ENSC 891: Directed Studies
Intergration of ns-BGP with ns-2.34
(ns-2.34-BGP)

Fall 2009

FINAL REPORT

Mohammad Reza Sahraei

mrs16@sfu.ca

Table of Contents

List of Figures

Index of Tables

1. ABSTRACT

The ns network simulator (popularly called ns-2, in reference to its current generation) is a discrete event simulator. It is popular in academia for its extensibility (due to its open source model) and plentiful online documentation. ns is widely used in the simulation of routing and multicast protocols, among others, and is heavily used in ad-hoc networking research. ns supports an array of popular network protocols, offering simulation results for wired and wireless networks alike. It can be also used as limited-functionality network emulator [1].

The Border Gateway Protocol (BGP) is an inter-Autonomous System routing protocol. Today, BGP is used as the core routing protocol of the Internet. It is built on experience achieved using Exterior Gateway Protocol (EGP) as defined in Request For Comments (RFC) 904 [2]. Border Gateway Protocol 4 (BGP-4) proposed in RFC 1771 by Y. Rekhter and T. Li from Network Working Group within the Internet Engineering Task Force (IETF). The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information of the list of Autonomous Systems (ASs) that reachability information traverses. This information is sufficient for constructing a graph of AS connectivity for this reachability, from which routing loops may be pruned and, at the AS level, some policy decisions may be enforced [3].

The BGP performance is changing due to the dynamic nature of the internet. Therefore, the research community needed to study the protocol in a network simulator. As the result, BGP was integrated into ns-2.27 in 2004 and was called ns-BGP [4].

ns-2 continued to develop and the new features added to the build apart from the ns-BGP add on. In order to make ns-BGP compatible with the latest build, a newer version of ns-BGP for ns-2.33 was implemented in 2008 [5]. This is the latest implementation of ns-BGP for ns-2.34.

2. INTRODUCTION

The Border Gateway Protocol, BGP, is a de facto inter-Autonomous Systems (ASs) routing protocol. The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information on the list of Autonomous Systems that reachability information traverses. This information is sufficient for constructing a graph of AS connectivity for this reachability, from which routing loops may be pruned and, at the AS level, some policy decisions may be enforced [6].

Since the Internet has a very dynamic nature and this has an effect on the performance of the routing protocols such as BGP. Normally, every five years or so, the research community performs a complete analysis on the protocol. Because the empirical data is hard to access and theoretical analysis lacks the accuracy to reflect the complete picture, simulation has become very attractive and practical for the academia and research communities. On the other hand, the Network Simulator better known as ns with an open source software policy is widely used by researchers all around the world. Therefore, the above reasons were a good motivation to implement the BGP protocol for ns-2. This happened by importing the code from BGP implementation in SSFNET and converting them to C++ and OTcl code in 2004 [4]. Later, the ns-BGP upgraded to be compatible with the latest version of ns, which was ns-2.33 in 2008 [5]. This project implements ns-BGP for the latest stable ns release ns-2.34. Figure 1 shows the ns-BGP progress and timeline.

In order to differentiate the ns-BGP implementation for different ns-2 release, any referral in this paper includes the ns release number in ns-BGP name (e.g. ns-BGP implementation in ns-2.34 is referred to as ns-2.34-BGP.

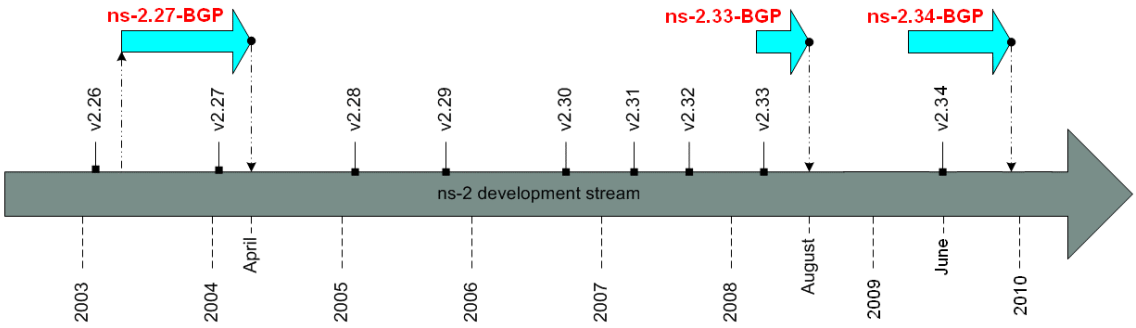

Figure 1: ns-BGP Progress and Timeline.

The ns-2.34-BGP project has these phases:

- Technology learning phase:

  Before attempting the project, I had to review the technologies used in the ns-BGP. Also, it was required to understand how BGP protocol worked. In this phase, I studied and reviewed these technology fundamentals and concepts:
  - Tcl/Tk [7]
  - ns-2.33 and ns-2.34 network simulator [8, 9]
  - BGP-4 [3]

- Integration environment setup phase:

  ns-2 runs under Linux, Mac OS X, and Windows via Cygwin. I chose Linux as the integration environment for the project and used two different flavors of Linux: Xubuntu 9.4 and Xandros 3.0.2. I had to install gcc C compiler to be able to compile and run the ns distributions. I installed ns-2.33 and ns-2.34 on two separate computers. There are necessary changes to environment parameters in .bashrc file. The changes setup path for various components used in the distribution: OTcl, nam, etc. I discovered and used a required patch for ns-2.33 distribution to work under Xubuntu 9.4 [10].

- Code integration phase:

  This phase consists of two sub-phases: The first sub-phase integrates the ns-BGP files into the ns files system tree structure. These files do not have any overlap with the ns distribution files. The second sub-phase required incorporating or changing ns code to be compatible with ns-BGP. This sub-phase needed extra attention to the logic of the overlapping codes implemented in ns-2.33-BGP in order to retain the functionality of the classes and procedures in ns-2.34.BGP in tact.

- Code compilation phase:

  Integration of ns-2.33-BGP code into ns-2.34 was not sufficient to generate an error free compile result. Some of the C syntax and library calls had to be modified to resolve the error reported by gcc C compiler.

- Project verification and validation phase:

  This phase includes verification and validation of the code to ensure the functionality of the ns-2.34-BGP build is identical to ns-2.33-BGP. There are some standard test cases defined in ns-2.27-BGP under: ~/ns-allinone-2.27/ns-2.27/tcl/bgp/test directory. I used these test cases in ns-2.34-BGP and compared the result of the outputs and trace output (.nam) with ns-2.33-BGP.

- Project deliverables phase:

  This phase includes the tasks related to preparation of the report, presentation slides, and consolidation of ns-2.34-BGP code release.

2.1.ns Overview

The ns network simulator (popularly called ns-2, in reference to its current generation) is a discrete event simulator. It is popular in academia for its extensibility (due to its open source model) and plentiful online documentation. ns is popularly used in the simulation of routing and multicast protocols, among others, and is heavily used in ad-hoc networking research. ns supports an array of popular network protocols, offering simulation results for wired and wireless networks alike. It can be also used as limited-functionality network emulator [1].

Development of ns began in 1989 as a variant of the REAL network simulator. By 1995, ns had gained support from DARPA (Defense Advanced Research Projects Agency), the VINT (Virtual Inter Network Test-bed) project at LBNL (Ernest Orlando Lawrence Berkeley National Laboratory), Xerox PARC (Palo Alto Research Center, Inc.), UCB (University of California, Berkeley), and USC/ISI (University of Southern California / Information Sciences Institute) [1].

ns-2 is implemented in C++ and OTcl, which is an object oriented variant of Tcl. The user benefits from OTcl scripts to describe the network and to configure the system. C++ is used to implement codes that are executed frequently. C++ is also preferable when the efficiency and performance has higher priority over the simplicity and flexibility in writing the code.

ns-2 supports various routing algorithms and transport protocols such as UDP, TCP, and SCTP. It generates traffic in forms of Constant Bit Rate (CBR), Exponential and Pareto distributions, and using trace files. The simulation process can be visually traced by a companion package called network simulator or nam. ns-2 supports Linux, Mac OS, and Windows via Cygwin. The latest stable version is ns-2.34, which was released in June 2009.

ns-3 is the third generation of ns that has begun development as of July 1, 2006 and is projected to take four years. It is funded by the institutes like University of Washington, Georgia Institute of Technology and the ICSI Center for Internet Research with collaborative support from the Planète research group at INRIA Sophia-Antipolis. Currently ns-3 is in development phase. It is an event based network simulator [1].

2.2.BGP Overview

Autonomous System (AS) is a collection of routers under a single technical administrator. The routing protocol inside of the AS is called Interior Gateway Protocol (IGP). Routing Information Protocol (RIP), Interior Gateway Routing Protocol (IGRP), Enhanced Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First (OSPF), and Intermediate system to intermediate system (IS-IS) are examples of IGP protocol.

The Border Gateway Protocol (BGP) is an inter-Autonomous System routing protocol. Today, BGP is used as the core routing protocol of the Internet. It is primarily used to exchange network layer reachability information (NLRI) between ASs. It is built on experience achieved using Exterior Gateway Protocol (EGP) as defined in Request For Comments (RFC) 904 [2]. Border Gateway Protocol 4 (BGP-4) proposed in RFC 1771 by Y. Rekhter and T. Li from Network Working Group within the Internet Engineering Task Force (IETF). The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information of the list of Autonomous Systems that reachability information traverses. This information is sufficient for constructing a graph of AS connectivity for this reachability, from which routing loops may be pruned and, at the AS level, some policy decisions may be enforced [3]. BGP requires to run on a reliable layer, therefore, it uses TCP protocol to send and receive its messages. BGP uses TCP port 179. Figure 2 shows an example of network architecture routing protocol of four ASs.

