IMPLEMENTATION OF BGP IN A NETWORK SIMULATOR

by

Tony Dongliang Feng
B.Sc., Zhongshang University, 1997

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
In the School
of
Computing Science

SIMON FRASER UNIVERSITY

April 2004

APPROVAL

Name: **Tony Dongliang Feng**

Degree: **Master of Science**

Title of Thesis: **Implementation of BGP in a Network Simulator**

Examining Committee:

Chair: **Dr. Petra Berenbrink**
Assistant Professor

Dr. Ljiljana Trajkovic
Senior Supervisor
Professor

Dr. Uwe Glässer
Supervisor
Associate Professor

Dr. Qianping Gu
Examiner
Associate Professor
Computing Science, SFU

Date Approved: _____

ABSTRACT

Border Gateway Protocol (BGP) is the inter-domain routing protocol currently employed in the Internet. Internet growth imposes increasing requirements on BGP performance. Recent studies revealed that performance degradations in BGP are due to the highly dynamic nature of the Internet. Undesirable properties of BGP, such as poor integrity, slow convergence, and divergence, have been reported by the research community. Theoretical analysis and empirical measurements have been employed in the past, albeit with certain limitations. Simulations allow more realistic experiments with fewer simplifications than the theoretical approach. They also provide more enhanced flexibility than empirical studies permit.

In this thesis, we describe the design and implementation of a BGP-4 model (*ns-BGP*) in the network simulator ns-2 by porting the BGP-4 implementation from SSFNet. The *ns-BGP* node is based on the existing ns-2 unicast node and the *SSF.OS.BGP4* model from SSFNet. In order to provide socket support and at the same time maintain the structure of *SSF.OS.BGP4*, we also ported to ns-2 *TcpSocket*, the socket layer implementation of SSFNet. In order to support the IPv4 addressing and packet forwarding, the basic address classifier in ns-2 was replaced with a new address classifier named *IPv4Classifier*. We also modified *FullTcpAgent*, the TCP agent used by *TcpSocket*, to support user data transmission.

We performed a suite of validation tests to ensure that the *ns-BGP* model complies with the BGP-4 specifications, including BGP-4 features such as: basic peer session management (keep and drop peer), route selection, reconnection, internal BGP (iBGP), and route reflection. Finally, in the scalability analysis of ns-BGP, we showed that the model scales with respect to the number of peer sessions and the size of routing tables.

DEDICATION

To my wife Michelle, my mom and dad.

ACKNOWLEDGEMENTS

I would like to give my special thanks to my advisor, Dr. Ljiljana Trajkovic, for her guidance, support, and encouragement during my studies at Simon Fraser University. Thank you for your valuable instructions and ideas regarding the research and your continuous and patient advising regarding my academic writing.

My gratitude goes to Dr. Qianping Gu and Dr. Uwe Glässer for serving on my examining committee. Thank you for your precious time in reviewing my thesis and providing insightful comments and suggestions. I would also like to thank Dr. Petra Berenbrink for chairing the thesis defense.

I would like to express my sincere thanks to the people who helped me throughout this project and without whom this thesis would not have been finished. Many thanks to Zheng Wang for the implementation of TcpSocket, Rob Balantyne for the contribution of the IPv4Classifier, and the author of SSF.OS.BGP4, Brain J. Premore, for providing an excellent BGP implementation.

My sincere thanks go to my fellow graduate students in the Communication Networks Laboratory: Hao Johnson Chen, Hao Leo Chen, Jiaqing James Song, Qing Kenny Shao, Grace Hui Zhang, Savio Lau, Nikola Cackov, and Nenad Laskovic, for their support and wonderful friendship.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

Adj-RIBs-In	set of RIBs for incoming routes from adjacent routers
Adj-RIBs-Out	set of RIBs for outgoing routes from adjacent routers
AS	autonomous system
BGP	Border Gateway Protocol
BGP-4	Border Gateway Protocol version 4
CIDR	classless inter-domain routing
DML	Domain Modeling Language
EIGRP	Enhanced Interior Gateway Routing Protocol
FSM	finite state machine
eBGP	external BGP
iBGP	internal BGP
IDR	inter-domain routing
IGP	interior gateway protocol
IS-IS	Intermediate System to Intermediate System
ISP	Internet service provider
Loc-RIB	RIB for locally used routes
MED	Multiple Exit Discriminator
MPLS	Multiprotocol Label Switching

MRAI	minimum route advertisement interval
NLRI	Network Layer Reachability Information
OSPF	Open Shortest Path First
RFC	Request for Comments
RIB	routing information base
RIP	Routing Information Protocol
SSFNet	Scalable Simulation Framework Network models
SSF.OS.BGP4	SSFNet BGP
VPN	virtual private network

CHAPTER 1:
INTRODUCTION

The Internet began as an academic experiment in the late 1960s and has become a world-wide data network that is used for mission critical applications. The Internet is no longer owned by a single entity. It is a conglomeration of tens of thousands of independently managed computer networks.

Most Internet communications are based on data transfers over connections between pairs of hosts. Routing is the act of moving data (usually divided into packets) across a network from a source to a destination. Routing involves two basic activities: determining optimal routing paths and transporting packets through a network. The latter activity is also called as packet forwarding. Although packet forwarding is relatively straightforward, path determination can be very complex [21].

Routing in the Internet is performed on two levels (intra-domain and inter-domain) implemented by two sets of protocols. Interior gateway protocols (IGPs) [22], such as Routing Information Protocol (RIP) [26], Enhanced Interior Gateway Routing Protocol (EIGRP, Cisco's proprietary protocol) [22], Intermediate System to Intermediate System (IS-IS) [22], and Open Shortest Path First (OSPF) [22], route packets within a single Autonomous System (intra-domain). Exterior gateway protocols (EGPs) [22], such as EGP and Border Gateway Protocol (BGP) [36], route packets between Autonomous Systems (inter-domain).

1.1 BGP weaknesses

As the *de facto* inter-domain routing protocol, BGP-4 was designed in the mid 90s for a much smaller Internet. With the help of Classless Inter-domain Routing (CIDR) [4] and other modifications, it has survived several years of Internet exponential growth. It is still an open

research question how much growth BGP will be able to sustain [8]. Apart from growth issue, the research community has identified three main weaknesses concerning BGP [10].

1.1.1 Poor integrity

Routing protocols of the Internet are vulnerable to attacks and BGP is no exception. Misconfigured or deliberately malicious sources can disrupt overall Internet behavior by injecting bogus routing information into the distributed BGP routing database (by modifying, forging, or replaying BGP packets) [28].

1.1.2 Slow convergence

Labovitz et al., [25] measured routing changes in the Internet and showed that there can be considerable delay in BGP convergence due to subsequent exploration of errant paths. They observed that the delayed convergence bears an adverse effect on end-to-end traffic delay, causing packet loss and intermittent disruption of connectivity.

Griffin and Premore [14] used simulations to show how convergence is affected by the Minimum Route Advertisement Interval (MRAI) timer setting and to explore its impact on various topologies. The ubiquitously used default value of the MRAI timer appears to be much higher than necessary.

Mao et al., [27] showed that route flap damping can significantly exacerbate the convergence times of relatively stable routes. Such abnormal behavior arises from the interaction of flap damping with BGP path exploration during route withdrawal and route announcement.

1.1.3 Divergence

Varadhan et al., [38] analyzed route oscillations in simple ring topologies and showed that the independently defined routing policies of different autonomous systems can cause BGP to diverge and result in persistent route oscillations.

Labovitz et al., [24] used large traces of BGP update messages to characterize the instability of routes to destination prefixes. They also highlighted the adverse effects of inter-domain routing instability.

