

---

# **BGP BASED SOLUTION FOR INTERNATIONAL ISP BLOCKING**

by

AHMAD SALAM ABDULLATIF ALREFAI

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the Requirements for the degree

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

Dhahran, Saudi Arabia

December, 2009

# ***BGP BASED SOLUTION FOR INTERNATIONAL ISP BLOCKING***

BY

Ahmad Salam Abdullatif Alrefai

A Thesis Presented to the  
DEANSHIP OF GRADUATE STUDIES

**KING FAHD UNIVERSITY OF PETROLEUM & MINERALS**

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the  
Requirements for the Degree of

**MASTER OF SCIENCE**

In

COMPUTER ENGINEERING

December, 2009

# KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

## DEANSHIP OF GRADUATE STUDIES

This Thesis, written by AHMAD SALAM ABDULLATIF ALREFAI under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE IN COMPUTER ENGINEERING.


### Thesis Committee



Dr. Ashraf S. Hasan Mahmoud (Advisor)



Dr. Marwan Abu-Amara (Co-advisor)



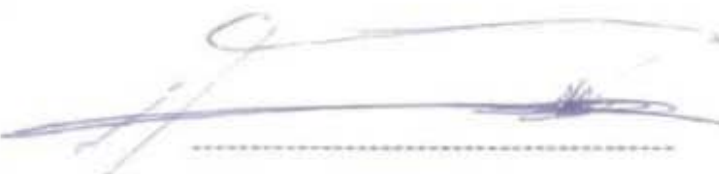
Dr. Mohammed Sqalli (Member)



Dr. Farag Azzedin (Member)



Dr. Alaaeldin Amin (Member)



Dr. Basem Al-Madani  
(Department Chairman)



Dr. Salam Zummo  
(Dean of Graduate Studies)

22/6/10

Date

*Dedication*

*To my*

*Mother, Father, Sisters, & Brother...*



## *ACKNOWLEDGEMENT*

First and foremost I would like to thank Allah for giving me the power, enthusiasm, and knowledge to complete this work.

I also would like to thank the University King Fahd University of Petroleum and Minerals where I have spent 9 years of my life.

I wish to express my appreciation to Dr. Ashraf S. Hasan Mahmoud who served as my major advisor. His excellent feedback and comments were of major advantage to me to complete this work. His personality and way of thinking was of major help to me to crystallize this work. Also, I would like to thank Dr. Marwan Abu-Amara my co-advisor for his constructive feedback until this work saw the light. In addition, I would like to thank Dr. Mohammed Sqalli who is an advising committee member whose useful comments and support helped a lot in shaping the thesis. Moreover, my thanks and appreciation should go to Dr. Farag Azzedin, who served as a committee member, for his suggestions and comments which helped a lot in making this work real. All my thanks and appreciation are also due to Dr. Alaaeldin Amin from whom I learnt how to be a good engineer and how to be a good person, and without his encouragement this work would not have been completed. I would like also to thank the students who were working in the same project for the interesting discussion we had about the project.

I would like also to thank Dr. Bruno Quoitin for the useful email discussions on the subject matter, and for the very nice personality he has.

I would like also to express my appreciation to every faculty who taught me at King Fahd University of Petroleum and Minerals and I benefited from their experience. These includes Dr. Guldad Khatak at the Physics department, Dr. Salaheddin Adam at the Information and Computer Science department, Dr. Aiman Almaleh at the Computer Engineering department, in addition to Dr. Muhammad Alawi Albar and Dr. Abdulla Al-Daffa in mathematics department.

I would like also to express my deep thanks to the current Computer Engineering department chairman Dr. Basem Almadani, and to the previous chairman of Computer Engineering department Dr. Adnan Gutub, and the chairman of the Information and Computer Science department Dr. Kanaan Faisal for their advices and support.

I would like also to thank my mother for her enthusiasm for my work and for her continuous advices to me from the time I was child until today. Without her support and prayers, the thesis could never have been achieved. My thanks are also due to my father for support and advise which shaped my personality to be the person who seeks the truth and to be a good researcher.

## TABLE OF CONTENT

Acknowledgement .....	iv
Table of Content .....	vi
List of Tables .....	x
Table of Figures.....	xi
Code Snippet.....	xvi
Thesis Abstract.....	xviii
ملخص الرسالة .....	xix
<b>CHAPTER 1 THE PROBLEM.....</b>	<b>1</b>
1.1 INTRODUCTION.....	1
1.2 MOTIVATION .....	2
1.3 PROBLEM DESCRIPTION.....	2
1.4 SCOPE.....	7
1.5 ASSUMPTIONS .....	8
1.6 LIMITATIONS .....	8
1.7 THESIS ORGANIZATION.....	8
1.8 SUMMARY .....	10
<b>CHAPTER 2 BACKGROUND .....</b>	<b>11</b>
2.1 INTERNET .....	11
2.2 BORDER GATEWAY PROTOCOL .....	16
<b>CHAPTER 3 LITERATURE SURVEY .....</b>	<b>21</b>
3.1 INTRODUCTION.....	21
3.2 BGP RFCs .....	22
3.3 BGP MULTIHOMING .....	22

3.4	BGP THREATS & ATTACKS .....	23
3.5	BGP SECURITY .....	25
3.6	MOAS CONFLICT .....	27
3.7	PREFIX HIJACKING .....	29
3.8	BGP MISCONFIGURATION .....	34
3.9	OVERLAY PROTOCOLS .....	35
3.10	BGP TRAFFIC ENGINEERING .....	37
3.11	BGP SIMULATION TOOLS .....	38
3.12	BGP BEHAVIOR UNDER STRESS .....	39
3.13	BGP CONVERGENCE ANALYSIS .....	41
 <b>CHAPTER 4 QUALITATIVE ANALYSIS OF METHODS FOR CIRCUMVENTING MALICIOUS ISP BLOCKING</b>		
<b>.....</b>		<b>43</b>
4.1	INTRODUCTION.....	43
4.2	METHODS TO COMBAT INTERNET SERVICE DENIAL .....	44
4.2.1	<i>BGP Tuning</i> .....	44
4.2.2	<i>Virtual Peering</i> .....	53
4.2.3	<i>Virtual Transit (Hijack the Hijacker)</i> .....	56
4.3	ANALYSIS AND DISCUSSION .....	59
4.4	SUMMARY .....	64
 <b>CHAPTER 5 DESIGN, IMPLEMENTATION &amp; VALIDATION OF SOLUTION USING BGP TUNING BASED</b>		
<b>APPROACH.....</b>		<b>65</b>
5.1	INTRODUCTION.....	65
5.2	INITIAL SETUP.....	66
5.2.1	<i>Devices Used</i> .....	67
5.2.2	<i>Mapping Devices Usage to our Problem</i> .....	69

5.3	CONTROLLING TRAFFIC USING SUPPORTED OPNET FEATURES.....	70
5.3.1	<i>Normal Simulation Run</i> .....	71
5.3.2	<i>Control of Outgoing Traffic</i> .....	76
5.3.3	<i>Controlling Incoming Traffic</i> .....	79
5.3.4	<i>Use of Community</i> .....	84
5.4	MODIFICATION OF OPNET IMPLEMENTATION.....	87
5.4.1	<i>OPNET Process Model</i> .....	89
5.4.2	<i>Building a Malicious Router</i> .....	89
5.5	BUILDING COUNTERMEASURES AGAINST MALICIOUS ACTS .....	99
5.5.1	<i>Summary of BGP in OPNET</i> .....	99
5.5.2	<i>BGP Modifications for Scheduled Reconfigurations</i> .....	101
5.6	SUMMARY .....	142
 <b>CHAPTER 6 PERFORMANCE EVALUATION OF BGP TUNING TECHNIQUES TO CIRCUMVENT MALICIOUS ACT.....</b>		<b>143</b>
6.1	INTRODUCTION.....	143
6.2	COMPARISON .....	148
6.2.1	<i>Percentage of traffic drop</i> .....	148
6.2.2	<i>Convergence Time</i> .....	151
6.2.3	<i>Throughput</i> .....	156
6.3	CASES WHEN THE SIMULATION FAILS.....	193
6.4	RECOMMENDATIONS .....	195
6.5	SUMMARY .....	196
 <b>CHAPTER 7 CONCLUSION &amp; FUTURE WORK.....</b>		<b>197</b>
7.1	CONCLUSION.....	197
7.2	FUTURE WORK.....	198

<b>APPENDIX A</b>	<b>APPLICATION CONFIGURATION .....</b>	<b>199</b>
A.1	TCP CONFIGURATION .....	199
A.2	HTTP CONFIGURATION.....	201
A.3	FTP CONFIGURATION .....	203
A.4	VOIP CONFIGURATION .....	204
<b>APPENDIX B</b>	<b>BASELINE THROUGHPUT AND APPLICATION TRAFFIC .....</b>	<b>207</b>
B.1	BASELINE THROUGHPUT .....	207
B.2	BASELINE APPLICATION TRAFFIC.....	215
<b>APPENDIX C</b>	<b>CODE CHANGE .....</b>	<b>222</b>
C.1	CHANGES IN BGP MODULE .....	222
C.1.1	<i>Changes in bgp Process .....</i>	<i>222</i>
C.1.2	<i>Modification in bgp_conn Process.....</i>	<i>238</i>
C.1.3	<i>Shortening .....</i>	<i>240</i>
C.1.4	<i>More Specific Prefixes.....</i>	<i>244</i>
C.2	MODIFICATION IN IP PROTOCOL.....	248
C.2.1	<i>IP Dispatch Process.....</i>	<i>248</i>
<b>APPENDIX D</b>	<b>MANUAL TO RUN THE WORK.....</b>	<b>254</b>
D.1	PROCEDURE .....	254
<b>REFERENCES</b>	<b>.....</b>	<b>257</b>
Vita.....		261



## *LIST OF TABLES*

TABLE 4-1 COMPARISON BETWEEN METHODS. ....	63
TABLE 6-1 LIST OF OUTPUT FIGURES, BRIEF DEFINITION AND WHY THEY ARE PICKED .....	147

## TABLE OF FIGURES

FIGURE 1-1 MALICIOUS IISP BLOCKING. ....	4
FIGURE 1-2 IDENTITY HIDING TECHNIQUES. ....	6
FIGURE 1-3 TRAFFIC ENGINEERING BASED TECHNIQUES. ....	7
FIGURE 2-1 OVERVIEW OF THE INTERNET. ....	12
FIGURE 2-2 SAMPLE OF ADDRESS SPACE DELEGATION [7] ....	14
FIGURE 2-3 BORDER GATEWAY PROTOCOL FINITE STATE MACHINE ....	19
FIGURE 4-1 CONTROL OF OUTGOING TRAFFIC USING BGP TUNING ....	45
FIGURE 4-2 AS PATH SHORTENING ....	48
FIGURE 4-3 MORE SPECIFIC PREFIXES.....	50
FIGURE 4-4 CONTROL THE TRAFFIC USING COMMUNITIES .....	52
FIGURE 4-5 VIRTUAL PEERING BASED SOLUTION .....	54
FIGURE 4-6 VIRTUAL TRANSIT .....	57
FIGURE 5-1 BASELINE CONFIGURATION OF VALIDATION .....	67
FIGURE 5-2 INCOMING AND OUTGOING TRAFFIC IN NORMAL MODE .....	72
FIGURE 5-3 IP FORWARDING TABLE FOR ROUTER 2. ....	73
FIGURE 5-4 IP FORWARDING TABLE OF ROUTER 5.....	74
FIGURE 5-5 CONVERGENCE ACTIVITY AND DURATION OF NORMAL MODE .....	75
FIGURE 5-6 HIGHER LOCAL PREFERENCE TO ROUTES ADVERTISED BY ROUTER 4. ....	77
FIGURE 5-7 BGP ROUTING TABLE OF ROUTER 2 IN NORMAL LOCAL-PREF EXPERIMENT.....	78
FIGURE 5-8 CONVERGENCE ACTIVITY AND DURATION OF NORMAL LOCAL-PREF EXPERIMENT .....	79
FIGURE 5-9 INCOMING TRAFFIC TO ROUTER 2 USING PREPENDING .....	81
FIGURE 5-10 BGP ROUTING TABLE OF ROUTER3 IN PREPEND EXPERIMENT .....	82
FIGURE 5-11 BGP TABLE OF ROUTER 5 IN PREPEND EXPERIMENT.....	82
FIGURE 5-12 CONVERGENCE ACTIVITY AND DURATION FOR NORMAL PREPENDING EXPERIMENT .....	83

FIGURE 5-13 INCOMING TRAFFIC TO ROUTER 2 USING COMMUNITY .....	85
FIGURE 5-14 BGP ROUTING TABLE OF ROUTER 5 IN THE COMMUNITY EXPERIMENT. ....	86
FIGURE 5-15 CONVERGENCE ACTIVITY AND DURATION OF COMMUNITY EXPERIMENT .....	87
FIGURE 5-16 IP_DISPATCH PROCESS MODEL.....	90
FIGURE 5-17 IP_RTE_CENTRAL_CPU PROCESS MODEL. ....	91
FIGURE 5-18 INTERFACE TO CONFIGURE MALICIOUS ROUTER. ....	93
FIGURE 5-19 ACTIVITY DIAGRAM OF BLACKHOLING METHOD .....	95
FIGURE 5-20 THROUGHPUT IN A MALICIOUS CONFIGURATION EXPERIMENT.....	97
FIGURE 5-21 BGP TABLE OF ROUTER 5 IN MALICIOUS EXPERIMENT.....	98
FIGURE 5-22 BGP PROCESS. ....	100
FIGURE 5-23 BGP CONN PROCESS .....	101
FIGURE 5-24 RECONFIGURATION OVERVIEW. ....	102
FIGURE 5-25 MODIFIED BGP PROCESS MODEL .....	103
FIGURE 5-26 HANDLING RECONFIGUREIN A CHILD PROCESS .....	106
FIGURE 5-27 HANDLING OF RECONFIGIN PARENT.....	107
FIGURE 5-28 SPECIFICATION OF TIME WHEN APPLYING ROUTE MAP. ....	108
FIGURE 5-29 THROUGHPUT TRAFFIC AFTER APPLYING LOCALPREF IN THE PRESENCE OF A MALICIOUS ROUTER.....	109
FIGURE 5-30 IP FORWARDING TABLE OF ROUTER 2 IN LOCAL-PREF AND MALICIOUS EXPERIMENT.....	110
FIGURE 5-31 IP FORWARDING TABLE OF ROUTER 2 AT TIME 71 IN THE LOCAL-PREF AND MALICIOUS EXPERIMENT .....	111
FIGURE 5-32 BGP ROUTING TABLE OF ROUTER 2 AT TIME 71 FOR LOCAL-PREF AND MALICIOUS EXPERIMENT.....	112
FIGURE 5-33 BGP ROUTING TABLE OF ROUTER 2 IN LOCAL-PREF AND MALICIOUS EXPERIMENT. ....	113
FIGURE 5-34 MESSAGE EXCHANGE FOR THE MALICIOUS AND LOCAL-PREF .....	114
FIGURE 5-35 CONVERGENCE ACTIVITY OF THE LOCAL PREF AND MALICIOUS EXPERIMENT .....	116
FIGURE 5-36 RECONFIGOUT ACTIVITY DIAGRAM.....	118
FIGURE 5-37 THROUGHPUT BETWEEN ROUTER 2 AND ROUTER3, ROUTER 4 AND PACKET DROP OF ROUTER 3 .....	119
FIGURE 5-38 BGP ROUTING TABLE OF ROUTER 5 IN MALICIOUS AND COMMUNITY EXPERIMENT .....	120

FIGURE 5-39 BGP ROUTING TABLE OF ROUTER 5 MALICIOUS & COMMUNITY EXPERIMENT AT TIME 71 BEFORE SENDING UPDATE MESSAGE WITH COMMUNITY NUMBER.....	121
FIGURE 5-40 MESSAGE PASSING BETWEEN ROUTERS IN COMMUNITY AND MALICIOUS EXPERIMENT.....	122
FIGURE 5-41 CONVERGENCE ACTIVITY IN COMMUNITY AND MALICIOUS .....	124
FIGURE 5-42 SHORTENING CONFIGURATION.....	126
FIGURE 5-43 THROUGHPUT BETWEEN ROUTER 2 AND ROUTERS 3 & 4 AND DROPPED TRAFFIC OF ROUTER 3 .....	128
FIGURE 5-44 BGP ROUTING TABLE OF ROUTER 5 IN SHORTENING, LOCALPREF, MALICIOUS EXPERIMENT .....	129
FIGURE 5-45 BGP ROUTING TABLE OF ROUTER 5 IN SHORTENING, LOCAL-PREF, MALICIOUS EXPERIMENT .....	130
FIGURE 5-46 MESSAGE PASSING IN SHORTENING, LOCALPREF, AND MALICIOUS EXPERIMENT .....	131
FIGURE 5-47 CONVERGENCE ACTIVITY AND DURATION FOR SHORTENING EXPERIMENT .....	133
FIGURE 5-48 MORE SPECIFIC PREFIX INTERFACE.....	135
FIGURE 5-49 MORE SPECIFIC PREFIX IMPLEMENTATION .....	136
FIGURE 5-50 INCOMING AND OUTGOING TRAFFIC OF ROUTER 2 IN MORE SPECIFIC, LOCAL PREFERENCE, MALICIOUS EXPERIMENT .....	138
FIGURE 5-51 BGP ROUTING TABLE OF ROUTER 5 OF MORE SPECIFIC, LOCAL PREF, MALICIOUS EXPERIMENT .....	139
FIGURE 5-52 MESSAGE PASSING IN MORE SPECIFIC, LOCAL PREF, MALICIOUS EXPERIMENT .....	140
FIGURE 5-53 CONVERGENCE ACTIVITY AND DURATION OF MORE SPECIFIC, LOCAL PREF, AND MALICIOUS EXPERIMENT .....	141
FIGURE 6-1 EVALUATION NETWORK SETUP .....	144
FIGURE 6-2 NETWORK SHOWING THE LINKS THAT WILL BE LOADED WITH TRAFFIC .....	145
FIGURE 6-3 PACKET DROP PERCENTAGES.....	149
FIGURE 6-4 BGP CONVERGENCE TIME FOR 0.1 SECONDS DELAY OF INTERNET. ....	153
FIGURE 6-5 BGP CONVERGENCE TIME FOR 5 SECOND DELAY .....	155
FIGURE 6-6 OUTGOING THROUGHPUT FROM ROUTER 2 TO ROUTER 3 FOR HTTP. ....	157
FIGURE 6-7 THROUGHPUT FROM ROUTER 2 TO ROUTER 3 FOR FTP APPLICATION .....	158
FIGURE 6-8 THROUGHPUT FROM ROUTER 2 TO ROUTER 3 FOR VOIP APPLICATION .....	159
FIGURE 6-9 INCOMING THROUGHPUT TO ROUTER 2 FROM ROUTER 3. ....	161

FIGURE 6-10 THROUGHPUT FROM ROUTER 3 TO ROUTER 2 IN FTP APPLICATION .....	163
FIGURE 6-11 THROUGHPUT FROM ROUTER 3 TO ROUTER 2 FOR VOIP APPLICATION .....	165
FIGURE 6-12 OUTGOING TRAFFIC FROM ROUTER 2 TO ROUTER 4 FOR HTTP APPLICATION. ....	167
FIGURE 6-13 THROUGHPUT FROM ROUTER 2 TO ROUTER 4 FOR FTP APPLICATION. ....	170
FIGURE 6-14 THROUGHPUT FROM ROUTER 2 TO ROUTER 4 IN VOIP APPLICATION .....	172
FIGURE 6-15 INCOMING TRAFFIC FROM ROUTER 4 TO ROUTER 2 FOR HTTP APPLICATION.....	173
FIGURE 6-16 THROUGHPUT FROM ROUTER 4 TO ROUTER 2 IN FTP APPLICATION. ....	175
FIGURE 6-17 THROUGHPUT FROM ROUTER 4 TO ROUTER 2 IN VOIP APPLICATION. ....	177
FIGURE 6-18 HTTP PACKET SENT .....	179
FIGURE 6-19 HTTP PACKET RECEIVED .....	181
FIGURE 6-20 FTP PACKETS SENT FROM LAN_EAST. ....	183
FIGURE 6-21 FTP PACKET RECEIVED TO LAN EAST. ....	185
FIGURE 6-22 VOIP PACKET SENT FROM LAN_EAST .....	187
FIGURE 6-23 VOIP PACKET RECEIVED IN LAN_EAST .....	189
FIGURE 6-24 PAGE RESPONSE TIME FOR HTTP CLIENT .....	191
FIGURE 6-25 FTP DOWNLOAD RESPONSE TIME. ....	192
FIGURE 6-26 PACKET DROP IN VOIP, COMMUNITY SOLUTION, EXPONENTIAL WITH 5 SECOND DELAY, 80% LINK LOAD .....	194
FIGURE A-1 TCP CONFIGURATION. ....	200
FIGURE A-2 HTTP CONFIGURATION.....	201
FIGURE A-3 HTTP PAGE PROPERTIES. ....	202
FIGURE A-4 SIZE OF IMAGE.....	202
FIGURE A-5 HTTP SERVER SELECTION. ....	203
FIGURE A-6 FTP CONFIGURATION. ....	204
FIGURE A-7 VOIP CONFIGURATION. ....	205
FIGURE A-8 SILENCE LENGTH CONFIGURATION. ....	205

FIGURE A-9 TALK SPURT LENGTH.....	206
FIGURE B-1 BASELINE HTTP THROUGHPUT FROM ROUTER 2 TO ROUTER 3. ....	208
FIGURE B-2 BASELINE HTTP THROUGHPUT FROM ROUTER 3 TO ROUTER 2.....	209
FIGURE B-3 BASELINE FTP THROUGHPUT FROM ROUTER 2 TO ROUTER 3.....	210
FIGURE B-4 BASELINE FTP THROUGHPUT FROM ROUTER 3 TO ROUTER 2.....	211
FIGURE B-5 BASELINE THROUGHPUT FOR FTP APPLICATION, INTER-REQUEST IS 60. ....	212
FIGURE B-6 BASELINE FTP THROUGHPUT FROM ROUTER 3 TO ROUTER 2 WHEN THE INTER-REQUEST IS 60 .....	213
FIGURE B-7 BASELINE VOIP THROUGHPUT FROM ROUTER 2 TO ROUTER 3.....	214
FIGURE B-8 BASELINE VOIP THROUGHPUT FROM ROUTER 3 TO ROUTER 2.....	215
FIGURE B-9 BASELINE HTTP TRAFFIC SENT OF LAN_EAST.....	216
FIGURE B-10 BASELINE HTTP TRAFFIC RECEIVED IN LAN_EAST.....	217
FIGURE B-11 BASELINE FTP TRAFFIC SENT IN LAN_EAST. ....	218
FIGURE B-12 BASELINE FTP TRAFFIC RECEIVED IN LAN_EAST. ....	219
FIGURE B-13 BASELINE VOIP TRAFFIC SENT FROM ROUTER 2 TO ROUTER 3.....	220
FIGURE B-14 BASELINE VOIP TRAFFIC RECEIVED IN LAN_EAST. ....	221
FIGURE C-1 LOCATION OF MODIFICATIONS IN BGP PROCESS .....	223
FIGURE C-2 SCHEDULE RECONFIGURATION STATE VARIABLE ADDITION .....	224
FIGURE D-1 OPNET INSTALLATION FOLDER .....	254
FIGURE D-2 MODIFIED FOLDERS.....	255
FIGURE D-3 OP_MODELS FOLDERS .....	255



## CODE SNIPPET

CODE SNIPPET C-1 DEFINITIONS OF CONDITIONS IN HEADER BLOCK .....	224
CODE SNIPPET C-2 DEFINITION OF TIMED_POLICY IN HEADER BLOCK .....	225
CODE SNIPPET C-3 DEFINITION OF MODIFIED POLICY READ METHOD .....	225
CODE SNIPPET C-4 MODIFICATION OF DEFINITION OF READING POLICY FUNCTION .....	226
CODE SNIPPET C-5 LOCAL VARIABLES ADDED TO READ POLICY FUNCTION .....	226
CODE SNIPPET C-6 ALLOCATING MEMORY TO TIMED_POLICY VARIABLE .....	227
CODE SNIPPET C-7 READING THE TIME .....	227
CODE SNIPPET C-8 STORING TIMED POLICY IN SCHEDULED RECONFIGURATION .....	228
CODE SNIPPET C-9 MODIFIED USE OF READING POLICY FUNCTION IN BGP_NBR_ADDR_FAMILY_PARAMS_READ FUNCTION ...	229
CODE SNIPPET C-10 CREATION OF RECONFIGURATION LIST DONE IN BGP_SV_INIT .....	229
CODE SNIPPET C-11 RECONFIGURE STATE CODE .....	230
CODE SNIPPET C-12 RECONFIGUREIN CODE .....	231
CODE SNIPPET C-13 CONTINUE RECONFIGUREIN CODE .....	232
CODE SNIPPET C-14 CONTINUE RECONFIGUREIN CODE .....	233
CODE SNIPPET C-15 RECONFIGUREOUT .....	234
CODE SNIPPET C-16 CONTINUE RECONFIGUREOUT CODE .....	235
CODE SNIPPET C-17 CONTINUE CONFIGUREOUT CODE .....	236
CODE SNIPPET C-18 CONTINUE CONFIGUREOUT CODE .....	237
CODE SNIPPET C-19 ESTABLISHED STATE MODIFICATION IN BGP_CONN PROCESS .....	238
CODE SNIPPET C-20 DEFINITION OF BGP_CONN_APPLY_MAP_RIB_IN .....	239
CODE SNIPPET C-21 CONTINUE DEFINITION OF BGP_CONN_APPLY_MAP_RIB_IN .....	240
CODE SNIPPET C-22 MODIFICATION OF PREPEND FUNCTION TO CALL THE SHORTEN FUNCTION .....	241
CODE SNIPPET C-23 DEFINITION OF BGP_SUPPORT_AS_PATH_REMOVE_FIRST OR SHORTEN FUNCTION .....	242
CODE SNIPPET C-24 CONTINUE DEFINITION OF SHORTEN FUNCTION .....	243

CODE SNIPPET C-25 MODIFYING BGP_NBR_ADDR_FAMILY_PARAMS_READ FUNCTION TO CALL	
BGP_NEIGHBOR_MORE_SPECIFIC_PREFIX_READ .....	244
CODE SNIPPET C-26 DEFINITION OF BGP_NEIGHBOR_MORE_SPECIFIC_PREFIX_READ FUNCTION. ....	245
CODE SNIPPET C-27 CONTINUE OF DEFINITION OF BGP_NEIGHBOR_MORE_SPECIFIC_PREFIX_READ FUNCTION. ....	246
CODE SNIPPET C-28 CONTINUE OF DEFINITION OF BGP_NEIGHBOR_MORE_SPECIFIC_PREFIX_READ FUNCTION. ....	247
CODE SNIPPET C-29 DEFINITION OF IPT_RTE_BLACKHOLE_FROM STRUCTURE. ....	248
CODE SNIPPET C-30 ADDING IPT_RTE_BLACKHOLE_FROM TO IPT_RTE_MODULE_DATA.....	249
CODE SNIPPET C-31 LOCAL VARIABLES ADDED TO IP_DISPATCH_INIT_PHASE_2.....	249
CODE SNIPPET C-32 MODIFYING IP_DISPATCH_INIT_PHASE_2 TO READ VALUES FOR MORE SPECIFIC PREFIX CONFIGURATION.	
.....	250
CODE SNIPPET C-33 ADDING LOCAL VARIABLE TO IP_RTE_CENTRAL_CPU_ARRIVAL FUNCTION. ....	251
CODE SNIPPET C-34 CALLING IP_RTE_BLACKHOLE_TRAFFIC FUNCTION FROM INSIDE IP_RTE_CENTRAL_CPU_PACKET_ARRIVAL.	
.....	251
CODE SNIPPET C-35 DEFINITION OF IP_RTE_BLACKHOLE_TRAFFIC. ....	252
CODE SNIPPET C-36 THE IMPLEMENTATION OF IP_RTE_BLACKHOLE_TRAFFIC_FUNCTION. ....	252
CODE SNIPPET C-37 THE IMPLEMENTATION OF IP_RTE_BLACKHOLE_TRAFFIC_FUNCTION. ....	253

## *THESIS ABSTRACT*

Name: Ahmad Salam Abdullatif Alrefai

Title: BGP based Solution for International ISP Blocking

Major Field: Computer Engineering

Date of Degree: December 2009

The problem that this thesis tackles is finding and evaluating a counter measure to the malicious Internet blocking of a local region by International Internet Service Providers (IISP) that blackhole traffic destined to or originated from the local region while advertising paths to destinations. The thesis only considers solutions that are based on Border Gateway Protocol. The major contributions of this thesis are three fold. The first one is the proposal of three BGP based solutions; namely BGP tuning, virtual peering, and virtual transit. While the control of outgoing traffic using BGP tuning is done by setting the value of local preference, the control of the incoming traffic can be done using AS-Path shortening, advertising more specific prefixes, or using community techniques. The second contribution is the implementation of the problem model and BGP tuning based solutions in OPNET. The third contribution is the performance evaluation of the BGP tuning solutions in terms of packet drop, convergence time, throughput, and application specific measurements.

## ملخص الرسالة

الاسم: احمد سلام بن عبد اللطيف الرفاعي

عنوان الرسالة: حل يعتمد على بروتوكول بوابة الحدود (BGP) لمشكلة الحجب من مزود الانترنت الدولي

التخصص: هندسة الحاسب الآلي

تاريخ التخرج: ديسمبر (كانون الأول)، 2009

تهدف هذه الأطروحة لإيجاد حل لمشكلة حجب الانترنت عن منطقة محلية من قبل مزود دولي خبيث للانترنت لها. يسقط هذا المزود الخبيث الحزم الكترونية المتجهة إلى أو الصادرة من تلك المنطقة المحلية مع قيامه بالاعلان عن قدرته على إيصال الحزم إلى أماكنها. تدرس الأطروحة الحلول المعتمدة على بروتوكول بوابة الحدود (BGP). وتقدم الأطروحة ثلاثة مساهمات رئيسية. المساهمة الأولى هي اقتراح ثلاثة حلول معتمدة على بروتوكول بوابة الحدود وهي توليف البروتوكول (BGP) والتناظر الافتراضي والعابر الافتراضي. باستخدام توليف البروتوكول (BGP)، من الممكن تحديد قيمة تفضيل المحلية للتحكم بحركة المرور للحزم الصادرة، وأيضاً باستخدام نفس الطريقة من الممكن استخدام طريقة تقصير الطريق، أو إعلان بادئات أكثر تحديداً، أو استخدام تقنيات المجتمع للتحكم بحركة المرور للحزم الواردة. المساهمة الثانية هي بناء نموذج للمشكلة وحلها عن طريق توليف بروتوكول بوابة الحدود (BGP) باستخدام برنامج الـ (OPNET). المساهمة الثالثة هي تقييم أداء هذه التقنيات من حيث الحزم المسقطة، وقت التقارب، والانتاجية، بالإضافة لبعض القياسات الخاصة بتطبيقات محددة.

---

# ***CHAPTER 1***

## ***THE PROBLEM***

### ***1.1 INTRODUCTION***

The Internet is one of the most important means to communicate with others, and to obtain or to provide services. The number of people using the Internet in March 2009 exceeded 1.5 billion users which constitutes about one quarter of the whole population in the world [1]. Enhancing the resiliency, availability, and security of the Internet is one of the major requirements for Internet Service Providers (ISPs) and for Internet users who benefit from its services. If the Internet is unavailable, then the user will not be able to access services during the unavailability time. If it is not resilient or not secure then, in the presence of any problem such as Internet blockage or attack, the access to the Internet will be denied and the user will not be able to get to the subscribed services.

There are several causes of Internet outage. For example, Internet outage may occur due to malicious attacks such as distributed denial of service (DDoS) attacks. Also, it might be due to software or hardware failure, or misconfiguration of routers. Link cuts can also cause Internet unavailability. International Internet Service Providers (IISPs) have the capability to deny Internet for specific regions. This can be done either at the

application level or at the routing level [2]. This problem arises because, in this case, some services of the Internet will not be available due to intentional and malicious act of IISPs. Accessing the Internet even when the IISP is denying the service must be a concern for Regional Internet Service Providers (RISPs) who want to provide resilient Internet access for their subscribers.

## ***1.2 MOTIVATION***

Since network security and availability are of major concern for Saudi Arabia, it should not completely depend on one International ISP to get access to the Internet. So resiliency of the Internet is a must for the local ISP. Therefore, the potential for International ISPs to block access to the Internet maliciously must be investigated and counter measures must be proposed. This work will enable us to better understand the BGP weaknesses and provide higher Internet availability for the region.

## ***1.3 PROBLEM DESCRIPTION***

Local ISPs which provide Internet reachability for their customers are getting the Internet access through IISPs. These IISPs are paid in a customer-provider relationship in order to provide Internet access. The IISPs can make use of the routing protocol they run



in order to maliciously block access to some services. These IISPs can also claim that they have a route to destinations inside the local region while blackholing the traffic destined to the prefix owned by the local region. Therefore, a malicious IISP can be defined as a provider that advertises a prefix of a local region and advertises Internet prefixes to the local region, while blackholing the traffic of the local region to the Internet and blackholing the traffic coming from the Internet to the local region. Figure 1-1 depicts the described scenario. This thesis focuses on finding a solution to bypass the malicious IISP despite its false advertisements. While the blockage detection is not part of this work, however it is simple to detect and implement [3]. This can be done by implementing a traceroute-like technique to identify that the traffic stops at the malicious IISP [3].

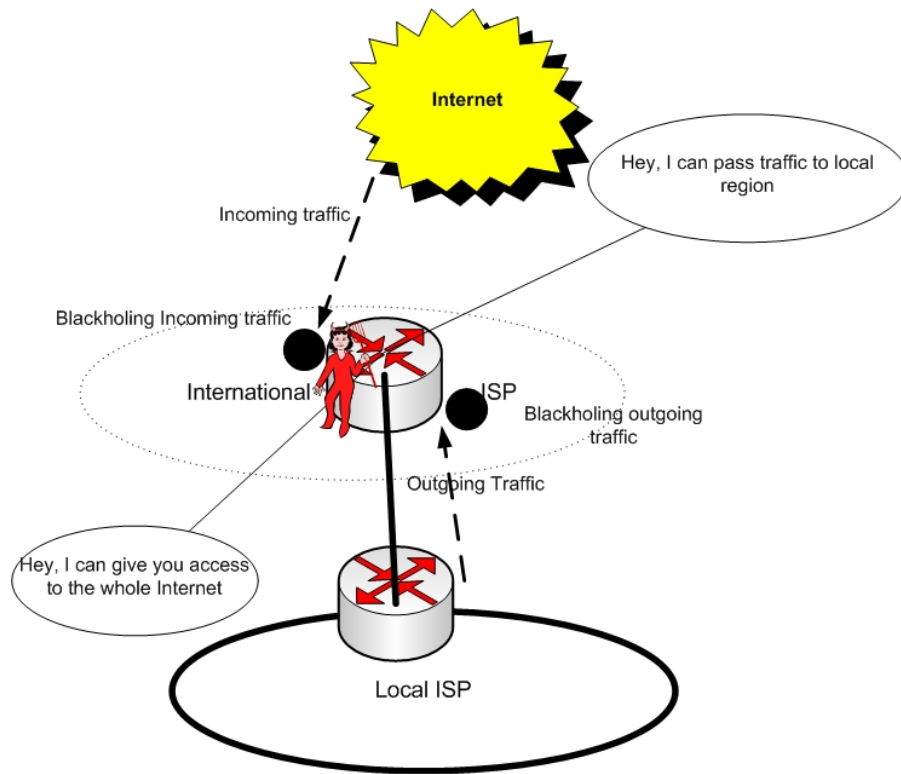


Figure 1-1 Malicious IISP blocking.

Two major approaches exist to solve the problem. One approach is to hide one's identity by using NATing, tunneling, etc. Another approach is to have more than one IISP, i.e. multihoming, and control the traffic either by using an overlay protocol, fine tuning, or modifying the BGP protocol to perform traffic engineering. The latter approach is used to direct the outgoing and incoming traffic so that they do not pass through the malicious IISP. Figure 1-2 and Figure 1-3 illustrate these approaches. We will concentrate in this thesis on the BGP based solutions and not on NATing or pure tunneling. It should

be noted that if the IISP is only denying service access without false advertisements with the intention to blackhole legitimate traffic, then the existence of another IISP that is not denying the access will solve the problem. This is because all incoming and outgoing traffic will pass through the other IISP since it is the only provider that advertises the prefixes of the local region and a BGP session can be established with it alone. A solution needs to be investigated if the IISP is advertising the prefix of a specific region to the Internet and/or advertising Internet prefixes to this region while blackholing traffic coming to the local region from the Internet and/or traffic going from this region to the Internet.

In Figure 1-2, the local ISP may be supported by a cooperating neighboring router to set up a tunnel, or to use the path through a neighbor while changing the source address (NATing) in order to send traffic to destination. In the same way, the destination should direct its traffic through the cooperating neighboring router.

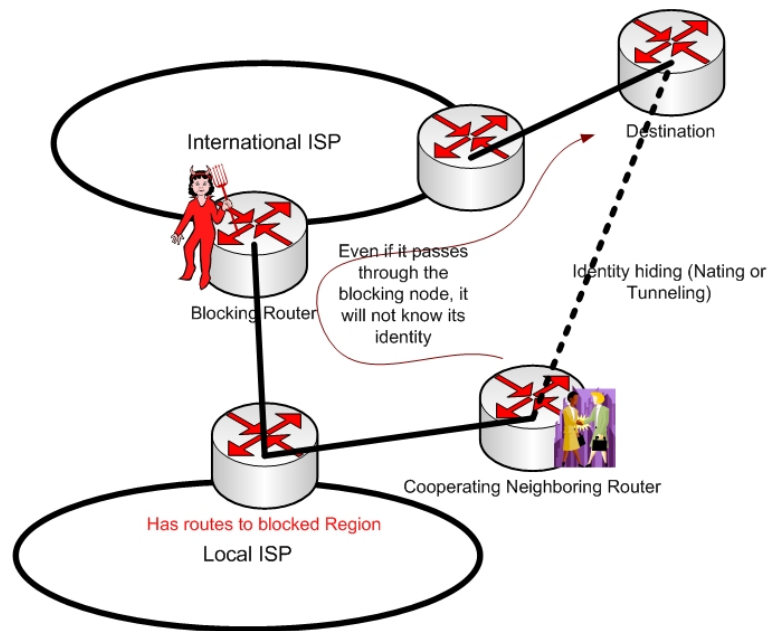


Figure 1-2 Identity hiding techniques.

Figure 1-3 depicts traffic engineering approaches that focus on directing the traffic, both incoming and outgoing, through the good or non-malicious IISP. This may be performed through configurations of the BGP protocol, and may also be done through some cooperative algorithms [4].

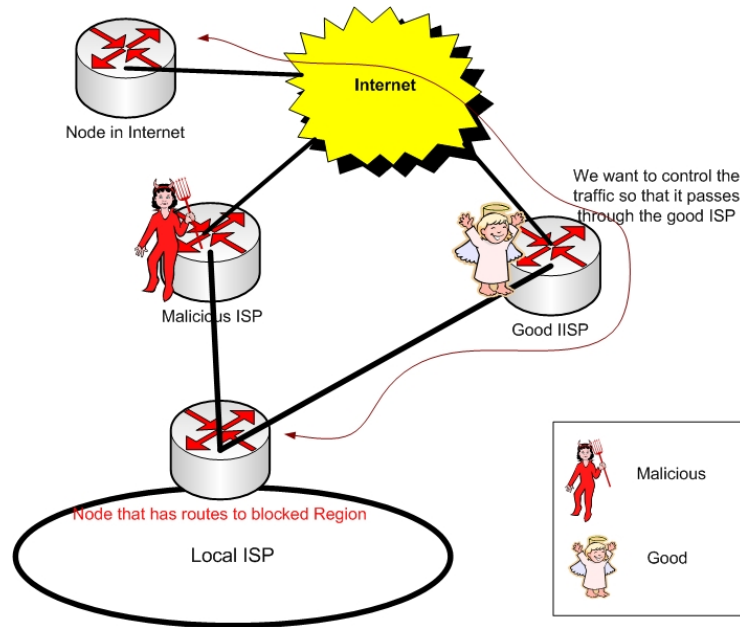


Figure 1-3 Traffic engineering based techniques.

## 1.4 SCOPE

Simulation is used for testing the validity of the solution. We do not consider deploying a real experiment on the Internet. We focus in our research how to direct the traffic rather than how to detect whether there is a blockage or not. Since we will not focus on hiding one's identity, we will not consider the case when the path of the traffic must pass through the blocking ISP, although we propose techniques on how to tackle this issue.

## ***1.5 ASSUMPTIONS***

1. We assume that we will be able to add another International ISP as part of the solution.
2. We assume that only one of the ISPs is blocking the Internet access.
3. We assume that we know apriori which ISP is blocking.

## ***1.6 LIMITATIONS***

One limitation is that, if the traffic need to pass through the malicious International ISP, then the traffic might be filtered. Another limitation is that the control generally of incoming traffic is usually undeterministic which means that we do not have assurance that the traffic will go where intended. In addition, deterministic control of traffic requires communication with those who are going to forward traffic to us [4].

## ***1.7 THESIS ORGANIZATION***

The thesis is organized as follow. In CHAPTER 1, we provide formal description of the problem, its scope, the limitations and the focus of the research. In CHAPTER 2, we provide the basic background knowledge to understand the rest of the thesis where the



Internet and the Border Gateway Protocol (BGP) are introduced. If the reader has this background he can safely skip CHAPTER 2. In CHAPTER 3, a literature survey of related work is presented. The reader does not need to go through the literature survey to understand the work, but reading the literature survey will allow the reader to look at different angles and areas investigated in the past. In CHAPTER 4, we propose three major BGP based techniques to solve the problem. The techniques are BGP tuning, virtual peering, and virtual transit. BGP tuning includes a set of techniques to control the incoming traffic, namely, community, shortening, and more specific prefixes and to control the outgoing traffic by the setting local-preference attribute. These techniques are qualitatively analyzed and compared with each other. In CHAPTER 5, we describe the design, implementation, and validation of BGP tuning techniques in OPNET. In CHAPTER 6, a performance evaluation of BGP tuning techniques is presented. The techniques are studied for different configurations to test the packet drop, convergence time, throughput, and application specific packets sent and received. In CHAPTER 7, a conclusion and future direction for the thesis are presented. The appendices show the code modification in OPNET and instructions on how to run the code and the simulation.

## ***1.8 SUMMARY***

In this chapter, the problem is defined and possible ways of solving it are presented. The concentration of the work of the thesis is declared. The scope, assumptions, and limitations are noted. Finally, the chapter presents the organization of the thesis.

---

## ***CHAPTER 2***

### ***BACKGROUND***

#### ***2.1 INTERNET***

Internet is a large network consisting of a very large number of smaller networks. Nodes and links are the main components of networks. Nodes can either be end systems or hosts and intermediate systems or routers. Different types of links exist; one can refer to [5] to know more about Internet connections and links. Information travels from a source to a destination through a series of nodes called path. Routing Protocols select the path through which information travels and it does that by exchanging reachability information between nodes. Reachability Information is the information that enables routers to know where other hosts and routers are.

Networks are managed by ISPs that can be classified into tiers depending on the size of the networks the ISP operates. Tier 1 ISPs often have global backbone networks while tier 2 ISPs have regional wide networks. Tier 3 ISPs normally provide Internet access to end users [6]. A network that is administered by a single organization is called an *Autonomous System (AS)*. The Internet is composed of a large number of Autonomous

Systems (ASes). Figure 2-1 depicts a general overview of the Internet. Each cloud in Figure 2-1 represents an AS.

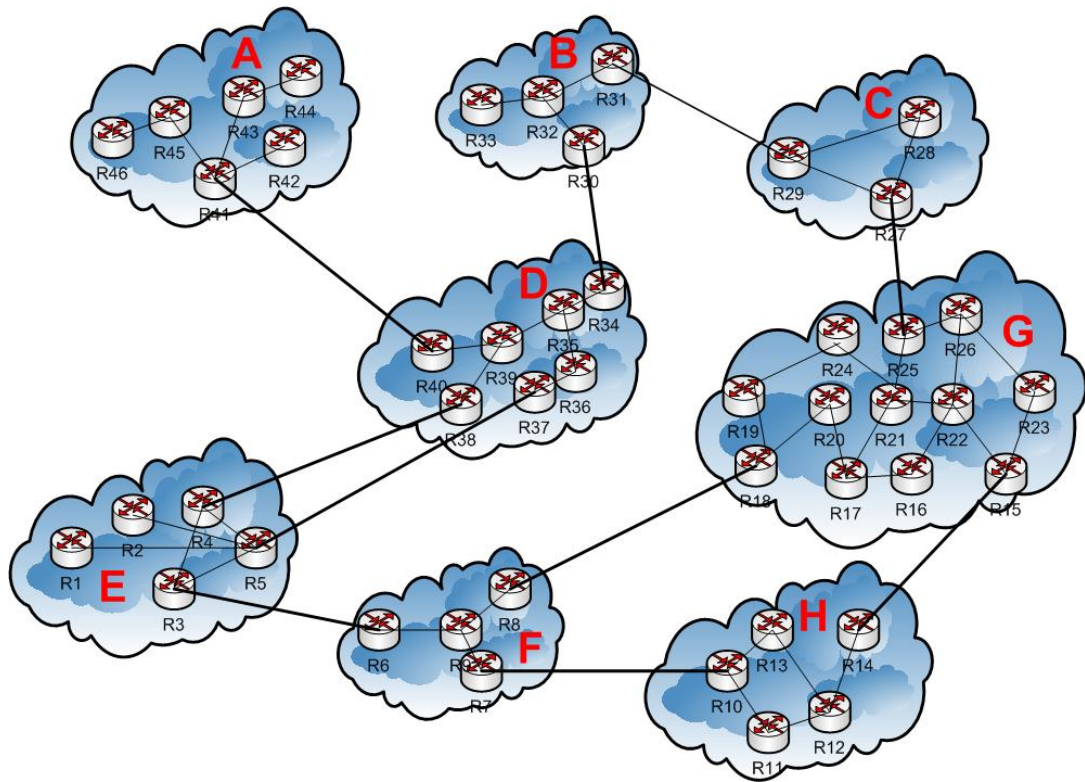


Figure 2-1 Overview of the Internet.

A *transit* network is a network that delivers traffic from one network to another, whereas, a *stub* network is a network that does not work as a transit network. The stub network only carries traffic if it is either a source or a destination of that traffic. The relationship between routers in the Internet is either *peer-to-peer* where routers

exchange traffic in one to one relationship without money involvement, or *customer-provider* where the customer pays money for the provider for connectivity to the Internet. A *multihomed* customer is one that has more than one provider that it is connected to.

The fundamental authority of all IP addresses is the Internet Assigned Number Authority (IANA) [6]. Initially when the Internet was not big, organizations apply directly to IANA. In 1993, the US Department of Commerce chose the Internet Corporation of Assigned Numbers (ICANN) for administration of IP address Space [7]. Four Regional Internet Registries (RIRs) are created to allocate addresses for particular regions. The American Registry for Internet Numbers (ARIN) is responsible for administering the delegation of IP address space in North America. Most of the address space allocation in Europe, the Middle East, and North Africa is delegated by the Réseaux IP Européens (RIPE). Asia-Pacific Network Information Center (APNIC), as its name suggests delegates IP addresses in Asia and the Pacific region. Finally, Latin American and Caribbean Internet Address Registry (LACTIC) delegates addresses to the corresponding region. Figure 2-2 shows a sample of address delegation in the Internet [7].

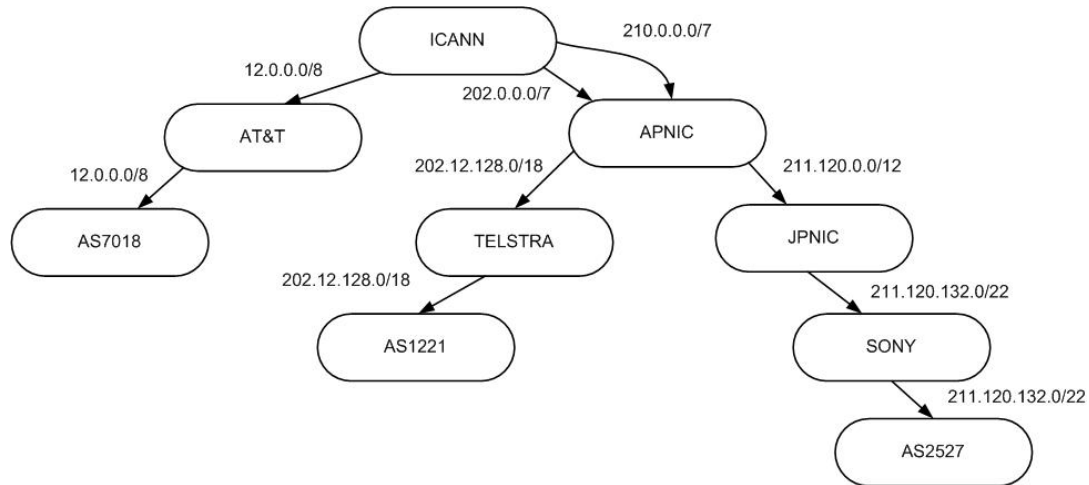


Figure 2-2 Sample of Address Space Delegation [7]

ISPs, in general, delegate address space to their customers, i.e., organizations' and enterprises [7]. It is difficult to know which ISPs delegated which IP prefix to which organization since we have a huge number of IP prefixes (more than 180000) and there is no rule that controls delegation of IP addresses [6].

Contractual agreements or Service Level Agreements (SLAs) are defined between ISPs and their clients to determine the quality of service that must be provided. BGP allows one to filter received routes from other peers and to filter advertisements to them; also it might select routes based on specific conditions. BGP succeeded to be a "policy-based inter-domain routing protocol." [7]

Within each AS, routers run an Interior Gateway Protocol (IGP) such as the Routing Information Protocol (RIP) or the Open Shortest Path First (OSPF) protocol. On the other hand, for communication between ASes, routers run an Exterior Gateway Protocol (EGP) such as Border Gateway Protocol (BGP).

There are two main types of routing protocols; namely, distance vector and link state. In a distance vector protocol, a routing table is retained and this routing table is made up of a set of vectors that correspond to a specific destination in a network. Occasionally, nodes advertise their routing tables to their direct neighbors who update their routing table based on some distance metric and the advertisements they receive. RIP is an example of a distance vector protocol. In the link state protocol, each node floods Link State Advertisement (LSA) – which contains a link identifier, its state, and its neighbors – to every other node in the network. All nodes build an exact link state database and run a common algorithm (Dijkstra's) to compute the shortest path according to some cost. Examples of such routing protocols include OSPF and Intermediate System protocol (IS-IS).

## ***2.2 BORDER GATEWAY PROTOCOL***

The Border Gateway Protocol (BGP) defined in RFC 4271 [8] is the de facto standard for routing in the Internet. BGP is a path vector protocol which is based on a distance vector approach. While a simple distance vector protocol, such as RIP, has a simple metric coupled to it, BGP utilizes many attributes and enables the selection of the best route based on a local policy. Attributes include the `Local-Pref` which enables the ISP administrator to set which route it prefers for a specific destination, and the `AS-PATH` which is a sequence of AS numbers defining a route. BGP is used by ASes to determine how to route traffic to other ASes. Thus, the main function of BGP is the exchange of *Network Reachability Information* (NRI) between BGP systems. This information includes a set of ASes which is sufficient for building a graph of ASes in order to reach a destination. In this case, a *BGP speaker*, which is a router that runs the BGP protocol, can eliminate routing loops by deleting a repeated AS number in the path and enforcing routing policies. BGP uses a destination-based forwarding paradigm where the router's decision to forward a packet is only based on the destination address. So, BGP supports only policies that obey the rules of this forwarding paradigm, meaning that in order to



construct a policy it should be based only on the destination address. In order to have a reliable communication for BGP, BGP runs over the Transmission Control Protocol (TCP).

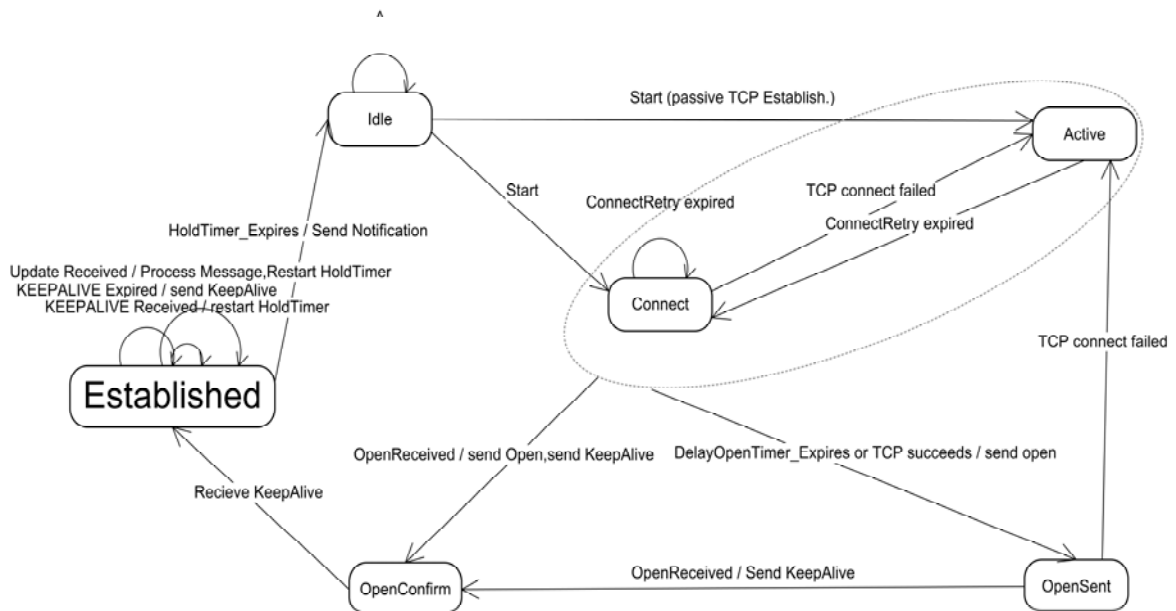
BGP defines a number of rules that help in selecting the best route. The first rule is to take the route with a higher degree of preference that is usually set by the `Local-pref` attribute. If more than one route has the same local preference, BGP will look for the shortest `AS_Path`. If they are the same, BGP will look for the lowest `Multi_Exit_Disc` (MED) value if they have the same `Next_Hop` value. Then, if a decision cannot be taken, the path with the lowest cost to the `Next_Hop` of the route is selected. Finally, the route with the lowest BGP identifier is chosen if more than one route is still present. [6]

BGP, in its basic implementation, provides no protection and security and that makes it vulnerable to many attacks [9]. There are three main drawbacks of BGP. The first one is that BGP does not check that the message is not modified (Integrity). Also, it does not make sure that the message is new or not replayed (freshness). Moreover, it does not verify that the originator of the message is a legitimate one (origin authentication). Second, BGP does not check the authority of announcing an AS-Path by a specific

Autonomous System. Finally, the genuineness of path attributes announced by a specific AS is not checked in BGP. [7,9]

In order for BGP to function, four basic messages are exchanged. A KEEPALIVE message may be sent regularly to ensure a live connection. A NOTIFICATION message is sent in case of errors or special conditions and then the connection is closed. An UPDATE message is used to advertise routes, which contains a set of destination and path attributes. More than one route can be advertised in a single UPDATE message by including more than one prefix in the network layer reachability information (NLRI) field in the UPDATE message. In case a previously advertised route is no longer available for use, a BGP speaker can inform its peers by advertising this route in the WITHDRAWN ROUTES field of the UPDATE message, or in a replacement route of the NLRI, or closing the connection to remove all routes. [8]

Figure 2-3 shows a schematic of the finite state machine implemented by BGP.



**Figure 2-3 Border Gateway Protocol Finite State Machine**

This diagram depicts the most important events that cause a change to the next states. Error events or unexpected messages usually lead to going to the Idle State after freeing all resources [these are not shown in Figure 2-3 for clarity]. The most important state is the 'ESTABLISHED' state where update messages can be exchanged between peers. Each peer connection is initially in the 'IDLE' state. When it receives manual or automatic start it goes to the 'CONNECT' state after initializing resources and starting TCP session with peers. If the start signal comes with a passive TCP establishment, this means to wait for the other peer to start a TCP connection, then it goes to the active state. If the `ConnectRetry_Timer` event happens, which is used to control the time

to retry the connection, and then it is restarted and stays in the 'Connect' state. If TCP connection fails the system goes to the 'ACTIVE' state where it waits for a connection. Whether we are in the 'CONNECT' or 'ACTIVE' states, if TCP succeed or DelayOpenTimer\_Expires [which is event waiting for the other side to start opening the session], the system sends an 'Open' message, to open a BGP session, and goes to 'OPEN SENT' message. If the system is in the 'CONNECT' or 'ACTIVE' and it received Open message, it sends 'Open' message and then a 'KeepAlive' message and goes to the 'OPENCONFIRM' state. If the system is in 'OPENSENT' state and it receives an 'Open' message, it sends a 'KeepAlive' message and goes to the 'OpenConfirm' state. If the peer receives a 'KeepAlive' from the other side it goes to the 'Established' state where a 'KeepAlive' and 'Update' messages are interchanged. If 'KeepAlive' message is received we restart the 'HoldTime' timer. If KeepAliveTimer\_Expired event occurs the BGP speaker resend a 'KeepAlive' message and restart the timer. If an 'Update' message is received, the 'Update' is processed and the HoldTime Timer is restarted. In the case when the 'HoldTime' timer expires, the router sends a 'Notification' message and changes the state to IDLE.

---

## ***CHAPTER 3***

### ***LITERATURE SURVEY***

#### ***3.1 INTRODUCTION***

In this chapter, we survey the literature related to our work. Since BGP is the main concern of our work, we initially look at some of the main BGP RFCs. Then, we present the BGP multihoming issue, because in our solution, we will consider that the local region is connected to more than one provider (multihoming). After that, we survey the issue of BGP threats and attacks to determine what the weak points of BGP that might be considered are. This is followed by a survey about BGP security techniques. The Multiple Origin Autonomous System (MOAS) conflict is then looked at, since it is used in some prefix hijacking detection systems which we discuss in the section that follows. We consider prefix hijacking because in our problem the ISP is ‘hijacking’ or advertising the local region’s prefix and then blackholing the traffic. In the following section we discuss BGP misconfiguration which could have the same effect as BGP Attacks and MOAS. In order to control traffic, we can either have an overlay protocol or utilize traffic engineering and modify an existing protocol (BGP). The two sections that follow discuss each one of them. BGP simulation tools are then investigated. Then, we present an

overview of a paper that describes the behavior of BGP under stress. Finally, a summary of a paper that analysis BGP convergence time is presented.

### ***3.2 BGP RFCs***

There are some of Request for Comments (RFCs) that describe BGP. The main one is RFC 4271 which includes the details of the protocol [8]. This RFC obsoletes the old description about the protocol in RFC 1771 [10]. BGP Security Vulnerability Analysis (RFC 4272) presents some security related topics with BGP, and lists the drawbacks of BGP protocol because it does not have a protection against modification, addition, or deletion of BGP messages [9]. RFC 2439 describe ways used to dampen a misbehaving route in BGP [11].

### ***3.3 BGP MULTIHOMING***

Liu X and Xiao L. surveyed the current research in multihomed stub networks [12]. They presented the two major multihoming technologies: BGP Multihoming and NAT Multihoming. BGP multihoming means that the internal router contributes to the overall BGP system, and is connected to more than one external BGP speaker. NAT Multihoming means that the local autonomous system has a server between it and multiple providers.

A non-aggregation problem is the problem of not being able to aggregate prefixes with each other because they are neither contiguous nor overlapping. NAT multihoming can easily avoid this problem, since it can easily translate addresses into addresses that belong to the provider prefixes so that the provider can aggregate them with its prefixes. BGP multihoming guarantees the uniqueness of IP addresses and is able to control the routing policy.

### ***3.4 BGP THREATS & ATTACKS***

Nordstrom and Dovrolis [13] discussed the types of BGP attacks and pointed out four main goals of BGP attacks. The first goal is *blackholing*, which is to drop the traffic that arrives to it. *Redirection* is the second goal, which is to send the traffic to a different destination and reveal the secret information contained in it. The third goal is *Supervision* which is a redirection but with the intention to modify the data and then send the packet to the correct destination. Finally, the fourth goal of BGP attacks is to *cause instability in the network*, which could happen by sending successive advertisements and withdrawals. This attack could happen by sending a false update or by prefix hijacking through announcing a prefix that the hijacker does not own or advertising a path it does not have. Another type of attack is link flapping, i.e., the

announcement of a link failure then another announcement of the recovery of the same link several times, to trigger a route flap dampening [11]. An incident of network instability due to prefix hijacking happened in April 1997 when AS7007 advertised most of the routes of the Internet causing an Internet outage for more than two hours [13,14,15]. Another incident of network instability occurred in April 2001 when AS3561 forwarded a huge number of wrong advertisements from one of its downstream customers causing problems in connectivity [14,16]. The deaggregation of a prefix can attract traffic anywhere in the Internet due to the longest prefix match mechanism. It can then blackhole the victim network. This is because when a router selects where to route the traffic directed to a specific destination, the router first looks in its routing table for the longest prefix that matches the destination IP address even before running any selection process. There are many proposed counter measures against these types of attacks discussed in [13], two of them are discussed. The first one is the use of route filtering in order to enable ASes to filter out malicious or faulty updates; however, this requires them to know what to filter. In order to get ownership information of prefixes, Internet routing registries (IRR) databases can be consulted; however, these databases are not up-to-date [13]. The second solution is the use of a Secure Border Gateway



Protocol S-BGP [17]. Although this method can provide high security against attacks, it will add a high overhead on the Internet.