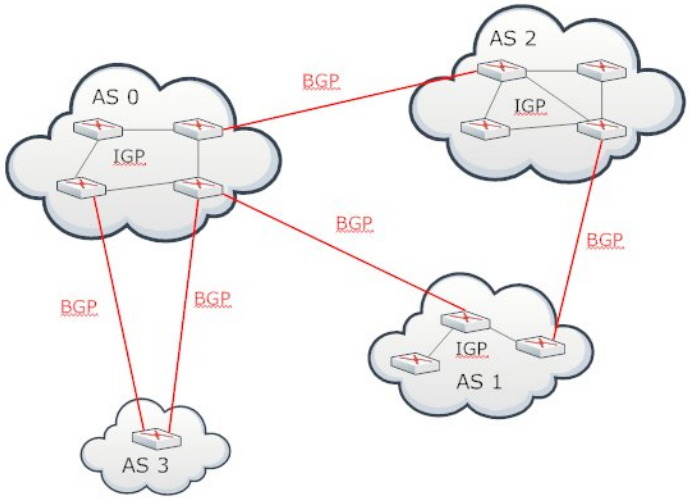

Figure 2: Network Architecture Routing Protocol.

The BGP protocol uses the following messages:

- OPEN
- UPDATE
- KEEPALIVE
- NOTIFICATION

The router that advertises to its peer on another AS is called BGP speaker. Two speakers first establish a TCP connection on port 179. Then, they exchange messages to open and confirm BGP connection parameters using OPEN message. The entire routing table is sent to the peer using UPDATE message. The speakers send any incremental change to the routing table via UPDATE message. The KEEPALIVE message is periodically sent to ensure the BGP connection is still active. A NOTIFICATION message is sent when an error condition in a message is detected or the hold timer expired. After sending the NOTIFICATION message, the speaker gracefully closes the TCP connection and remove the path to the speaker the error is detected from the routing table, and if necessary, advertise the new change to the other peers. Figure 3 shows an example of exchanging BGP messages between two speakers of different ASs. BGP speaker of AS1 fails to send a KEEPALIVE message within the Hold Time.

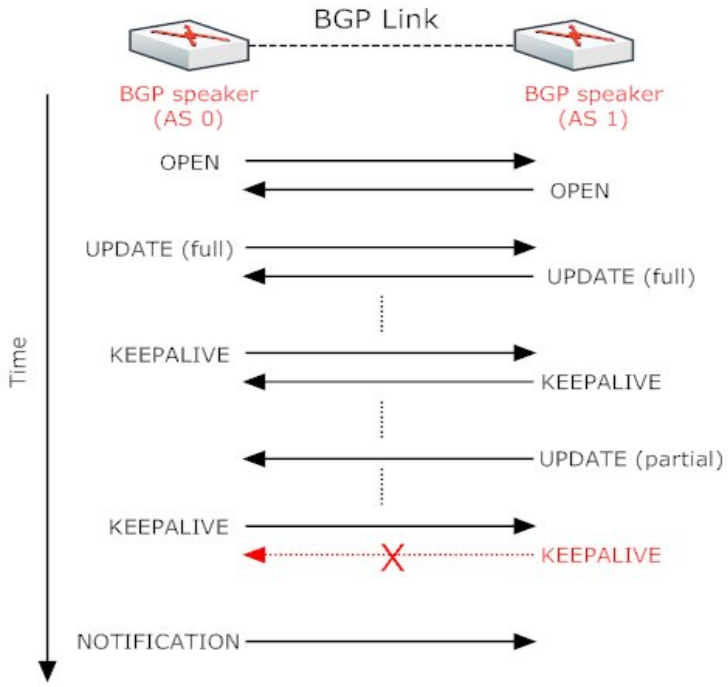

Figure 3: BGP Message Exchange.

2.3.ns-BGP Overview

ns-BGP for the first time was ported and integrated to ns-2 (ns-2.27-BGP) in 2004 [4]. This effort involved the port of the BGP and TcpSocket modules from SSFNet [11]. SSFNet is a Java-based network simulator which is comprised of a simulation engine and a configuration language known as the Domain Modeling Language (DML). Since SSFNet was implemented using object oriented technology, its BGP module was a natural candidate for ns-2 integration. Additionally, IPv4 addressing and packet forwarding was also incorporated into ns-2 to accommodate the SSFNet BGP dependencies [5].

Within ns-2, unicast routing is achieved using forwarding and control planes where the forwarding plane classifies and forwards packets to their destinations nodes using classifier and routing modules while the control plane provisions the route creation, computation, routing algorithms, and management of the routing tables. Figure 4 illustrates the ns-BGP unicast structure which is based the native ns-2 unicast architecture [5].

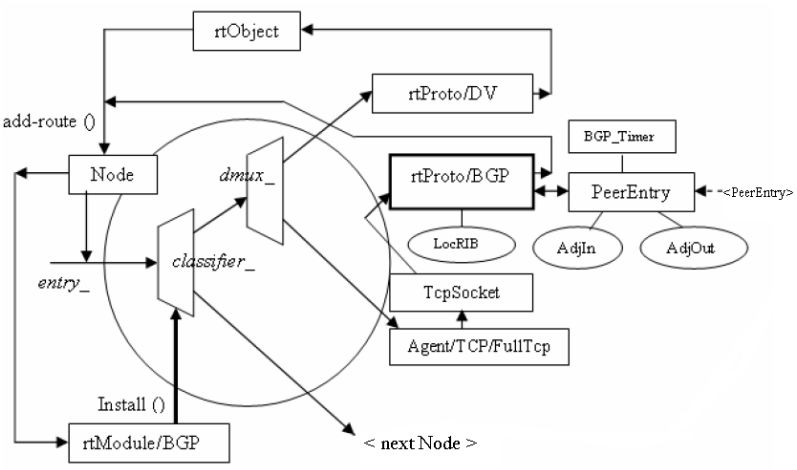

Figure 4: Unicast structure of ns-BGP.

In ns-2.27-BGP, *TcpSocket* has been modified to support socket and at the same time maintain the structure of SSFNet BGP. ns-2 address classifier has been replaced with *IPv4Classifier* to support IPv4 addressing and packet forwarding. *FullTcpAgent*, which is the TCP agent for *TcpSocket*, is modified to support user data transmission.

The *TcpSocket* class implements a UNIX-like socket application programming interface (API) which includes standard calls such as bind(), connect(), listen(), close(), read(), and write(). Additionally, it also provisions data queuing and callback functions [5].

*classifier_* in Figure 4 is an *IPv4Classifier*. A new routing module *rtModule/BGP* manages the *IPv4Classifier* and is a replacement of the basic routing module *rtModule/Base*. *TcpSocket* has been added to the modified *FullTcpAgent*, encapsulating the TCP services into a socket interface. A new routing protocol *rtProtoBGP* relies only on *TcpSocket* for packet transmission. *rtProto/BGP* has one *PeerEntry* for each peer. *PeerEntry* establishes and closes a peer session and exchanges BGP messages with a peer. Each instance of *PeerEntry* contains one *AdjIn*, one *AdjOut*, and a variable *BGP_Timer*. *LocRIB*, *AdjIn*, and *AdjOut* correspond to the three parts of the BGP Routing Information Base (RIB): Loc-RIB, Adj-RIBs-In, and Adj-RIBs-Out. *BGP_Timer* provides support for the BGP timing features [4].

3. Hardware Platform

In this project, I used two computers with the following specifications:

- Primary hardware:
  - Toshiba Satellite
  - Intel® Pentium® 4 CPU 2.4 GHz / 1 GB RAM
  - Linux Xandros 3.0.2

- Additional hardware:
  - Dell Inspiron
  - Pentium® Dual-Core CPU T4200 2.0 GHz / 3 GB RAM
  - Linux Xubuntu 9.4

Originally the project started on Toshiba Satellite with Linux Xubuntu 9.4. However, approximately half way through the project I added a new Dell computer and installed Xubuntu 9.4 on it. Thereafter, I installed Linux Xandros 3.0.2. I used the Thoshiba computer to run and to test ns-2.33-BGP and also used the Dell computer to implement and verify ns-2.34-BGP.

4. ns-2.33-BGP Analysis

ns-2.33-BGP files and directories are archived into a .tgz tar ball. For a complete list of the files and directories refer to Appendix 11.2. Figure 5 illustrates the files, which organized into a series of directories.

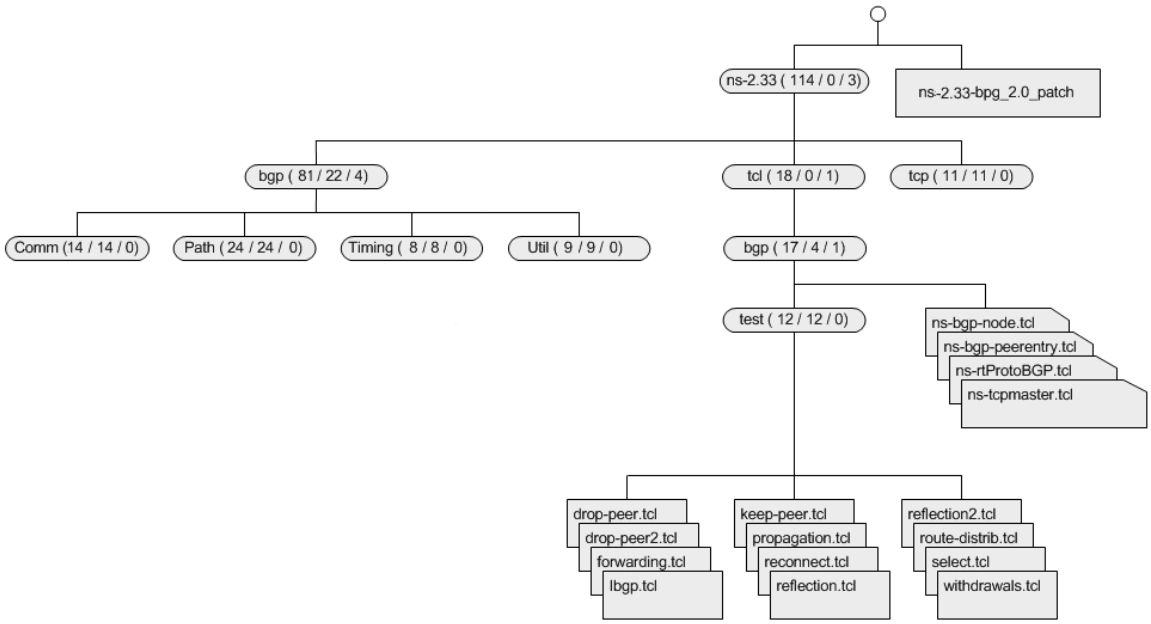

Figure 5: ns-2.33-BGP file and directory structure.