Griffin et al., [16] established that the problem of checking the convergence properties is NP-complete, even with full knowledge of the routing policies of each AS. In later efforts, Griffin and Wilfong [17] presented a sufficient condition for a convergent routing system and proposed an abstract model of BGP, called Simple Path Vector Protocol (SPVP), which can identify and suppress policy-based oscillations.

1.2 Employed approaches

Various techniques, including theoretical analysis, empirical measurement, and simulations, have been employed in previous research on BGP. Despite the useful results, both theoretical and empirical approaches have certain limitations.

1.2.1 Empirical measurement

By collecting and analyzing genuine traffic data from the Internet, empirical studies have reported certain unexpected pathological behavior of BGP [24], [25], [32]. Because the Internet is a collection of independently managed entities, empirical measurements need to be well considered, strategically deployed, and collaboratively maintained. The high cost to implement system changes also makes it very difficult to perform controlled comparisons of protocol variants.

1.2.2 Analytical approach

Most theoretical analyses have been aimed at proving fundamental properties of routing protocols, such as whether they are guaranteed to converge [13], [15]-[18]. However, as the network topologies and protocols become more and more complex, theoretical models that are highly simplified have become increasingly inadequate for many practical scenarios.

1.2.3 Simulations

Because of the drawback of empirical measurement and analytic methods, there has been a considerable increase in the use of simulations for analysis of communication networks. Simulations allow more realistic experiments with fewer simplifications than the theoretical approach. They also provide more enhanced flexibility than empirical studies permit [34].

In a simulation scenario, there is full control of the system and desired modifications are possible. It is more cost-effective to test proposed and novel extensions of network protocols using simulations than to deploy them in a real system. The process of model development via simulations requires the system to be well studied and understood. This process frequently uncovers problems that were unknown or not well-understood. Finally, by using visual representation (such as animation) to demonstrate the behavior of the system, a simulation feels more "real" than other methods used for system analysis.

1.3 Contribution

The goal of this thesis is to describe the implementation, validation, and scalability analysis of a BGP-4 model (ns-BGP) in the network simulator ns-2. A summary of the contributions follows.

1.3.1 Implementation of a BGP-4 model in ns-2

We implemented a BGP-4 model, the current version of BGP, in the network simulator ns-2 [30] by porting the BGP-4 implementation from SSFNet [35]. The new model is compliant with the specification [37] and also includes several extensions and experimental features. The model is well documented and hierarchically organized so that it can be easily understood, modified, and extended.

1.3.2 Validation of the ns-BGP model

We have also implemented a suite of validation tests to verify the fundamental behavior of ns-BGP. These validation tests cover the basic maintenance of peer session (keep and drop

peer), route advertisement and withdrawal, route selection, internal BGP (iBGP), and route reflection. These tests illustrated the validity of our ns-BGP implementation.

1.3.3 Analysis of the scalability property of ns-BGP

We analyzed the scalability properties of ns-BGP both with respect to the number of peer sessions and the size of the routing tables under a variety of network topologies. The analysis shows that the internal data structures and employed algorithms are scalable in terms of the number of peer sessions and the size of routing tables.

1.4 Organization of the thesis

The thesis is organized as follows. In Chapter 2, we provide background on inter-domain routing, BGP, ns-2, and SSFNet. The design and implementation of ns-BGP are described in Chapter 3. Three validation tests for route selection, peer reconnection, and route reflection are presented in Chapter 4. We analyze the scalability of ns-BGP in Chapter 5 and we conclude with Chapter 6.

CHAPTER 2:
BACKGROUND

Routers are devices that direct traffic between hosts. They build routing tables that contain routing information about the best paths to all the destinations that they know how to reach. Inter-domain routing protocols, such as BGP, were introduced because the intra-domain routing protocols do not scale well in networks that go beyond the enterprise level, with thousands of nodes and hundreds of thousands of routes [21]. In this chapter, we introduce the background information regarding inter-domain routing, BGP, and the related BGP implementation.

2.1 Inter-domain routing

The Internet consists of thousands of interconnected Autonomous Systems (ASs) loosely defined as a set of routers and networks under the same administration. A typical AS could be the network of a university, corporation, or an Internet Service Provider (ISP). Each AS is identified by a 16-bit *AS number*. This number is assigned by the numbering authorities in the way similar to the IP address assignment. Routing through the Internet depends on routing between ASs (inter-domain) and routing inside the ASs (intra-domain). Figure 2.1 shows the protocols implementing these two categories of routing.

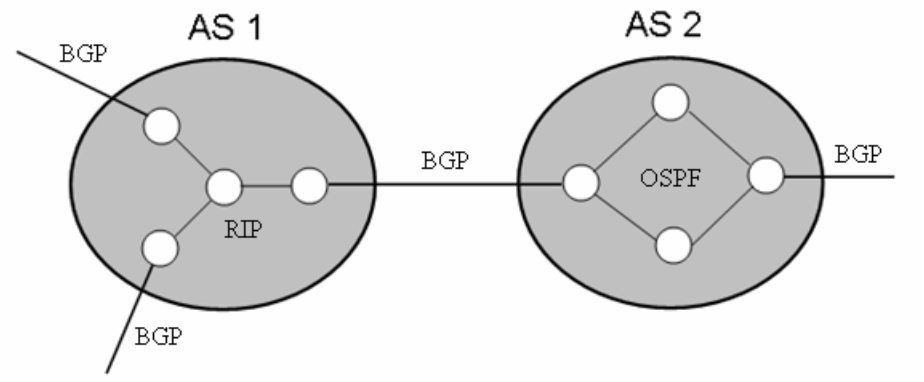


Figure 2.1: Inter-domain and intra-domain routing protocols.

Inside a single domain, routers employ interior gateway protocols (IGPs) to discover and exchange information about the internal networks to which they are directly connected. Routers from different ASs use exterior gateway protocols (EGPs), such as BGP, to exchange reachability information and determine the end-to-end path for packets traversing through multiple ASs. At the boundary of each AS, BGP border routers exchange routing information of IP address blocks, called prefixes. Each prefix consists of a 32-bit address and a mask length indicating the size of the network. For example, 192.0.1.0/24 represents a block of 256 addresses ranging from 192.0.1.0 to 192.0.1.255.

2.2 BGP overview

BGP is categorized as a *path vector protocol*, a variant of *distance vector protocol*. Instead of distributing link cost information, it propagates full path information to avoid cycles. BGP employs TCP as its transport protocol, which ensures transport reliability and eliminates the need for BGP to handle retransmission, acknowledgement, and sequencing. Routers that use BGP are called *BGP speakers*. Two BGP speakers that participate in a BGP session are called *neighbors* or *peers*. Peer routers exchange four types of messages: *open*, *update*, *notification*, and *keep-alive*. The update message carries routing information while the remaining three messages handle session management [37].

2.2.1 Peer session management

The routers that support BGP usually wait for BGP connections on port 179. A router that wants to establish a peer session will first open a TCP connection to port 179 on the peer router. Once the connection is set up, each side sends an *open* message to negotiate the session's parameters. In order to constantly monitor the reachability of their neighbors, the BGP routers send regularly *keep-alive* messages. During the opening exchange, the BGP routers announce a *hold time*, the maximum interval during which the peer should have to wait between successive messages. Failure to receive a message during the interval specified by *hold time*, will indicate

that the peer dose not function properly. If a BGP router receives an ill-formatted or erroneous message, or if it fails to receive any message during a period longer than the hold time, it will report the error to its peer by sending a *notification* message, delete all routes associated with this connection, and then gracefully close the TCP connection [22].

2.2.2 Exchange routing information

To exchange routing information, two BGP routers first establish a peer session. After the session is established, the peers then exchange their full routing tables via a series of BGP messages. The routers are expected to memorize the paths provided by their peers. After the initial route exchanges, each router sends only incremental updates for new or modified routes.