### ***3.5 BGP SECURITY***

Since there is no one point that controls the routing and because of the presence of many autonomous systems, each with its own policy, security is very important in the routing protocol that controls this system [18]. Nicholes and Mukherjee [18] surveyed literature from 1988 to 2007 related to BGP security. They presented several methodologies used to secure BGP against attacks. Such attacks might be attacks on BGP peering like TCP attacks (e.g., TCP RST which drops the peer connection and TCP SYN flood which makes a connection with an invalid peer), and mounting attacks on BGP messages like eavesdropping, spoofing, modification, replaying, and deletion of messages. The attack might be on BGP routers like remote login or misconfiguration of a router. Misconfiguration of a router causes the router to claim a prefix that it does not own or export what should be filtered. Physical attacks include power failure, environmental failure, and link failure. These attacks cause loss of network connectivity, and problems in network reachability (like claim of a prefix), propagation of an invalid path; and if this propagated path is shorter, it will be accepted by most routers. Nickoles

and Mukherjee then discussed ways to protect the Internet against these types of attacks. They divided the methodologies into five major types. The first type is the use of cryptography and attestation. Attestation means to prove the identity of the one who provided the routing information. This is done by signing the advertisement. The second way is the use of a database that is either a history database or topology of prefixes database. A history database stores routing information based on previously advertised prefixes for specific ASes for some times and uses heuristics to validate paths. A topology of prefixes database validates the path given the knowledge of topology from a central router. The Third way is overlay, in which, routers cooperate together in order to validate route updates. Overlay does not require encryption and that reduces the memory and computation overhead on routers; however, the absence of encryption will make overlay susceptible to the same type of attacks that BGP has. The forth way is the penalty which is based on delaying misbehaving routes. One example is BGP Route Flap Damping described in RFC 2439 [11] which penalizes a BGP router based on continuous announcement and withdrawal. Finally, Data-Plane Testing means to test paths in the Data-Plane either by traffic monitoring (passive) or by packet dropping (active).

Some studies of BGP focus on some changes of BGP attribute, like AS Path. For example, Blazakis et al. analyzed the behavior of AS Paths in the Internet and Introduced

the AS Path Edit Distance metric [19]. This metric can be used to represent changes of the AS path in the Internet, and this allows modeling and quantification of AS-Path values at specific ASes. They modified the Levenshtein distance [20], which is a measure of similarity between two strings, in order to perform comparison by considering AS numbers within an AS-Path as entities rather than characters. Since a change of the last AS path number should have a higher weight, a MAX\_EDIT\_DISTANCE attribute can be set to a larger value to represent this change. They tested their metric on the Internet and they found out that most changes in the path include one AS path and almost all of them are less than four. The AS-path Origin change appears as spikes, so path changes can be observed with previously known AS path anomalies.

### **3.6 MOAS CONFLICT**

Zhao et al. described that the occurrence of more than one Autonomous System appears to originate the same prefix and called it Multiple Origin Autonomous System (MOAS) [21]. RFC 1930 recommended that a prefix should be originated from only one AS [22]. The authors defined MOAS analytically by having two paths that advertise for the same prefix, then there is a MOAS if the last autonomous system in the AS path of one of them is different from the other. MOAS can be divided into three different types.

The first type is `OrigTranAS` where an AS is an origin in one path and transit in another. The second type is `SplitView` where one of the ASes in the AS path announce different routes to different neighbors. The third type is `DistinctPaths` where two paths are completely distinct. There are a number of reasons of MOAS. At the Exchange point address, every router can advertise the prefixes associated with the exchange point address. This does not cause problems since every router knows how to reach any prefix and lasts for long time. The second reason is multihoming without BGP when we have two ASes having a static link or IGP protocol between them. So, one AS can advertise a prefix from the other because it can reach the prefix directly. This one also lasts for long time; however, it causes problems in case the link fails between the two ASes. The third cause is multihoming with a private AS number. This happens when a customer is multihomed and the providers remove the private AS number from the path. This also lasts for long time and does not cause problems. This also could happen due to aggregation mentioned in RFC 1930 [22] or Anycast which suggests the use of multiple origins. Finally, MOAS could happen due to a malicious or a faulty configuration of the routers. As we can see, long lived MOAS could be due to exchange point or multihoming (without BGP or private AS). Also, short lived MOAS could be due to faults or intentional attacks and a change of a provider for a non-BGP customer.

### **3.7 PREFIX HIJACKING**

The prefix hijack attack can be addressed by using a prefix hijack alert system (PHAS) [23]. PHAS is an email notification system that alerts a prefix owner whenever there is a change in the origin AS that owns the prefix. Every day there are a number of prefix changes and most of them are valid. However, only the AS that owns the prefix is capable of differentiating between a valid origin change and a prefix hijack [23]. The system examines the data collected in RouteViews [24] and notifies the prefix owner about any possible hijack. The user initially needs to register in the system and specify the prefixes to monitor. While monitoring, if there is any change in origin, the user will be notified via email (user can provide multiple emails to increase the chance of viewing any notification). A filter can be implemented at the user side to filter out any notification of valid origin change. PHAS succeeded to detect known prefix hijacks and the time reported by the system matches the time mentioned in other sources.

Similarly, Zheng *et al.* [25] built an IP hijacking detection system. Their system depends on two observations noticed when there is no hijacking. The first observation is that the number of hops from a source to the prefix generally does not change (stable). The second observation is that the path from a source to the prefix covers the path from

the source to a reference point along the original path that is topologically close to the prefix. The paper focuses on two types of hijacking; the first type is *imposture* where the attacker imitates the behavior of the victim by responding to the sender of the hijacked traffic, and the second type is *interception* where the attacker spies on the traffic and records its content and then forwards it to the correct destination. These attacks are more challenging than blackholing which is dropping attracted traffic. The framework can be described as follows. For each prefix, a number of monitors are selected. Then, monitors measure network locations of prefixes to detect whether there is any considerable change in hop count distance measurements. The location does not change frequently, and when the difference between the previous location of the prefix and the current location exceeds a certain threshold, the path disagreement module is triggered. Because not every change in the location of the prefix is hijacking, and prefix hijack usually targets one specific prefix while a legitimate action involves a larger number of prefixes, the path disagreement test can be used to determine if a hijack occurs. A reference point is specified such that it is very close to the prefix and at the same time has an IP address that does not belong to the prefix. For each prefix, a number of monitors are chosen. The path to the reference point is compared with the path to the prefix and if they are different then a hijack is identified.

Hu and Mao implemented a prefix hijacking identification system [3]. This technique is based on collecting data from the control plane (passively collected BGP updates) and from the data plane (*fingerprinting*). Fingerprinting removes the ambiguity about an expected IP hijacking occurrence because it is based on information that identifies the hijacker. It is not possible for an attack to affect the whole Internet; more specifically routers which are close to the legitimate prefix owner most likely will not be affected. Fingerprinting based consistency check is based on information like Host OS properties, IP Identifier, TCP timestamp, and ICMP timestamp. The authors devised a way for each type of IP prefix attack type. The first type is to hijack a prefix by announcing its ownership. This will lead to a Multiple Origin Autonomous System (MOAS) conflict. However, MOAS can be a result of legitimate actions when there is multihoming with static link or multihoming with private AS numbers. So, fingerprinting based consistency check is used. Probing is performed to get fingerprints, then a check for any discrepancy is performed. This check will imply a potential IP hijack attack. The second way of attacks is to hijack a prefix and its AS by announcing a route to the hijacked prefix. For this case, we need to monitor all updates and perform an Edge popularity constraint and notice whether the prefix obtained was observed before. This update is highly suspicious if it is not the case. A geographic constraint is then applied to check if the two ASes are

physically separated. Finally, a relationship constraint can be conducted to check if there is any violation to routing policies inferred from AS relationships. The third type of hijacking is to hijack a subnet of a prefix by announcing a subnet of a prefix that it does not own. This will lead to subMOAS (MOAS of subnet of a prefix). Legitimate occurrence of subMOAS can occur due to similar reasons mentioned above about MOAS in addition to the case when a customer is multihomed to two providers. The first one aggregates the prefix with its own, and the other one cannot do aggregation and announce a path to a subnet of the prefix announced by the first router. To detect subMOAS, we need to monitor subMOAS updates. The authors assume that the provider will not hijack a customer's routes on purpose because there is no economical motivation for that and it is easy to discover this type of attack through traceroute-like probing [3]. The detection is performed by monitoring subMOAS updates. Then a customer-provider check is performed by deducing a relationship between Autonomous Systems by assuming that legitimate AS-Paths are *valley-free*. Valley free means that if the path goes down it cannot go up again, or no AS-Path can have a customer-provider edge after a provider-customer edge and there is not more than one peer-to-peer edge. Since a tier-1 AS is the highest point, routes before it are all customer-provider edges and what appears after a tier-1 AS are all provider-customer edges. The AS-path starts with the AS number of a



customer that announces its prefix to its provider then its provider which is a customer of another provider adds its own AS number to the AS-Path. This continues until a tier-1 AS is reached. After this point every provider announces to its customer that a destination can be reached through them and as a result adding their AS numbers to the AS-Path. A reflect-scan technique needs to be performed in order to fingerprint the victim network. This can be done by probing the prefix of interest for IP ID and then sending a spoofed message to an address outside the subnet of a prefix but inside a prefix with the source IP address being an IP of the subnet of interest. Then, we probe again the IP ID from that prefix. If this IP ID is incremented by 2, then we declare that there is an attack, and if it is decremented by one then there is no attack. Usually, IP ID is incremented by one by the system for every message sent. The fourth type is to hijack a subnet of a prefix and its AS by announcing a path to a subnet of a prefix and add it to its AS-Path. This attack causes neither MOAS nor subMOAS and utilizes the longest prefix hijacking to attract traffic. The detection can be done by monitoring new non-subMOAS prefixes and then performing edge, geographic, and relationship constraints and then doing a reflect scan.

### **3.8 BGP MISCONFIGURATION**

Mahajan et al. [14] studied the impact of misconfiguration on the routing load and global connectivity. Their paper provides an explanation on why misconfigurations occur and suggests techniques to lessen their rate of recurrence and impact. They focused their study on two major misconfigurations. *Origin Misconfiguration* can be due to incorrectly summarizing the address space resulting in the injection of more specific prefixes into the global routing table. It can also be due to a hijack, i.e., announcing a prefix that one does not own, or it can happen because of propagation of prefixes that are not intended to be announced. The second main misconfiguration is *export misconfiguration* which occurs when the AS Path violates routing policies, i.e., when a router exports a route that should have been filtered. They studied the impact of misconfiguration routing load, connectivity disruption, and policy violation. In analyzing origin misconfiguration, short lived events are considered due to misconfiguration. Origin misconfiguration can be classified into three types. The first type is *self de-aggregate*, when a prefix de-aggregates or does not summarize its own prefix. *Related origin* is the second type, which occurs when the origin of one route appears in the AS path of the other. The third type is *false origin*, which happens when the prefix is

announced by a different origin as a result of address space hijack. To analyze “export misconfiguration”, the authors benefited from Gao’s two observations about AS relationships [26]. The first observation is that every legitimate AS-path must be valley-free, which means that once the AS-path descend, it will never ascend once more, i.e., when an edge from provider to customer appears, no customer to provider link will appear after it in the AS-path. The second observation states that only one peer-to-peer edge can appear in the AS-path and at the highest point of the path. It can be deduced that the AS is provider if it has larger number of neighbors [26]. So, short lived paths that contradict with valley-free conditions or have more than one peer-to-peer connection are identified as possible misconfiguration. Email survey to prefix owners is conducted to evaluate the correctness of results. According to [14], misconfigurations can be due to initialization bug, dependence on upstream filtering, legacy configuration, redistribution, non correct use of communities, hijacks and many others.

### ***3.9 OVERLAY PROTOCOLS***

Andersen [27] described in his master thesis the use of Resilient Overlay Networks (RON) for the detection of Internet outage and finding an alternative path to enhance reachability. The purpose of RON is to increase the reliability of a distributed application

by finding paths that BGP cannot find. An Overlay network is a “virtual network created on top of an existing network” where nodes work as routers for each others. The idea of RON is based upon the fact that the Internet is a “mesh of interconnected networks”, so many paths might exist. By limiting the number of nodes, one can allow more information about paths to be exchanged and hence be able to decide on a better path. RON can provide detection of faults, more reliability for applications, and better performance.

In [28], an Overlay Policy Control Architecture (OPCA) is proposed. The purpose of OPCA is to support fault tolerance and improve routing between ASes. OPCA consists of five major components. A policy agent is responsible of announcing and enforcing policies to perform a route selection. A policy database helps policy agents to decide what to select as a route. Measurement information helps on deciding when to make a change in the route and measure the effectiveness of such a change. This information includes E-BGP link characteristics (e.g., capacity and bandwidth load) and customer-server traffic characterization (e.g., total bandwidth, average latency) between the customer and the provider. The forth one is PA directory which is used to know which ASes implement OPCA. The last one is the AS Topology & Relationship Mapper (RMAP) in

order to decide the flow of traffic and routing on the Internet according to route export rules.

### ***3.10 BGP TRAFFIC ENGINEERING***

Quoitin [4] designed and implemented a BGP modeling tool called C-BGP which can be used to compute routes in a large scale network topology [29]. Then, he studied the approaches of controlling outgoing and incoming traffic when the Regional ISP is multihomed. Usually, this control is either for the purpose of having a backup route or for load balancing. In his dissertation, he proposed a cooperative approach, called virtual peering, in order to provide a deterministic approach of controlling the incoming traffic [30]. His approach modifies slightly the BGP protocol by automating the establishment of virtual peering and adding BGP messages specifically to accomplish that. Only end systems need to have these modifications and there is no effect on intermediate systems. He used virtual peering in order to make load balancing for the incoming traffic and for selecting the path with the lowest delay. In this thesis, a virtual peering solution is adapted in order to solve the problem of malicious Internet blockage. CHAPTER 4 provides more details about virtual peering.

### ***3.11 BGP SIMULATION TOOLS***

In addition to C-BGP discussed in section 3.10 written by Quoitin [4], a BGP++ is a C++ implementation of BGP over NS-2 and GTNets. It is built by extending Zebra bgpd open source code that simulates BGP [31]. Confederations, route reflection, and flap dampening are implemented in addition to other BGP specific techniques. BGP++ also supports BGP confederations, route reflection, flap dampening, and route-refresh capability.

Wojciechowski proposed, in his master thesis, to build a BGP simulator [32]. In order to make a large-scale simulation of something similar to the Internet, he abstracted the intra-domain communication, network, and protocol. The author claims that the reason of BGP simulation is that BGP is too sophisticated to interpret analytically. The way the simulator is built is not to compute protocol parts that are not important for the simulator outcome. For this reason, link delays are ignored, no connection creation is used, and no KeepAlive messages are implemented. The author used a mathematical function in order to model the delay of message propagation inside Autonomous Systems. Since the author intended to calculate the convergence time, he considered two main factors that slow the convergence time. The first one is Minimum Route

Advertisement Interval (MRAI), which stands for the least amount of time that must elapse between two consecutive update messages sent to the same neighboring router, and Route flap dampening, which is a technique that ignores an update from a router that continuously sends announcements and withdrawals by penalizing it. The author made his simulation to match real life as much as possible, and to make the entity run like if it is in real time. A distributed homogeneous cluster is used for emulation and Java is used as a programming language. An AS is modeled as a 'compute Node'. He models the arrangement of events and providing some control by using a 'coordinator Node'. In order to validate his experiment, he used BGP beacons which study and observe special prefixes. These beacons use Route Views Monitors, which are sinks that only make peer connection with other peers and store advertisements and withdrawals information [32] [33]. He compared the convergence time of his simulator and BGP beacons and described why there are some differences.

### ***3.12 BGP BEHAVIOR UNDER STRESS***

Wang et al. [34] studied the behavior of BGP under stressful circumstances, when reliable routing is desired. In order to perform their experiment they used the following classification. The BGP update is divided into either announcement or withdrawal.

Announcement can be divided into new announcement, duplicate, or implicit withdrawal. The new type can be a flap, a table exchange, a new announcement after withdraw different attribute (NADA)<sup>1</sup>, or a plain new announcement. The implicit withdrawal can be subdivided into SPATH when there is no change of the AS-path or DPATH where there is a change in the AS-path. The largest change is in BGP table exchange and implicit withdrawal; however, about 40.2% comes from monitoring points. Most BGP sessions restarted numerous times during the time of the worm attack, and every system reset means transferring the whole routing table, so even a small number of session resets can result in a huge amount of BGP updates. Because of fast recovery, the authors concluded that these updates are due to congestion or routing problems. The authors mentioned also some drawbacks of BGP. Prosperous variety of routing policies causes incompleteness in the routing information that a monitoring point may receive and that causes the lack of necessary data to infer causes.

---

<sup>1</sup> This means that for a certain path, this path is first withdrawn, then a new announcement come for the same path but with different path attributes.



### ***3.13 BGP CONVERGENCE ANALYSIS***

Griffin and Primore [35] analyzed experimentally BGP convergence time. The authors studied the relationship between the convergence time and the timer that is used to limit the rate of transmitting routing messages which is called Minimum Route Advertisement Interval (MRAI). This timer damps some of the oscillations inherent in the path vector approach to routing. The authors also utilized Lobovitz observations about convergence time [36,37]. Lobovitz noticed that setting of two functionalities reduces the convergence delay. The first one is sender side loop detection (SSLD), which is an optimization in which a router discovers AS-path loops before an announcement is sent to a neighbor. The second one is withdrawal rate limiting (WRATE), which limits the rate of withdrawal messages as well as advertisement. They divided the experiments into two kinds of experiments; UP: when a single destination is advertised and the system is left to converge, and DOWN: when an origin withdraws the destination it announced in the UP experiment and the system is left to converge. The authors came to the following conclusions. For each network and for each kind of experiment, the number of BGP updates goes down as we go from the value of 0 of MRAI and we increase it until we reach an optimal value  $M_u$  beyond which the number of BGP updates is stable. In the

same way, as we increase the MRAI going from 0, the required time to converge goes down until we reach an optimal value  $M_t$  beyond which the required time to converge increases. The authors concluded also that while the number of updates is constant, the minimum time to converge is directly proportional with the average router workload. In spite of the dramatic decrease of convergence time by choosing an optimal value of MRAI ( $M_t$ ), this value is not the same for all networks and it may not be easy to compute. Setting the value of WRATE can increase or decrease the convergence time depending on the network and kind of experiment. WRATE has an insignificant effect when MRAI is set to the optimal value  $M_t$ . The convergence time is either decreased or not changed by setting the SSLD value.

---

## ***CHAPTER 4***

### ***QUALITATIVE ANALYSIS OF METHODS FOR CIRCUMVENTING MALICIOUS ISP BLOCKING***

#### ***4.1 INTRODUCTION***

In this chapter, the methods to circumvent malicious acts by IISP are explained and their advantages and disadvantages are discussed. The first solution is based on BGP tuning is presented where the focus is on configuring the router(s) to direct the outgoing traffic and to influence the incoming traffic to pass through the good IISP. The second solution utilizes virtual peering which uses a multi-hop BGP session and establishes a tunnel to control the traffic through the intended ISP to provide a deterministic control of incoming traffic. For the third solution, we propose a virtual transit approach in which multiple routers distributed across the Internet work as a transit for the blocked local region. This solution extends virtual peering so that routers advertise a shorter path to other peers on the Internet. We qualitatively compare the three proposed solutions in terms of traffic filtering, setup overhead, communication overhead, difficulty to offset the solution, and scalability.

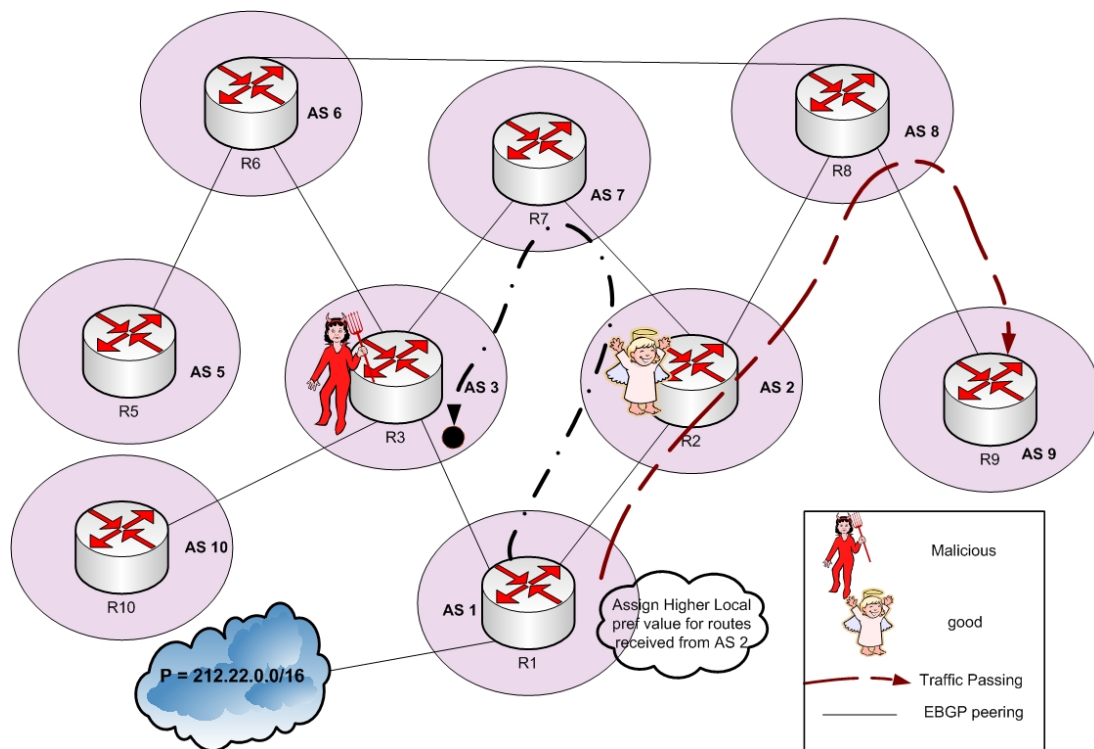
## ***4.2 METHODS TO COMBAT INTERNET SERVICE DENIAL***

### ***4.2.1 BGP Tuning***

BGP Tuning means to use the available BGP policies in order to modify the BGP selection process to enforce the selection of the intended route as the best one. BGP tuning has been used as a way for traffic engineering [38]. In order to make sure that the traffic passes through the good IISP, the outgoing traffic needs to be directed through the good IISP and the incoming traffic needs to be influenced to select the good IISP as the best path in its BGP selection process. Therefore, we look at ways of controlling the outgoing and incoming traffic to direct routes through the non-malicious IISP.

#### ***4.2.1.1 Control of Outgoing Traffic***

Controlling the outgoing traffic is easier than controlling the incoming traffic because it is easier to configure the local router to prefer a route than to affect the selection process of all other routers in the Internet that are outside the control of the local ISP to choose the path through a specific IISP. To control the outgoing traffic, the administrator of the RISP can set higher `local-pref` attributes of the routes learned from the good IISP. This assignment will ensure that all destinations reachable through



As shown in Figure 4-1, the traffic will go to all reachable destinations through the good IISP. For example, the traffic destined from AS1 to AS9 will pass through AS2 and AS8 and it will reach the destination through the good IISP. However, if we assume that

router R1 in AS 1 wants to send traffic to router R10 in AS10, then the only way to reach the destination is through router R3. In this case, the traffic will be filtered out and the destination will not be reachable. Setting up a tunnel or using any identity hiding technique can help to solve this problem.

#### ***4.2.1.2    Control of incoming Traffic***

Three ways to control traffic using BGP tuning will be investigated. These methods are AS Path shortening, more specific announcement, and communities. To achieve a better control of the incoming traffic, the three techniques can be combined and used together rather than individually.

##### ***4.2.1.2.1        AS Path Shortening***

In the selection process, when comparing two routes, if an AS Path of one route is shorter than the other, it will become more preferred. AS Path prepending has been widely used to make the path to a specific destination less preferred [39]. If the good IISP sends an announcement of our prefixes directly without adding the AS number of the local region in the AS-path, the length of its AS-path will be reduced by one. In this way, the incoming traffic can be influenced to come through the good IISP. In Figure 6, AS1,

which wants to control its traffic, will not have its AS number included in the AS-path advertised by AS2. In this way, AS7, for example, will prefer the route that comes from AS2 because it has a shorter AS Path to the prefix. However, if an AS has a local preference that leads to preferring routes to the prefix through the malicious router, then shortening the route will not influence the traffic. This is because `local-pref` is looked at first in the selection process of BGP. In addition, if the only way to reach a destination is through the malicious router, then the routes will be blocked. All these scenarios are shown in Figure 4-2.

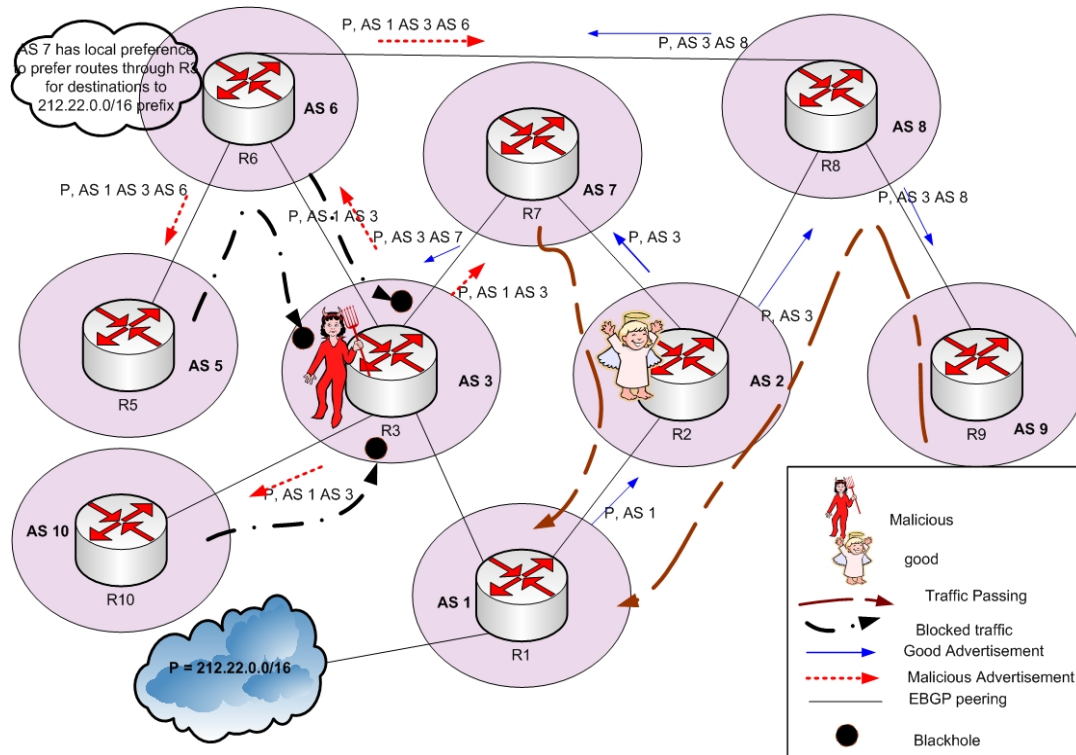


Figure 4-2 AS Path Shortening

As shown in Figure 4-2, the traffic will be influenced to go through the shorter path. For example, router R9 in AS9 will select to go through the path AS8 AS2 AS1. So, it passes through the good IISP, i.e., AS2. In case the path has a local preference to go through the malicious IISP, then advertising a shorter path will not help. For example, if R6 in AS6 decides to send traffic through the malicious IISP, AS3, because of local preference, then the traffic will be blackholed. Also, if the only path to a destination must pass through the malicious IISP as the case for router R10 in AS10, then the traffic



will be blackholed when destined to the prefix of AS1. This is because the traffic must pass through the malicious IISP, AS3.

#### **4.2.1.2.2      More Specific Announcement**

When forwarding traffic, the destination of the traffic will be matched to the longest match of the prefixes in the routing table. In this way, advertising a more specific prefix through the good IISP will make all the routers select to reach the destination through the good IISP. Figure 4-3 depicts this technique.

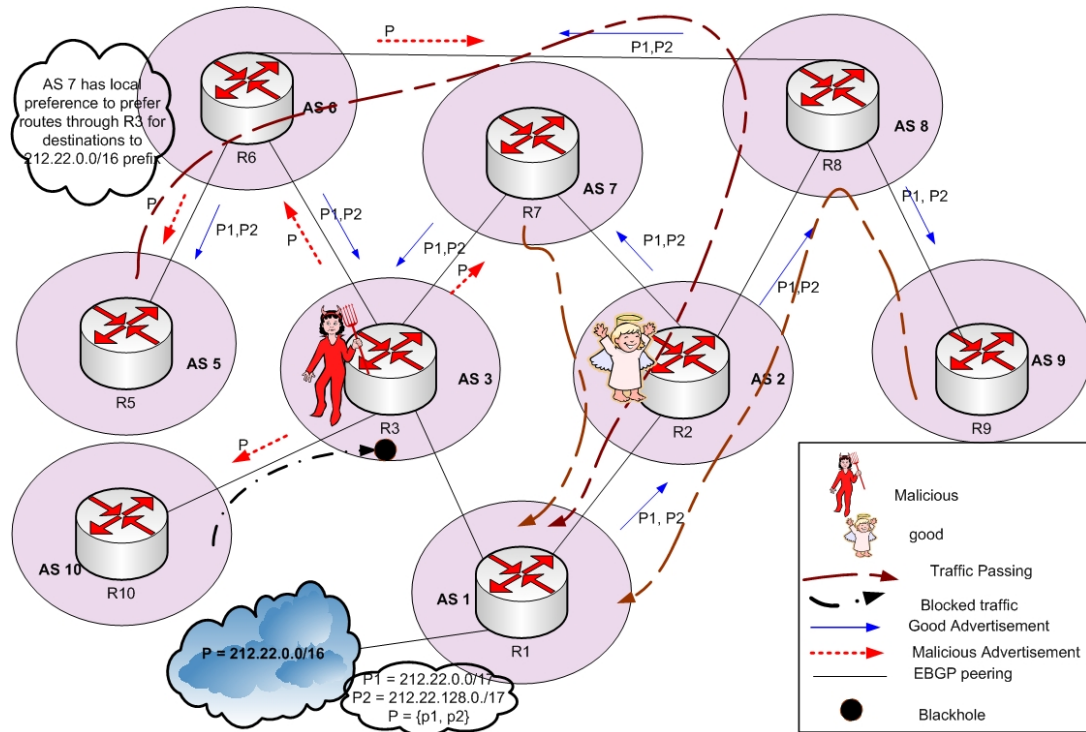


Figure 4-3 More Specific Prefixes

As shown in Figure 4-3, even if a path has a higher `local-pref` for routes that are destined to a certain prefix, a router will select the path with the longest prefix match even before it performs the selection process. For example, AS6 has a higher local preference to routes received from the malicious IISP AS3. However, because more specific prefixes are advertised to AS6 from AS8, the traffic will select the path AS6 AS8 AS2 AS1. Therefore, it will pass through the good IISP. Of course, if the only provider for

an AS is the malicious IISP, then the route must go through it and it will be blackholed like the case for router R10 in AS10.

#### **4.2.1.2.3      Communities**

The third scheme to direct the incoming traffic through the good IISP is through the use of communities. An AS can advertise a path and assign a certain community number to it. A route map condition can then be set in some of the ASes in between to assign a higher local preference for routes with the community number assigned to the advertised path. As a result, these ASes will prefer the paths through the good IISP. Figure 4-4 shows the use of community to control the traffic.



this community number. If some ASes select to direct the traffic through the malicious AS because it is the only way to reach the destination, then the traffic will be blocked since it will face a blackhole as in the case of router R10 in AS10.

### ***4.2.2 Virtual Peering***

Quoitin [4] proposed the use of virtual peering to deterministically control the incoming traffic through one of the providers. He used virtual peering to achieve load balancing of incoming traffic among providers and to reduce the latency by choosing paths that have the lowest delay. Since this method controls incoming traffic, we adapted it in order to direct the incoming traffic through the non-malicious IISP. The main contribution of [4] is the automation of setting up the virtual peering by using a virtual peering controller (VPC) that manages virtual peering establishment and removal. Figure 4-5 depicts the use of virtual peering to solve the malicious IISP blocking.

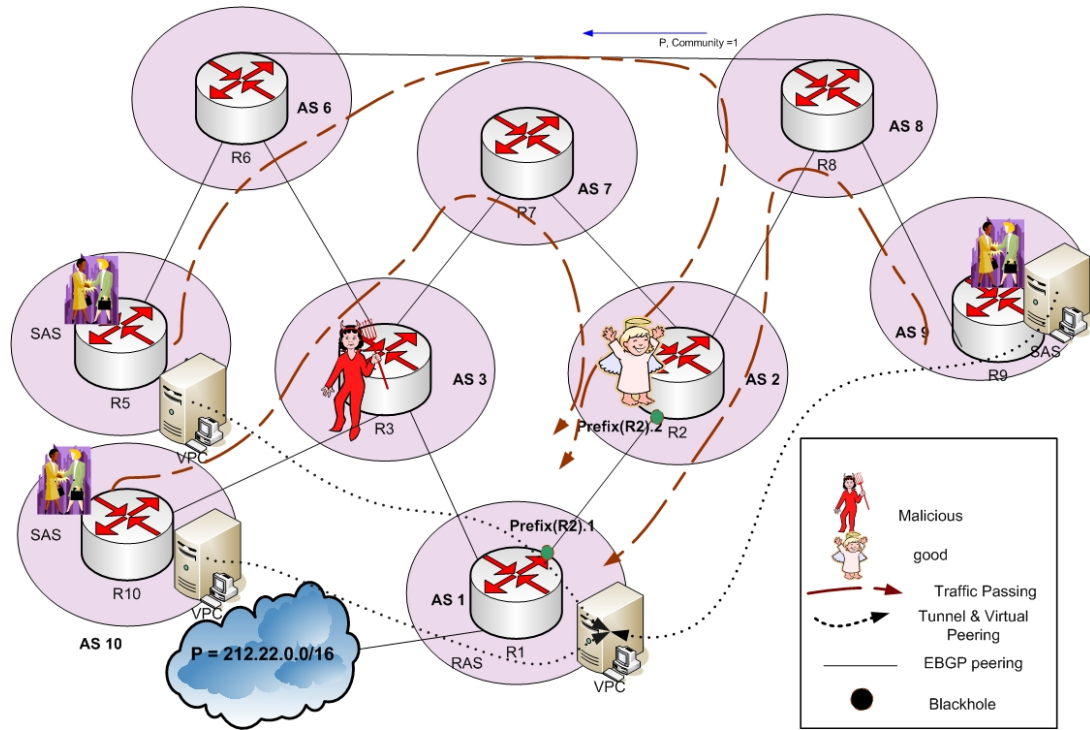


Figure 4-5 Virtual Peering based Solution

The requestor autonomous system (RAS), which is the destination ISP, requests the establishment and removal of virtual peering connections. The source autonomous system (SAS) originates the traffic destined to RAS. The VPC in RAS needs to know the IP address of the virtual peering controller (VPC) in SAS. This can be done either manually or automatically by including the IP address of the VPC in the BGP update message as an extended community. When the RAS knows the IP address, it can establish a multi-hop BGP session with the VPC in the SAS. It can then send a Virtual Peering Establishment

(VPE) or a Virtual Peering Removal (VPR) to the SAS. After that, one of the border routers in the SAS will establish a tunnel with one of the border routers of the RAS so that the IP address used as destination address belongs to a prefix owned by the provider AS, which is the good IISP for our case. Knowing that the ISP assigns one of its IP addresses to the connection between the border routers, this IP address is used as a destination to the tunnel. Hence, the malicious router will not be able to figure out that this IP address belongs to the blocked prefix. The router can then decapsulate the traffic and send it to the destination of interest. In order to force routes to go through the tunnel, a higher `local-pref` is assigned to them. One thing to note is that even if the traffic passes through the malicious IISP, the traffic will not be blocked because it will be destined to an IP not belonging to the prefix of interest.

The multi-hop BGP connection should be established such that the traffic of BGP messages does not pass through the malicious IISP. If they pass through the malicious ISP, a BGP connection will not be established, since the BGP messages will be blackholed. Therefore, the choice of the IP address of the VPC in a virtual peering must be the same as the one used to set up the tunnel. Therefore, virtual peering shall be adapted such that the VPC is the same router that is used for establishing the tunnel to force traffic to pass through the good IISP.

### ***4.2.3 Virtual Transit (Hijack the Hijacker)***

In virtual peering, every source of traffic must do virtual peering in order to control all traffic coming from or going to the Internet. This is not practical for our solution, so a number of modifications of virtual peering can be done to make the solution scalable. We refer to this as a virtual transit solution. The main difference is that the transit router will advertise the prefix to other routers in the Internet. Another difference is that the AS Path can be set as a path of length two: the transit AS and the originating (or provider) AS. A number of virtual transit routers may be distributed in the Internet attracting the traffic destined to the prefix of interest and then passing it to the destination in a tunnel established in the same way as it is done with virtual peering. The concept of virtual transit is illustrated in Figure 4-6.



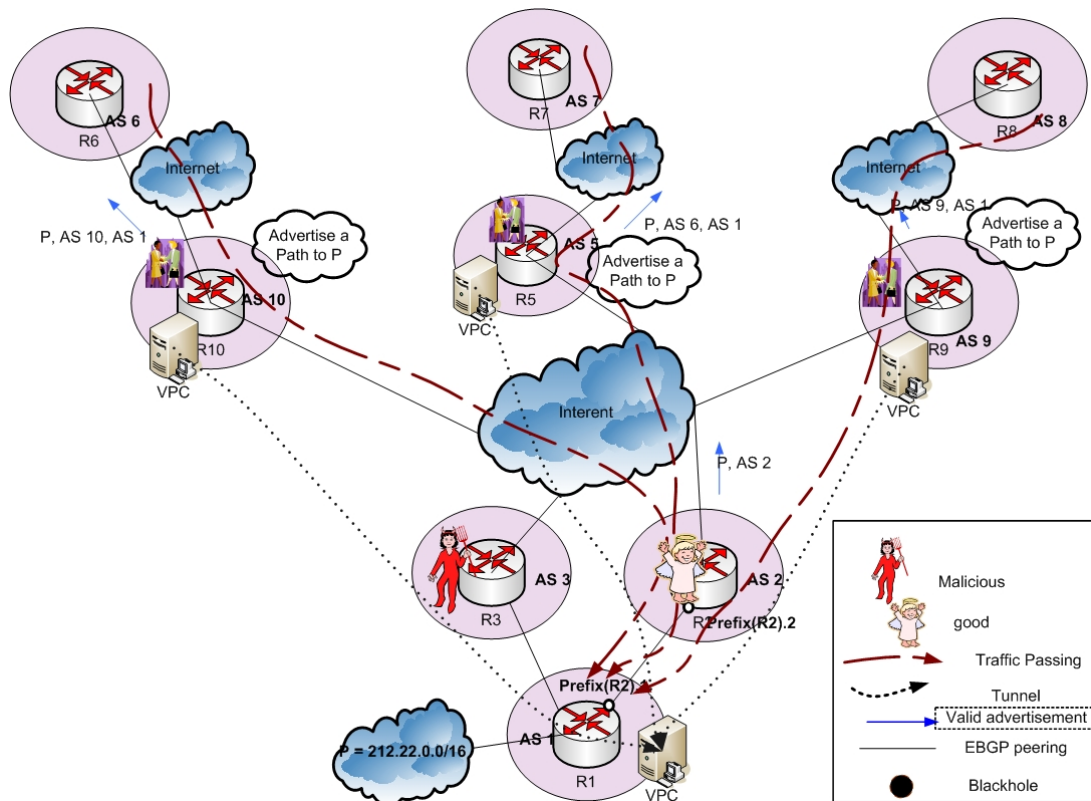


Figure 4-6 Virtual Transit

As shown in Figure 4-6, the VPC in AS10 is a cooperative router. This router establishes a virtual transit connection with the VPC in AS1 in the same way virtual peering is established. However, it advertises to other prefixes in the Internet that it has a path to R1, and in order to influence more traffic, it shortens the AS-Path to two. The philosophy behind this is that since a tunnel is established between AS10 and AS1, then

the path will consider only these ASes in its advertisement. It may also advertise more specific prefixes in order to attract the maximum traffic.

The idea of virtual transit can also be called ‘hijack the hijacker’. To clarify this, we can see that R3, a seemingly legitimate provider, is advertising a path to the local region to attract traffic with the ill intention of blackholing the traffic. The advertised path is superficially valid because AS3 is a provider for the local region and is geographically close. The Internet Routing Registry (IRR), a distributed database maintaining routing information and priorities, may even list AS3 as the legitimate ISP for AS1. On the other hand, the routers R10, R5, and R9, though are not obliged to advertise paths leading to the local region, they are doing so even with their distant geographical location and lack of physical connection to the local region. These routers (i.e. R10, R5, and R9) are cooperating with the local region to provide connectivity despite the malicious activity of R3. It is worth mentioning that with some implementation of secure BGP, this incongruity may identify the cooperative routers as hijackers, while it accepts the advertisements by the malicious router. Therefore one may consider this proposed mechanism as an ethical hijacking of the hijacking provider in order to counteract its malicious actions and gain access to the Internet.

### ***4.3 ANALYSIS AND DISCUSSION***

The use of local preference to control the outgoing traffic is good for most cases except for the case when the route must pass through the malicious provider. A tunnel might be used in order to hide the identity in this case and complete the solution. The use of AS Path shortening can attract traffic to the good IISP. However, major flows of the traffic will probably not be attracted. Examples include those routers that have a local preference to prefer the malicious route, those routers that can still see the path through the malicious route as a shorter path depending on their location, and those routers where the malicious router is the only provider. The use of more specific prefixes can attract more traffic through the good IISP. There is a limit, however, on the length of the prefix, and those that exceed this prefix can be filtered out by routers [4]. In addition, a disadvantage is that traffic will be blackholed when the only path to the destination must pass through the malicious router.

Virtual peering combines the benefits of traffic engineering techniques and hiding one's identity. Moreover, it provides a deterministic technique for the traffic to be forwarded to its destination through a specific provider. However, it requires cooperation between the sources of traffic and the destination. This means that every

source of traffic must do virtual peering with the destination in order to reach the destination through the good IISP. The rationale behind having a virtual transit is to make a limited number of virtual transit routers that advertise a prefix in a short AS path. These virtual transit routers will attract almost all of the Internet traffic if they are distributed around the globe and a sufficient number of them are installed. However, using a virtual transit does not completely guarantee that all the traffic will pass through the good IISP and will not be filtered, e.g., if a router has the malicious IISP as its only direct provider, then its traffic will be filtered out.

In Table 4-1, we provide a comparison amongst all the methods considered in this study in terms of the following criteria: traffic filtering, setup overhead, communication overhead, difficulty to combat the method, and scalability. Each of these criteria is discussed in the next few paragraphs.

Traffic filtering refers to the amount of the traffic that is expected to be filtered out (blackholed) using the methods listed in the columns. The method that leads to no filtering of traffic is virtual peering because a tunnel needs to be established to an IP address that does not belong to the blocked prefix between every source of traffic and the destination. This tunnel hides the identity in case the traffic flows pass through the

malicious IISP. Almost all of the traffic will not be filtered in the case of virtual transit because most likely the traffic will be attracted by one of the distributed cooperative routers that establish a tunnel to the affected routers. A very small amount might be filtered if one source of traffic is attracted by the malicious ISP because it might be its only provider. In BGP tuning, if it happens that the traffic needs to pass through the malicious router, then it will be filtered out.

The second criteria in Table 4-1 is the setup overhead which is a measure of the time needed and the difficulty level faced in order to get all the required configurations performed to execute the method. The setup overhead is high in virtual peering and virtual transit, with respect to BGP tuning. This is because of the establishment and removal mechanisms of a virtual peer and virtual transit connections. This overhead is, relatively, medium for communities because configurations on cooperative routers are still needed. The setup overhead for AS-path shortening and more specific prefixes is very small since the configuration is needed only on one (the RISP) or two (the RISP and the good IISP) routers.

In contrast, the communication overhead refers to the number of messages that need to be exchanged between routers before the method is effective. Only virtual

peering and virtual transit have some communication overhead because of the establishment of the multi-hop BGP connection. For BGP tuning based methods, there are no external exchanged messages between routers other than those that are part of normal BGP conversations.

The difficulty to combat the method is a measure of the amount of effort required by the malicious IISP to overcome the solution. One obvious disadvantage of BGP tuning based techniques is that the malicious IISP can easily mimic the tuning implemented by the local region to neutralize all the benefits gained. For example, it can shorten the AS path to be more preferred in the selection process. It can also advertise more specific prefixes so it can gain advantage of the longest prefix match. Moreover, it can advertise routes with the same community number advertised by the good IISP to combat the advantage of community. Although the malicious IISP can also make virtual peering and virtual transit, it is however very difficult to know that the traffic is destined to the blocked prefix and that this technique is used, since that traffic is apparently not sent to the prefix that is hijacked.

Finally, in terms of scalability which refers to the easiness of extending the method or using it for the entire Internet, BGP tuning techniques provide the most scalability since

the configuration will affect the decision taken by the traffic in the Internet without any needed connection. In the case of virtual peering, if the blocked local region wants to direct all the traffic of the Internet to the destination through the good IISP, then a virtual peer needs to be established between all the sources of traffic and the destination. Virtual transit dramatically reduces the number of needed established tunnels while having almost all of the traffic of the Internet directed to the destination through the good IISP.

**Table 4-1 Comparison between Methods.**

	BGP Tuning			Virtual Peering	Virtual Transit
	AS-Path Shortening	More Specific Prefixes	Communities		
<b>Filtering the traffic</b>	Medium	Small	Small	No	Very Small
<b>Setup overhead</b>	Small	Small	Medium	High	High
<b>Communication overhead</b>	No	No	No	Medium	Medium
<b>Difficulty to combat the method</b>	Easy	Easy	Easy	Difficult	Difficult
<b>Scalability</b>	High	High	High	Small	High

#### ***4.4 SUMMARY***

In this chapter, three techniques to overcome the problem of Internet embargo imposed by a malicious international Internet service provider (IISP) are discussed. These techniques are BGP tuning, virtual peering, and virtual transit techniques.



---

## ***CHAPTER 5***

### ***DESIGN, IMPLEMENTATION & VALIDATION OF SOLUTION***

### ***USING BGP TUNING BASED APPROACH***

#### ***5.1 INTRODUCTION***

In the previous chapter, a qualitative analysis of methods to circumvent malicious IISP blackholing the local region's traffic while advertising reachability information to it and to the rest of the Internet is presented. In this chapter, the concentration is on the implementation of the methods that use BGP tuning to control outgoing and incoming traffic using OPNET.

OPNET [40] is a commercial tool used to design, test, and simulate the performance of a specific network. OPNET has an implementation of almost all the well known protocols. OPNET version 14.5 PL3 [41] provides BGP support and configuration. A user can configure autonomous systems and EBGP and IBGP connections. He can also configure policies to change the route attributes to control incoming and outgoing traffic. These policies include the setting of the local preference (`local-pref`), the use

of community, and the prepending of AS-Path. However, OPNET in its implementation of BGP does not allow reconfiguring the BGP in the middle of the simulation. In addition, AS-Path shortening and configuring more specific prefixes methods for controlling incoming traffic are not supported. Moreover, configuring a router to be malicious is not implemented by OPNET.

The material in this chapter is presented as follows. First, the baseline configuration to serve as reference our work using BGP simulation is shown. Next, the use of traffic control using BGP basic implementation is presented. Finally and most importantly, our changes to BGP in order to be able to simulate the real scenarios of having a malicious node, to be able to do the changes in the middle of the simulation and support, and to be able to configure shortening and more specific prefixes are discussed.

## ***5.2 INITIAL SETUP***

In order to control the outgoing and incoming traffic and to validate our modification, we need to have a network setup that is able to validate the implementation. Aboelela book [42] has a simple experiment about BGP. The experiment is modified to match the needs of our simulation. Figure 5-1 shows the baseline configuration of the experiments done afterward.

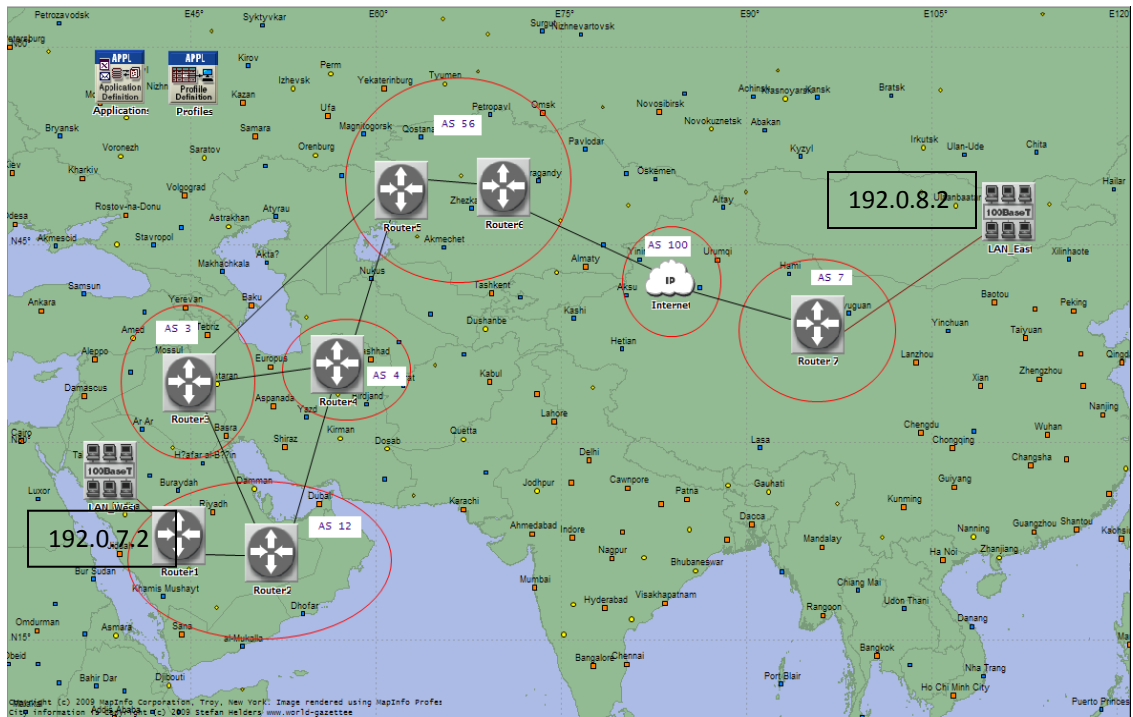


Figure 5-1 Baseline configuration of validation

### 5.2.1 Devices Used

**ethernet4\_slip8\_gtwy** router: is a gateway router that supports IP, UDP, RIP, Ethernet (IEEE 802.3), OSPF and SLIP protocols. Ethernet and other protocols. It has 4 Ethernet 10BaseT/100BaseT connections and 2 serial Line IP connections at selectable data rates. [43]

The baseline configuration consists of seven routers are labeled from Router 1 to Router 7 are used. These routers are configured to support the BGP protocol. IBGP protocol is configured within each autonomous system and EBGP protocol is configured between routers in different ASes. An ellipse is drawn in Figure 5-1 to represent an autonomous system.

**100BaseT\_LAN** object: is used to model a Fast Ethernet LAN which has any number of clients and one server. This object can support a number of applications including FTP, Email, Databases etc. One can also configure the switching speed and the number of workstations. [43]

In our baseline simulation, we use two objects named LAN\_East and LAN\_West. LAN\_West is configured to use all types of applications or to work as a server for any type of application. LAN\_East is configured to consider LAN\_West as its destination and its HTTP server, so it will use the HTTP application.

**ip32\_cloud** node: represents an Internet cloud and has 32 serial line interfaces (only two of them are used in our experiment). It supports the RIP, UDP, IP, OSPF, BGP, IGRP and TCP protocols. The 'Packet Latency' can be configured which specifies the time the incoming IP datagram spends until it is forwarded out through one of the interfaces. This

delay can be configured to take any probability distribution. Also, the “Packet Discard Ratio” can be configured which determines the percentage of traffic to be discarded.

[43]

This node is used to model the delay of the Internet to study its effect on the convergence time. In this chapter, we will just assume the delay to be 0.001 second, however in CHAPTER 6 different values will be considered.

Bidirectional **PPP\_DS3** link: used to connect nodes that run the IP protocol using ip3\_dgram packet format with DS3 (44.736 Mbps) data rate [43]. In our simulation, this type of link is used to connect routers with each other.

Bidirectional **100BaseT** link: is an Ethernet connection that operates at 100 Mbps. This link allows the configuration of “Propagation Speed” in meters/second. In case the “Delay” is set to ‘distance based’, the propagation delay is calculated from the speed [43]. This link is used in the simulation to connect the router to the LAN objects.

### ***5.2.2 Mapping Devices Usage to our Problem***

In our simulation, we assume that AS 12 which contains Router 1 and Router 2 is in the local region and LAN\_West contains the prefix of interest that is blocked. LAN\_East

communicates with LAN\_West to get services. Router 2 is the Regional provider and it has two International providers, i.e., Router 3 and Router 4. In the initial runs that uses OPNET supported features only, it is intended to get the traffic away of Router 3 and let it pass through Router 4. In the runs that include the added features and modified OPNET, Router 3 will be configured to be malicious and the traffic is directed away from the malicious router, and Router 4 will be considered the good ISP.

### ***5.3 CONTROLLING TRAFFIC USING SUPPORTED OPNET FEATURES***

As mentioned before, OPNET allows the configuration of routing policy to control the outgoing traffic and to influence traffic to come through one of the International providers if the local region is multihomed as the case in our initial setup. OPNET supports setting the value of the local preference of routes advertised from a neighboring router. This allows the administrator to set this value to control the outgoing traffic. In addition, OPNET allows modifying the AS-Path by prepending, to make the routes through a specific route less preferred. Also, the use of community is supported to allow the agreement with some remote routers to prefer routes with a certain community number. In this section, we will investigate the usage of these

configurations to control the traffic. No malicious router will be configured in this section.

### ***5.3.1 Normal Simulation Run***

The normal run of simulation means the result of the simulation without any special BGP configuration to control outgoing and incoming traffic. Figure 5-2 shows the traffic between Router 2 and Router 3 and between Router 2 and Router 4 in both directions.

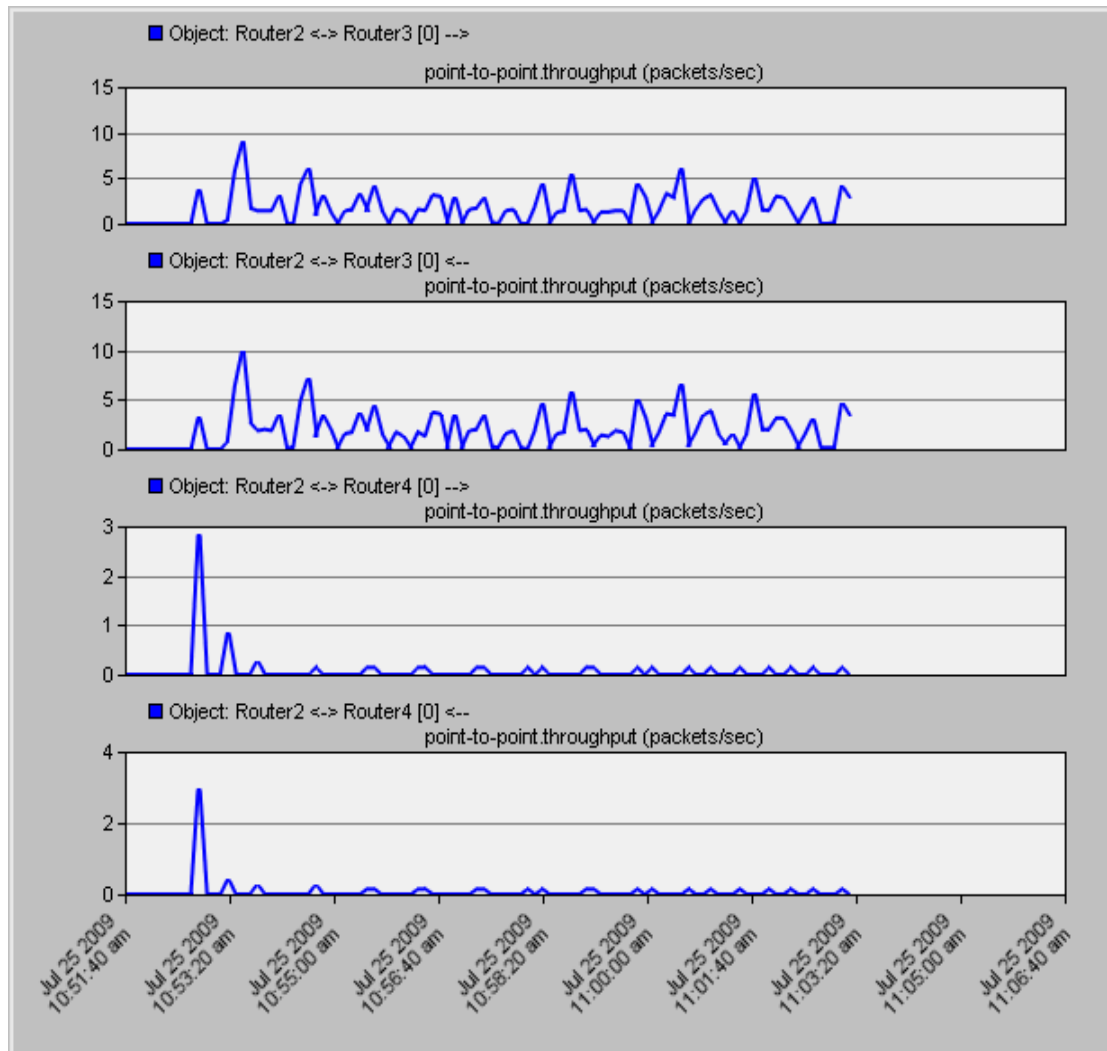


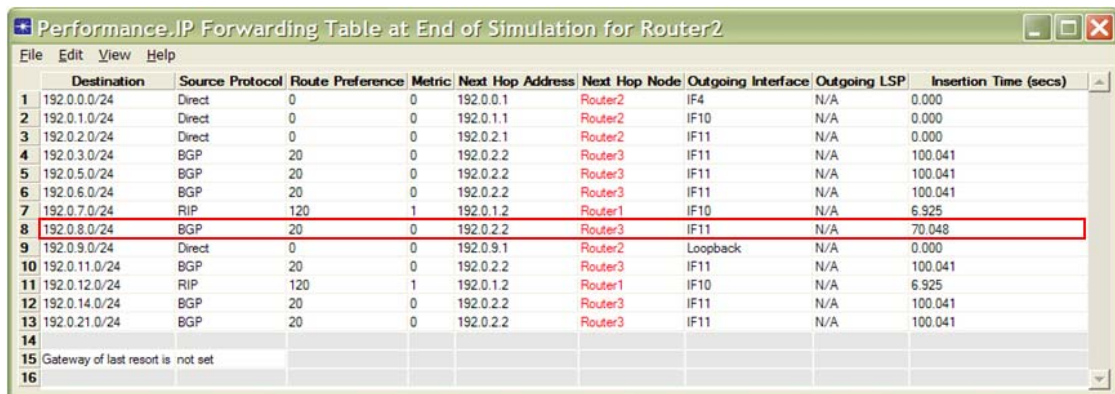
Figure 5-2 Incoming and Outgoing Traffic in Normal Mode

The x-axis represents the time in seconds, while the y-axis represents the throughput in packets/second. We notice that the traffic goes through Router 3 in both directions. To double check, we can see the IP forwarding table of Router 2 and Router 5. They both



determine the incoming and outgoing traffic of the local region, i.e., AS 12, as shown in Figure 5-1.

Figure 5-3 shows the IP forwarding table for router 2.



