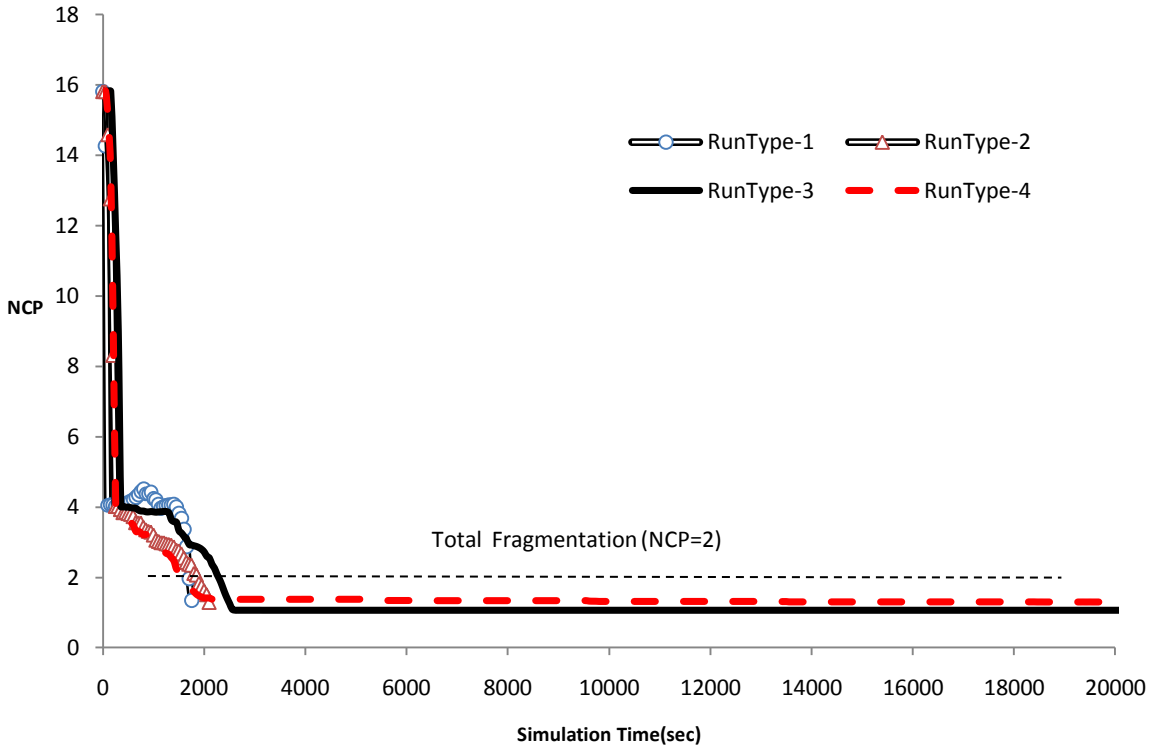
Attack Class 2.0%



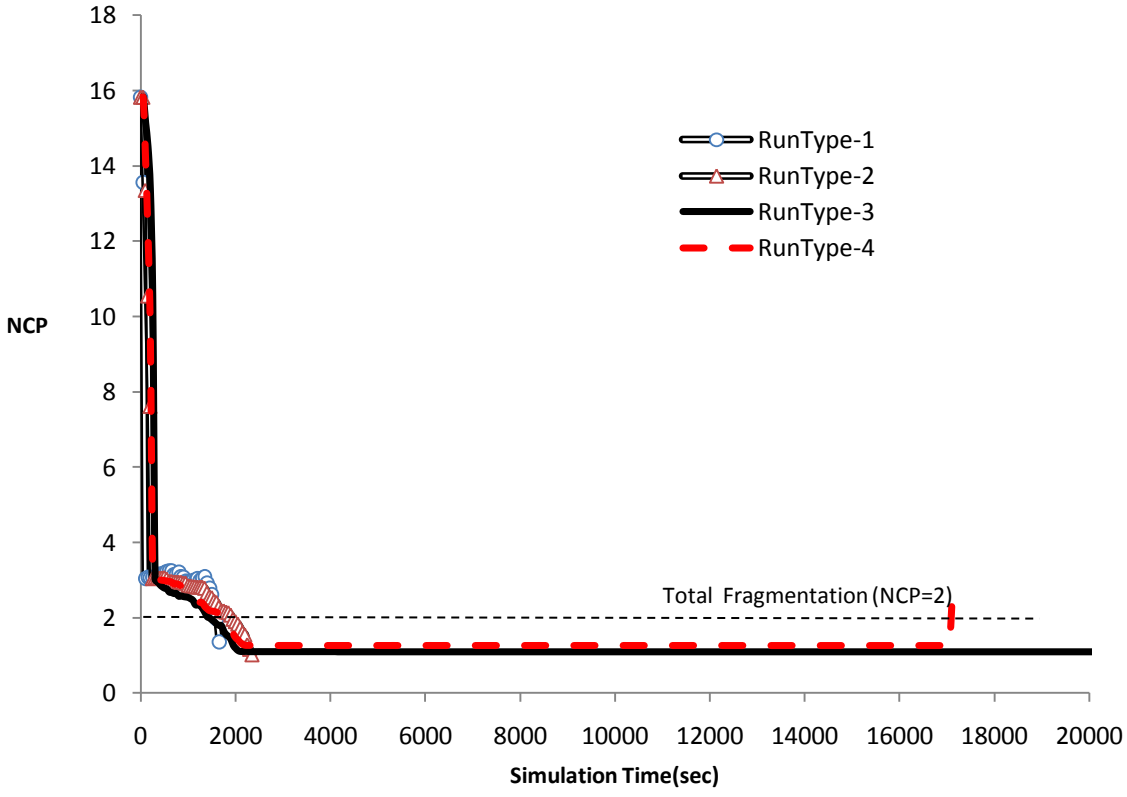
Attack Class 2.5%



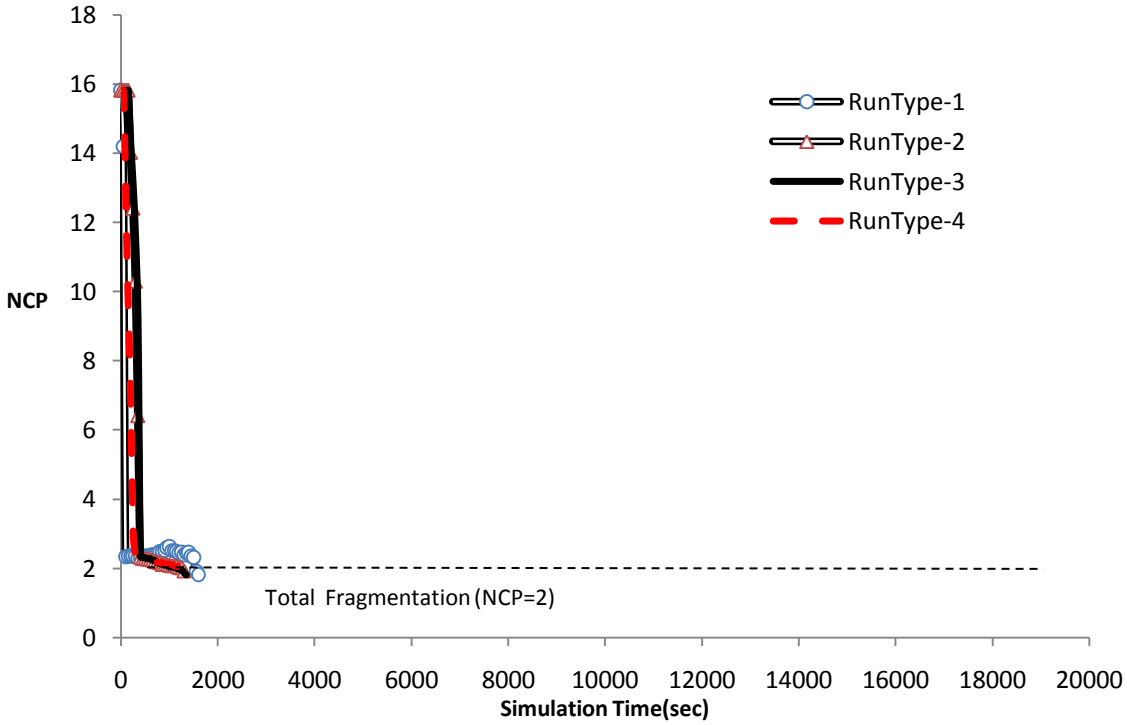
Attack Class 3.0%



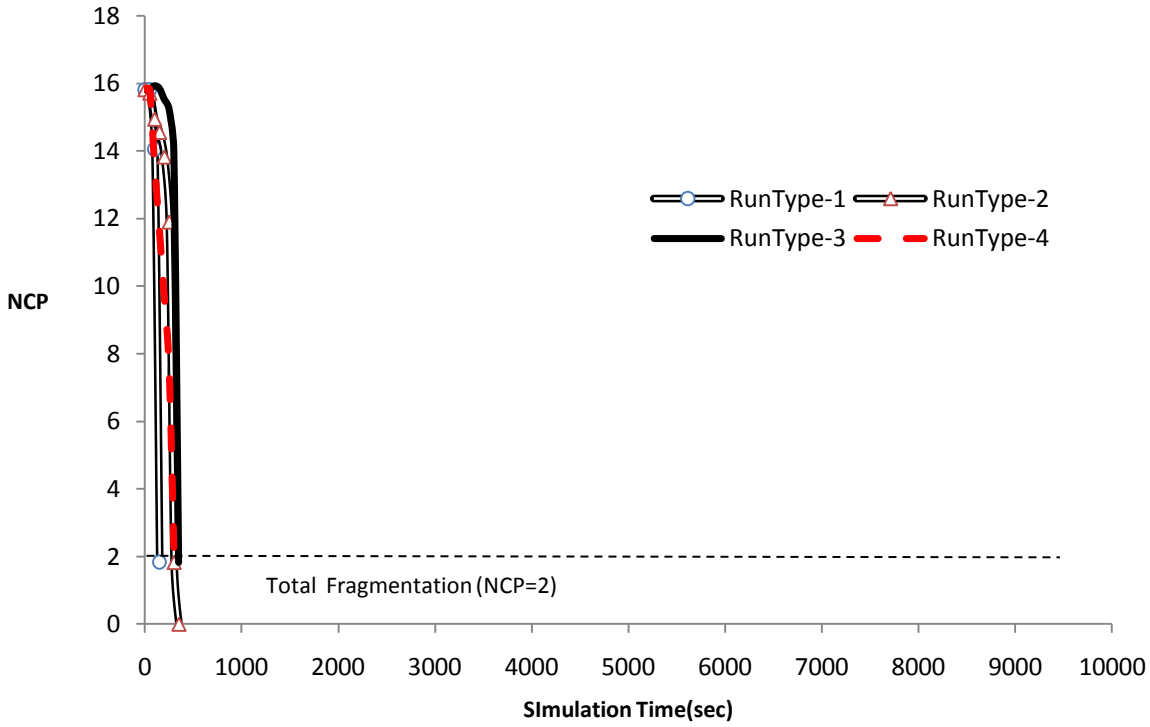
Attack Class 3.5%



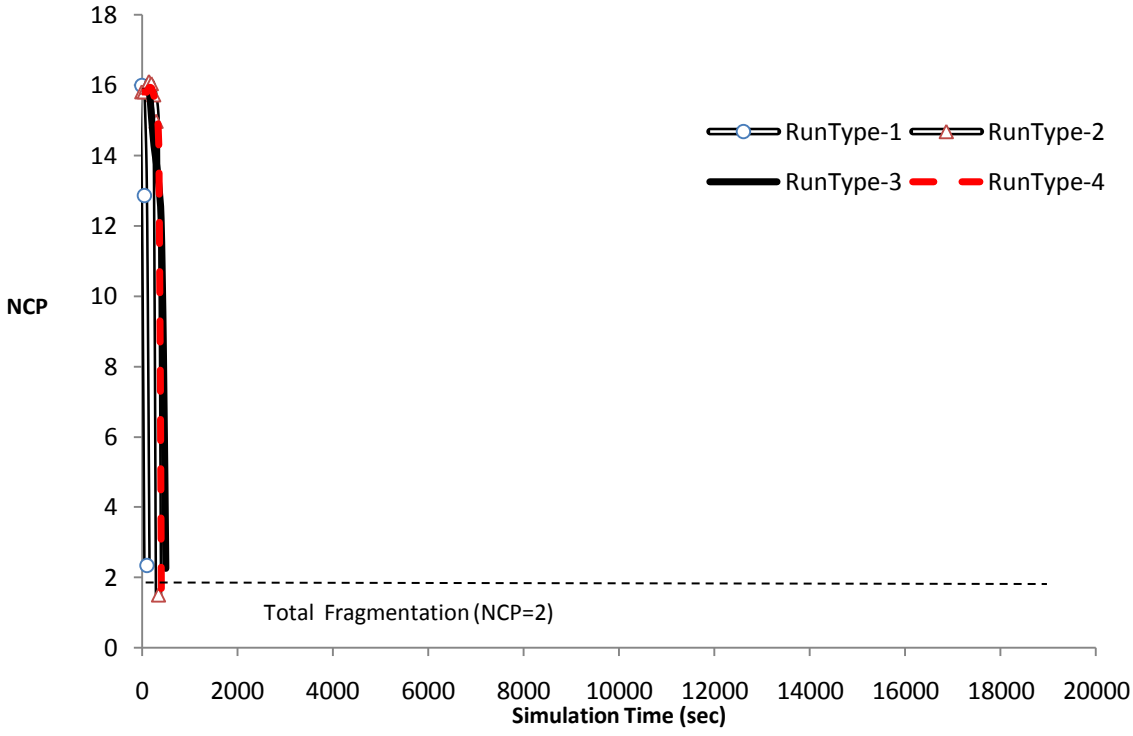
Attack Class 4.0%



Attack Class 4.5%



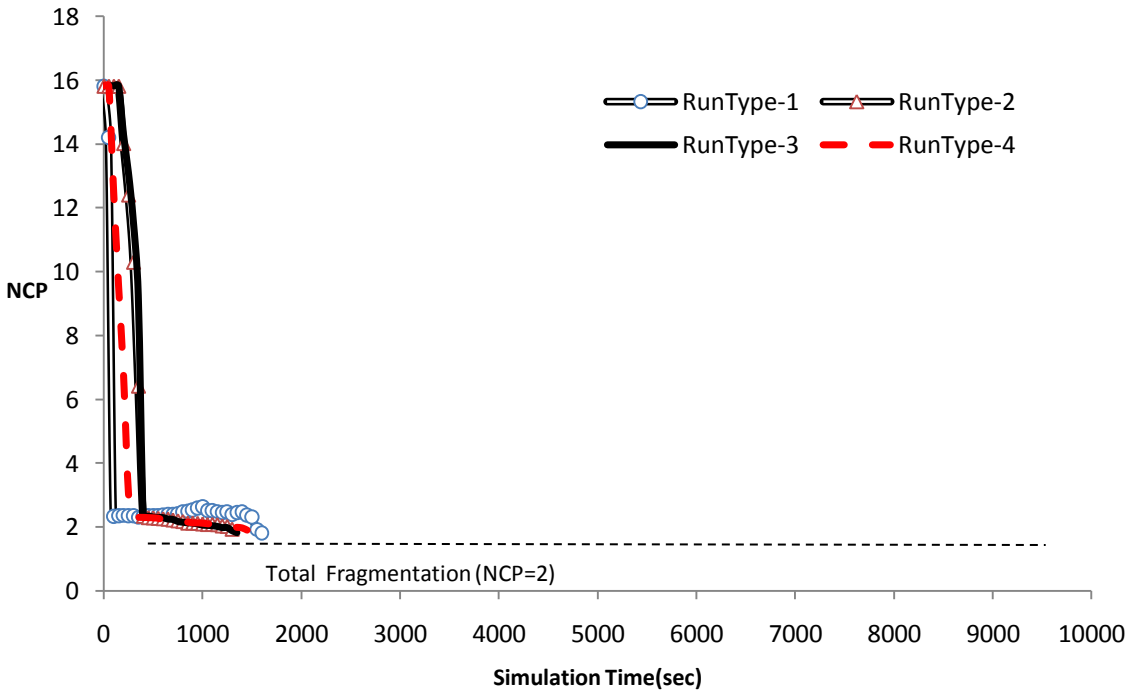
Attack Class 5.0%



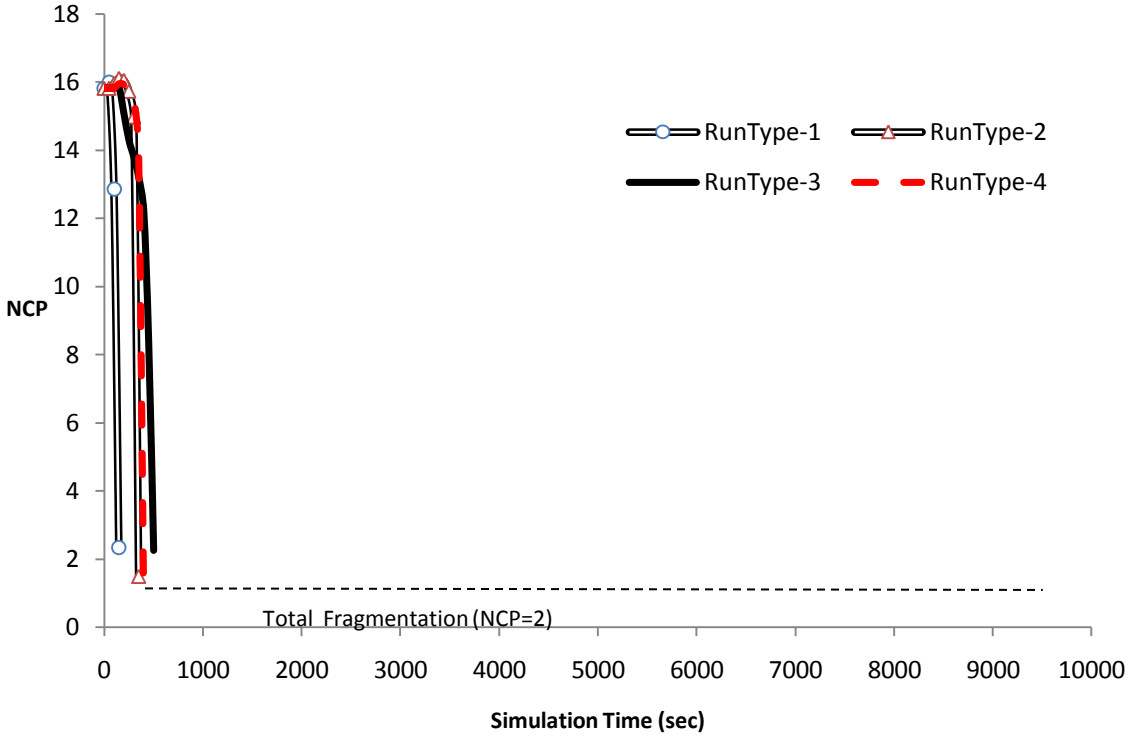
Appendix G

Network Connectivity Parameter Results by Attack Effect

Attack Effect 1 - Attack Classes - 0.5% through 3.5%



Attack Effect 2 - Attack Classes - 4.0% through 5.0%



Appendix H

Run Type 2 Results for Network Stability and Node-Pair Type Counts

Ten run type 2 simulations are presented in this section. These simulations represented the affects of the denial-of-service attacks against the pre-attack network. These simulations represented protection strategy 1.