The notation in the figure 5 is as:

- The ellipses represent directories. The number (X/Y/Z) in each ellipse shows:
  - X – number of the files and directories in this directory and its sub-directories.
  - Y – number of the files in current directory only.
  - Z – number of sub-directories in this directory only.
- The angled rectangles represent core ns-2.33-BGP Tcl scripts.
- The rectangles represent ns-2.33-BGP test cases scripts.

The total number of files and directories are shown in the figure 5. Overall, there are 104 source files in the tar ball including: 41 C++ files (.cc), 46 header files (.h), 16 tcl files, and one patch file.

The first step to install ns-2.33-BGP is to copy and extract the tar ball into the ns-2 top level, which makes the following modifications:

- Creates a patch file in the top level directory
- Adds new files for TcpSocket under /tcp
- Creates a bgp sub-directory under /tcl
- Creates a test sub-directory under /bgp with all the test scripts files

The second step is to apply the patch file in order to update related ns files. The last step is to recompile ns files.

5. Integration

The objective of the project is to integrate ns-BGP to the current release of the simulator: ns-2.34. In addition, all of subsequent enhancement, development, and bug fixes that have taken place, since ns-2.33-BGP was released, needed to be retained. This phase consists of two sub-phases:

The first sub-phase integrates the ns-BGP files into the ns files system tree structure. These files do not have any overlap with the ns distribution files. These files are consolidated into a tar ball. These files include the C codes and the tcl scripts that required to be copied under ns-2.34 sub-directory. Overall, there are 104 source files including: 41 C++ files (.cc), 46 header files (.h), 16 tcl files, and one patch file gets copied under the ns tree structure. There is a Readme.txt that gets extracted into /bgp sub-directory.

The second sub-phase required incorporating or changing ns code to be compatible with ns-BGP. This sub-phase needed extra attention to the logic of the overlapping codes implemented in ns-2.33-BGP in order to retain the functionality of the classes and the procedures in ns-2.34.BGP in tact. The extracted patch file, ns-2.34-bgp_2.0_patch, contains modification of 37 files. The 16 files are merged with the logic of ns-BGP, which are listed in Table 1. The Merging Complexity column indicates the amount of work spent on each file.

	Files	Merging Complexity
1	ns-2.34/common/node.cc	Low
2	ns-2.34/common/node.h	Low
3	ns-2.34/common/packet.h	Moderate
4	ns-2.34/common/simulator.cc	Moderate
5	ns-2.34/Makefile.in	Low
6	ns-2.34/routing/route.cc	Low
7	ns-2.34/routing/rtmodule.cc	Low
8	ns-2.34/routing/rtmodule.h	Low
9	ns-2.34/tcl/lib/ns-default.tcl	Low
10	ns-2.34/tcl/lib/ns-lib.tcl	Low
11	ns-2.34/tcl/lib/ns-node.tcl	Moderate
12	ns-2.34/tcp/rq.cc	Low
13	ns-2.34/tcp/rq.h	Low
14	ns-2.34/tcp/scoreboard-rq.cc	Low
15	ns-2.34/tcp/tcp-full.cc	Moderate
16	ns-2.34/tcp/tcp-full.h	Moderate

Table 1: Complexity of the Merging file in the Patch.

The 21 files in the patch file are related to the logic of BGP. They are modified because they caused a compile error. Table2 lists the files and amount of work spent on each file.

	Files	Compilation Complexity
1	ns-2.33/tcp/tcp_master.h	Low
2	ns-2.33/tcp/tcp_socket.h	Low
3	ns-2.33/tcp/receive_queue.cc	Low
4	ns-2.33/tcp/send_queue.cc	Moderate
5	ns-2.33/bgp/Util/ipaddress.h	Low
6	ns-2.33/bgp/Comm/bgpmessage.h	Low
7	ns-2.33/bgp/route.h	Low
8	ns-2.33/bgp/Path/segment.h	Low
9	ns-2.33/bgp/Path/aspath.h	Low
10	ns-2.33/bgp/Timing/bgp_timer.h	Low
11	ns-2.33/bgp/rtProtoBGP.h	Low
12	ns-2.33/bgp/routeInfo.h	Low
13	ns-2.33/bgp/peer-entry.h	Low
14	ns-2.33/bgp/Timing/mraiperpeertimer.h	Low
15	ns-2.33/bgp/Path/attribute.cc	Low
16	ns-2.33/bgp/Timing/mraitimer.cc	Low
17	ns-2.33/bgp/Path/classifier-ipv4src.h	Low
18	ns-2.33/bgp/Util/ipaddress.cc	Low
19	ns-2.33/bgp/Util/stringmanip.cc	Low
20	ns-2.33/bgp/Path/aggregator.cc	Low
21	ns-2.33/bgp/Path/segment.h	Low

Table 2: Complexity of fixing the files in the Patch that caused error.

6. Validation

In order to validate and verify the functionality of ns-2.34-BGP to ensure the behaviour is identical to the previous version, ns-2.33-BGP, significant amount of work spent on the project. This was absolutely critical to make sure the tedious job of code integration was correct and the result of the two ns-BGP are the same. There are 12 Tcl scripts test cases under ~/ns-allinone-2.34/ns-2.34/tcl/bgp/test directory that examine the basic functions of BGP. Table 3 lists these scripts along with the result of the execution of each for ns-2.33-BGP and ns-2.34-BGP. I noticed that reflection2.tcl causes core dump for both ns-2.33-BGP and ns-2.34-BGP.

	Test Scripts	Execution Result		Comment
		ns-2.33-BGP	ns-2.34-BGP	
1	drop-peer.tcl	Passed	Passed	
2	drop-peer2.tcl	Passed	Passed	
3	forwarding.tcl	Passed	Passed	
4	ibgp.tcl	Passed	Passed	
5	keep-peer.tcl	Passed	Passed	
6	propagation.tcl	Passed	Passed	
7	reconnect.tcl	Passed	Passed	
8	reflection.tcl	Passed	Passed	
9	reflection2.tcl	**Failed**	**Failed**	The script generates core dump
10	route-distrib.tcl	Passed	Passed	
11	select.tcl	Passed	Passed	
12	withdrawals.tcl	Passed	Passed	

Table 3: ns-BGP Execution Results.

Here is the output and the core dump on screen.

REFLECTION2 VALIDATION TEST:

Three ASes(AS0, AS1 and AS2) connected in a line, the middle one(AS0) containing eight BGP routers, the others just one each. AS0 has two clusters: cluster 1000 and 2000. Cluster 1000 has two reflectors: n0 and n1. n2, n3 and n4 are reflection clients of both n0 and n1. Cluster 2000 contains one reflector n5, which has n6 and n7 as its reflection clients.

```
                AS 1        AS 0        AS 2
                 n8 }------{ n0-7 }------{ n9
Simulation starts ...

 time: 0.23
 cbr0 starts to send UDP segments to n10.
classifier _o36
        0 offset
        0 shift
        2147483647 mask
        0 slots


--- Classfier::no-slot{} default handler (tcl/lib/ns-lib.tcl) ---
```

```
        _o36: no target for slot -1
        _o36 type: Classifier/IPv4
content dump:
---------- Finished standard no-slot{} default handler ----------
```

The test script files generate output on standard screen and in a nam file. Table 4 shows the results of comparison of standard output and nam files for both ns-2.33-BGP and ns-2.34-BGP. The result of both comparisons is identical.

	Test Scripts	ns-2.33-BGP		ns-2.34-BGP		Contents Comparison Results	
		Output Filename	Output File Size	Output Filename	Output File Size	File	Standard Output
1	drop-peer.tcl	drop-peer.nam	18865	drop-peer.nam	18865	Passed	Passed
2	drop-peer2.tcl	drop-peer2.nam	36845	drop-peer2.nam	36845	Passed	Passed
3	forwarding.tcl	forwarding.nam	3181463	forwarding.nam	3181463	Passed	Passed
		forwarding.out	1476510	forwarding.out	1476510	Passed	Passed
4	ibgp.tcl	ibgp.nam	24278	ibgp.nam	24278	Passed	Passed
5	keep-peer.tcl	keep-peer.nam	23662	keep-peer.nam	23662	Passed	Passed
6	propagation.tcl	propagation.nam	19579	propagation.nam	19579	Passed	Passed
7	reconnect.tcl	reconnect.nam	39401	reconnect.nam	39401	Passed	Passed
8	reflection.tcl	reflection.nam	86745	reflection.nam	86745	Passed	Passed
9	reflection2.tcl	reflection2.nam	84545	reflection2.nam	84545	Passed	Passed
10	route-distrib.tcl	route-distrib.nam	11550	route-distrib.nam	11550	Passed	Passed
11	select.tcl	select.nam	31025	select.nam	31025	Passed	Passed
12	withdrawals.tcl	withdrawals.nam	14910	withdrawals.nam	14910	Passed	Passed

Table 4: ns-BGP Output Comparison.

7. Related Work

Many different BGP capable simulators exist in the community. Some them are commercial such as OPNET while others are implemented by academic and research communities and follow an open source policy standard. SSFNet (Scalable Simulation Framework Net) is a public-domain standard for discrete-event simulation of large, complex systems in Java and C++ [11]. SSFNet supports BGP; in fact ns-BGP ported the BGP logic from SSFNet. C-BGP is a dedicated BGP solver rather than simulator [12]. GNU Zebra BGP daemon was integrated into ns-2 almost at the same time ns-BGP was initially developed [13]. BGP++ [14] is a BGP module for ns-2 and GTNetS network simulators. It is actually a port of the Zebra BGP daemon and adapted to a C++ environment [5].