	Destination	Source Protocol	Route Preference	Metric	Next Hop Address	Next Hop Node	Outgoing Interface	Outgoing LSP	Insertion Time (secs)
1	192.0.0.0/24	Direct	0	0	192.0.0.1	Router2	IF4	N/A	0.000
2	192.0.1.0/24	Direct	0	0	192.0.1.1	Router2	IF10	N/A	0.000
3	192.0.2.0/24	Direct	0	0	192.0.2.1	Router2	IF11	N/A	0.000
4	192.0.3.0/24	BGP	20	0	192.0.2.2	Router3	IF11	N/A	100.041
5	192.0.5.0/24	BGP	20	0	192.0.2.2	Router3	IF11	N/A	100.041
6	192.0.6.0/24	BGP	20	0	192.0.2.2	Router3	IF11	N/A	100.041
7	192.0.7.0/24	RIP	120	1	192.0.1.2	Router1	IF10	N/A	6.925
8	192.0.8.0/24	BGP	20	0	192.0.2.2	Router3	IF11	N/A	70.048
9	192.0.9.0/24	Direct	0	0	192.0.9.1	Router2	Loopback	N/A	0.000
10	192.0.11.0/24	BGP	20	0	192.0.2.2	Router3	IF11	N/A	100.041
11	192.0.12.0/24	RIP	120	1	192.0.1.2	Router1	IF10	N/A	6.925
12	192.0.14.0/24	BGP	20	0	192.0.2.2	Router3	IF11	N/A	100.041
13	192.0.21.0/24	BGP	20	0	192.0.2.2	Router3	IF11	N/A	100.041
14									
15	Gateway of last resort is not set								
16									

Figure 5-3 IP forwarding table for Router 2.

The IP address of LAN\_EAST is 192.0.8.2 so it belongs to the prefix 192.0.8.0/24. We notice that the 'Next Hop Node' to this prefix is through router 3. We noticed also that the route is inserted in the IP forwarding table at 70.048 seconds which is 0.048 seconds after the start of BGP.

Figure 5-4 shows the IP forwarding table of router 5.

	Destination	Source Protocol	Route Preference	Metric	Next Hop Address	Next Hop Node	Outgoing Interface	Outgoing LSP	Insertion Time (secs)
1	192.0.0.0/24	BGP	20	0	192.0.3.1	Router3	IF10	N/A	70.043
2	192.0.1.0/24	BGP	20	0	192.0.3.1	Router3	IF10	N/A	70.043
3	192.0.2.0/24	BGP	20	0	192.0.3.1	Router3	IF10	N/A	70.043
4	192.0.3.0/24	Direct	0	0	192.0.3.2	Router5	IF10	N/A	0.000
5	192.0.5.0/24	Direct	0	0	192.0.5.1	Router5	IF4	N/A	0.000
6	192.0.6.0/24	Direct	0	0	192.0.6.1	Router5	IF11	N/A	0.000
7	192.0.7.0/24	BGP	20	0	192.0.3.1	Router3	IF10	N/A	70.043
8	192.0.8.0/24	IBGP	200	0	192.0.14.1	Router6	Unresolved	N/A	70.035
9	192.0.9.0/24	BGP	20	0	192.0.3.1	Router3	IF10	N/A	70.043
10	192.0.11.0/24	Direct	0	0	192.0.11.1	Router5	Loopback	N/A	0.000
11	192.0.12.0/24	BGP	20	0	192.0.3.1	Router3	IF10	N/A	70.043
12	192.0.14.0/24	RIP	120	1	192.0.5.2	Router6	IF4	N/A	8.582
13	192.0.21.0/24	RIP	120	1	192.0.5.2	Router6	IF4	N/A	8.582
14									
15	Gateway of last resort is not set								
16									

Figure 5-4 IP forwarding table of Router 5

The IP address for LAN\_WEST is 192.0.7.2 which is under the prefix 192.0.7.0/24. This entry in the table has its 'Next Hop Node' as Router 3 which is inserted at 70.043, seconds so it is at 0.052 seconds after the start of BGP protocol. The source protocol that it learned the route from is BGP.

Figure 5-5 shows the convergence activity and duration of the normal simulation scenario. The upper part of the figure (Network Convergence Activity) shows the value of 1 in the time when there is a convergence activity, otherwise the value is 0. The lower part of the figure (Network Convergence Duration) shows in the x-axis the time when there is the convergence started of a certain activity, and the y-axis shows the time it takes to converge.

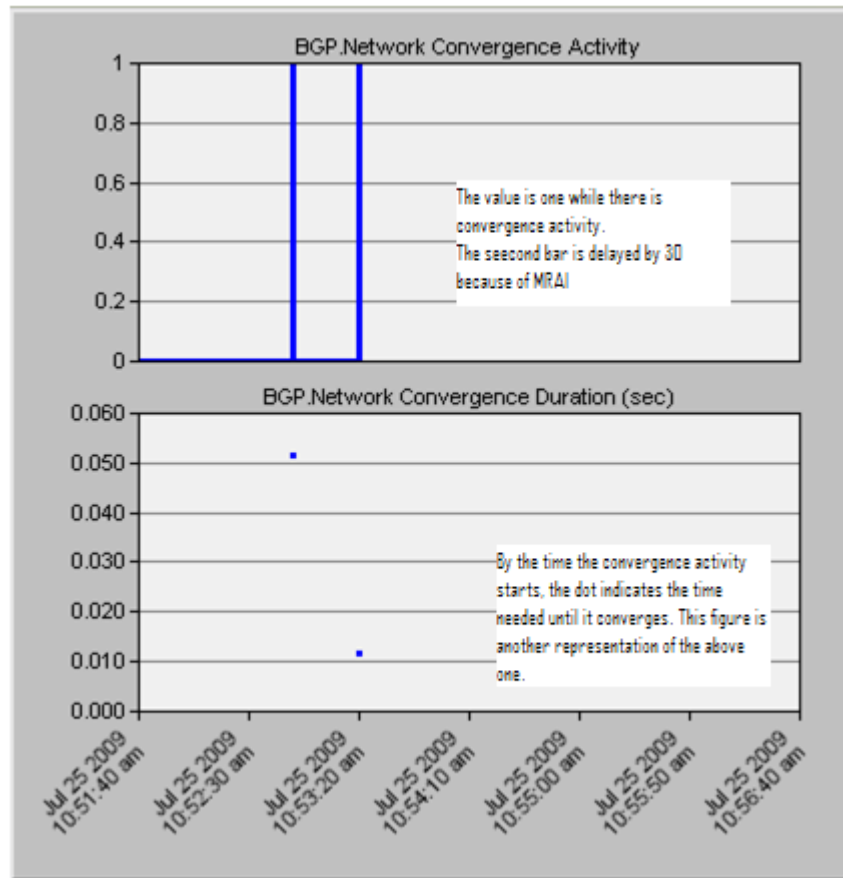


Figure 5-5 Convergence activity and duration of normal mode

Figure 5-5 shows that there are two convergence activities. The first one takes about 0.052 seconds and the second one takes about 0.012 seconds separated by 30 seconds time. The first one is due to the start of BGP protocol, while the second one is due to minimum route advertisement interval (MRAI) which is set to 30 seconds. The MRAI time is a BGP parameter that can be configured and 30 seconds is its default value. MRAI

delay is enforced in the following situation. First, one of the routers receives an advertisement of a route to a certain destination. Then the same router receives another announcement to the same destination. Moreover it happens that this route is better than the previously announced one according to BGP selection process. This router will not propagate the second route to the same routers it announced to it before until MRAI time pass.

### ***5.3.2 Control of Outgoing Traffic***

To control outgoing traffic, the route attribute `local-pref` can be set to a higher value for routes that are coming from a certain neighbor. This affects the BGP selection process to prefer routes through the intended ISP. This can be done by configuring a route map that states if a route is received accept it and assign a higher `local-pref` to it, for example, 150, which is higher than the default value, i.e., 100. Then the route map is assigned as a policy to specific neighbor, Router 4 in AS4 in this case, and in the direction of 'in'. In other words, Router 2 is configured to have this route map applied for each route learned from Router 4. Therefore, each route learned from Router 4 will have higher local preference and will be selected as best route according to BGP selection process.

Figure 5-6 shows the outgoing traffic from Router 2 to Router 3 and from Router 2 to Router 4, after applying the policy.

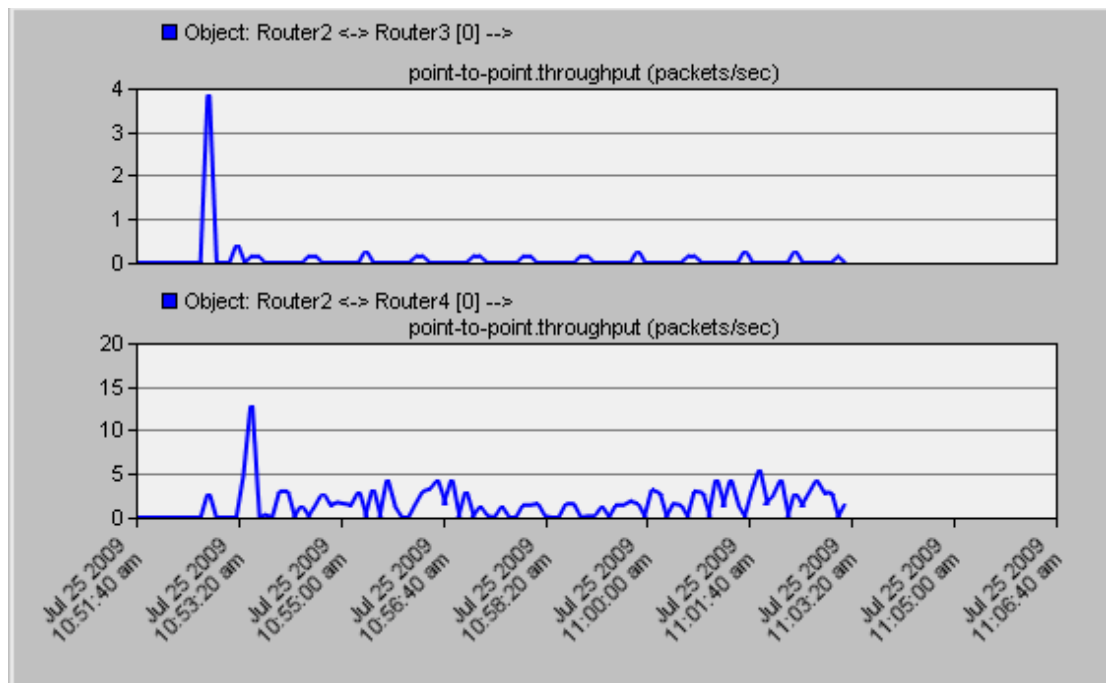


Figure 5-6 Higher local preference to routes advertised by Router 4.

As a result of applying a higher local preference to routes learned from Router 4, the outgoing traffic from Router 2 goes through Router 4. To check this, we show look at the BGP routing table of Router 2 which is shown in Figure 5-7.

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
2	192.0.1.0/24	Direct	192.0.1.1	Router2	IF10	0	100	32768		Incomplete		
3	192.0.2.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
4	192.0.3.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
5	192.0.5.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
6	192.0.6.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
7	192.0.7.0/24	RIP	192.0.1.2	Router1	IF10	1	100	32768		Incomplete		
8	192.0.8.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56 100 7	Incomplete		
9	192.0.9.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
10	192.0.11.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
11	192.0.12.0/24	RIP	192.0.1.2	Router1	IF10	1	100	32768		Incomplete		
12	192.0.14.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
13	192.0.21.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
14												

Figure 5-7 BGP routing table of Router 2 in Normal Local-Pref Experiment

As we can see the prefix 192.0.8.0/24 has its next hop node as Router 4. This is because the local preference is set to 150 which is more than the default value of local preference of 100.

Figure 5-8 shows the convergence activity and duration for this experiment. We notice similar behavior to the baseline experiment.

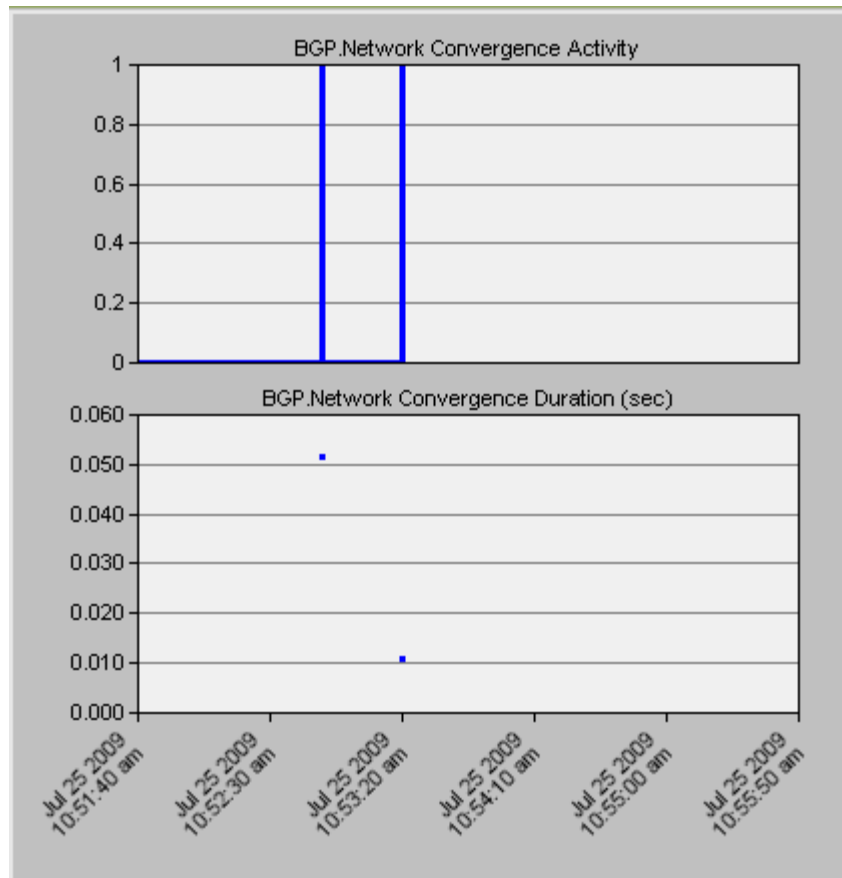


Figure 5-8 Convergence activity and duration of Normal Local-Pref Experiment

### 5.3.3 Controlling Incoming Traffic

Two ways will be considered in the control of incoming traffic using OPNET, namely, prepending and community. The next section shows the use of prepending, and the section that follows shows the use of community to control the incoming traffic.

### ***5.3.3.1    Use of Prepending***

Prepending is to increase the length of an AS-Path; this is usually followed by advertising the path through a specific neighbor in order to control incoming traffic to prefer the routes learned through other neighboring providers. A router may be configured to lengthen the AS-Path by adding its own AS-Number to the AS-Path more than one time before advertising it to one of the neighbors. In this case, the AS-Path becomes longer making the route less preferred. So to make the traffic less preferred through router 3, a prepended advertisement is sent to AS 3 and a normal advertisement is sent to AS 4.

A route map is created that prepends the AS-Path by the same autonomous system (AS12), and then the route map is applied to the neighbor residing in AS3 in the 'out' direction. Figure 5-9 shows the incoming traffic to router 2 from both router 3 and router 4 after applying prepending.



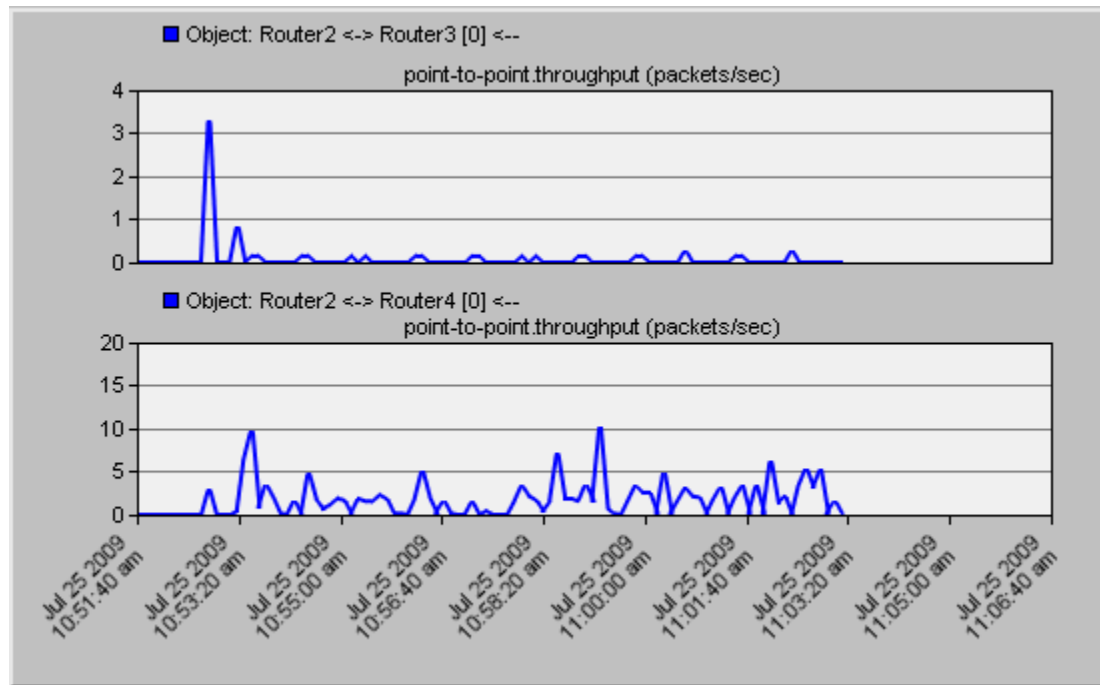
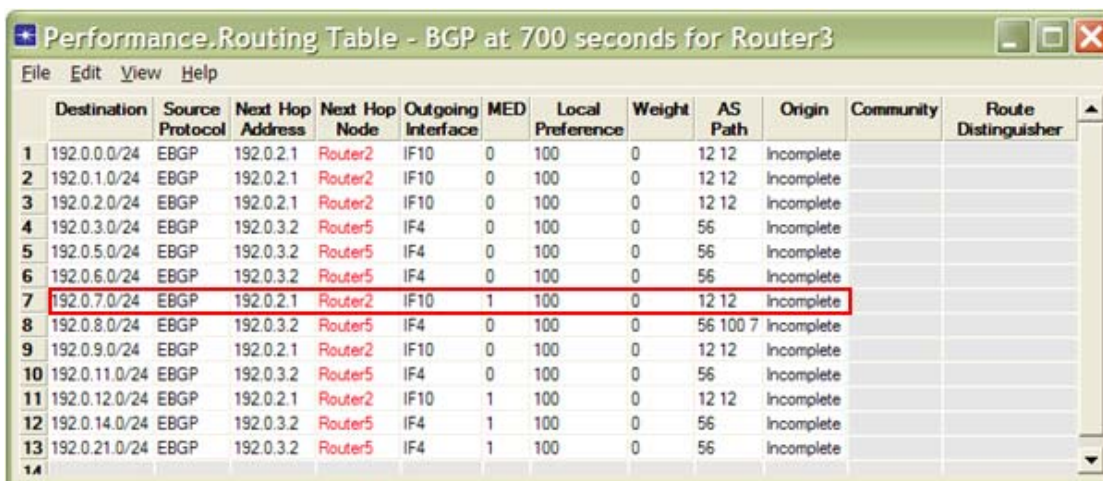


Figure 5-9 Incoming traffic to Router 2 using prepending

Figure 5-9 shows that the traffic travels to Router 2 through Router 4 in AS4. The BGP routing table of Router 3, presented in Figure 5-10, shows how the route is prepended through it. It is shown for the prefix 192.0.7.0/24, which contains the IP address of the LAN\_West, has AS-Path [12 12] of length 2. This makes the route less preferred when advertised to Router 5.

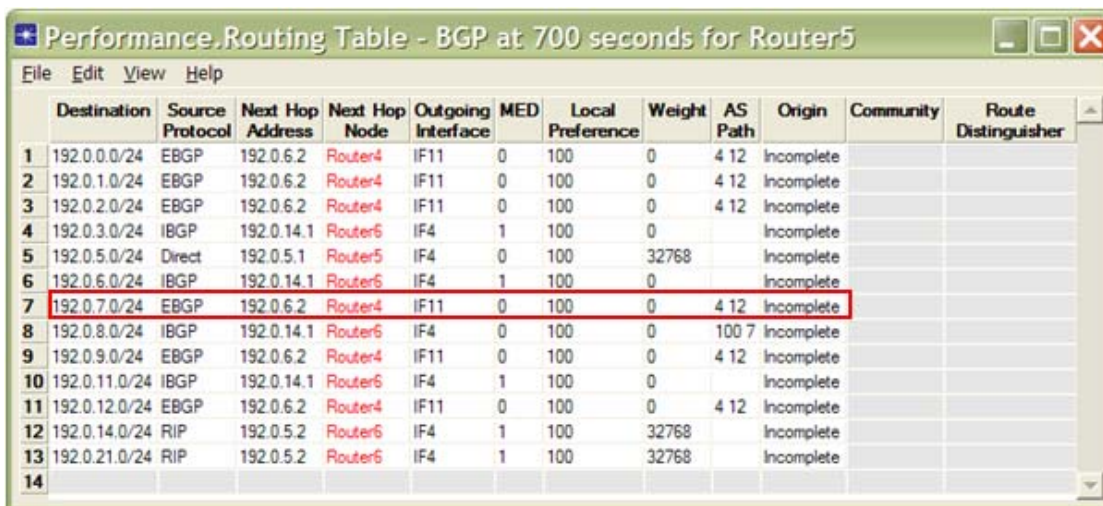


Performance Routing Table - BGP at 700 seconds for Router3

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.2.1	Router2	IF10	0	100	0	12 12	Incomplete		
2	192.0.1.0/24	EBGP	192.0.2.1	Router2	IF10	0	100	0	12 12	Incomplete		
3	192.0.2.0/24	EBGP	192.0.2.1	Router2	IF10	0	100	0	12 12	Incomplete		
4	192.0.3.0/24	EBGP	192.0.3.2	Router5	IF4	0	100	0	56	Incomplete		
5	192.0.5.0/24	EBGP	192.0.3.2	Router5	IF4	0	100	0	56	Incomplete		
6	192.0.6.0/24	EBGP	192.0.3.2	Router5	IF4	0	100	0	56	Incomplete		
7	192.0.7.0/24	EBGP	192.0.2.1	Router2	IF10	1	100	0	12 12	Incomplete		
8	192.0.8.0/24	EBGP	192.0.3.2	Router5	IF4	0	100	0	56 100 7	Incomplete		
9	192.0.9.0/24	EBGP	192.0.2.1	Router2	IF10	0	100	0	12 12	Incomplete		
10	192.0.11.0/24	EBGP	192.0.3.2	Router5	IF4	0	100	0	56	Incomplete		
11	192.0.12.0/24	EBGP	192.0.2.1	Router2	IF10	1	100	0	12 12	Incomplete		
12	192.0.14.0/24	EBGP	192.0.3.2	Router5	IF4	1	100	0	56	Incomplete		
13	192.0.21.0/24	EBGP	192.0.3.2	Router5	IF4	1	100	0	56	Incomplete		

Figure 5-10 BGP routing table of Router3 in Prepend Experiment

Figure 5-11 shows the BGP routing table for Router 5.



Performance Routing Table - BGP at 700 seconds for Router5

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4 12	Incomplete		
2	192.0.1.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4 12	Incomplete		
3	192.0.2.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4 12	Incomplete		
4	192.0.3.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
5	192.0.5.0/24	Direct	192.0.5.1	Router5	IF4	0	100	32768		Incomplete		
6	192.0.6.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
7	192.0.7.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4 12	Incomplete		
8	192.0.8.0/24	IBGP	192.0.14.1	Router6	IF4	0	100	0	100 7	Incomplete		
9	192.0.9.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4 12	Incomplete		
10	192.0.11.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
11	192.0.12.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4 12	Incomplete		
12	192.0.14.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
13	192.0.21.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		

Figure 5-11 BGP table of Router 5 in Prepend Experiment

As shown in Figure 5-11, Router 5 selected Router 4 as Next Hop Node to the prefix 192.0.7.0/24 because it has a shorter AS-Path [4 12] than through Router 3 which has the AS-Path [3 12 12].

Figure 5-12 shows the convergence activity for this network. The behavior is similar to the baseline experiment.

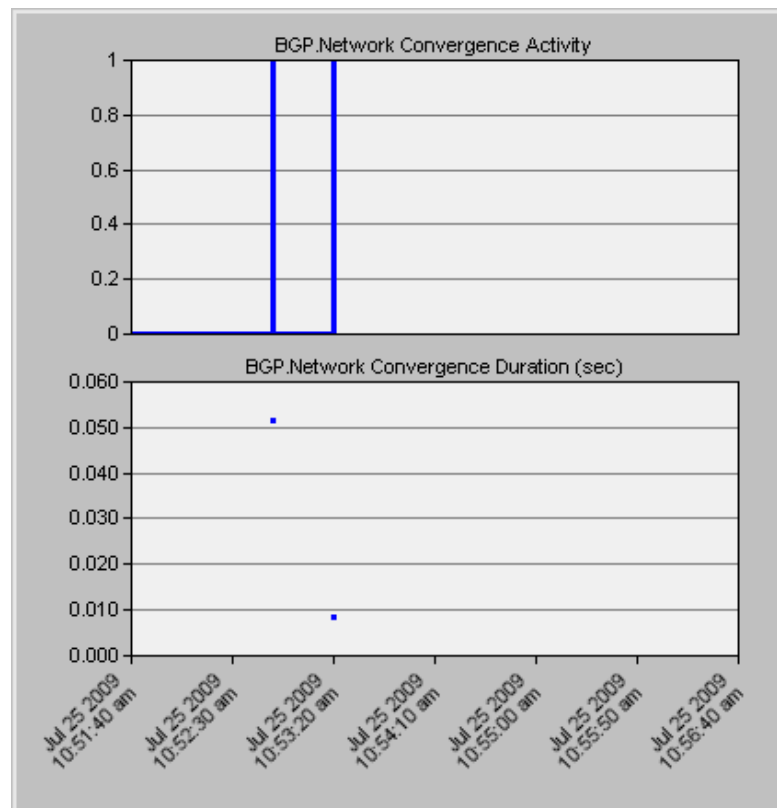


Figure 5-12 Convergence activity and duration for normal prepending experiment

### ***5.3.4 Use of Community***

Another way of controlling incoming traffic can be done through the use of community. Community is an attribute of each route that can be received or advertised. The community attribute can have a list of community numbers. These community numbers can be interpreted by other routers and apply certain action for certain communities.

Router 5 can be configured as follows. If it receives an advertisement with a specific community number, then it will set a higher local preference for that route. A route map can be configured in Router 2 to set every route to community 12:144 (this is one way of representing the community number, i.e., AS number followed by a colon then the community number in that AS), and then apply this route map for every route advertised to router 4. Router 5 can then check for any advertisement if it contains the same community number then it will assign a higher local preference to it, e.g., 150.

Figure 5-13 shows the throughput of incoming traffic to router 2 from router 3 and from router 4.

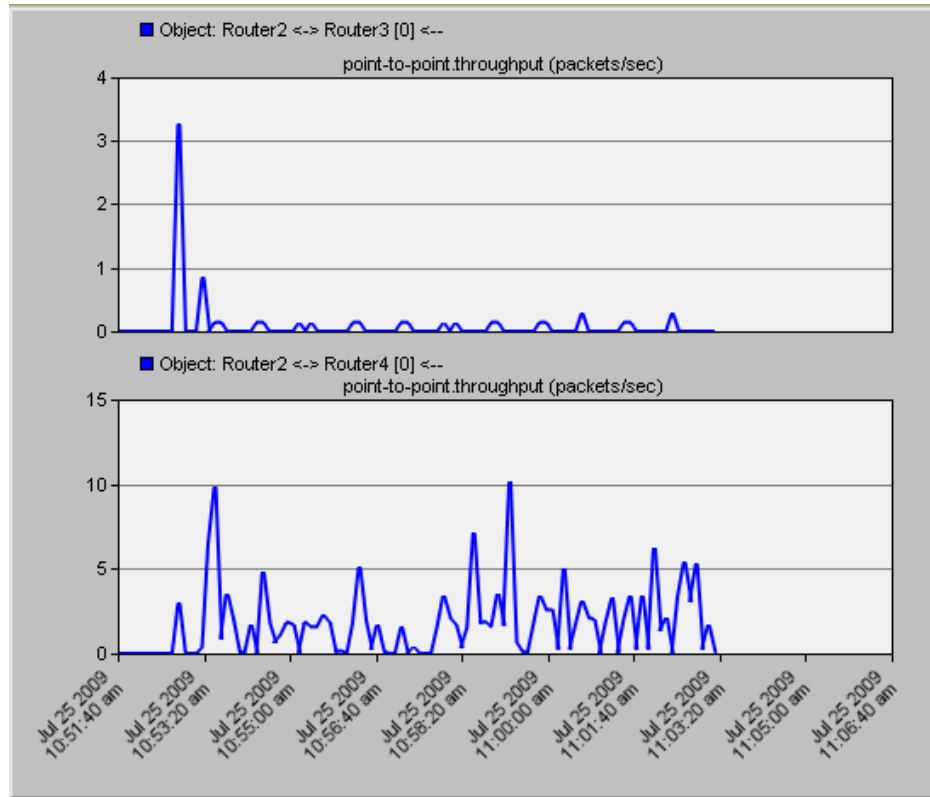


Figure 5-13 Incoming traffic to router 2 using community

Figure 5-13 shows a behavior of the traffic as in the case of prepending. Traffic to the prefix owned by Router 2 chooses the path through Router 4. To see how the community works, Figure 5-14 shows the BGP routing table of router 5.

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
2	192.0.1.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
3	192.0.2.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
4	192.0.3.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
5	192.0.5.0/24	Direct	192.0.5.1	Router5	IF4	0	100	32768		Incomplete		
6	192.0.6.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
7	192.0.7.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
8	192.0.8.0/24	IBGP	192.0.14.1	Router6	IF4	0	100	0	100 7	Incomplete		
9	192.0.9.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
10	192.0.11.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
11	192.0.12.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
12	192.0.14.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
13	192.0.21.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
14												

Figure 5-14 BGP routing table of router 5 in the community experiment.

It can be seen that all the routes whose community list contains 12:144 have their local preference is set to 150. For example, the prefix 192.0.7.0/24 whose community list is set to [12:144], has a local preference set to 150 by Router 5. A route map is defined in Router 5 as the following: if the route has the community number 12:144 in its community list, then it assigns the value 150 to its local preference. As a result, the route through Router 4 is preferred.

Figure 5-15 shows the convergence activity in this network which depicts similar behavior to the baseline experiment.

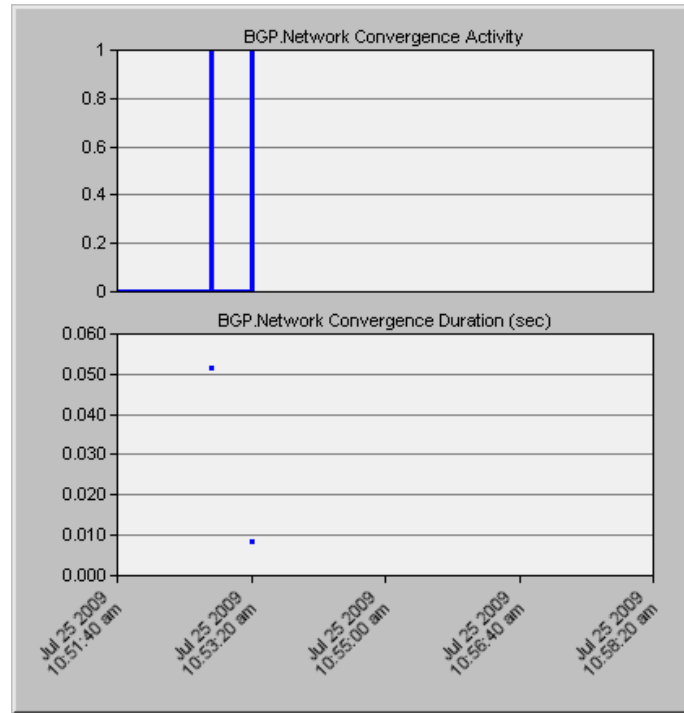


Figure 5-15 Convergence Activity and Duration of Community Experiment

## 5.4 MODIFICATION OF OPNET IMPLEMENTATION

OPNET Implementation does not allow making reconfiguration in the middle of the simulation. The only way to trigger BGP activity reconfiguration in the middle of simulation can be done through failing a node or a link in a specified time. So we want to be able to specify a time at which the malicious ISP starts blackholing traffic and specify a time for applying counter measures or reconfiguring BGP in the middle of simulation.

OPNET does not allow BGP reconfiguration during simulation, and the configuration is only limited to the start of simulation.

The following sections are presented as follows. First, a very short summary of the OPNET state model is presented. Then, we show our implementation of the malicious router. In that section, we include a short overview of part of the IP protocol implementation in OPNET, because it is needed in implementing the malicious router. After that, our modification to support reconfiguration is presented. The modification is introduced by a summary of the BGP protocol, because it is important to comprehend the modification of BGP protocol to do reconfiguration. Then, a section is presented to show the modification needed for the control of outgoing traffic. The section that follows, shows our modification to control the incoming traffic and that includes the use of community, shortening, and more specific prefixes. Since the implementation of shortening and more specific prefixes is not supported by OPNET, we show how we implemented them. In each section, a validation using the same Network setup shown in Figure 5-1 will be presented.



### ***5.4.1 OPNET Process Model***

A process model is drawn in OPNET by a set of states representing a finite state machine of the protocol. The states can be forced or unforced (Blocking). The forced states are represented by a lighter gray level if black and white printing is used or a green color if colored printing is used. When the forced state is triggered, all the statements inside it are executed. Conditions to move to other states are directly evaluated and then execution moves to the following state. This is not the case when an unforced or blocking state is used. Unforced states are represented by a darker gray level when black and white printing is used or a red color if colored printing is used. When a state enters a blocking state, it just executes the “enter” executive of the state and then blocks or waits until a new interrupt or event occurs; and then it evaluates the conditions and decides what the next state is.

### ***5.4.2 Building a Malicious Router***

In this section, a quick description of the IP protocol implementation is shown. Then, we demonstrate the needed changes to build the malicious router.

### 5.4.2.1 Summary of IP implementation

The IP protocol in OPNET consists of a number of processes. The root process is ip\_dispatch which initializes all variables and creates the needed children processes.

Figure 5-16 shows the state diagram of ip\_dispatch process.

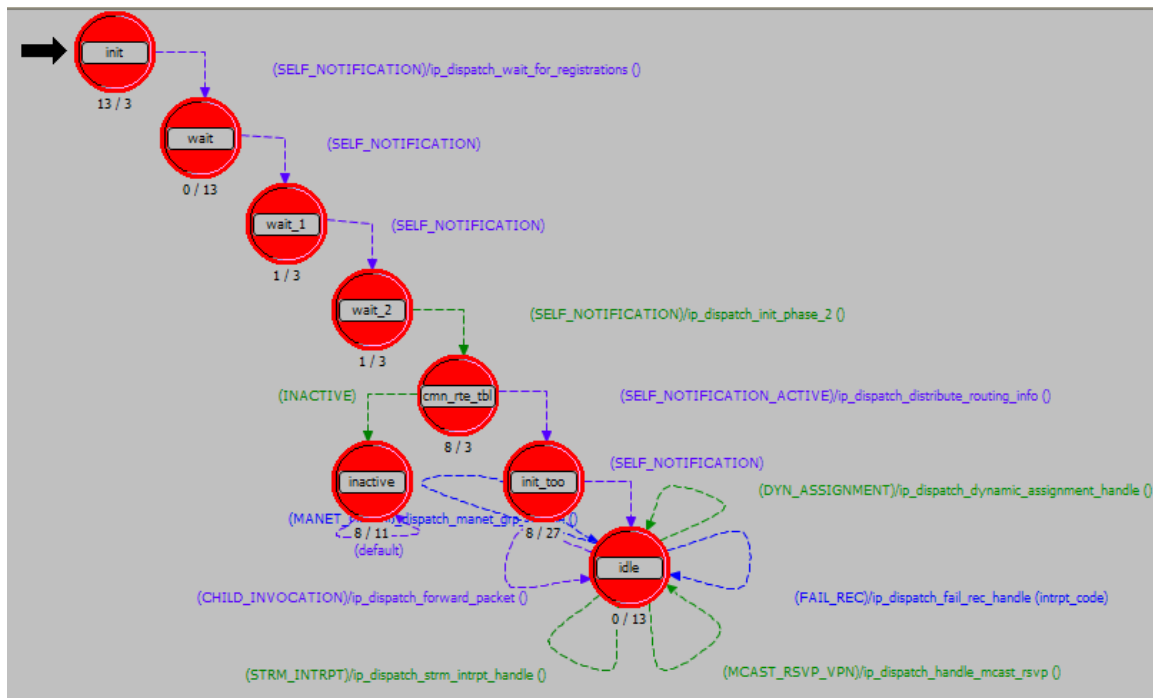


Figure 5-16 ip\_dispatch process model.

One of the children processes that is responsible for routing is the ip\_rte\_central\_cpu process which has only one server and only one queue for all

packets. In our modification, we assume<sup>2</sup> that this process is used for routing. We also assume the packet format is IPv4. Figure 5-17 shows the states of the `ip_rte_central_cpu` process model.

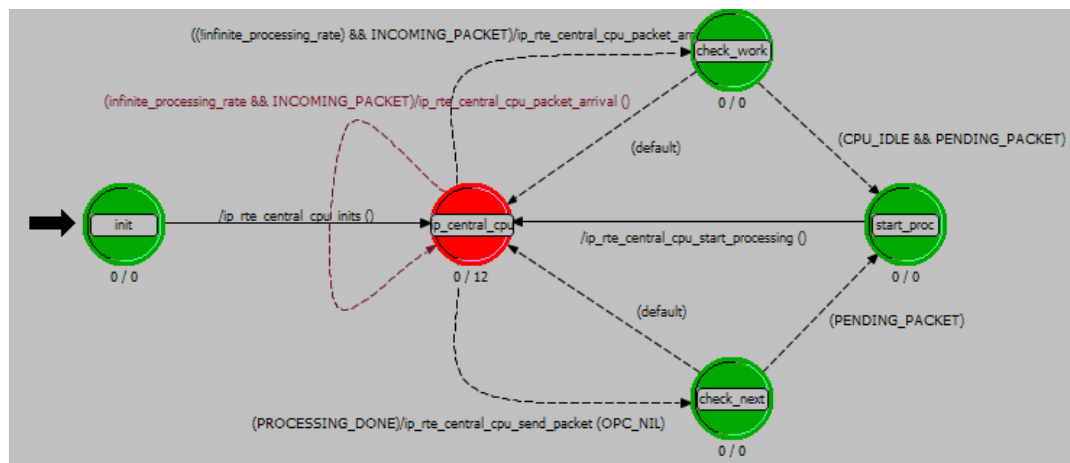


Figure 5-17 `ip_rte_central_cpu` process Model.

In the case of packet arrival, the method `ip_rte_central_cpu_packet_arrival()` is called. The three forced states to the right of the blocking state '`ip_central_cpu`' are just to add delay in case the processing rate is not infinite.

---

<sup>2</sup> In our simulation this process is used for routing, however OPNET offers another type of routing.

#### **5.4.2.2 Malicious Router Implementation**

A malicious router is a router that works as expected in term of IP protocol, but drops the packets that are destined to a destination belonging to a certain prefix. At the same time, the malicious router still advertises paths to that destination. Implementing a malicious node requires a change in the IP protocol, described above, to be able to configure a router to drop a packet if it is originated from or directed to an address that belongs to a certain prefix. At the same time, this router advertises forged paths to the rest of the Internet. An interface is provided for the user who wants to configure the malicious node to enter the time when to start blackholing and the prefixes that need to be blackholed.

#### **5.4.2.3 Exact Modifications of the IP protocol Model**

In the `bgp_dispatch` process, shown in Figure 5-16, an interface is added to enable the user to configure a malicious router. The interface is shown in Figure 5-18. The user will be able to enter the time at which the blackholing should start and the prefixes that this router shall blackhole. To see the code modification of IP protocol, refer to Appendix C section C.2.

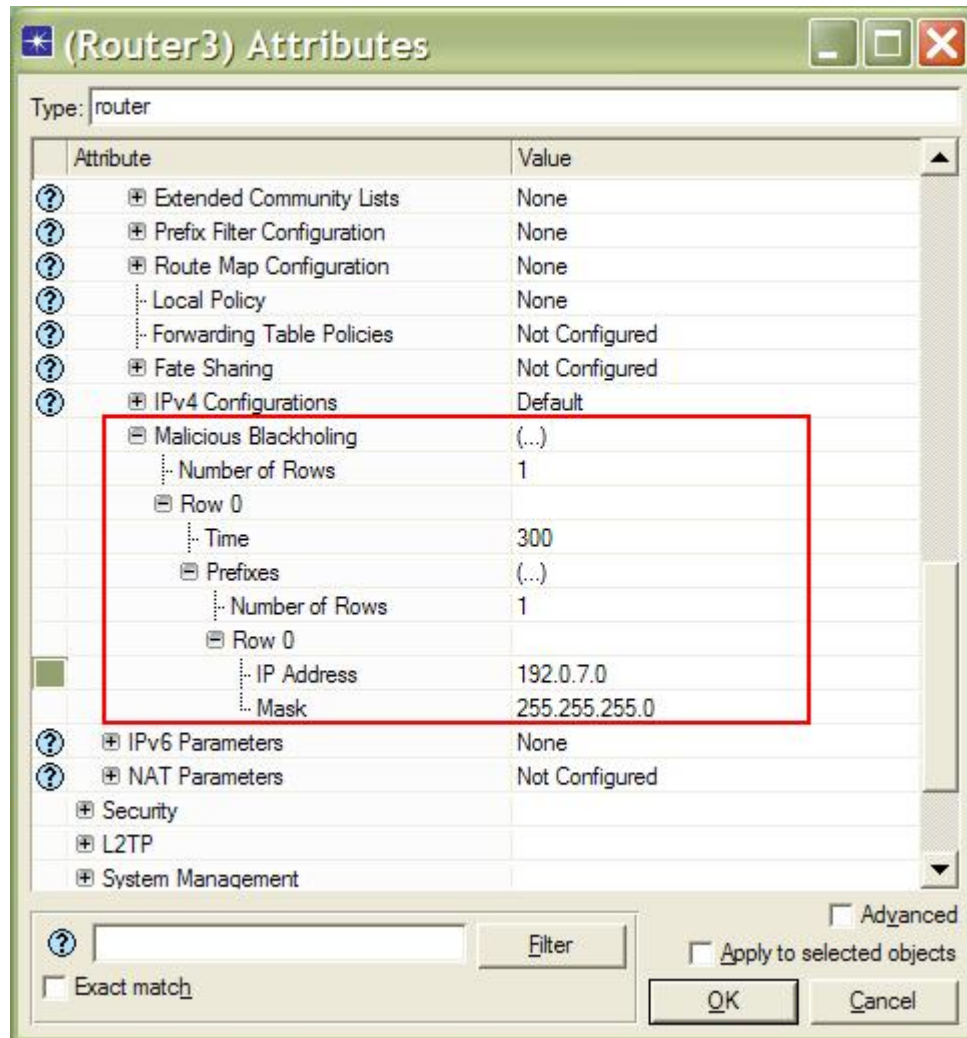


Figure 5-18 Interface to Configure Malicious Router.

In the `ip_rte_central_cpu` process shown in Figure 5-17, the method `ip_rte_central_cpu_packet_arrival()` is called whenever there is an incoming packet. So, this method is modified to call another method we define, i.e.,

---

`ip_rte_blackhole_traffic(..)`, which takes as parameters the packet and other needed information and return true if the packet is dropped due to blackholing. Figure 5-19 shows the activity diagram for this method. Refer to Appendix C section C.2.1.2 to see how the `ip_rte_blackhole_traffic` is called. Refer to Appendix C section C.2.1.3 to see the implementation of blackholing function.

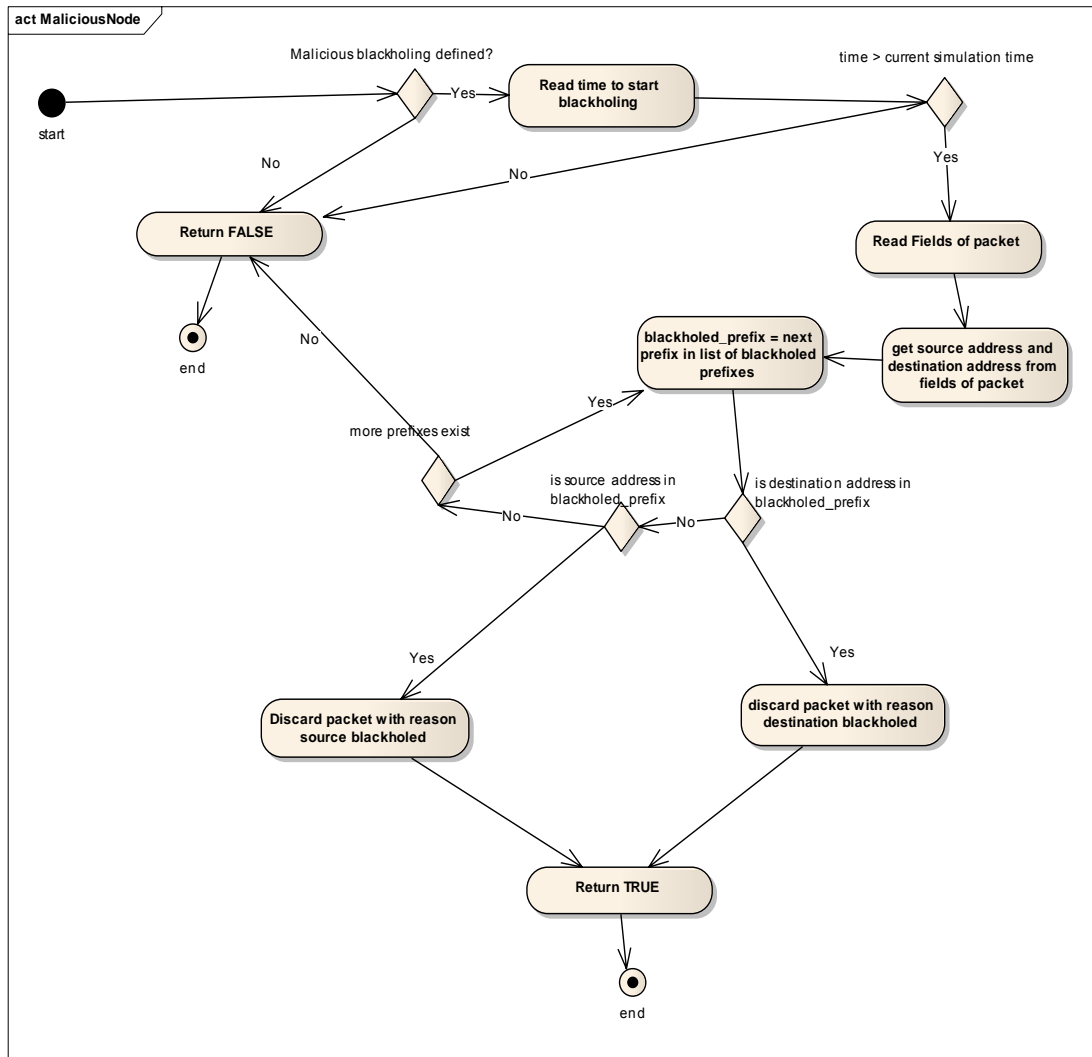


Figure 5-19 Activity diagram of blackholing method

As shown in the activity diagram, the method first checks if there is a definition of malicious blackholing. If there is, it will read the time, otherwise it will return FALSE

meaning that no blackholing occurred. It then reads the source and destination of the packet and loops through the list of blackholed prefixes to find whether the packet source address or destination address fall in any of the blackholed prefixes. It will then discard the packet noting the reason and return that the packet is blackholed if there is a match. Otherwise, it will return that no blackholing occurred. In this malicious blackholing, a built in discard function is called `ip_rte_dgram_discard`.

After running the simulation with a malicious router, blackholing is started at time 300 as shown in the configuration in Figure 5-18 and the prefix 192.0.7.0/24 is blackholed. Figure 5-20 shows the throughput traffic between Router 2 and Router 3 and between Router 2 and Router 4, in addition to the dropped traffic of Router 3 at the bottom of the figure. The vertical line indicates when the malicious node started blackholing the traffic.



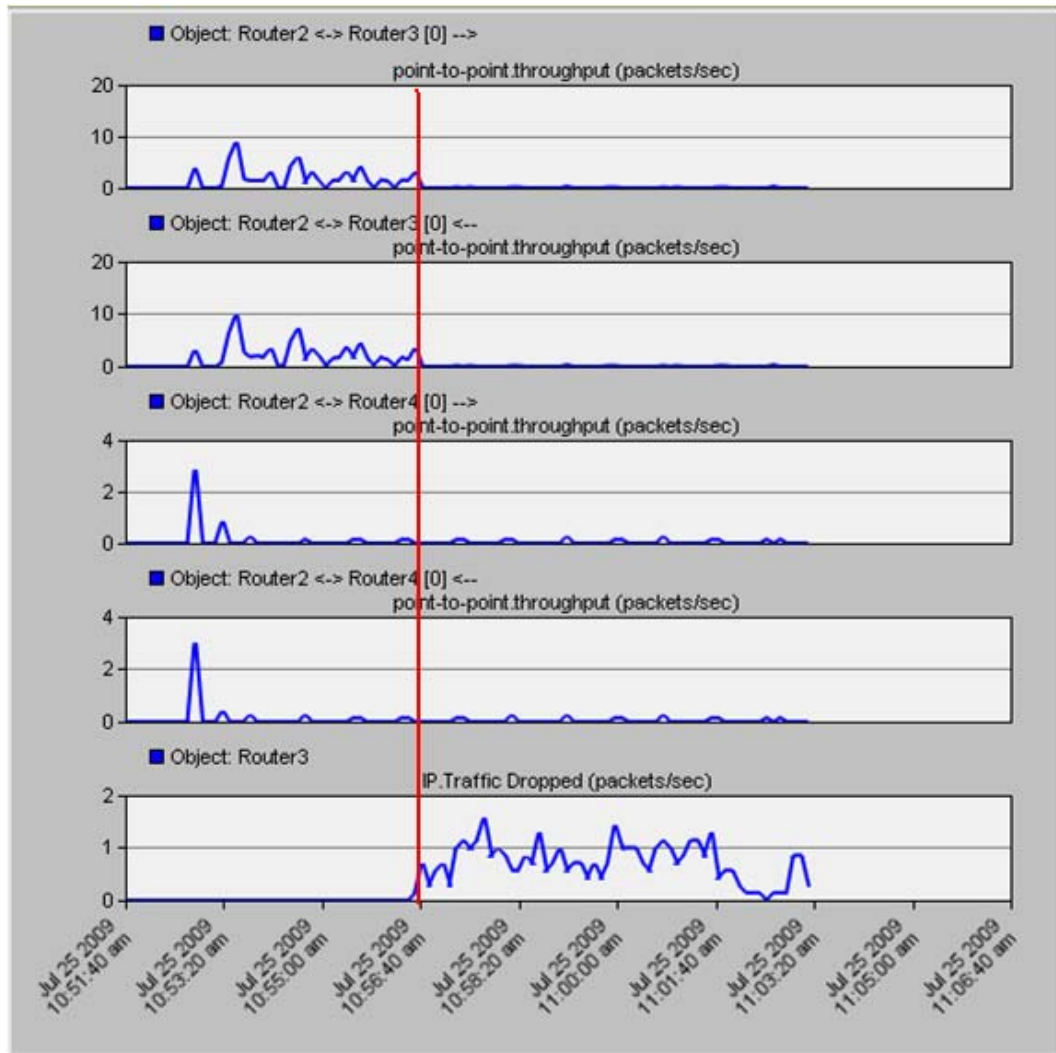


Figure 5-20 Throughput in a malicious configuration experiment

It is clear from the first two plots of Figure 5-20 that the traffic destined to or originated from Router 2 passes through Router 3. Then, the traffic stops passing at time 300 due to blackholing. It can be observed also that starting from the same time there is

a number of dropped packets but this number of dropped packets is less than the number of packets that were passing. This is due to the application used, i.e., HTTP which is built on top of TCP. TCP has congestion control implemented, so in case of message loss, TCP reduces the rate of sending messages. Also HTTP has timeout implemented if there is no acknowledgement resulting in less traffic sent.

To see that the malicious Router is advertising a path to other routers, Figure 5-21 shows the BGP table of Router 5.

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
2	192.0.1.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
3	192.0.2.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
4	192.0.3.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
5	192.0.5.0/24	Direct	192.0.5.1	Router5	IF4	0	100	32768		Incomplete		
6	192.0.6.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
7	192.0.7.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
8	192.0.8.0/24	IBGP	192.0.14.1	Router6	IF4	0	100	0	100 7	Incomplete		
9	192.0.9.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
10	192.0.11.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
11	192.0.12.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
12	192.0.14.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
13	192.0.21.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
14												

Figure 5-21 BGP table of Router 5 in Malicious Experiment

It is clear that Router 5 has next hop of the prefix 192.0.7.0/24 as Router 3 and through the AS-Path [3 12].

## ***5.5 BUILDING COUNTERMEASURES AGAINST MALICIOUS ACTS***

In this section, a detailed description of different BGP tuning methodologies running in the middle of the simulation after the malicious router has started blackholing the traffic. So this section is organized as follows. First a very short summary of OPNET implementation of BGP protocol is presented. Then, a description of the implementation needed to control the outgoing traffic is shown. This is followed by presenting the implementation needed to control the incoming traffic.

### ***5.5.1 Summary of BGP in OPNET***

The BGP implementation in OPNET consists of two processes: `bgp` process, and `bgp_conn` process. Figure 5-22 shows the state diagram of the `bgp` process.

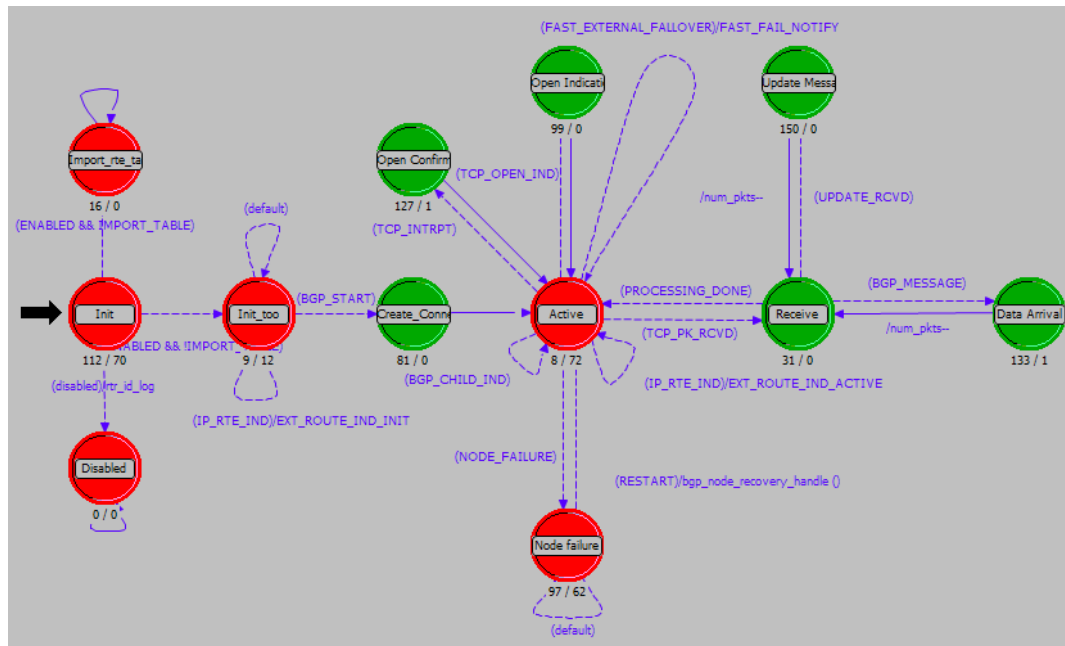


Figure 5-22 bgp process.

The 'bgp' process does the basic initialization and dynamically creates the connections with other peers and manages them. After the creation of a connection, the process will be in the Active state most of the time because it only goes after that to forced states except when the node fails. When a BGP message arrives, it also invokes the corresponding child process. A child process is an instance of the `bgp_conn` process model that is shown in Figure 5-23.

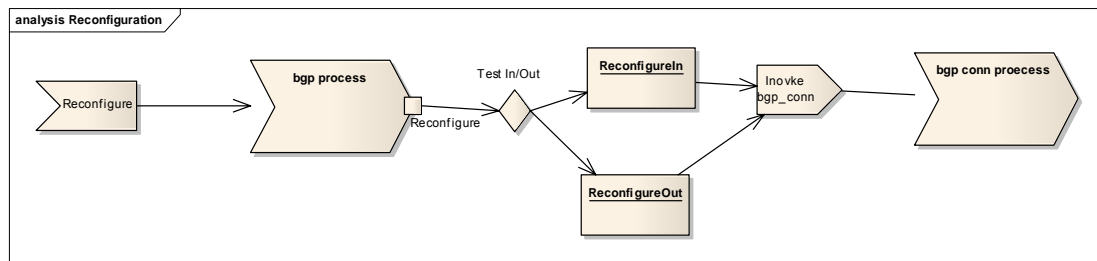


### 5.5.2 BGP Modifications for Scheduled Reconfigurations

An interface is provided to the user for every type of BGP tuning reconfiguration proposed. This interface enables the user to enter the time at which the effect of the reconfiguration should take place. The program takes all the reconfiguration requests

and stores them in a list. And then schedules a 'self' interrupt at the time specified. Refer to Appendix C section C.1 to see modification in the BGP protocol.

Figure 5-24 shows a general overview of the methodology of making a reconfiguration in general.



**Figure 5-24 Reconfiguration overview.**

As shown in Figure 5-24, the information needed for reconfiguration is read and scheduled at a specific time when the RECONFIGURE event occurs. After that, the direction of reconfiguration is tested whether it is 'IN' or 'OUT' to determine how to handle the interrupt. The child process will be invoked appropriately to complete the reconfiguration task.

Figure 5-25 shows the modified BGP protocol to get the information needed. Since the state should be in active state when the event occurs, it is handled there. A reconfiguration state is created to test whether the reconfiguration is to affect 'in'

(incoming updates) or 'out' (outgoing updates) and then it switches to reconfigureIn or reconfigureOut state accordingly.

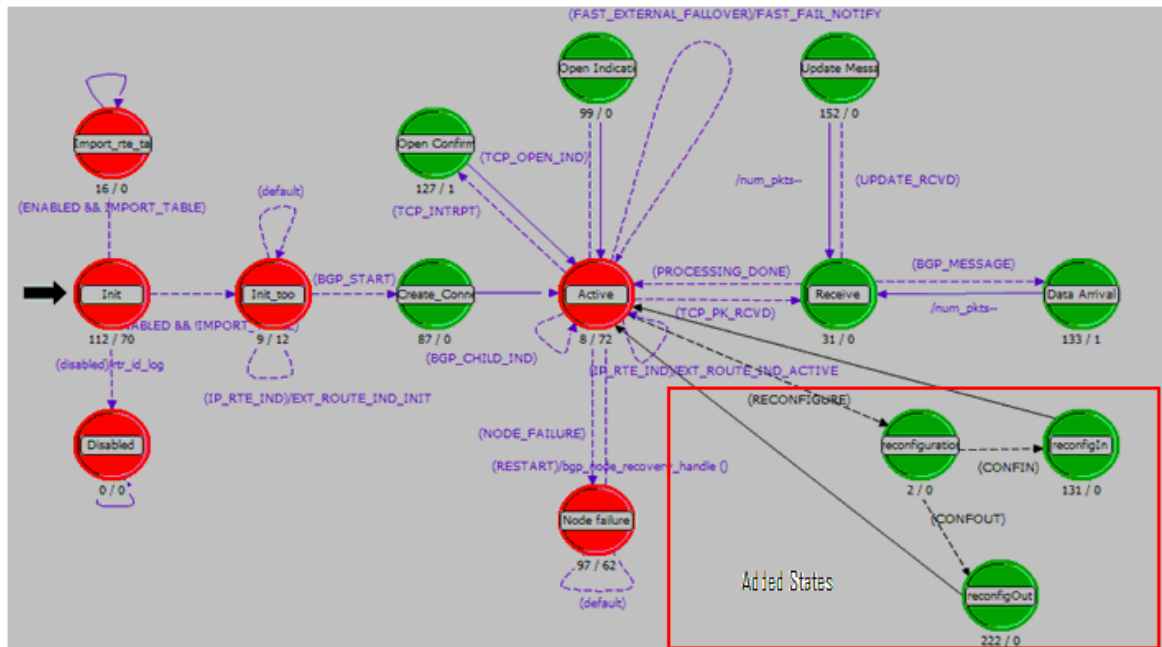


Figure 5-25 Modified bgp process model

As we can see, when the event of reconfiguring occurs, the RECONFIGURE condition will evaluate to true and the reconfiguration state is entered. At this state, the direction is checked and if the configuration is for the incoming routes, then the 'reconfigIn' state is entered; and when the direction is Out, then the 'reconfigOut' state is entered. As soon as the code is executed, the state will go back to the Active state, and this

execution includes the invocation of the child process 'bgp\_conn' for reading some database, i.e. incoming and outgoing advertised routes, or for distributing the new route to other neighbors as it will be shown in the sections that follow.