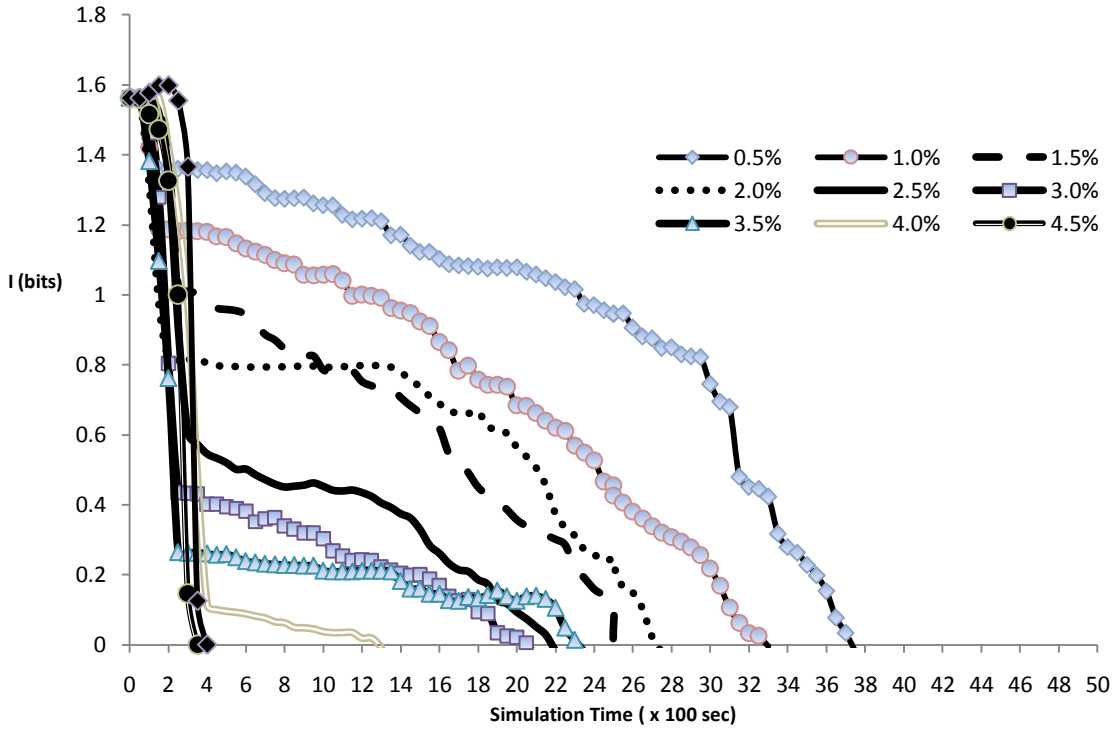


Figure H.1 Network Stability – Run type 2, information transfer

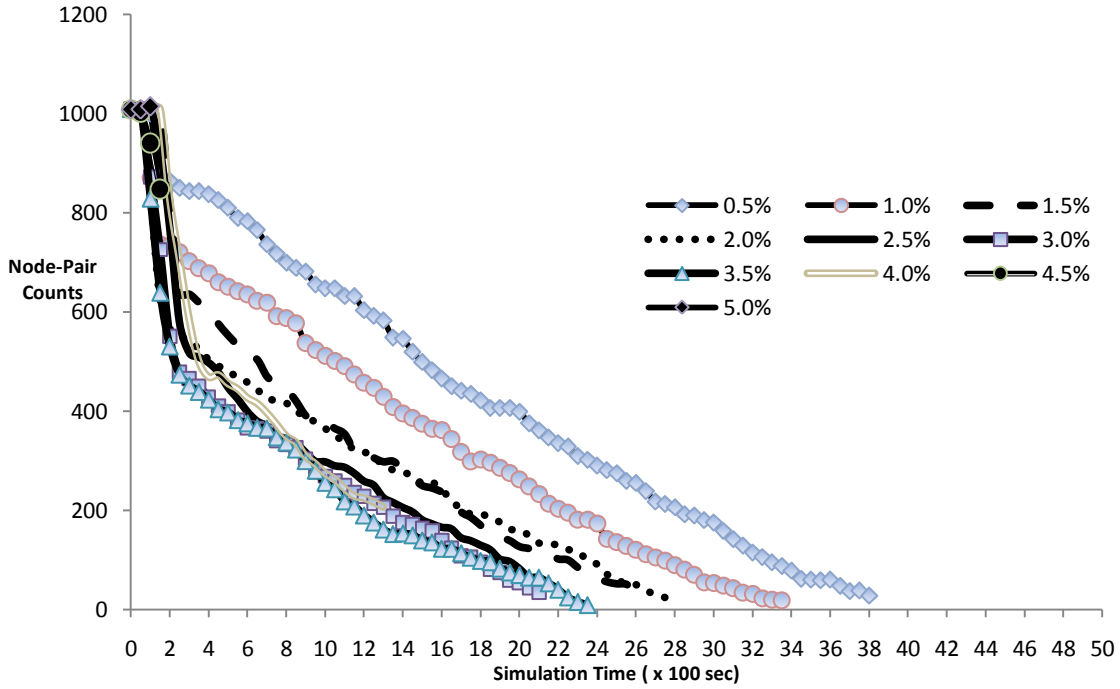


Figure H.2. Run type 2, Node-pair Type 1-2 Counts, By Attack Class, first 5000 seconds

Behaviors in the first 400 seconds

As shown in Figure H.1, all run type 2 attack classes experienced a sudden decline in information transfer in the first 300 to 400 seconds. The information transfer for attack classes 4.5% and 5.0 % rapidly declined and met their terminal conditions in the first 400 seconds. The information transfer decrease in the first 400 seconds experienced by all attack classes was influenced by the relative attack severity. Over the first 400 seconds, the rate and magnitude of the information transfer loss increased with an increase in attack severity. Over the first 400 seconds, information transfer decreased for all attack classes by 3.3% to 25.0% from the simulation's pre-attack conditions. The information transfer rate of decline during this period ranged from approximately 0.06 to 0.47 bits/sec .

Along with the information transfer decreases shown in Figure H.1 there was a corresponding decrease in the number of node-pairs of type 1-2. As shown in Figure H.2, there was also sudden decline in the number of node-pairs of type 1-2 in the first 300 to 400

seconds. Over the first 300 to 400 seconds, the number of node-pairs of type 1-2 decreased for all attack classes by 4.9% to 25.0% from the simulation's pre-attack conditions. The node-pair count rate of decline during this period ranged from approximately 50 to 201 node-pairs/sec.

Behaviors after the first 400 seconds

As shown in Figure H.1, attack classes 0.5% through 4.0% established a local minimum value at approximately 400 seconds. This local minimum level decreased with an increase in attack severity. After the local minimum value was established, the information transfer decreased for each attack class at a slower rate ranging from 0.01 to 0.03 bits/sec. This slow rate of decrease continued until each attack class encountered its terminal conditions.

During this period of slower information transfer decline shown in Figure H.1, Figure H.2 depicts the corresponding decrease in the number of node-pairs of type 1-2. As shown in Figure H.2, after the first 400 seconds, attack classes 0.5% through 4.0% declined over time at a significantly lower rate ranging from 24 to 33 node-pairs/sec. This slower rate of node-pair loss continued for each attack class until the terminal conditions for information transfer were met.

Critical Threshold and Terminal Condition Behaviors

Figures H.1 and H.2 did not exhibit a critical threshold. For the information transfer data shown in Figure H.1, the most severe attack classes, 4.5% and 5.0%, met their terminal conditions the earliest. Attack classes 3.0% and 4.0% were the next attack classes to meet their terminal conditions. Attack classes 0.5% and 1.0% achieved the terminal conditions

latest. The terminal conditions for the remainder of the attack classes varied and attack severity did not have a significant influence.

Appendix I

Run Type 3 Results for Network Stability and Node-Pair Type Counts

Ten run type 3 simulations are presented in this section. These simulations represented the affects of the denial-of-service attacks against the pre-attack network. These simulations represented protection strategy 2.

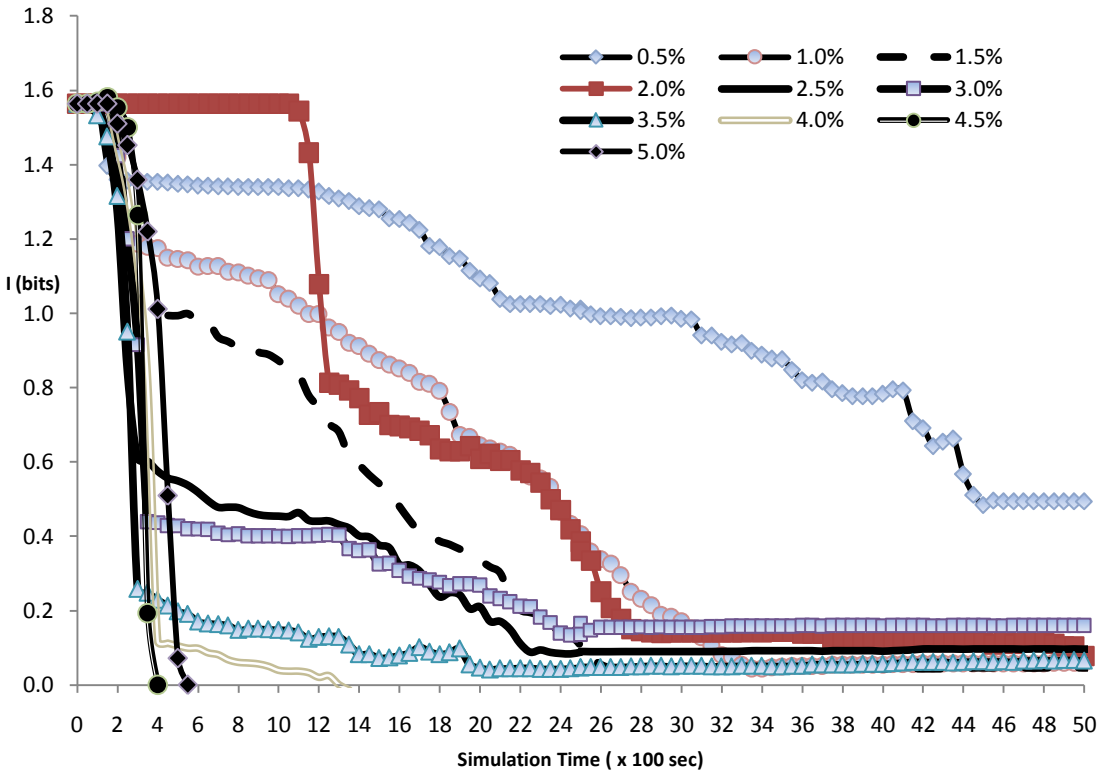


Figure I.1. Network Stability – Run type 3, information transfer

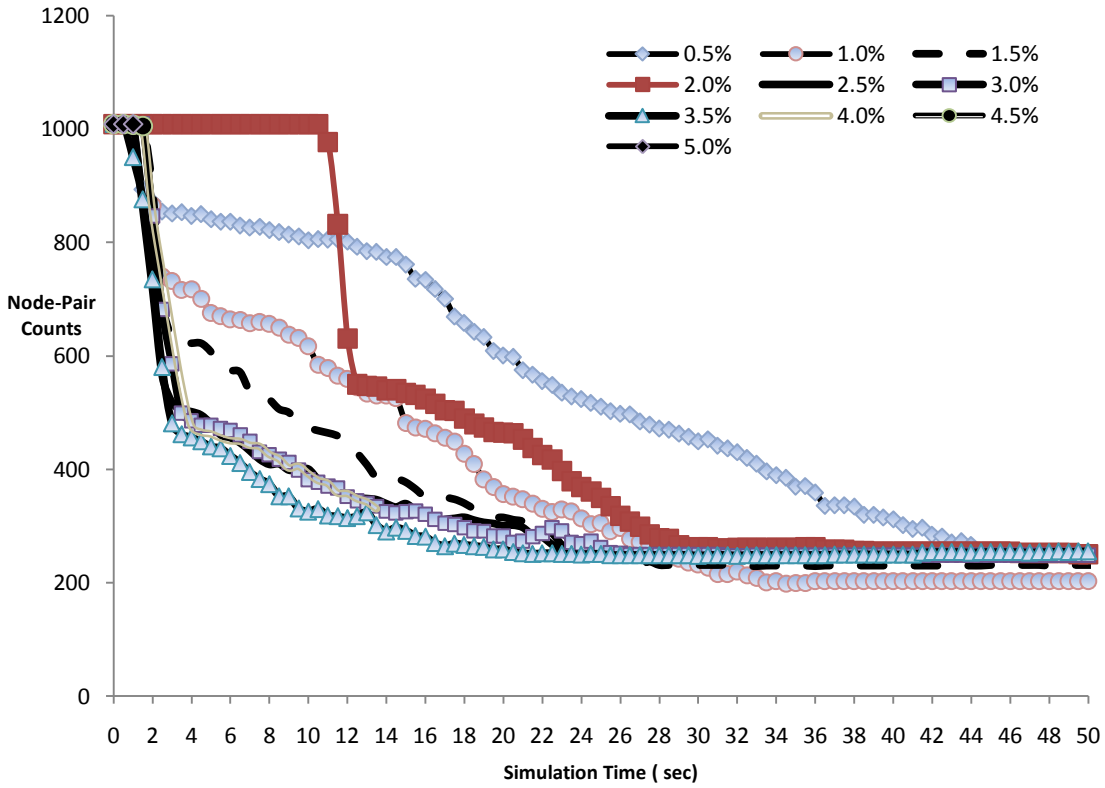


Figure I.2. Run type 3, Node-pair Type 1-2 Counts, By Attack Class, first 5000 seconds

Behaviors in the first 400 seconds

As shown in Figure I.1, all run type 3 attack classes experienced a sudden decline in information transfer in the first 300 to 400 seconds. The information transfer for attack classes 4.5% rapidly declined and met its terminal conditions in the first 400 seconds. It is shown in Figure I.1 that the information transfer loss for attack class 5.0% did not meet its terminal conditions until approximately 600 seconds. In the first 400 seconds, the decrease in information transfer for attack class 5.0% was less pronounced than attack class 4.5%. With the exception of attack class 5.0%, the information transfer decrease in the first 400 seconds experienced by all attack classes was influenced by the relative attack severity. Over the first 400 seconds, the rate and magnitude of the information transfer loss increased with an increase in attack severity. Over the first 400 seconds, information transfer decreased for

all attack classes by 3.4% to 25.1% from the simulation's pre-attack conditions. The information transfer rate of decline during this period ranged from approximately 0.06 to 0.41 bits/sec .

Along with the information transfer decreases shown in Figure I.1 there was a corresponding decrease in the number of node-pairs of type 1-2. As shown in Figure I.2, there was also sudden decline in the number of node-pairs of type 1-2 in the first 400 seconds. Over the first 300 to 400 seconds, the number of node-pairs of type 1-2 decreased for all attack classes by 4.0% to 25.0% from the simulation's pre-attack conditions. The node-pair count rate of decline during this period ranged from approximately 49 to 169 node-pairs/sec. Figure I.2 depicts that the loss of node-pairs by attack class varied. Attack severity did not influence the loss of node-pairs.

Behaviors after the first 400 seconds

As shown in Figure I.1, attack classes 0.5% through 4.0% established a local minimum value at approximately 400 seconds. This local minimum level decreased with an increase in attack severity. After the local minimum value was established, the information transfer decreased for each attack class at a slower rate ranging from 0.01 to 0.04 bits/sec. This slow rate of decrease continued until each attack class encountered its equilibrium point. Attack classes 4.0% through 5.0% did not encounter an equilibrium point.

During this period of slower information transfer decline shown in Figure I.1, Figure I.2 depicts the corresponding decrease in the number of node-pairs of type 1-2. As shown in Figure I.2, after the first 400 seconds, attack classes 0.5% through 4.0% declined over time at

a significantly lower rate ranging from 8 to 19 node-pairs/sec. This slower rate of node-pair loss continued until each of these classes encountered its equilibrium point.

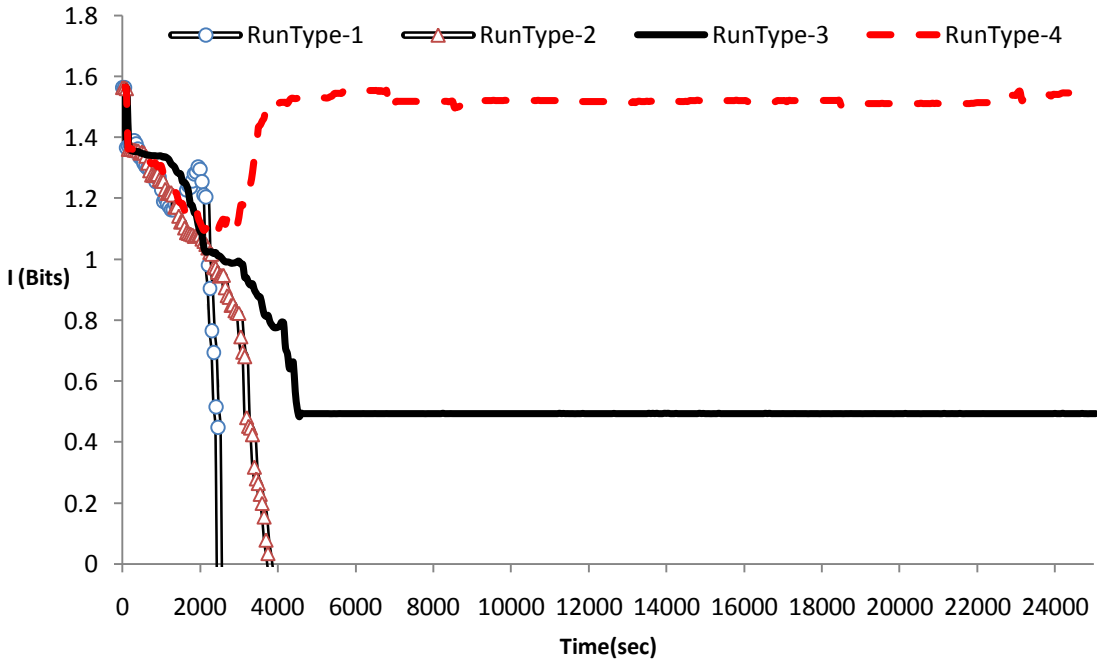
Critical Threshold, Terminal Condition and Equilibrium Behaviors

Figures I.1 and I.2 did not exhibit a critical threshold. As shown in Figure I.1, attack classes 0.5% through 3.5% encountered an equilibrium point after the initial 1,800 seconds. No equilibrium point occurred for attack class 4.0% which met its terminal conditions around 1,300 seconds. Attack classes 4.5% and 5.0% did not encounter an equilibrium point; they met their terminal conditions in the first 400 to 600 seconds. The equilibrium point and level varied by attack class. The equilibrium levels attack classes 0.5% through 3.5% occurred at approximately 200 node-pairs. The minimum node-pair count for attack class 4.0%, 4.5% and 5.0% at their terminal conditions was 330, 482 and 490 node-pairs respectively.