8. Future Work

The future work can be categorized in the following notes:

- Address the problem making reflection2.tcl causing core dump
- Incorporate the ns-2.34-BGP code in ns-2 distribution
- Add route flap damping
- Add adaptive minimal route advertisement interval (MRAI)
- Add policy routing
- Incorporate the ns-BGP code in ns-3 distribution

9. Conclusion

I have reached the objective of the project to integrate ns-BGP to the current release of the simulator: ns-2.34 conditioned on all of subsequent enhancement, development, and bug fixes that have taken place in ns build be retained. However, I discovered that one of twelve script tests, reflection2.tcl, causes core dump in both ns-2.33-BGP and ns-2.34-BGP. In order to proceed with the described future work, the first vital step would be to address the logical problem causing the core dump.

10. APPENDICES

10.1. Acronyms

API	Application Programming Interface
AS	Autonomous System
BGP	Border Gateway Protocol
DARPA	Defense Advanced Research Projects Agency
DML	Domain Modeling Language
EGP	Exterior Gateway Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
FSM	Finite State Machine
IETF	Internet Engineering Task Force
IBGP	Interior Border Gateway Protocol
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Providers
ns-2	Network Simulator 2
ns-3	Network Simulator 3
ns-BGP	Network Simulator-Border Gateway Protocol
NLRI	Network Layer Reachability Information
OSPF	Open Shortest Path First
OTcl	Object Tool Command Language
RFC	Request for Comments
RIP	Routing Information Protocol
RTP	Real Time Protocol
SSFNet	Scalable Simulation Framework Network
SSFNet.OS.BGP4	SSFNet's BGP model
STL	Standard Template Library
Tcl	Tool Command Language
TCP	Transmission Control Protocol
Tk	Tool kit
UDP	User Datagram Protocol
VINT	Virtual Internetwork Testbed

10.2. ns-2.33-BGP files

The initial ns-2.33-BGP release archive contents are:

ns-2.33/
ns-2.33/bgp/
ns-2.33/bgp/classifier-ipv4src.cc
ns-2.33/bgp/Readme.txt
ns-2.33/bgp/peer-entry.cc
ns-2.33/bgp/routeinfo.cc
ns-2.33/bgp/ribelement.cc
ns-2.33/bgp/rtProtoBGP.cc
ns-2.33/bgp/Util/
ns-2.33/bgp/Util/ipaddress.h
ns-2.33/bgp/Util/bitstring.cc
ns-2.33/bgp/Util/stringmanip.h
ns-2.33/bgp/Util/stringmanip.cc
ns-2.33/bgp/Util/radixtree.h
ns-2.33/bgp/Util/radixtreenode.h
ns-2.33/bgp/Util/bit.h
ns-2.33/bgp/Util/bitstring.h
ns-2.33/bgp/Util/ipaddress.cc
ns-2.33/bgp/locrib.cc
ns-2.33/bgp/rtProtoBGP.h
ns-2.33/bgp/routeinfo.h
ns-2.33/bgp/adjribin.h
ns-2.33/bgp/classifier-ipv4.cc
ns-2.33/bgp/locrib.h
ns-2.33/bgp/route.cc
ns-2.33/bgp/adjribout.h
ns-2.33/bgp/Comm/
ns-2.33/bgp/Comm/bgpmessage.h
ns-2.33/bgp/Comm/startstopmessage.h
ns-2.33/bgp/Comm/transportmessage.h
ns-2.33/bgp/Comm/openmessage.h
ns-2.33/bgp/Comm/transportmessage.cc
ns-2.33/bgp/Comm/openmessage.cc
ns-2.33/bgp/Comm/notificationmessage.h
ns-2.33/bgp/Comm/updatemessage.h
ns-2.33/bgp/Comm/bgpmessage.cc
ns-2.33/bgp/Comm/updatemessage.cc
ns-2.33/bgp/Comm/startstopmessage.cc
ns-2.33/bgp/Comm/keepalivemessage.cc
ns-2.33/bgp/Comm/keepalivemessage.h
ns-2.33/bgp/Comm/notificationmessage.cc
ns-2.33/bgp/global.h
ns-2.33/bgp/peer-entry.h
ns-2.33/bgp/route.h
ns-2.33/bgp/adjribout.cc
ns-2.33/bgp/adjribin.cc

ns-2.33/bgp/Timing/
ns-2.33/bgp/Timing/bgp_timer.h
ns-2.33/bgp/Timing/mraitimer.cc
ns-2.33/bgp/Timing/bgp_timer.cc
ns-2.33/bgp/Timing/mraitimer.h
ns-2.33/bgp/Timing/timeoutmessage.cc
ns-2.33/bgp/Timing/timeoutmessage.h
ns-2.33/bgp/Timing/mraiperpeertimer.cc
ns-2.33/bgp/Timing/mraiperpeertimer.h
ns-2.33/bgp/ribelement.h
ns-2.33/bgp/Path/
ns-2.33/bgp/Path/localpref.h
ns-2.33/bgp/Path/originatorid.h
ns-2.33/bgp/Path/atomicaggregate.cc
ns-2.33/bgp/Path/aggregator.cc
ns-2.33/bgp/Path/attribute.h
ns-2.33/bgp/Path/attribute.cc
ns-2.33/bgp/Path/aspath.h
ns-2.33/bgp/Path/clusterlist.cc
ns-2.33/bgp/Path/origin.h
ns-2.33/bgp/Path/clusterlist.h
ns-2.33/bgp/Path/segment.cc
ns-2.33/bgp/Path/med.h
ns-2.33/bgp/Path/origin.cc
ns-2.33/bgp/Path/nexthop.cc
ns-2.33/bgp/Path/nexthop.h
ns-2.33/bgp/Path/aggregator.h
ns-2.33/bgp/Path/community.h
ns-2.33/bgp/Path/atomicaggregate.h
ns-2.33/bgp/Path/segment.h
ns-2.33/bgp/Path/med.cc
ns-2.33/bgp/Path/localpref.cc
ns-2.33/bgp/Path/originatorid.cc
ns-2.33/bgp/Path/community.cc
ns-2.33/bgp/Path/aspath.cc
ns-2.33/bgp/classifier-ipv4.h
ns-2.33/bgp/classifier-ipv4src.h
ns-2.33/tcl/
ns-2.33/tcl/bgp/
ns-2.33/tcl/bgp/ns-bgp-node.tcl
ns-2.33/tcl/bgp/ns-rtProtoBGP.tcl
ns-2.33/tcl/bgp/ns-bgp-peerentry.tcl
ns-2.33/tcl/bgp/test/
ns-2.33/tcl/bgp/test/reconnect.tcl
ns-2.33/tcl/bgp/test/keep-peer.tcl
ns-2.33/tcl/bgp/test/forwarding.tcl

ns-2.33/tcl/bgp/test/withdrawals.tcl
ns-2.33/tcl/bgp/test/ibgp.tcl
ns-2.33/tcl/bgp/test/route-distrib.tcl
ns-2.33/tcl/bgp/test/drop-peer2.tcl
ns-2.33/tcl/bgp/test/select.tcl
ns-2.33/tcl/bgp/test/drop-peer.tcl
ns-2.33/tcl/bgp/test/reflection.tcl
ns-2.33/tcl/bgp/test/propagation.tcl
ns-2.33/tcl/bgp/test/reflection2.tcl
ns-2.33/tcl/bgp/ns-tcpmaster.tcl
ns-2.33/tcp/
ns-2.33/tcp/receive_queue.cc
ns-2.33/tcp/send_queue.cc
ns-2.33/tcp/tcp_socket.cc
ns-2.33/tcp/tcp_data.cc
ns-2.33/tcp/tcp_data.h
ns-2.33/tcp/tcp_master.h
ns-2.33/tcp/continuation.h
ns-2.33/tcp/receive_queue.h
ns-2.33/tcp/tcp_master.cc
ns-2.33/tcp/send_queue.h
ns-2.33/tcp/tcp_socket.h
ns-2.33-bgp_2.0_patch