#### ***5.5.2.1    Modification to Control The Outgoing Traffic***

local-pref is used in order to control the outgoing traffic. Changing the local-pref parameter in the middle of simulation is not possible in version 14.5 of OPNET release. Therefore, we modified the built-in implementation of BGP to add this capability. To see the code of the control of outgoing traffic refer to Appendix C section C.1.1.6.

To control the outgoing traffic, a modification of the local-pref of routes received from a specific neighbor is needed. Then, the selection process is triggered to prefer routes with a higher local-pref. Therefore, the change needs to be done on routes received from the preferred neighbor. In this case, the direction of advertisement is 'In'. To implement the reconfigIn, a similar activity to what is done in the Update message state needs to be implemented, except that this time the route is read from the 'rib-in' and not from the advertisement. 'rib-in' is a database that is used to store every advertisement received from a specific neighbor. When a reconfiguration event occurs,



the route map policy associated with the event, i.e., setting higher local preference, is applied on the 'rib-in' routes. After this, the modified routes are treated as if a new route is received. So, the BGP selection process works normally to select the best path. Of course, it shall select the routes with higher local preferences. Finally, the routing table is updated which will affect the forwarding decision by the IP protocol.

In the 'reconfigureIn' state, the corresponding child is invoked which and shall be in the Established state. The child process is modified to handle this invocation and achieve the required modification to the route. The child process will check the 'rib-in' for this neighbor, apply the route map on the routes, and then add the modified routes to a list visible to the parent process, i.e., bgp process. Figure 5-26 shows how the child process handles reconfiguration.

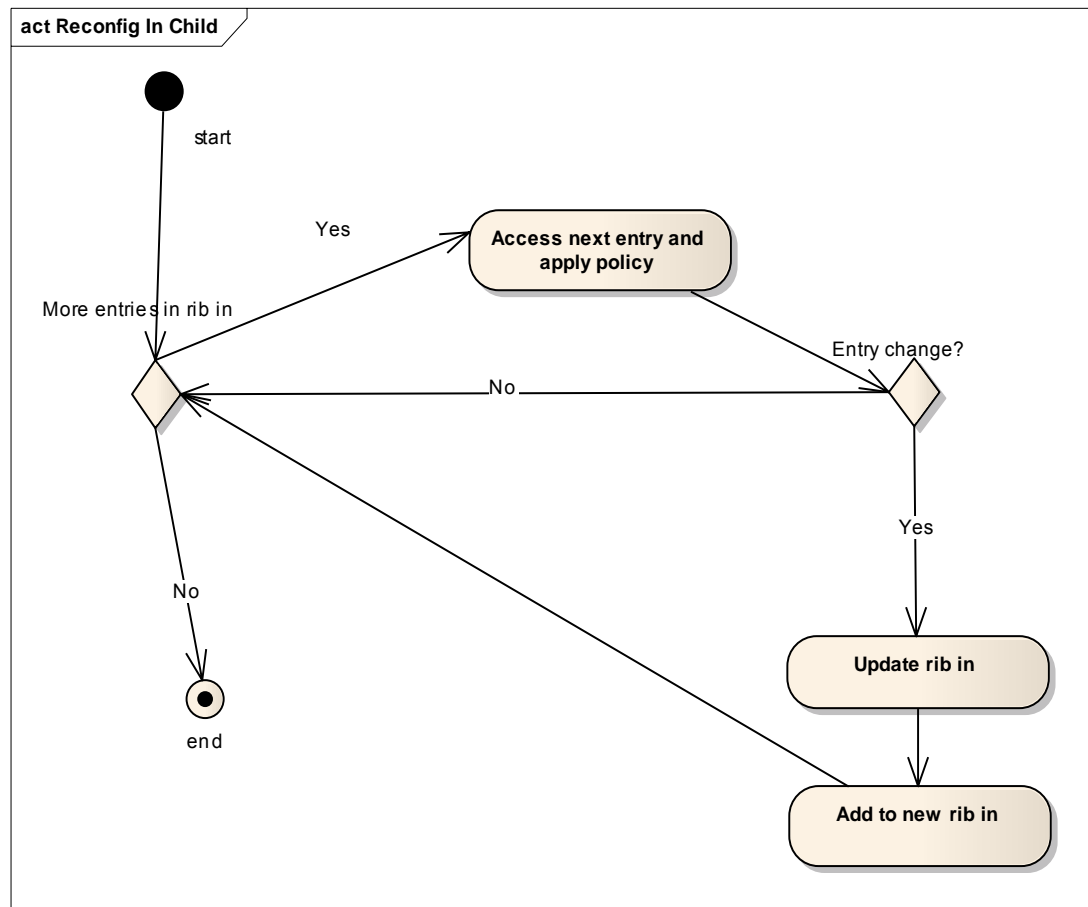


Figure 5-26 Handling ReconfigureIn in a child process

The parent process then processes the newly added routes to 'rib-in' one by one and determines if they are better than the existing routes to the destination. If that is the case, then the local-rib is changed in addition to IP routing table and the new route will be propagated to other neighbors. Figure 5-27 shows the handling of reconfigIn of the parent process after the child is returned.

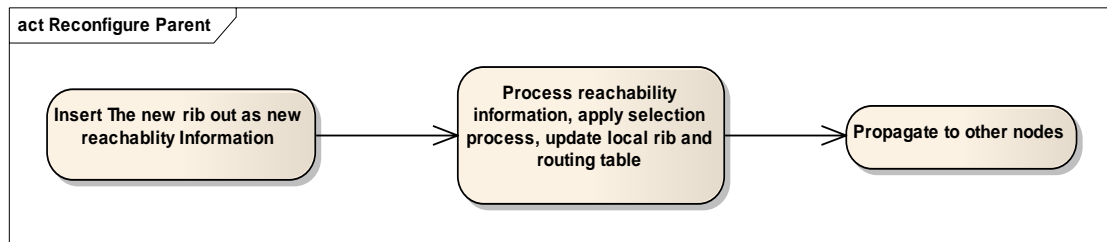


Figure 5-27 Handling of ReconfigIn in parent

As shown in Figure 5-27, the updated 'rib-in' routes are used to update the local rib and routing table by replacing an existing route to a specific entry if the route is preferred over the existing one. Also, it is clear from this figure, our reuse of two major functions in OPNET to handle reachability information and to propagate it to other routes.

The experiment is conducted by having a malicious router starting at time 300, and configuring 'local-pref' to be applied at time 350. Figure 5-28 shows an example of specifying the time when applying a route map.

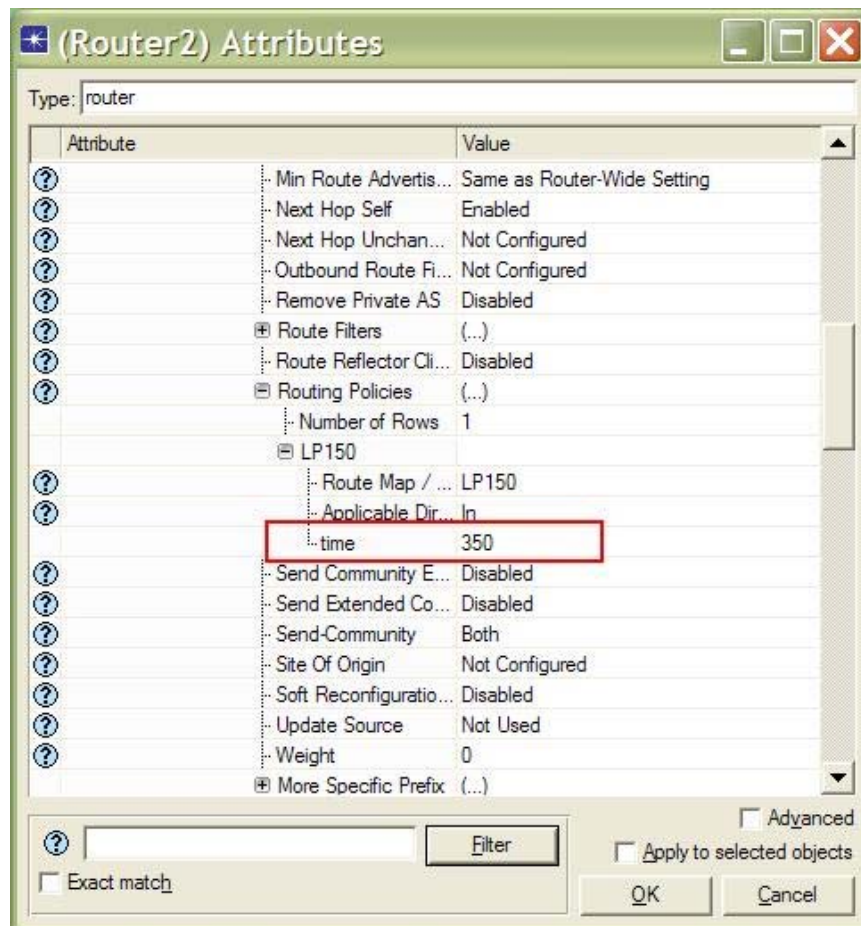


Figure 5-28 Specification of time when applying route map.

After running the experiment, the incoming and outgoing traffic of Router 2 with Router 3 and of Router 2 with Router 4, in addition to the traffic drop are shown in Figure 5-29.

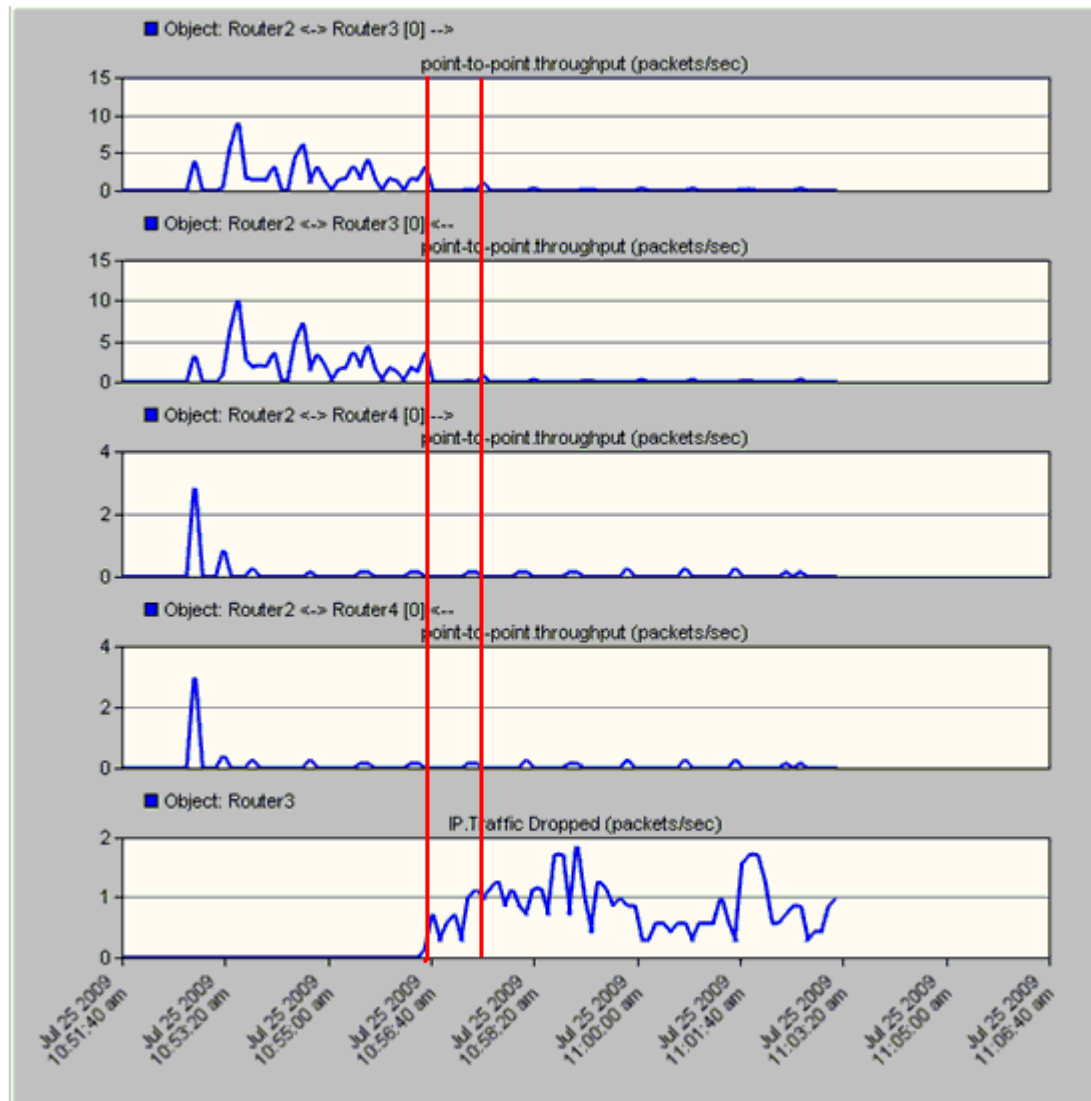
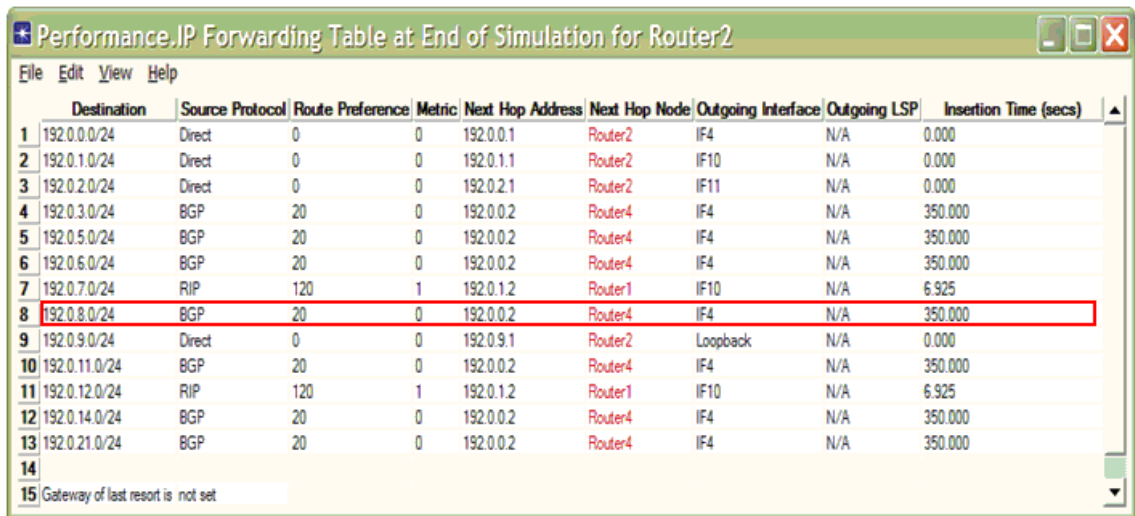


Figure 5-29 Throughput traffic after applying localpref in the presence of a malicious router

Referring to Figure 5-29, it is clear that applying a higher local-pref to the route does not show that the outgoing traffic is passing through Router 4. To make sure that a change has occurred, we can see the IP forwarding table of Router 2 in Figure 5-30.



	Destination	Source Protocol	Route Preference	Metric	Next Hop Address	Next Hop Node	Outgoing Interface	Outgoing LSP	Insertion Time (secs)
1	192.0.0.0/24	Direct	0	0	192.0.0.1	Router2	IF4	N/A	0.000
2	192.0.1.0/24	Direct	0	0	192.0.1.1	Router2	IF10	N/A	0.000
3	192.0.2.0/24	Direct	0	0	192.0.2.1	Router2	IF11	N/A	0.000
4	192.0.3.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	350.000
5	192.0.5.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	350.000
6	192.0.6.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	350.000
7	192.0.7.0/24	RIP	120	1	192.0.1.2	Router1	IF10	N/A	6.925
8	192.0.8.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	350.000
9	192.0.9.0/24	Direct	0	0	192.0.9.1	Router2	Loopback	N/A	0.000
10	192.0.11.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	350.000
11	192.0.12.0/24	RIP	120	1	192.0.1.2	Router1	IF10	N/A	6.925
12	192.0.14.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	350.000
13	192.0.21.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	350.000
14									
15	Gateway of last resort is not set								

Figure 5-30 IP forwarding table of Router 2 in Local-Pref and Malicious experiment

It is clear that at time 350, the prefix to 192.0.8.0/24 is inserted in the table. This route entry was initially through Router 2. To see this, Figure 5-31 shows the IP forwarding table for Router 2 at time 71.

	Destination	Source Protocol	Route Preference	Metric	Next Hop Address	Next Hop Node	Outgoing Interface	Outgoing LSP	Insertion Time (secs)
1	192.0.0.0/24	Direct	0	0	192.0.0.1	Router2	IF4	N/A	0.000
2	192.0.1.0/24	Direct	0	0	192.0.1.1	Router2	IF10	N/A	0.000
3	192.0.2.0/24	Direct	0	0	192.0.2.1	Router2	IF11	N/A	0.000
4	192.0.3.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	70.037
5	192.0.5.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	70.037
6	192.0.6.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	70.037
7	192.0.7.0/24	RIP	120	1	192.0.1.2	Router1	IF10	N/A	6.925
8	192.0.8.0/24	BGP	20	0	192.0.2.2	Router3	IF11	N/A	70.048
9	192.0.9.0/24	Direct	0	0	192.0.9.1	Router2	Loopback	N/A	0.000
10	192.0.11.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	70.037
11	192.0.12.0/24	RIP	120	1	192.0.1.2	Router1	IF10	N/A	6.925
12	192.0.14.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	70.037
13	192.0.21.0/24	BGP	20	0	192.0.0.2	Router4	IF4	N/A	70.037

Figure 5-31 IP forwarding table of Router 2 at time 71 in the Local-pref and Malicious experiment

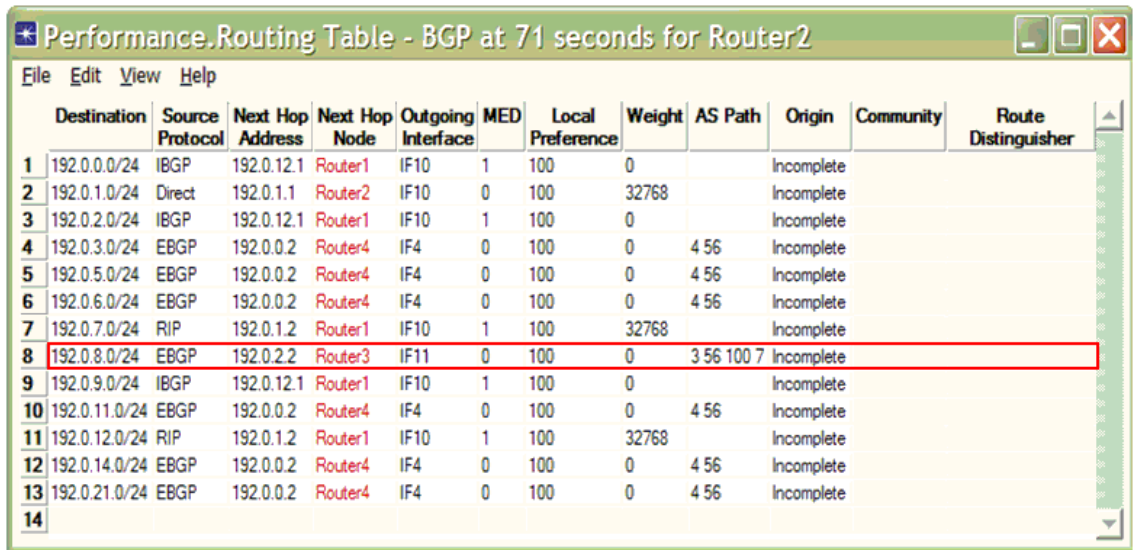
Figure 5-31 shows that the entry that has in the destination to the prefix 192.0.8.0/24 has Router3 as the 'Next Hop Node'. The local preference to this destination at time 71 was set to 100. Figure 5-32 shows the BGP routing table at time 71.

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
2	192.0.1.0/24	Direct	192.0.1.1	Router2	IF10	0	100	32768		Incomplete		
3	192.0.2.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
4	192.0.3.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
5	192.0.5.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
6	192.0.6.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
7	192.0.7.0/24	RIP	192.0.1.2	Router1	IF10	1	100	32768		Incomplete		
8	192.0.8.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56 100 7	Incomplete		
9	192.0.9.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
10	192.0.11.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
11	192.0.12.0/24	RIP	192.0.1.2	Router1	IF10	1	100	32768		Incomplete		
12	192.0.14.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		
13	192.0.21.0/24	EBGP	192.0.0.2	Router4	IF4	0	150	0	4 56	Incomplete		

Figure 5-32 BGP routing table of Router 2 at time 71 for Local-pref and Malicious Experiment

Figure 5-33 shows the BGP routing table of Router 2. This table shows that because this route has a higher local preference it is selected as the best to 192.0.8.0/24 destination.





Performance Routing Table - BGP at 71 seconds for Router2

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
2	192.0.1.0/24	Direct	192.0.1.1	Router2	IF10	0	100	32768		Incomplete		
3	192.0.2.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
4	192.0.3.0/24	EBGP	192.0.0.2	Router4	IF4	0	100	0	4 56	Incomplete		
5	192.0.5.0/24	EBGP	192.0.0.2	Router4	IF4	0	100	0	4 56	Incomplete		
6	192.0.6.0/24	EBGP	192.0.0.2	Router4	IF4	0	100	0	4 56	Incomplete		
7	192.0.7.0/24	RIP	192.0.1.2	Router1	IF10	1	100	32768		Incomplete		
8	192.0.8.0/24	EBGP	192.0.2.2	Router3	IF11	0	100	0	3 56 100 7	Incomplete		
9	192.0.9.0/24	IBGP	192.0.12.1	Router1	IF10	1	100	0		Incomplete		
10	192.0.11.0/24	EBGP	192.0.0.2	Router4	IF4	0	100	0	4 56	Incomplete		
11	192.0.12.0/24	RIP	192.0.1.2	Router1	IF10	1	100	32768		Incomplete		
12	192.0.14.0/24	EBGP	192.0.0.2	Router4	IF4	0	100	0	4 56	Incomplete		
13	192.0.21.0/24	EBGP	192.0.0.2	Router4	IF4	0	100	0	4 56	Incomplete		
14												

Figure 5-33 BGP Routing Table of Router 2 in Local-pref and Malicious Experiment.

Although the routing table of Router 2 shows that the router has the next hop to LAN\_East destination as Router 4, and Router 4 is not malicious, we cannot see any outgoing traffic after Router 3 becomes malicious. The reason is because of the way the application works. LAN\_East runs as a client requesting from the HTTP server in LAN\_West some services. Simply LAN\_East did not reply because it does not receive requests. Message exchange between routers is depicted in Figure 5-34. In this figure, Router 1, Router 6, Router 7 and the Internet are not shown for clarity and because they don't play a major role in the decision, but they only forward messages. So, between

LAN\_West and Router 2, Router 1 resides and between Router 5 and LAN\_East, Router 6, the Internet, and Router 7 reside.

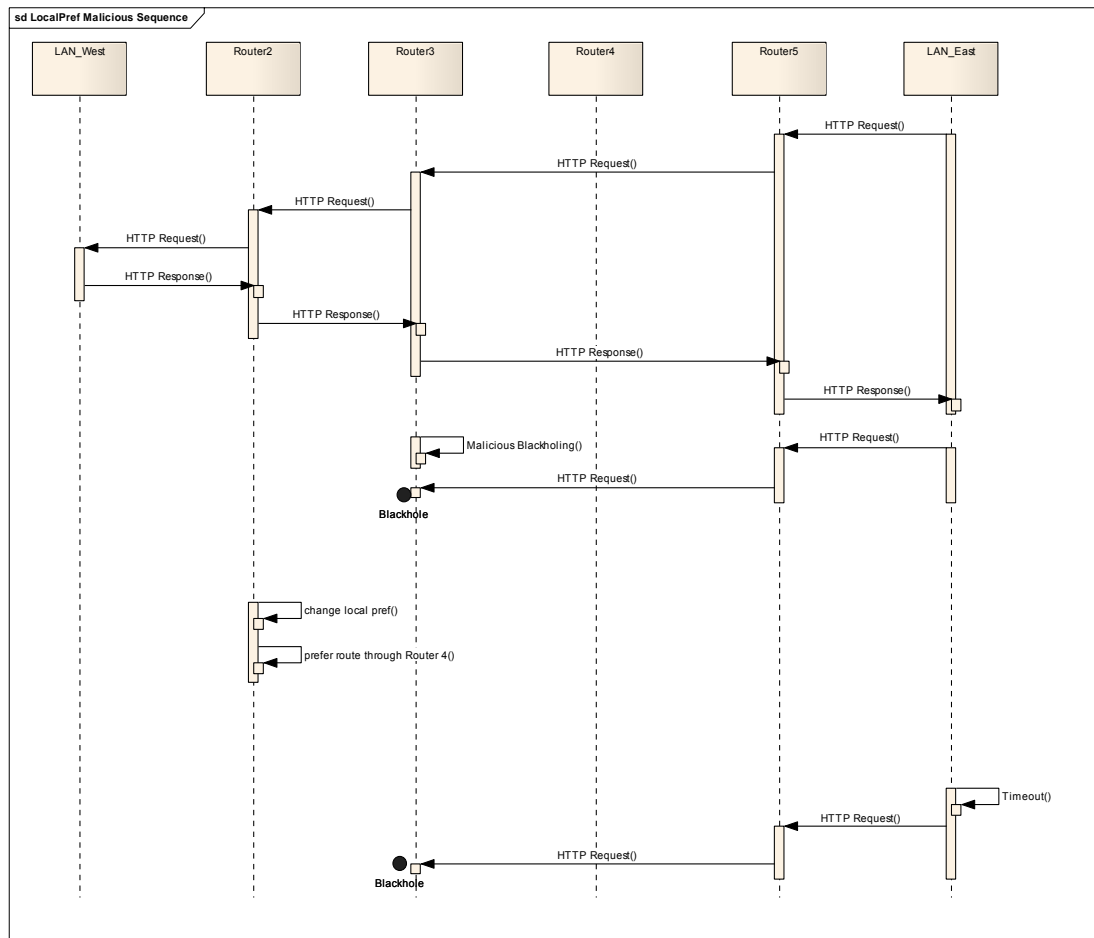


Figure 5-34 Message exchange for the Malicious and Local-Pref

As shown in Figure 5-34, Initially, LAN\_East sends an HTTP request to LAN\_West and it gets a reply as expected. After that, Router 3 starts to behave maliciously; Router 2 will

not receive the request. So even after the `local-pref` is changed, because the incoming traffic for this application is coming from the malicious router, no reply is expected.

Figure 5-35 shows the convergence activity of the network.

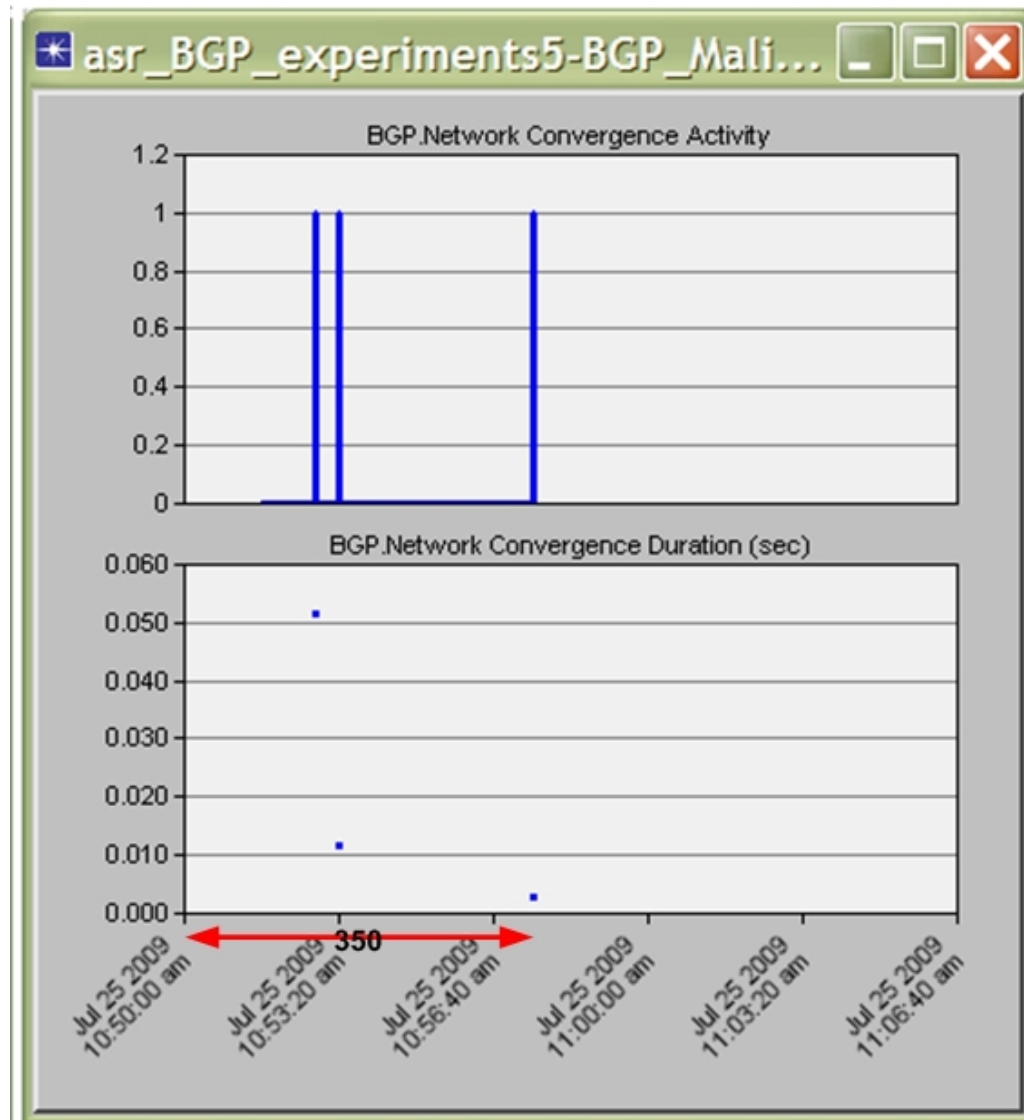


Figure 5-35 Convergence activity of the local pref and malicious experiment

Due to change in the local-pref at time 350 seconds, BGP needs less than 0.005 seconds to converge according to the convergence activity shown in Figure 5-35.

#### ***5.5.2.2    Control of Incoming Traffic: The Use of Community***

Among the proposed BGP tuning methods for controlling the incoming traffic and solving the problem of malicious IISP, the community method is readily implemented. The only thing needs to be done is to allow the reconfiguration of the 'OUT' routes in the middle of the simulation. In this section, a description of the methodology to allow applying route maps that target the 'OUT' routes, i.e., the routes that are advertised to other BGP speakers or routers. This is done in order to influence the routers to prefer the paths through the good ISP in the middle of simulation. Then, this is experimented by the use the community method.

The user configures the 'out' configuration in the same way he configures the 'in' configuration. The procedure to do this reconfiguration in the middle of the simulation is in the following way. First, the policy is applied on the 'rib-out' table of a specific neighbor (the good IISP) then the modified route is sent to that specific neighbor. Figure 5-36 shows the implementation of the 'reconfigureOut' state.

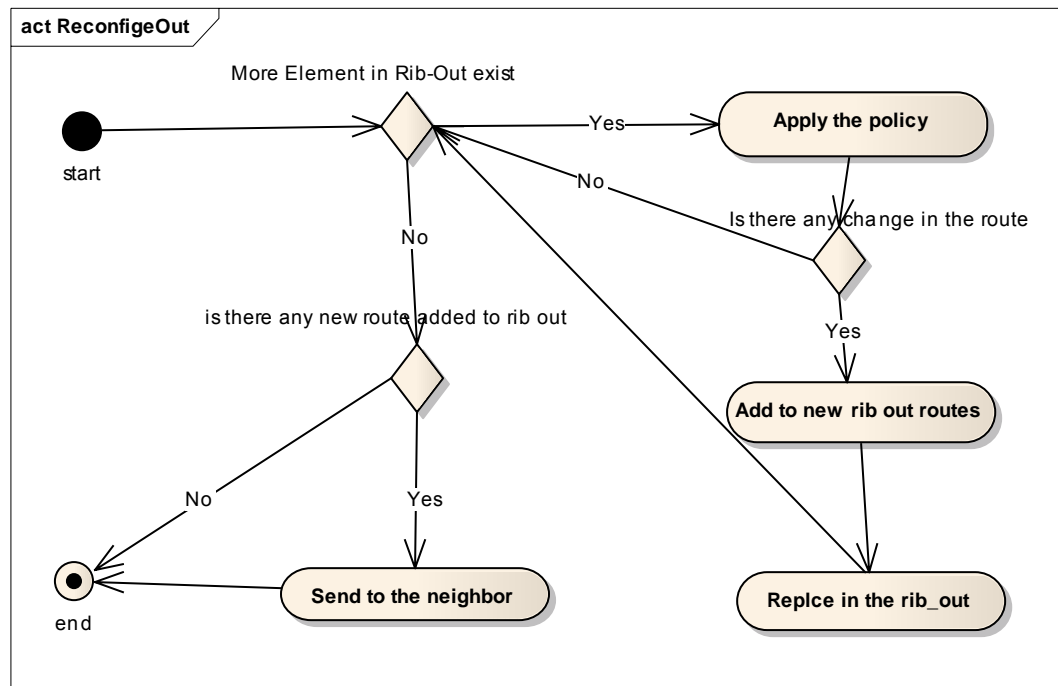


Figure 5-36 ReconfigOut Activity Diagram

The first experiment in controlling the incoming traffic is by the use of community. At time 350, Router 2 resends the routes in 'rib out' to Router 4 but with certain community number 12:144. When this advertisement is received by Router 5, Router 5 will give a higher local preference to these routes. As a result, the incoming traffic is influenced to prefer paths through the good IISP, i.e., Router 4. Figure 5-37 shows the throughput between Router 2 and Router 3 and between Router 2 and Router 4 in both directions. In addition, the figure shows the packets dropped due to malicious act that started at time 300.

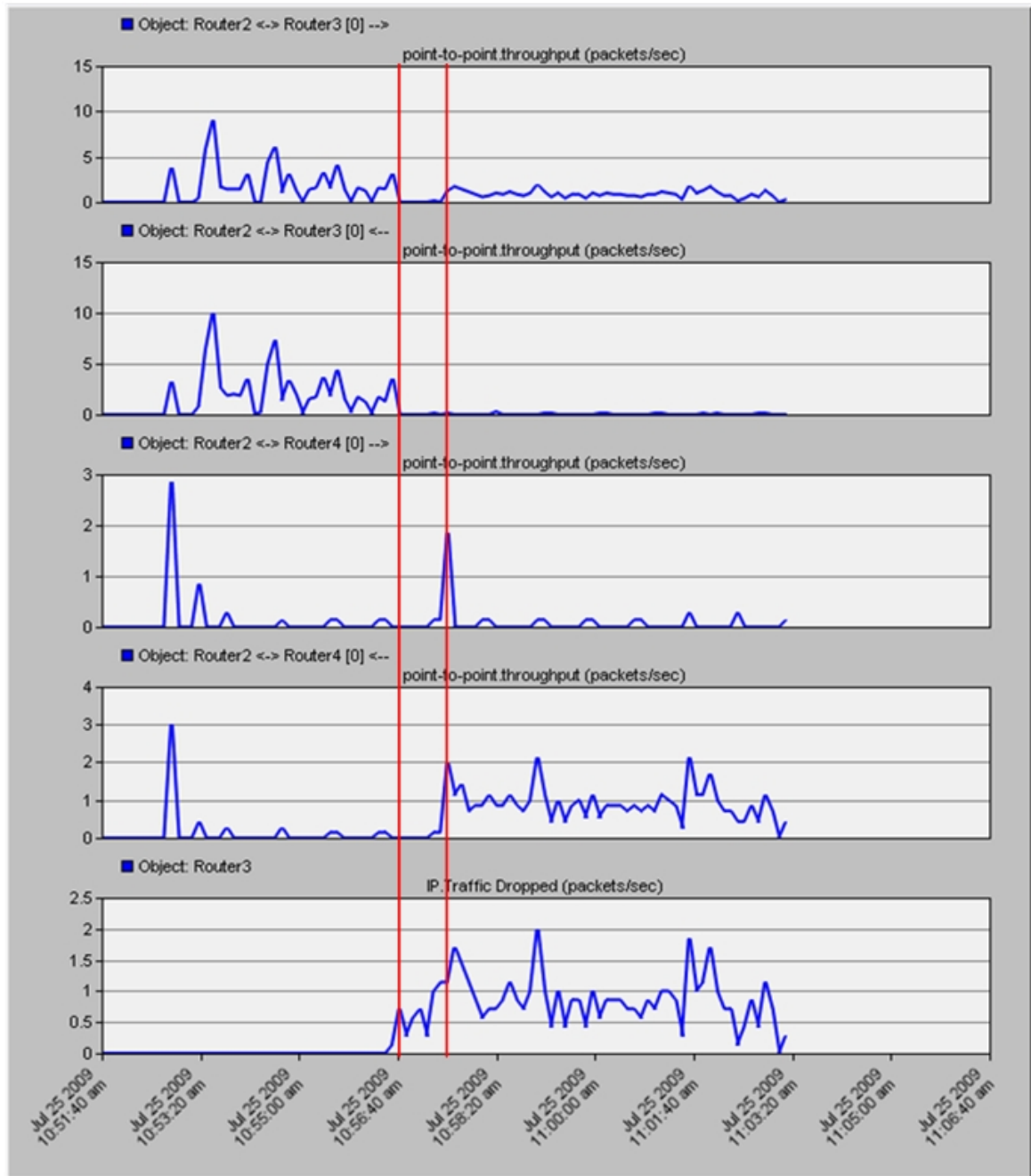


Figure 5-37 Throughput between Router 2 and Router3, Router 4 and Packet drop of Router 3

As shown in Figure 5-37, the traffic going from Router 2 to Router 3 stops for some time and then some traffic appears again. Also there is some traffic coming from Router 4 to Router 2 after applying the change for the incoming traffic. There is also an equivalent number of dropped traffic even after the policy is applied. The reason for this behavior is that the incoming traffic arrives through Router 4 but the outgoing traffic is going through Router 3. So this traffic is blackholed in Router 3 and the response does not reach to Router 3.

Figure 5-38 shows the BGP routing table of Router 5.

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
2	192.0.1.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
3	192.0.2.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
4	192.0.3.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
5	192.0.5.0/24	Direct	192.0.5.1	Router5	IF4	0	100	32768		Incomplete		
6	192.0.6.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
7	192.0.7.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
8	192.0.8.0/24	IBGP	192.0.14.1	Router6	IF4	0	100	0	100 7	Incomplete		
9	192.0.9.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
10	192.0.11.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
11	192.0.12.0/24	EBGP	192.0.6.2	Router4	IF11	0	150	0	4 12	Incomplete	[12:144]	
12	192.0.14.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
13	192.0.21.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
14												

Figure 5-38 BGP routing table of Router 5 in malicious and community experiment



As shown in Figure 5-38, the route to the prefix 192.0.7.0/24 has the community set to [12:144], so its local preference is set to 150. That is why the route through Router 4 is preferred. Figure 5-39 shows the BGP routing table at time 71 to see the previous decision taken before sending an Update message that contains the community number.

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
2	192.0.1.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
3	192.0.2.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
4	192.0.3.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
5	192.0.5.0/24	Direct	192.0.5.1	Router5	IF4	0	100	32768		Incomplete		
6	192.0.6.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
7	192.0.7.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
8	192.0.8.0/24	IBGP	192.0.14.1	Router6	IF4	0	100	0	100 7	Incomplete		
9	192.0.9.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
10	192.0.11.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
11	192.0.12.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
12	192.0.14.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
13	192.0.21.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
14												

Figure 5-39 BGP routing table of Router 5 malicious & community experiment at time 71 before sending update message with community number.

The route to the prefix 192.0.7.0/24 at time 71 is through Router 3 and has local preference of 100 and no community set.

Figure 5-40 shows the message passing between routers in this experiment.

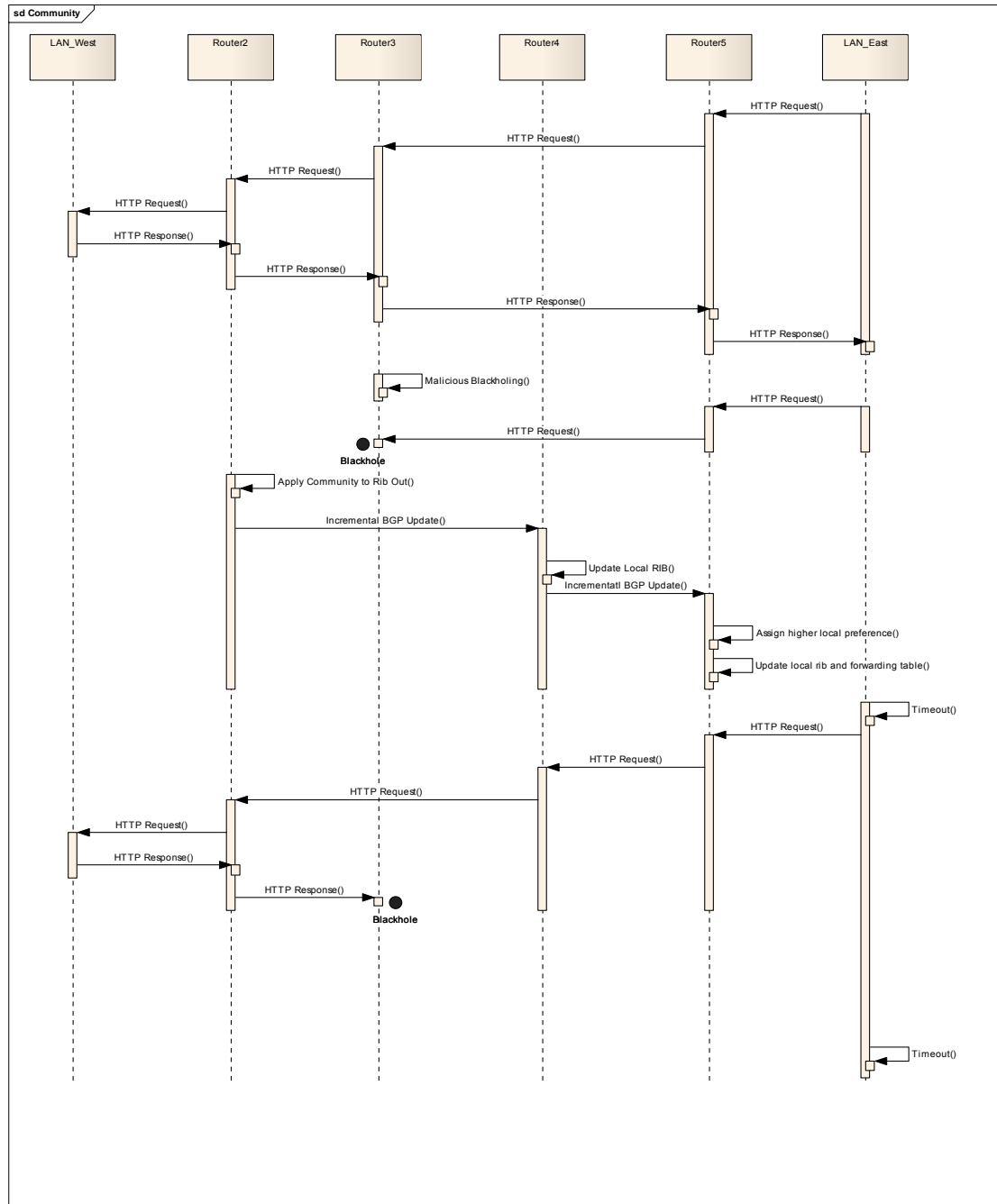


Figure 5-40 Message Passing between routers in community and malicious experiment

As shown in Figure 5-40, after the community is applied, the requests are received normally; however, the replies are blackholed, so the application as a whole did not work. Therefore, In order to make the application work fully, both incoming and outgoing traffic must be directed through the good IISP.

Figure 5-41 shows the convergence activity and duration of the community and malicious experiment.

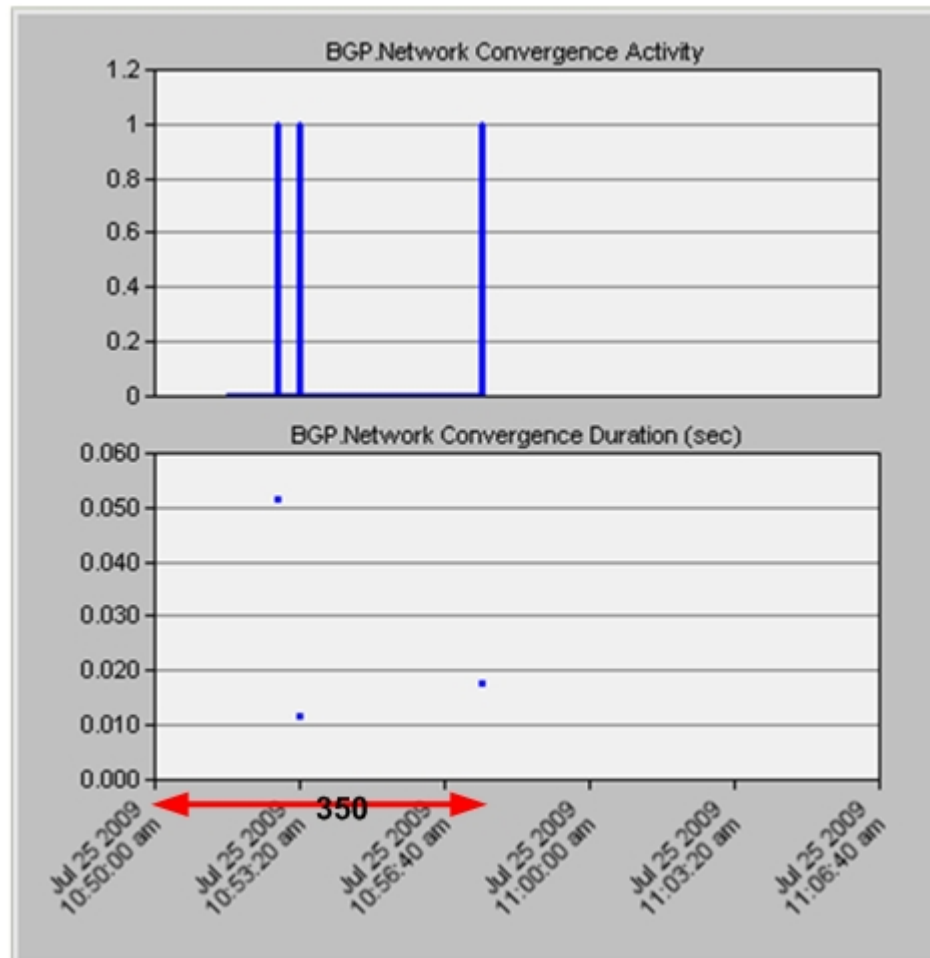


Figure 5-41 Convergence Activity in Community and Malicious

The network takes about 0.018 seconds to converge after the change is applied by the use of community.

### **5.5.2.1 Shortening**

As we mentioned earlier, shortening is not supported by BGP. To configure prepending, the user shall specify in the route map that he wants to change the `AS-Path` and set 'prepend' as an Action, then set what the user wants to prepend. After that, the user can apply this route map for the IISP that he wants the traffic not to pass through. In the OPNET implementation, if the user did specify nothing to prepend, the `AS-Path` is not affected. This has the same effect as if the user did not set the 'set as' property in the route map. So, I utilized this to implement the shortening. Figure 5-42 shows how the user can configure shortening by just having the 'Set Value' in the route map to set to empty. Refer to Appendix C section C.1.3 to see the shortening code.

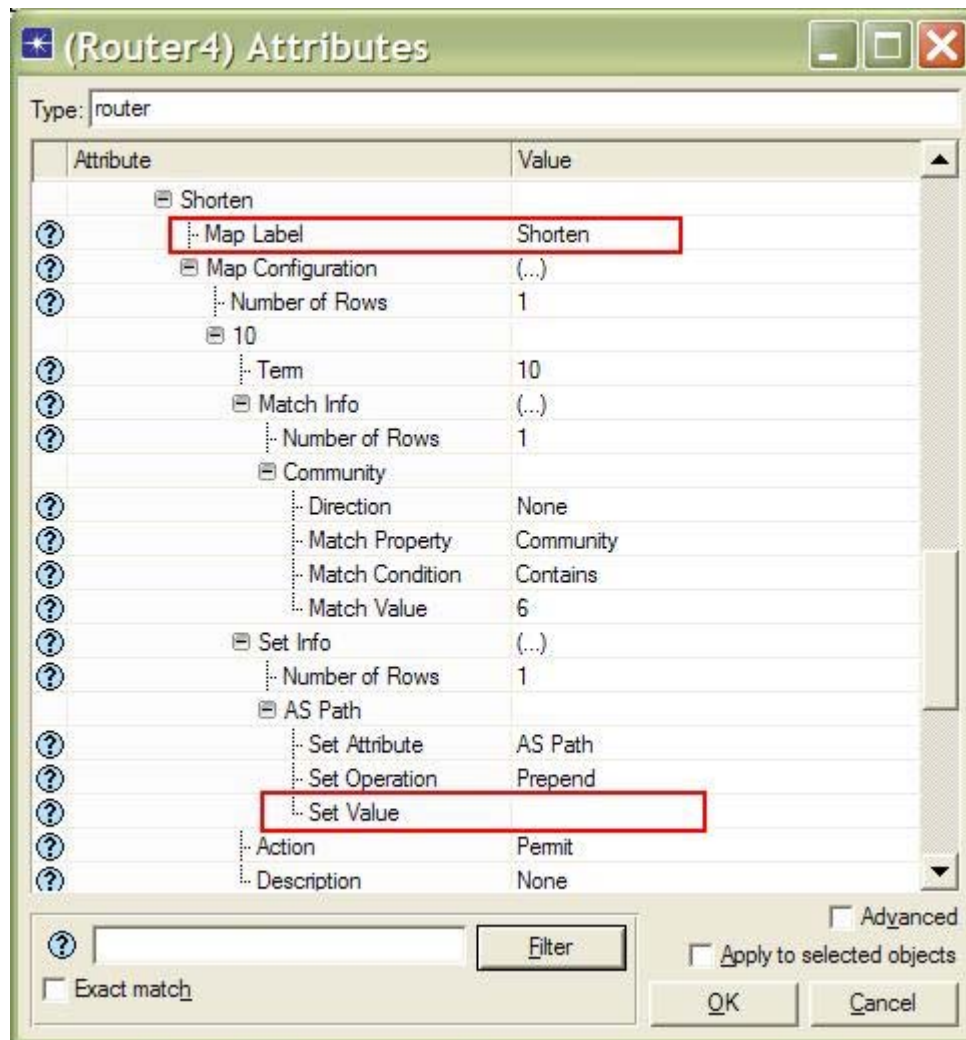


Figure 5-42 Shortening Configuration

In order to implement shortening, two routers need to be involved, i.e., the regional ISP and the international ISP. An agreement between them has to be accomplished, and that is if the route is advertised with certain community number then it shall be

shortened. The regional ISP is configured to send a route update with a certain community number at a specific time. The good International ISP (IISP) is configured to check any route that is coming and if it has this community number, it will shorten the route. Shortening the route means to remove the AS number of the regional ISP and then advertise the route by adding only the AS number of the IISP. The advertised route to the Internet should then be shorter by one than the one advertised by the malicious ISP attracting more traffic through the good one. All the 'rib-out' that are advertised previously to the neighbor will have to be sent again with their community number set to a specific value.

For this experiment, shortening and local preference will be applied at the same time, i.e., 350, in order to check whether the traffic recovers after the malicious router starts being malicious at time (300). Figure 5-43 shows the throughput between Router 2 and Router 3 and between Router 2 and Router 4 in both directions in addition to the traffic dropped at Router 3.

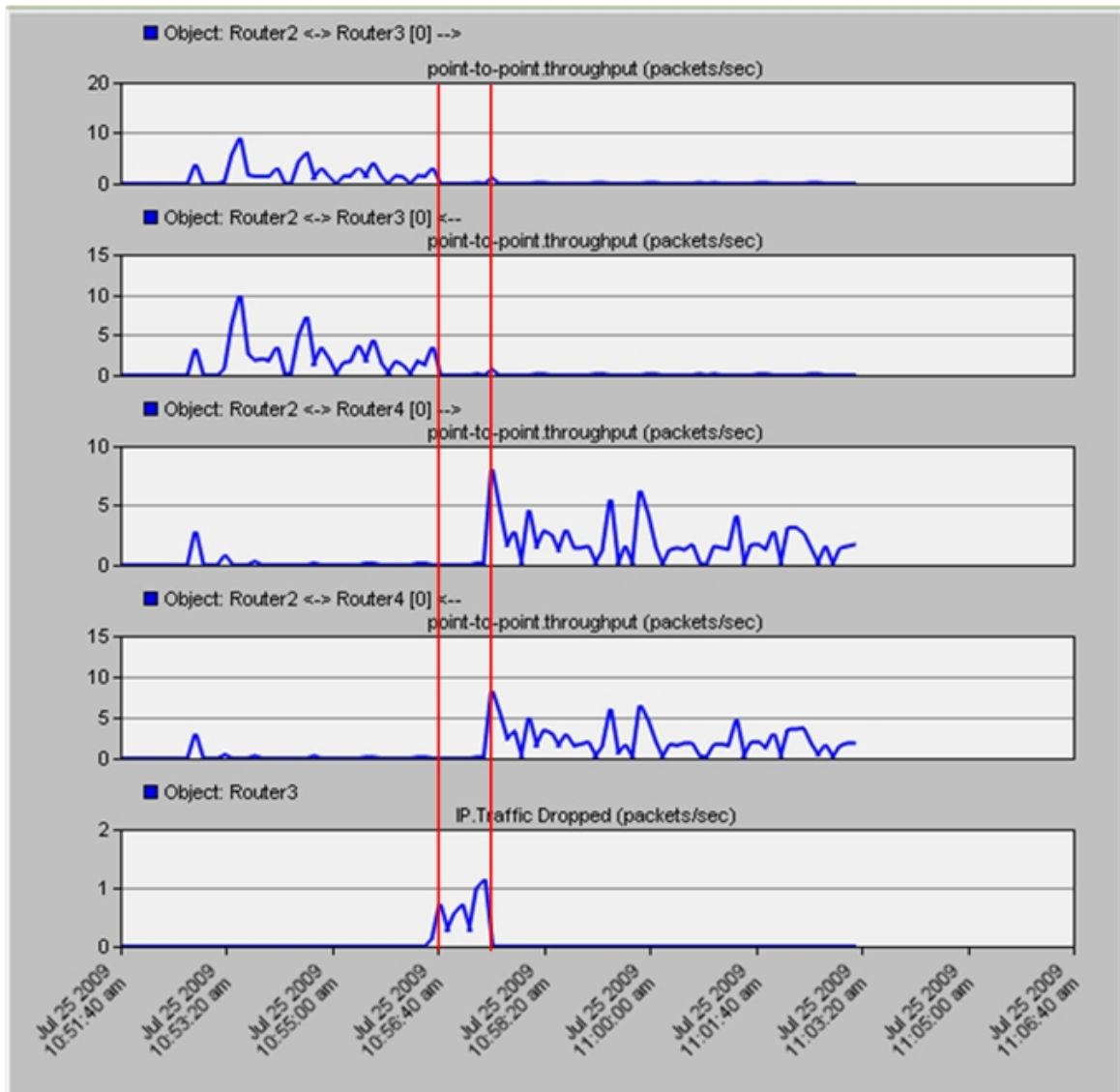
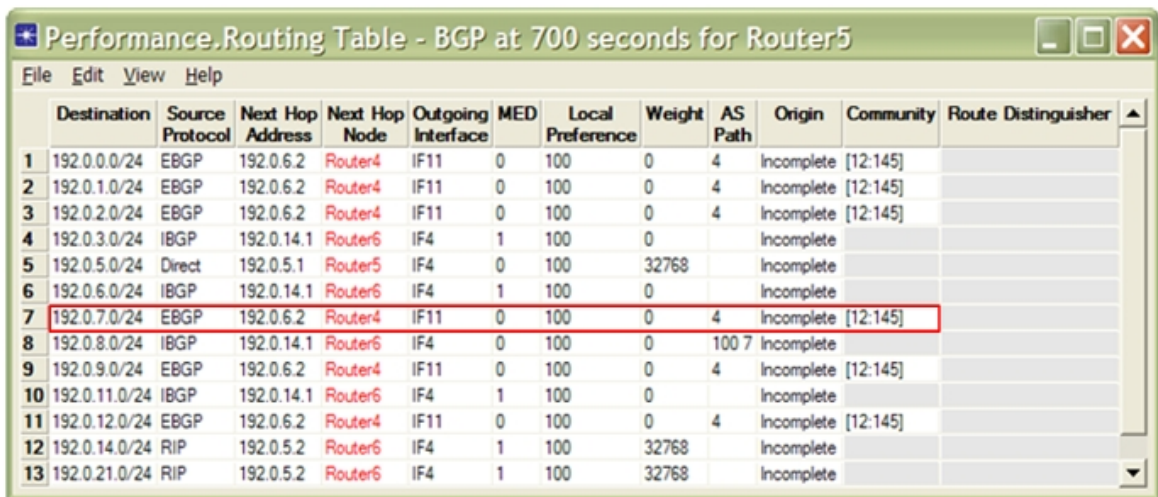


Figure 5-43 Throughput between Router 2 and Routers 3 & 4 and dropped traffic of Router 3

It is clear from Figure 5-43 that when both the incoming and outgoing traffic go through the good IISP, the traffic will recover. We see that only for small amount of time



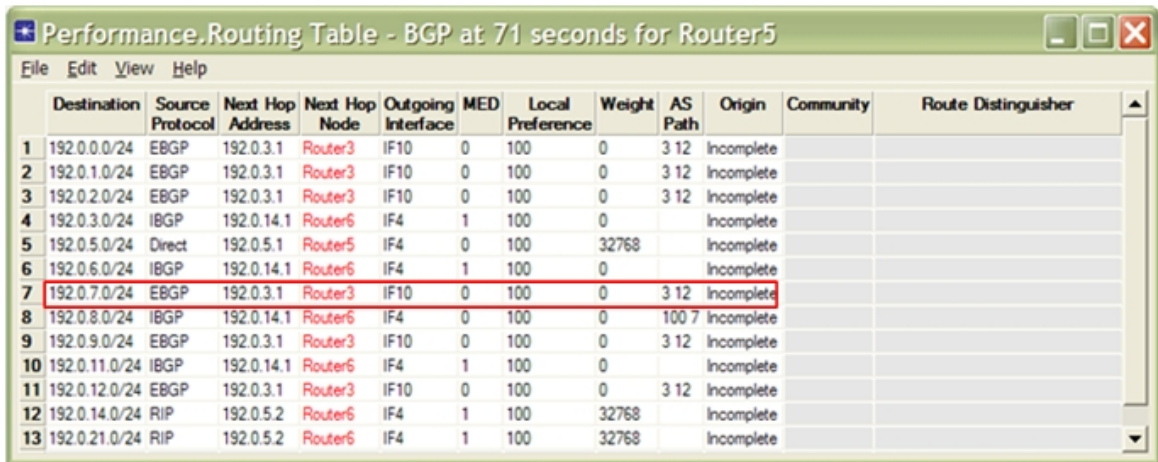
there is a traffic drop due to the malicious act, and then the traffic continues to pass but through Router 4 to its destination. To see the changes occurred, Figure 5-44 shows the Router 5 BGP routing table at the end of simulation.



	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4	Incomplete	[12:145]	
2	192.0.1.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4	Incomplete	[12:145]	
3	192.0.2.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4	Incomplete	[12:145]	
4	192.0.3.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
5	192.0.5.0/24	Direct	192.0.5.1	Router5	IF4	0	100	32768		Incomplete		
6	192.0.6.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
7	192.0.7.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4	Incomplete	[12:145]	
8	192.0.8.0/24	IBGP	192.0.14.1	Router6	IF4	0	100	0	100 7	Incomplete		
9	192.0.9.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4	Incomplete	[12:145]	
10	192.0.11.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
11	192.0.12.0/24	EBGP	192.0.6.2	Router4	IF11	0	100	0	4	Incomplete	[12:145]	
12	192.0.14.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
13	192.0.21.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		

Figure 5-44 BGP routing table of router 5 in shortening, localpref, malicious experiment

As shown in Figure 5-44, the route to prefix 192.0.7.0/b24 has its AS-Path set to 4 of length 1, although there are actually two ASes to reach this destination. But, because Router 4 has shortened the route, Router 5 preferred it. Figure 5-45 shows the BGP routing table of Router 5 at time 71, before the receipt of the shortened route through Router 4. This figure shows that for the same prefix, Router 3 was chosen as the next hop which has an AS-Path length of 2.