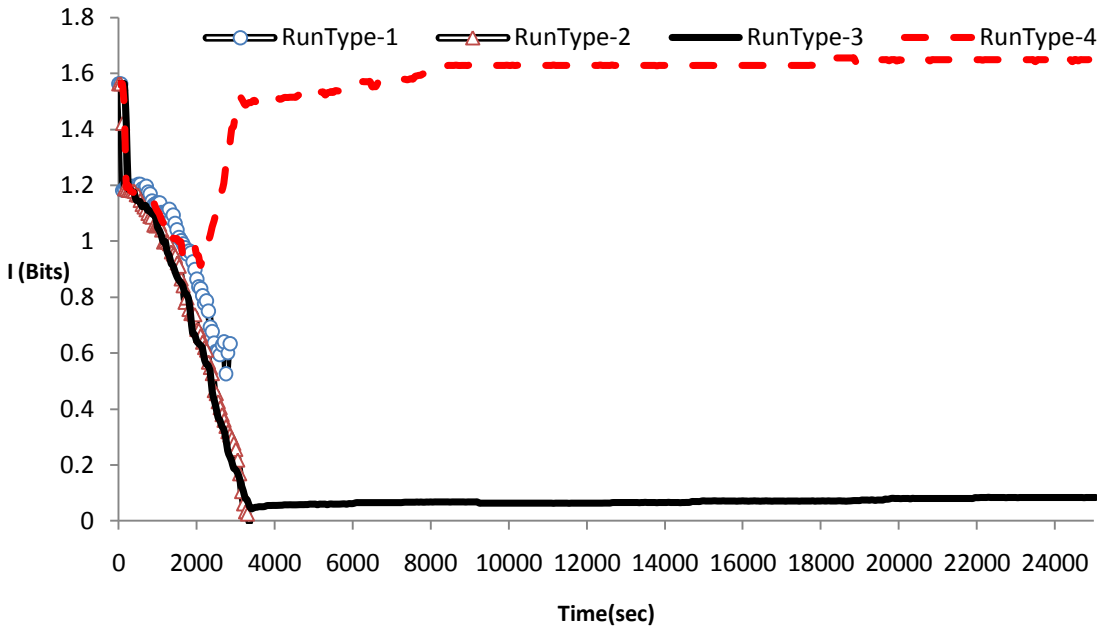
Appendix J

Mutual Information Transfer Results by Attack Class

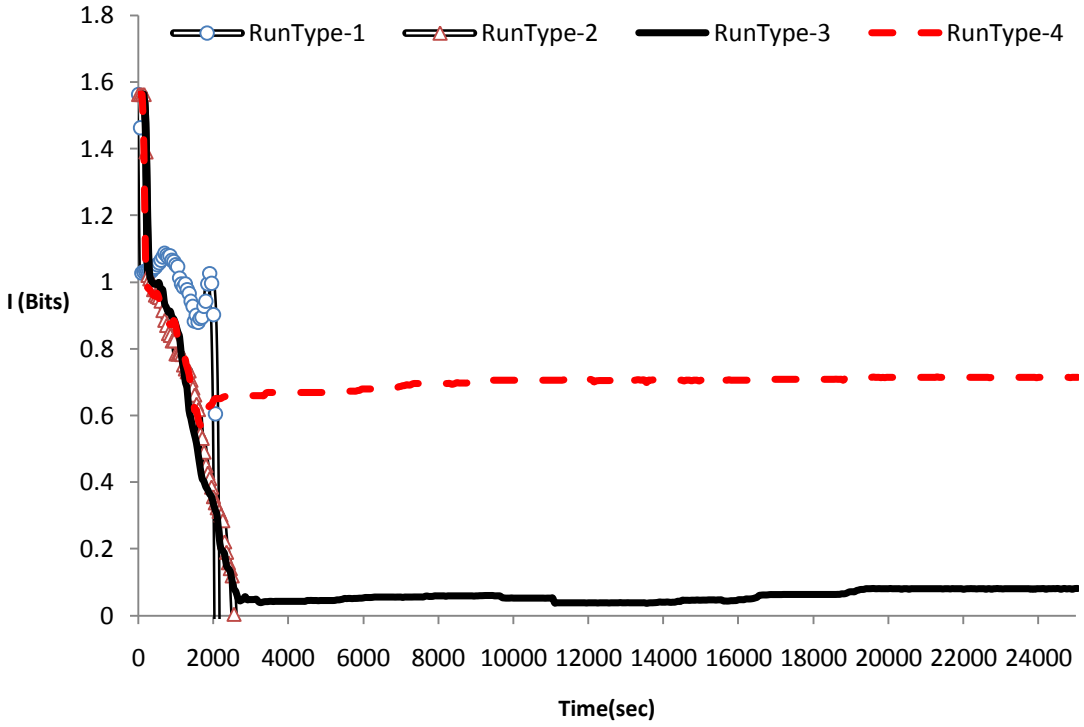
Attack Class 0.5 %



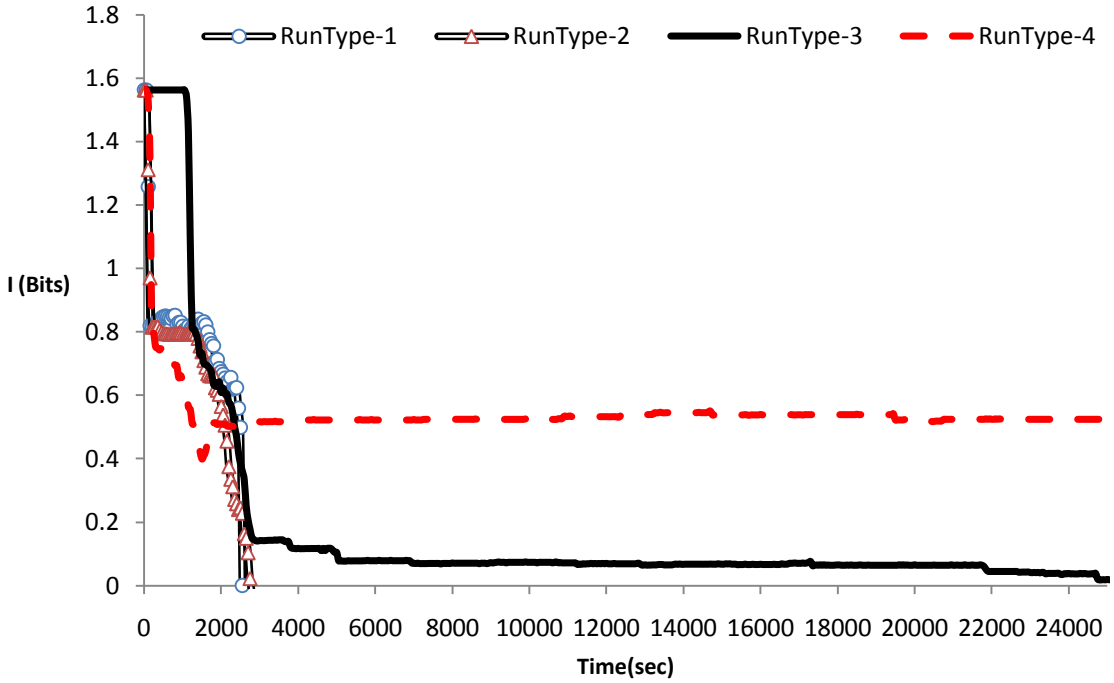
Attack Class 1.0 %



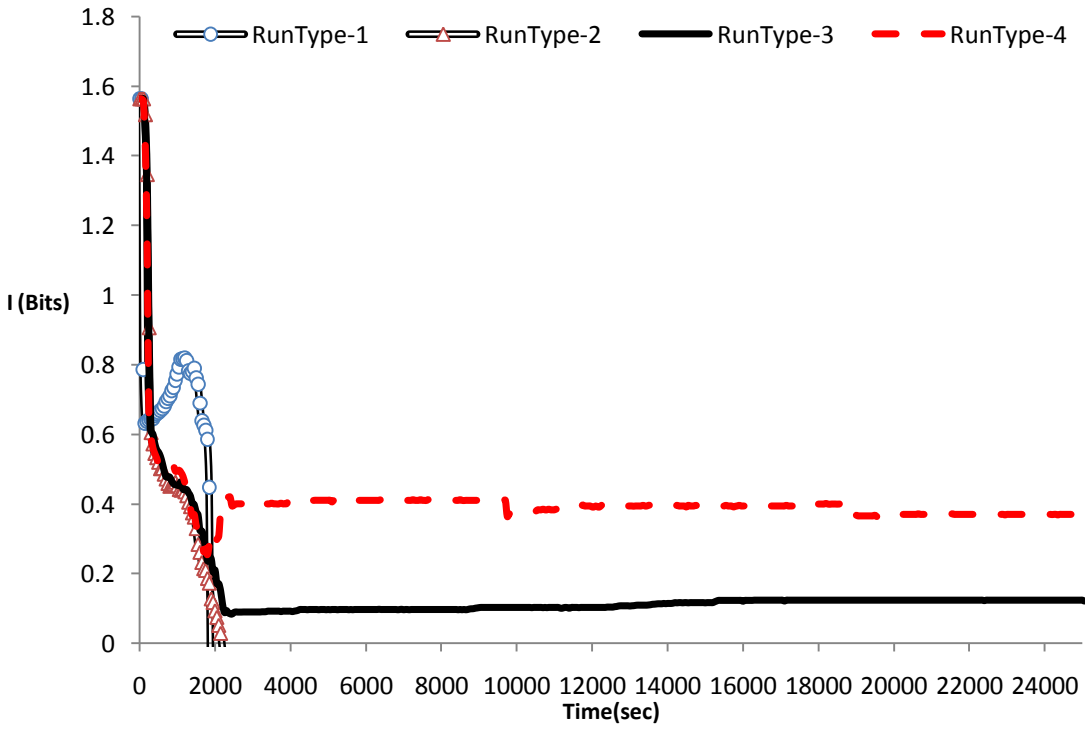
Attack Class 1.5 %



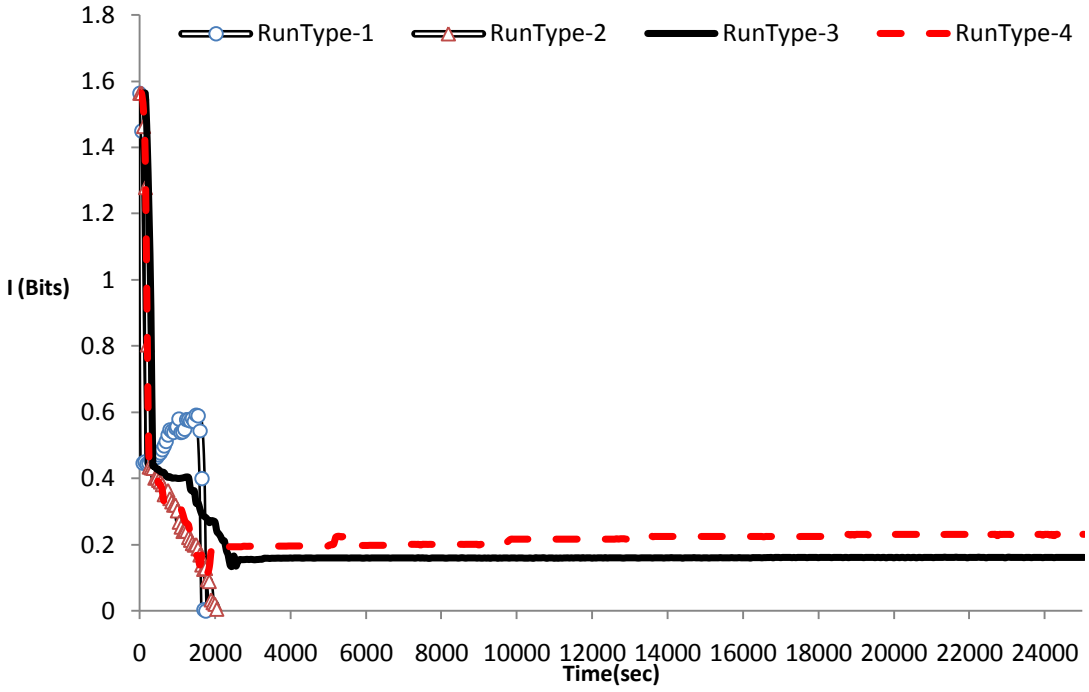
Attack Class 2.0 %



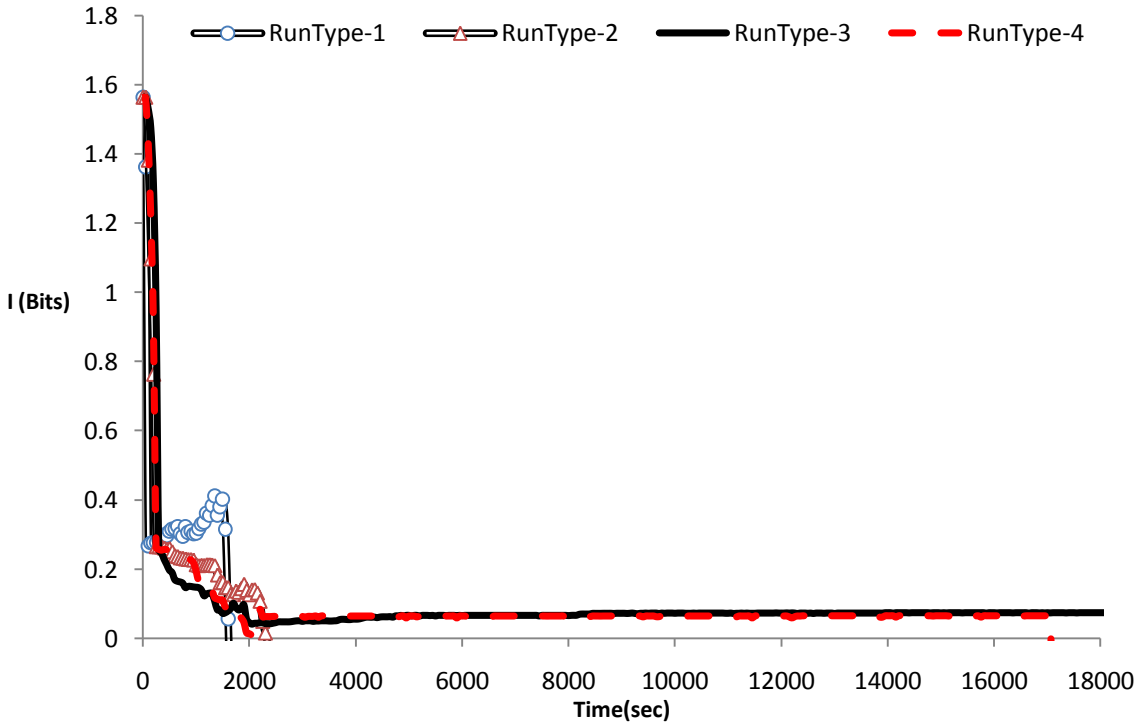
Attack Class 2.5 %



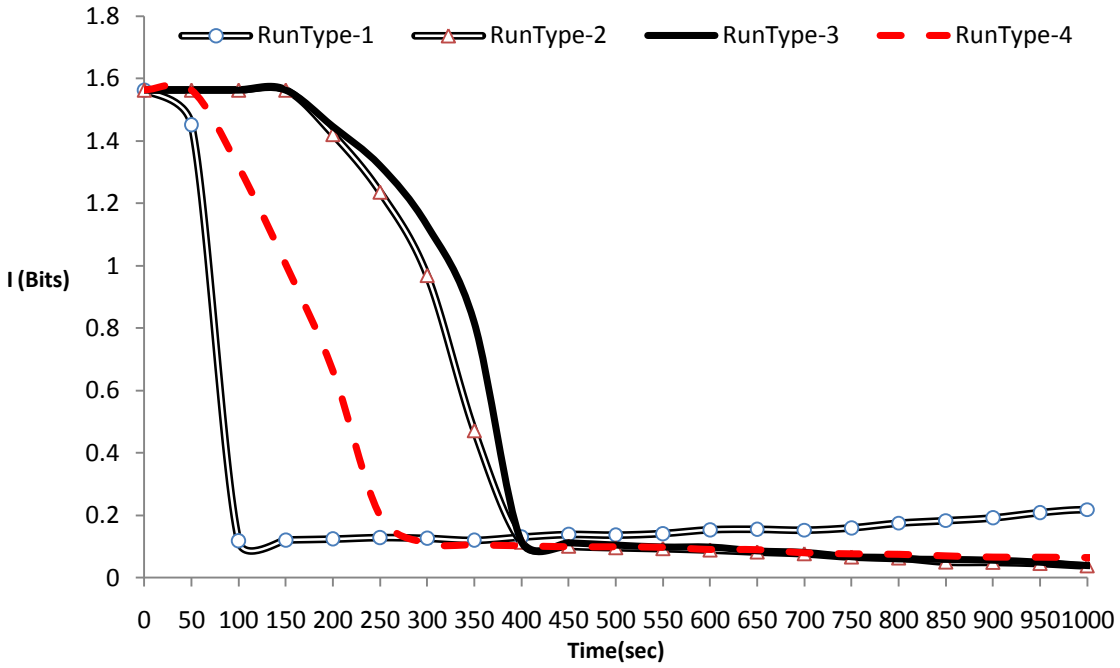
Attack Class 3.0 %



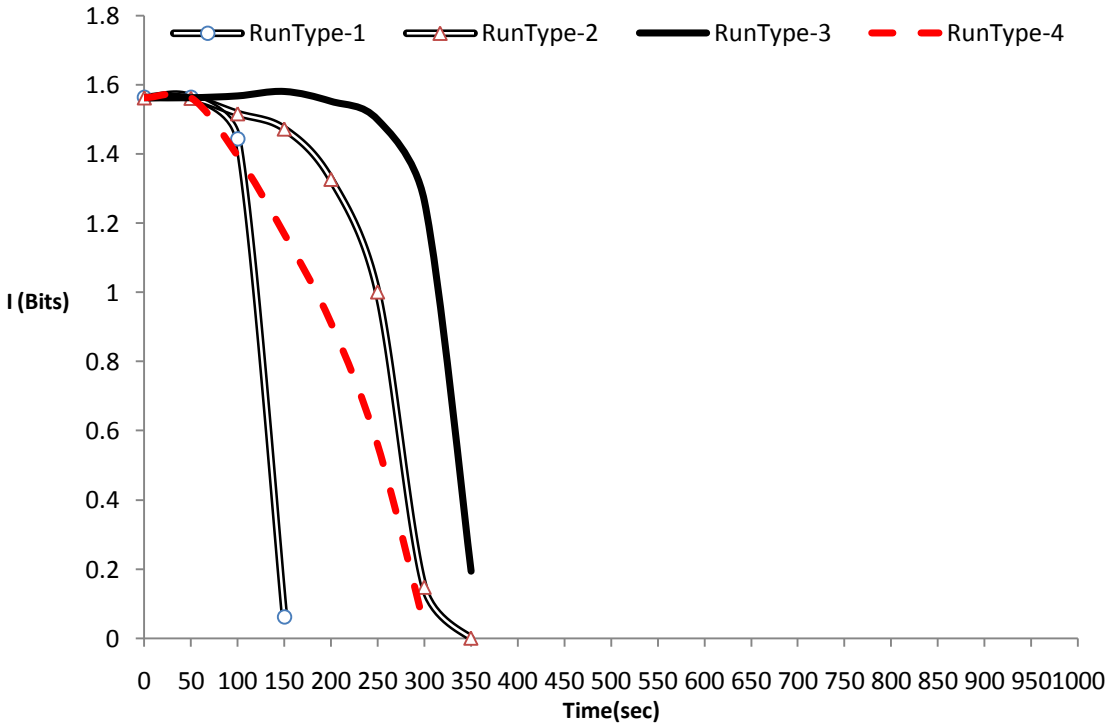
Attack Class 3.5 %



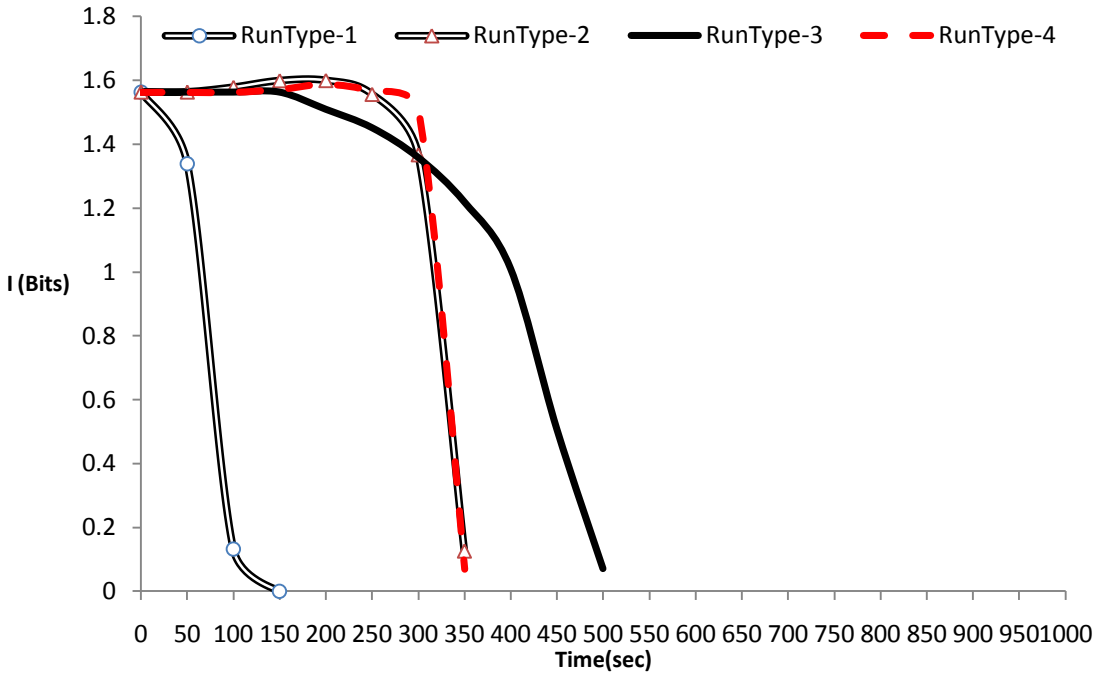
Attack Class 4.0 %



Attack Class 4.5 %



Attack Class 5.0 %

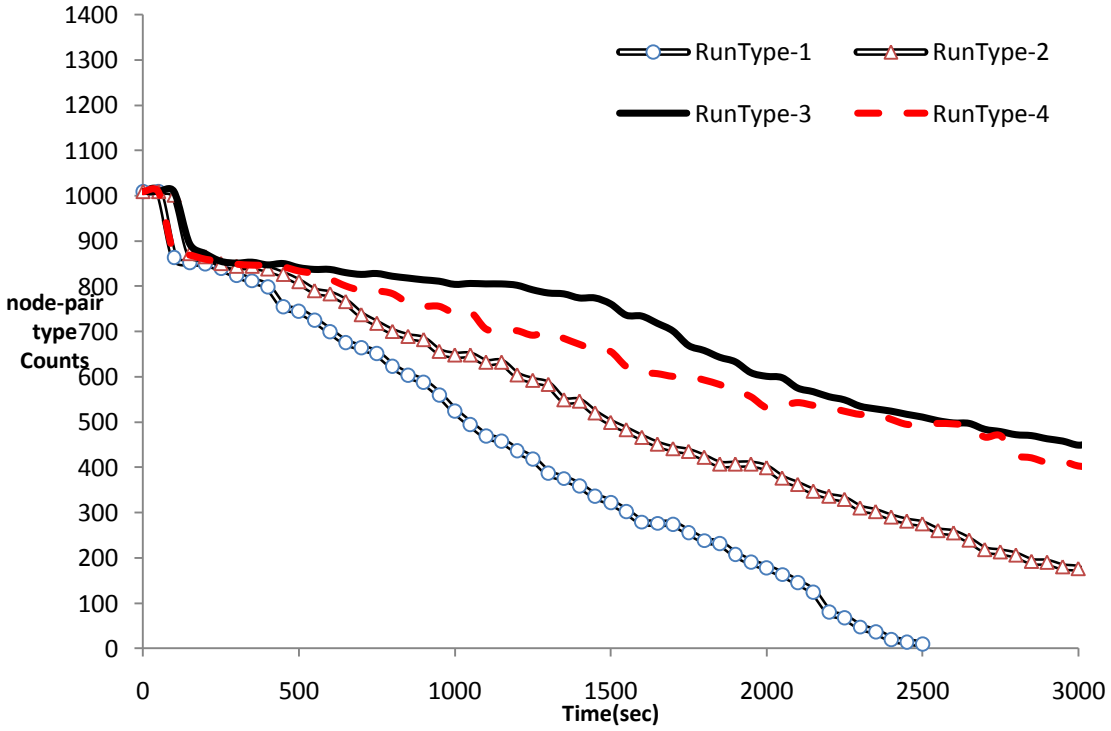


Appendix L

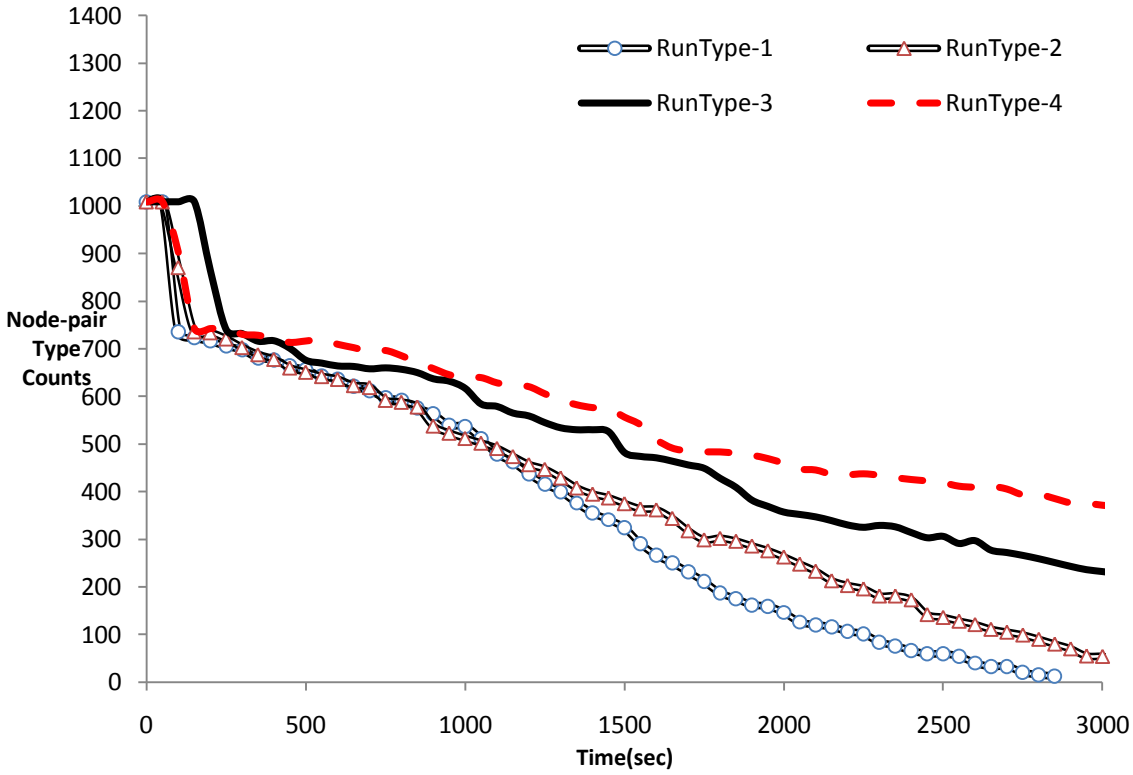
Node-Pair Type 1-2 Counts Results by Attack Class

The Figures in this appendix section represent graphs that were not presented in Chapter V.