Update messages can contain two types of reachability information: advertisements and withdrawals. An advertisement notifies its recipient of a new route to the destination prefix, whereas a withdrawal revokes a route it announced before. Beside the reachable information, an update message also contains a variable number of path attributes that describe the property of the route, including *AS path*, *next-hop*, *local preference*, *origin type*, and *multi-exit discriminator* (*MED*). The *AS path* attribute contains a list of the ASs the prefix has traversed. BGP uses the *AS path* for both loop detection and path selection. Upon receipt of a BGP update, each router examines the path vector and invalidates any route that includes the router's own AS number in the path. The *next-hop* attribute is the IP address of the router that must be used to reach the announced network. The *origin type* attribute identifies how the origin AS learned about the route: within the AS (static configuration), EGP (an obsolete exterior gateway protocol), or injection from another routing protocol. These origin types are known as IGP, EGP, and INCOMPLETE. The *multiple exit discriminator* attribute encourages the recipient to choose a particular exit point for sending traffic to the neighboring AS. A *local preference* attribute may be included in an iBGP message to help the recipient in ranking the paths learned from different routers within the same AS.

2.2.3 Route processing

There are two types of BGP peer sessions: external BGP (eBGP) for peers from different ASs and internal BGP (iBGP) for peers from the same AS. A BGP router may receive multiple paths to the same destination prefix from its eBGP and iBGP neighbors. Figure 2.2 shows the steps of BGP route processing. The router first applies import policies to filter out unwanted routes. For example, a BGP router may only accept advertisements with an AS path containing a set of trusted ASs. The router then invokes a decision process to select exactly one best route for each destination prefix by comparing the new routes to all other known routes to the same destination. The router applies a sequence of steps to narrow the set of candidate routes to a single choice. The best route will be installed in the router's forwarding table, while unselected routes are remembered for backup purposes.

Finally, the router applies export policies to manipulate attributes and decide whether to advertise the route to neighboring ASs. If the route is advertised, the router may modify some of the path attributes. It will at least add its own AS number to the *AS path*.

2.2.4 Route withdrawal

If a router receives a withdrawal, it first removes the invalidated route from its record. If the withdrawn route is currently the best route, the router looks into its backups and chooses a new preferred route or marks the prefix as unreachable. If the network is unreachable, a withdrawal must be sent to all peers who learned the route through earlier announcements.

2.2.5 Route reflection

In standard iBGP implementations, all BGP routers within the AS are fully meshed so that external routing information is redistributed among all routers within the AS. This type of implementation may present scaling problems when an AS has a large number of internal BGP speakers. Route reflection provides one way to decrease BGP control traffic, minimizing the number of update messages sent within an AS [2].

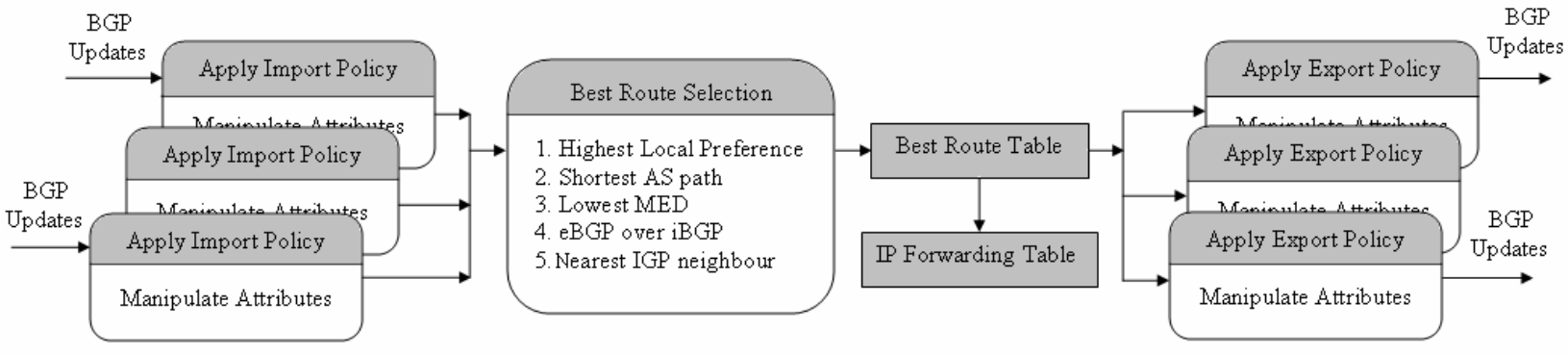


Figure 2.2: BGP route processing.

The route reflection concept is based on the idea of appointing a concentration router (*reflector*) to act as a focal point for iBGP sessions. In route reflection, BGP systems are arranged in *clusters*. Each cluster consists of at least one router that acts as a *route reflector*, along with any number of *client peers*. BGP peers outside the cluster are called *nonclient peers*. The route reflector reflects (redistributes) routing information to every client peer and to all nonclient peers. Because the route reflector redistributes routes within the cluster, the BGP routers in the cluster do not have to be fully meshed.

When the route reflector receives a route, it selects the best path. Then, if the route arrived from a nonclient peer, the route reflector sends the route to all client peers within the cluster. If the route arrived from a client peer, the route reflector sends it to all nonclient peers and to all client peers except the originator. During this process, none of the client peers sends routes to other client peers.

2.3 ns-2 network simulator

We implemented ns-BGP as an extension to the latest version of ns-2 network simulator (ns-2.27) [30]. ns-2 was developed at the ISI (University of Southern California). It was originally developed as an extension to the REAL network simulator. ns-2 is currently part of the collaborative *VINT* project involving USC/ISI, Xerox PARC, LBNL, and UC Berkeley. As one of the most popular discrete event network simulators [1], ns-2 supports simulation of TCP, routing, and multicast protocols over wired and wireless networks. ns-2 is written in both C++ and OTcl and employs an object-oriented paradigm. C++ is used for the low level implementation of packet oriented processing, where performance is important. OTcl is a scripting language used for higher level implementation, where flexibility is more important. A graphical animator *nam* is used to visualize simulation results.

2.4 BGP implementation in SSFNet

SSF.OS.BGP4 [33] is the BGP-4 model in the SSFNet [35] network simulation package. SSFNet is a Java-based simulator for modeling large communication networks. It includes a simulation kernel, an open source suite of network component models, a management suite, and a configuration language called Domain Modeling Language (DML). SSF.OS.BGP4, implemented in Java by Brian J. Premore [34], was designed with a purely object-oriented approach. A suite of tests is included in SSF.OS.BGP4 to ensure that the model complies with the BGP-4 specifications [34]. We ported to ns-2 the class hierarchy that was used to implement the BGP-4 model in SSF.OS.BGP4.

2.5 Related work in BGP implementation

OPNET [31], a commercial network simulator, also provides substantial support for BGP. However, differences between OPNET and ns-2 would have made porting the BGP model from OPNET to ns-2 rather difficult. GNU Zebra (written in C) is a free routing software package [19] that supports BGP [20] and other routing protocols. The Zebra BGP daemon has been recently ported to ns-2 [6]. Our project has been developed in parallel. We preferred the SSF.OS.BGP4 implementation because of its object oriented paradigm.

CHAPTER 3:
DESIGN AND IMPLEMENTATION OF NS-BGP

The ns-BGP classes are derived from the existing ns-2 class hierarchy. A brief introduction to the ns-2 unicast routing structure is first provided. Based on this structure, we describe the unicast routing structure of the ns-BGP model and its supported features.

3.1 ns-2 unicast routing structure

The ns-2 unicast routing structure consists of the forwarding and the control planes [30], as shown in Figure 3.1. Components of the forwarding plane are enclosed by an ellipse and components of the control plane are enclosed by a trapezoid.