10.3. ns-2.34-BGP patch file

This Section contains the details of the changes in the ns-2.34-BGP patch file that is required to alter ns-2.34 codes to make it compatible with ns-BGP.

```
diff -rc ns-2.34_original/common/node.cc ns-2.34/common/node.cc
*** ns-2.34_original/common/node.cc          2009-06-14 10:35:45.000000000 -0700
--- ns-2.34/common/node.cc            2009-10-12 18:30:52.000000000 -0700
***************
*** 139,144 ****
--- 139,151 ----
  Node::command(int argc, const char*const* argv)
  {
            Tcl& tcl = Tcl::instance();
+         // Merged by Will Hrudey, Reza Sahraei
+         // Modified by Tony Feng for BGP.
+         if (strcmp(argv[1], "as") == 0) {
+                                   as_num_ = atoi(argv[2]);
+                        return TCL_OK;
+         }
+         // End Tony Feng
            if (argc == 2) {
  #ifdef HAVE_STL
                        // Mods for Nix-Vector Routing
***************
*** 214,220 ****
```

```
                                }
                                addNeighbor(node);
                                return TCL_OK;
!                       }
                }
                return ParentNode::command(argc,argv);
  }
--- 221,237 ----
                                }
                                addNeighbor(node);
                                return TCL_OK;
!               // Merged by Will Hrudey, Reza Sahraei
!                       } // Modified by Tony Feng for BGP.
!                 else if (strcmp(argv[1], "add-AS-neighbor") == 0) {
!                               Node * node = (Node *)TclObject::lookup(argv[2]);
!                               if (node == 0) {
!                                       tcl.resultf("Invalid node %s", argv[2]);
!                           return (TCL_ERROR);
!                               }
!                               ASnbs.push_back(node);
!                               return TCL_OK;
!                       } // End Tony Feng.
                }
                return ParentNode::command(argc,argv);
  }
diff -rc ns-2.34_original/common/node.h ns-2.34/common/node.h
*** ns-2.34_original/common/node.h          2009-06-14 10:35:45.000000000 -0700
--- ns-2.34/common/node.h              2009-10-12 18:49:07.000000000 -0700
***************
*** 59,64 ****
--- 59,66 ----
  #include "energy-model.h"
  #include "location.h"
  #include "rtmodule.h"
+ // Merged by Will Hrudey, Reza Sahraei
+ #include <list>

  class NixNode;
  class LinkHead;
***************
*** 130,135 ****
--- 132,139 ----

                inline int address() { return address_;}
                inline int nodeid() { return nodeid_;}
+       // Merged by Will Hrudey, Reza Sahraei
+               inline int as_number() { return as_num_; } //Added by Tony Feng
                inline bool exist_namchan() const { return (namChan_ != 0); }

                virtual int command(int argc, const char*const* argv);
***************
*** 155,160 ****
--- 159,167 ----
                void addNeighbor(Node *node);

                static Node* get_node_by_address(nsaddr_t);
```

```
+       // Merged by Will Hrudey, Reza Sahraei
+       //AS neighbor list for BGP autoconfig, added by Tony Feng
+       list<Node*> ASnbs;

          //routines for supporting routing
          void route_notify (RoutingModule *rtm);
****************
*** 168,173 ****
--- 175,182 ----
          LIST_ENTRY(Node) entry;  // declare list entry structure
          int address_;
          int nodeid_;                    // for nam use
+       // Merged by Will Hrudey, Reza Sahraei
+          int as_num_;     // Added by Tony Feng for BGP

          // Nam tracing facility
        Tcl_Channel namChan_;
diff -rc ns-2.34_original/common/packet.h ns-2.34/common/packet.h
*** ns-2.34_original/common/packet.h        2009-06-14 10:35:44.000000000 -0700
--- ns-2.34/common/packet.h          2009-10-12 21:29:17.000000000 -0700
****************
*** 183,189 ****
  static const packet_t PT_AOMDV = 61;

        // insert new packet types here
! static packet_t       PT_NTYPE = 62; // This MUST be the LAST one

  enum packetClass
  {
--- 183,194 ----
  static const packet_t PT_AOMDV = 61;

        // insert new packet types here
! // Merged by Will Hrudey, Reza Sahraei
! static const packet_t PT_RTPROTO_BGP = 62;  // For bgp implementation, added by Tony Feng
! static const packet_t PT_TCPMASTER = 63;
! static const packet_t PT_PEERENTRY = 64; // end Tony Feng
!
! static packet_t       PT_NTYPE = 65; // This MUST be the LAST one

  enum packetClass
  {
****************
*** 303,308 ****
--- 308,317 ----
                        name_[PT_RTCP]= "rtcp";
                        name_[PT_RTP]= "rtp";
                        name_[PT_RTPROTO_DV]= "rtProtoDV";
+             // Merged by Will Hrudey, Reza Sahraei
+                       name_[PT_RTPROTO_BGP]= "rtProtoBGP"; // For bgp implementation, added by Tony
Feng
+                       name_[PT_PEERENTRY]= "PeerEntry";
+                       name_[PT_TCPMASTER]= "tcpmaster"; // end Tony Feng
                        name_[PT_CtrMcast_Encap]= "CtrMcast_Encap";
                        name_[PT_CtrMcast_Decap]= "CtrMcast_Decap";
                        name_[PT_SRM]= "SRM";
```

```
diff -rc ns-2.34_original/common/simulator.cc ns-2.34/common/simulator.cc
*** ns-2.34_original/common/simulator.cc   2009-06-14 10:35:44.000000000 -0700
--- ns-2.34/common/simulator.cc     2009-10-12 21:32:01.000000000 -0700
***************
*** 189,195 ****
                                        nh = rtobject_->lookup_flat(i, j);
                                        if (nh >= 0) {
                                                NsObject *l_head = get_link_head(nodelist_[i], nh);
!                                               sprintf(tmp, "%d", j);
                                                nodelist_[i]->add_route(tmp, l_head);
                                        }
                                }
--- 189,200 ----
                                        nh = rtobject_->lookup_flat(i, j);
                                        if (nh >= 0) {
                                                NsObject *l_head = get_link_head(nodelist_[i], nh);
!                                       // Merged by Will Hrudey, Reza Sahraei
!                                               // Modified by Tony Feng. We use the node address instead of
node_id.
!                                       Tcl::instance().evalf("[Simulator instance] get-node-by-id %d",j);
!                                       Node * node = (Node*) TclObject::lookup(Tcl::instance().result());
!                                       sprintf(tmp, "%d", node->address());
!                                       // End Tony Feng
                                                nodelist_[i]->add_route(tmp, l_head);
                                        }
                                }
diff -rc ns-2.34_original/routing/route.cc ns-2.34/routing/route.cc
*** ns-2.34_original/routing/route.cc           2009-06-14 10:35:43.000000000 -0700
--- ns-2.34/routing/route.cc            2009-10-12 22:05:04.000000000 -0700
***************
*** 393,406 ****
  {
          check(src);
          check(dst);
          adj_[INDEX(src, dst, size_)].cost = cost;
  }
  void RouteLogic::insert(int src, int dst, double cost, void* entry_)
  {
          check(src);
          check(dst);
!         adj_[INDEX(src, dst, size_)].cost = cost;
!         adj_[INDEX(src, dst, size_)].entry = entry_;
  }

  void RouteLogic::reset(int src, int dst)
--- 393,426 ----
  {
          check(src);
          check(dst);
+       // Merged by Will Hrudey, Reza Sahraei
+       // Modified by Tony Feng. check if src and dst come from the same as.
+       // Note that index = nodeid +1.
+       Tcl& tcl = Tcl::instance();
+         tcl.evalf("[[Simulator instance] get-node-by-id %d] set as_num_",src-1);
+         int as_num_src = atoi(tcl.result());
+       tcl.evalf("[[Simulator instance] get-node-by-id %d] set as_num_",dst-1);
```

```
+           int as_num_dst = atoi(tcl.result());
+        if (as_num_src == as_num_dst)
+        // End Tony Feng.
            adj_[INDEX(src, dst, size_)].cost = cost;
  }
  void RouteLogic::insert(int src, int dst, double cost, void* entry_)
  {
            check(src);
            check(dst);
!       // Merged by Will Hrudey, Reza Sahraei
!       // Modified by Tony Feng. check if src and dst come from the same AS.
!       Tcl& tcl = Tcl::instance();
!           tcl.evalf("[[Simulator instance] get-node-by-id %d] set as_num_",src-1);
!           int as_num_src = atoi(tcl.result());
!       tcl.evalf("[[Simulator instance] get-node-by-id %d] set as_num_",dst-1);
!           int as_num_dst = atoi(tcl.result());
!       if (as_num_src == as_num_dst) {
!             adj_[INDEX(src, dst, size_)].cost = cost;
!             adj_[INDEX(src, dst, size_)].entry = entry_;
!       }
!       // End Tony Feng.
  }

  void RouteLogic::reset(int src, int dst)
diff -rc ns-2.34_original/routing/rtmodule.cc ns-2.34/routing/rtmodule.cc
*** ns-2.34_original/routing/rtmodule.cc      2009-06-14 10:35:43.000000000 -0700
--- ns-2.34/routing/rtmodule.cc        2009-10-14 21:15:40.000000000 -0700
***************
*** 142,147 ****
--- 142,158 ----
            bind("classifier_", (TclObject**)&classifier_);
  }

+ // Merged by Will Hrudey, Reza Sahraei
+ // Added by Tony Feng for BGP.
+ static class BGPRoutingModuleClass : public TclClass {
+ public:
+           BGPRoutingModuleClass() : TclClass("RtModule/BGP") {}
+           TclObject* create(int, const char*const*) {
+                     return ( new BGPRoutingModule );
+           }
+ } class_bgp_module;
+ // End Tony Feng.
+
  int RoutingModule::command(int argc, const char*const* argv)
  {
            Tcl& tcl = Tcl::instance();
***************
*** 509,511 ****
--- 520,546 ----
                     next_rtm_->add_route(dst, target);
  }

+ // Merged by Will Hrudey, Reza Sahraei
+ // Added by Tony Feng for BGP.
+ BGPRoutingModule::BGPRoutingModule() { }
```

```
+
+ int BGPRoutingModule::command(int argc, const char*const* argv) {
+           Tcl& tcl = Tcl::instance();
+           if ( argc == 3 ) {
+                     if ( strcmp(argv[1], "route-notify") == 0 ) {
+                       Node *node = (Node *)(TclObject::lookup(argv[2]));
+                                 if (node == NULL) {
+                                           tcl.add_errorf("Invalid node object %s", argv[2]);
+                                           return TCL_ERROR;
+                                 }
+                                 if (node != n_) {
+                                           tcl.add_errorf("Node object %s different from n_", argv[2]);
+                                           return TCL_ERROR;
+                                 }
+                                 n_->route_notify(this);
+                                 return TCL_OK;
+                     }
+           }
+           return (RoutingModule::command(argc, argv));
+ }
+ // End Tony Feng.
diff -rc ns-2.34_original/routing/rtmodule.h ns-2.34/routing/rtmodule.h
*** ns-2.34_original/routing/rtmodule.h       2009-06-14 10:35:43.000000000 -0700
--- ns-2.34/routing/rtmodule.h         2009-10-14 21:19:04.000000000 -0700
***************
*** 67,73 ****
  class Node;
  class VirtualClassifier;
  class DestHashClassifier;
!

  class RoutingModule : public TclObject {
  public:
--- 67,77 ----
  class Node;
  class VirtualClassifier;
  class DestHashClassifier;
! // Merged by Will Hrudey
! // Added by Tony Feng for BGP.
! class IPv4Classifier;
! class rtProtoBGP;
! // End Tony Feng.

  class RoutingModule : public TclObject {
  public:
***************
*** 178,181 ****
--- 182,197 ----
           virtual void add_route(char *dst, NsObject *target){}
  };

+ // Merged by Will Hrudey, Reza Sahraei
+ // Added by Tony Feng for BGP.
+ class BGPRoutingModule : public RoutingModule {
+ public:
+        BGPRoutingModule();
```

```
+       virtual const char* module_name() const {return "BGP";}
+       virtual int command (int argc, const char* const * argv);
+ protected:
+       IPv4Classifier *classifier_;
+       rtProtoBGP *bgp_agent_;
+ };
+ // End Tony Feng.
  #endif //  ns_rtmodule_h
diff -rc ns-2.34_original/tcl/lib/ns-default.tcl ns-2.34/tcl/lib/ns-default.tcl
*** ns-2.34_original/tcl/lib/ns-default.tcl      2009-06-14 10:35:41.000000000 -0700
--- ns-2.34/tcl/lib/ns-default.tcl      2009-10-14 21:21:14.000000000 -0700
***************
*** 1345,1350 ****
--- 1345,1370 ----
  Agent/rtProto/DV set INFINITY               [Agent set ttl_]
  Agent/rtProto/DV set advertInterval          2

+ # Merged by Will Hrudey, Reza Sahraei
+ # Added by Tony Feng for BGP
+ Agent/rtProto/BGP set connretry_interval_ 120
+ Agent/rtProto/BGP set masoi_ 15
+ Agent/rtProto/BGP set cluster_num 0
+ Agent/rtProto/BGP set bgp_id_ 0