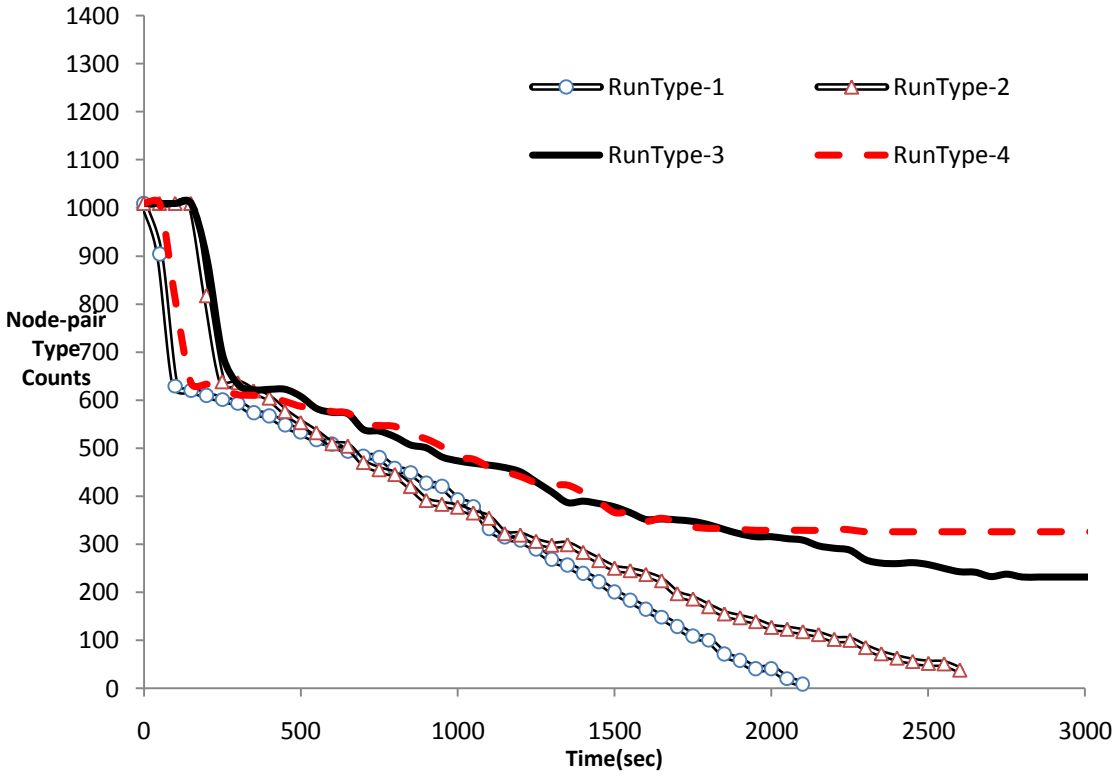
Attack Class 0.5 %



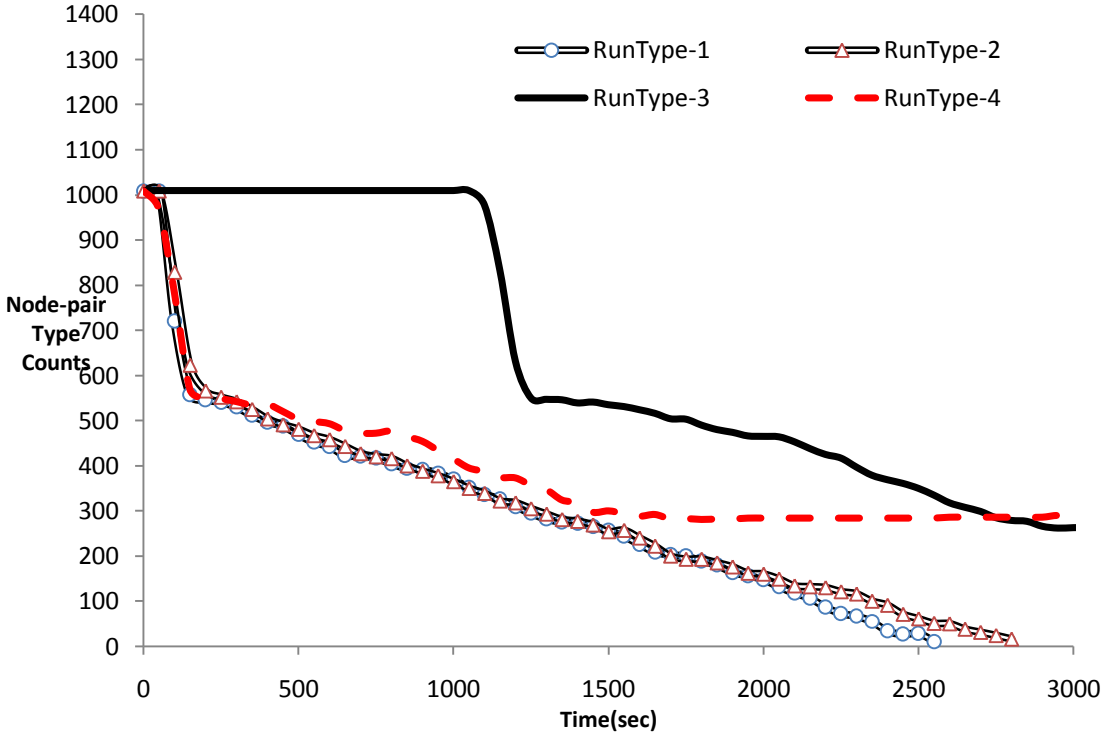
Attack Class 1.0%



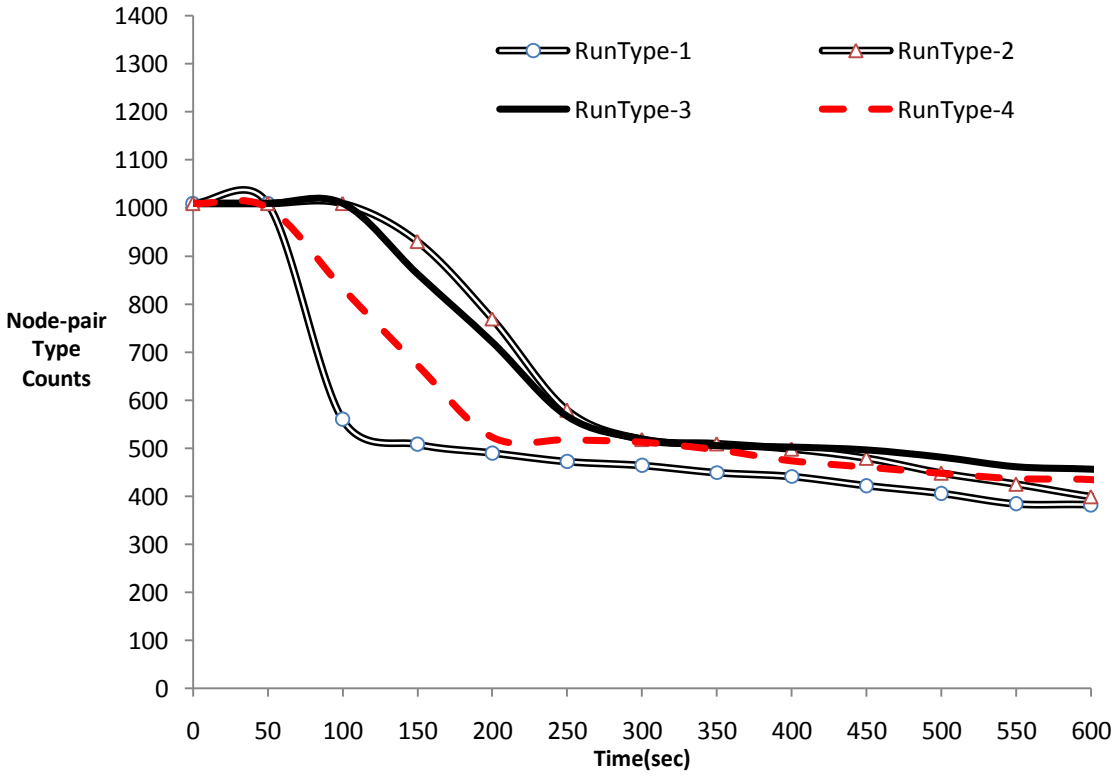
Attack Class 1.5%



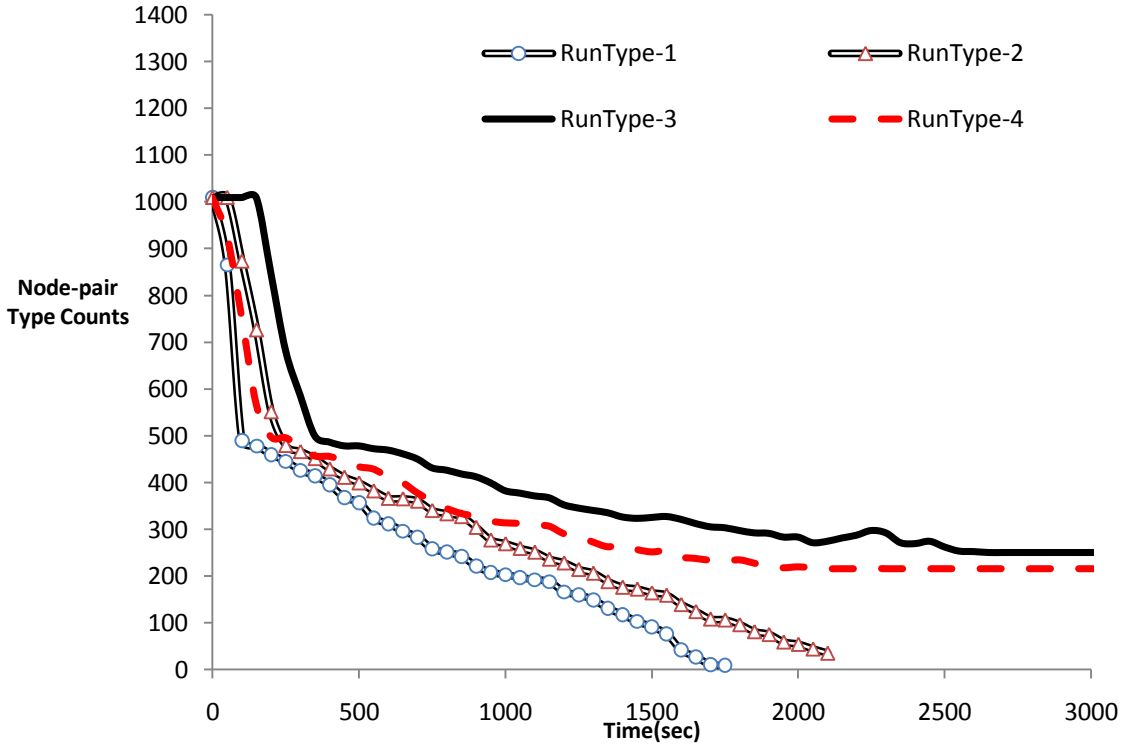
Attack Class 2.0%



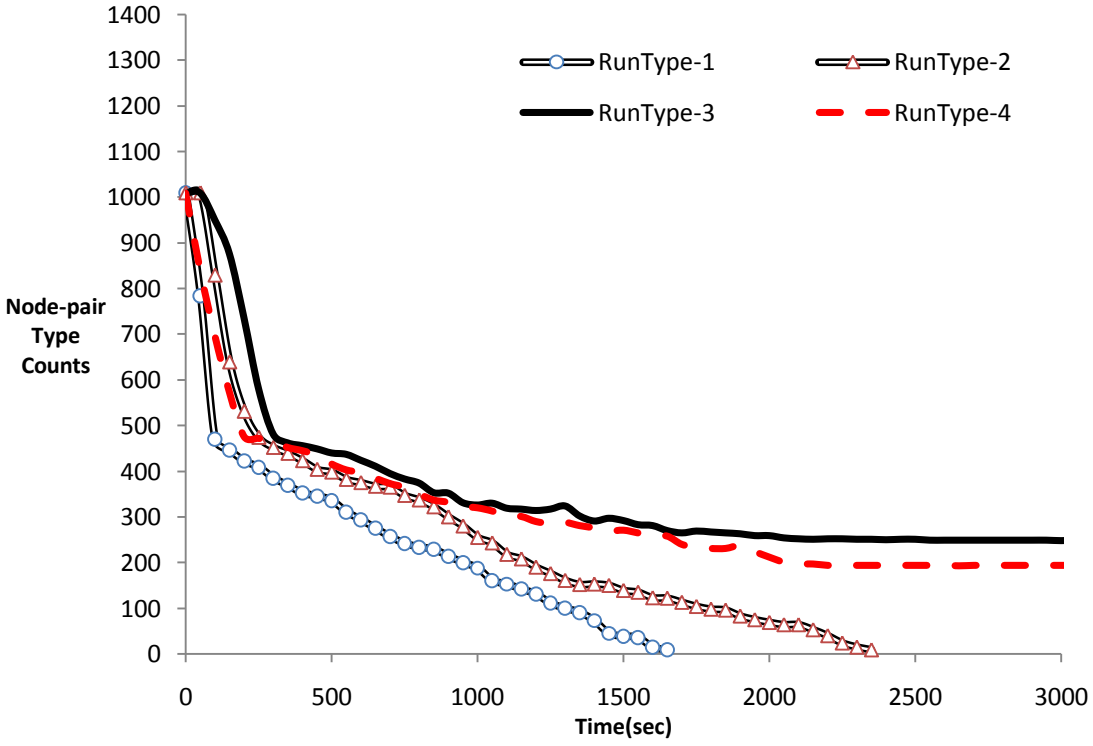
Attack Class 2.5%



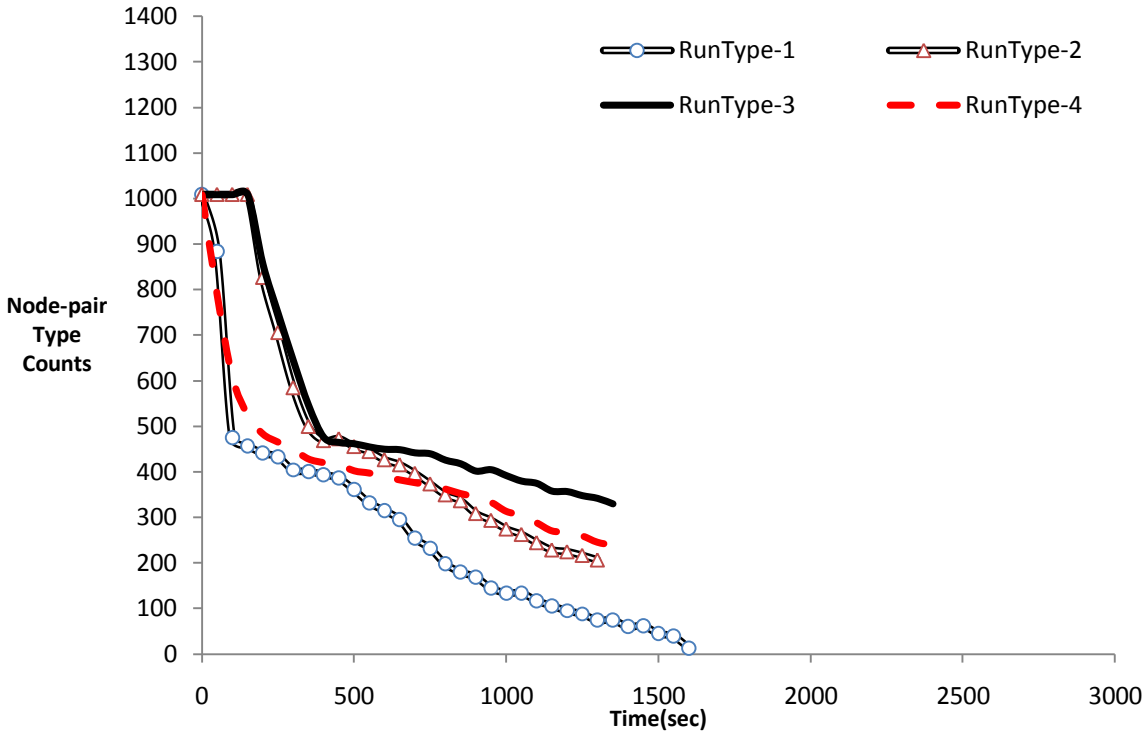
Attack Class 3.0%



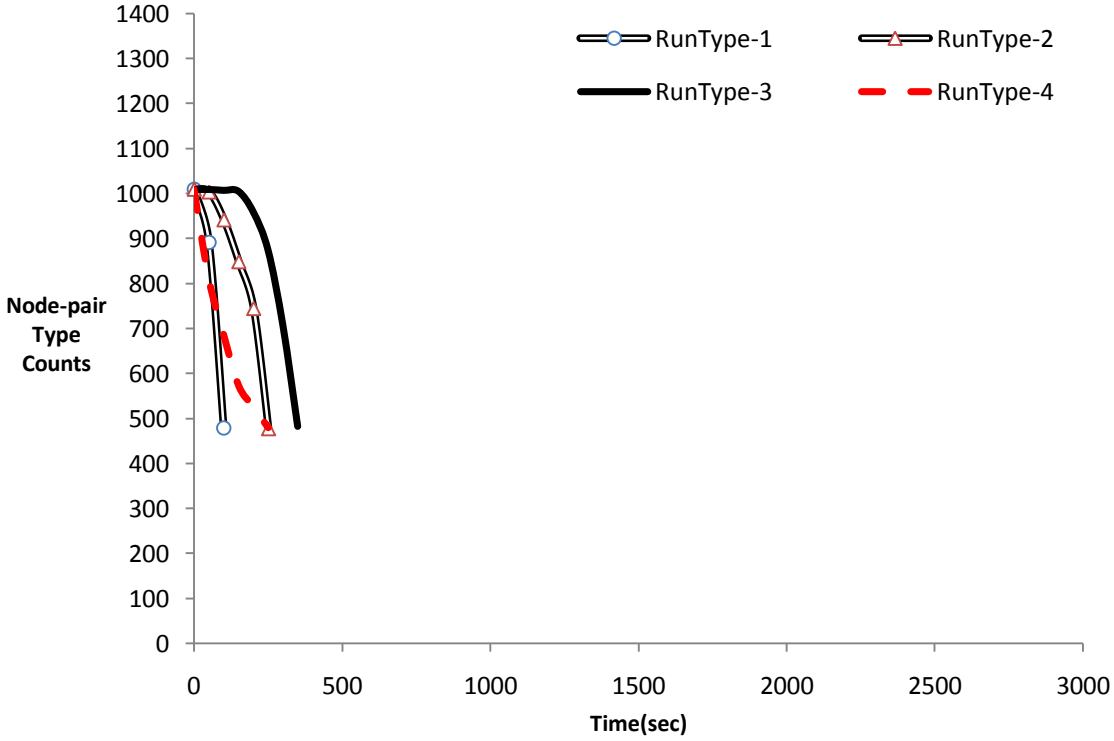
Attack Class 3.5%



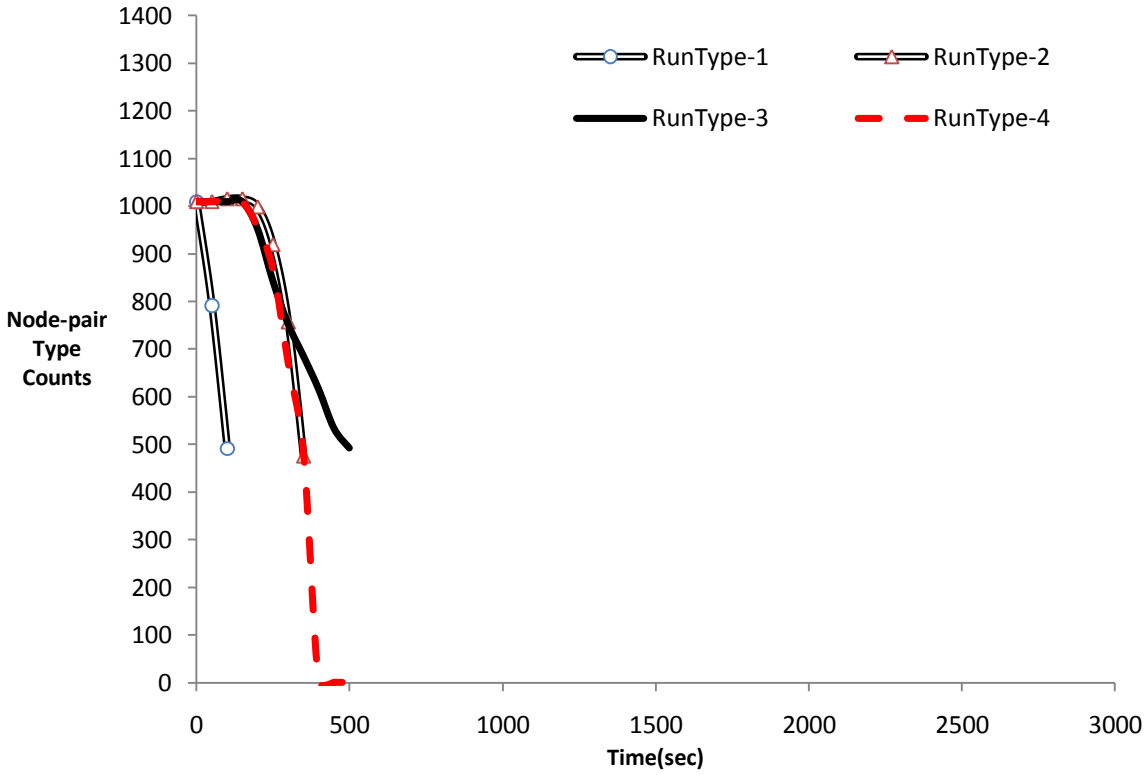
Attack Class 4.0%



Attack Class 4.5%



Attack Class 5.0%



Appendix L

Information Transfer Loss Results by Run Type

The data in this section represents the percent information transfer loss in the first 400 seconds of each simulation. Numbers in parenthesis indicate negative rates (loss). '*' indicates no data due to terminal condition already being met.

Run Type 1 - Information Transfer Loss for the first 400 seconds

	First 100 Seconds		*First 200 Seconds*		*First 300 Seconds*		*First 400 Seconds*	
Attack Class	Loss bits	% Loss	Loss bits	% Loss	Loss bits	% Loss	Loss bits	% Loss
0.5%	0.2	12.7%	0.1	6.1%	0.1	3.7%	0.1	3.2%
1.0%	0.4	24.5%	0.2	11.8%	0.1	7.9%	0.1	6.0%
1.5%	0.5	34.4%	0.3	17.0%	0.2	11.3%	0.1	8.4%
2.0%	0.3	19.6%	0.4	23.9%	0.2	15.7%	0.2	11.7%
2.5%	0.8	49.8%	0.5	29.7%	0.3	19.6%	0.2	14.6%
3.0%	1.1	71.6%	0.6	35.9%	0.4	23.7%	0.3	17.6%
3.5%	1.3	83.1%	0.6	41.3%	0.4	27.3%	0.3	20.3%
4.0%	1.4	92.6%	0.7	46.1%	0.5	30.7%	0.4	23.0%
4.5%	1.5	96.2%	0.8	50.1%	0.5	33.4%	0.4	25.1%
5.0%	1.5	96.2%	0.8	50.1%	0.5	33.4%	0.4	25.1%

Run Type 2 - Information Transfer Loss for the first 400 seconds

	First 100 Seconds		*First 200 Seconds*		*First 300 Seconds*		*First 400 Seconds*	
Attack Class	Loss bits	% Loss	Loss bits	% Loss	Loss bits	% Loss	Loss bits	% Loss
0.5%	0.0	0.2%	0.1	6.5%	0.1	4.4%	0.1	3.3%
1.0%	0.1	9.1%	0.2	12.1%	0.1	8.1%	0.1	6.2%
1.5%	0.0	0.0%	0.1	5.5%	0.2	11.8%	0.1	9.4%
2.0%	0.3	16.1%	0.4	24.0%	0.2	16.0%	0.2	12.2%
2.5%	0.0	0.0%	0.1	7.0%	0.3	20.5%	0.3	16.3%
3.0%	0.1	6.4%	0.4	24.3%	0.4	24.2%	0.3	18.6%
3.5%	0.2	11.7%	0.4	25.7%	0.4	27.8%	0.3	20.9%
4.0%	0.0	0.0%	0.1	4.6%	0.2	12.7%	0.4	23.2%
4.5%	0.0	2.9%	0.1	7.5%	0.5	30.2%	0.4	25.0%
5.0%	0.0	0.0%	0.0	0.0%	0.1	4.2%	0.4	25.1%

Run Type 3 - Information Transfer Loss for the first 400 seconds

	First 100 Seconds		*First 200 Seconds*		*First 300 Seconds*		*First 400 Seconds*	
Attack Class	Loss bits	% Loss	Loss bits	% Loss	Loss bits	% Loss	Loss bits	% Loss
0.5%	0.0	0.0%	0.1	6.5%	0.1	4.5%	0.1	3.4%
1.0%	0.0	0.0%	0.1	4.6%	0.1	8.1%	0.1	6.2%
1.5%	0.0	0.0%	0.0	2.7%	0.2	11.5%	0.1	9.1%
2.0%	0.0	0.0%	0.0	0.0%	0.0	0.0%	0.0	0.0%
2.5%	0.0	0.0%	0.1	9.5%	0.3	20.4%	0.2	15.8%
3.0%	0.0	0.0%	0.1	4.5%	0.2	13.9%	0.3	18.1%