The forwarding plane is responsible for classifying and forwarding packets to the destination nodes. It includes various types of connected classifiers and routing modules. Classifiers deliver the incoming packets either to the appropriate agent or to the outgoing link. A routing module manages a node's classifier and provides an interface to the control plane. Address classifier (*classifier_*) and port classifier (*dmux_*) are two types of classifiers (trapezoids) in an ns-2 unicast node. A *classifier_* examines the destination address of an arriving packet and forwards the packet to the *dmux_* if the node is the packet's destination. Otherwise, the *classifier_* sends the packet to a downstream node. The *dmux_* forwards the packet to an agent corresponding to the packet's destination port number.

The control plane handles route computation and creation and the maintenance of routing tables. It also implements specific routing algorithms. The components of the control plane are *route logic*, *route object*, *route peer*, and *routing protocol*. The *route logic* is the centrally created and maintained routing table. *Route objects* are employed only in simulations of dynamic routing. The *route object* associated with a node acts as a coordinator for the node's routing instances. A

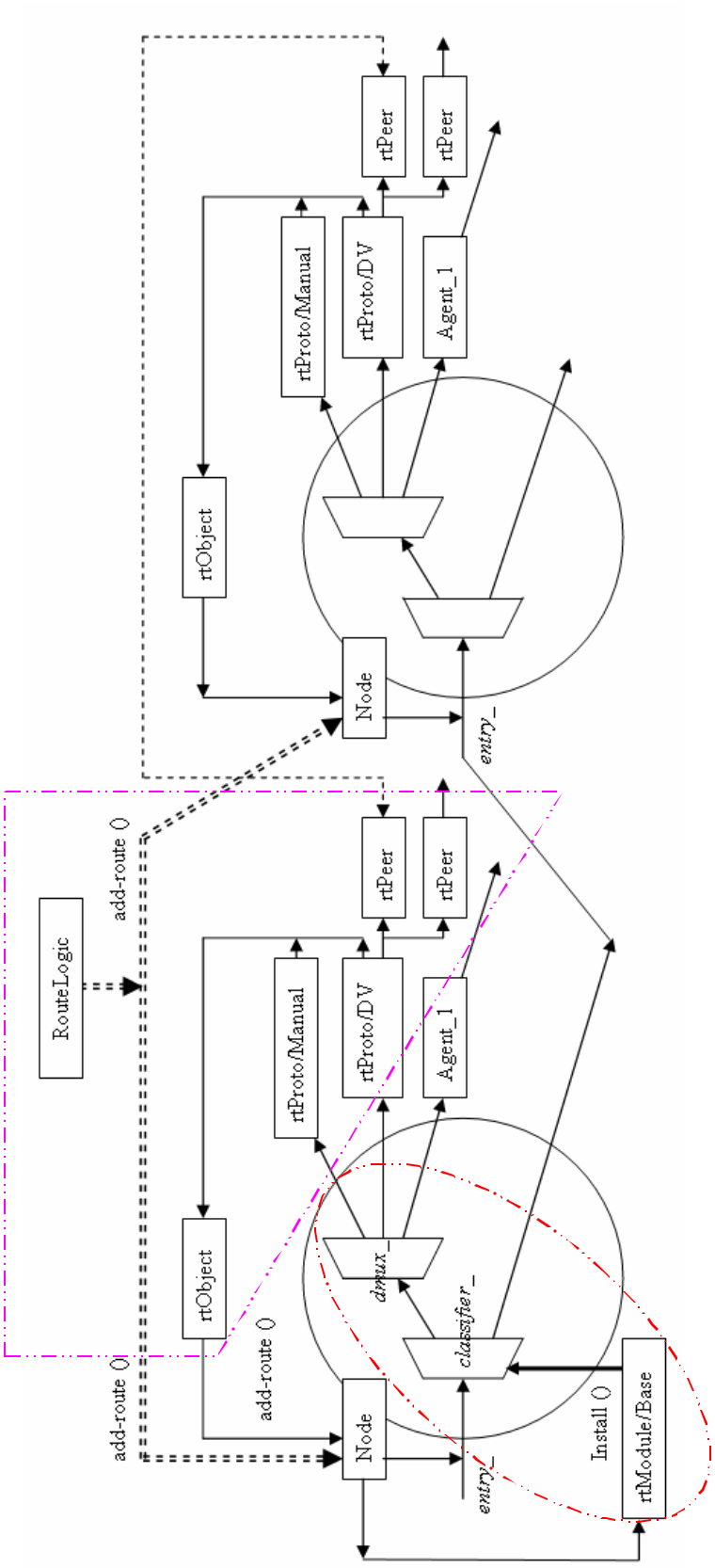


Figure 3.1: ns-2 unicast routing structure.

route peer object acts as a container object used by the *routing protocol*: it stores the address of the peer agent, the metric, and the preference for each route advertised by the peer. *Routing protocols* implement specific routing algorithms, such as distance vector and link state algorithms [22].

3.2 ns-BGP unicast routing structure

The ns-BGP node is based on the existing ns-2 unicast node and the SSF.OS.BGP4 model from SSFNet. We converted the SSF.OS.BGP4 model to ns-2 and added the socket layer as well as IPv4 addressing and packet forwarding schemes.

In order to provide socket support and at the same time maintain the structure of SSF.OS.BGP4, we also ported to ns-2 *TcpSocket*, the socket layer implementation of SSFNet. In order to support the IPv4 addressing and packet forwarding, the basic address classifier was replaced with a new address classifier named *IPv4Classifier*. To support user data transmission, we modified *FullTcpAgent* [30], the TCP agent for *TcpSocket*.

Figure 3.2 shows the unicast structure of ns-BGP. The address classifier *classifier_* 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 [37]. *BGP_Timer* provides support for the BGP timing features (timers).

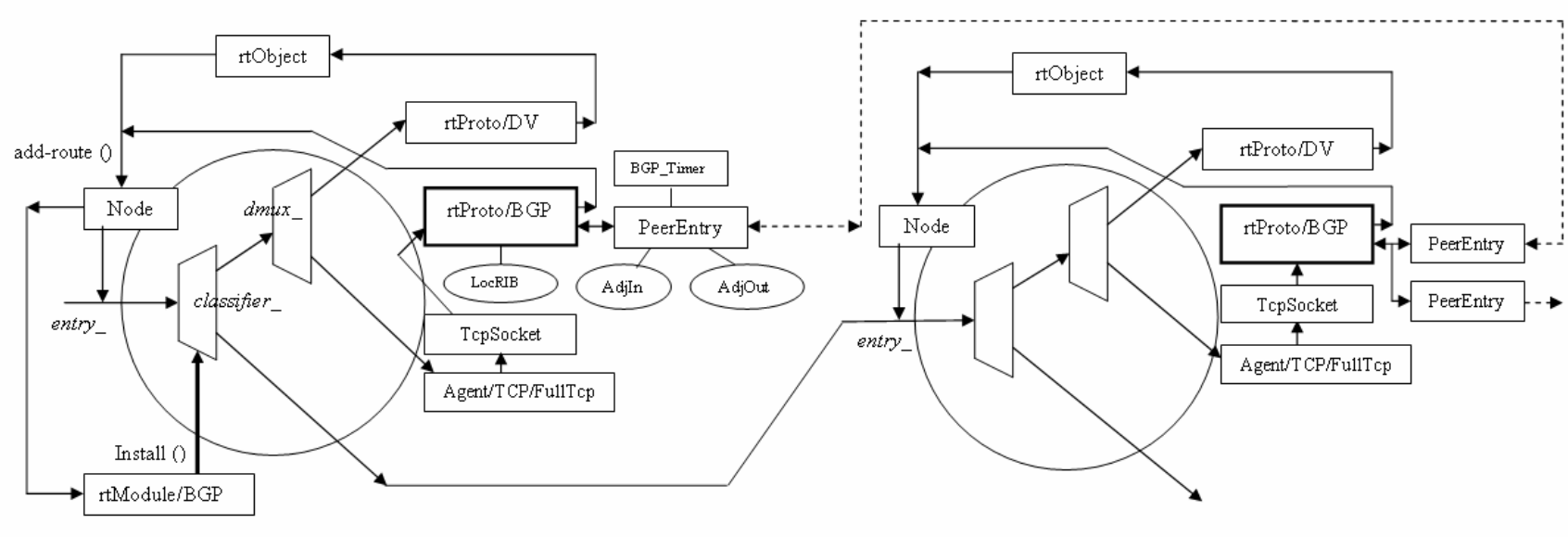


Figure 3.2: Unicast routing structure of ns-BGP.