+ Agent/rtProto/BGP set as_num_ 0
+ Agent/rtProto/BGP set auto_config_ false
+ Agent/rtProto/BGP set preference_  80
+
+ # PeerEntry
+ Agent/PeerEntry set ipaddr_ 0
+ Agent/PeerEntry set as_num_ 0
+ Agent/PeerEntry set bgp_id_ 0
+ Agent/PeerEntry set return_ipaddr_ 0
+ Agent/PeerEntry set hold_time_ 90
+ Agent/PeerEntry set keep_alive_interval_ 30
+ Agent/PeerEntry set mrai_ 30
+ # End Tony Feng
+
  Agent/Encapsulator set status_ 1
  Agent/Encapsulator set overhead_ 20

diff -rc ns-2.34_original/tcl/lib/ns-lib.tcl ns-2.34/tcl/lib/ns-lib.tcl
*** ns-2.34_original/tcl/lib/ns-lib.tcl          2009-06-14 10:35:41.000000000 -0700
--- ns-2.34/tcl/lib/ns-lib.tcl 2009-10-21 23:12:39.000000000 -0700
***************
*** 229,234 ****
--- 229,243 ----
  }

  source ns-qsnode.tcl
+ # Mergedy by Will Hrudey, Reza Sahraei
+ # Added by Tony Feng for BGP
+ source ../bgp/ns-bgp-node.tcl
+ source ../bgp/ns-rtProtoBGP.tcl
+ source ../bgp/ns-bgp-peerentry.tcl
+
+ #TCPMaster
```

```
+ source ../bgp/ns-tcpmaster.tcl
+ # End Tony Feng

  # Obsolete modules
  #source ns-wireless-mip.tcl
**********
*** 395,400 ****
--- 404,421 ----
            }
  }

+ # Merged by Will Hrudey, Reza Sahraei
+ # Added by Tony Feng for BGP
+ Simulator instproc BGP { val } {
+           if { $val == "ON" } {
+                     Node enable-module BGP
+                     Node disable-module Base
+           } else {
+                     Node disable-module BGP
+                     Node enable-module Base
+           }
+ }
+ # End Tony Feng

  Simulator instproc PGM { val } {
        if { $val == "ON" } {
**********
*** 1113,1118 ****
--- 1134,1147 ----
            # Register this simplex link in nam link list. Treat it as
            # a duplex link in nam
            $self register-nam-linkconfig $link_($sid:$did)
+         # Merged by Will Hrudey, Reza Sahraei
+           # Added by Tony Feng for BGP.
+           if { [$n1 set as_num_] != [$n2 set as_num_] } {
+           # n1 and n2 reside in different AS, add a route to n2 in n1's classifier.
+                     $n1 add-route [$n2 set address_] [[set link_($sid:$did)] set head_]
+                     $n1 cmd add-AS-neighbor $n2
+           }
+           # End Tony Feng.
  }

  #
diff -rc ns-2.34_original/tcl/lib/ns-node.tcl ns-2.34/tcl/lib/ns-node.tcl
*** ns-2.34_original/tcl/lib/ns-node.tcl         2009-06-14 10:35:41.000000000 -0700
--- ns-2.34/tcl/lib/ns-node.tcl           2009-10-14 21:32:24.000000000 -0700
**********
*** 66,82 ****
            eval $self next $args

        $self instvar id_ agents_ dmux_ neighbor_ rtsize_ address_ \
!                                   nodetype_ multiPath_ ns_ rtnotif_ ptnotif_

            set ns_ [Simulator instance]
            set id_ [Node getid]
            $self nodeid $id_ ;# Propagate id_ into c++ space
```

```
            if {[llength $args] != 0} {
!                       set address_ [lindex $args 0]
            } else {
                        set address_ $id_
            }
            $self cmd addr $address_; # Propagate address_ into C++ space
            #$ns_ add-node $self $id_
            set neighbor_ ""
--- 66,97 ----
            eval $self next $args

        $self instvar id_ agents_ dmux_ neighbor_ rtsize_ address_ \
!                               nodetype_ multiPath_ ns_ rtnotif_ ptnotif_ as_num_

            set ns_ [Simulator instance]
            set id_ [Node getid]
            $self nodeid $id_ ;# Propagate id_ into c++ space

+       # Merged by Will Hrudey, Reza Sahraei
+       # Modified by Tony Feng for BGP
            if {[llength $args] != 0} {
!                       if {[llength $args] == 1} {
!                       set arg_0 [lindex $args 0]
!                               if { [scan $arg_0 "%s" ] != -1 } {
!                                       # create node with arg_0 of [as_num:ipaddr] format
!                                       $self parse-addr $arg_0
!                               } else {
!                                       # create node with arg_0 of int value
!                                       set address_ $arg_0
!                                       set as_num_ 0
!                               }
!                       }
            } else {
                        set address_ $id_
+               set as_num_ 0
            }
+           # End Tony Feng
+
            $self cmd addr $address_; # Propagate address_ into C++ space
            #$ns_ add-node $self $id_
            set neighbor_ ""
diff -rc ns-2.34_original/tcp/rq.cc ns-2.34/tcp/rq.cc
*** ns-2.34_original/tcp/rq.cc          2009-06-14 10:35:44.000000000 -0700
--- ns-2.34/tcp/rq.cc           2009-10-14 21:37:39.000000000 -0700
***************
*** 298,305 ****
   * last seq# number in the segment plus one
   */

  TcpFlag
! ReassemblyQueue::add(TcpSeq start, TcpSeq end, TcpFlag tiflags, RqFlag rqflags)
  {

            int needmerge = FALSE;
--- 298,306 ----
```

```
   * last seq# number in the segment plus one
   */

+ // Merged by Will Hrudey, Reza Sahraei
  TcpFlag
! ReassemblyQueue::add(TcpSeq start, TcpSeq end, TcpFlag tiflags, RqFlag rqflags, AppData* data)
  {

            int needmerge = FALSE;
***************
*** 329,334 ****
--- 330,338 ----
                        head_->rqflags_ = rqflags;
                        head_->cnt_ = initcnt;

+               // Merged by Will Hrudey, Reza Sahraei
+               head_->data = data; //Added by Zheng Wang for BGP
+
                        total_ = (end - start);

                        //
***************
*** 500,505 ****
--- 504,512 ----
                        n->prev_ = p;
                        n->next_ = q;

+               // Merged by Will Hrudey, Reza Sahraei
+               n->data = data; //Added by Zheng Wang for BGP
+
                        push(n);

                        if (p)
***************
*** 530,535 ****
--- 537,545 ----
                                rcv_nxt_ = end;
                        }

+               // Merged by Will Hrudey, Reza Sahraei
+                 toReceiveQueue->enqueue((TcpData*)(q->data)); //Added by Zheng Wang for BGP
+
                        return tiflags;
                }
  }
diff -rc ns-2.34_original/tcp/rq.h ns-2.34/tcp/rq.h
*** ns-2.34_original/tcp/rq.h           2009-06-14 10:35:44.000000000 -0700
--- ns-2.34/tcp/rq.h            2009-10-14 21:44:31.000000000 -0700
***************
*** 68,73 ****
--- 68,75 ----

  #include <stdio.h>
  #include <stdlib.h>
+ // Merged by Will Hrudey, Reza Sahraei
+ #include "receive_queue.h" //Added by Zheng Wang for BGP
```

```
 /*
  * ReassemblyQueue: keeps both a stack and linked list of segments
***************
*** 99,111 ****
                        TcpFlag pflags_;  // flags derived from tcp hdr
                        RqFlag  rqflags_;// book-keeping flags
                        int     cnt_;               // refs to this block
                };

 public:
                ReassemblyQueue(TcpSeq& rcvnxt) :
                        head_(NULL), tail_(NULL), top_(NULL), bottom_(NULL), hint_(NULL), total_(0),
rcv_nxt_(rcvnxt) { };
                int empty() { return (head_ == NULL); }
!               int add(TcpSeq sseq, TcpSeq eseq, TcpFlag pflags, RqFlag rqflags = 0);
                int maxseq() { return (tail_ ? (tail_->endseq_) : -1); }
                int minseq() { return (head_ ? (head_->startseq_) : -1); }
                int total() { return total_; }
--- 101,116 ----
                        TcpFlag pflags_;  // flags derived from tcp hdr
                        RqFlag  rqflags_;// book-keeping flags
                        int     cnt_;               // refs to this block
+               // Merged by Will Hrudey, Reza Sahraei
+               AppData* data;          // Added by Zheng Wang for BGP
                };

 public:
                ReassemblyQueue(TcpSeq& rcvnxt) :
                        head_(NULL), tail_(NULL), top_(NULL), bottom_(NULL), hint_(NULL), total_(0),
rcv_nxt_(rcvnxt) { };
                int empty() { return (head_ == NULL); }
!       // Merged by Will Hrudey, Reza Sahraei
!               int add(TcpSeq sseq, TcpSeq eseq, TcpFlag pflags, RqFlag rqflags = 0, AppData* data = 0);
//Modified by Zheng Wang for BGP
                int maxseq() { return (tail_ ? (tail_->endseq_) : -1); }
                int minseq() { return (head_ ? (head_->startseq_) : -1); }
                int total() { return total_; }
***************
*** 121,126 ****
--- 126,133 ----
                   return (clearto(rcv_nxt_));
                }
                void dumplist();   // for debugging
+       // Merged by Will Hrudey, Reza Sahraei
+               void connRevQueue(ReceiveQueue* revQueue){toReceiveQueue = revQueue;} //Added by
Zheng Wang for BGP

                // cache of allocated seginfo blocks
                static seginfo* newseginfo();
***************
*** 143,148 ****
--- 150,158 ----
                // within TCP to set rcv_nxt and thus to set the ACK field.  It is also
                // used in the SACK sender as sack_min_
```

```
+       // Merged by Will Hrudey, Reza Sahraei
+       ReceiveQueue* toReceiveQueue;  //Added by Zheng Wang for BGP
+
          TcpSeq& rcv_nxt_;          // start seq of next expected thing
          TcpFlag coalesce(seginfo*, seginfo*, seginfo*);
          void fremove(seginfo*);    // remove from FIFO
diff -rc ns-2.34_original/tcp/scoreboard-rq.cc ns-2.34/tcp/scoreboard-rq.cc
*** ns-2.34_original/tcp/scoreboard-rq.cc     2009-06-14 10:35:44.000000000 -0700
--- ns-2.34/tcp/scoreboard-rq.cc       2009-10-14 21:46:11.000000000 -0700
***************
*** 84,90 ****

          for(int i = 0 ; i < tcph->sa_length() ; i++){
                  //printf("l: %i r: %i\n", tcph->sa_left(i), tcph->sa_right(i));
!                 rq_.add(tcph->sa_left(i), tcph->sa_right(i), 0);
          }
          changed_ = changed_  || (old_total != rq_.total());
          return 0;
--- 84,91 ----

          for(int i = 0 ; i < tcph->sa_length() ; i++){
                  //printf("l: %i r: %i\n", tcph->sa_left(i), tcph->sa_right(i));
!            // Merged by Will Hrudey, Reza Sahraei
!                 rq_.add(tcph->sa_left(i), tcph->sa_right(i), 0, NULL);
          }
          changed_ = changed_  || (old_total != rq_.total());
          return 0;
diff -rc ns-2.34_original/tcp/tcp-full.cc ns-2.34/tcp/tcp-full.cc
*** ns-2.34_original/tcp/tcp-full.cc 2009-06-14 10:35:44.000000000 -0700
--- ns-2.34/tcp/tcp-full.cc   2009-10-14 22:08:57.000000000 -0700
***************
*** 342,347 ****
--- 342,370 ----
          return;
  }

+ // Merged by Will Hrudey, Reza Sahraei
+ /*
+  * send a string of nBytes, added by Zheng Wang for BGP
+  */
+ void
+ FullTcpAgent::advance_bytes(int nBytes, const char* const data, Continuation* caller)
+ {
+     if (writeCont != NULL){
+       printf("write error - socket already in blocking write\n");
+       if (caller != NULL)
+         caller->failure();
+       return;
+     }
+     if(toSendQueue==NULL)  {
+       toSendQueue = new SendQueue();
+     }
+     if (data!=NULL) {
+       writeCont = caller;
+       toSendQueue->enqueue(nBytes,data);
+     }
```

```
+     advance_bytes(nBytes);
+ }
+
  /*
   * the byte-oriented interface: advance_bytes(int nbytes)
   */
***************
*** 465,474 ****
--- 488,503 ----
            case TCPS_LISTEN:
                    cancel_timers();
                    newstate(TCPS_CLOSED);
+               // Merged by Will Hrudey, Reza Sahraei
+               if(mySocket) // added by Tony Feng, informs the socket that tcp closed successfully
+                 mySocket->disconnected();
                    finish();
                    break;
            case TCPS_SYN_SENT:
                    newstate(TCPS_CLOSED);
+               // Merged by Will Hrudey, Reza Sahraei
+               if(mySocket) // added by Tony Feng, informs the socket that tcp closed successfully
+                 mySocket->disconnected();
                    /* fall through */
            case TCPS_LAST_ACK:
                    flags_ |= TF_NEEDFIN;
***************
*** 721,726 ****
--- 750,758 ----
        int tiflags = tcph->flags();
            int fillshole = (start == rcv_nxt_);
            int flags;
+
+        // Merged by Will Hrudey, Reza Sahraei
+           AppData* data = pkt->userdata();   //Added by Zheng Wang for BGP

            // end contains the seq of the last byte of
            // in the packet plus one
***************
*** 731,737 ****
                    abort();
            }

!           flags = rq_.add(start, end, tiflags, 0);

            //present:
            //
--- 763,770 ----
                    abort();
            }

!       // Merged by Will Hrudey, Reza Sahraei
!           flags = rq_.add(start, end, tiflags, 0, data); //Modified by Zheng Wang for BGP

            //present:
            //
***************
```

```
*** 756,761 ****
--- 789,839 ----
        return (flags);