	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
2	192.0.1.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
3	192.0.2.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
4	192.0.3.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
5	192.0.5.0/24	Direct	192.0.5.1	Router5	IF4	0	100	32768		Incomplete		
6	192.0.6.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
7	192.0.7.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
8	192.0.8.0/24	IBGP	192.0.14.1	Router6	IF4	0	100	0	100 7	Incomplete		
9	192.0.9.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
10	192.0.11.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
11	192.0.12.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
12	192.0.14.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
13	192.0.21.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		

Figure 5-45 BGP routing table of Router 5 in shortening, local-pref, malicious experiment

Figure 5-46 shows the message passing between routers for this experiment.

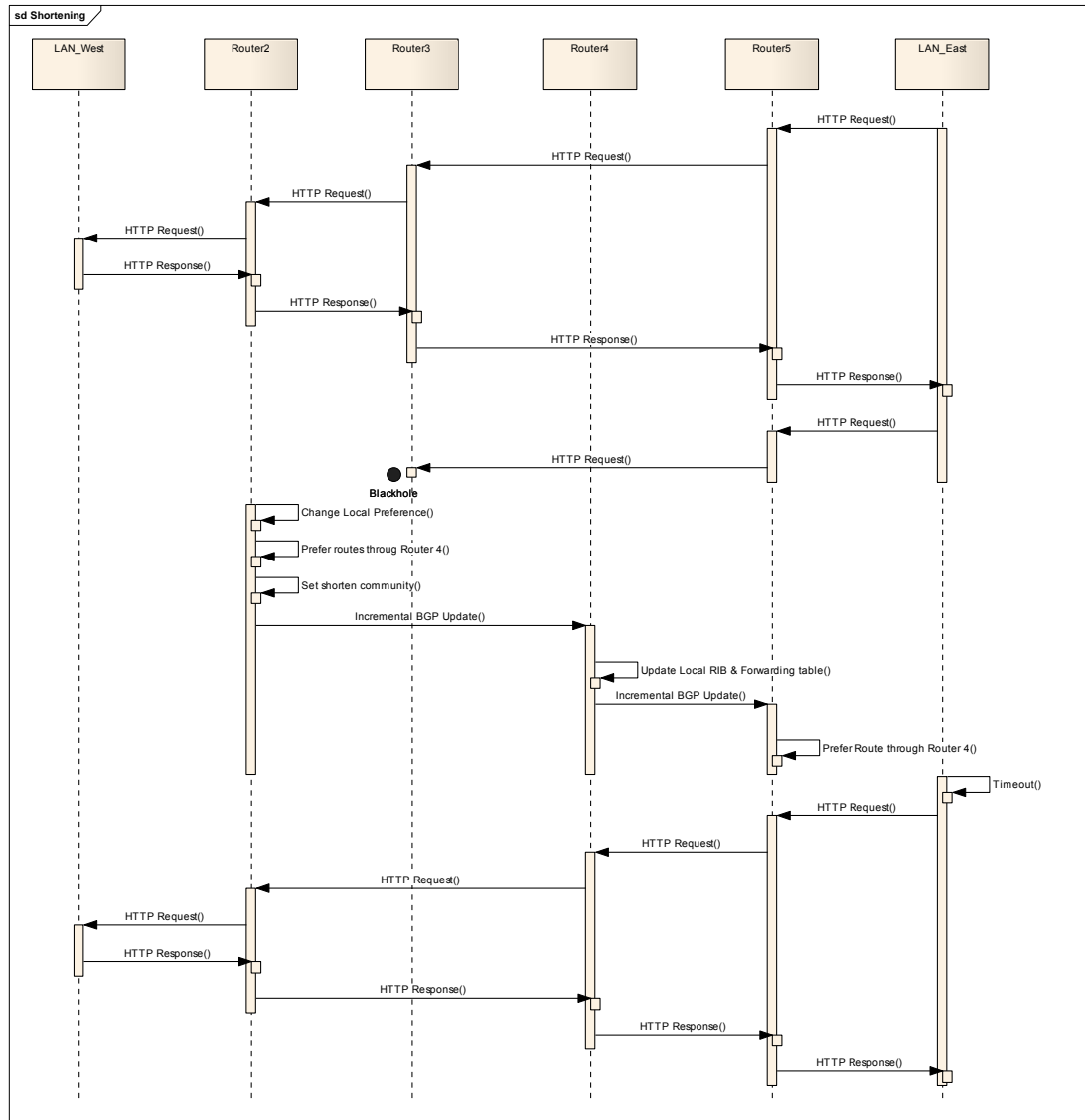


Figure 5-46 Message Passing in Shortening, LocalPref, and malicious experiment

As shown in the Figure 5-46, applying `local-pref` and shortening will switch the routing table of Router 2 and Router 5 to select the path through Router 4 for the communicating nodes.

Figure 5-47 shows the convergence activity for this configuration.

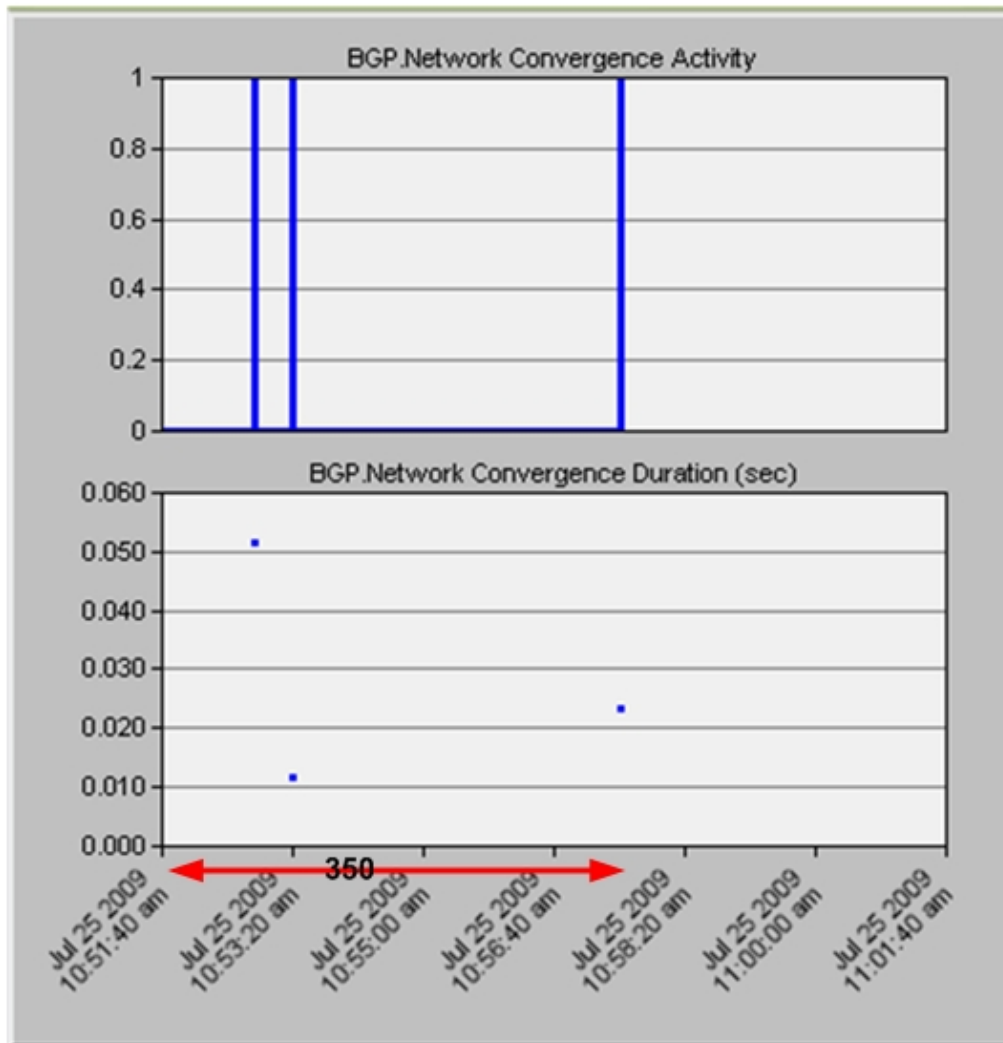


Figure 5-47 Convergence activity and duration for shortening experiment

As shown in Figure 5-47, the third line/dot represents the change that occurs in the middle of the simulation at time 350. It takes about 0.022 seconds to converge.

#### **5.5.2.2 More Specific Prefixes**

OPNET allows the user to configure routers to advertise certain prefixes even if they are not redistributed from other protocols. However, it does not allow specifying a neighbor to whom the advertisement should go. We modified OPNET implementation of BGP so that the user can specify the neighbor, the main prefix, and the more specific prefixes which will take the same route attribute values of the main prefix. Figure 5-48 shows the interface provided to the user to configure more specific prefixes. Refer to Appendix C section C.1.4 to see the code of more specific prefixes.

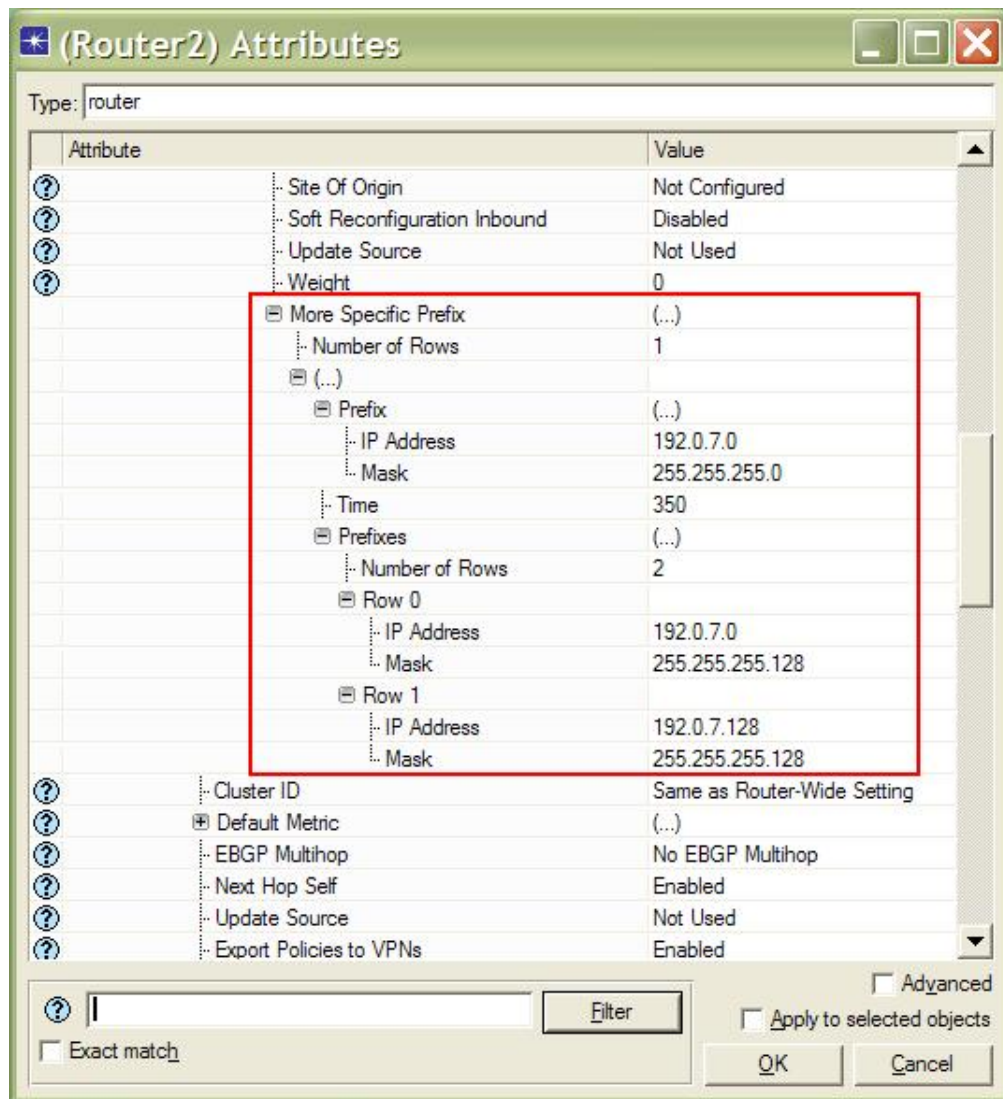


Figure 5-48 More specific prefix interface

The protocol will first search the 'rib-out' for the specified prefix. And then, it will create routes with prefixes assigned as more specific with the same route attribute of

the found route. Then, the more specific prefixes are added to 'rib-out' and sent to other neighbors. Figure 5-49 shows the procedure of implementing more specific prefixes.

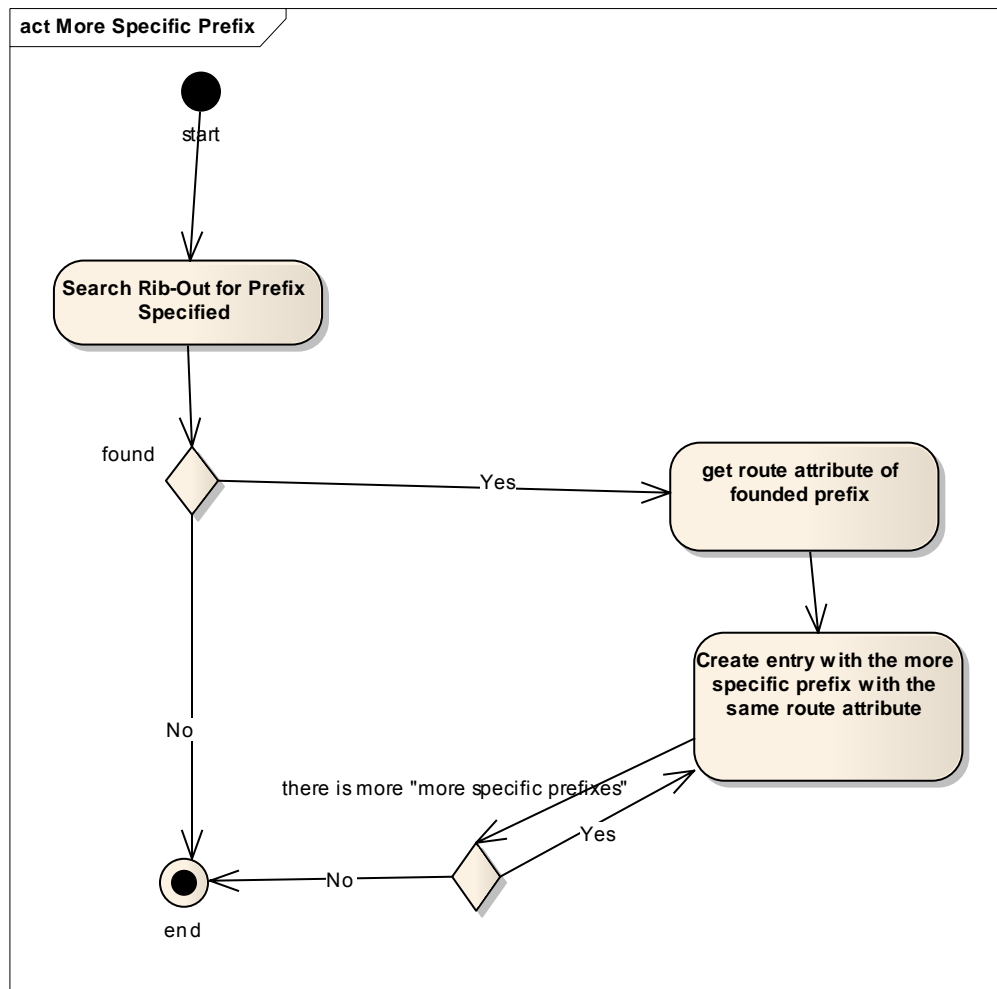


Figure 5-49 More specific prefix Implementation



In this experiment, the more specific prefix event and the increase local preference event are scheduled at the same time, i.e., 350, to control both the incoming and outgoing traffic. The malicious action starts at time (300). Figure 5-50 shows the incoming and outgoing traffic to Router 2 for both Router 3 and Router 4, in addition to the dropped traffic by Router 3 (the malicious router). The figure shows that the traffic changed its direction to the good IISP after a period of time during which there are some dropped packets.

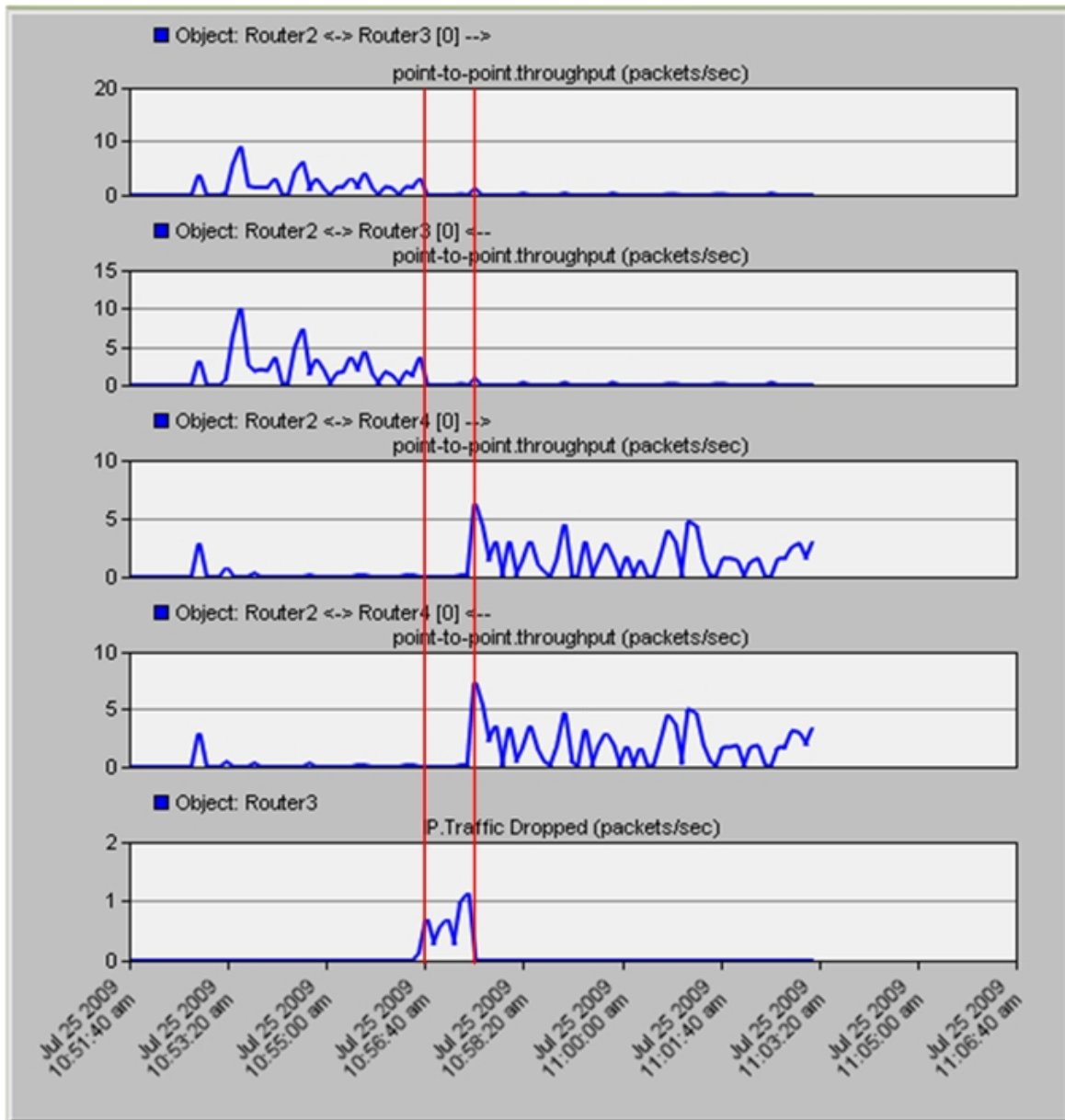


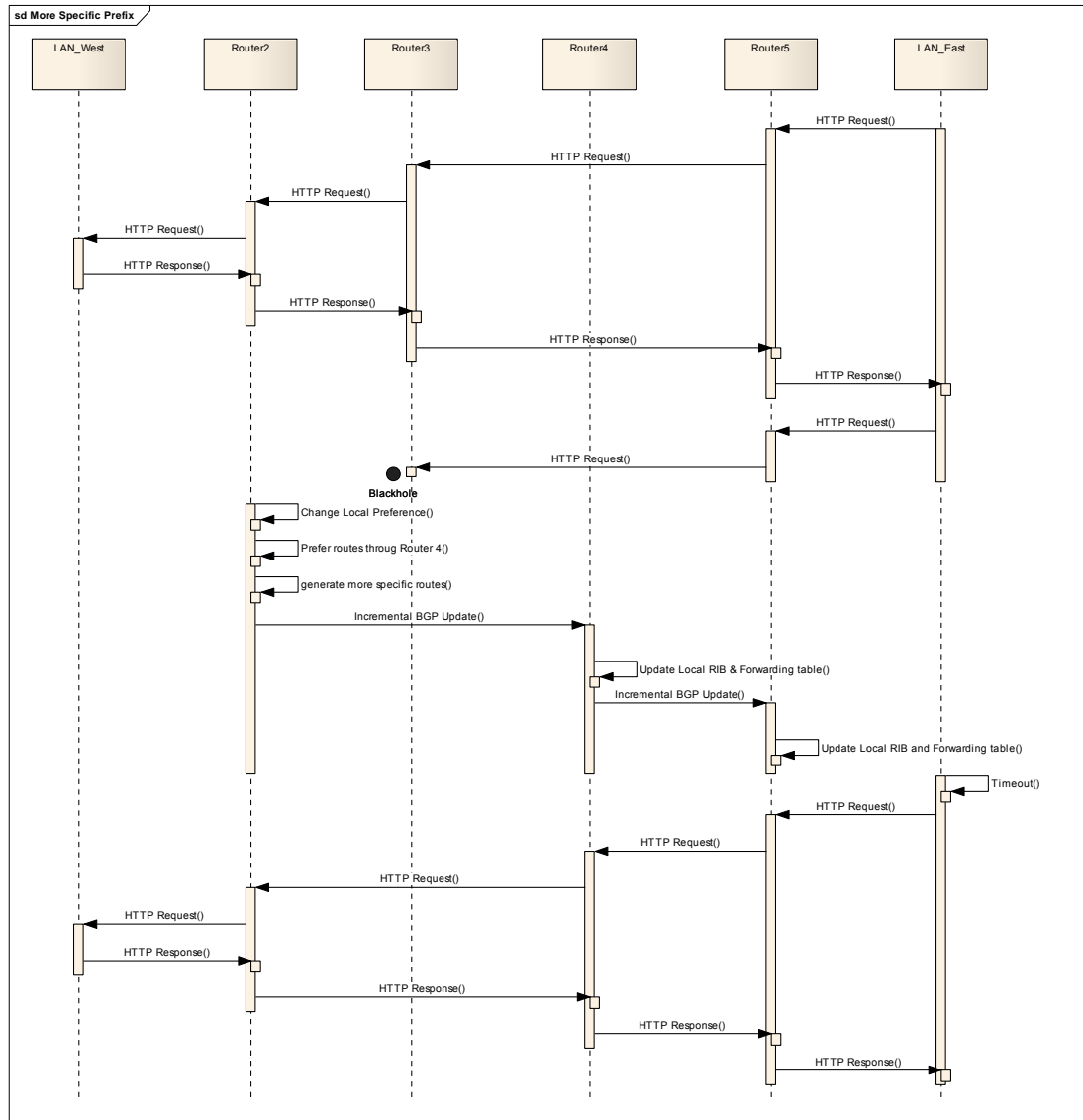
Figure 5-50 incoming and outgoing traffic of Router 2 in more specific, local preference, malicious experiment

Figure 5-51 shows the BGP routing table of Router 5. The table shows clearly the addition of the prefixes 192.0.7.0/25 and 192.0.7.128/25. These prefixes exist with the prefix 192.0.7.0/24, and because of the longest prefix match these prefixes are chosen in routing traffic to its destination.

Figure 5-52 shows the message passing in this experiment which is similar to the shortening message passing, except that at this time a new route that has its AS-Path shortened is advertised and added to the BGP routing table and the IP forwarding table.

	Destination	Source Protocol	Next Hop Address	Next Hop Node	Outgoing Interface	MED	Local Preference	Weight	AS Path	Origin	Community	Route Distinguisher
1	192.0.0.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
2	192.0.1.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
3	192.0.2.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
4	192.0.3.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
5	192.0.5.0/24	Direct	192.0.5.1	Router5	IF4	0	100	32768		Incomplete		
6	192.0.6.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
7	192.0.7.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
8	192.0.7.0/25	EBGP	192.0.6.2	Router4	IF11	0	100	0	4 12	Incomplete		
9	192.0.7.128/25	EBGP	192.0.6.2	Router4	IF11	0	100	0	4 12	Incomplete		
10	192.0.8.0/24	IBGP	192.0.14.1	Router6	IF4	0	100	0	100 7	Incomplete		
11	192.0.9.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
12	192.0.11.0/24	IBGP	192.0.14.1	Router6	IF4	1	100	0		Incomplete		
13	192.0.12.0/24	EBGP	192.0.3.1	Router3	IF10	0	100	0	3 12	Incomplete		
14	192.0.14.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		
15	192.0.21.0/24	RIP	192.0.5.2	Router6	IF4	1	100	32768		Incomplete		

Figure 5-51 BGP routing table of router 5 of more specific, local pref, malicious experiment



**Figure 5-52 Message Passing in More specific, local pref, malicious experiment**

Figure 5-53 shows the convergence activity of this experiment.

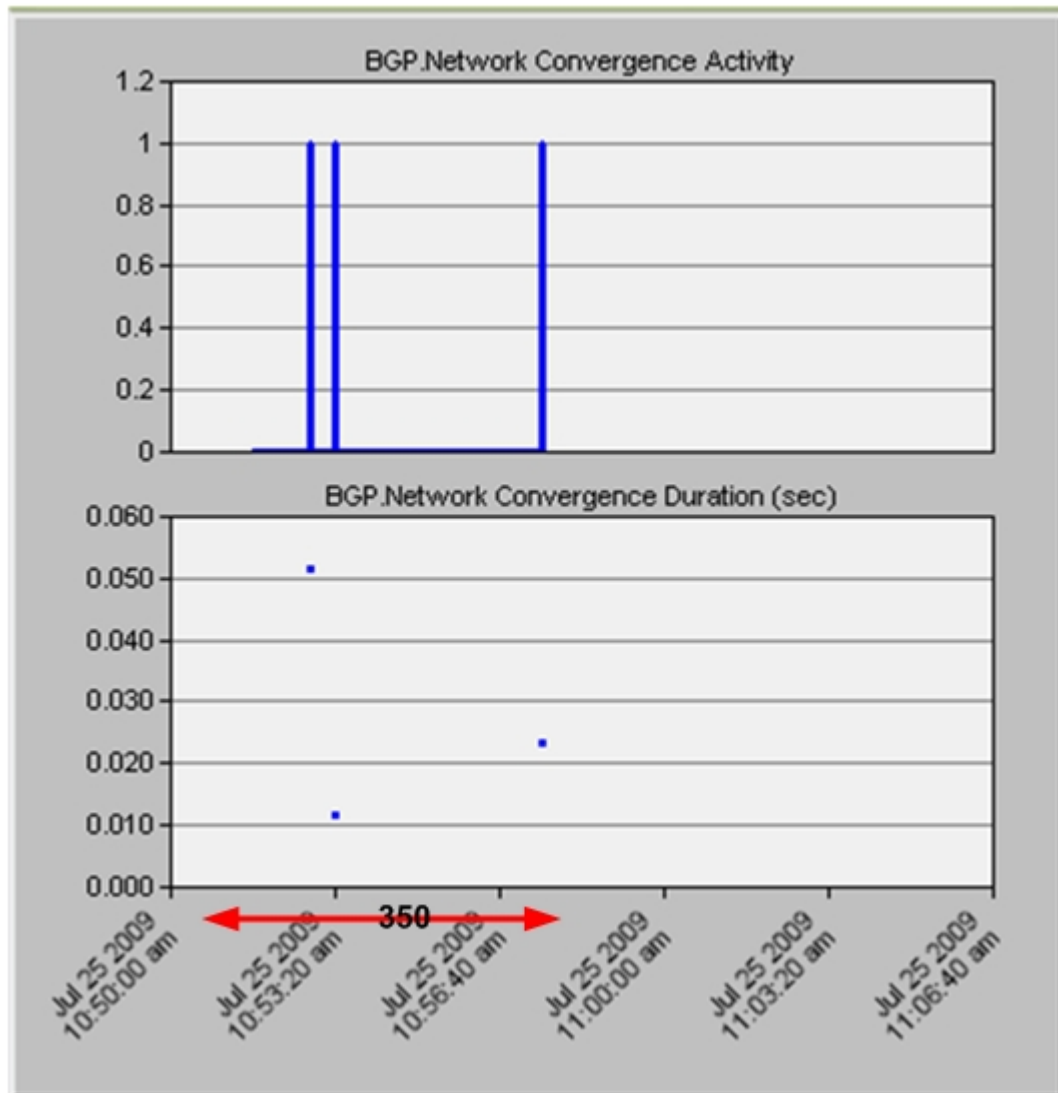


Figure 5-53 Convergence activity and duration of more specific, local pref, and malicious experiment

The third line/dot in Figure 5-53 represents the convergence of the network for this experiment. It takes about 0.022 seconds for the network to converge.

## **5.6 SUMMARY**

This chapter describes the use of OPNET to simulate BGP tuning techniques for solving the malicious problem. It shows our modification of the OPNET code in addition to its impact on convergence. All the methods and the modifications are supported by validation scenarios which include throughput, drop rate, convergence time, and routing tables.

---

## ***CHAPTER 6***

### ***PERFORMANCE EVALUATION OF BGP TUNING TECHNIQUES TO CIRCUMVENT MALICIOUS ACT***

#### ***6.1 INTRODUCTION***

In CHAPTER 5, the implementation and validation of BGP tuning techniques to solve the Internet blocking by the malicious IISP are presented. In this chapter, the focus is on the performance evaluation of different BGP tuning based solutions with different network configurations in terms of Internet delay, traffic type, and network loads.

The same network topology used in the last chapter will be used in this performance evaluation. Figure 6-1 shows the network topology that will be used in the evaluation. In this chapter, the blackholing starts at time 300, the solution starts at time 360, and the total simulation duration is 2000 seconds. The link used in performance evaluation between routers is **PPP\_DS1** that has a data rate of 1.544 Mbps that is lower than the **PPP\_DS3** link used in the previous chapter and which has a data rate of 44.736 Mbps.

This change is done in order to study the effect of the network load the network within an acceptable simulation time.

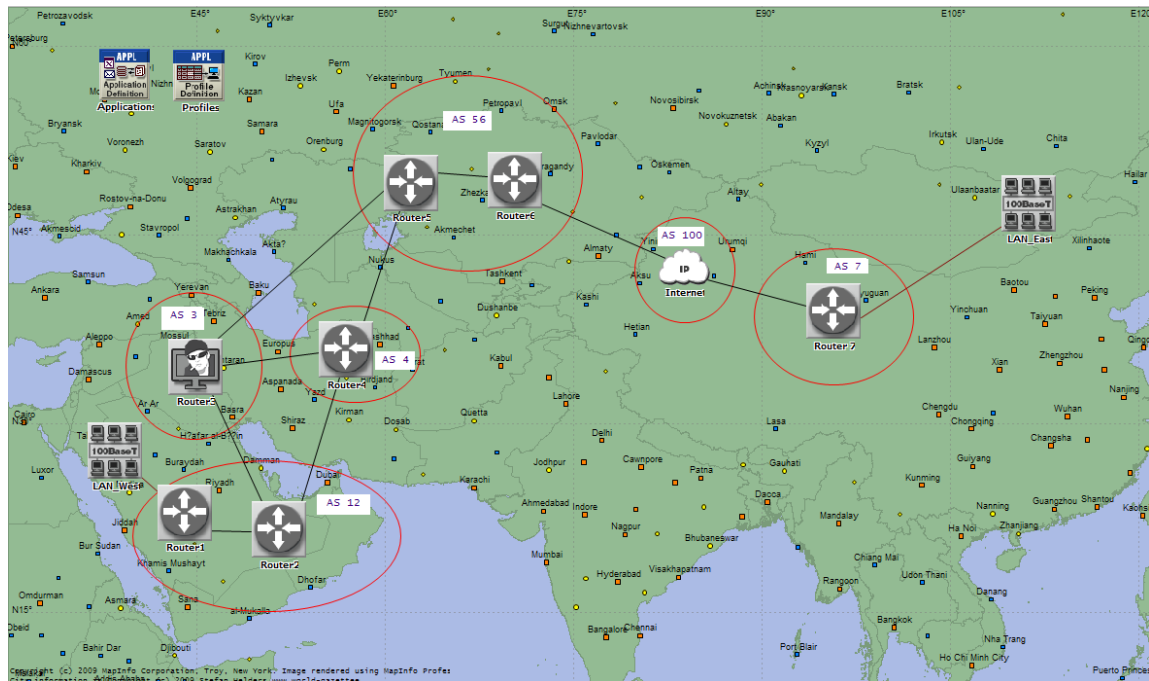


Figure 6-1 Evaluation Network Setup

The solutions that will be evaluated are the use of community, the use of shortening, and the use of more specific prefixes to control incoming traffic. These methods are combined with the setting of the `local-pref` attribute in order to control the outgoing traffic. The delay of the Internet node modeled by the `IP_cloud` follows an exponential distribution with a mean of either 0.1 seconds or 5 seconds is used to



represent two extreme cases. The applications used to generate the traffic are HyperText Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Voice over Internet Protocol (VOIP). Both HTTP and FTP run over TCP protocol. The links connecting the IP cloud to Router 6 and Router 7, see Figure 6-2, are loaded at 20%, 50%, and 80%. So knowing the link capacity, one can configure the links to be loaded with specific traffic amount equivalent to any of the three percentages.

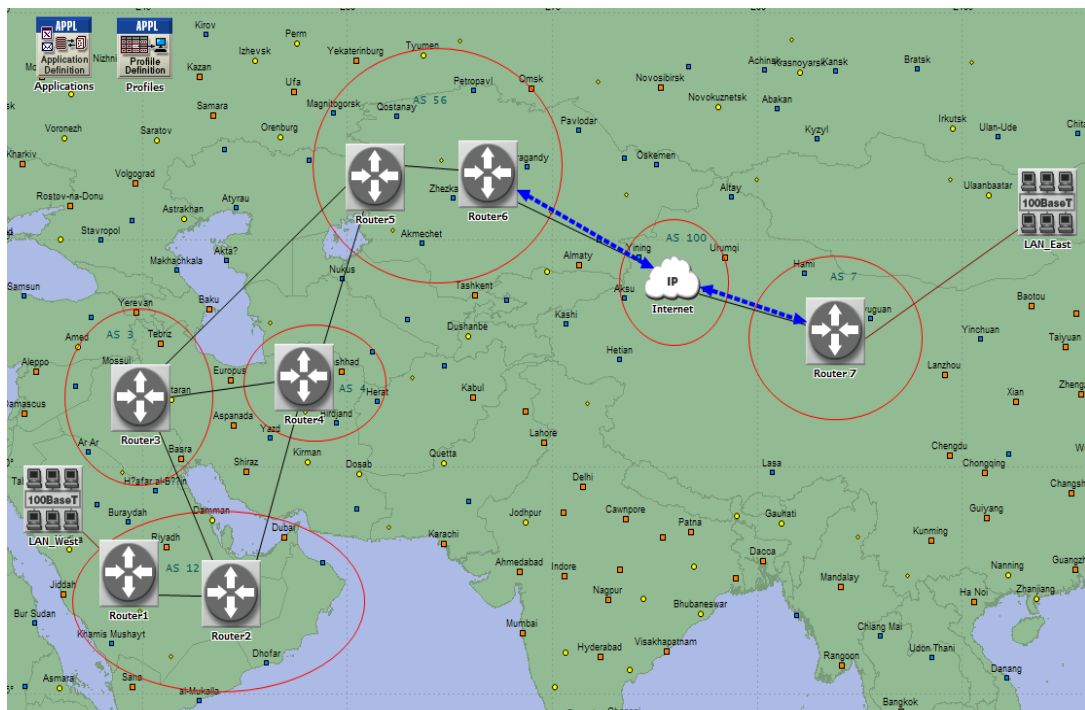


Figure 6-2 Network showing the links that will be loaded with traffic.

A number of outcomes are investigated in the evaluation. The first outcome is the percentage of packet drop to the total number of application packets that are generated during 2000 seconds of simulation time. The time of blackholing is 60 seconds starting at 300, and the solution is applied at time 360. The convergence time of BGP protocol in the whole network is the second outcome. The third outcome is the throughput in the link between Router 2 and Router 3 and the link between Router 2 and Router 4 in both directions. Finally, application specific outcome includes the traffic sent from and received to LAN\_East and the response time for an HTTP page and an FTP download. These results are generated for each proposed solution, for each application or traffic type, for each link load, and for each delay of the Internet. Each experiment runs for 20 times in HTTP and FTP applications and for 5 times in VOIP application; and either the mean or the mean and the confidence interval are displayed in the figures. The results for the VOIP application when the delay is exponential with 5 second as mean and when the load on the link is either 50% or 80% are excluded, because the simulation has some problems. For more information on this, refer to section 6.3. Table 6-1 shows a list of output figures, a brief definition and why it is chosen.

Table 6-1 List of output figures, brief definition and why they are picked

Outcome	Figures	Definition	Why Chosen
<b>Percentage of Packet drop</b>	Figure 6-3	The number of packets that are dropped over the total number of packets sent in both directions	Packet drop is the main drawback of blackholing and so it is chosen to see how the routers behave during the time of blackholing
<b>Convergence time</b>	Figure 6-4 and Figure 6-5	The time until all Routers update their BGP tables according to the BGP update messages that are part of applying the solution	Since the solution is BGP based, the convergence time is an important measurement to study the effectiveness of the solution and to study the behavior of the solutions in different configuration scenarios
<b>Throughput</b>	Figure 6-6 to Figure 6-17	The throughput in bits per second is studied in links between router 2 and router 3 and between router 2 and router 4 in both directions	Since these two links are the links connecting the regional ISP to the good and malicious IISP, we are interested in studying the behavior of the link in both directions
<b>Traffic Sent/Received (application level)</b>	Figure 6-18 to Figure 6-23	The application packets that are sent from the LAN_East and the application packet received as seen by LAN_East	To see the effect of blackholing and solution on different applications
<b>Response Time</b>	Figure 6-24 and Figure 6-25	For HTTP and FTP applications	To see the effect of receiving a page in HTTP or downloading a file in FTP at the time of blackholing and when the solution is applied

---

## 6.2 *COMPARISON*<sup>3</sup>

### 6.2.1 *Percentage of traffic drop*

Figure 6-3 shows the percentages of packet drop for different solutions. The x-axis displays the experiment configuration and the y-axis displays the percentage of packet drop. In the x-axis there are three-valued configuration parameters. The first parameter is the application type, the second one is the mean of the exponentially distributed delay of the Internet, and the third one is the link load. The type of solution applied is displayed in the legend in the bottom of the figure. The vertical bars are the confidence intervals of the readings with 95% confidence Interval.

---

<sup>3</sup> Refer to Appendix A to see the application default configurations for TCP, HTTP, FTP and VOIP.

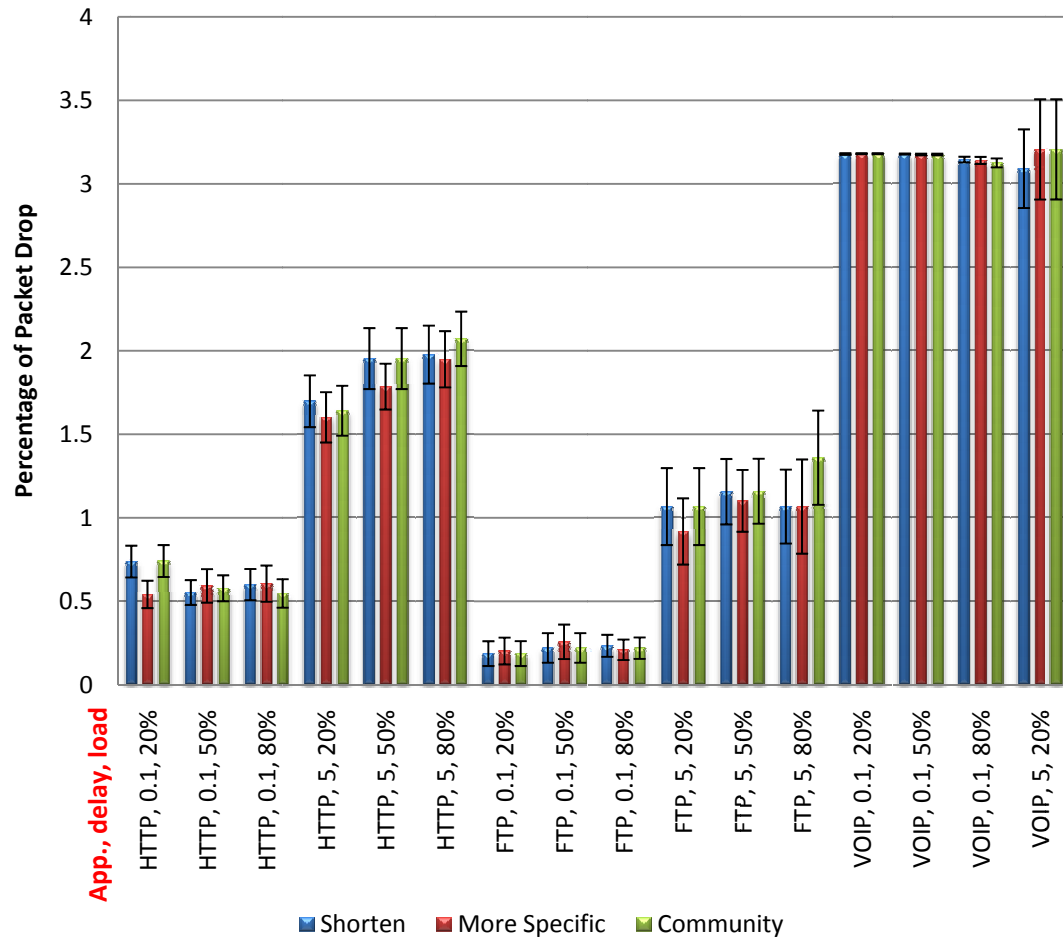


Figure 6-3 Packet drop percentages

The VOIP application has the highest percentage of packet drop, as it is about 6 times the percentage of packet drop in HTTP application with 0.1 delay of the Internet. This is because VOIP continues to send traffic at the time of blackholing at the same rate it did before the blackholing. This happens since VOIP is a real-time application which runs

over UDP and does not wait for an acknowledgement to send the next packet. Therefore, this results in a higher percentage of packet drop. The confidence interval of VOIP application for the 0.1 mean delay of the Internet is very small because of the consistency of packet sending irrespective of blackholing. However, the confidence interval of the percentage of packet drop for the VOIP application increases when the delay is 5 seconds because of the increase of the randomness<sup>4</sup> of the delay that each packet experiences. This also applies to all experiments with 5 seconds delay. The percentage of packet drop in HTTP is about double the percentage of packet drop in FTP application in both 0.1 second delay and 5 seconds delay. This difference is due to the nature of each application and how it is configured. To illustrate this, the interarrival time for HTTP messages is set to 60 seconds as a mean, while the inter-request time for FTP has a mean of 360 seconds as shown in Figure A-2 and Figure A-6 respectively. So, the difference is large in terms of interarrival. Increasing the delay of the Internet results in more packet drop, e.g., in HTTP, the 5 seconds delay causes the packet drop to be about three times more. The reason is that TCP adjusts its retransmission based on the Round Trip Time. The increase of the delay increases the Round Trip Time and that

---

<sup>4</sup> Because when the mean is 5 seconds in the exponential distribution, the standard deviation will be also 5 seconds and this increases the randomness of the delay.

makes the TCP detection of packet loss slower. Hence, the behavior of TCP to reduce the rate at which packets are sent is delayed resulting in more packet drop. Moreover, due to the delay of the Internet node, more packets will be in the network from the time the blackholing starts, and this will result in more packets being dropped. Also the delay of the Internet increases the convergence needed for BGP, resulting in more time the packets continue to pass through the malicious node and resulting in more packet drop. Assigning different loads has insignificant effect on the percentage of packet drop. This is because only two links are loaded on the network and also even when the links are 80% loaded, this will leave space for all other traffic on the network that the experiments need. Some of the experiments show slight increase in packet drop due to adding more load, and this is clear in the HTTP application when the delay of the Internet is 5 seconds. Different solutions have similar effects in terms of percentage of packet drop.

### ***6.2.2 Convergence Time***

Figure 6-4 shows the convergence time for the 0.1 seconds as a mean delay of the Internet. The x-axis displays the experiment configuration and the y-axis displays the convergence time needed in seconds. The vertical bars depict the 95% confidence interval. It is worth mentioning that when the delay of the Internet is of a certain mean,

BGP message appears to experience this delay twice. This is because when a new advertisement to the Internet node arrives the delay is experienced. And, if as a result of this advertisement a another advertisement need to be sent another delay is imposed. As shown in Figure 6-4, lower convergence times are exhibited for the more specific prefix solution than the shortening and community solutions. This is because of the way the solution is implemented. In shortening and community techniques, the algorithm look at 'rib-out' in order to know which prefix needs to be advertised. In more specific prefixes technique the user only enters the prefix of the LAN\_East. So only two more specific prefixes needs to be advertised compared to six prefixes in shortening and community solutions. See Figure 5-38 for community, Figure 5-44 for shortening, and Figure 5-51 for more specific prefixes. In practice, shortening and community need not to advertise more than one prefix for this example. However, the more specific prefix solution might need to advertise more than one to cover the whole range of the prefix.



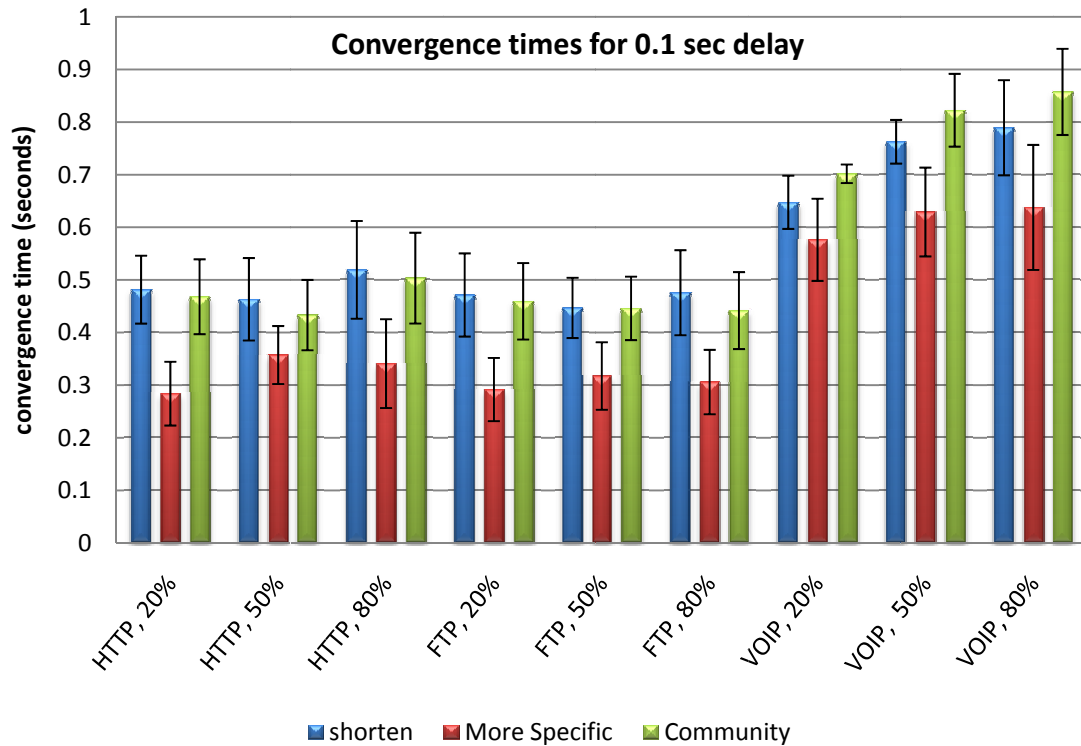


Figure 6-4 BGP convergence time for 0.1 seconds delay of Internet.

Figure 6-4 also shows that the effect of loading the two Internet node links on the convergence time is minor. This is because only two links are loaded and also because only one type of traffic is present at any time, and that does not need much of the link capacity. A slight increase in convergence time can be noticed with the increase of load especially in VOIP. VOIP shows this increase because of its high traffic demand which causes some delay in messages transferred across the network. Since, VOIP generates

considerably larger number of packets than HTTP and FTP applications, so VOIP has the highest convergence time. This is normal since VOIP simulates a voice call which generates more traffic and the total throughput will be more than FTP and HTTP. Look at A.2, A.3, and A.4 for HTTP, FTP and VOIP configurations sequentially.

Figure 6-5 shows the time needed to converge in case the mean of Internet delay is 5 seconds. The x-axis displays the experiment configuration and the y-axis displays convergence time needed in seconds. The vertical bars depict the 95% confidence interval. A very high increase in convergence time is obtained with comparison to the case for the 0.1 seconds delay of the Internet. The main reason is the large difference between the two values, i.e., 5 seconds and 0.1 seconds as delays. In addition, the increase in randomness of the 5 seconds delay<sup>5</sup> causes higher convergence time. Due to large delays different BGP messages experience, the MRAI timer will be triggered adding another 30 seconds of delay.

---

<sup>5</sup> Due to the use of exponential distribution which has the standard deviation equals to the mean of 5 seconds.

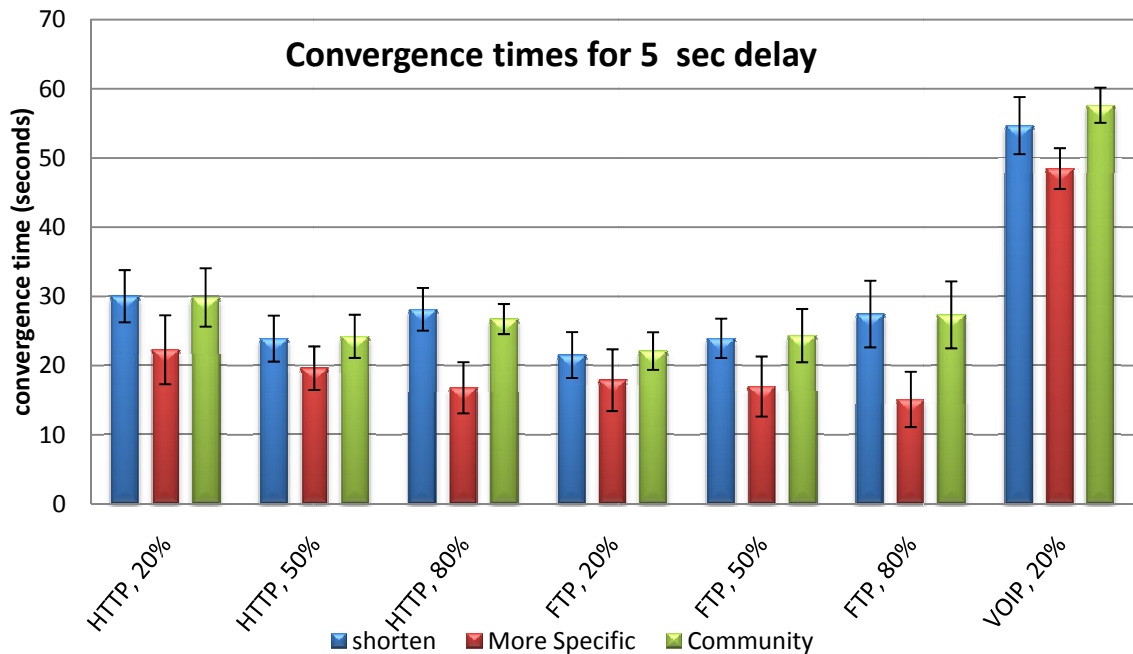


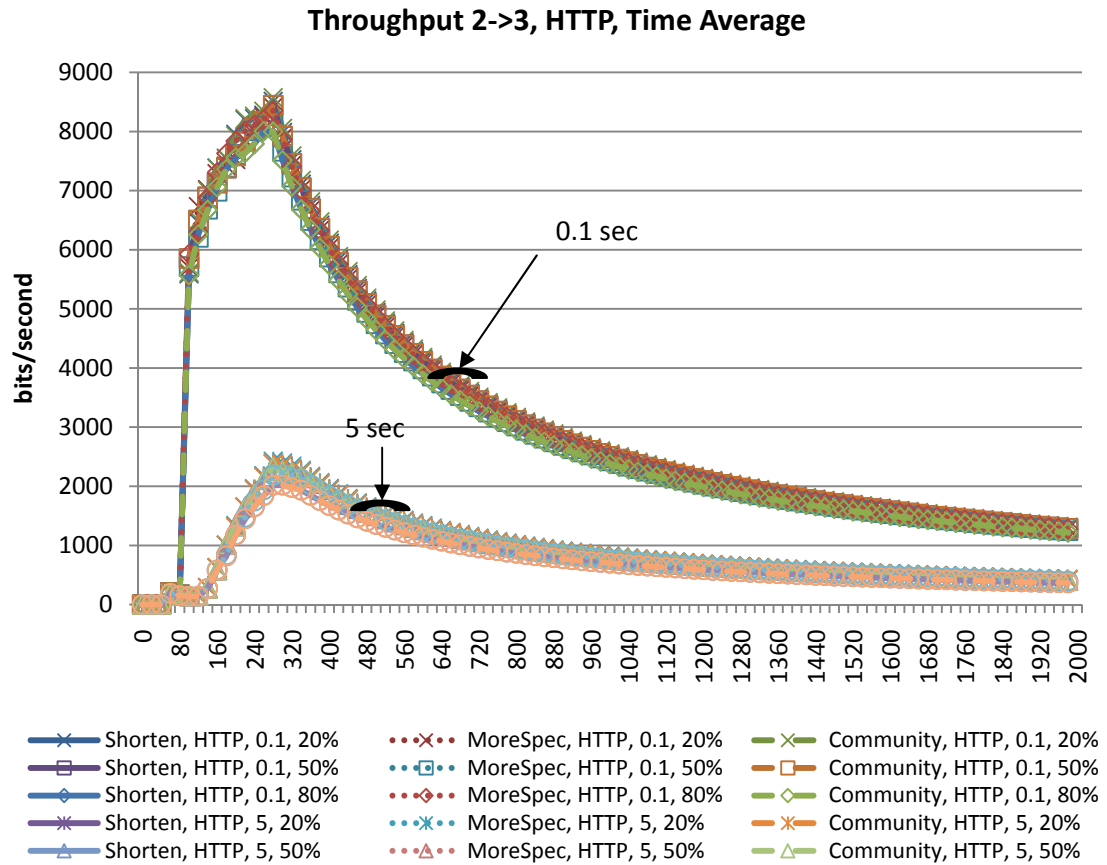
Figure 6-5 BGP convergence time for 5 second delay

Figure 6-5 also shows that the more specific prefixes solution needs less time to converge than the other two solutions. In the FTP application, the higher the link load, the higher the convergence time. In general, the effect of the load on the two links is minor. The time needed to converge in VOIP application is about 2 times the time needed for HTTP and FTP applications; and this is because of the high bandwidth consumed by the VOIP application.

### ***6.2.3 Throughput***

In this section, the throughput in bits per second is studied for the two links that connect Router 2 to the two providers is presented. The throughput is investigated in both directions. A figure is provided for each application. The effect of applying the solution can only be seen in the throughput between Router 2 and Router 4, the good IISP. The throughput between Router 2 and Router 3 show the effect of blackholing on different network configurations. In these figures, only the mean throughput of multiple runs is drawn. Confidence Intervals are available but are not drawn for the purpose of not distracting the figure more. Refer to Appendix B section B.1 for baseline throughput, i.e. the throughput results without applying neither malicious activity nor applying solutions.

Figure 6-6 shows the outgoing throughput, in bits/second, from Router 2 to Router 3 for the HTTP application.



**Figure 6-6 Outgoing throughput from Router 2 to Router 3 for HTTP.**

Figure 6-7 shows the throughput in bits/second for the FTP application configurations also from Router 2 to Router 3.

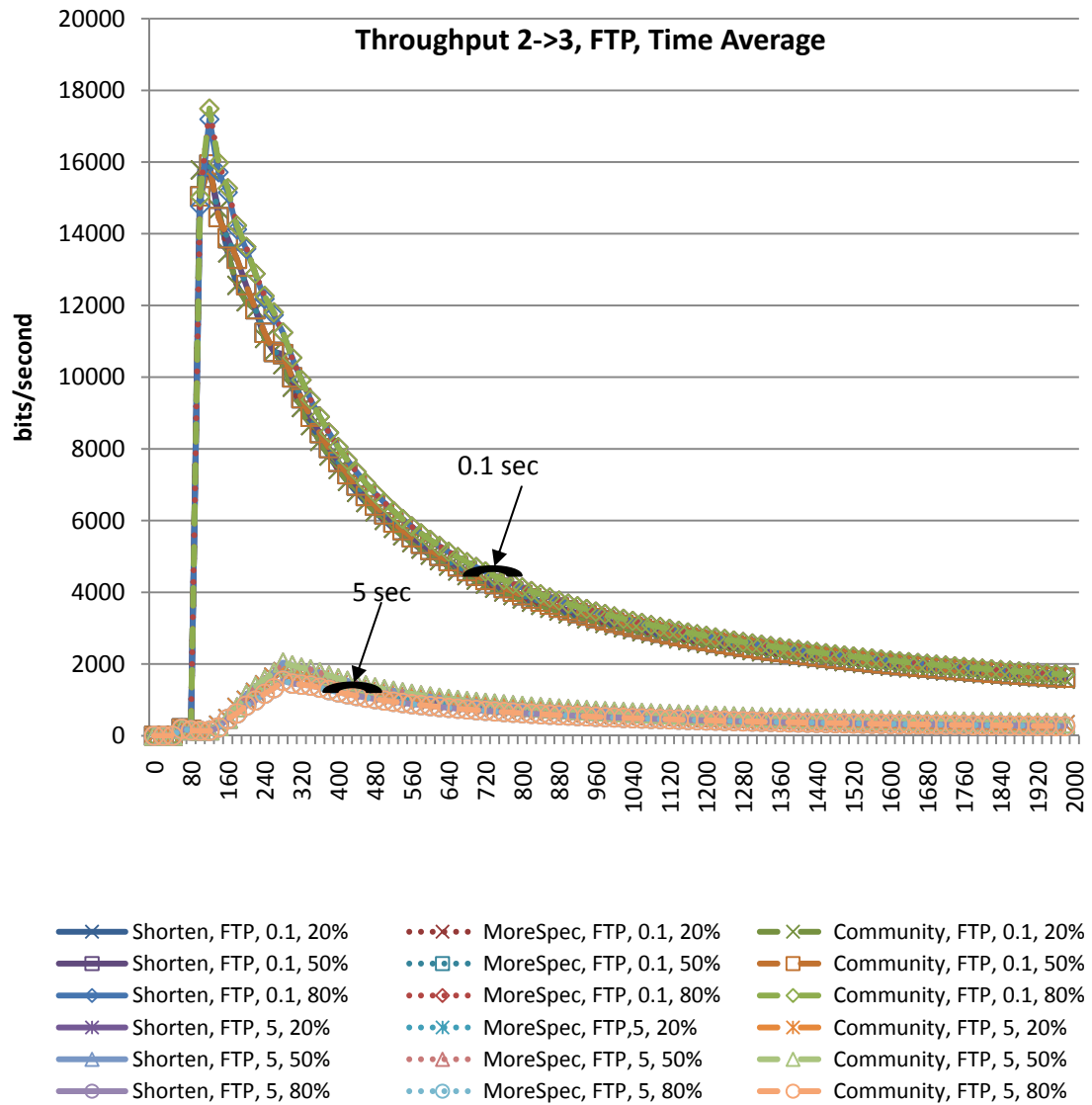
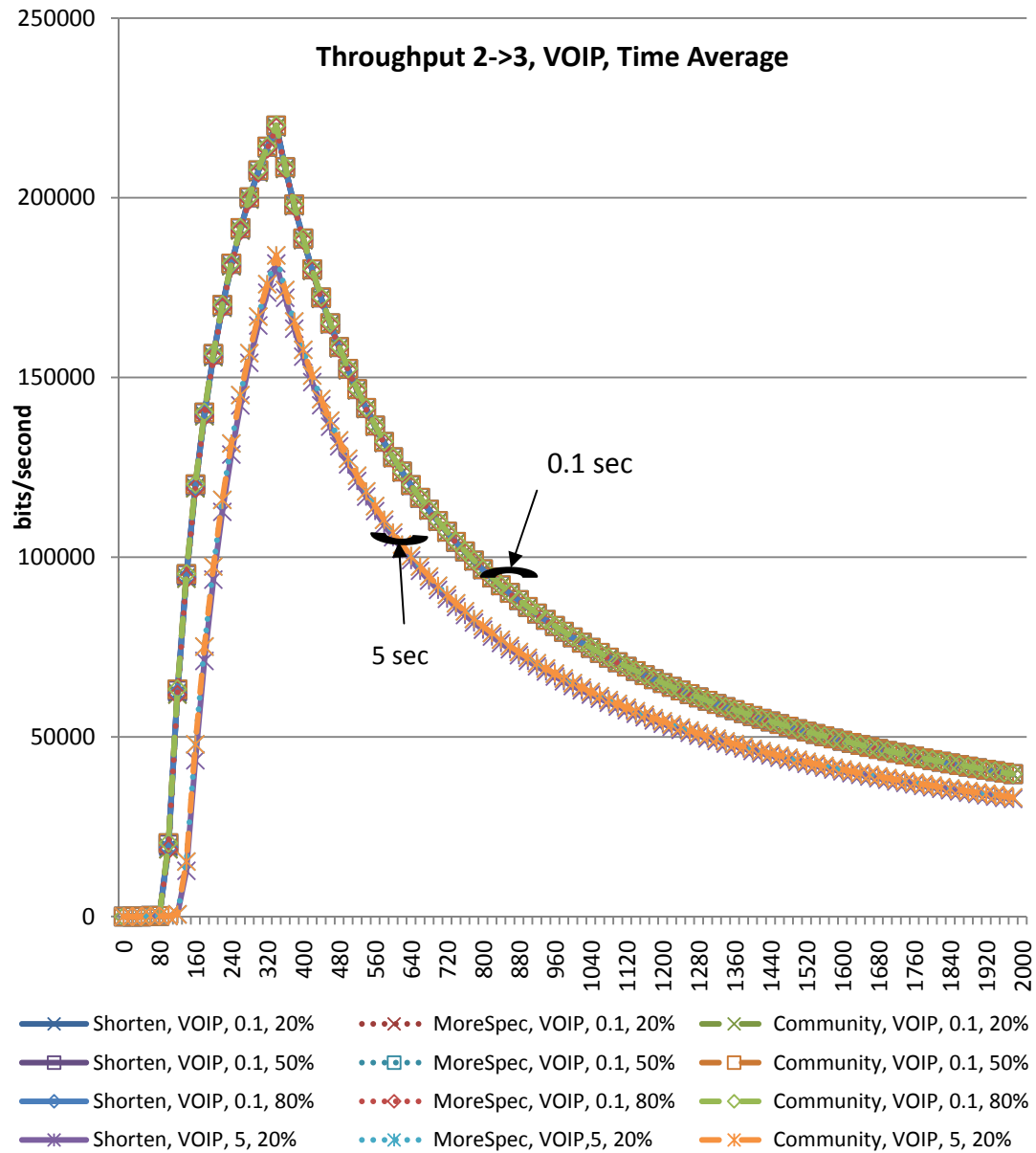


Figure 6-7 Throughput from Router 2 to Router 3 for FTP application

Figure 6-8 shows the throughput in bits/second for the VOIP application configurations also from Router 2 to Router 3.