The five important classes of ns-BGP are *TcpSocket*, *IPv4Classifer*, *rtModule/BGP*, *rtProtoBGP*, and *BGP_Timer*.

3.2.1 TcpSockets

A socket is an *Application Programming Interface* (API) used in network communications. Socket applications treat network connections as UNIX file descriptors. Similar to files, communication endpoints can be written to, read from, or deleted.

The *TcpSocket* class is an implementation of the sockets API, similar to UNIX implementations. Its most important functions are: *bind*, *listen*, *connect*, *close*, *read*, and *write*. The *TcpSocket* interface involved implementation of blocking calls using the *Continuation* caller, a class consisting of two callback functions: *Success* and *Failure*. Necessary data structures and classes, such as queue classes that store the data and a *TcpData* class that contains the transmitted user data, were also added to ns-2. The *FullTcpAgent* was modified to send and receive data packets containing user data and to inform the corresponding *TcpSocket* of changes in the TCP status.

We implemented blocking calls using *Continuation* caller, which is a container of two callback functions: *Success* and *Failure*. Blocking function calls are widely used in network programming environments. For example, if a user function is about to send a data packet using socket service, the caller will be blocked until the socket connection succeeds (returns *Success* to calling function) or fails (returns *Failure* to calling function). Network performance is often unpredictable, due to traffic congestion for instance. In this case, *Continuation* caller can synchronize the calling function and the called function.

We added the following data structures and classes to support TcpSocket capable of user data transmission: *SendQueue* class that stores the data requested to be sent by sender TCP agent, *ReceiveQueue* class that stores the received data from the sender, and *TcpData* class that contains the transmitted user data.

3.2.2 IPv4Classifier

The *IPv4Classifier* is derived from *Classifier*. It is implemented as one of the ns-2 dual classes (in both C++ and OTcl). The *IPv4Classifier* uses *map* from the C++ Standard Template Library to store and search the routing table. To classify an incoming packet, the *IPv4Classifier* examines the packet's destination address. It then matches this address in the routing table of the classifier in order to find the route with the longest prefix match.

3.2.3 rtModule/BGP

The *rtModule/BGP*, a new *routing module* implemented in Tcl, provides a registration interface. When a node is created, active route models must register with the node. This registration replaces the existing classifier objects in the node.

3.2.4 rtProtoBGP

The *rtProtoBGP* class (*Agent/rtProto/BGP*) is implemented as an ns-2 dual class. An instance of this class implements BGP-4 in a node. This new *routing protocol* performs all the BGP operations: establishing BGP peer sessions, learning multiple paths via internal and external BGP speakers, selecting the best path and storing it into the IP forwarding table (*IPv4Classifier*), and managing the BGP finite state machine.

3.2.5 BGP_Timer

BGP_Timer is derived from the ns-2 *TimerHandler* class. It provides support for the BGP timing features, such as the *start-up* timer, *keep-alive* timer, *hold* timer, and the *Minimum Route Advertisement Interval* (MRAI) timer. During the auto-configuration process, a *start-up* timer is scheduled for each BGP agents. When the *start-up* timer expires, it will bring up the BGP agent (rtProtoBGP) to try to establish peer connections with its BGP neighbors. When a *keep-alive* timer expires, it will trigger the BGP agent to send out a keep-alive message to its peer. Expiration of a *hold* timer indicates the failure of a BGP agent to receive a message during the hold time interval from a peer. In this case, the BGP agent will report the error to its peer by

sending a *notification* message. The MRAI timers are used to space out by *M* seconds (default value 30) consecutive updates for the same destination.

3.3 Supported features

The implementation of the ns-BGP is compliant with the BGP-4 specification RFC 1771 [37]. It includes several optional protocol extensions and additional experimental features. We implemented experimental features: sender-side loop detection, withdrawal rate limiting, unjittered *Minimum Route Advertisement Interval* timer, and per-peer and per-destination rate limiting. Implemented optional features are Multiple Exit Discriminator, Aggregator, Community, Originator ID, and Cluster List path attributes. We have also implemented route reflection. Nevertheless, the current implementation does not support the multiprotocol extensions for BGP-4 [3].

CHAPTER 4:
VALIDATION TESTS

SSF.OS.BGP4 included a suite of tests that ensured that the SSF.OS.BGP4 model complies with the BGP-4 specifications, including BGP-4 features such as: basic peer session maintenance (keep-alive and hold timer operation), route advertisement and withdrawal, route selection, internal BGP (iBGP), and route reflection [34]. We implemented most of these validation tests in ns-2 and tested the same network topologies as employed in the SSFNet validation tests [33]. We also introduced a new validation test for route reflection [2]. The test scripts used for validation tests are included in Appendix A.

4.1 Route selection validation test

This test checks whether a BGP speaker chooses routes properly when there is more than one path to a particular destination. BGP bases its decision on the values of path attributes. Following is an ordered list of rules used to determine the best path (also shown in Figure 2.2):

- prefer the path with the largest *Local Preference*
- prefer the path with the shortest *AS path*
- prefer the path with the lowest *multiple exit discriminator* (*MED*)
- prefer external (eBGP) over internal (iBGP) paths
- prefer the path with the lowest IGP metric to the BGP next hop.

Since the *Local Preference* path attribute is not considered in this validation test, the best route will be the route with the shortest *AS path*.

4.1.1 Network topology

Figure 4.1 shows the network topology used for the simulation of route selection. The network consists of three ASs. Each AS contains one node: AS 0, AS 1, and AS 2 contain node 0, 1, and 2, respectively. The IP address of each node is shown in Table 4.1. The addressing scheme is: 10.(AS number).(node number).1.

Table 4.1: IP addresses used in the route selection validation test.

node 0	10.0.0.1
node 1	10.1.1.1
node 2	10.2.2.1

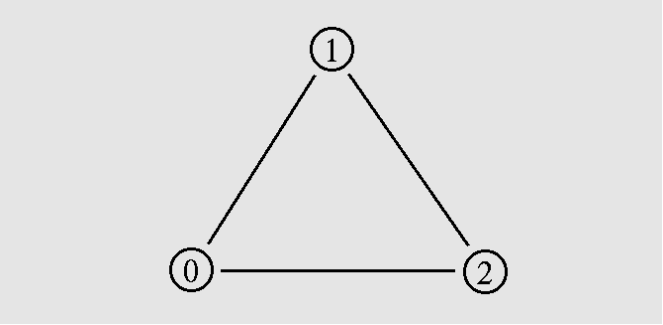


Figure 4.1: Network topology used in the route selection validation test.

4.1.2 BGP configuration and event scheduling

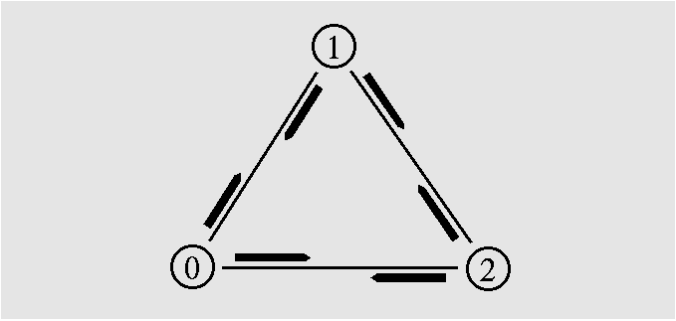
BGP agents were configured for each of the three nodes (0, 1, and 2). They are fully meshed using external BGP (eBGP) connections. At 0.25 s, the BGP agent in node 0 advertises a new route for IP address 10.0.0.0/24. At 39.0 s, ns-2 displays the all routing tables from BGP agents. The simulation terminates at 40.0 s.