  }

+ // Merged by Will Hrudey, Reza Sahraei
+ //Added by Zheng Wang for BGP
+ void FullTcpAgent::read(char* buffer, int nbytes, Continuation* caller)
+ {
+     if (readCont != NULL){
+       printf("read error - socket already in blocking read\n");
+       if (caller != NULL)
+         caller->failure();
+       return;
+     }
+
+     // if requested data is in the buffer, get it from buffer
+     // We use the fact that a data object arrives in the first TCP segment.
+     if(nbytes <= dataReceived) {
+       dataReceived -= nbytes;
+       if(toReceiveQueue->is_empty()) {
+                printf("receive queue is empty, exit\n");
+               return;
+       }
+       toReceiveQueue->retrieve_data(nbytes,buffer);
+               caller->success();
+     } else {
+       inbuffer = buffer;
+       readCont = caller;
+       readSize = nbytes;
+     }
+  }
+
+ void FullTcpAgent::recvBytes(int bytes)
+  {
+     dataReceived+= bytes;
+     if((readCont != NULL) && (dataReceived >= readSize)) {
+               toReceiveQueue->retrieve_data(readSize,inbuffer);
+               mySocket->appCallWaiting = false;
+       //mySocket->app_call_waiting = NULL;
+       dataReceived -= readSize;
+       Continuation* rc = readCont;
+       readCont = NULL;
+       rc->success();
+
+     }
+     Agent::recvBytes(bytes);
+  }
+ // End Zheng Wang
+
  /*
   * utility function to set rcv_next_ during inital exchange of seq #s
   */
**********
*** 915,920 ****
--- 993,1011 ----
```

```
 //prpkt(p);
 //}

+       // Merged by Will Hrudey, Reza Sahraei
+         //Added by Zheng Wang for BGP
+                 //Set data field
+                 if(toSendQueue)
+                 {
+                         if(!toSendQueue->is_empty())
+                         {
+                                 TcpData* pData = toSendQueue->get_data(seqno,datalen);
+                                 p->setdata(pData);
+                         }
+                 }
+         //End Zheng Wang
+
          send(p, 0);

          return;
***************
*** 1299,1304 ****
--- 1390,1404 ----

          if (ackno == maxseq_) {
                  cancel_rtx_timer();           // all data ACKd
+             // Merged by Will Hrudey, Reza Sahraei
+               // Added by Tony Feng for Socket::wirte()
+             if(writeCont != NULL) {
+               Continuation *  app_call_waiting = writeCont;
+               writeCont = NULL;
+               mySocket->appCallWaiting = false;
+               app_call_waiting->success();
+             }
+             // End Tony Feng
          } else if (progress) {
                  set_rtx_timer();
          }
***************
*** 1640,1645 ****
--- 1740,1749 ----
                                  //          changes DELACK to ACKNOW and calls tcp_output()
                                  rcv_nxt_ += datalen;
                                  flags_ |= TF_DELACK;
+
+                 // Merged by Will Hrudey, Reza Sahraei
+                                 toReceiveQueue->enqueue((TcpData*)pkt->userdata()); // Added by Zheng
Wang for BGP
+
                                  recvBytes(datalen); // notify application of "delivery"
                                  //
                                  // special code here to simulate the operation
***************
*** 1722,1728 ****
                                  fid_ = iph->flowid();
                  }
```

```
!                       newstate(TCPS_SYN_RECEIVED);
                        goto trimthenstep6;

        /*
--- 1826,1842 ----
                                fid_ = iph->flowid();
                        }

!               // Merged by Will Hrudey, Reza Sahraei
!                       //Modified by Zheng Wang
!                       if( mySocket!=NULL && mySocket->isListening) {
!                     hdr_ip* iph1 = hdr_ip::access(pkt);
!                     //Received SYN for listening tcp, we create a new reading socket.
!                             tcpMaster->newInComing(pkt,mySocket);
!                     return;
!                       } else {
!                               newstate(TCPS_SYN_RECEIVED);
!               }
!                       // End Zheng Wang
                        goto trimthenstep6;

        /*
***************
*** 1832,1837 ****
--- 1946,1956 ----
                                        flags_ &= ~TF_NEEDFIN;
                                        tiflags &= ~TH_SYN;
                                } else {
+                       // Merged by Will Hrudey, Reza Sahraei
+                       //Added by Zheng Wang
+                       if (mySocket)
+                                       mySocket->connected();
+                       //End Zheng Wang
                                        newstate(TCPS_ESTABLISHED);
                                }

***************
*** 2092,2097 ****
--- 2211,2218 ----
                                newstate(TCPS_FIN_WAIT_1);
                        flags_ &= ~TF_NEEDFIN;
                } else {
+                       // Merged by Will Hrudey, Reza Sahraei
+                       mySocket->listeningSocket->addConnection(mySocket); //Added by Zheng Wang for
BGP
                        newstate(TCPS_ESTABLISHED);
                }

***************
*** 2335,2340 ****
--- 2456,2467 ----
                        case TCPS_CLOSING:      /* simultaneous active close */;
                                if (ourfinisacked) {
                                        newstate(TCPS_CLOSED);
+                       // Merged by Will Hrudey, Reza Sahraei
+                       // Added by Tony Feng for BGP
```

```
+                       if(mySocket)  {
+                           mySocket->disconnected(); // Informs the socket that tcp closed successfully
+                       }
+                       // End Tony Feng
                                cancel_timers();
                        }
                        break;
**********
*** 2348,2353 ****
--- 2475,2486 ----
                        // K: added state change here
                        if (ourfinisacked) {
                                newstate(TCPS_CLOSED);
+                       // Merged by Will Hrudey, Reza Sahraei
+                       // Added by Tony Feng for BGP
+                       if(mySocket)  {
+                           mySocket->disconnected(); // Informs the socket that tcp closed successfully
+                       }
+                       // End Tony Feng
                                finish(); // cancels timers, erc
                                reset(); // for connection re-use (bug fix from ns-users list)
                                goto drop;
**********
*** 2635,2641 ****
            * Due to F. Hernandez-Campos' fix in recv(), we may send an ACK
            * while in the CLOSED state.  -M. Weigle 7/24/01
            */
!           if (state_ == TCPS_LISTEN) {
                    // shouldn't be getting timeouts here
              if (debug_) {
                        fprintf(stderr, "%f: FullTcpAgent(%s): unexpected timeout %d in state %s\n",
--- 2768,2775 ----
            * Due to F. Hernandez-Campos' fix in recv(), we may send an ACK
            * while in the CLOSED state.  -M. Weigle 7/24/01
            */
!       // Merged by Will Hrudey, Reza Sahraei
!           if (state_ == TCPS_LISTEN && (mySocket==NULL || !mySocket->isListening)) { // Modified
by Tony Feng for BGP
                    // shouldn't be getting timeouts here
              if (debug_) {
                        fprintf(stderr, "%f: FullTcpAgent(%s): unexpected timeout %d in state %s\n",
diff -rc ns-2.34_original/tcp/tcp-full.h ns-2.34/tcp/tcp-full.h
*** ns-2.34_original/tcp/tcp-full.h  2009-06-14 10:35:44.000000000 -0700
--- ns-2.34/tcp/tcp-full.h    2009-10-14 22:16:07.000000000 -0700
**********
*** 39,44 ****
--- 39,52 ----

  #include "tcp.h"
  #include "rq.h"
+ // Merged by Will Hrudey, Reza Sahraei
+ //Added by Zheng Wang
+ #include "send_queue.h"
+ #include "receive_queue.h"
+ #include "tcp_socket.h"
+ #include "tcp_master.h"
```

```
+ #include "continuation.h"
+ //End Zheng Wang

  /*
   * most of these defines are directly from
**********
*** 114,137 ****
  };

  class FullTcpAgent : public TcpAgent {
  public:
          FullTcpAgent() :
                  closed_(0), pipe_(-1), rtxbytes_(0), fastrecov_(FALSE),
          last_send_time_(-1.0), infinite_send_(FALSE), irs_(-1),
          delack_timer_(this), flags_(0),
          state_(TCPS_CLOSED), recent_ce_(FALSE),
-         last_state_(TCPS_CLOSED), rq_(rcv_nxt_), last_ack_sent_(-1) { }

!         ~FullTcpAgent() { cancel_timers(); rq_.clear(); }
          virtual void recv(Packet *pkt, Handler*);
          virtual void timeout(int tno);          // tcp_timers() in real code
          virtual void close() { usrclosed(); }
          void advanceby(int);          // over-rides tcp base version
          void advance_bytes(int);    // unique to full-tcp
        virtual void sendmsg(int nbytes, const char *flags = 0);
        virtual int& size() { return maxseg_; } //FullTcp uses maxseg_ for size_
          virtual int command(int argc, const char*const* argv);
          virtual void reset();                   // reset to a known point
  protected:
          virtual void delay_bind_init_all();
          virtual int delay_bind_dispatch(const char *varName, const char *localName, TclObject *tracer);
--- 122,181 ----
  };

  class FullTcpAgent : public TcpAgent {
+ // Merged by Will Hrudey, Reza Sahraei
+ friend class TcpMaster;
  public:
          FullTcpAgent() :
                  closed_(0), pipe_(-1), rtxbytes_(0), fastrecov_(FALSE),
          last_send_time_(-1.0), infinite_send_(FALSE), irs_(-1),
          delack_timer_(this), flags_(0),
          state_(TCPS_CLOSED), recent_ce_(FALSE),

!           // Merged by Will Hrudey, Reza Sahraei
!         last_state_(TCPS_CLOSED), rq_(rcv_nxt_), last_ack_sent_(-1)
!           {   // Modified by Zheng Wang for BGP
!                   toSendQueue = new SendQueue();
!               toReceiveQueue = new ReceiveQueue();
!                   rq_.connRevQueue(toReceiveQueue);
!               dataReceived = 0;
!             readCont = NULL;
!             writeCont = NULL;
!                   }
!
!         ~FullTcpAgent()
```

```
!               {    cancel_timers();
!                    rq_.clear();
!                    if(toSendQueue)
!                                 delete toSendQueue;
!                         if(toReceiveQueue)
!                                 delete toReceiveQueue;
!               }    // End Zheng Wang
!
                virtual void recv(Packet *pkt, Handler*);
                virtual void timeout(int tno);          // tcp_timers() in real code
                virtual void close() { usrclosed(); }
                void advanceby(int);        // over-rides tcp base version
                void advance_bytes(int);    // unique to full-tcp
+
+               // Merged by Will Hrudey, Reza Sahraei
+                  // Added by Zheng Wang for BGP
+                  void advance_bytes(int nBytes, const char* const data, Continuation* caller);
+                  void recvBytes(int bytes);  // Overrides Agent's recvBytes();
+                  void read(char* buffer, int nbytes, Continuation* caller);
+                  // End Zheng Wang
+
               virtual void sendmsg(int nbytes, const char *flags = 0);
               virtual int& size() { return maxseg_; } //FullTcp uses maxseg_ for size_
                  virtual int command(int argc, const char*const* argv);
                  virtual void reset();                      // reset to a known point
+
+               // Merged by Will Hrudey, Reza Sahraei
+                  // Added by Zheng Wang
+                  ReceiveQueue* getRevQueue() {return toReceiveQueue;}
+               TcpSocket* mySocket;
+               TcpMaster* tcpMaster;
+                  // End Zheng Wang
+
 protected:
                  virtual void delay_bind_init_all();
                  virtual int delay_bind_dispatch(const char *varName, const char *localName, TclObject *tracer);
**********
*** 243,248 ****
--- 287,304 ----
                  double recent_;             // ts on SYN written by peer
                  double recent_age_;         // my time when recent_ was set

+               // Merged by Will Hrudey, Reza Sahraei
+                  // Added by Zheng Wang for BGP
+                  SendQueue* toSendQueue;
+                  ReceiveQueue* toReceiveQueue;
+                  int dataReceived;
+                  Continuation* readCont;
+               Continuation* writeCont;
+                  int readSize;
+               int writeSize;
+                  char* inbuffer;
+                  // End Zheng Wang
+
                  /*
                   * setting iw, specific to tcp-full, called
```

```
* by TcpAgent::reset()
```