3.5%	0.0	2.0%	0.1	8.0%	0.4	27.9%	0.3	21.4%
4.0%	0.0	0.0%	0.1	3.7%	0.1	9.2%	0.4	23.2%
4.5%	0.0	-0.3%	0.0	0.3%	0.1	6.4%	0.4	25.1%
5.0%	0.0	0.0%	0.0	1.7%	0.1	4.4%	0.1	8.9%

Run Type 4 - Information Transfer Loss for the first 400 seconds

	First 100 Seconds		*First 200 Seconds*		*First 300 Seconds*		*First 400 Seconds*	
Attack Class	Loss bits	% Loss	Loss bits	% Loss	Loss bits	% Loss	Loss bits	% Loss
0.5%	0.0	0.0%	0.1	6.4%	0.1	4.4%	0.1	3.4%
1.0%	0.0	0.0%	0.2	11.8%	0.1	8.1%	0.1	6.2%
1.5%	0.0	0.0%	0.3	17.3%	0.2	12.5%	0.2	9.7%
2.0%	0.0	1.3%	0.4	24.3%	0.3	17.3%	0.2	13.2%
2.5%	0.0	0.4%	0.2	13.3%	0.3	20.8%	0.3	16.3%
3.0%	0.0	3.0%	0.3	22.4%	0.4	24.4%	0.3	18.4%
3.5%	0.1	8.0%	0.3	21.1%	0.4	27.9%	0.3	20.9%
4.0%	0.2	15.7%	0.4	28.8%	0.5	31.0%	0.4	23.4%
4.5%	0.2	10.8%	0.3	20.9%	0.5	32.7%	0.4	25.1%
5.0%	0.0	0.0%	0.0	-0.8%	0.0	1.2%	0.4	25.1%

Appendix N

Rate of Information Transfer Loss Results by Run Type

The data in this section represents the rate of information transfer loss of each simulation. Numbers in parenthesis indicate negative rates (loss). '*' indicates no data due to terminal condition already being met. Each cell represents bits/sec at time t.

Run Type 1 – Rate of Information Transfer Loss

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
First 100	(0.20)	(0.38)	(0.54)	(0.31)	(0.78)	(1.12)	(1.30)	(1.45)	(1.50)	(1.43)
100 to terminal conditions	(0.03)	(0.03)	(0.02)	(0.02)	(0.01)	(0.00)	(0.00)	0.01	*	*

Run Type 2 – Rate of Information Transfer Loss

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
First 400	(0.06)	(0.11)	(0.19)	(0.21)	(0.32)	(0.36)	(0.41)	(0.35)	(0.47)	(0.37)
400 to terminal conditions	(0.03)	(0.04)	(0.04)	(0.03)	(0.03)	(0.02)	(0.01)	(0.01)	*	*

Run Type 3 – Rate of Information Transfer Loss

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
First 400	(0.06)	(0.13)	(0.18)	0.00	(0.31)	(0.32)	(0.41)	(0.30)	(0.37)	(0.13)
400 to 3200	(0.02)	(0.04)	(0.04)	(0.06)	(0.02)	(0.01)	(0.01)	(0.01)	*	*
3200 to terminal conditions	(0.00)	0.00	0.00	0.00	0.00	0.00	0.00	*	*	*

Run Type 4 – Rate of Information Transfer Loss

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
First 400	(0.06)	(0.12)	(0.19)	(0.26)	(0.33)	(0.37)	(0.42)	(0.45)	(0.51)	(0.06)
400 to 3200	(0.01)	0.01	(0.01)	(0.01)	(0.00)	(0.01)	(0.01)	(0.01)	*	(0.01)
3200 to terminal conditions	0.00	0.00	0.00	0.00	(0.00)	0.00	(0.00)		*	*

Appendix N

Node-pair type 1-2 Count Loss Results by Run Type

The data in this section represents node-pair type 1-2 count loss of each simulation. Numbers in parenthesis indicate negative rates (loss). '*' indicates no data due to terminal condition already being met. NP loss column represents the number of node-pairs.