4.1.3 Simulation results

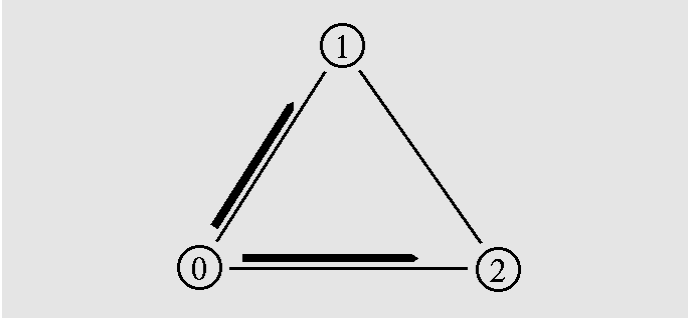
The simulation sequence of events is shown in Table 4.2. Simulation results displayed by *nam* are shown in Figure 4.2.

Table 4.2: Sequence of simulation events.

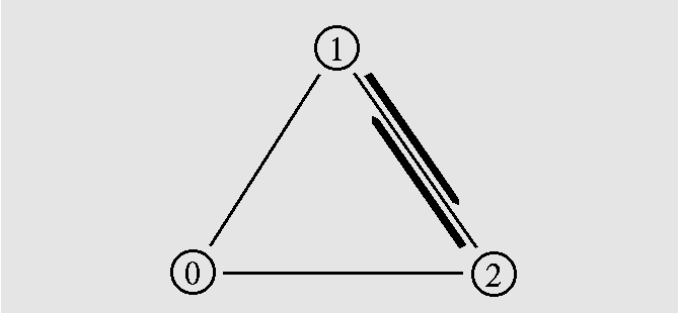
0.0503 s	Figure 4.2(a): TCP SYN segments are exchanged between BGP peers, establishing the underlying TCP connections.
0.2507 s	Figure 4.2(b): Node 0 originates an update message advertising to node 1 and node 2 the route for network 10.0.0.0/24.
0.2525 s	Figure 4.2(c): Nodes 1 and 2 propagate the route advertisement to each other.



(a) Establishing TCP connections (0.0503 s).



(b) Node 0 advertises a route (0.2507 s).



(c) Nodes 1 and 2 propagate the route (0.2525 s).

Figure 4.2: Snapshots of simulation results in the route selection test.

During the simulation run, node 1 and node 2 both learned two routes for the IP address 10.0.0.0/24 originated by node 0. One of these two routes is received directly from node 0 (Figure 4.2(b)), while the other route is exchanged between nodes 1 and 2 (Figure 42(c)). We first consider node 1. The *AS path* of the route that node 1 received directly from node 0 contains only AS 0, thus, the length of this route's *AS path* is 1. The *AS path* of the route that received from node 2 contains AS 0 and AS 2, thus, the *AS path* length is 2. According to the rules of the best route selection, node 1 should favor the route that it received directly from node 0 over the route received from node 2. Node 2 followed similar decision processes.

The routing tables from the BGP agents at 39.0 s show (status codes are: * valid, > best, i – internal) the proper choices of the best route in three nodes:

BGP routing table of node0
BGP table version is 2, local router ID is 10.0.0.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	self	-	-	-	

BGP routing table of node1 ← node name
BGP table version is 1, local router ID is 10.1.1.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	10.0.0.1	-	-	-	0

← AS path
← destination IP address

BGP routing table of node2
BGP table version is 1, local router ID is 10.2.2.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	10.0.0.1	-	-	-	0

4.2 Reconnection validation test

This test checks the ability of a BGP speaker to re-establish a peer session with a former peer. In this test, a BGP speaker establishes two peer sessions, but the session with one of them is later broken. The two BGP speakers that are disconnected then attempt to re-establish a session.

4.2.1 Network topology

Figure 4.3 shows the network topology used for simulation of route reconnection. The network consists of three ASs. Each AS contains one node: AS 0, AS 1, and AS 2 contain nodes 0, 1, and 2, respectively. The IP address of each node is shown in Table 4.3. We used the same addressing scheme as in Section 4.1: 10.(AS number).(node number).1.

Table 4.3: IP addresses used in the reconnection validation test.

node 0	10.0.0.1
node 1	10.1.1.1
node 2	10.2.2.1

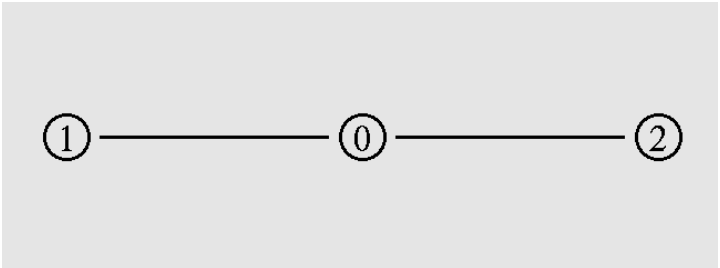


Figure 4.3: Network topology in the reconnection validation test.

4.2.2 BGP configuration and event scheduling

BGP agents are configured for each of the three nodes (0, 1, and 2). External BGP (eBGP) connections exist between nodes 0 and 1, as well as nodes 0 and 2. For nodes 0 and 2, the

values for *hold timer* and *keep-alive timer* intervals of BGP agents are default values (hold time: 90 s, keep-alive: 30 s) suggested in RFC 1771 [37]. In order to observe the reconnection behavior of ns-BGP, we increase the *keep-alive timer* interval of the BGP agent in node 1 to 200 s. By doing so, BGP agent in node 0 will not receive any keep-alive message before its *hold* timer expires, which will trigger the session re-establishment.

At 0.25 s, the BGP agent in node 0 advertises a new route for IP address 10.0.0.0/24. At 0.35 s, the BGP agent in node 1 advertises a new route for IP address 10.1.1.0/24. At 0.45 s, the BGP agent in node 2 advertises a route for IP address 10.2.2.0/24. At 28 s, 90.38 s, and 119.0 s, ns-2 displays all routing tables from BGP agents. The simulation terminates at 120.0 s.

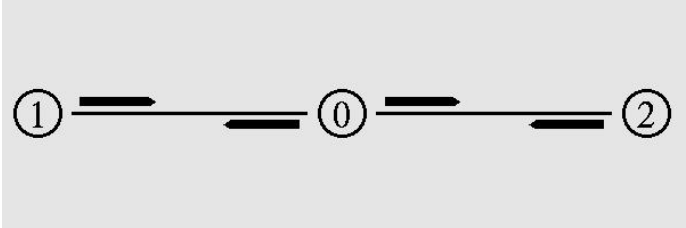
4.2.3 Simulation results

The simulation sequence of events is shown in Table 4.4. Simulation results displayed by *nam* are shown in Figure 4.4.

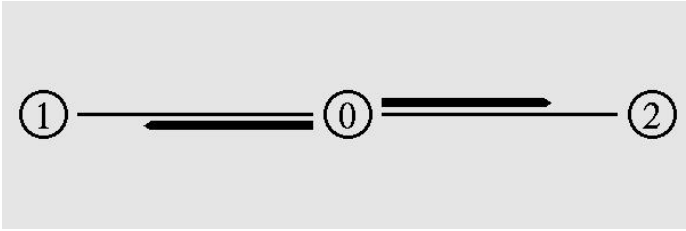
Table 4.4: Sequence of simulation events.