11. REFERENCES

[1] Wikipedia, ns [Online]. Available: http://en.wikipedia.org/wiki/Ns_(simulator).

[2] D. Millis, "Exterior Gateway Protocol Formal Specification," *RFC 904*, BBN, April 1984.

[3] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," *RFC 1771*, March 1995.

[4] T. D. Feng, R. Ballantyne, and Lj Trajkovic, "Implementation of BGP in a network simulator," *Proc. Applied Telecommunication Symposium*, *ATS '04*, Arlington, Virginia, Apr. 2004.

[5] ns-BGP integration with ns-2.33 [Online]. Available: http://www.ensc.sfu.ca/~ljilja/cnl/projects/BGP-ns-2.33/ENSC-891_Summer08_report_hrudey.pdf.

[6] BGP, Border Gateway Protocol [Online].
Available: http://www.networksorcery.com/enp/protocol/bgp.htm.

[7] P. Raines and J. Tranter, *TCL/TK in a nutshell*. O'Reilly, 1999.

[8] ns-2 [Online]. Available: http://www.isi.edu/nsnam/ns.

[9] ns-2 manual [Online]. Available: http://www.isi.edu/nsnam/ns/doc/index.html.

[10] S. Supittayapornpong Exploration [Online]. Available: http://suchaxplore.blogspot.com/2008/07/install-ns2-allinone-233-on-ubuntu-804.html.

[11] SSFNet [Online]. Available: http://www.ssfnet.org/homePage.html.

[12] C-BGP's Wiki [Online]: Available: http://cbgp.info.ucl.ac.be/wiki/index.php.

[13] GNU Zebra BGP [Online]. Available: http://www.zebra.org/zebra/BGP.html#BGP.

[14] BGP++ [Online]. Available: http://www.ece.gatech.edu/research/labs/MANIACS/BGP++.