Run Type 1 – Node-pair type 1-2 Count Loss, First 400 Seconds

	First 100 Seconds		*First 200 Seconds*		*First 300 Seconds*		*First 400 Seconds*	
Attack Class	NP loss	% loss	NP loss	% loss	NP loss	% loss	NP loss	% loss
0.5%	0.0	0.0%	78.5	7.8%	56.7	5.6%	49.3	4.9%
1.0%	273.0	27.1%	146.5	14.5%	103.7	10.3%	83.0	8.2%
1.5%	380.0	37.7%	199.5	19.8%	138.7	13.7%	110.5	11.0%
2.0%	288.0	28.5%	231.5	22.9%	159.7	15.8%	128.0	12.7%
2.5%	449.0	44.5%	259.5	25.7%	181.7	18.0%	142.0	14.1%
3.0%	519.0	51.4%	274.5	27.2%	194.3	19.3%	153.5	15.2%
3.5%	539.0	53.4%	293.5	29.1%	208.0	20.6%	164.3	16.3%
4.0%	534.0	52.9%	284.0	28.1%	201.7	20.0%	153.8	15.2%
4.5%	530.0	52.5%	504.5	50.0%	336.3	33.3%	252.3	25.0%
5.0%	531.0	52.6%	504.5	50.0%	336.3	33.3%	252.3	25.0%

Run Type 2 – Node-pair type 1-2 Count Loss, First 400 Seconds

Attack Class	*First 100 Seconds*		*First 200 Seconds*		*First 300 Seconds*		*First 400 Seconds*	
	NP loss	% loss	NP loss	% loss	NP loss	% loss	NP loss	% loss
0.5%	8.0	0.8%	72.0	7.1%	55.0	5.5%	42.8	4.2%
1.0%	138.0	13.7%	137.5	13.6%	102.0	10.1%	82.8	8.2%
1.5%	0.0	0.0%	96.0	9.5%	124.3	12.3%	101.3	10.0%
2.0%	180.0	17.8%	221.5	22.0%	155.7	15.4%	126.3	12.5%
2.5%	0.0	0.0%	120.0	11.9%	163.7	16.2%	127.8	12.7%
3.0%	136.0	13.5%	229.0	22.7%	181.0	17.9%	145.0	14.4%
3.5%	180.0	17.8%	239.0	23.7%	185.7	18.4%	146.5	14.5%
4.0%	0.0	0.0%	91.0	9.0%	141.3	14.0%	135.0	13.4%
4.5%	68.0	6.7%	132.5	13.1%	336.3	33.3%	252.3	25.0%
5.0%	0.0	0.0%	5.5	0.5%	84.0	8.3%	252.3	25.0%

Run Type 3 – Node-pair type 1-2 Count Loss, First 400 Seconds

Attack Class	*First 100 Seconds*		*First 200 Seconds*		*First 300 Seconds*		*First 400 Seconds*	
	NP loss	% loss	NP loss	% loss	NP loss	% loss	NP loss	% loss
0.5%	0.0	0.0%	68.5	6.8%	52.7	5.2%	40.5	4.0%
1.0%	0.0	0.0%	72.0	7.1%	92.3	9.2%	73.0	7.2%
1.5%	0.0	0.0%	57.0	5.6%	125.7	12.5%	96.8	9.6%
2.0%	0.0	0.0%	0.0	0.0%	0.0	0.0%	0.0	0.0%
2.5%	0.0	0.0%	144.5	14.3%	163.3	16.2%	126.8	12.6%
3.0%	0.0	0.0%	81.0	8.0%	141.0	14.0%	130.8	13.0%
3.5%	59.0	5.8%	137.0	13.6%	176.0	17.4%	138.3	13.7%
4.0%	0.0	0.0%	76.5	7.6%	121.7	12.1%	133.3	13.2%
4.5%	2.0	0.2%	25.0	2.5%	100.7	10.0%	252.3	25.0%
5.0%	0.0	0.0%	28.5	2.8%	85.7	8.5%	98.5	9.8%

Run Type 4 – Node-pair type 1-2 Count Loss, First 400 Seconds

	First 100 Seconds		*First 200 Seconds*		*First 300 Seconds*		*First 400 Seconds*	
Attack Class	NP loss	% loss	NP loss	% loss	NP loss	% loss	NP loss	% loss
0.5%	137.0	13.6%	74.5	7.4%	53.3	5.3%	40.8	4.0%
1.0%	107.0	10.6%	133.0	13.2%	92.7	9.2%	71.8	7.1%
1.5%	202.0	20.0%	188.0	18.6%	132.7	13.1%	100.8	10.0%
2.0%	226.0	22.4%	230.0	22.8%	155.7	15.4%	118.3	11.7%
2.5%	177.0	17.5%	243.5	24.1%	165.7	16.4%	134.0	13.3%
3.0%	255.0	25.3%	255.5	25.3%	177.7	17.6%	138.3	13.7%
3.5%	312.0	30.9%	266.5	26.4%	181.3	18.0%	141.0	14.0%
4.0%	402.0	39.8%	263.0	26.1%	186.7	18.5%	147.3	14.6%
4.5%	323.0	32.0%	241.5	23.9%	336.3	33.3%	252.3	25.0%
5.0%	-2.0	-0.2%	26.5	2.6%	112.7	11.2%	122.3	12.1%

Appendix O

Node-pair Type 1-2 Count Loss Rate by Run Type

The data in this section represents the rate of count loss of node-pair type 1-2 for each simulation. Numbers in parenthesis indicate negative rates (loss). '*' indicates no data due to terminal condition already being met.

Run Type 1 – Node-pair Type 1-2 Count Loss (node-pairs/sec)

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
First 100	0.0	(273.0)	(380.0)	(288.0)	(449.0)	(519.0)	(539.0)	(534.0)	(530.0)	(518.0)
100 to terminal conditions	(38.3)	(30.0)	(33.0)	(23.3)	(28.3)	(28.3)	(29.0)	(32.3)	*	*

Run Type 2 – Node-pair Type 1-2 Count Loss (node-pairs/sec)

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
First 400	(50.50)	(87.93)	(130.2)	(137.2)	(162.5)	(168.5)	(165.7)	(161.2)	(201.7)	(126.1)
400 to terminal conditions	(24.04)	(23.54)	(24.75)	(20.27)	(23.43)	(23.21)	(21.23)	(33.01)	*	*

Run Type 3 – Node-pair Type 1-2 Count Loss (node-pairs/sec)

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
First 400	(49.0)	(95.7)	(126.0)	0.00	(160.5)	(159.8)	(169.6)	(150.4)	(134.4)	(107.6)
400 to 3200	(17.71)	(19.03)	(14.10)	(31.90)	(8.46)	(8.34)	(6.31)	(15.67)	*	*
3200 to terminal conditions	(0.12)	(0.02)	0.03	0.00	0.00	0.00	0.02	*	*	*

Run Type 4 – Node-pair Type 1-2 Count Loss (node-pairs/sec)

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
First 400	(39.9)	(77.6)	(107.2)	(123.2)	(148.4)	(141.7)	(132.7)	(127.0)	(206.2)	(139.0)
400 to 3200	(16.7)	(14.2)	(9.9)	(8.0)	(8.3)	(7.1)	(8.1)	(20.1)	*	*
3200 to terminal conditions	(0.1)	(0.1)	(0.0)	(0.0)	(0.1)	(0.0)	(0.0)	*	*	*

Appendix P

Attack Detection by Run Type

Numbers in parenthesis indicate negative rates (loss).

Run Type 1 – Attack Detection (All time in simulation seconds)

Attack Class	*Terminal Condition Time*	*Attack Start Time*	*Attack End Time*	*Attacked Nodes*	*Average Attack Node Degree*	*Attack Intensity (node/sec)*	*Attack Time Detect Research Estimate*	*Percent Variance from Actual Attack Time*
0.5%	25,500	103	108	60	47	12.0	105	2%
1.0%	2,850	64	91	124	41	4.6	65	2%
1.5%	2,100	45	62	179	38	10.5	50	11%
2.0%	2,500	74	114	241	35	6.0	75	1%
2.5%	2,300	76	103	303	31	11.2	80	5%
3.0%	1,750	42	76	363	29	10.7	45	7%
3.5%	1,650	47	88	421	26	10.3	35	-26%
4.0%	1,600	44	82	478	24	12.6	45	2%
4.5%	90	45	89	544	22	12.4	50	11%
5.0%	88	41	86	613	20	13.6	45	10%
Average	4,043	58	90	333	31	10	60	3%
Median	1,925	46	89	333	30	11	50	9%

Run Type 2 – Attack Detection (All time in simulation seconds)

Attack Class	*Terminal Condition Time*	*Attack Start Time*	*Attack End Time*	*Attacked Nodes*	*Average Attack Node Degree*	*Attack Intensity (node/sec)*	*Attack Time Detect Research Estimate*	*Percent Variance from Actual Attack Time*
0.5%	3,800	99	126	60	47	2.2	100	1.0%
1.0%	3,350	70	126	124	41	2.2	75	7.1%
1.5%	2,600	155	235	179	38	2.2	160	3.2%
2.0%	2,800	65	169	241	35	2.3	65	0.0%
2.5%	2,200	124	262	303	31	2.2	130	4.8%
3.0%	2,100	54	213	363	29	2.3	55	1.9%
3.5%	2,350	62	244	421	26	2.3	65	4.8%
4.0%	1,300	153	363	478	24	2.3	155	1.3%
4.5%	288	47	288	544	22	2.3	50	6.4%
5.0%	350	71	350	613	20	2.2	80	12.7%
Average	2,040	82	231	334	32	2.2	86	5.4%
Median	2,275	71	240	333	30	2	78	9.9%

Run Type 3 – Attack Detection (All time in simulation seconds)

Attack Class	*Terminal Condition Time*	*Attack Start Time*	*Attack End Time*	*Attacked Nodes*	*Average Attack Node Degree*	*Attack Intensity (node/sec)*	*Attack Time Detect Research Estimate*	*Percent Variance from Actual Attack Time*
0.5%	25,000	126	157	60	47	1.9	125	(0.8%)
1.0%	25,000	161	228	124	41	1.9	160	(0.6%)
1.5%	25,000	166	261	179	38	1.9	165	(0.6%)
2.0%	25,000	1,084	1,211	241	35	1.9	1,100	1.5%
2.5%	25,000	100	259	303	31	1.9	100	0.0%
3.0%	25,000	159	344	363	29	2.0	155	(2.5%)
3.5%	25,000	56	276	421	26	1.9	55	(1.8%)
4.0%	1,350	153	400	478	24	1.9	150	(2.0%)
4.5%	348	58	348	544	22	1.9	70	20.7%
5.0%	470	155	470	613	20	1.9	155	0.0%
Average	409	107	409	579	21	1.9	113	10.3%
Median	25,000	154	310	333	30	2	153	(1.0%)

Run Type 4 – Attack Detection (All time in simulation seconds)

Attack Class	*Terminal Condition Time*	*Attack Start Time*	*Attack End Time*	*Attacked Nodes*	*Average Attack Node Degree*	*Attack Intensity (node/sec)*	*Attack Time Detect Research Estimate*	*Percent Variance from Actual Attack Time*
0.5%	25,000	108	135	60	47	2.2	105	(2.8%)
1.0%	25,000	124	181	124	41	2.2	125	0.8%
1.5%	25,000	102	182	179	38	2.2	100	(2.0%)
2.0%	25,000	82	192	241	35	2.2	80	(2.4%)
2.5%	25,000	94	232	303	31	2.2	95	1.1%
3.0%	25,000	52	213	363	29	2.3	50	(3.8%)
3.5%	17,050	55	242	421	26	2.3	55	0.0%
4.0%	1,450	53	258	478	24	2.3	50	(5.7%)
4.5%	286	54	286	544	22	2.3	50	(7.4%)
5.0%	384	105	384	613	20	2.2	115	9.5%
Average	15,028	90	238	334	32	2.2	90	(0.7%)
Median	25,000	88	223	333	30	2	88	(0.6%)

Appendix Q

Attack Detection and Node-Pair Type 1-1 Counts Results by Run Type

This data represents the counts of node-pair type 1-1 used for attack detection. It depicts the first 200 seconds of each simulation as discussed in Chapter VI. Blank cells indicate no data due to terminal condition already being met. Each cell represents node-pair type 1-1 count at simulation time t.

Run Type 1 – Attack Detection

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
0	36	36	36	36	36	36	36	36	36	36
5	36	36	36	36	36	36	36	36	36	36
10	36	36	36	36	36	36	36	36	36	36
15	36	36	36	36	36	36	36	36	36	36
20	36	36	36	36	36	36	36	36	36	36
25	36	36	36	36	36	36	36	36	36	36
30	36	36	36	36	36	36	36	36	36	36
35	36	36	36	36	36	36	74	36	36	36
40	36	36	36	36	36	36	319	36	36	36
45	36	36	36	36	36	186	593	132	36	352
50	36	36	360	36	36	471	761	428	414	698
55	36	36	708	36	36	769	958	827	750	1033
60	36	36	1080	36	36	993	1246	1087	1043	1287
65	36	64	1204	36	36	1224	1505	1276	1285	1472
70	36	176	1198	36	36	1482	1760	1542	1468	1649
80	36	506	1194	250	356	1768	1834	1812	1770	1859

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
85	36	680	1188	408	674	1752	1828	1892	1877	1933
90	36	879	1180	567	1002	1746	1824	1886	1937	1983
95	36	913	1176	751	1276	1736	1818	1883		
100	36	925	1170	907	1475	1728	1812	1880		
105	214	927	1164	1111	1607	1718	1812	1868		
110	497	933	1162	1310	1601	1708	1802	1866		
115	497	929	1158	1448	1593	1696	1790	1854		
120	485	935	1158	1446	1583	1684	1786	1854		
125	483	941	1158	1442	1571	1676	1786	1848		
130	481	943	1160	1440	1559	1662	1776	1836		
135	481	942	1156	1440	1547	1658	1774	1834		
140	481	942	1156	1432	1545	1652	1766	1832		
145	481	941	1150	1430	1535	1646	1762	1818		
150	481	935	1148	1428	1525	1643	1752	1810		

Run Type 2 – Attack Detection

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
0	36	36	36	36	36	36	36	36	36	36
5	36	36	36	36	36	36	36	36	36	36
10	36	36	36	36	36	36	36	36	36	36
15	36	36	36	36	36	36	36	36	36	36
20	36	36	36	36	36	36	36	36	36	36
25	36	36	36	36	36	36	36	36	36	36
30	36	36	36	36	36	36	36	36	36	36
35	36	36	36	36	36	36	36	36	36	36
40	36	36	36	36	36	36	36	36	36	36
45	36	36	36	36	36	36	36	36	36	36
50	36	36	36	36	36	36	36	36	56	36
55	36	36	36	36	36	50	36	36	82	36
60	36	36	36	36	36	84	36	36	116	36
65	36	36	36	46	36	134	90	36	146	36
70	36	36	36	161	36	196	162	36	166	36
75	36	106	36	247	36	244	234	36	190	36
80	36	174	36	335	36	304	322	36	233	52
85	36	240	36	425	36	348	396	36	267	54
90	36	324	36	485	36	402	462	36	303	62

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
95	36	424	36	555	36	459	522	36	331	62
100	58	476	36	641	36	491	604	36	353	66
105	150	546	36	703	36	521	672	36	373	70
110	220	616	36	771	36	571	734	36	409	80
115	310	690	36	833	36	643	801	36	439	86
120	387	785	36	909	36	672	857	36	469	88
125	483	879	36	964	38	725	933	36	507	102
130	499	907	36	1053	72	781	995	36	553	108
135	507	907	36	1091	134	828	1029	36	581	116
140	509	911	36	1171	180	872	1085	36	591	128
145	509	914	36	1225	244	937	1147	36	631	138
150	509	919	36	1261	302	985	1189	36	669	142
155	509	922	36	1325	338	1043	1253	80	697	156
160	509	925	110	1387	401	1093	1285	154	749	158
165	520	926	180	1435	461	1165	1317	200	775	170
170	521	927	238	1453	495	1210	1345	287	797	182
175	521	928	302	1453	545	1268	1393	335	855	198
180	521	931	350	1454	615	1334	1433	385	889	214
185	521	932	406	1459	661	1405	1463	445	913	220
190	521	936	489	1462	701	1465	1493	497	959	236
195	524	941	566	1465	759	1539	1535	541	1008	268
200	529	942	628	1468	820	1595	1598	591	1043	300
155	36	36	36	36	36	36	36	36	36	36
160	36	36	36	36	36	36	36	36	36	36
165	36	36	36	36	36	36	36	36	36	36
170	36	36	36	36	36	36	36	36	36	36
175	36	36	36	36	36	36	36	36	36	36
180	36	36	36	36	36	36	36	36	36	36
185	36	36	36	36	36	36	36	36	36	36
190	36	36	36	36	36	36	36	36	36	36
195	36	36	36	36	36	36	36	36	36	36
200	36	36	36	36	36	36	36	36	36	36

Run Type 3 – Attack Detection

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
0	36	36	36	36	36	36	36	36	36	36
5	36	36	36	36	36	36	36	36	36	36
10	36	36	36	36	36	36	36	36	36	36
15	36	36	36	36	36	36	36	36	36	36
20	36	36	36	36	36	36	36	36	36	36
25	36	36	36	36	36	36	36	36	36	36
30	36	36	36	36	36	36	36	36	36	36
35	36	36	36	36	36	36	36	36	36	36

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
40	36	36	36	36	36	36	36	36	36	36
45	36	36	36	36	36	36	36	36	36	36
50	36	36	36	36	36	36	36	36	36	36
55	36	36	36	36	36	36	60	36	36	36
60	36	36	36	36	36	36	76	36	42	36
65	36	36	36	36	36	36	102	36	50	36
70	36	36	36	36	36	36	126	36	50	36
75	36	36	36	36	36	36	156	36	52	36
80	36	36	36	36	36	36	174	36	66	36
85	36	36	36	36	36	36	191	36	72	36
90	36	36	36	36	36	36	233	36	78	36
95	36	36	36	36	36	36	271	36	84	36
100	36	36	36	36	78	36	289	36	102	36
105	36	36	36	36	124	36	317	36	102	36
110	36	36	36	36	172	36	337	36	106	36
115	36	36	36	36	206	36	385	36	118	36
120	36	36	36	36	234	36	421	36	122	36
125	133	36	36	36	298	36	475	36	124	36
130	199	36	36	36	357	36	501	36	136	36
135	273	36	36	36	401	36	525	36	148	36
140	341	36	36	36	443	36	559	36	160	36
145	431	36	36	36	497	36	584	36	164	36
150	476	36	36	36	557	36	636	62	180	36
155	500	36	36	36	587	50	686	98	196	78
160	503	78	36	36	645	124	714	155	200	120
165	505	132	93	36	669	192	766	229	232	152
170	507	190	139	36	713	244	808	265	260	186
175	511	240	187	36	749	300	846	331	298	214
180	512	300	257	36	815	348	894	379	326	238
185	515	364	295	36	867	404	952	431	346	265
190	515	420	333	36	921	480	996	467	364	287
195	517	474	383	36	964	522	1028	494	390	303
200	518	542	443	36	1014	566	1070	532	430	347
155	36	36	36	36	36	36	36	36	36	36
160	36	36	36	36	36	36	36	36	36	36
165	36	36	36	36	36	36	36	36	36	36
170	36	36	36	36	36	36	36	36	36	36
175	36	36	36	36	36	36	36	36	36	36
180	36	36	36	36	36	36	36	36	36	36
185	36	36	36	36	36	36	36	36	36	36
190	36	36	36	36	36	36	36	36	36	36
195	36	36	36	36	36	36	36	36	36	36
200	36	36	36	36	36	36	36	36	36	36

Run Type 4 – Attack Detection

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
0	36	36	36	36	36	36	36	36	36	36
5	36	36	36	36	36	36	36	36	36	36
10	36	36	36	36	36	36	36	36	36	36
15	36	36	36	36	36	36	36	36	36	36
20	36	36	36	36	36	36	36	36	36	36
25	36	36	36	36	36	36	36	36	36	36
30	36	36	36	36	36	36	36	36	36	36
35	36	36	36	36	36	36	36	36	36	36
40	36	36	36	36	36	36	36	36	36	36
45	36	36	36	36	36	36	36	36	36	36
50	36	36	36	36	36	50	36	68	54	36
55	36	36	36	36	36	76	112	172	126	36
60	36	36	36	36	36	113	180	240	190	36
65	36	36	36	36	36	141	232	306	244	36
70	36	36	36	36	36	161	297	384	332	36
75	36	36	36	36	36	199	329	448	400	36
80	36	36	36	58	36	235	399	526	440	36
85	36	36	36	94	36	283	453	584	535	36
90	36	36	36	152	38	333	511	648	573	36
95	36	36	36	196	72	371	543	734	647	36
100	36	36	76	224	134	407	585	782	717	36
105	78	36	120	262	180	443	663	869	753	42
110	160	36	193	320	232	497	715	911	799	46
115	242	36	239	378	296	551	785	987	865	52
120	314	38	301	453	336	583	835	1061	927	54
125	399	102	383	493	395	666	881	1097	969	58
130	499	170	437	559	455	690	903	1157	1005	72
135	499	244	497	633	495	760	959	1217	1037	78
140	499	300	579	691	545	846	1002	1261	1078	84
145	501	364	647	747	615	900	1054	1301	1102	90
150	503	428	725	827	651	961	1106	1329	1158	108

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
155	503	530	793	899	699	1007	1136	1366	1190	114
160	505	596	871	977	757	1081	1182	1392	1230	124
165	505	670	949	1041	819	1151	1210	1436	1268	130
170	505	756	1031	1117	868	1215	1260	1489	1330	132
175	505	869	1144	1197	924	1293	1326	1517	1377	140
180	505	903	1208	1285	976	1331	1388	1561	1433	152
185	505	903	1210	1402	1044	1393	1441	1619	1463	168
190	505	903	1210	1454	1102	1500	1485	1652	1499	172
195	505	903	1210	1456	1158	1553	1515	1684	1527	188
200	521	903	1214	1460	1226	1639	1561	1728	1553	194

Appendix R

Attack Detection and Node-Pair Type 1-1 Counts Results for Fraction Changed by Run Type

This data represents the fraction change in counts of node-pair type 1-1 from the pre-attack state. This data was used for attack detection. It depicts the first 200 seconds of each simulation as discussed in Chapter VI. Blank cells indicate no data due to terminal condition already being met. Each cell represents the increase in the number of node-pairs of type 1-1 of the previous number of node-pairs at the previous time.