**Figure 6-8 Throughput from Router 2 to Router 3 for VOIP application**

From Figure 6-6, Figure 6-7, and Figure 6-8, it can be noticed that the traffic has the same pattern with close values except when the delay of the Internet is different. Throughput is inversely proportional to the time, and the increase of the Internet delay makes the throughput less. This is mainly due to TCP congestion control which slows down the transmission rate when the traffic experiences more delay. Referring to Figure 6-8, the outgoing traffic from Router 2 to Router 3 in the VOIP application with 5 seconds delay of the Internet is shown to be less because of the slower initial time to converge when the delay is more. This results in postponing when the VOIP starts sending.

The following set of figures shows the throughput for different applications from Router 3 to Router 2. Figure 6-9 shows the throughput from Router 3 to Router 2 for the HTTP application configuration.



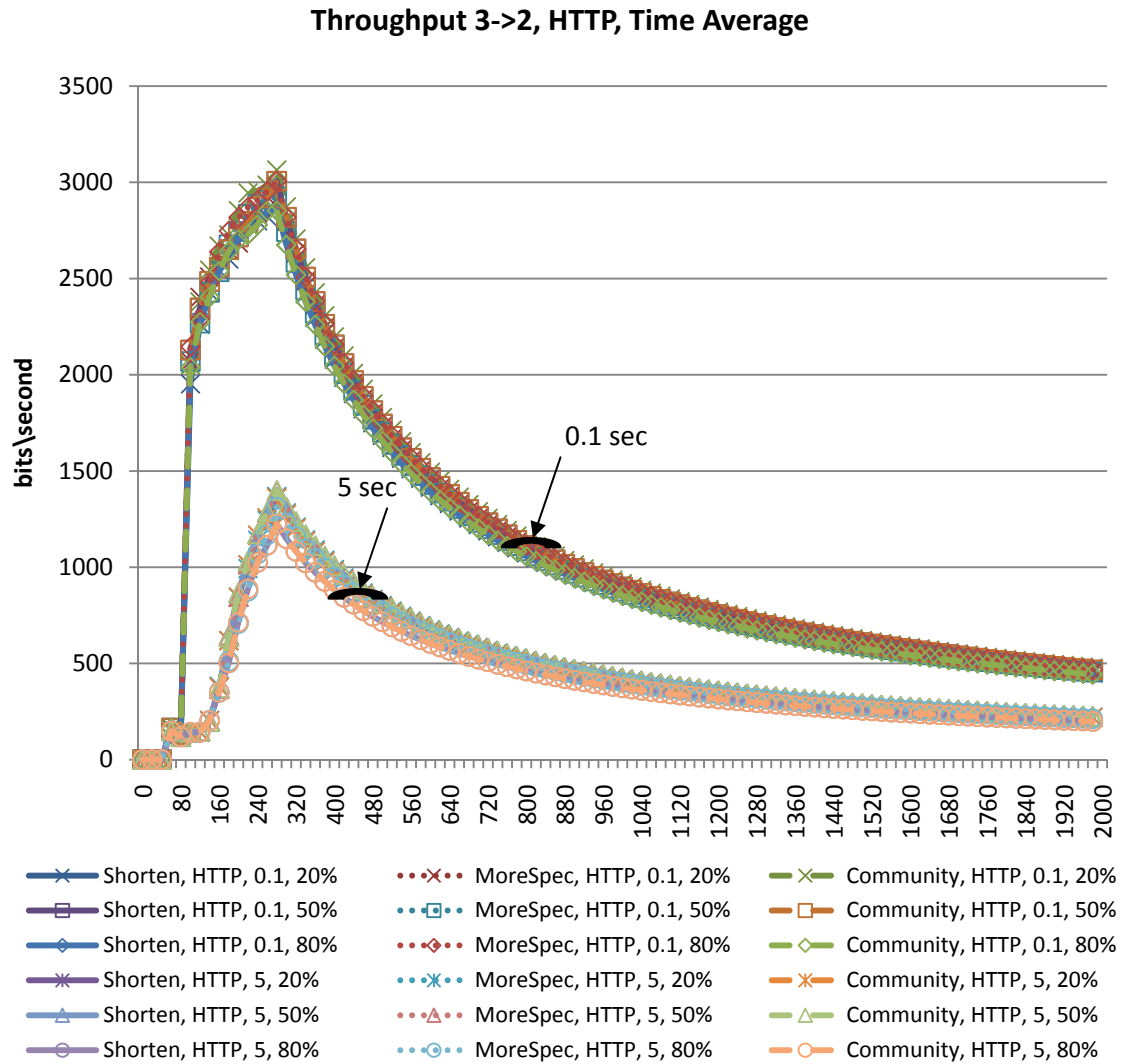


Figure 6-9 Incoming Throughput to Router 2 from Router 3.

Figure 6-10 shows the throughput from Router 3 to Router 2 in bits per second for the FTP application. The figure shows that higher throughput is exhibited when the delay

of the Internet is less. The delay of the Internet node affects the initial time needed for convergence. And also the increase of the time a packet experiences in the Internet reduces the rate at which the packet arrives. This results in the difference in throughput.

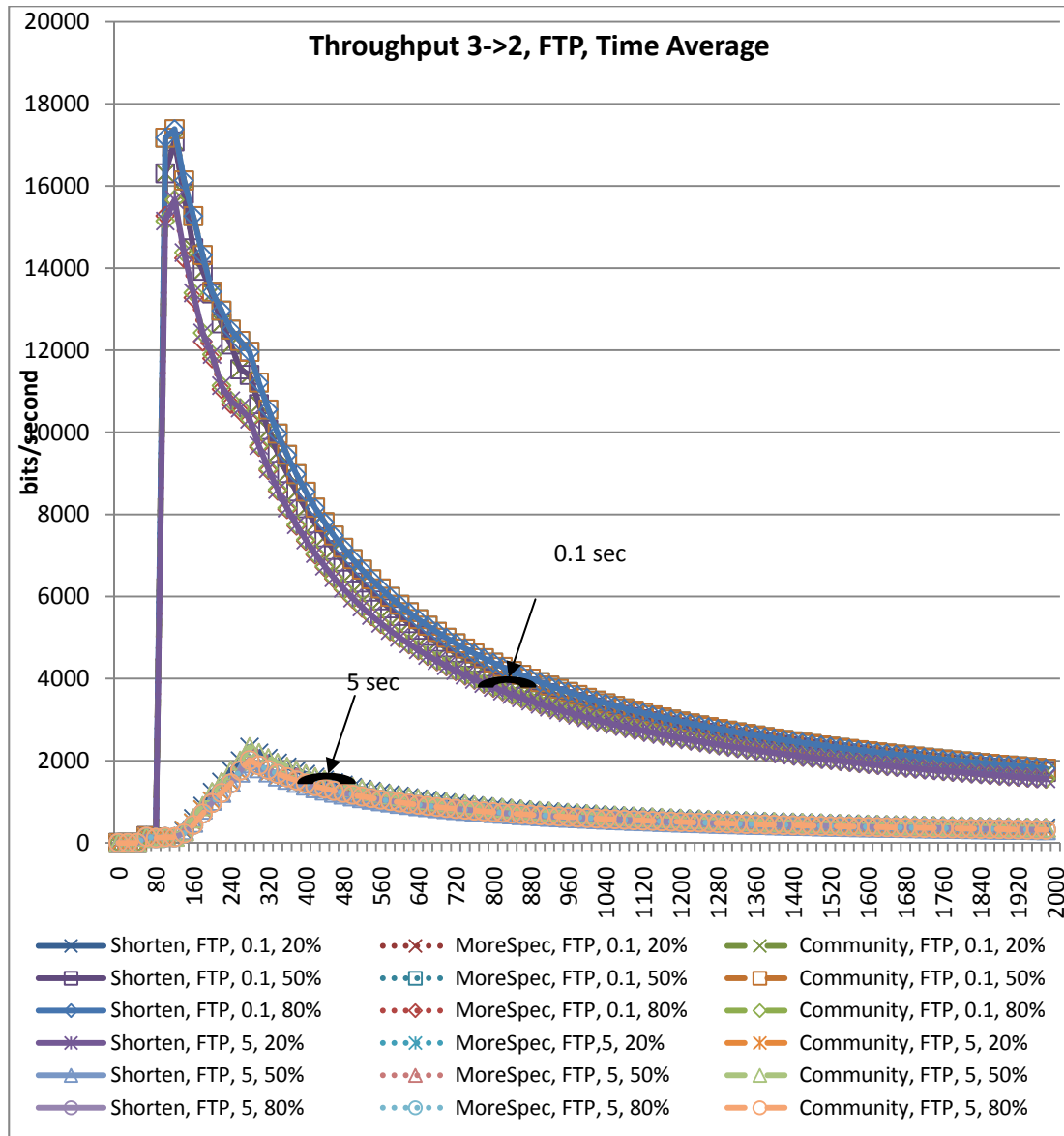


Figure 6-10 Throughput from Router 3 to Router 2 in FTP application

Figure 6-11 shows the throughput from Router 3 to Router 2 in the VOIP application. Like in the HTTP and FTP applications, a lower throughput is exhibited when the delay of the Internet is higher.

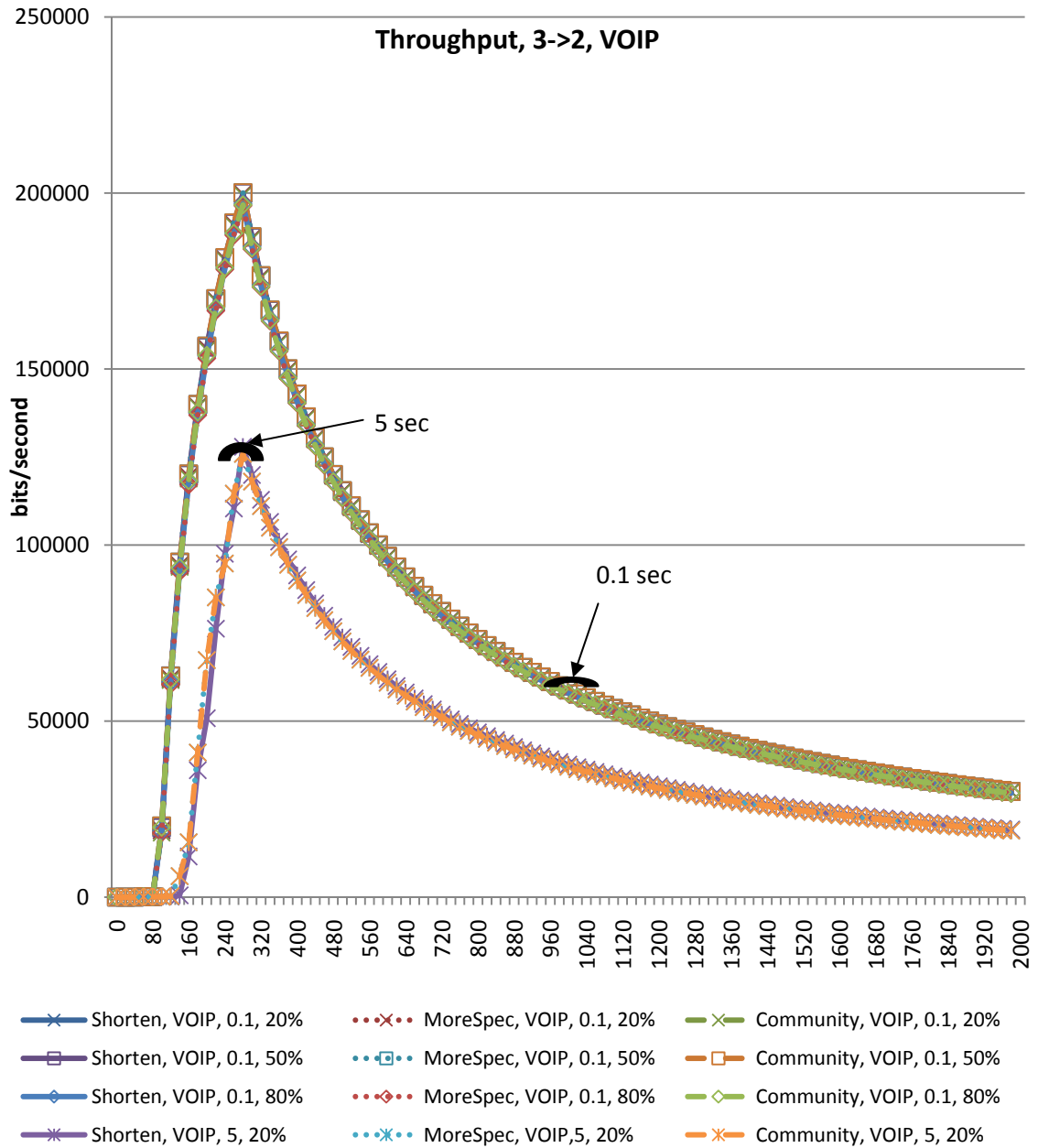


Figure 6-11 Throughput from Router 3 to Router 2 for VOIP application

Figure 6-6 through Figure 6-11 show that the 'time averaged' throughput decreases at time 300 because blackholing starts at this time. The solution is applied at time 360. The following figures show how starting a solution will enable the traffic to transfer through Router 4.

The following set of figures shows the throughput from Router 2 to Router 4. Figure 6-12 shows the throughput from Router 2 to Router 4 in bits per second for HTTP application. The throughput for 0.1 delay of the Internet reaches 8500 bits per second<sup>6</sup> while the throughput for the 5 seconds delay stabilize at about 4000 bits per second which is less than half the throughput of the 0.1 delay of the Internet. Different solutions exhibit a similar behavior in terms of outgoing traffic through Router 4.

---

<sup>6</sup> If it left to continue it will reach 10000 bits per second refer to Figure B-1 in Appendix B.

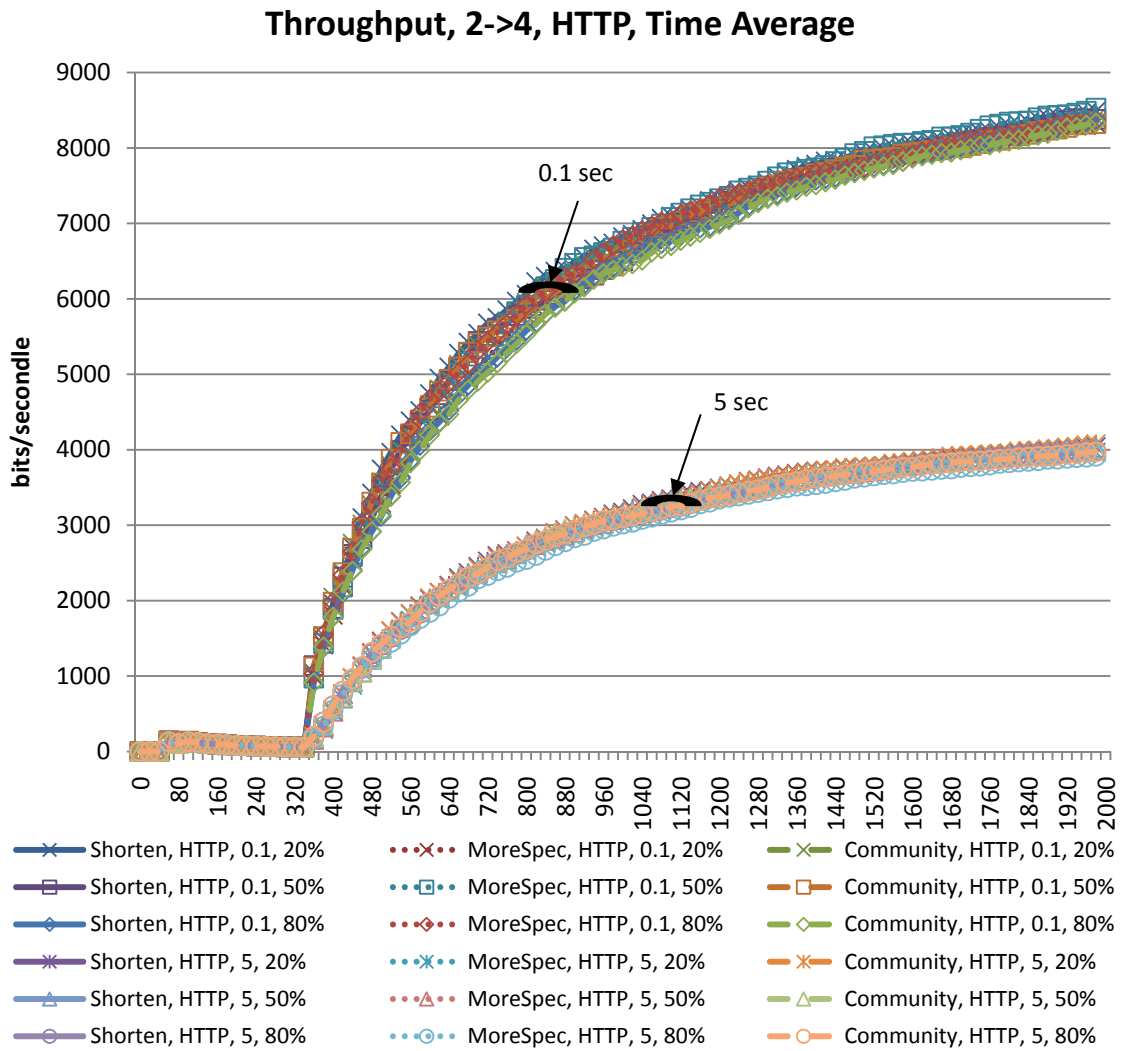


Figure 6-12 Outgoing traffic from Router 2 to Router 4 for HTTP application.

Comparing Figure 6-12 to Figure 6-6, we notice that the traffic of the 0.1 seconds goes up to 8000 bits per second in both figures. However, for the case of the 5 seconds delay, the throughput goes up to about 2000 bits per second in Figure 6-6 and then goes down which is different than the level it reached in Figure 6-12 of 4000 bits per second. The reason is because of blackholing that occurs before the maximum throughput level reached its maximum value. This is due to the high initial convergence time needed for the 5 seconds delay. Of course, the use of a time average diagram which depends not only on the current value causes this display.

Figure 6-13 shows the traffic from Router 2 to Router 4 in the FTP application. In general, slight increase in throughput can be noticed when the load is less, and more throughput is expected when the delay is less. In FTP, the difference between the 0.1 seconds delay and the 5 seconds delay cannot be seen in the figure. This is because the number of FTP messages are infrequent compared to HTTP messages where the inter-request time configured is 360 seconds<sup>7</sup>. Another thing to note is that the FTP application for 5 seconds delay does not have time to completely establish the FTP

---

<sup>7</sup> Look at Figure B-3 to see that the two throughput eventually meet together at 6000 bits per second. However when increasing the inter-request to 60 there will be a difference between the 0.1 seconds delay and the 5 seconds delay as shown in Figure B-5.



connection. So after the solution is applied, the FTP connection needs to be established. This results in additional traffic, FTP connection setup, added to the throughput. This analysis can be seen more clearly in Figure 6-21 where the FTP traffic received to LAN\_East node is shown. Comparing Figure 6-13 to Figure 6-7, it can be noticed that there is a difference initially between the 0.1 and the 5 seconds delay of Internet. This is because of the slow initial convergence of the 5 seconds delay scenario. In addition, there is an initial setup of FTP that is needed. In case the delay of the Internet is 0.1 seconds, the initial setup completed before the start of blackholing. However, it is not completed in the case where the delay of Internet is 5 seconds. Refer to Figure 6-20 and Figure 6-21 to see this.

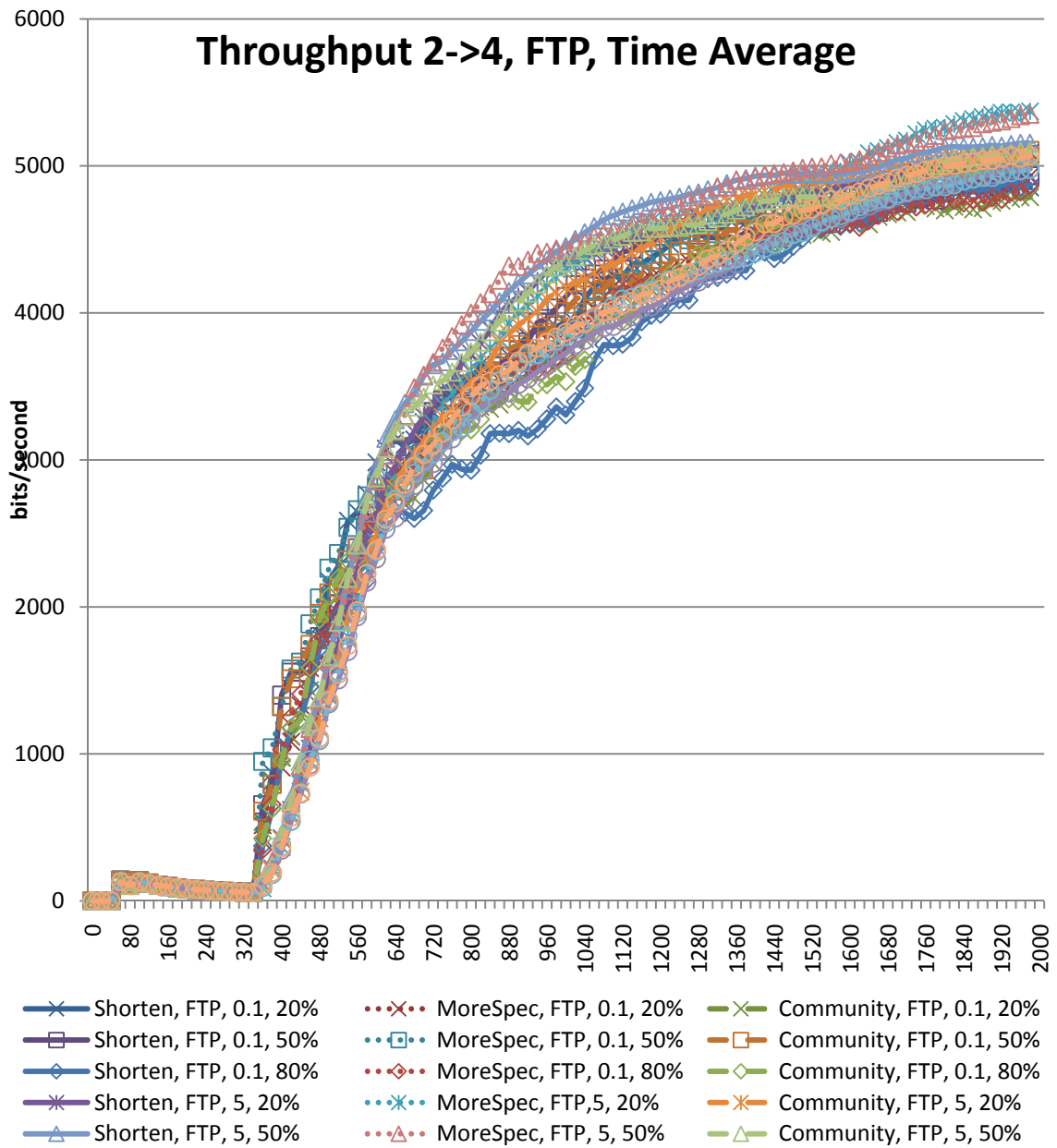


Figure 6-13 throughput from Router 2 to Router 4 for FTP application.

Figure 6-14 shows the throughput from Router 2 to Router 4 for the VOIP application. All different application configurations coincide with each other. There is no difference between the 0.1 seconds delay and 5 seconds delay because VOIP is a real time application that runs over UDP. And UDP implements no flow control or congestion control. This results in a continuous sending of traffic from Router 2.

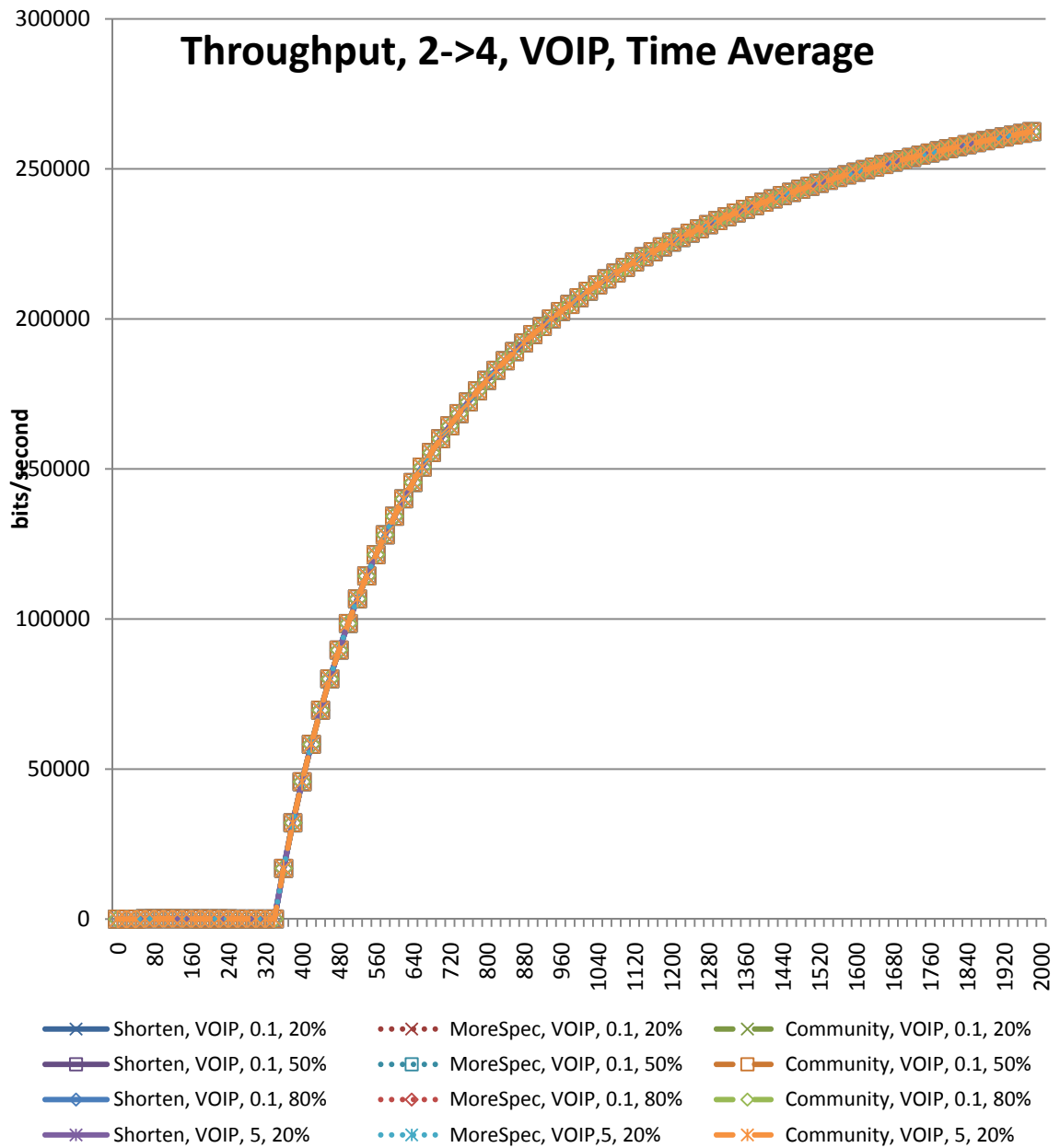


Figure 6-14 Throughput from Router 2 to Router 4 in VOIP application

Figure 6-15 shows the incoming throughput to Router 2 from Router 4 for the HTTP application.

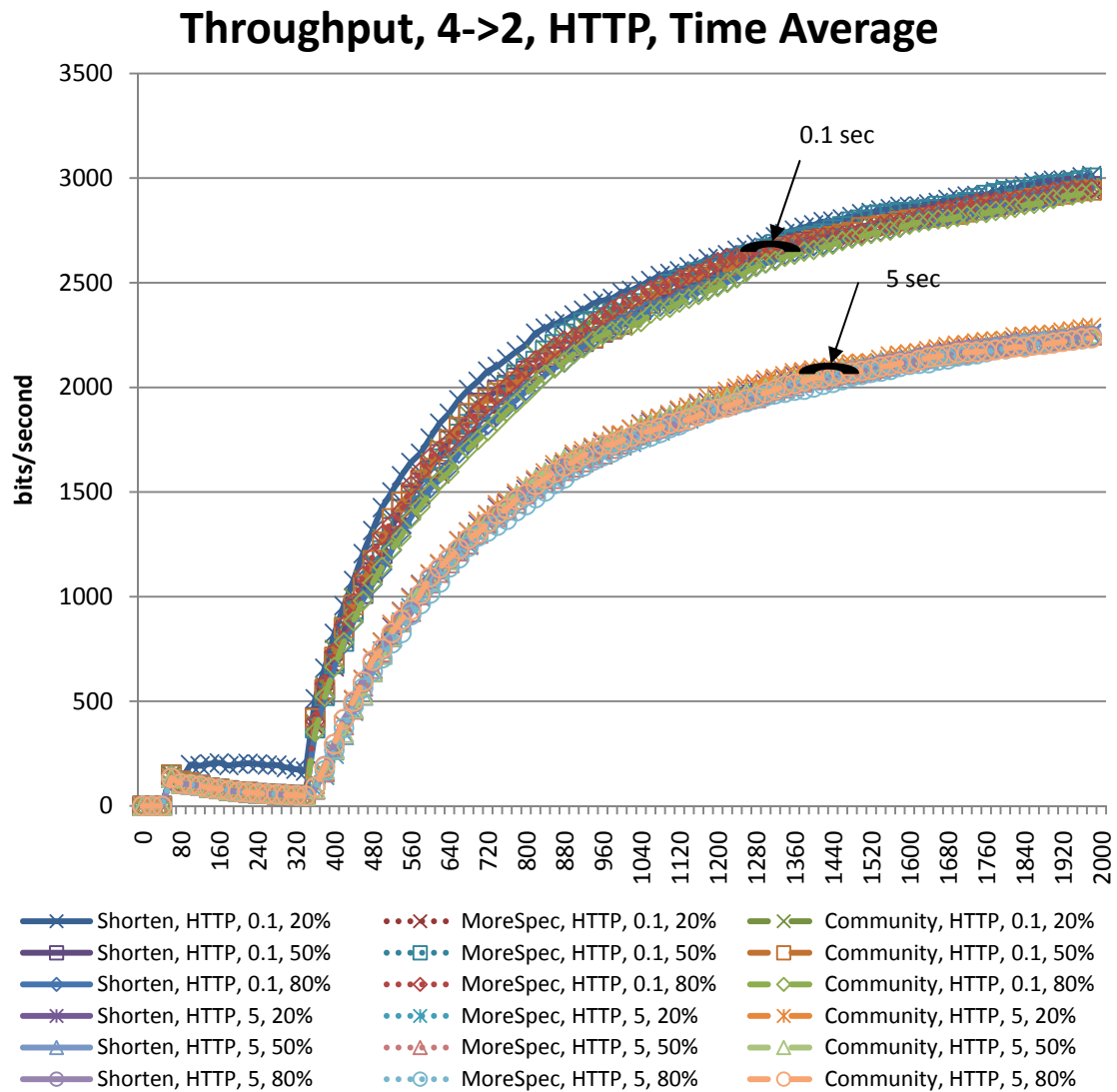


Figure 6-15 Incoming traffic from Router 4 to Router 2 for HTTP application.

Figure 6-15 shows a slight increase in the throughput for lower loads. The ratio of the throughput in 0.1 seconds delay of the Internet with respect to 5 seconds is about 6:5 at time 2000 seconds.

Figure 6-16 shows the throughput from Router 4 to Router 3 in the FTP application for different solutions, load, and delay. Higher load results in lower throughput in the FTP.

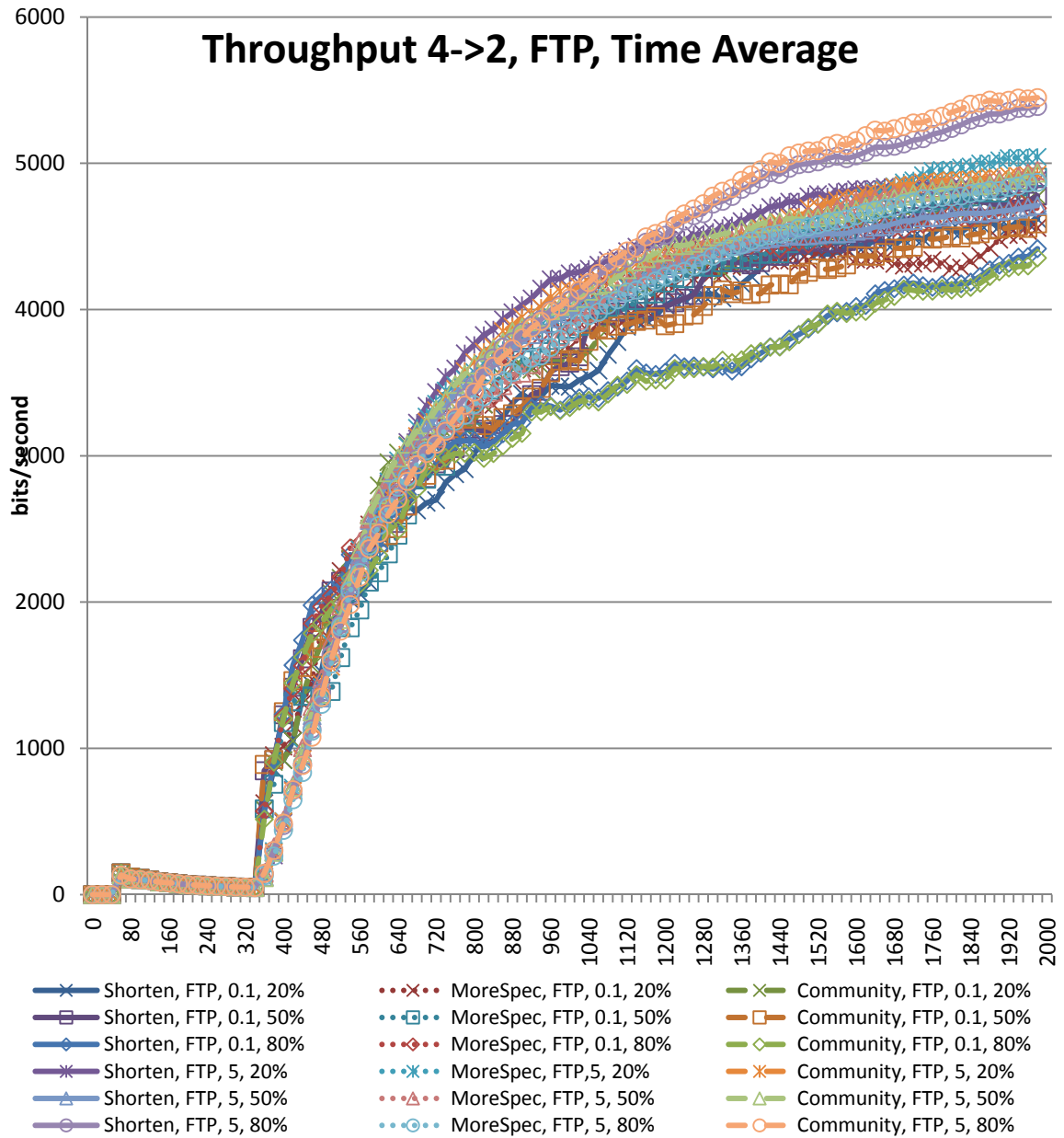


Figure 6-16 Throughput from Router 4 to Router 2 in FTP application.

The difference due to load and to Internet delay does not show in the FTP application, and this is similar to Figure 6-13, where the TCP congestion control is not triggered because of the high inter-request time.<sup>8</sup>

Observing Figure 6-16 and Figure 6-13, it is clear that the FTP throughputs in both directions are similar to each other. This is because the client either downloads or uploads files, and by referring to Figure A-6 in Appendix A, it is clear that the 'Get' command (download) is 50%. So, the 'Set' command is the other 50%, and they should have similar traffic throughput. This can be contrasted with the HTTP throughput in Figure 6-16 and Figure 6-12 where the size of the file sent from the server to the client is larger; and that is why the outgoing traffic from Router 2 to Router 4 is more than the incoming traffic from Router 4 to Router 2.

Figure 6-17 shows the throughput from Router 4 to Router 2 in the VOIP application. Increasing the load or the delay of the Internet slightly reduces the throughput.

---

<sup>8</sup> Compare Figure B-2 to Figure B-4 to see that the traffic become distinct when the interrequest time is reduced to 60 seconds similar to HTTP interarrival time.



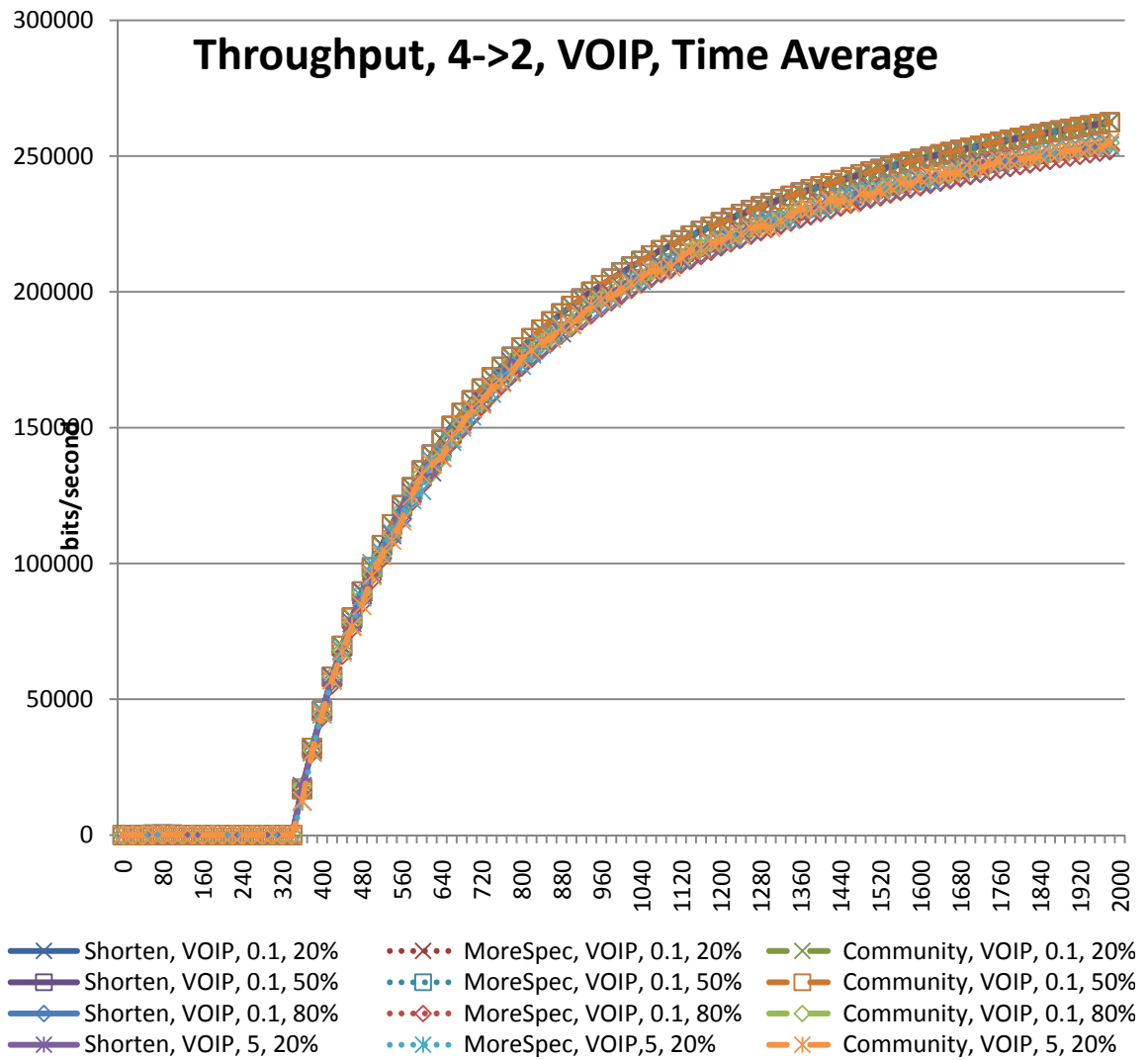


Figure 6-17 Throughput from Router 4 to Router 2 in VOIP application.

### **6.2.3.1    Application Level Throughput**

In this section, for different applications, figures are shown that display traffic sent from the LAN\_East and received by LAN\_East are shown.

Figure 6-18 shows the HTTP traffic sent from the LAN\_East subnet. This figure clearly shows that the 'packets per second' during the period of blackholing (300-360) reduces to zero.

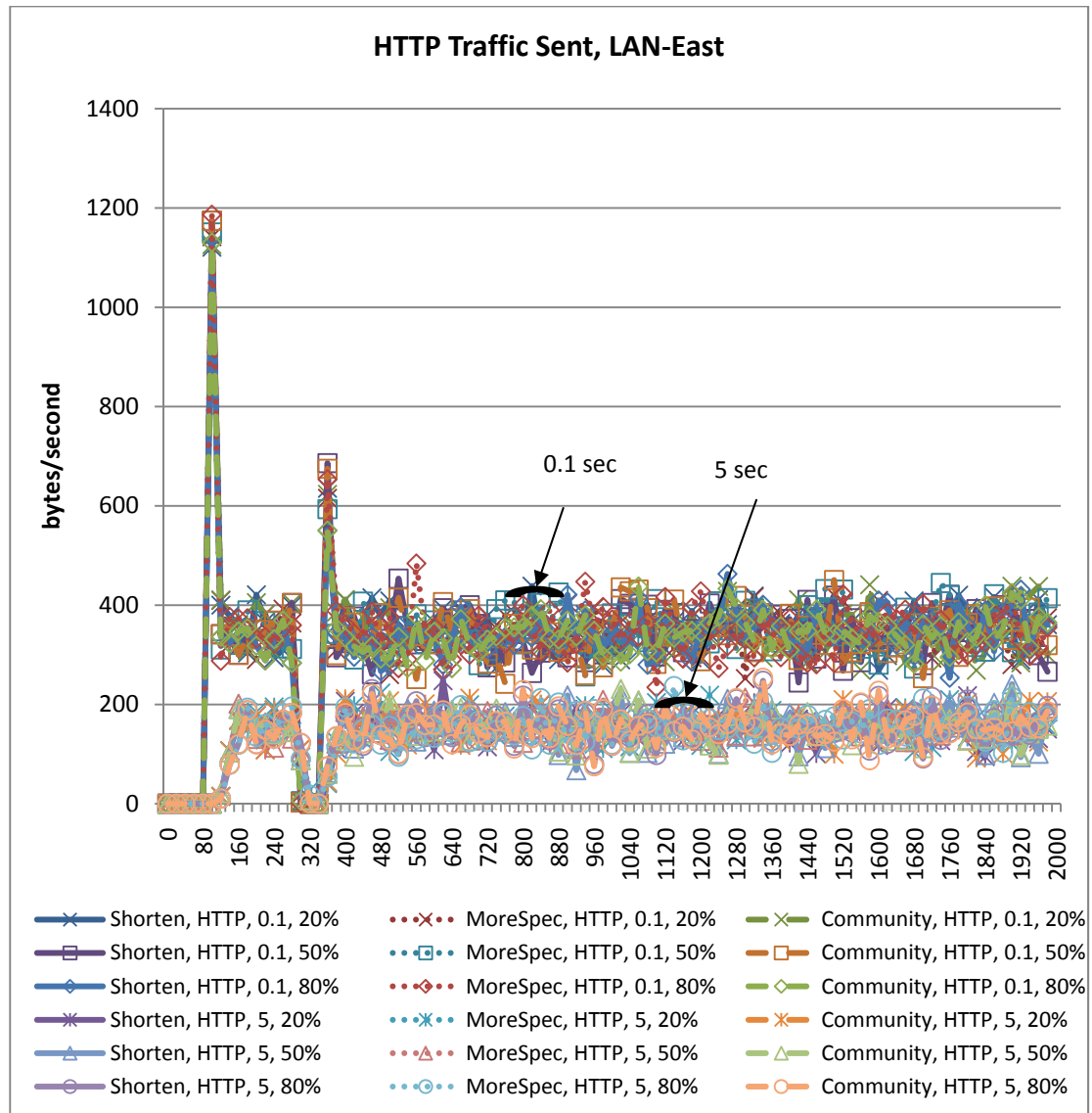


Figure 6-18 HTTP Packet Sent

Figure 6-18 also shows lesser number of packets sent when the delay of the Internet is more. The packets sent are double when the delay is 0.1 in comparison with 5 second delay. Figure 6-19 shows the HTTP packets received by LAN\_East.

### HTTP Traffic Received, LAN\_East

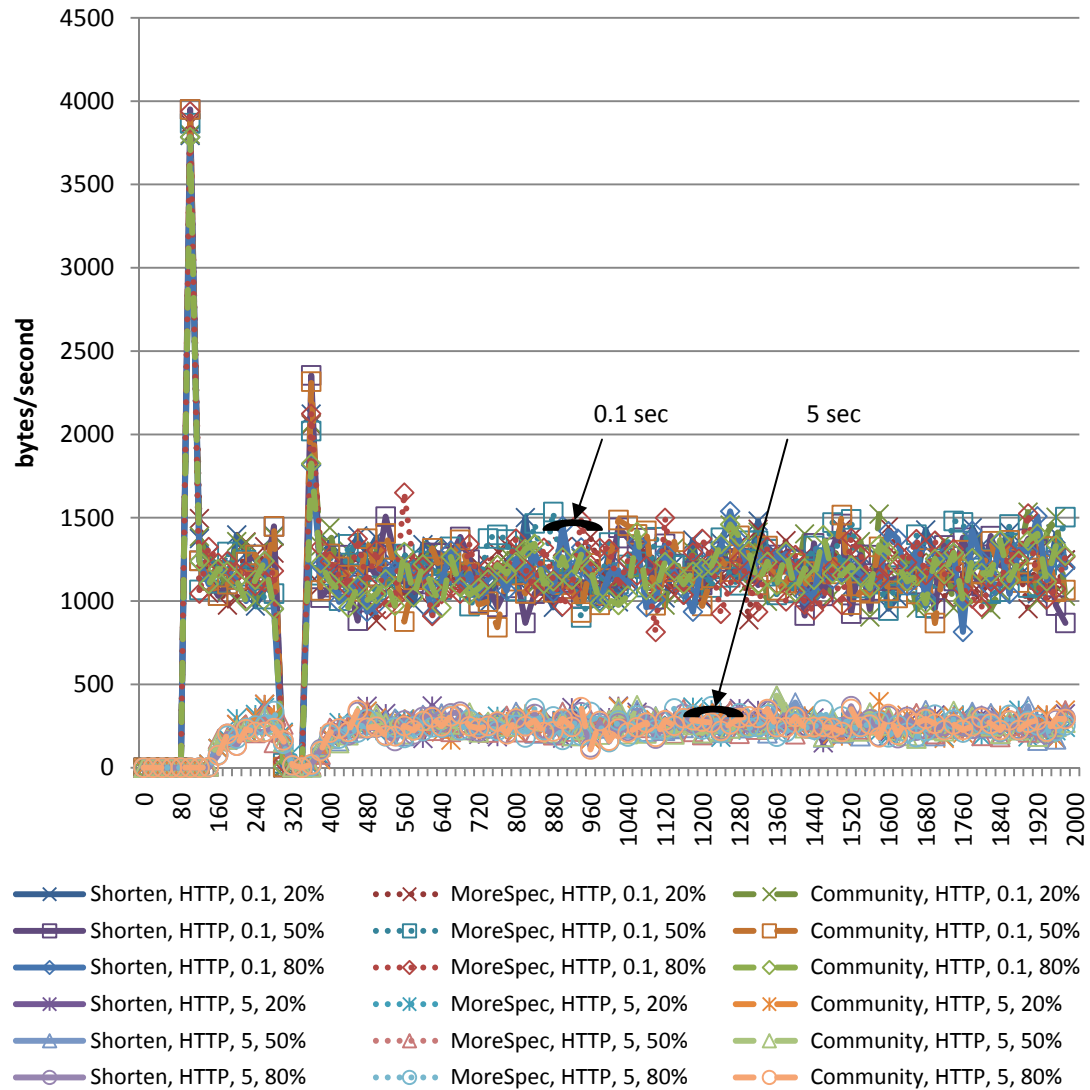


Figure 6-19 HTTP Packet Received

The packets received reduce to 0 in all experiments during the period of blackholing. Faster recovery can be observed when the Internet delay is less.

Figure 6-20 shows the FTP packets sent from the LAN\_East subnet. The figure clearly shows that the number of packets sent during the time of blackholing reduces to zero.

### FTP Traffic Sent, LAN-East

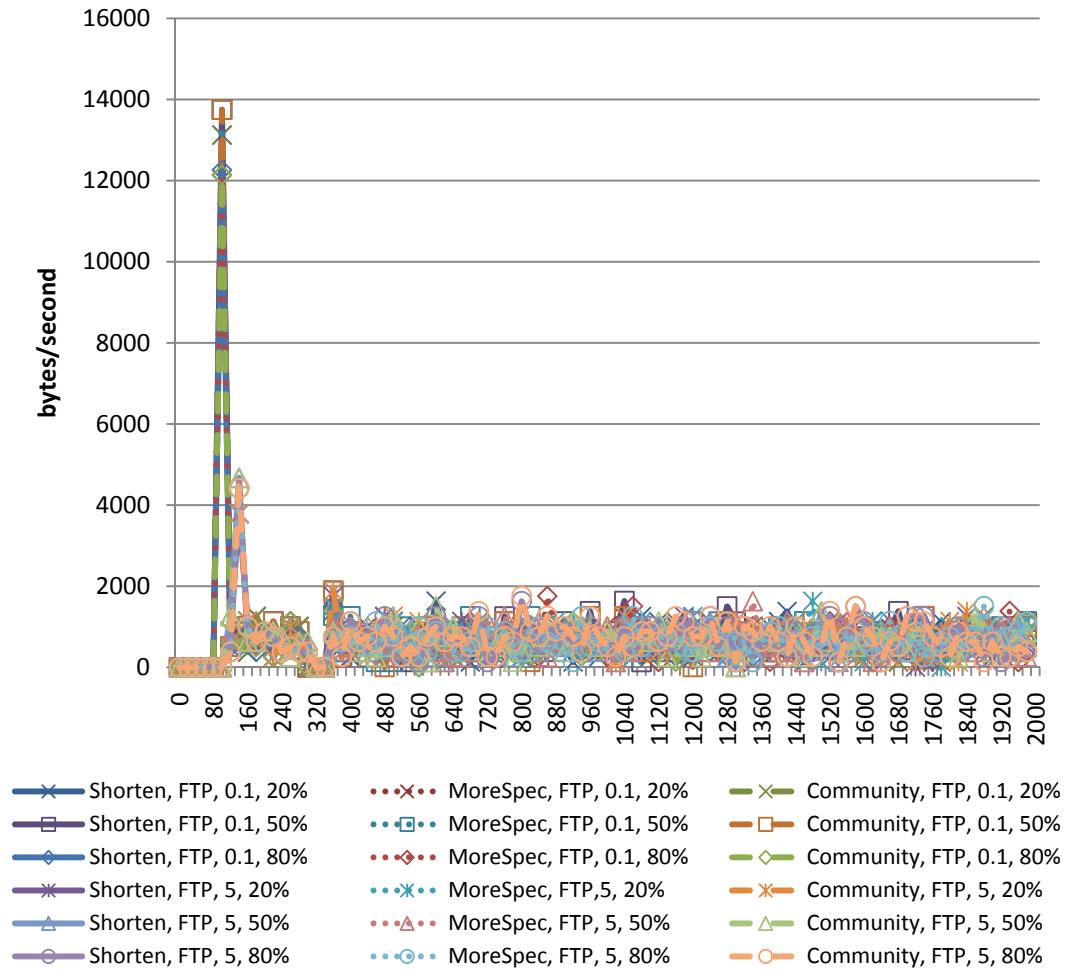


Figure 6-20 FTP Packets sent from LAN\_East.

Figure 6-21 shows the FTP packets per second received by LAN\_East. The figure clearly shows that number of packets per seconds that are received during the period of blackholing is zero. The figure also shows that for the scenarios that include 5 seconds

delay of the Internet, the packets are not received until well after the recovery of blackholing. That is because of the nature of FTP having low number of packets in addition to the time needed for BGP to converge initially. By the time there is a reply, the blackholing starts, and when the solution is applied and network converges, it starts the initialization of FTP. This initialization or setup causes the initial increase of throughput after the solution is applied.



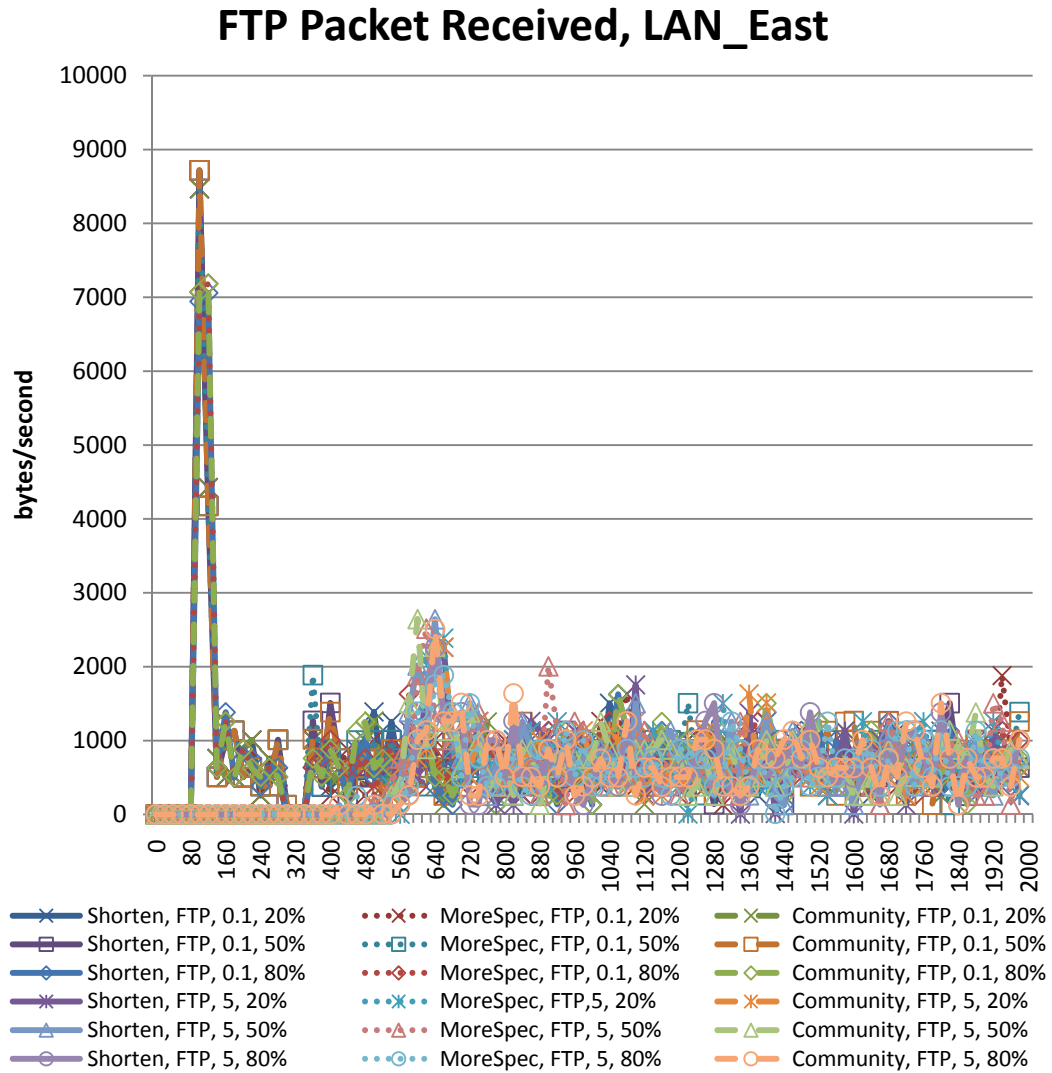


Figure 6-21 FTP Packet Received to LAN East.

Figure 6-22 shows the VOIP packet sent from LAN\_East. It is shown that the VOIP application continues sending messages even in the time of blackholing because it runs

over user datagram protocol (UDP). The traffic is smooth in Figure 6-22 because this is the traffic sent from LAN\_East and the statistic is calculated with respect to LAN\_East. The traffic did not start experiencing the delay of the Internet node or any load.