0.0503 s	Figure 4.4(a): TCP SYN segments are exchanged between BGP peers, establishing the underlying TCP connections.
0.2507 s	Figure 4.4(b): node 0 originates an update message advertising to both nodes 1 and 2 the route for network 10.0.0.0/24.
0.3507 s	Figure 4.4(c): node 1 originates an update message advertising to node 0 the route for network 10.1.1.0/24.
0.3523 s	Figure 4.4(d): node 0 propagates to node 2 the route for network 10.1.1.0/24.
92.2034 s	Figure 4.4(e): in node 0, the hold timer for the peer session with node 1 expires. Node 0 sends a notification message to node 1 informing it of the error and sends

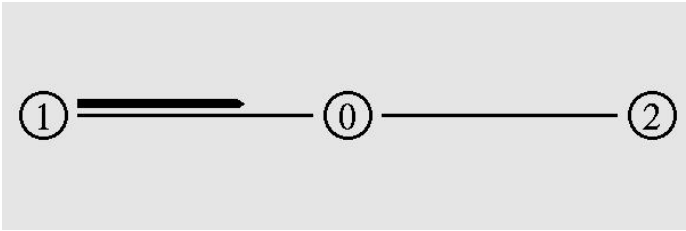
	a route withdrawal to node 2 revoking the route for network 10.1.1.0/24.
92.2534 s	Figure 4.4(f): node 0 re-establishing the underlying TCP connection with node 1.
92.4021 s	Figure 4.4(g): after the session re-establishment, nodes 0 and 1 exchange routing information.
92.4038 s	Figure 4.4(h): node 0 propagates to node 2 the route for network 10.1.1.0/24.



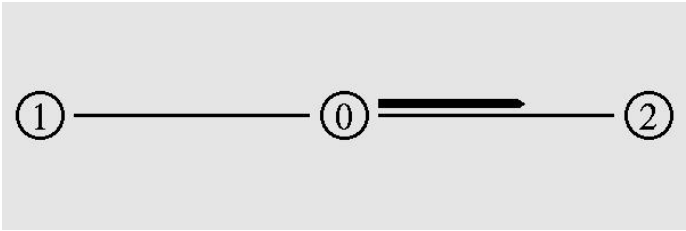
(a) Establishing TCP connections (0.0503 s).



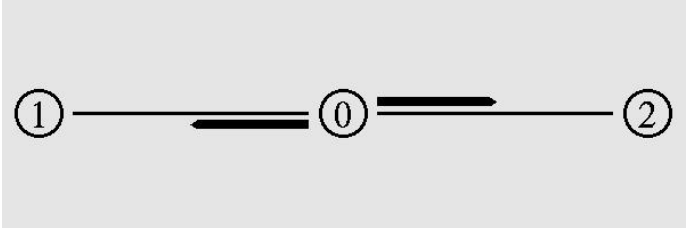
(b) Node 0 originates a route to nodes 1 and 2 (0.2507 s).



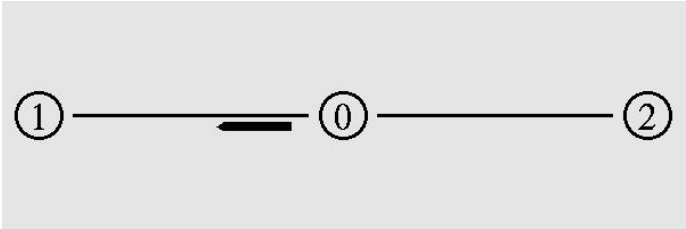
(c) Node 1 originates a route to node 0 (0.3507 s).



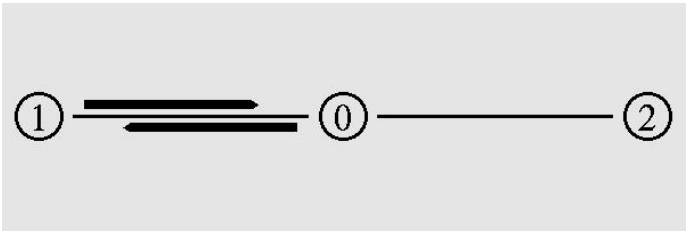
(d) Node 0 propagates the route to node 2 (0.3523 s).



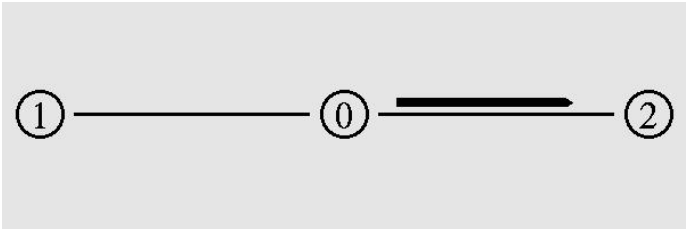
(e) Node 0 sends a notification to node 1 and a withdrawal to node 2 (92.2034 s).



(f) Node 0 re-establishing TCP connection with node 1 (92.2534 s).



(g) Node 0 and 1 exchange routing information (92.4021 s).



(h) Node 0 propagates the route to node 2 (92.4038 s)

Figure 4.4: Snapshots of *nam* simulation results of reconnection test.

The routing tables from all BGP agents at 28 s, 90.38 s, and 119 s, respectively, illustrate that every BGP agents learned the routes announced by other BGP agents by 28 s. At 90.38 s, due to the failure of the peer session between nodes 0 and 1, nodes 0 and 2 already removed the route for network 10.1.1.0/24 that was originated by node 1. Node 1 also deleted the routes for networks 10.0.0.0/24 and 10.2.2.0/24, which it learned from node 0. After the re-establishment of their peer session, nodes 0 and 1 exchanged all the routing information they had and the routing tables converged for the second time at 119 s.

time: 28
dump routing tables in all BGP agents:

BGP routing table of node0
BGP table version is 10, local router ID is 10.0.0.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	self	-	-	-	
*>	10.1.1.0/24	10.1.1.1	-	-	-	1
*>	10.2.2.0/24	10.2.2.1	-	-	-	2

BGP routing table of node1
BGP table version is 16, local router ID is 10.1.1.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	10.0.0.1	-	-	-	0
*>	10.1.1.0/24	self	-	-	-	
*>	10.2.2.0/24	10.0.0.1	-	-	-	0 2

BGP routing table of node2
BGP table version is 10, local router ID is 10.2.2.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	10.0.0.1	-	-	-	0
*>	10.1.1.0/24	10.0.0.1	-	-	-	0 1
*>	10.2.2.0/24	self	-	-	-	

time: 90.38
dump routing tables in all BGP agents:

BGP routing table of node0
BGP table version is 23, local router ID is 10.0.0.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	self	-	-	-	
*>	10.2.2.0/24	10.2.2.1	-	-	-	2

BGP routing table of node1
BGP table version is 42, local router ID is 10.1.1.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.1.0/24	self	-	-	-	

BGP routing table of node2
BGP table version is 23, local router ID is 10.2.2.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	10.0.0.1	-	-	-	0
*>	10.2.2.0/24	self	-	-	-	

time: 119
dump routing tables in all BGP agents:

BGP routing table of node0
BGP table version is 30, local router ID is 10.0.0.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	self	-	-	-	
*>	10.1.1.0/24	10.1.1.1	-	-	-	1
*>	10.2.2.0/24	10.2.2.1	-	-	-	2

BGP routing table of node1
BGP table version is 56, local router ID is 10.1.1.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	10.0.0.1	-	-	-	0
*>	10.1.1.0/24	self	-	-	-	
*>	10.2.2.0/24	10.0.0.1	-	-	-	0 2

BGP routing table of node2
BGP table version is 30, local router ID is 10.2.2.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.0.0.0/24	10.0.0.1	-	-	-	0
*>	10.1.1.0/24	10.0.0.1	-	-	-	0 1
*>	10.2.2.0/24	self	-	-	-	

4.3 Route reflection validation test

Implementing route reflection can help address the scalability problem in iBGP connections. However, without a full BGP mesh inside the AS, redundancy and reliability become an issue. If a route reflector fails, its clients will become isolated. Redundancy requires the existence of multiple route reflectors in a cluster where clients can simultaneously peer with multiple routers. If one route reflector fails, the other(s) should still be available. The goal of this simulation test is to validate the behavior of multiple reflectors inside a BGP cluster [21].