Run Type 1 – Attack Detection (fraction change of node-pairs from previous time interval)

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0	0.0	1.1	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0	0.0	7.9	0.0	0.0	0.0
45	0.0	0.0	0.0	0.0	0.0	4.2	15.5	2.7	0.0	8.8
50	0.0	0.0	9.0	0.0	0.0	12.1	20.1	10.9	10.5	18.4
55	0.0	0.0	18.7	0.0	0.0	20.4	25.6	22.0	19.8	27.7
60	0.0	0.0	29.0	0.0	0.0	26.6	33.6	29.2	28.0	34.8
65	0.0	0.8	32.4	0.0	0.0	33.0	40.8	34.4	34.7	39.9
70	0.0	3.9	32.3	0.0	0.0	40.2	47.9	41.8	39.8	44.8
75	0.0	9.0	32.2	1.3	0.0	48.1	50.1	45.8	45.3	47.6
80	0.0	13.1	32.2	5.9	8.9	48.1	49.9	49.3	48.2	50.6

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
85	0.0	17.9	32.0	10.3	17.7	47.7	49.8	51.6	51.1	52.7
90	0.0	23.4	31.8	14.8	26.8	47.5	49.7	51.4	52.8	54.1
95	0.0	24.4	31.7	19.9	34.4	47.2	49.5	51.3		
100	0.0	24.7	31.5	24.2	40.0	47.0	49.3	51.2		
105	4.9	24.8	31.3	29.9	43.6	46.7	49.3	50.9		
110	12.8	24.9	31.3	35.4	43.5	46.4	49.1	50.8		
115	12.8	24.8	31.2	39.2	43.3	46.1	48.7	50.5		
120	12.5	25.0	31.2	39.2	43.0	45.8	48.6	50.5		
125	12.4	25.1	31.2	39.1	42.6	45.6	48.6	50.3		
130	12.4	25.2	31.2	39.0	42.3	45.2	48.3	50.0		
135	12.4	25.2	31.1	39.0	42.0	45.1	48.3	49.9		
140	12.4	25.2	31.1	38.8	41.9	44.9	48.1	49.9		
145	12.4	25.1	30.9	38.7	41.6	44.7	47.9	49.5		
150	12.4	25.0	30.9	38.7	41.4	44.6	47.7	49.3		

Run Type 2 – Attack Detection (fraction change of node-pairs from previous time interval)

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
55	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0
60	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	1.3	0.0
65	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0	2.2	0.0
70	0.0	0.0	0.0	0.3	0.0	2.7	1.5	0.0	3.1	0.0
75	0.0	0.0	0.0	3.5	0.0	4.4	3.5	0.0	3.6	0.0
80	0.0	1.9	0.0	5.9	0.0	5.8	5.5	0.0	4.3	0.0
85	0.0	3.8	0.0	8.3	0.0	7.4	7.9	0.0	5.5	0.4
90	0.0	5.7	0.0	10.8	0.0	8.7	10.0	0.0	6.4	0.5
95	0.0	8.0	0.0	12.5	0.0	10.2	11.8	0.0	7.4	0.7

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
100	0.0	10.8	0.0	14.4	0.0	11.8	13.5	0.0	8.2	0.7
105	0.6	12.2	0.0	16.8	0.0	12.6	15.8	0.0	8.8	0.8
110	3.2	14.2	0.0	18.5	0.0	13.5	17.7	0.0	9.4	0.9
115	5.1	16.1	0.0	20.4	0.0	14.9	19.4	0.0	10.4	1.2
120	7.6	18.2	0.0	22.1	0.0	16.9	21.3	0.0	11.2	1.4
125	9.8	20.8	0.0	24.3	0.0	17.7	22.8	0.0	12.0	1.4
130	12.4	23.4	0.0	25.8	0.1	19.1	24.9	0.0	13.1	1.8
135	12.9	24.2	0.0	28.3	1.0	20.7	26.6	0.0	14.4	2.0
140	13.1	24.2	0.0	29.3	2.7	22.0	27.6	0.0	15.1	2.2
145	13.1	24.3	0.0	31.5	4.0	23.2	29.1	0.0	15.4	2.6
150	13.1	24.4	0.0	33.0	5.8	25.0	30.9	0.0	16.5	2.8
155	13.1	24.5	0.0	34.0	7.4	26.4	32.0	0.0	17.6	2.9
160	13.1	24.6	0.0	35.8	8.4	28.0	33.8	1.2	18.4	3.3
165	13.1	24.7	2.1	37.5	10.1	29.4	34.7	3.3	19.8	3.4
170	13.4	24.7	4.0	38.9	11.8	31.4	35.6	4.6	20.5	3.7
175	13.5	24.8	5.6	39.4	12.8	32.6	36.4	7.0	21.1	4.1
180	13.5	24.8	7.4	39.4	14.1	34.2	37.7	8.3	22.8	4.5
185	13.5	24.9	8.7	39.4	16.1	36.1	38.8	9.7	23.7	4.9
190	13.5	24.9	10.3	39.5	17.4	38.0	39.6	11.4	24.4	5.1
195	13.5	25.0	12.6	39.6	18.5	39.7	40.5	12.8	25.6	5.6
200	13.6	25.1	14.7	39.7	20.1	41.8	41.6	14.0	27.0	6.4
155	13.7	25.2	16.4	39.8	21.8	43.3	43.4	15.4	28.0	7.3
160	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
165	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
170	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
175	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
180	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
185	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
190	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
195	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
200	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Run Type 3 – Attack Detection (fraction change of node-pairs from previous time interval)

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
55	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0
60	0.0	0.0	0.0	0.0	0.0	0.0	1.1	0.0	0.2	0.0
65	0.0	0.0	0.0	0.0	0.0	0.0	1.8	0.0	0.4	0.0
70	0.0	0.0	0.0	0.0	0.0	0.0	2.5	0.0	0.4	0.0
75	0.0	0.0	0.0	0.0	0.0	0.0	3.3	0.0	0.4	0.0
80	0.0	0.0	0.0	0.0	0.0	0.0	3.8	0.0	0.8	0.0
85	0.0	0.0	0.0	0.0	0.0	0.0	4.3	0.0	1.0	0.0
90	0.0	0.0	0.0	0.0	0.0	0.0	5.5	0.0	1.2	0.0
95	0.0	0.0	0.0	0.0	0.0	0.0	6.5	0.0	1.3	0.0
100	0.0	0.0	0.0	0.0	1.2	0.0	7.0	0.0	1.8	0.0
105	0.0	0.0	0.0	0.0	2.4	0.0	7.8	0.0	1.8	0.0
110	0.0	0.0	0.0	0.0	3.8	0.0	8.4	0.0	1.9	0.0
115	0.0	0.0	0.0	0.0	4.7	0.0	9.7	0.0	2.3	0.0
120	0.0	0.0	0.0	0.0	5.5	0.0	10.7	0.0	2.4	0.0
125	2.7	0.0	0.0	0.0	7.3	0.0	12.2	0.0	2.4	0.0
130	4.5	0.0	0.0	0.0	8.9	0.0	12.9	0.0	2.8	0.0
135	6.6	0.0	0.0	0.0	10.1	0.0	13.6	0.0	3.1	0.0
140	8.5	0.0	0.0	0.0	11.3	0.0	14.5	0.0	3.4	0.0
145	11.0	0.0	0.0	0.0	12.8	0.0	15.2	0.0	3.6	0.0
150	12.2	0.0	0.0	0.0	14.5	0.0	16.7	0.7	4.0	0.0
155	12.9	0.0	0.0	0.0	15.3	0.4	18.1	1.7	4.4	1.2
160	13.0	1.2	0.0	0.0	16.9	2.4	18.8	3.3	4.6	2.3
165	13.0	2.7	1.6	0.0	17.6	4.3	20.3	5.4	5.4	3.2
170	13.1	4.3	2.9	0.0	18.8	5.8	21.4	6.4	6.2	4.2
175	13.2	5.7	4.2	0.0	19.8	7.3	22.5	8.2	7.3	4.9
180	13.2	7.3	6.1	0.0	21.6	8.7	23.8	9.5	8.1	5.6
185	13.3	9.1	7.2	0.0	23.1	10.2	25.4	11.0	8.6	6.4

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
190	13.3	10.7	8.3	0.0	24.6	12.3	26.7	12.0	9.1	7.0
195	13.4	12.2	9.6	0.0	25.8	13.5	27.6	12.7	9.8	7.4
200	13.4	14.1	11.3	0.0	27.2	14.7	28.7	13.8	10.9	8.6

Run Type 4 – Attack Detection (fraction change of node-pairs from previous time interval)

Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.9	0.5	0.0
55	0.0	0.0	0.0	0.0	0.0	1.1	2.1	3.8	2.5	0.0
60	0.0	0.0	0.0	0.0	0.0	2.1	4.0	5.7	4.3	0.0
65	0.0	0.0	0.0	0.0	0.0	2.9	5.4	7.5	5.8	0.0
70	0.0	0.0	0.0	0.0	0.0	3.5	7.3	9.7	8.2	0.0
75	0.0	0.0	0.0	0.0	0.0	4.5	8.1	11.4	10.1	0.0
80	0.0	0.0	0.0	0.6	0.0	5.5	10.1	13.6	11.2	0.0
85	0.0	0.0	0.0	1.6	0.0	6.9	11.6	15.2	13.9	0.0
90	0.0	0.0	0.0	3.2	0.1	8.3	13.2	17.0	14.9	0.0
95	0.0	0.0	0.0	4.4	1.0	9.3	14.1	19.4	17.0	0.0
100	0.0	0.0	1.1	5.2	2.7	10.3	15.3	20.7	18.9	0.0
105	1.2	0.0	2.3	6.3	4.0	11.3	17.4	23.1	19.9	0.2
110	3.4	0.0	4.4	7.9	5.4	12.8	18.9	24.3	21.2	0.3
115	5.7	0.0	5.6	9.5	7.2	14.3	20.8	26.4	23.0	0.4

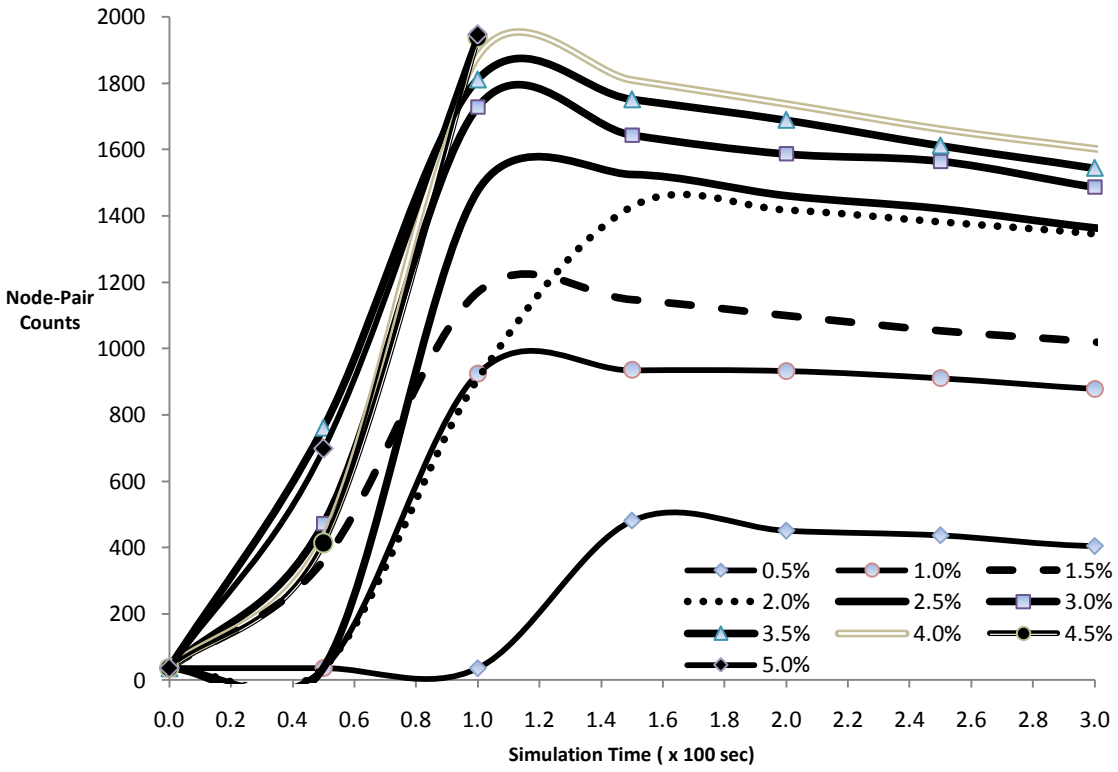
Time (sec)	*0.5%*	*1.0%*	*1.5%*	*2.0%*	*2.5%*	*3.0%*	*3.5%*	*4.0%*	*4.5%*	*5.0%*
120	7.7	0.1	7.4	11.6	8.3	15.2	22.2	28.5	24.8	0.5
125	10.1	1.8	9.6	12.7	10.0	17.5	23.5	29.5	25.9	0.6
130	12.9	3.7	11.1	14.5	11.6	18.2	24.1	31.1	26.9	1.0
135	12.9	5.8	12.8	16.6	12.8	20.1	25.6	32.8	27.8	1.2
140	12.9	7.3	15.1	18.2	14.1	22.5	26.8	34.0	28.9	1.3
145	12.9	9.1	17.0	19.8	16.1	24.0	28.3	35.1	29.6	1.5
150	13.0	10.9	19.1	22.0	17.1	25.7	29.7	35.9	31.2	2.0
155	13.0	13.7	21.0	24.0	18.4	27.0	30.6	36.9	32.1	2.2
160	13.0	15.6	23.2	26.1	20.0	29.0	31.8	37.7	33.2	2.4
165	13.0	17.6	25.4	27.9	21.8	31.0	32.6	38.9	34.2	2.6
170	13.0	20.0	27.6	30.0	23.1	32.8	34.0	40.4	35.9	2.7
175	13.0	23.1	30.8	32.3	24.7	34.9	35.8	41.1	37.3	2.9
180	13.0	24.1	32.6	34.7	26.1	36.0	37.6	42.4	38.8	3.2
185	13.0	24.1	32.6	37.9	28.0	37.7	39.0	44.0	39.6	3.7
190	13.0	24.1	32.6	39.4	29.6	40.7	40.3	44.9	40.6	3.8
195	13.0	24.1	32.6	39.4	31.2	42.1	41.1	45.8	41.4	4.2
200	13.5	24.1	32.7	39.6	33.1	44.5	42.4	47.0	42.1	4.4

Appendix T

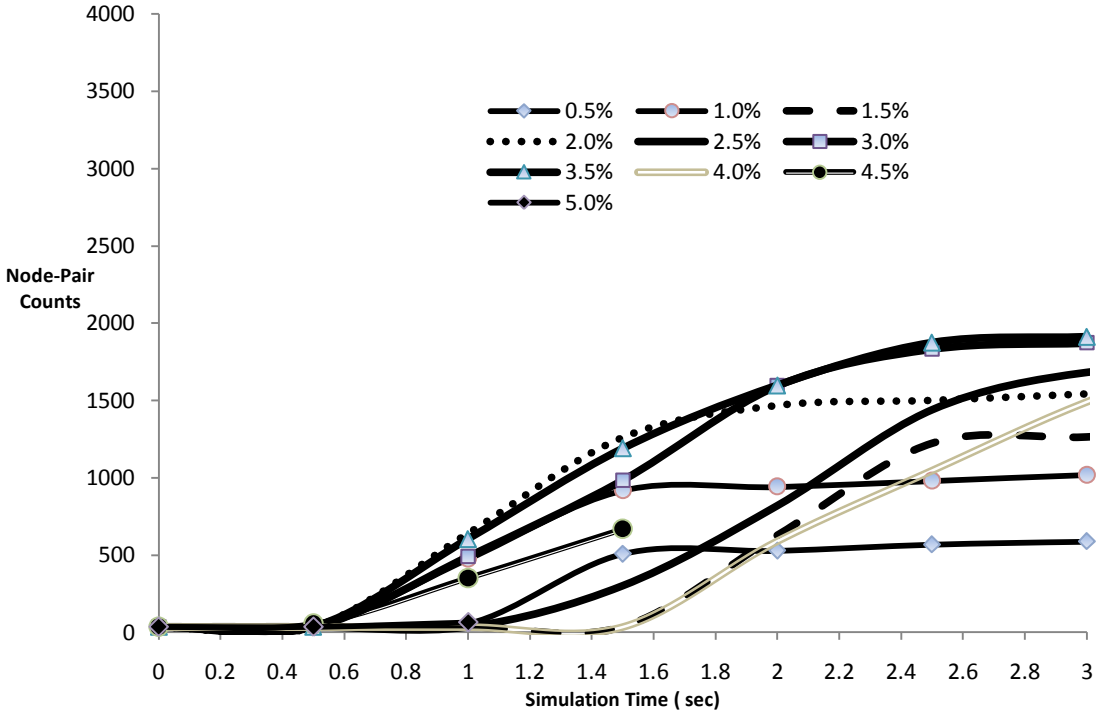
Attack Detection and Node-Pair Type 1-1 Counts Results for First 300 Seconds by Run Type

These Figures represent the change in counts of node-pair type 1-1 from the pre-attack state. This data was used for attack detection. It depicts the first 300 seconds of each simulation as discussed in Chapter VI.