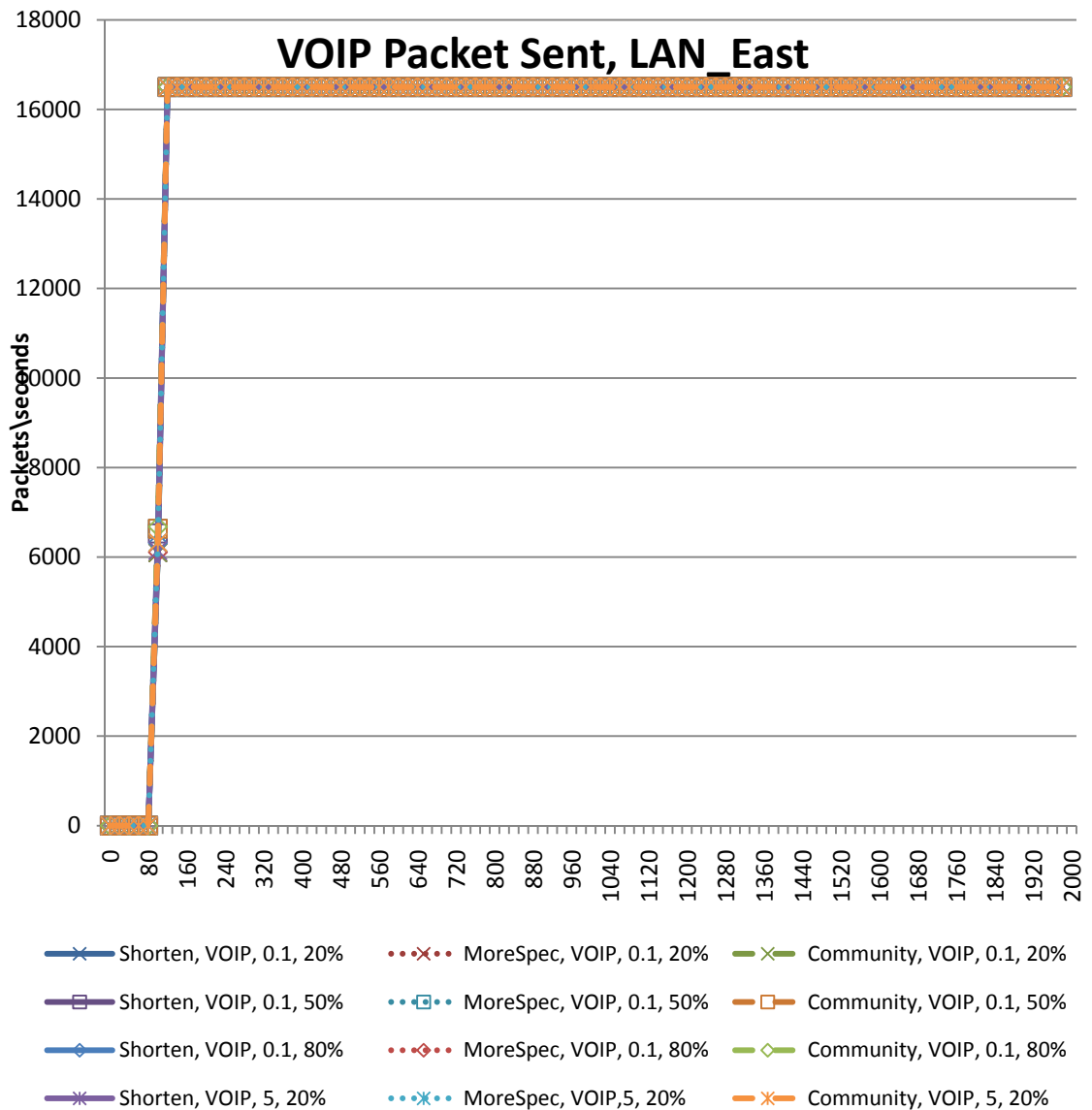


Figure 6-22 VOIP Packet Sent from LAN\_East

Figure 6-23 shows the VOIP packet received by LAN\_East. The figure clearly shows that the VOIP traffic goes to zero in the time of blackholing. Having exponential delay

with 5 seconds as a mean for the VOIP application results in a fluctuating behavior of the traffic received. This is because every packet before it is received, it experiences a different delay because of the higher mean and variance values of the 5 seconds delay.

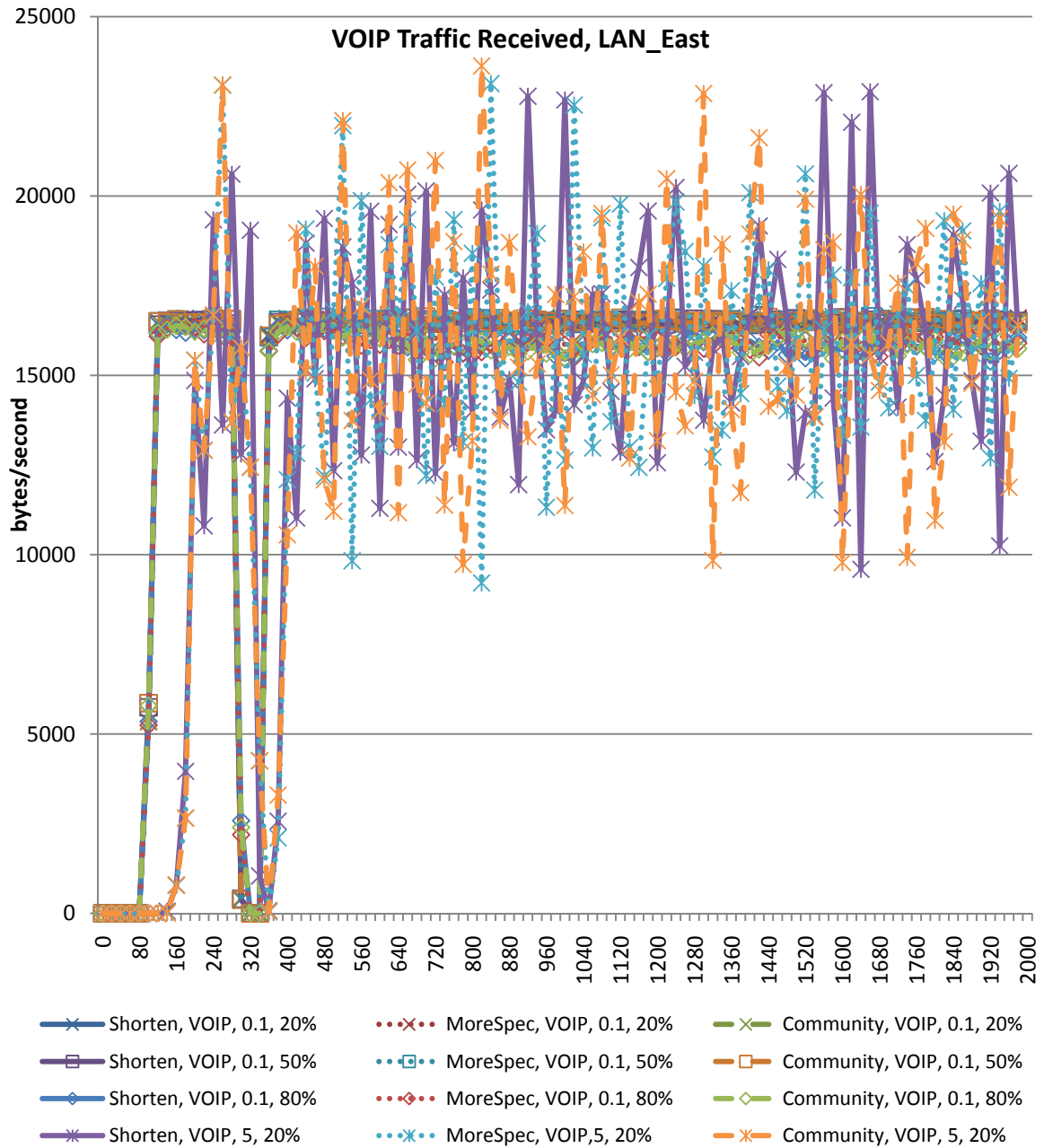


Figure 6-23 VOIP Packet Received in LAN\_East

Figure 6-24 shows the page response time for the HTTP application for different solutions, load, and delay of the Internet.

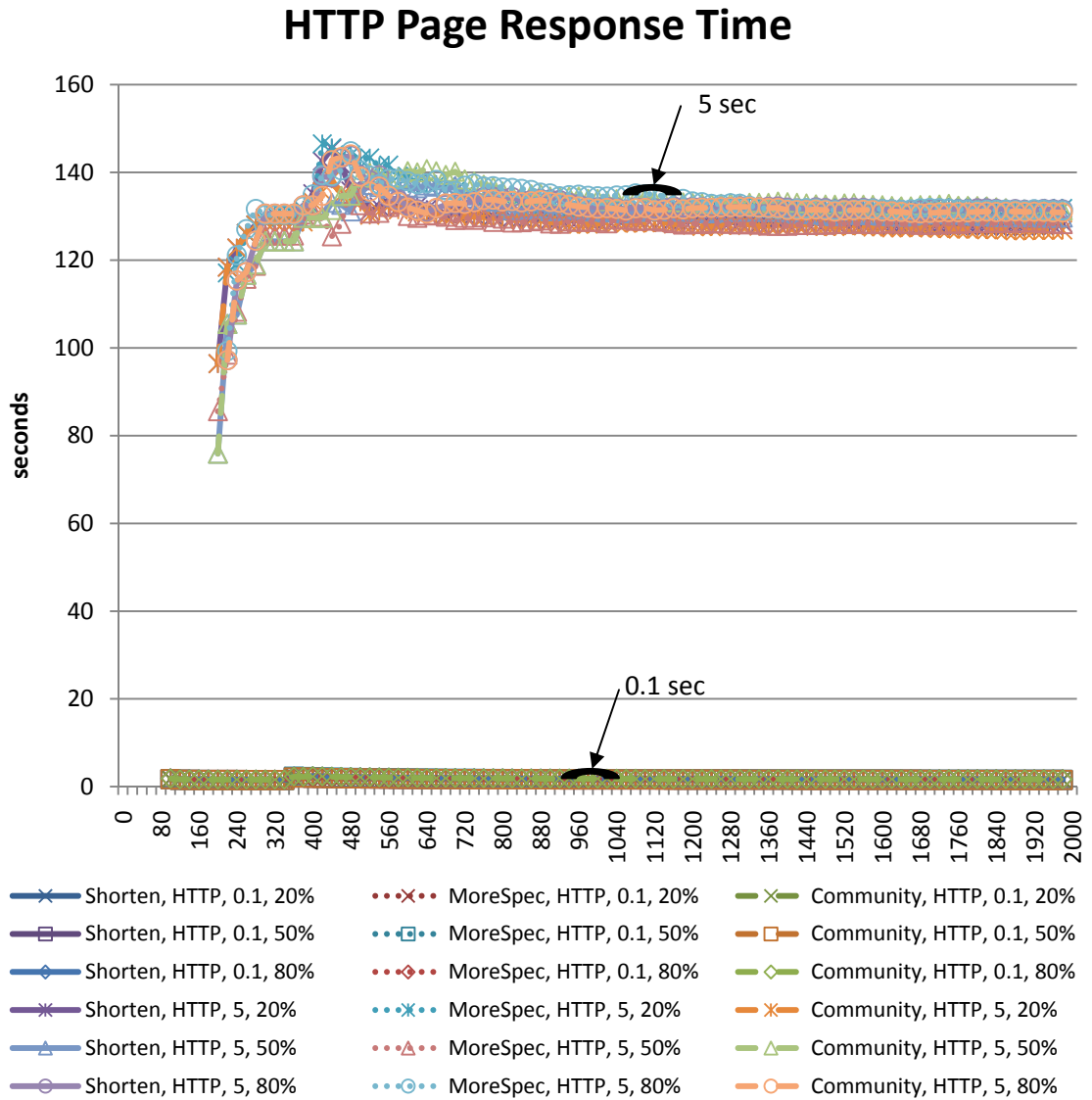


Figure 6-24 Page Response time for HTTP client

Figure 6-24 shows that there is an increase in response time at the time the solution is applied. The page response time increases significantly when having a higher Internet

delay. Similar discussion can be conveyed on the FTP download response time shown in Figure 6-25.

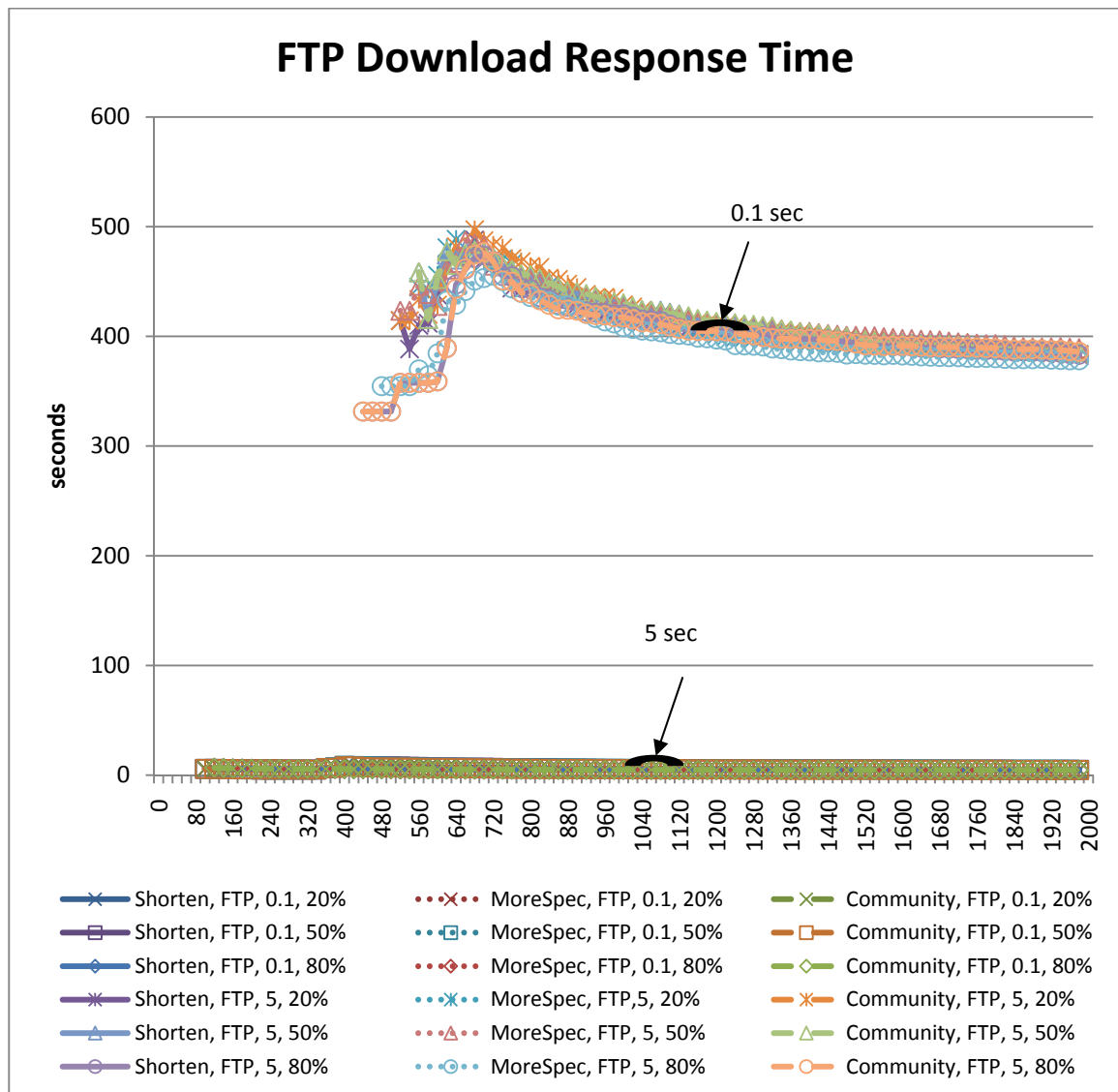


Figure 6-25 FTP download response time.



### **6.3 CASES WHEN THE SIMULATION FAILS**

In cases of having exponential delay with 5 seconds as a mean, and for the VOIP application, and for either 50% load or 80% load on the link, the traffic, in the simulation, switches back to going through the malicious router. This occurs in most of the cases when the link load is 50% and in all the cases when the link load is 80%. Sometimes, it even switches to the malicious router then switches back to the non-malicious one and then returns to the malicious router. This case and others are shown in Figure 6-26.

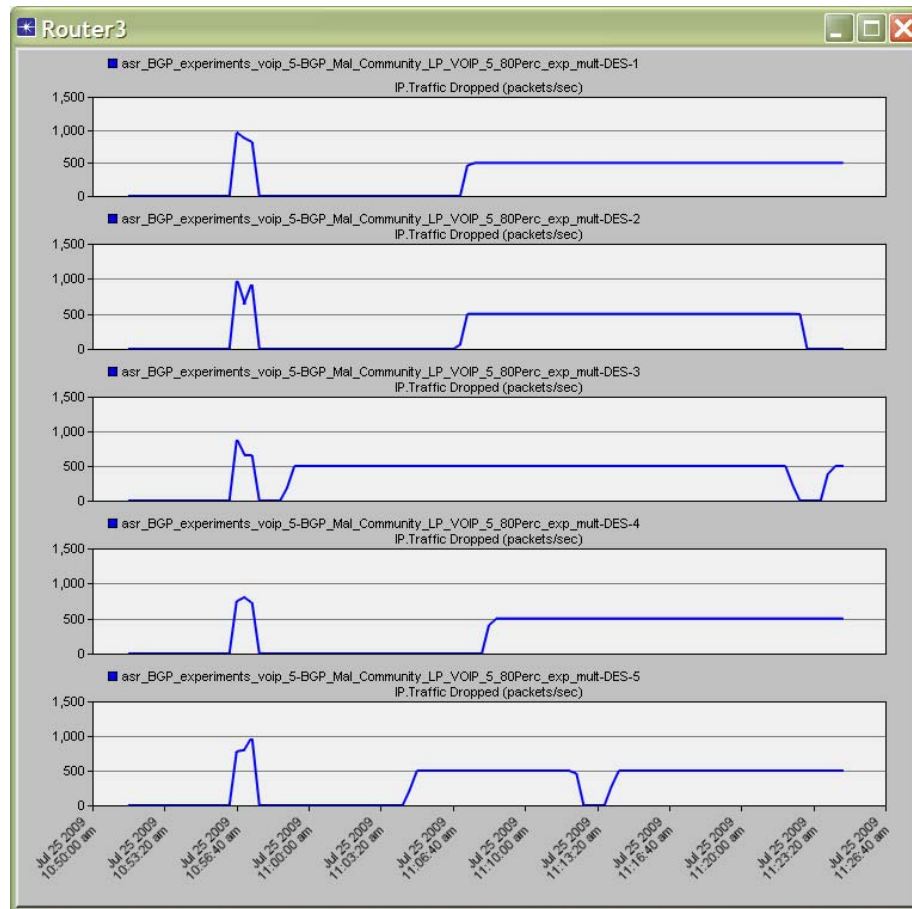


Figure 6-26 Packet drop in VOIP, Community Solution, exponential with 5 second delay, 80% link load

The main reason why this occurs is because of the use of the exponential delay and 5 seconds as mean for the delay of the Internet. In simulation, the Internet is modeled as a router with this specification resulting in delayed and out of order BGP messages. The VOIP application and high link load work as a catalyst to switch the traffic back to the

blackholing router. This is because the VOIP application has high traffic demand; and high link load causes some delay to the packets.

By working in debugging mode and monitoring the Internet node, it can be observed that from the time the BGP update messages are sent, the Internet starts creating multiple new BGP connections with neighbors. Such a behavior does not stop until one of them changes its state to ESTABLISHED. As a result, the real connection with one of the routers changes its state to IDLE resulting in a withdraw message distributed among nodes in the network. Router 2 drops the routes to the destination through Router 4 and switches its traffic to Router 3 (the malicious router). So, this cannot be considered as a disadvantage of the solution, but rather a simulation related problem.

## **6.4 RECOMMENDATIONS**

BGP tuning techniques are a good way of controlling the traffic. Real time applications will suffer most from the packet drop period so this shall be taken into account by the user or designer of these applications. With the increase of Internet delay, the throughput is less and the packet drop is more, but no other solution from the solutions suggested can do better because the results are network specific and improvement might have to consider other protocols.

## **6.5 SUMMARY**

In this chapter, we evaluated the performance of solving the methods using the BGP tuning techniques. The solution type, delay of Internet, type of application, and the load in terms of putting more load on specific links are the factors studied in our simulation. The percentage of packet drop, convergence time, throughput, application packet sent and received, page response time, and download response time are the metrics used in the evaluation.

---

## ***CHAPTER 7***

### ***CONCLUSION & FUTURE WORK***

#### ***7.1 CONCLUSION***

The goal of this thesis work is to propose, implement, and evaluate mechanisms to solve the problem of Internet blocking. This Internet blocking is done by a malicious IISP that isolates a local region by blackholing its traffic and announcing capabilities to deliver the traffic to its intended destination. Only BGP based solutions are considered.

In order for the application to work, the traffic in both directions should not pass through the malicious node. Therefore, both the incoming and outgoing traffic for a single application must be controlled to pass through the good IISP. Controlling the incoming traffic is more complex than controlling the outgoing traffic, because controlling the outgoing traffic is a local decision while controlling the incoming traffic involves influencing the traffic in the Internet to come through the good IISP.

The thesis proposes three techniques to solve the problem, namely BGP tuning, virtual peering and virtual transit. It provides a qualitative comparison between them also. The thesis then describes the design, implementation, and validation of BGP tuning techniques. A performance evaluation is provided of BGP tuning techniques. The

performance is conducted in terms of type of solution, Internet delay, application type, and load. BGP convergence time, throughput, packet drop, and response time are the metrics used for comparison.

## **7.2 FUTURE WORK**

The implementation of virtual peering and virtual transit is going to be the next step of this research. Then these methods can be compared with BGP tuning methods. Virtual peering and virtual transit might take longer time to converge because of the setup needed. These techniques, however, provide more deterministic control of traffic in addition to the identity hiding employed by tunneling.

This work can be extended by applying these techniques in real life scenarios to test their effect on the Internet. Real case scenarios can give better results in terms of traffic blocked, applications that succeeded to work, and other related issues. One can also relax the assumption of having only one provider being malicious. Multiple ISPs in the path might be malicious and the problem then could be how to avoid them. Also, one can look at what if the provider analyzes traffic inside the tunnel and drops the packets. Some encryption might be needed. Detection of malicious activities in all of these scenarios is also important for future research.

---

## ***Appendix A***

### ***APPLICATION CONFIGURATION***

In order to make it easy to interpret some results, we show in this section the configurations needed for each protocol of interest. The configurations of TCP, HTTP, FTP and VOIP are shown. The configuration of each application is shown. And also the default parameters of OPNET are used in our experiments.

#### ***A.1 TCP CONFIGURATION***

Figure A-1 shows the TCP configuration used.

?	TCP Parameters	(...)
?	Version/Flavor	Unspecified
?	Maximum Segment Size (bytes)	Auto-Assigned
?	Receive Buffer (bytes)	8760
?	Receive Buffer Adjustment	None
?	Receive Buffer Usage Threshold (o...	0.0
?	Delayed ACK Mechanism	Segment/Clock Based
?	Maximum ACK Delay (sec)	0.200
?	Maximum ACK Segments	2
?	Slow-Start Initial Count (MSS)	2
?	Fast Retransmit	Enabled
?	Duplicate ACK Threshold	3
?	Fast Recovery	Reno
?	Window Scaling	Disabled
?	Selective ACK (SACK)	Disabled
?	ECN Capability	Disabled
?	Segment Send Threshold	MSS Boundary
?	Active Connection Threshold	Unlimited
?	Nagle Algorithm	Disabled
?	Kam's Algorithm	Enabled
?	Timestamp	Disabled
?	Initial Sequence Number	Auto Compute
?	Retransmission Thresholds	Attempts Based
?	Initial RTO (sec)	3.0
?	Minimum RTO (sec)	1.0
?	Maximum RTO (sec)	64
?	RTT Gain	0.125
?	Deviation Gain	0.25
?	RTT Deviation Coefficient	4.0
?	Timer Granularity (sec)	0.5
?	Persistence Timeout (sec)	1.0
?	Connection Information	Do Not Print
?	Acceleration	Disabled

Figure A-1 TCP Configuration.



## A.2 HTTP CONFIGURATION

Figure A-2 shows the configuration of HTTP application. The page Interarrival time is exponentially distributed with mean 60 seconds. Moreover the HTTP version used is HTTP 1.1.

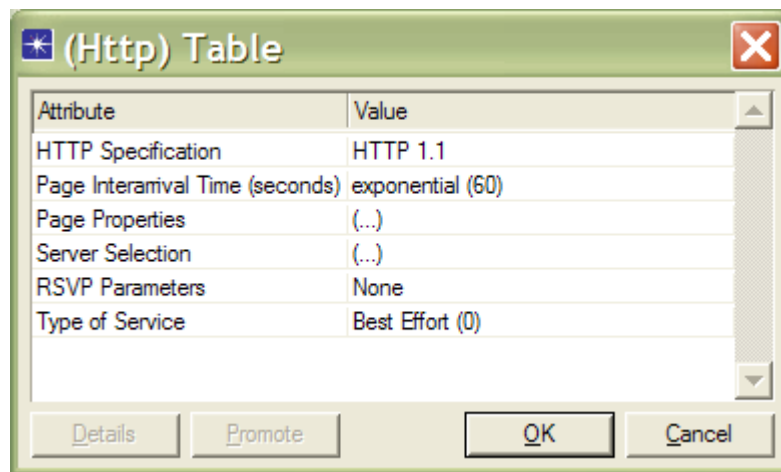


Figure A-2 HTTP Configuration.

Figure A-3 shows the properties of an HTTP page. Each page has five medium sized images in addition to 1000 bytes page size.

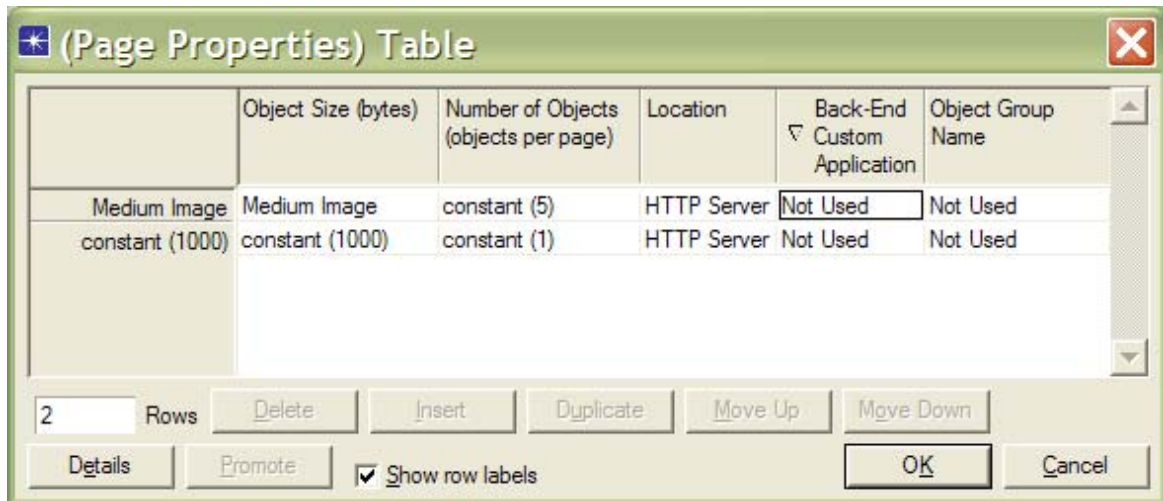


Figure A-3 HTTP Page Properties.

Figure A-4 shows the size of the image properties. Medium image has a uniform distribution between 500 and 2000 bytes. So, its mean is 1750 bytes. The average page size is  $1000 + 5 * 1750 = 9750$  bytes.

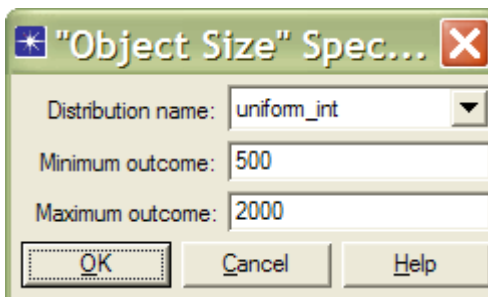


Figure A-4 Size of Image.

The server selection is shown in Figure A-5. The number of pages per server is exponentially distributed with mean 10 pages.

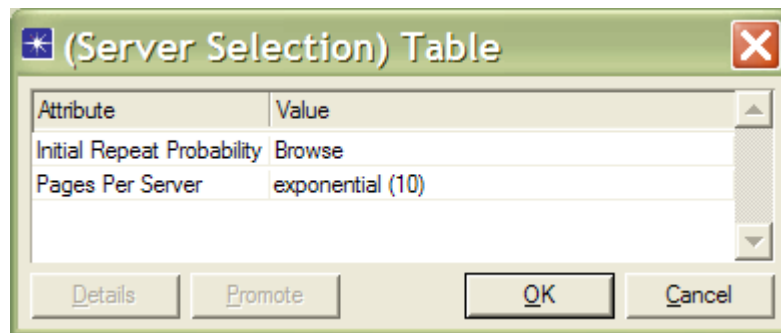


Figure A-5 HTTP Server Selection.

### ***A.3 FTP CONFIGURATION***

Figure A-6 shows the FTP configuration used. The interarrival time is exponentially distributed with mean 360 seconds. The Get command is 50% of total commands. That means that 50% for the 'Set' command. The file size is 50000 bytes.

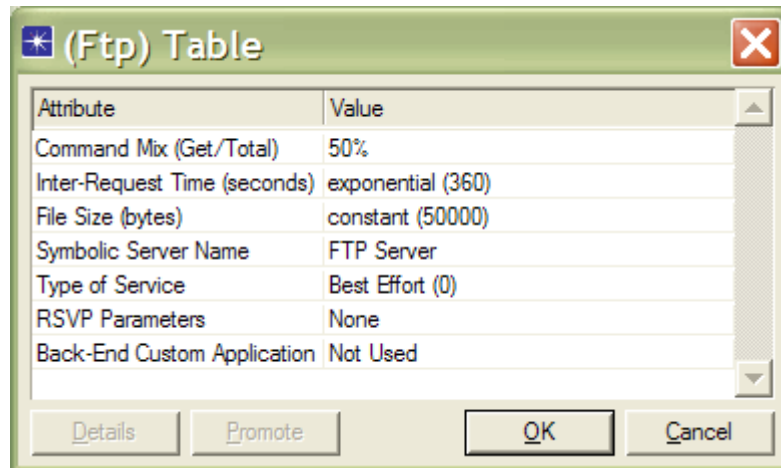


Figure A-6 FTP Configuration.

## A.4 VOIP CONFIGURATION

Figure A-7 shows the VOIP configurations used. The encoding scheme is GSM FR and there is one voice frame per packet. The compression and decompression delays are both 0.02 seconds.

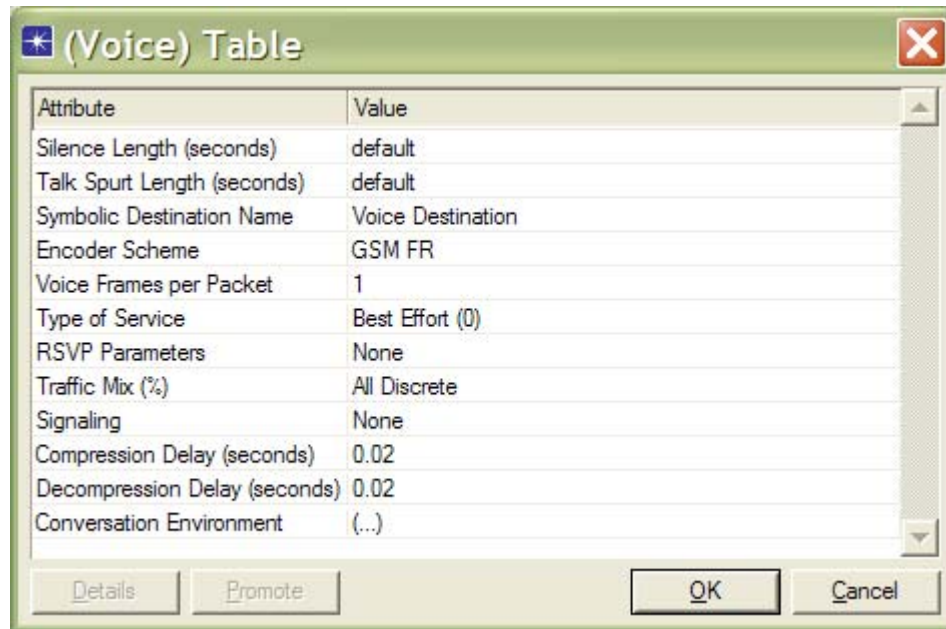


Figure A-7 VOIP Configuration.

The silence length configuration is shown in Figure A-8. Both incoming and outgoing are exponentially distributed with mean 0.65 seconds.

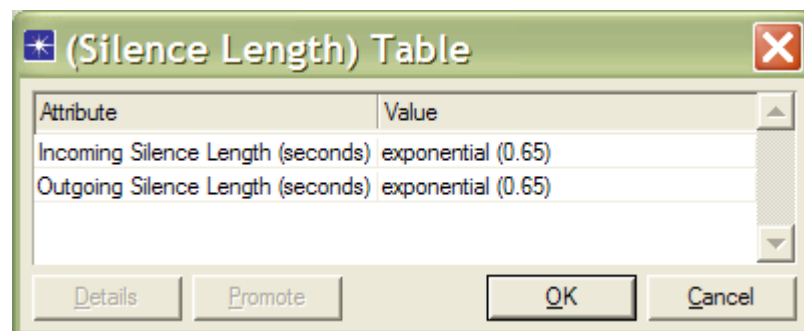
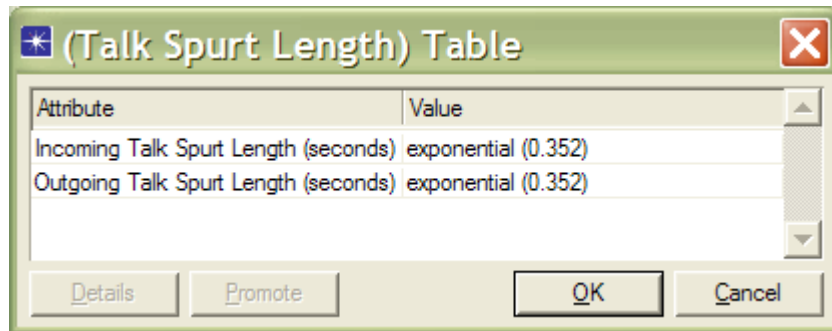


Figure A-8 Silence Length configuration.

The talk spurt length is shown in Figure A-9. Both incoming and outgoing talk spurt length is exponentially distributed with a mean of 0.352 seconds.



Attribute	Value
Incoming Talk Spurt Length (seconds)	exponential (0.352)
Outgoing Talk Spurt Length (seconds)	exponential (0.352)

Figure A-9 Talk Spurt Length.

---

## ***Appendix B***

### ***BASELINE THROUGHPUT AND APPLICATION TRAFFIC***

In this appendix a set of baseline figures are shown. These figures are for the runs of throughput and application traffic presented in CHAPTER 6. Basically, these figures are simple run of different application for different loads (20%, 50%, and 80%) and for different Internet delay (0.1 second and 5 seconds) without the intervention of malicious blackholing and application of solution. The presentation of these figures will help the thesis reader see the behavior of the traffic in case the traffic goes normally without any change. And as a result, it helps in strengthening the understanding of why specific traffic behaves in certain way.

#### ***B.1 BASELINE THROUGHPUT***

In throughput the traffic pass normally between Router 2 and Router 3. So in this section, only throughputs between Router 2 and Router 3 in both directions are shown.

Figure B-1 shows the throughput from Router 2 to Router 3 for HTTP application.

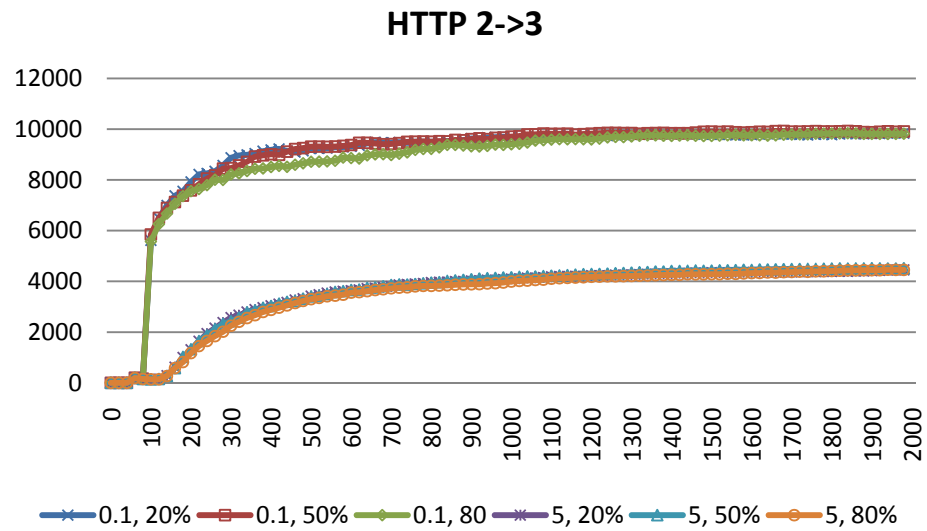


Figure B-1 Baseline HTTP throughput from Router 2 to Router 3.

Figure B-2 shows the throughput from Router 3 to Router 2 for HTTP application.



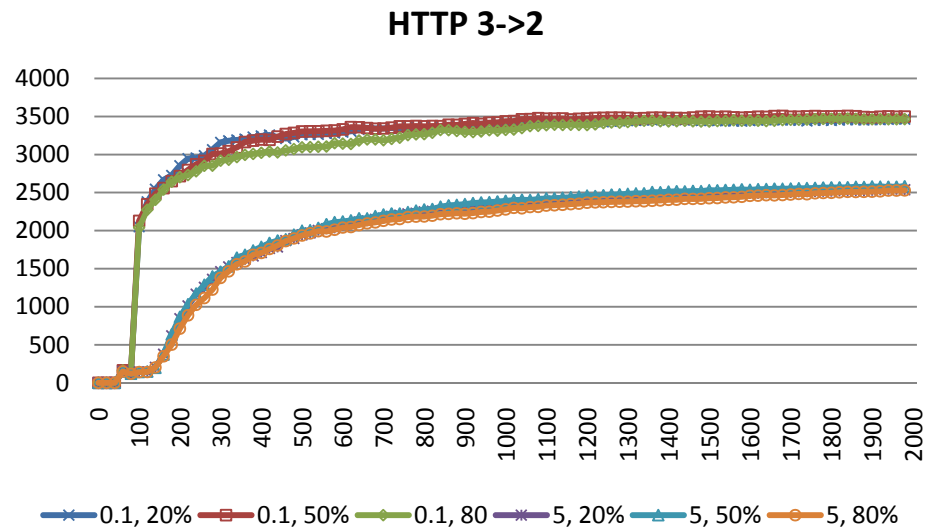


Figure B-2 Baseline HTTP Throughput from Router 3 to Router 2.

Figure B-3 shows the baseline throughput for FTP from Router 2 to Router 3.

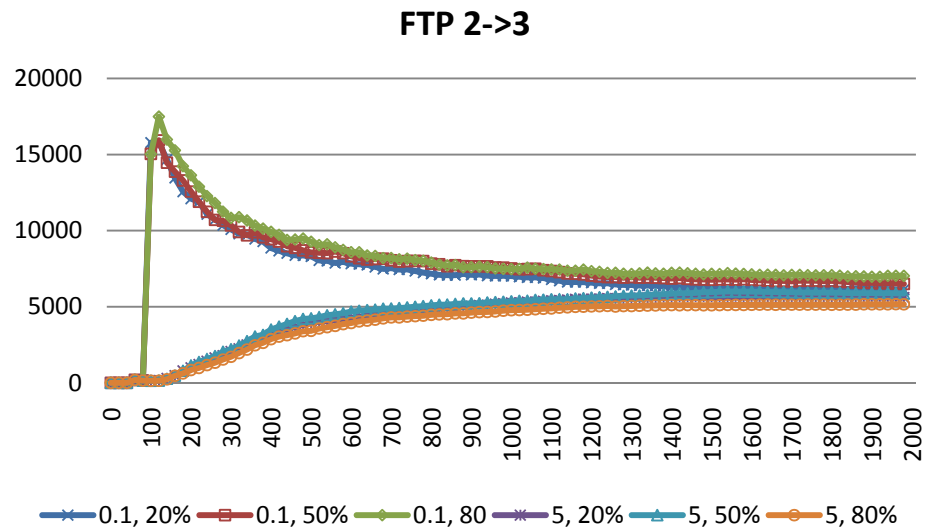


Figure B-3 Baseline FTP throughput from Router 2 to Router 3.

Figure B-4 shows the Baseline throughput for FTP from Router 3 to Router 2.

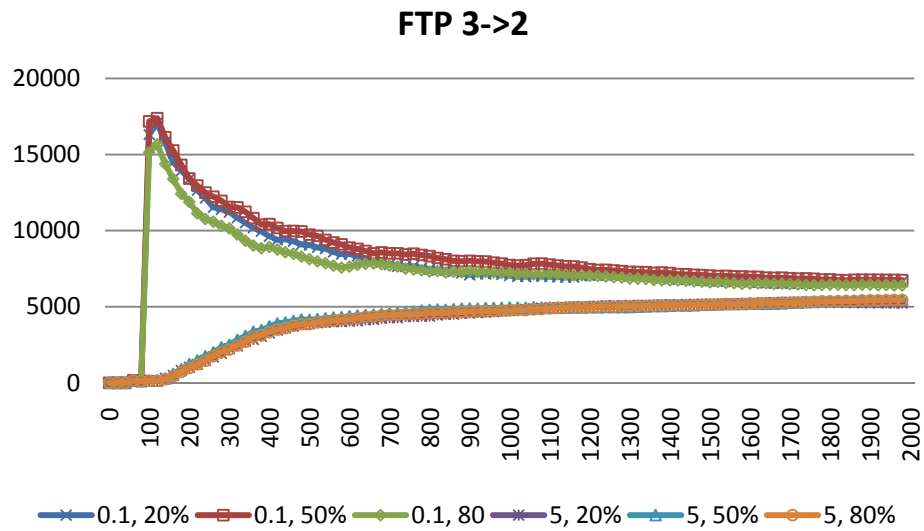


Figure B-4 Baseline FTP throughput from Router 3 to Router 2.

Figure B-5 shows the FTP throughput from Router 2 to Router 3 when the Inter-request time is 60 seconds. The figure shows the case when just the load is 20%.

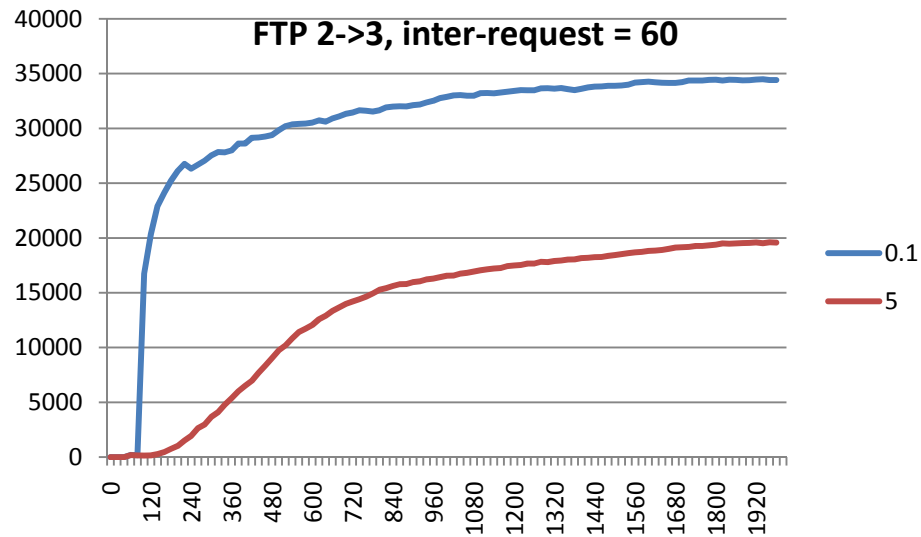


Figure B-5 baseline throughput for FTP application, inter-request is 60.

Figure B-6 shows the baseline FTP throughput from Router 3 to Router 2 when the inter-request time is 60 seconds. The figure shows the traffic when the load is just 20%.

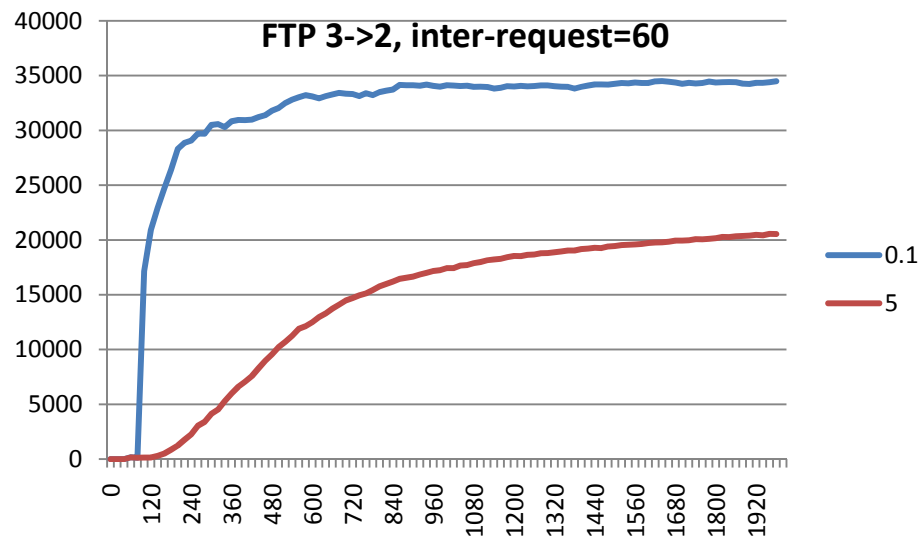


Figure B-6 Baseline FTP throughput from Router 3 to Router 2 when the inter-request is 60

Figure B-7 shows the baseline throughput for VOIP application from Router 2 to Router 3.

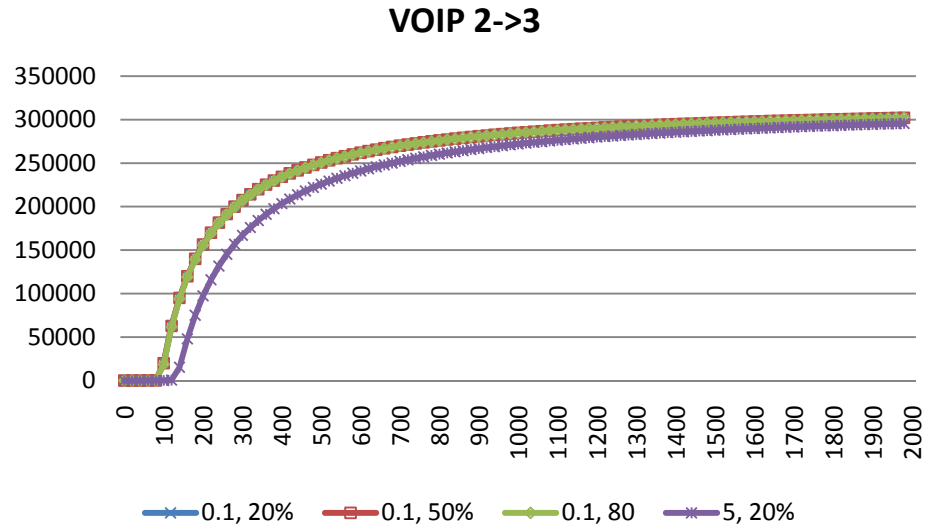


Figure B-7 Baseline VOIP throughput from Router 2 to Router 3.

Figure B-8 shows the baseline throughput from Router 3 to Router 2 for VOIP application.

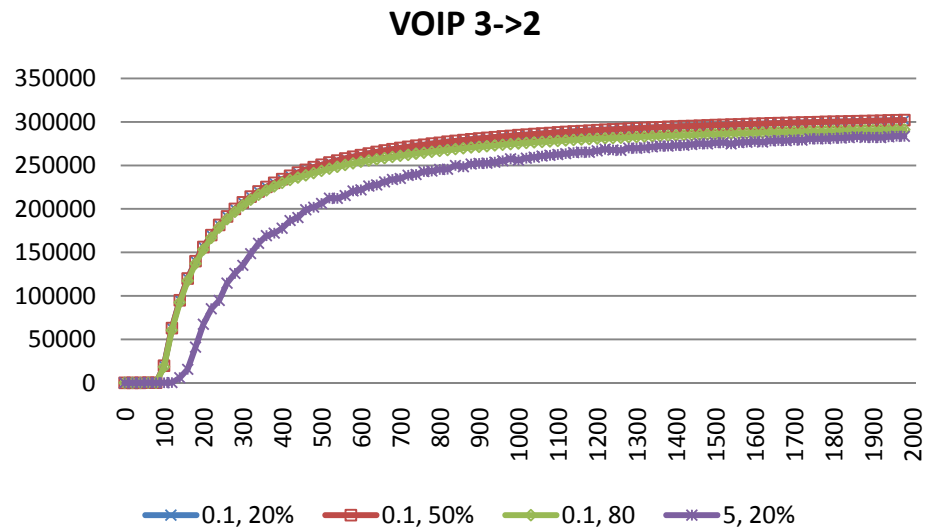


Figure B-8 Baseline VOIP throughput from Router 3 to Router 2.

## ***B.2 BASELINE APPLICATION TRAFFIC***

In this section the application level traffic sent from or received by LAN\_East are shown in the case when there is no malicious blackholing or any solution is applied.

Figure B-9 shows the baseline HTTP traffic sent from LAN\_East.

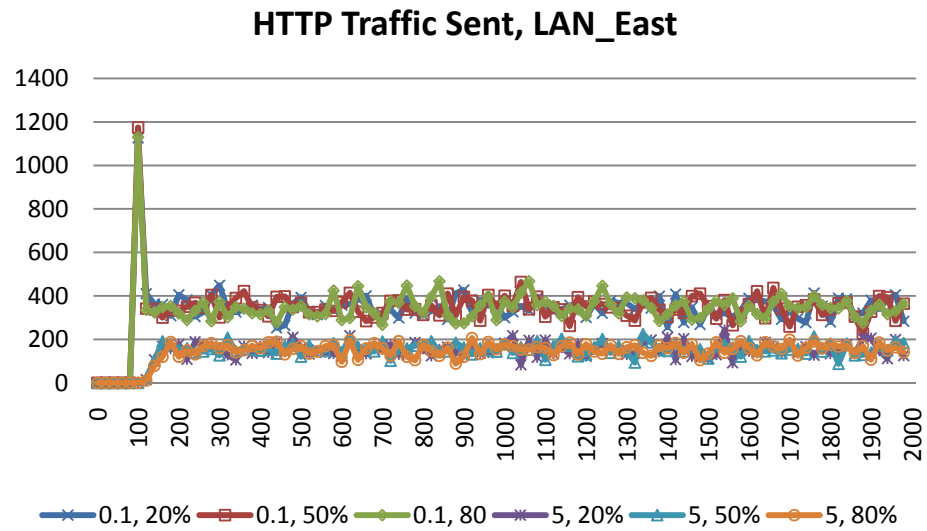


Figure B-9 Baseline HTTP traffic Sent of LAN\_East.

Figure B-10 shows the baseline HTTP traffic received by LAN\_East.



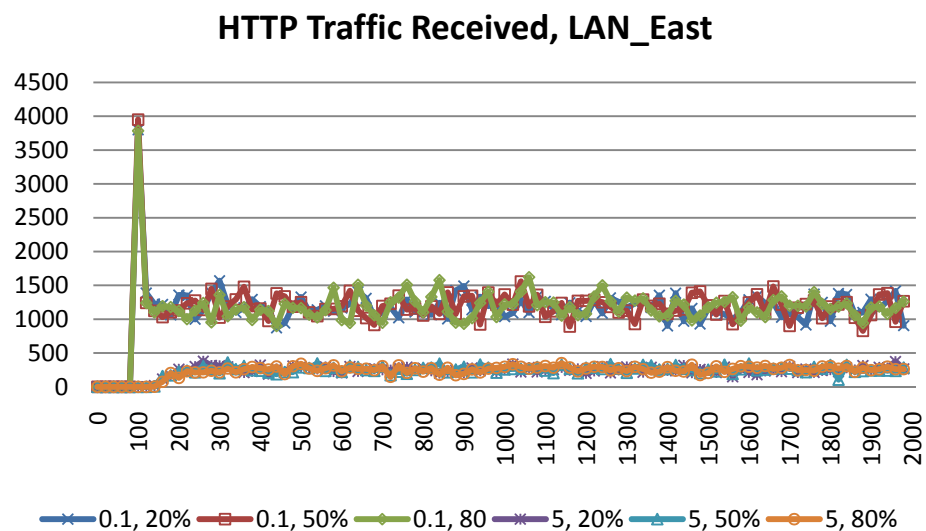


Figure B-10 Baseline HTTP Traffic Received in LAN\_East.

Figure B-11 shows the FTP traffic sent by LAN\_East.

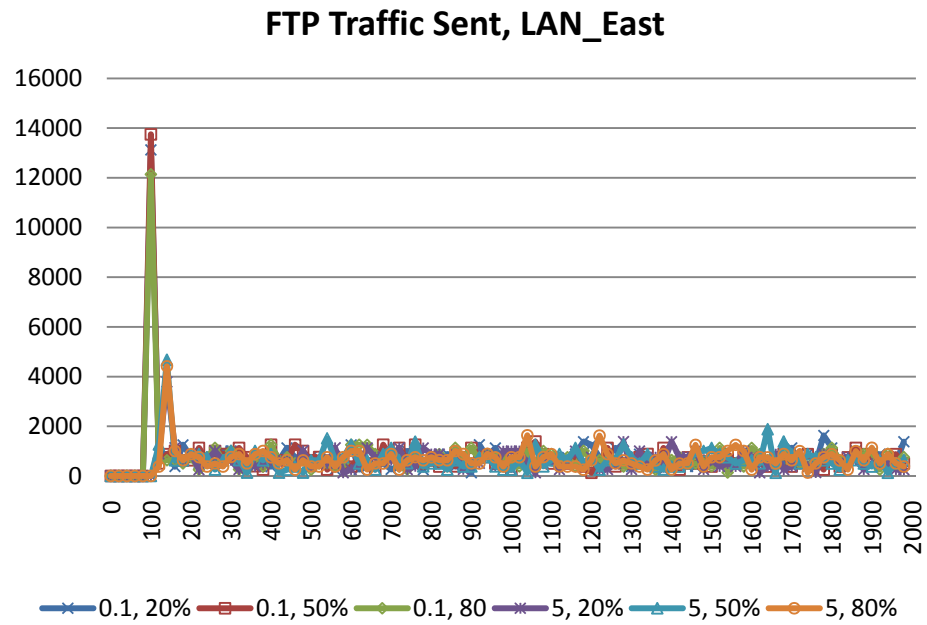


Figure B-11 Baseline FTP Traffic Sent in LAN\_East.

Figure B-12 shows the baseline FTP traffic received by LAN\_East.

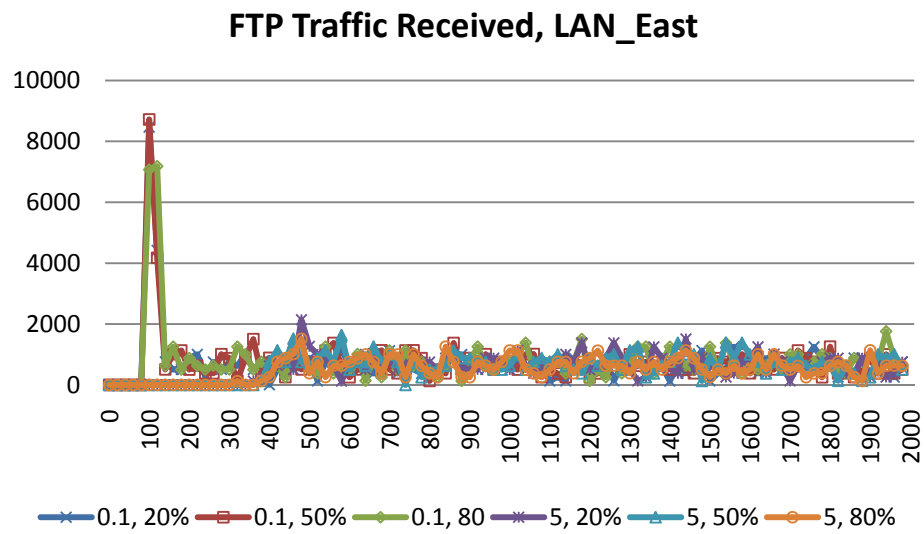


Figure B-12 Baseline FTP Traffic Received in LAN\_East.

Figure B-13 shows the baseline VOIP traffic sent by LAN\_East.

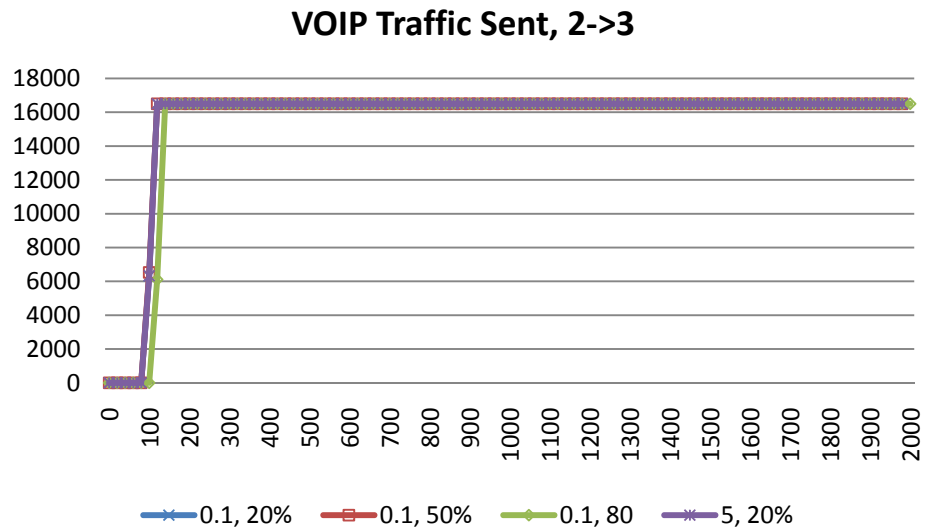


Figure B-13 Baseline VOIP Traffic Sent from Router 2 to Router 3.

Figure B-14 shows the baseline VOIP traffic received by LAN\_East.

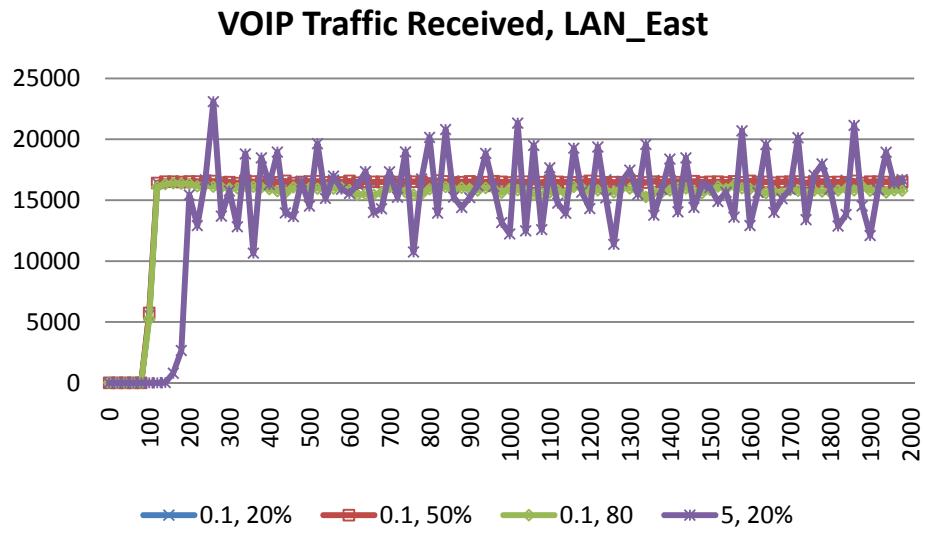


Figure B-14 Baseline VOIP Traffic Received in LAN\_East.

## ***Appendix C***

### ***CODE CHANGE***

#### ***C.1 CHANGES IN BGP MODULE***

##### ***C.1.1 Changes in bgp Process***

As shown in section CHAPTER 5section 5.5.2 that a number of states are added as shown in Figure C-1.