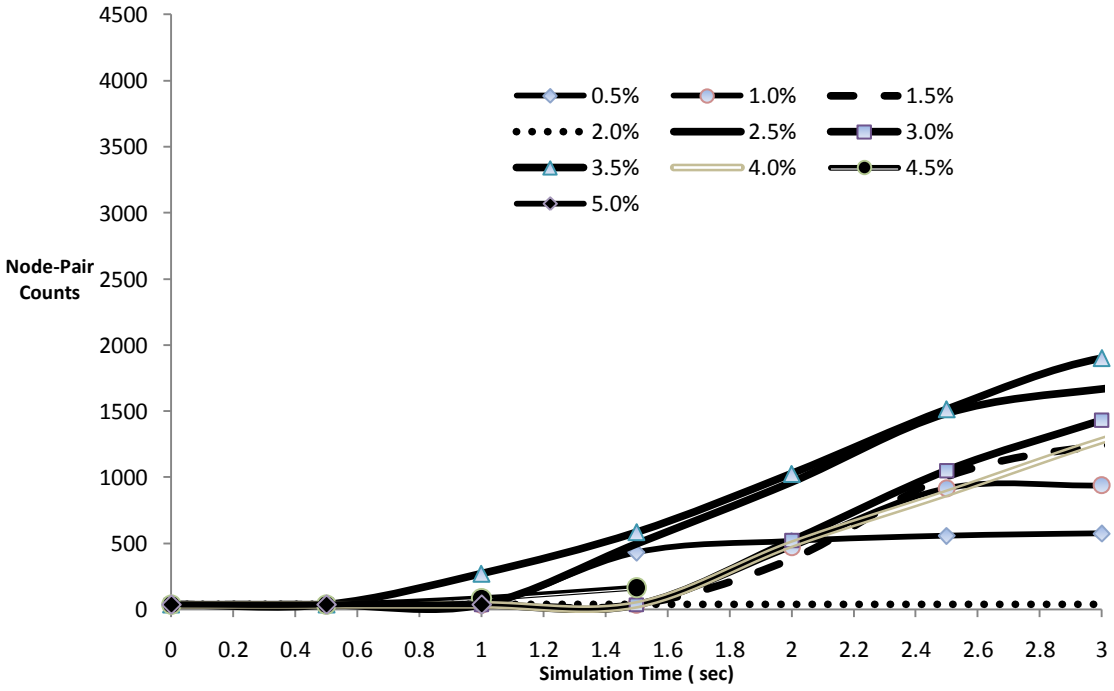
Run Type 1 – Attack Detection – Node-Pair Type 1-1 Counts, First 300 seconds



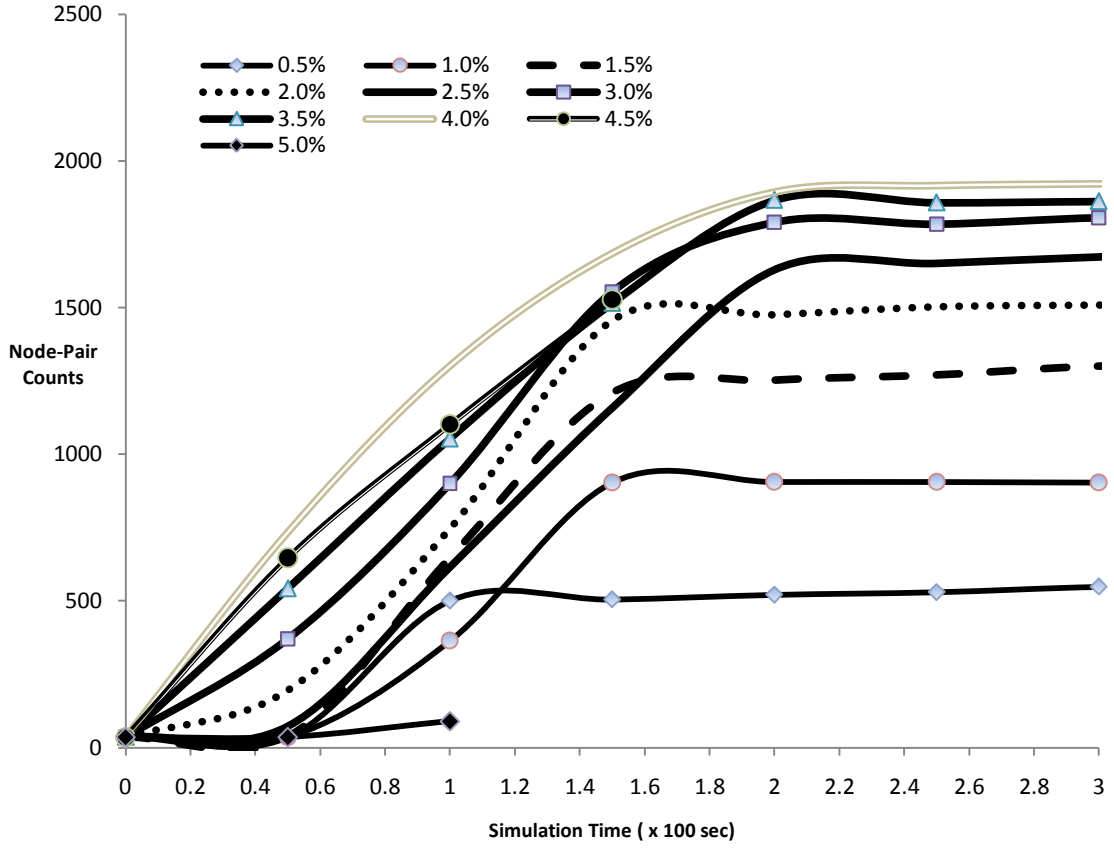
Run Type 2 – Attack Detection – Node-Pair Type 1-1 Counts, First 300 seconds



Run Type 3 – Attack Detection – Node-Pair Type 1-1 Counts, First 300 seconds



Run Type 4 – Attack Detection – Node-Pair Type 1-1 Counts, First 300 seconds



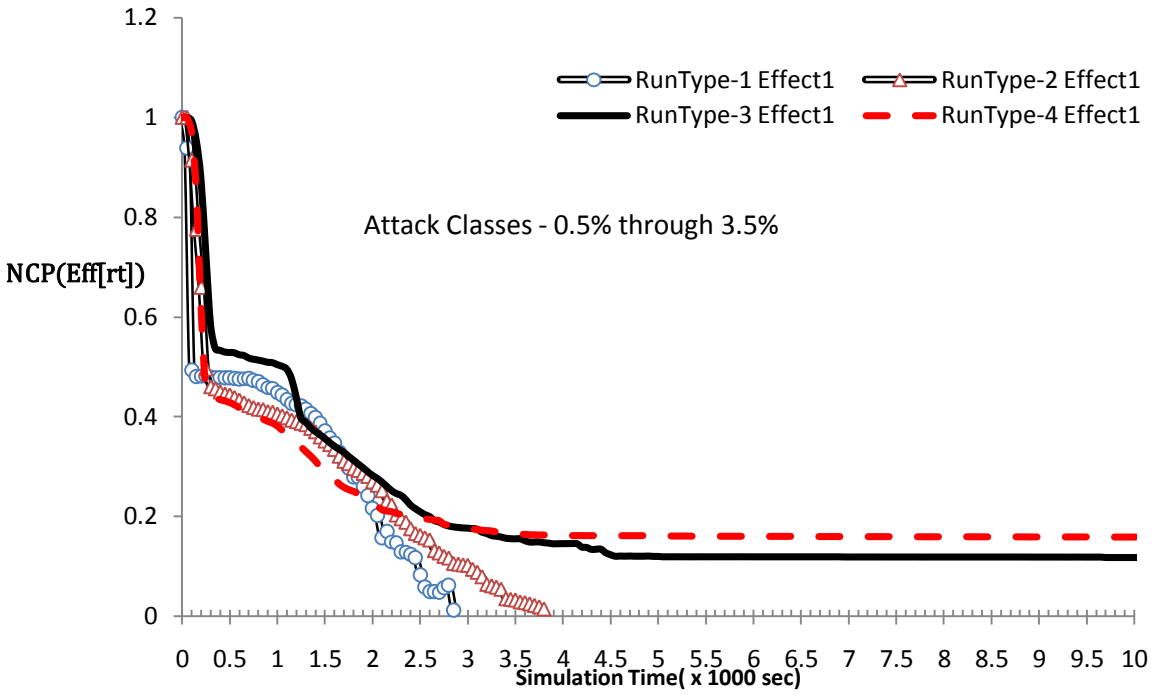
Appendix T

Network Connectivity Parameter Efficiency Results by Attack Effect

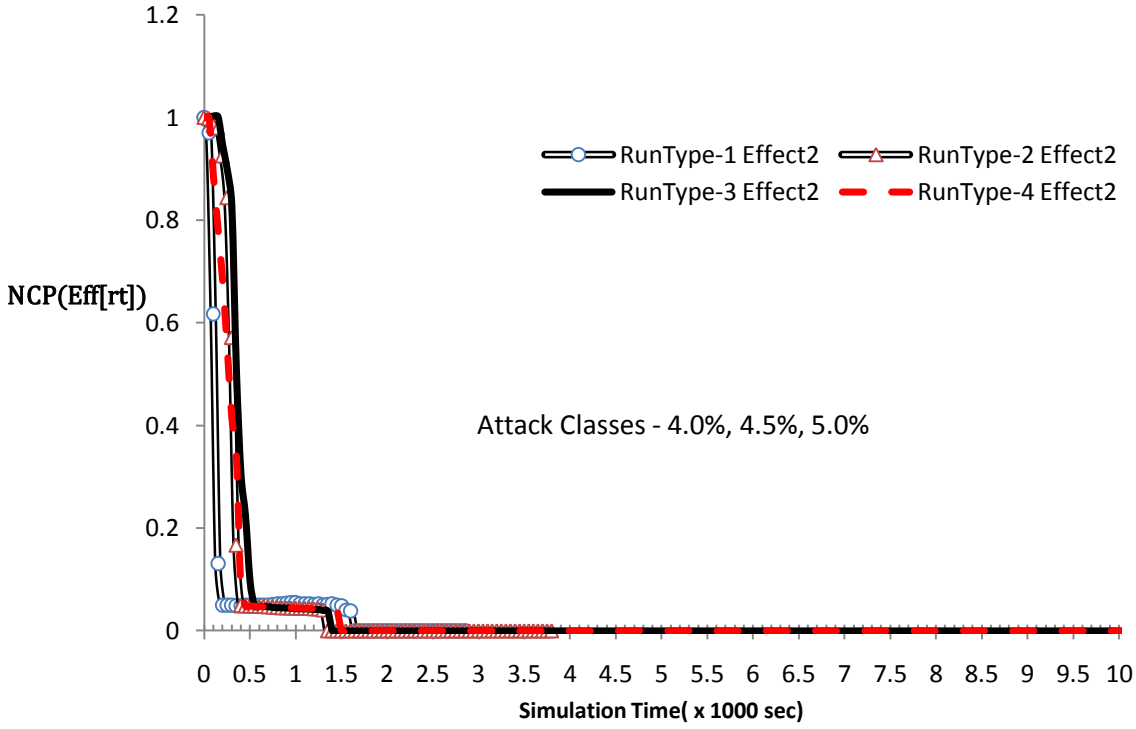
NCP Efficiency is described in Chapter VI. An efficiency term for NCP was developed for this research. The relative stability during the simulation was represented as:

$NCP(Eff[rt])_t = \frac{\sum_{ac\ 0}^{ac\ n} NCP_{\ t}}{\sum_{ac\ 0}^{ac\ n} NCP_{t=0}}$. Where the attack classes, ac_n represents n attack classes as previously defined and $rt = \{1,2,3,4\}$. For example, the NCP efficiency of run type 1 at time t is represented as the sum of all network connectivity parameter values at time t for all attack classes from ac_0 to ac_n with the same run type divided by the sum of the network connectivity parameter values at the pre-attack time for each attack class with the same run type. In the remainder of this chapter, the research will consider to groups of attack classes based on the attack effect. The efficiencies will be presented for attack classes 0.5% to 3.5% (n = 7 for attack effect 1) and 4.0% through 5.0% (n = 3, for attack effect 2).

NCP Efficiency for Attack Effect 1 – Attack classes 0.5% to 3.5%



NCP Efficiency for Attack Effect 2 – Attack classes 4.0%, 4.5% and 5.0%



Appendix U

Information Transfer and Node-Pair Type Efficiency Results by Attack Effect

The attack class influence on the simulations discussed previously led this research to segregate the attack results into 2 classifications. Simulations executed against attack classes 0.5%, 1.0%, 1.5%, 2.0%, 2.5%, 3.0% and 3.5% were classified as attack effect 1. The remaining 3 most severe attack classes, 4.0%, 4.5% and 5.0%, were classified as attack effect 2.

As shown in Chapter V, each attack effect 1 simulation: 1) experienced an equilibrium point for run type 3 and 4 simulations, 2) achieved a distinct local minimum influenced by attack severity for all run types, 3) stabilized at these local minima at a relatively constant value for significant period of time before the terminal conditions were achieved. Attack effect 2 simulations experienced relatively rapid network degradation without exhibiting any of the trends observed in the attack effect 1 simulations. The local minima, terminal conditions and equilibrium point were previously defined.

This section will discuss the implications on network stability of protecting a node based on its membership in a specific node-pair type, type 1-2. As previously discussed, if one of the two nodes in a node-pair is protected then it cannot be removed from the network during the attack simulations. The preservation of these node-pairs will selectively protect the overall network stability from cascading node failures.

Node-pair and Information Transfer Efficiency

This research employed a relative measure to study changes in the network's connectivity during the simulated denial-of-service attacks. The relative measure to study the network stability was denoted as information transfer efficiency. The relative measure used for analysis of node-pair type behaviors was denoted node-pair efficiency. Node-pair and information transfer efficiency considers post attack stability relative to pre-attack stability. Each was computed for attack effect 1 and attack effect 2 simulations. The relative stability for information transfer for each simulation was represented as:

$$I(Eff[rt])_t = \frac{\sum_{ac\ 0}^{ac\ n} I_{\ t}}{\sum_{ac\ 0}^{ac\ n} I_{\ t=0}}$$

The relative node-pair counts for each simulation were represented as:

$$NP(Eff[rt])_t = \frac{\sum_{ac\ 0}^{ac\ 9} Number\ \ of\ node-pairs_{\ t}}{\sum_{ac\ 0}^{ac\ 9} Number\ \ of\ node-pairs_{\ t=0}}$$

Where I is the information transfer and there exists n attack classes, ac_n; $rt = \{1,2,3,4\}$ represents the 4 run types. For attack effect 1, $n = 7$ and for attack effect 2, $n = 3$. Network connectivity parameter (NCP) efficiency results can be found in the appendix.

The efficiency terms represent the proportion of either NP or I remaining after the attack at time t. Efficiencies below 1 indicate a decrease in information efficiency (node-pair count) for the combined attack class data and efficiencies greater than 1 indicate an increase in information transfer (node-pair count) efficiency for the combined attack class data. The pre-attack efficiencies at $t = 0$ were equal to one.

Attack Effects – Network Protection

Each run type shown in Figure U.1 represents the cumulative behaviors of 10 simulation runs, 7 for attack effect 1 and 3 for attack effect 2. Attack effect 1 simulation executions are depicted in Figures U.1a and b. As shown in Figures U.1a and b, the network degradation for run types 3 and 4 were greatly reduced and the network stabilized at an equilibrium point. There was no equilibrium for run types 1 and 2. The data for run types 3 and 4 in Figures U.1a and b indicates that the network was protected from further degradation. The network stability stabilized at approximately 50% for run type 4 as opposed to complete degradation observed in run types 1 and 2. It shows the protection affect of protection strategies 2 and 3 for run types 3 and 4 respectively. The counts shown in Figure U.1b were consistent with the changes in information transfer stability shown in Figure U.1a. Figure U.1b indicates that the network stabilized at the same time as the node-pair type 1-2 counts stabilized. The efficiency analysis for attack effect 2 is presented in Figures U.1c and d. There was no protection effect for the most severe attacks.

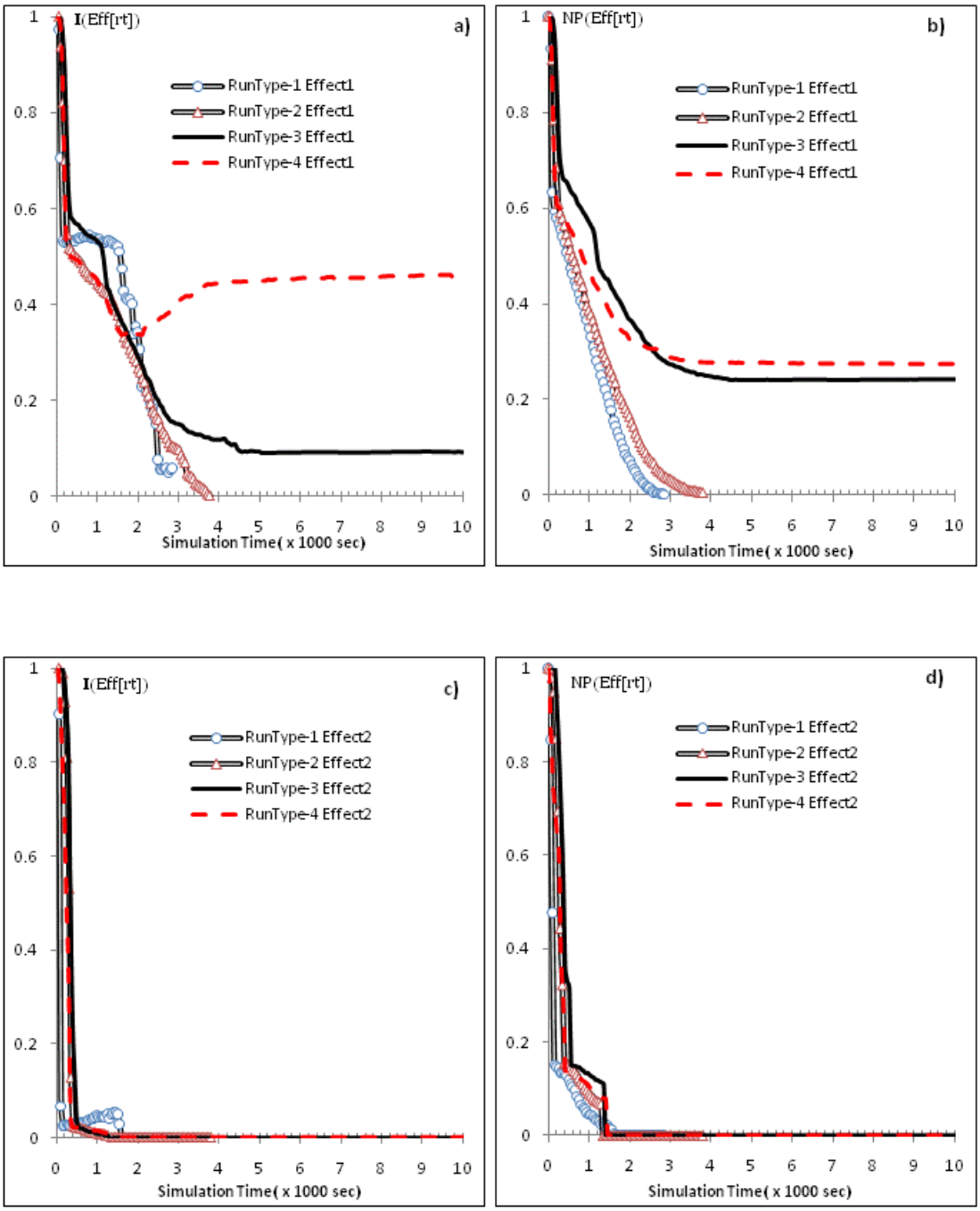


Figure U.1. Average information transfer and node-pair type efficiency by Attack effect 1 and 2 and run type, 10 simulations for each run type, a and b) attack effect 1, information transfer and node-pair type 1-2 counts respectively, c and d) attack effect 2, information transfer and node-pair type 1-2 counts respectively.