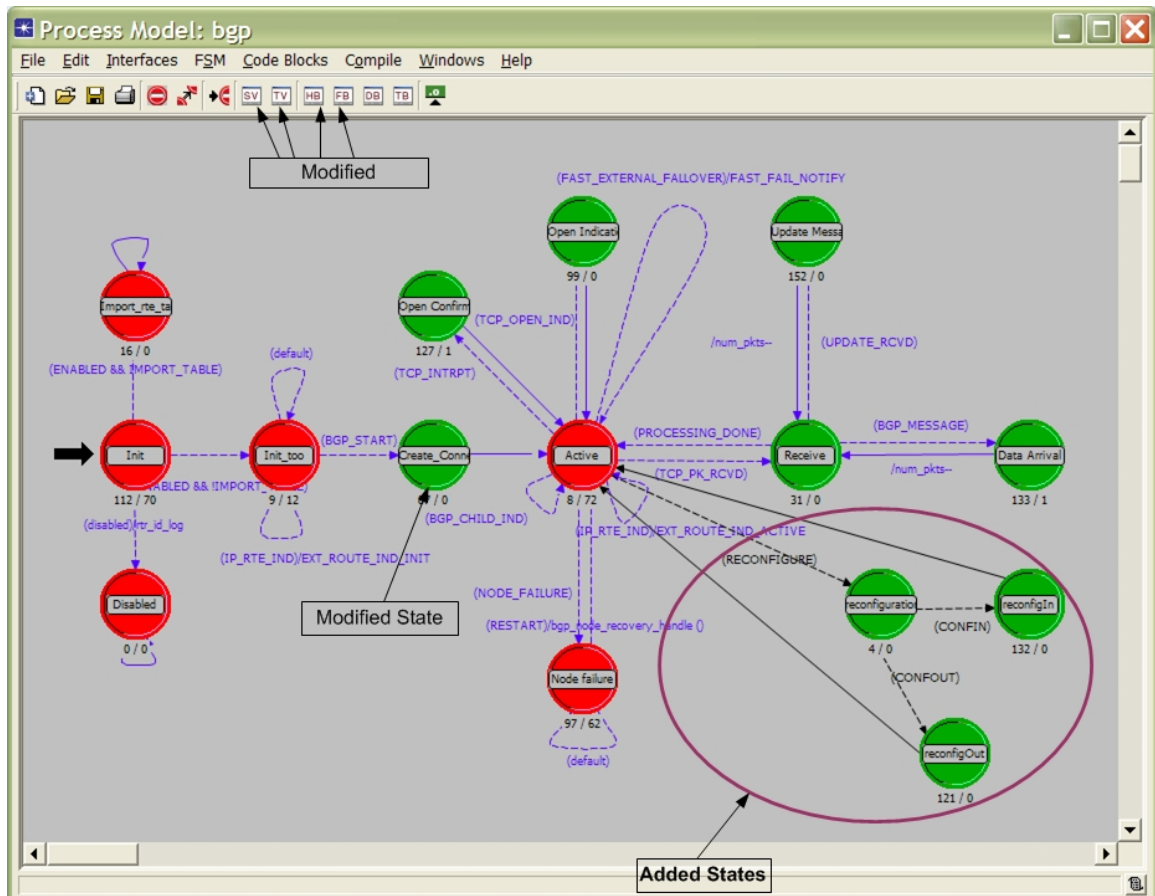


Figure C-1 Location of modifications in bgp process

### C.1.1.1 State Variables

In state variable the `scheduled_reconfiguration` state is added

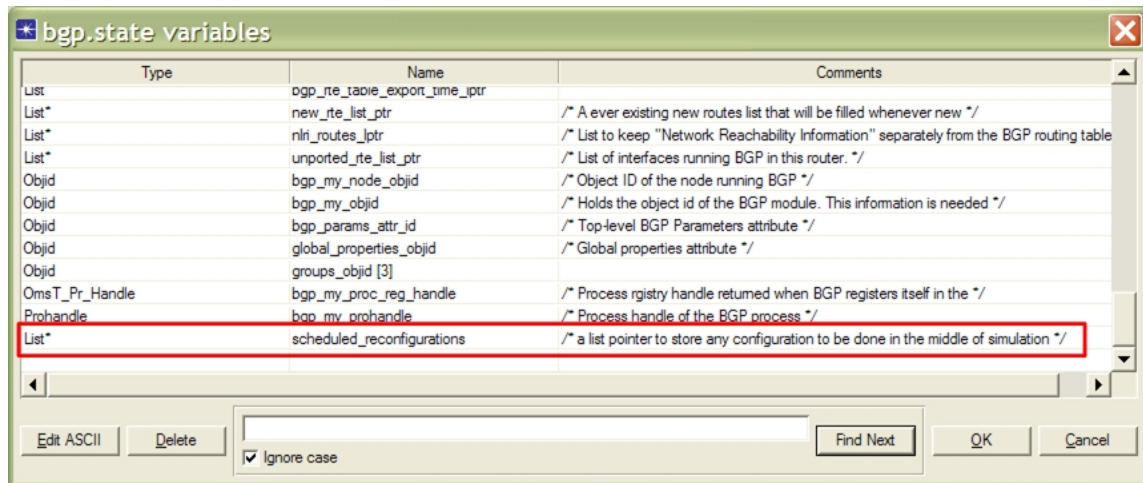


Figure C-2 Schedule Reconfiguration state variable addition

### C.1.1.2 Changes in Header Block

Code Snippet C-1 shows the definitions of conditions to move to added states.

```
/* Alrefai Code Snippet Start */
#define RECONFIGURE (intrpt_type == OPC_INTRPT_SELF) &&(intrpt_code >= 0)
#define CONFIN (isIn == OPC_TRUE)
#define CONFOUT (isIn == OPC_FALSE)
/* End Code Snippet*/
```

Code Snippet C-1 Definitions of conditions in header block.

Code Snippet C-2 shows the definition of Timed\_Policy structure to store information to schedule BGP reconfiguration.



```

/* Alrefai Code Snippet Start */
/*structure for storing scheduled route maps*/
typedef struct Timed_Policy{
    IpT_Rte_Policy* rte_policy_ptr;
    double time;
    int neighbor_as;
    Boolean isIn;
    Boolean isMoreSpecific;
    BgpT_Mp_Prefix* prefix_ptr;
    List* mp_prefixes_list;
}Timed_Policy;
/* End Code Snippet*/

```

**Code Snippet C-2 Definition of Timed\_Policy in Header Block**

Code Snippet C-3 shows the definition of the method that reads the policy from the user configuration. The parameter `int addr_family_attr_id` is added.

```

/* Alrefai Code Snippet Start */
static void bgp_neighbor_policy_info_read (Objid
ith_neighbor_info_id, List** incoming_update_pib_ptr, List**
outgoing_update_pib_ptr, int neighbor_as_number, int
addr_family_attr_id, const char* vrf_name);
/* End Code Snippet*/

```

**Code Snippet C-3 Definition of modified policy read method**

### **C.1.1.3 Modifications in function block**

The method `bgp_neighbor_policy_info_read` is modified, the definition of the function is shown in Code Snippet C-4.

```

/* Alrefai Code Snippet Start */
static void
bgp_neighbor_policy_info_read (Objid ith_neighbor_info_id, List**
incoming_update_pib_ptr, List** outgoing_update_pib_ptr, int
neighbor_as_number, int addr_family_attr_id, const char* vrf_name)
/* End Code Snippet*/

```

#### Code Snippet C-4 Modification of Definition of Reading Policy Function

Code Snippet C-5 shows the added local variables to  
bgp\_neighbor\_policy\_info\_read function.

```

/* Alrefai Code Snippet Start */
    IpT_Rte_Policy*      rte_policy_ptr;
    Timed_Policy*        a_timed_policy;
    BgpT_Update_Dir      direction;
    double                time;
/* End Code Snippet*/

```

#### Code Snippet C-5 Local Variables Added to Read Policy Function.

Code Snippet C-6 shows the allocation of memory for Timed\_Policy inside  
bgp\_neighbor\_policy\_info\_read function.

```

/* Alrefai Code Snippet Start */
/* Read the policies and insert them into appropriate lists */
for (count_j = 0; count_j < num_applicable_policies; count_j++)
{
    a_timed_policy = (Timed_Policy*) op_prg_mem_alloc
(sizeof(Timed_Policy));
/* End Code Snippet*/

```

**Code Snippet C-6 Allocating Memory to Timed\_Policy Variable**

Code Snippet C-7 shows the read of the time of applying the time

```

/* Alrefai Code Snippet Start */
/* Read the policies and insert them into appropriate lists */
/* Obtain the applicable direction. */
op_ima_obj_attr_get (jth_policy_info_id, "Applicable Direction",
&direction);

/* Obtain the time */
op_ima_obj_attr_get (jth_policy_info_id, "time", &time);
/* End Code Snippet*/

```

**Code Snippet C-7 Reading the Time**

Code Snippet C-8 shows the code of storing Timed\_Policy in Rescheduled\_Configuration.

```

/* Alrefai Code Snippet Start */
/* Store the route map into into the list of route maps */
if (direction == BgpC_Update_Dir_In)
{
    if(time == 0.0)
    {
        /* Insert the route map into incoming update PIB */
        /* in the same order as they appear in the table */
        op_prg_list_insert (incoming_update_pib, rte_policy_ptr,
        OPC_LISTPOS_TAIL);
    }
    else
    {
        a_timed_policy->rte_policy_ptr = rte_policy_ptr;
        a_timed_policy->time = time;
        a_timed_policy->neighbor_as = neighbor_as_number;
        a_timed_policy->isIn = OPC_TRUE;
        a_timed_policy->isMoreSpecific = OPC_FALSE;
        printf("time should not be zero %d\n", (int)a_timed_policy-
        >time);
        op_prg_list_insert (scheduled_reconfigurations,
        a_timed_policy, OPC_LISTPOS_TAIL);
    }
}
else
{
    if(time == 0.0)
    {
        /* Insert the route map into incoming update PIB */
        op_prg_list_insert (outgoing_update_pib, rte_policy_ptr,
        OPC_LISTPOS_TAIL);
    }
    else
    {
        a_timed_policy->rte_policy_ptr = rte_policy_ptr;
        a_timed_policy->time = time;
        a_timed_policy->neighbor_as = neighbor_as_number;
        a_timed_policy->isIn = OPC_FALSE;
        a_timed_policy->isMoreSpecific = OPC_FALSE;
        op_prg_list_insert (scheduled_reconfigurations,
        a_timed_policy, OPC_LISTPOS_TAIL);
    }
}
/* End Code Snippet*/

```

Code Snippet C-8 Storing timed Policy in Scheduled Reconfiguration

Code Snippet C-9 shows the call of `bgp_neighbor_policy_info_read` inside `bgp_nbr_addr_family_params_read`.

```
/* Alrefai Code Snippet Start */
/* Read routing policy information. */
bgp_neighbor_policy_info_read (addr_family_id, &incoming_update_pib,
                               &outgoing_update_pib, bgp_conn_info_ptr->neighbor_as_number,
                               addr_family_attr_int, vrf_name);
/* End Code Snippet*/
```

**Code Snippet C-9 Modified use of reading policy function in `bgp_nbr_addr_family_params_read` function**

Code Snippet C-10 shows the creation of `scheduled_reconfiguration` list defined in the function `bgp_sv_init ()` in the function block.

```
/* Alrefai Code Snippet Start */
/* Initialization of scheduled reconfiguration list in          */
/* order to store any timed_policy to be conducted in          */
/* the middle of simulation                                     */
scheduled_reconfigurations = op_prg_list_create ();
/* End Code Snippet*/
```

**Code Snippet C-10 Creation of reconfiguration list done in `bgp_sv_init`**

#### **C.1.1.4 Reconfigure state**

Code Snippet C-11 shows the code of Reconfigure state. In this state the `timed_policy` is accessed to know which

```
/* Alrefai Code Snippet Start */
a_timed_policy =
(Timed_Policy*)op_prg_list_access(scheduled_reconfigurations , intrpt_code);
isIn = a_timed_policy->isIn;
/* End Code Snippet*/
```

**Code Snippet C-11 Reconfigure State Code**

#### **C.1.1.5 ReconfigIn State**

Code Snippet C-12, Code Snippet C-13, and Code Snippet C-14 show the code of ReconfigureIn State. In this state, the child process is called to handle the 'rib-in' database, modify them in order to control the outgoing traffic.

```

/* Alrefai Code Snippet Start */
In this state we need to get the Bgp_Conn_Info caused the interrupt, and
invoke the corresponding child, pass to the child the corresponding route
map that need to be changed, [the child should modify rib in if any change
happen] so if there any change the route chosen as best we need to update
local rib, routing tables and propagate to neighbor **/

num_entries = op_prg_list_size(bgp_connections_list_ptr);
for(count_i = 0; count_i < num_entries; count_i++)
{
    bgp_conn_info_ptr = op_prg_list_access (bgp_connections_list_ptr,
count_i);
    if(bgp_conn_info_ptr->neighbor_as_number == a_timed_policy-
>neighbor_as)
        break;
}
op_pro_invoke (bgp_conn_info_ptr->bgp_connection_prohandle, a_timed_policy-
>rte_policy_ptr);

/* If the packet contained unreachable destinations, they have to be */
/* removed from the Loc-RIB. */
if (bgp_conn_info_ptr->unreachable_rte_exists == OPC_TRUE)
{
    /* Get the list of unreachable routes form the mailbox area.*/
    unreachable_list_ptr = bgp_conn_info_ptr-
>unreachable_rte_list_ptr;
    if (ip_node_is_pe (ip_module_data_ptr) &&
        (bgp_conn_info_ptr->neighbor_site_vrf_name != OPC_NIL))
        bgp_prefix_list_ipv4_to_vpnv4_convert (unreachability_list_ptr,
bgp_conn_info_ptr->neighbor_site_vrf_name);

    /* Process the information and re-set the flag. */
    bgp_unfeasible_routes_process (unreachability_list_ptr,
bgp_conn_info_ptr->peer_id);
    /* Make sure that the flag is reset to false so that the next set of*/
    /* unfeasible routes can be properly communicated. */
    bgp_conn_info_ptr->unreachable_rte_exists = OPC_FALSE;
}
else
{
    /* Force the unreachability list pointer to be NULL. This value */
    /* will be passed to the procedure that will propagate the new */
    /* status of the Local-RIB to all the neighbors. */
    unreachability_list_ptr = OPC_NIL;
}
/* Check to see if new routes have been added to the RIB-In */

```

Code Snippet C-12 ReconfigureIn Code

```

/* Alrefai Code Snippet Continue... */
if (bgp_conn_info_ptr->adj_rib_in_ptr->num_new_routes > 0)
{
    /* Collect the new routes from the temporary list into the new */
    /* routes list. */
    for (count_i = 0; count_i < bgp_conn_info_ptr->adj_rib_in_ptr-
>num_new_routes; count_i++)
    {
        /* All new routes should be on top of the list. Access the */
        /* the new routes and add them to the list. */
        new_rte_entry_ptr = (BgpT_Rte_Entry*) op_prg_list_remove
(bgp_conn_info_ptr->adj_rib_in_ptr->new_routes_lptr, OPC_LISTPOS_HEAD);
        /* Set the admin distance on the new routes. */
        if (BgpC_Conn_Type_Ebgp == bgp_conn_info_ptr->bgp_connection_type)
        {
            new_rte_entry_ptr->admin = admin_distance;
        }
        else
        {
            new_rte_entry_ptr->admin = ibgp_admin_distance;
        }
        /* Check if this new entry is from a VPN site */
        /* */
        if (ip_node_is_pe (ip_module_data_ptr) &&
            (bgp_conn_info_ptr->neighbor_site_vrf_name != OPC_NIL))
        {
            /* Check if this new entry is from a VPN site. And RD and */
            /* values are not set for this route. If it is */
            /* then set the route distinguisher value for the entry */
            bgp_new_rte_at_vpn_pe_process (new_rte_entry_ptr, bgp_conn_info_ptr-
>neighbor_site_vrf_name);
        }
        op_prg_list_insert (new_rte_list_ptr, new_rte_entry_ptr, OPC_LISTPOS_HEAD)
        /* Continue till all the new routes have been inserted. */
    }
    /* Call the procedure that will process the new routes. */
    bgp_reachability_info_process (bgp_conn_info_ptr);
    /* Reset the number of new routes to 0 and destroy the temporary */
    /* list. */
    bgp_conn_info_ptr->adj_rib_in_ptr->num_new_routes = 0;
    op_prg_mem_free (bgp_conn_info_ptr->adj_rib_in_ptr->new_routes_lptr);
}

/* Find out the number of new routes that were entered into the local */
/* routing table. This would not only be the number of routes that */
/* that were received as a part of the advertisement, but also could */
/* contain the replacement routes that were selected after certain */
/* routes were termed infeasible. */
number_of_new_routes = op_prg_list_size (new_rte_list_ptr);

```

Code Snippet C-13 Continue ReconfigureIn Code



```

/* Alrefai Code Snippet Continue...*/

/* If any of the list is valid, then all the peer processes have to */
/* be notified about the change in the routing table status. */
if ((number_of_new_routes > 0) || unreachable_list_ptr != OPC_NIL)
{
/* Unless this is a dummy node representing an external AS, */
/* propagate the new routes to all the other neighbors. */
if (OPC_FALSE == is_external_as_node)
{
bgp_new_routes_propagate (unreachability_list_ptr, number_of_new_routes,
bgp_conn_info_ptr->peer_id);
}

/* Clean up just the new_rte_list_ptr by removing all the route */
/* entries in it. Be sure not to free up the memory of the */
/* route entries as these entries are used by the route tables. */
for (count_i = 0; count_i < number_of_new_routes; count_i++)
{
/* remove the routes entries from the new route list. */
op_prg_list_remove (new_rte_list_ptr, OPC_LISTPOS_HEAD);
}

/* Clean up the unreachable routes list. The prefixes in this list */
/* can be freed up. The will not be reference by any route entry. */
if (unreachability_list_ptr != OPC_NIL)
{
/* Destroy the list of unreachable routes.*/
bgp_support_rte_list_destroy (unreachability_list_ptr);
bgp_conn_info_ptr->unreachable_rte_list_ptr = OPC_NIL;
}
}
/* End Code Snippet*/

```

Code Snippet C-14 Continue ReconfigureIn Code

**C.1.1.6 ReconfigOut State**

```

/* Alrefai Code Snippet Start */
num_entries = op_prg_list_size(bgp_connections_list_ptr);
for(count_i = 0; count_i < num_entries; count_i++)
{
    bgp_conn_info_ptr = op_prg_list_access (bgp_connections_list_ptr, count_i);
    if(bgp_conn_info_ptr->neighbor_as_number == a_timed_policy->neighbor_as)
        break;
}
rte_list_ptr = bgp_conn_info_ptr->adj_rib_out_ptr->rte_list_ptr;
num_entries = op_prg_list_size (rte_list_ptr);
new_rte_list_ptr = op_prg_list_create();
if (! a_timed_policy->isMoreSpecific)
{
    for (count_i = 0; count_i < num_entries; count_i++)
    {
        rte_entry_ptr = (BgpT_Rte_Entry*) op_prg_list_access (rte_list_ptr,
count_i);
        new_rte_entry_ptr = bgp_support_rte_entry_copy(rte_entry_ptr);
        rte_maps = op_prg_list_create();
        op_prg_list_insert(rte_maps, a_timed_policy->rte_policy_ptr,
OPC_LISTPOS_TAIL);
        if(bgp_support_rte_filter_policy_apply(&new_rte_entry_ptr, rte_maps,
OPC_NIL, OPC_NIL, OPC_FALSE, &policy_edited,
        bgp_conn_info_ptr->bgp_connection_type, bgp_conn_info_ptr-
>local_info_ptr) == OPC_TRUE)
        {
            if (policy_edited == OPC_TRUE)
            {
                //bgp_support_rte_entry_print(rte_entry_ptr);
                new_mp_prefix_ptr = (BgpT_Mp_Prefix*)
bgp_support_mp_prefix_copy(rte_entry_ptr->dest_prefix_ptr);
                bgp_support_ith_rte_entry_replace (bgp_conn_info_ptr-
>adj_rib_out_ptr, count_i, new_rte_entry_ptr);
                op_prg_list_insert (new_rte_list_ptr, new_rte_entry_ptr,
OPC_LISTPOS_TAIL);
            }
        }
    }
}

```

**Code Snippet C-15 ReconfigureOut**

```

/* Alrefai Code Snippet Continue...*/
else
{
    /* Restrict this route. */
    bgp_support_rte_entry_destroy (new_rte_entry_ptr);

    /* Update the statistics that indicate the number */
    /* routes that were dropped due to route policies. */
    op_stat_write (bgp_conn_info_ptr->local_info_ptr-
>bgp_local_stats.num_policy_discards_local_stat_hdl, 1.0);
    op_stat_write (bgp_conn_info_ptr->local_info_ptr-
>bgp_local_stats.num_policy_discards_local_stat_hdl, 0.0);
}
}
if (ip_node_is_pe (ip_module_data_ptr) &&
    (bgp_conn_info_ptr->neighbor_site_vrf_name != OPC_NIL))
    bgp_prefix_list_ipv4_to_vpnv4_convert (bgp_conn_info_ptr-
>unreachable_rte_list_ptr, bgp_conn_info_ptr->neighbor_site_vrf_name);

number_of_new_routes = op_prg_list_size (new_rte_list_ptr);
//bgp_support_rte_table_print(bgp_conn_info_ptr->adj_rib_out_ptr);

bgp_conn_info_ptr->adj_rib_out_ptr->new_routes_lptr = op_prg_list_create();
bgp_conn_info_ptr->adj_rib_out_ptr->num_new_routes = 0;
for(count_i = 0; count_i < number_of_new_routes; count_i++)
{
    op_prg_list_insert(bgp_conn_info_ptr->adj_rib_out_ptr-
>new_routes_lptr,
    bgp_support_rte_entry_copy((Bgpt_Rte_Entry*)
op_prg_list_access(new_rte_list_ptr, count_i)), OPC_LISTPOS_TAIL);
}

```

Code Snippet C-16 Continue ReconfigureOut Code

```

        /* Alrefai Code Snippet Continue...*/
bgp_conn_info_ptr->adj_rib_out_ptr->num_new_routes =
op_prg_list_size(new_rte_list_ptr);
if(bgp_conn_info_ptr->adj_rib_out_ptr->num_new_routes > 0 ||
bgp_conn_info_ptr->unreachable_rte_exists == OPC_TRUE)
{
    op_pro_invoke (bgp_conn_info_ptr->bgp_connection_prohandle, OPC_NIL);
    for (count_i = 0; count_i < number_of_new_routes; count_i++)
    {
else
{
    /* Here we want to handle more specific prefixes*/
    /* read the prefix and the number of bits to divide*/
    /* search for it in rib out */
    /* if found store route attribute */
    /* divide it into list of prefixes */
    /* create the route with the same path attribute of rib out */
    /* add the routes to rib out */
    /* send it to the specific neighbor by invoking the process!! */

mp_prefix_ptr = a_timed_policy->prefix_ptr;
rte_entry_ptr = bgp_support_rte_entry_find (bgp_conn_info_ptr-
>adj_rib_out_ptr, mp_prefix_ptr, &location);
if (rte_entry_ptr != OPC_NIL)
{
    num_prefixes = op_prg_list_size (a_timed_policy->mp_prefixes_list);
    if (num_prefixes > 0)
    {
        bgp_conn_info_ptr->adj_rib_out_ptr->new_routes_lptr =
op_prg_list_create();
        bgp_conn_info_ptr->adj_rib_out_ptr->num_new_routes = 0;
        for (count_i = 0; count_i < num_prefixes; count_i++)
        {
            new_mp_prefix_ptr = op_prg_list_access (a_timed_policy-
>mp_prefixes_list, count_i);

```

Code Snippet C-17 Continue ConfigureOut Code

```

/* Alrefai Code Snippet Continue...*/
    new_rte_entry_ptr =
bgp_support_rte_entry_copy(rte_entry_ptr);
    new_rte_entry_ptr->dest_prefix_ptr = new_mp_prefix_ptr;
    op_prg_list_insert(bgp_conn_info_ptr->adj_rib_out_ptr-
>new_routes_lptr, new_rte_entry_ptr, OPC_LISTPOS_TAIL);

    if(bgp_conn_info_ptr->adj_rib_out_ptr->num_new_routes > 0)
    {
        op_pro_invoke (bgp_conn_info_ptr-
>bgp_connection_prohandle, OPC_NIL);
        for (count_i = 0; count_i < number_of_new_routes;
count_i++)
        {
            // remove the routes entries from the new route
list.
            op_prg_list_remove (new_rte_list_ptr,
OPC_LISTPOS_HEAD);
        }
    }
}
/* End Code Snippet*/

```

Code Snippet C-18 Continue ConfigureOut Code

### ***C.1.2 Modification in bgp\_conn Process***

The modification of bgp\_conn are done in two locations inside ESTABLISHED state and in the function block.

#### ***C.1.2.1 Established State***

In order to handle the self interrupt in the established state the code shown in Code Snippet C-19 is added.

```

/* Alrefai Code Snippet Start*/
case OPC_INTRPT_SELF:
{
    if (intrpt_code >= 0)
    {
        rte_policy_ptr = (IpT_Rte_Policy*) op_pro_argmem_access ();

        if (rte_policy_ptr == OPC_NIL)
        {
            bgp_conn_new_route_indication_handle ();
        }
        else
        {
            bgp_conn_apply_map_rib_in (rte_policy_ptr);
        }
    }

    /* The Hold timer has expired or the parent node has failed */
    /* Move to the IDLE state. */
    break;
}
/* End Code Snippet*/

```

**Code Snippet C-19 Established State Modification in bgp\_conn Process**

### **C.1.2.2 Function Block**

Code Snippet C-20 and Code Snippet C-21 show the definition of `bgp_conn_apply_map_rib_in`. This function apply the scheduled map on the rib-in database.

```

/* Alrefai Code Snippet Start */
static void bgp_conn_apply_map_rib_in (IpT_Rte_Policy* rte_policy_ptr)
{
    List* rte_maps;
    Boolean edit_status;
    List* rte_list_ptr;
    BgpT_Rte_Entry* rte_entry_ptr;
    BgpT_Rte_Entry* new_rte_entry_ptr;
    int num_entries;
    int count_i;
    /* what needed to be done
    1. loop the rib in      2. copy each entry
    3. apply policy        4. if accepted process the new route
    */
    FIN(bgp_conn_apply_map_rib_in (rte_policy_ptr));
    rte_list_ptr = bgp_my_adj_rib_in_ptr->rte_list_ptr;
    num_entries = op_prg_list_size (rte_list_ptr);
    bgp_my_adj_rib_in_ptr->new_routes_lptr = op_prg_list_create ();
    for (count_i = 0; count_i < num_entries; count_i++)
    {
        rte_entry_ptr = (BgpT_Rte_Entry*) op_prg_list_access
(rte_list_ptr, count_i);
        new_rte_entry_ptr = bgp_support_rte_entry_copy(rte_entry_ptr);
        /* because the method of applying policy only accept list of policies
        we need to create a list and insert rte_policy_ptr to it.*/
        rte_maps = op_prg_list_create();
        op_prg_list_insert(rte_maps, rte_policy_ptr, OPC_LISTPOS_TAIL);
    }
}

```

**Code Snippet C-20 Definition of `bgp_conn_apply_map_rib_in`.**

```

/* Alrefai Code Snippet Continue...*/
    if(bgp_support_rte_filter_policy_apply(&new_rte_entry_ptr,
rte_maps, OPC_NIL, OPC_NIL, OPC_TRUE, &edit_status, BgpC_Conn_Type_None,
conn_info_ptr->local_info_ptr) == OPC_TRUE)
    {
        if(edit_status)
        {
            bgp_conn_route_entry_process(new_rte_entry_ptr);
        }
    }
    else
    {
        //bgp_conn_previously_advertised_route_check
(new_rte_entry_ptr->dest_prefix_ptr);
        //bgp_support_rte_entry_destroy (new_rte_entry_ptr);
    }
}
if (bgp_my_adj_rib_in_ptr->num_new_routes == 0)
    op_prg_mem_free (bgp_my_adj_rib_in_ptr->new_routes_lptr);
FOUT;
}
/* End Code Snippet*/

```

Code Snippet C-21 Continue definition of `bgp_conn_apply_map_rib_in`.

### C.1.3 Shortening

For shortening, a change in the file `bgp_support.ex.c` `bgp_support_as_path_prepend` is to add the calling of the method that do the shortening `bgp_support_as_path_remove_first`. This is shown in Code Snippet C-22.



```
/* Alrefai Code Snippet start */
static void bgp_support_as_path_prepend (BgpT_Path_Attrs* path_attrs_ptr,
const IpT_Rte_Map_AsPath_List* as_list_ptr) {
    int*                new_segment_value_array;
    int                 as_seg_index;
    int                 as_list_index;
    int                 seg_length;
    BgpT_Path_Segment* path_segment_ptr;
    /** Prepend the ASes specified in the list to the AS Path. **/
    FIN (bgp_support_as_path_prepend (path_attrs_ptr, as_list_ptr));

    if (0 == as_list_ptr->num_as_numbers)
    {
        bgp_support_as_path_remove_first(path_attrs_ptr);
        FOUT;
    }
/* End Code Snippet*/
```

**Code Snippet C-22 Modification of Prepend Function to Call the Shorten Function**

The function `bgp_support_as_path_remove_first` is shown in Code Snippet C-23, Code Snippet C-24.

```

/* Alrefai Code Snippet Start*/
void
bgp_support_as_path_remove_first (BgpT_Path_Attrs* orig_path_attrs_ptr)
{
    int*          new_segment_value_array;
    int           ith_elem;
    int           seg_length;
    BgpT_Path_Segment* ith_path_segment_ptr;
    /** This function the last AS added to the first place of the list **/
    FIN (bgp_support_as_path_remove_first (orig_path_attrs_ptr));
    ith_path_segment_ptr = (BgpT_Path_Segment *)
        op_prg_list_access (orig_path_attrs_ptr->as_path_list_ptr,
        OPC_LISTPOS_TAIL);
    /* Find the length of the segment value. */
    seg_length = ith_path_segment_ptr->segment_length;
    if(seg_length <= 1)
    {
        //printf("I am changing the the segment insider");
        ith_path_segment_ptr = bgp_support_path_seg_mem_alloc ();
        ith_path_segment_ptr->segment_type =
        BgpC_Path_Seg_Type_As_Sequence;
        ith_path_segment_ptr->segment_length = 0;
        ith_path_segment_ptr->segment_value_array = OPC_NIL;
        op_prg_list_remove (orig_path_attrs_ptr->as_path_list_ptr,
        OPC_LISTPOS_TAIL);
        op_prg_list_insert (orig_path_attrs_ptr->as_path_list_ptr,
        ith_path_segment_ptr, OPC_LISTPOS_TAIL);
    }
    else
    {
        /* The memeber segment value is a array of AS numbers. Copy */
        /* that into a new array. */
        new_segment_value_array = (int*) prg_cmo_alloc
        (bgp_as_path_list_cmh, (seg_length-1)*sizeof (int));

        /* Done with adding. Exit the function */
        FOUT;
    }
}/* End Code Snippet*/

```

Code Snippet C-23 Definition of bgp\_support\_as\_path\_remove\_first or Shorten Function.

```

/* Alrefai Code Snippet Continue...*/
    for (ith_elem = 0; ith_elem < seg_length; ith_elem++)
    {
    }
    /* Copy the elements of the original array into new array. */
    for (ith_elem = 1; ith_elem < seg_length; ith_elem++)
    {
        new_segment_value_array [ith_elem - 1] =
ith_path_segment_ptr->segment_value_array [ith_elem];
    }
    /* Free up the Old segment value array. */
    if (seg_length > 0)
        op_prg_mem_free (ith_path_segment_ptr-
>segment_value_array);
    /* set the new value and increment the segment length. */
    ith_path_segment_ptr->segment_value_array =
new_segment_value_array;
    (ith_path_segment_ptr->segment_length)--;
    seg_length = ith_path_segment_ptr->segment_length;
}
--orig_path_attrs_ptr->as_path_length;
/* Done with adding. Exit the function */
FOUT;
}
/* End Code Snippet*/

```

**Code Snippet C-24 Continue Definition of Shorten Function.**

but this method needs to be defined in bgp\_support\_defs.h file as shown below

```

/* Alrefai Code Snippet Continue...*/
void bgp_support_as_path_remove_first (BgpT_Path_Attrs*
orig_path_attrs_ptr);
/* End Code Snippet*/

```

### ***C.1.4 More Specific Prefixes***

Changes are done in the function block.

The `bgp_nbr_addr_family_params_read` function is changed; lines of code shown in are added.

```
/* Alrefai Code Snippet Start */
/*****
/* More SPecific Prefixes */
/*****
/* Read scheduled more specific prefixes */
bgp_neighbor_more_specific_prefix_read (addr_family_id, addr_family,
bgp_conn_info_ptr->neighbor_as_number);
/* End Code Snippet*/
```

**Code Snippet C-25 Modifying `bgp_nbr_addr_family_params_read` function to call  
`bgp_neighbor_more_specific_prefix_read`**

This method is added `bgp_neighbor_more_specific_prefix_read` is shown in

```

/* Alrefai Code Snippet Start */
static void
bgp_neighbor_more_specific_prefix_read (Objid ith_neighbor_info_id,
InetT_Addr_Family addr_family, int neighbor_as_number)
{
    Objid                msp_objid;
    Objid                jth_msp_info_id, jth_prefix_info_id;
    Objid                prefix_id;
    Objid                first_prefix_id;
    Objid                prefixes_id;
    int                  num_msps;
    int                  count_i, count_j;
    char                 addr_str[64];
    InetT_Address        ntwk_addr, masked_ntwk_addr;
    InetT_Subnet_Mask    inet_smask;
    IpT_Address          ipv4_smask;
    BgpT_Ip_Prefix*      prefix_ptr;
    int                  num_prefixes;
    double               time;
    Timed_Policy*        timed_policy_ptr;
    BgpT_Mp_Prefix*      mp_prefix_ptr;

    FIN (bgp_neighbor_more_specific_prefix_read (ith_neighbor_info_id,
addr_family));

    op_ima_obj_attr_get (ith_neighbor_info_id, "More Specific Prefix",
&msp_objid);
    num_msps = op_topo_child_count (msp_objid, OPC_OBJTYPE_GENERIC);
    if (num_msps > 0)
    {
        timed_policy_ptr = (Timed_Policy*) op_prg_mem_alloc
(sizeof(Timed_Policy));
        timed_policy_ptr->isIn = OPC_FALSE;
        timed_policy_ptr->isMoreSpecific = OPC_TRUE;
        timed_policy_ptr->mp_prefixes_list = op_prg_list_create();
    }
    for (count_i = 0; count_i < num_msps; count_i++)
    {
        jth_msp_info_id = op_topo_child (msp_objid,
OPC_OBJTYPE_GENERIC, count_i);

        op_ima_obj_attr_get (jth_msp_info_id, "Prefix", &prefix_id);

```

**Code Snippet C-26 Definition of bgp\_neighbor\_more\_specific\_prefix\_read Function.**

```

/* Alrefai Code Snippet Start */
    first_prefix_id = op_topo_child (prefix_id,
OPC_OBJTYPE_GENERIC, 0);
    op_ima_obj_attr_get (first_prefix_id, "IP Address",
addr_str);
    ntwk_addr = inet_address_create (addr_str, addr_family);
    if (!inet_address_valid (ntwk_addr))
{bgp_invaidd_network_address_log_write (count_i, addr_str); continue;}
    op_ima_obj_attr_get (first_prefix_id, "Mask", addr_str);
    if (InetC_Addr_Family_v4 == addr_family)
        {if (0 == strcmp (addr_str,
BGPC_SUBNET_MASK_AUTO_ASSIGN_STR))
            {ipv4_smask = ip_default_smask_create
(inet_ipv4_address_get(ntwk_addr));}
            else{ipv4_smask = ip_address_create (addr_str);
                if (ip_address_equal (IPC_ADDR_INVALID,
ipv4_smask))
                    {bgp_invaidd_subnet_mask_log_write
(count_i, addr_str);continue;}}
            inet_smask = inet_smask_from_ipv4_smask_create
(ipv4_smask);
        }
    else{continue;}
    masked_ntwk_addr = inet_address_mask (ntwk_addr,
inet_smask);
    inet_address_destroy (ntwk_addr);
    prefix_ptr = inet_address_range_mem_alloc ();
    *prefix_ptr = inet_address_range_create_fast
(masked_ntwk_addr, inet_smask);
    mp_prefix_ptr =
bgp_support_mp_prefix_from_ip_prefix(prefix_ptr);
    timed_policy_ptr->prefix_ptr =
bgp_support_mp_prefix_copy(mp_prefix_ptr);

    op_ima_obj_attr_get (jth_msp_info_id, "Prefixes",
&prefixes_id);
    num_prefixes = op_topo_child_count (prefixes_id,
OPC_OBJTYPE_GENERIC);
    for (count_j = 0; count_j < num_prefixes; count_j++)
    {
        jth_prefix_info_id = op_topo_child (prefixes_id,
OPC_OBJTYPE_GENERIC, count_j);

```

**Code Snippet C-27** Continue of Definition of `bgp_neighbor_more_specific_prefix_read` Function.

```

/* Alrefai Code Snippet Start */
    op_ima_obj_attr_get (jth_prefix_info_id, "IP
Address", addr_str);
    ntwk_addr = inet_address_create (addr_str, addr_family);
    if (!inet_address_valid (ntwk_addr))
{bgp_invaidd_network_address_log_write (count_j, addr_str); continue;}
    op_ima_obj_attr_get (jth_prefix_info_id, "Mask", addr_str);
    if (InetC_Addr_Family_v4 == addr_family)
        {if (0 == strcmp (addr_str,
BGP_C_SUBNET_MASK_AUTO_ASSIGN_STR))
            {ipv4_smask = ip_default_smask_create
(inet_ipv4_address_get(ntwk_addr));}
            else{ipv4_smask = ip_address_create (addr_str);
                if (ip_address_equal (IPC_ADDR_INVALID,
ipv4_smask))
                    {bgp_invaidd_subnet_mask_log_write
(count_i, addr_str);continue;}}
            inet_smask = inet_smask_from_ipv4_smask_create
(ipv4_smask);
        }
        else{continue;}
        masked_ntwk_addr = inet_address_mask (ntwk_addr,
inet_smask);
        inet_address_destroy (ntwk_addr);
        prefix_ptr = inet_address_range_mem_alloc ();
        *prefix_ptr = inet_address_range_create_fast
(masked_ntwk_addr, inet_smask);
        mp_prefix_ptr =
bgp_support_mp_prefix_from_ip_prefix(prefix_ptr);
        op_prg_list_insert (timed_policy_ptr-
>mp_prefixes_list, mp_prefix_ptr, OPC_LISTPOS_TAIL);
    }
    op_ima_obj_attr_get (jth_msp_info_id, "Time", &time);
    timed_policy_ptr->time = time;
    timed_policy_ptr->neighbor_as = neighbor_as_number;
    op_prg_list_insert (scheduled_reconfigurations,
timed_policy_ptr, OPC_LISTPOS_TAIL);
}
    FOUT;
}
/* End Code Snippet*/

```

Code Snippet C-28 Continue of Definition of bgp\_neighbor\_more\_specific\_prefix\_read Function.

## ***C.2 MODIFICATION IN IP PROTOCOL***

### ***C.2.1 IP Dispatch Process***

In `ip_rte_support.h` file, the structure `IpT_Rte_Blackhole_From` is defined as shown in Code Snippet C-29. This structure is used to specify the prefixes to blackhole and the time to start blackholing.

```
/* Alrefai Code Snippet Start */
/*structure to specify a list of prefixes and time to start blackholing*/
typedef struct IpT_Rte_Blackhole_From
{
    //add here statistic to store the number of packets
    int number_of_packets;
    //add here statistic to store the number of blackholed traffic
    int number_of_blackholing;
    double time;
    List * prefixes;
}IpT_Rte_Blackhole_From;
/* End Code Snippet*/
```

**Code Snippet C-29 Definition of `IpT_Rte_Blackhole_From` Structure.**

This structure is included in another structure that is generally accessible by IP processes. The structure is `IpT_Rte_Module_Data` and the addition is shown in Code Snippet C-30.



```

/* Alrefai Code Snippet Start */
typedef struct IpT_Rte_Module_Data
{
    Objid                                module_id;
    Objid                                node_id;
    /* Omitted other definitions*/
    IpT_Rte_Blackhole_From*              blackhole_from_ptr;
    /* Added by Ahmad Salam Alrefai to inidicate time and
    prefixes to block*/
} IpT_Rte_Module_Data;
/* End Code Snippet*/

```

Code Snippet C-30 Adding IpT\_Rte\_Blackhole\_From to IpT\_Rte\_Module\_Data.

### **C.2.1.1 Function Block**

The function `ip_dispatch_init_phase_2` is modified to include local variables shown in Code Snippet C-31 and to read the configuration of More Specific Prefixes as shown in Code Snippet C-32.

```

/* Alrefai Code Snippet Start */
Objid                                malicious_blackholing_objid;
Objid                                first_blackholing_objid;
Objid                                prefixes_objid;
Objid                                ith_prefix_objid;
int                                  num_blackholing;
int                                  num_prefixes;
IpT_Rte_Blackhole_From*              blackhole_from_ptr;
double                               time;
List*                                prefixes;
char                                  addr_str[64];
InetT_Address                        ntwk_addr, masked_ntwk_addr;
InetT_Subnet_Mask                    inet_smask;
IpT_Address                          ipv4_smask;
int                                  count_i;
/* End Code Snippet*/

```

Code Snippet C-31 Local variables added to `ip_dispatch_init_phase_2`.

```

/* Alrefai Code Snippet Start */
/* read malicious blackholing information */
op_ima_obj_attr_get(module_data.ip_parameters_objid, "Malicious
Blackholing", &malicious_blackholing_objid);
num_blackholing = op_topo_child_count (malicious_blackholing_objid,
OPC_OBJTYPE_GENERIC);
if (num_blackholing > 0)
{
    first_blackholing_objid = op_topo_child (malicious_blackholing_objid,
OPC_OBJTYPE_GENERIC, 0);
    op_ima_obj_attr_get (first_blackholing_objid, "Time", &time);
    op_ima_obj_attr_get (first_blackholing_objid, "Prefixes",
&prefixes_objid);
    num_prefixes = op_topo_child_count (prefixes_objid, OPC_OBJTYPE_GENERIC);
    prefixes = op_prg_list_create ();
    for (count_i = 0; count_i < num_prefixes; count_i++)
    {
        ith_prefix_objid = op_topo_child (prefixes_objid,
OPC_OBJTYPE_GENERIC, count_i);
        op_ima_obj_attr_get (ith_prefix_objid, "IP Address", addr_str);
        ntwk_addr = inet_address_create (addr_str, InetC_Addr_Family_v4);
        if (!inet_address_valid (ntwk_addr)) {printf("network invalid"); continue;}
        op_ima_obj_attr_get (ith_prefix_objid, "Mask", addr_str);
        ipv4_smask = ip_address_create (addr_str);
        if (ip_address_equal (IPC_ADDR_INVALID,
ipv4_smask)) {printf("network invalid"); continue;}
        inet_smask = inet_smask_from_ipv4_smask_create (ipv4_smask);
        masked_ntwk_addr = inet_address_mask (ntwk_addr, inet_smask);
        inet_address_destroy (ntwk_addr);
        prefix_ptr = inet_address_range_mem_alloc ();
        *prefix_ptr = inet_address_range_create_fast (masked_ntwk_addr,
inet_smask);
        op_prg_list_insert (prefixes, prefix_ptr, OPC_LISTPOS_TAIL);
    }
    blackhole_from_ptr = op_prg_mem_alloc (sizeof
(IpT_Rte_Blackhole_From));
    blackhole_from_ptr->number_of_packets = 0;
    blackhole_from_ptr->number_of_blackholing = 0;
    blackhole_from_ptr->time = time;
    blackhole_from_ptr->prefixes = prefixes;
    module_data.blackhole_from_ptr = blackhole_from_ptr;
}
else
{
    module_data.blackhole_from_ptr = OPC_NIL;
}
/* End Code Snippet*/

```

**Code Snippet C-32 Modifying ip\_dispatch\_init\_phase\_2 to Read Values for More Specific Prefix Configuration.**

### **C.2.1.2 The ip\_rte\_central\_cpu Process**

In `ip_rte_central_cpu` the method `ip_rte_central_cpu_packet_arrival` is modified as shown in Code Snippet C-33 to define local variable and in Code Snippet C-34 to call `ip_rte_blackhole_traffic` function.

```
/* Alrefai Code Snippet Start */
Boolean                                is_blackholed;
/* End Code Snippet*/
```

**Code Snippet C-33 Adding Local Variable to `ip_rte_central_cpu_arrival` function.**

```
/* Alrefai Code Snippet Start */
result = ip_rte_packet_arrival (module_data_ptr,
                                &pkptr, instrm, &intf_ici_fdstruct_ptr, &rcvd_iface_info_ptr);
/*Alrefai start modification*/
is_blackholed = ip_rte_blackhole_traffic (module_data_ptr, pkptr);
if (result == OPC_FALSE || is_blackholed == OPC_TRUE)
/*@End of modification*/
{
    /* Packet was dropped in call */
    FOUT;
}
/* End Code Snippet*/
```

**Code Snippet C-34 Calling `ip_rte_blackhole_traffic` function from inside `ip_rte_central_cpu_packet_arrival`.**

### **C.2.1.3 Blackholing Definition and Implementation**

The method `ip_rte_blackhole_traffic` definition is shown in Code Snippet C-35. It is defined in the file `ip_rte_support.h`.

```
/* Alrefai Code Snippet Start */
/** Ahmad Salam Alrefai Method!! ***/
Boolean ip_rte_blackhole_traffic (IpT_Rte_Module_Data * iprmd_ptr,
Packet * pkptr);
/* End Code Snippet*/
```

**Code Snippet C-35 Definition of `ip_rte_blackhole_traffic`.**

In `ip_rte_support.ex.c` the function `ip_rte_blackhole_traffic` is implemented as shown in

```
/* Alrefai Code Snippet Start */
Boolean ip_rte_blackhole_traffic (IpT_Rte_Module_Data * iprmd_ptr, Packet *
pkptr)
{
    InetT_Address_Range *      blackholed_prefix;
    InetT_Address              dest_address;
    InetT_Address              src_address;
    int num_prefixes;
    IpT_Dgram_Fields* pk_fd_ptr;
    double time;
    List * prefixes;
    int count_i;
    FIN (ip_rte_blackhole_traffic (iprmd_ptr, pkptr));
    if (iprmd_ptr->blackhole_from_ptr == OPC_NIL)
    {
        FRET (OPC_FALSE);
    }
    iprmd_ptr->blackhole_from_ptr->number_of_packets++;
    time = iprmd_ptr->blackhole_from_ptr->time;
```

**Code Snippet C-36 The implementation of `ip_rte_blackhole_traffic_function`.**

```

/* Alrefai Code Snippet Continue...*/
if (op_sim_time() < time)
{
    FRET (OPC_FALSE);
}

prefixes = iprmd_ptr->blackhole_from_ptr->prefixes;
op_pk_nfd_access (pkptr, "fields", &pk_fd_ptr);
dest_address = pk_fd_ptr->dest_addr;
src_address = pk_fd_ptr->src_addr;
num_prefixes = op_prg_list_size(prefixes);
printf("number of packets: %d \n", iprmd_ptr->blackhole_from_ptr-
>number_of_packets);
for(count_i = 0; count_i < num_prefixes; count_i++)
{
    blackholed_prefix = (InetT_Address_Range *)op_prg_list_access
(prefixes, count_i);
    if (inet_address_range_check (dest_address, blackholed_prefix)
== PRGC_TRUE)
    {
        ip_rte_dgram_discard (iprmd_ptr, pkptr, op_pk_ici_get
(pkptr), "Discarded because destination address is blackholed:");
        iprmd_ptr->blackhole_from_ptr->number_of_blackholing++;

        printf("number of blackholing: %d \n", iprmd_ptr-
>blackhole_from_ptr->number_of_blackholing);
        FRET (OPC_TRUE);
    }
    else if (inet_address_range_check (src_address,
blackholed_prefix) == PRGC_TRUE)
    {
        ip_rte_dgram_discard (iprmd_ptr, pkptr, op_pk_ici_get
(pkptr), "Discarded because source address is blackholed:");
        iprmd_ptr->blackhole_from_ptr->number_of_blackholing++;
        printf("number of blackholing: %d \n", iprmd_ptr-
>blackhole_from_ptr->number_of_blackholing);
        FRET (OPC_TRUE);
    }
}
FRET (OPC_FALSE);
}
/* End Code Snippet*/

```

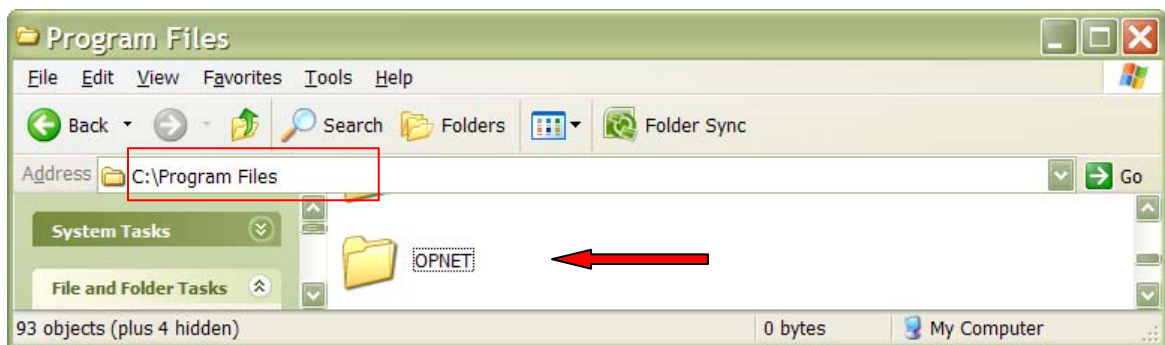
**Code Snippet C-37** The implementation of `ip_rte_blackhole_traffic_function`.

## ***Appendix D***

### ***MANUAL TO RUN THE WORK***

#### ***D.1 PROCEDURE***

1. Install OPNET version 14.5. By default OPNET is installed in “C:\Program Files” directory, and a folder called OPNET is created that contains all the files.



**Figure D-1 OPNET installation Folder**

2. Therefore, the default [OPNET DIRECTORY] is “C:\Program Files\OPNET”.
3. Go to [OPNET DIRECTORY]\ 14.5.A\models\std.
4. Keep a backup of the folders bgp, ip, include, as shown in

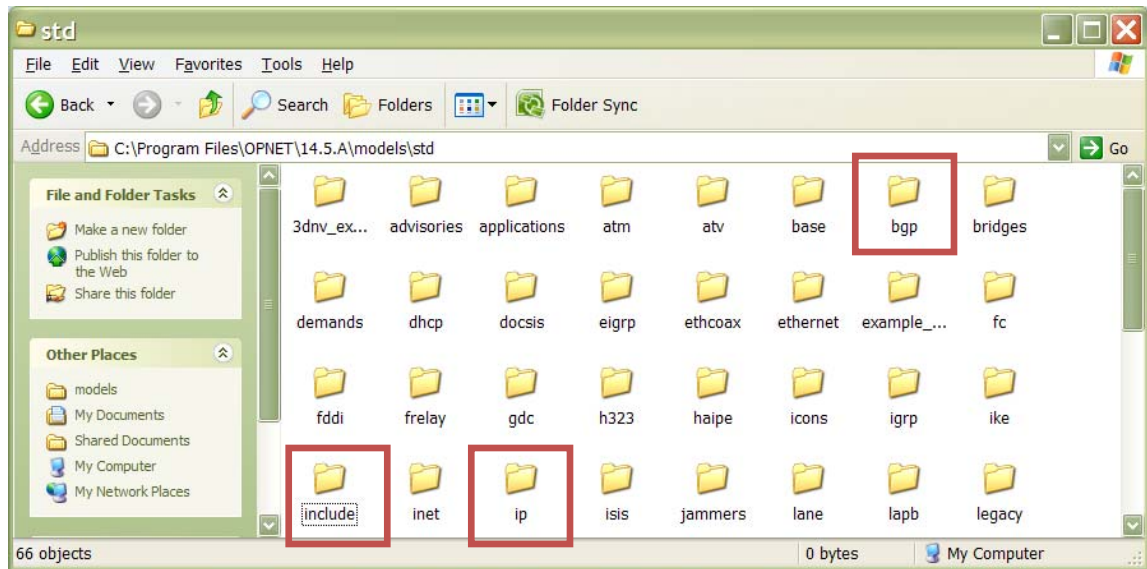


Figure D-2 Modified Folders

5. Replace those folders by our modified bgp, ip, and include folders.
6. The default [OPNET PROJECTS] is C:\Documents and Settings\[USER NAME]\op\_models as shown

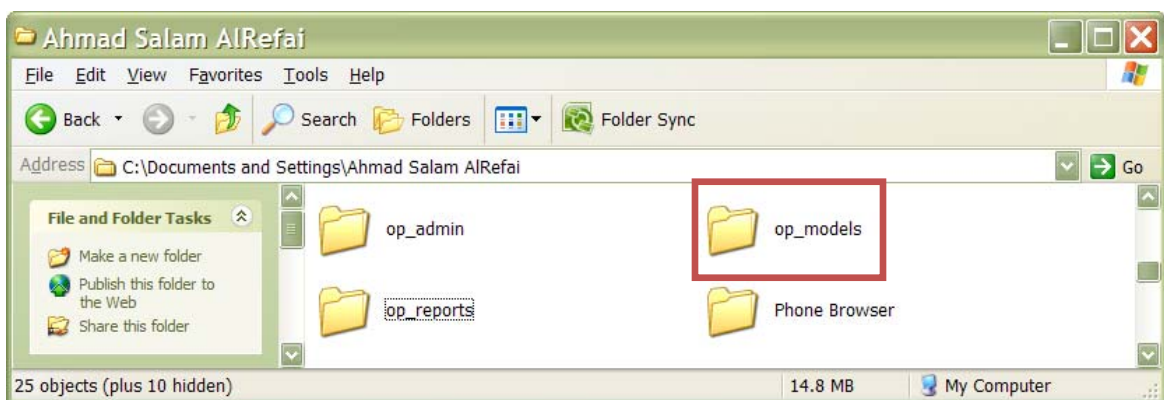


Figure D-3 op\_models folders

7. Put the project folder asr\_bgp\_experiments5.project in [OPNET PROJECTS]

8. Start OPNET Application and open asr\_BGP\_experiments5.prj inside the folder asr\_bgp\_experiments5.project.

Then you can run the simulation and generate any result you want.

Another method, instead of replacing the bgp, ip folders one can just put them in local directory, and then run any process from bgp and from ip, this causes OPNET to ask whether you want to overwrite the path and you should answer yes, use these modified bgp and ip instead and as a result causing OPNET to read the modified ip and bgp protocols. Include folder still needs to be overridden, however.



---

## REFERENCES

- [1] (2009, March) Internet World Stats. [Online].  
<http://www.internetworldstats.com/stats.htm>
- [2] Marwan Abu-Amara, Ashraf Mahmoud, Muhammad Sqalli, and Farag Azzedin, "Internet Access Denial by International Internet Service Providers: Analysis and Counter Measures," Research Proposal April 2008.
- [3] Xin Hu and Z. Morley Mao, "Accurate Real-time Identification of IP Prefix Hijacking," in *IEEE Symposium on Security and Privacy*, May 2007, pp. 20-23.
- [4] Bruno Quoitin, "BGP-based Interdomain Traffic Engineering," Departement d'Ingenierie Informatique, Universite catholique de Louvain, August 4, 2006.
- [5] (2006, April) Types of Internet Connections. [Online].  
[http://www.webopedia.com/quick\\_ref/internet\\_connection\\_types.asp](http://www.webopedia.com/quick_ref/internet_connection_types.asp)
- [6] Tao Wan, P.C. van Oorschot, and Evangelos Kranakis, "A Selective Introduction to Border Gateway Protocol (BGP) Security Issues," School of Computer Science, Carleton University, Ottawa, 2005.
- [7] Kevin Butler, Toni Farley, Patrick McDaniel, and Jennifer Rexford, "A Survey of BGP Security," 2005.
- [8] Y. Rekhter, T. Li, and S. Hares. (2006, January) IETF-A Border Gateway Protocol 4 (BGP-4). [Online]. <http://www.ietf.org/rfc/rfc4271.txt>
- [9] S. Murphy. (2006, January) IETF -BGP Security Vulnerabilities Analysis. [Online].  
<http://www.ietf.org/rfc/rfc4272.txt>
- [10] Y. Rekhter and T. Li. (1995, March) RFC 1771. [Online].  
<http://www.ietf.org/rfc/rfc1771.txt>
- [11] C. Villamizar, R. Chandra, and Govindan R. (1998, November) IETF. [Online].  
<http://www.ietf.org/rfc/rfc2439.txt>
- [12] Xiaomei Liu and Li Xiao, "A Survey of Multihoming Technology in Stub Networks: Current Research and Open Issues," , 2007.

- [13] Ola Nordstrom and Constantinos Dovrolis, "Beware of BGP Attacks," *ACM SIGCOMM Computer Communications Review*, vol. 34, no. 2, pp. 1-8, April 2004.
- [14] Ratul Mahajan, David Wetherall, and Tom Anderson, "Understanding BGP Misconfiguration," in *Proceedings of ACM Sigcomm*, August 2002, pp. 3-16.
- [15] Stephen A Misel. (1997, April) Merit Network Email List Archive. [Online]. <http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html>
- [16] James A. Farrar. (2001, April) Merit Network Email List Archives. [Online]. <http://www.merit.edu/mail.archives/nanog/2001-04/msg00209.html>
- [17] Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 582-592, April 2000.
- [18] Martin O. Nicholes and Biswanath Mukerjee, "A Survey of Security Techniques for the Border Gateway Protocol (BGP)," vol. 11, no. 1, First Quarter 2009.
- [19] Dionysus Blazakis, Manish Karir, and John S. Baras, "Analyzing BGP AS PATH Behavior in the Internet," in *roceedings of the 9th IEEE Global Internet Symposium*, April 2006.
- [20] Vladimir I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklad*, vol. 10, no. 8, pp. 707-710, 1966.
- [21] Xiaoliang Zhao et al., "An Analysis of BGP Multiple Origin AS (MOAS) Conflicts," in *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, November 2001.
- [22] J. Hawkinson and T. Bates. (1996, March) RFC 1930. [Online]. <http://tools.ietf.org/html/rfc1930>
- [23] Mohit Lad et al., "PHAS: A Prefix Hijack Alert System," in *the 15th conference on USENIX Security*, Vancouver, B.C., Canada , 2006.

- [24] University of Oregon. The Route Views Project. [Online]. <http://www.routeviews.org/>
- [25] Chanxi Zheng, Lusheng Ji, Dan Pei, Jia Wang, and Paul Francis, "A Light-Weight Distributed Scheme for Detecting IP prefix Hijacks in Real-Time," in *SIGCOMM'07*, Kyoto, August 2007.
- [26] Lixin Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 6, pp. 733 - 745, December 2001.
- [27] David G. Adnersen, "Resilient Overlay Networks," MASSACHUSETTS INSTITUTE OF TECHNOLOGY, May, Master Thesis 2001.
- [28] Sharad Agarwal, Chen-Nee Chuah, and Randy H. Katz, "OPCA: Robust Interdomain Policy Routing and Traffic Control," in *Open Architecture and Network Programming*, 2003.
- [29] Bruno Quoitin and Steve Uhlig, "Modeling the routing of an Autonomous System with C-BGP," vol. 19, no. 6, pp. 12-19, November 2005.
- [30] Bruno Quoitin and Olivier Bonaventure, "A cooperative approach to interdomain traffic," in *Proceedings of the 1st Conference on Next Generation Internet Networks Traffic Engineering*, Rome, Italy, 2005.
- [31] Modeling and Analysis of Network via Computer Simulation, Georgia Tech University. [Online]. <http://www.ece.gatech.edu/research/labs/MANIACS/BGP++/>
- [32] Maciej Wojciechowski, "Border Gateway Protocol Modeling and Simulation," VU University Amsterdam, Master's thesis July 2008.
- [33] Z. Morley Mao, Randy Bushy, Timothy G. Griffin, and Roughanx Matthew, "BGP Beacons," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2003.

- [34] Lan Wang et al., "Observation and Analysis of BGP Behavior under Stress," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, Marseille, France, 2002, pp. 183 - 195.
- [35] Timothy G. Griffin and Brian J. Premore, "An Experimental Analysis of BGP Convergence Time," in *Ninth International Conference on Network Protocols (ICNP)*, November 2001, pp. 53 - 61.
- [36] Craig Labovitz, G. Robert Malan, and Farnam Jahanian, "Internet routing instability," in *ACM SIGCOMM Computer Communication Review*, 1997, pp. 115 - 126.
- [37] Craig Labovitz, G. Robert Malan, and Farnam Jahanian, "Origins of Internet routing instability," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, New York, NY, USA, March 1999, pp. 218-226.
- [38] Bruno Quoitin, Cristel Pelsser, Louls Swinnen, Olivier Bonaventure, and Steve Uhlig, "Interdomain Traffic Engineering with BGP," vol. 41, no. 5, pp. 122-128, May 2003.
- [39] Rocky K. C. Chang and Michael Lo, "Inbound traffic engineering for multi-homed ASes using AS path prepending," in *Network Operations and Management Symposium*, vol. 1, 2004, pp. 98-102.
- [40] OPNET Application and Network Performance. [Online]. <http://www.opnet.com/>
- [41] (2008) OPNET Modeler.
- [42] Emad Aboelela, *Network Simulation Experiments Manual*, 4th ed.: Morgan Kaufmann, 2007, Network Experiments Manual for Peterson/Davie Computer Networks.
- [43] "OPNET Modeler Product Documentation," OPNET Technologies Inc., 2008.

---

## *VITA*

- Name: Ahmad Salam Alrefai
- Nationality: Syrian.
- Holder of BS degree in Computer Engineering from King Fahd University of Petroleum and Minerals [Dhahran-Saudi Arabia]. (2007)
- Holder of BS degree in Software Engineering from King Fahd University of Petroleum and Minerals [Dhahran-Saudi Arabia]. (2007)
- Completed MS degree requirements in Computer Engineering from King Fahd University of Petroleum and Minerals [Dhahran-Saudi Arabia]. (2009)
- Started PhD Program in Electrical and Computer Engineering at the University of Waterloo [Waterloo, Ontario, Canada]. (2010)
- Present Address: 268-350 COLUMBIA ST W – WATERLOO, ON – N2L 6P7 – CANADA
- Permanent Address in Saudi Arabia: P.O. Box: 295357 Riyadh 11351 Saudi Arabia
- Permanent Address in Syria  
منزل عبد اللطيف الرفاعي – طابق ٦، برج ١٢، جزيرة ٥ – مشروع دمر – دمشق – سوريا
- Email: [a.s.refai@gmail.com](mailto:a.s.refai@gmail.com)