4.3.1 Network topology

Figure 4.5 shows the network topology employed for simulation of route reflection. The network consists of three ASs: AS 0 containing eight nodes (0 through 7), AS 1 containing two nodes (8 and 10), and AS 2 with a single node (9). The address of each node is shown in Table 4.5. The addressing scheme is: 10.(AS number).(node number).1.

Table 4.5: IP addresses used in the route reflection validation test.

Nodes: 0 through 7	10.0.0.1 though 10.0.7.1
Nodes: 8 and 10	10.1.8.1 and 10.1.10.1
Node: 9	10.2.9.1

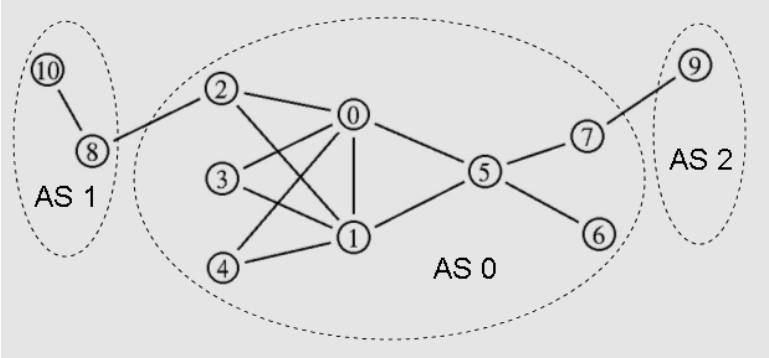


Figure 4.5: Network topology employed in the route reflection validation test.

4.3.2 BGP configuration

AS 0 contains two clusters. The first cluster contains two reflectors: nodes 0 and 1. The reflection clients of nodes 0 and 1 are nodes 2, 3, and 4. The second cluster has one reflector node (5), with nodes 6 and 7 as its clients. The three reflectors (nodes 0, 1, and 5) are fully connected via iBGP sessions. External BGP (eBGP) peer sessions exist between nodes 2 and 8, as well as between nodes 7 and 9.

4.3.3 Traffic source and event scheduling

A constant bit rate (CBR) traffic source attached to node 4 employs UDP as its transport protocol. It sends segments of 20 bytes every millisecond to the IP address of node 10 (10.1.10.1). The traffic source begins sending UDP segments at 0.23 s and stops sending them at 20.0 s. At 0.25 s, the BGP agent in node 8 sends a route advertisement for a network 10.1.10.0/24 that is within its AS (AS 1). At 0.35 s, the BGP agent in node 9 sends a route advertisement for

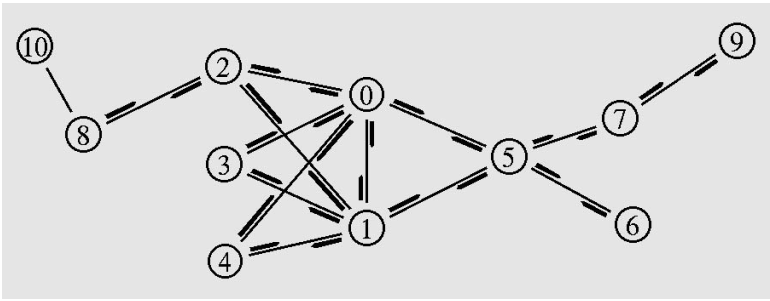
network 10.2.9.0/24 (AS 2). At 39.0 s, ns-2 displays all routing tables for BGP agents. The simulation terminates at 40.0 s.

4.3.4 Simulation results

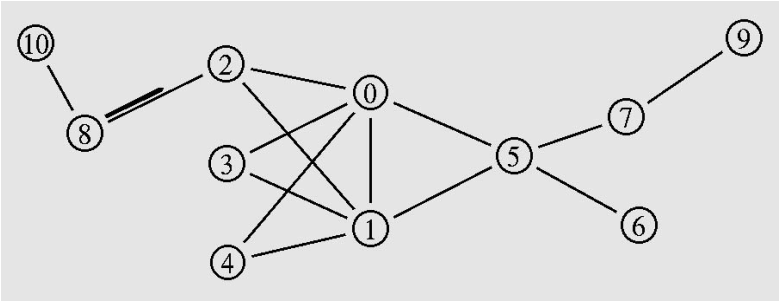
The simulation sequence of events is shown in Table 4.6. Simulation results displayed by *nam* are shown in Figures 4.6(a)–(g).

Table 4.6: Sequence of simulation events.

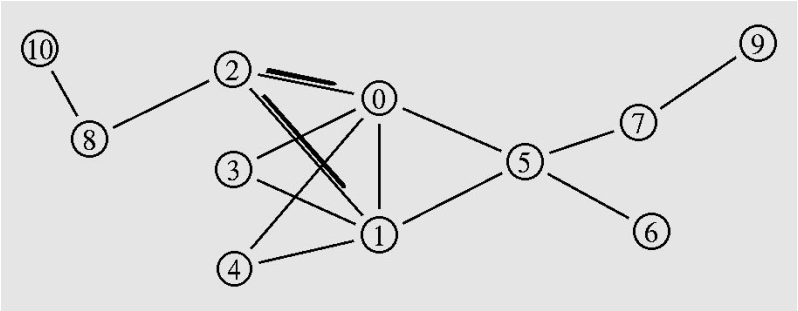
0.0503 s	Figure 4.6(a): TCP SYN segments are exchanged between BGP peers, establishing the underlying TCP connections.
0.2505 s	Figure 4.6(b): node 8 originates an update message advertising the route for network 10.1.10.0/24.
0.2525 s	Figure 4.6(c): node 2 propagates the route advertisement to nodes 0 and 1.
0.2561 s	Figure 4.6(d): route reflectors (nodes 0 and 1) reflect the route advertisement to their clients (nodes 3 and 4) and to their iBGP peers.
0.2568 s	Figure 4.6(e): node 5 reflects the route advertisement to its clients (nodes 6 and 7). Because node 4 now knows the route to network 10.1.10.0/24, the UDP segment will be forwarded to node 10. Before node 4 knowing this route, the UDP segments sending out by the traffic source are dropped at node 4.
0.2578 s	Figure 4.6(f): the second UDP segment is sent to the destination (node 10). Node 7 propagates the route advertisement to node 9.
0.2580 s	Figure 4.6(g): UDP segments are delivered to node 10.



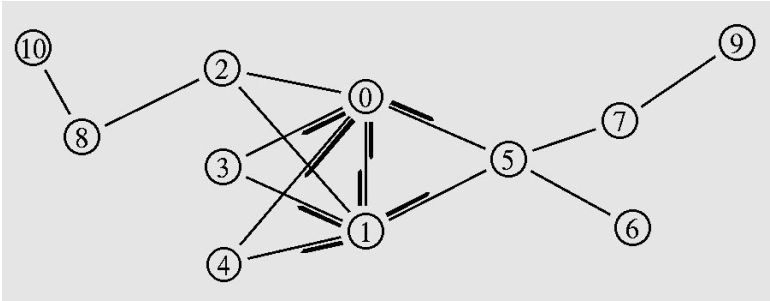
(a) Establishing TCP connections (0.0503 s).



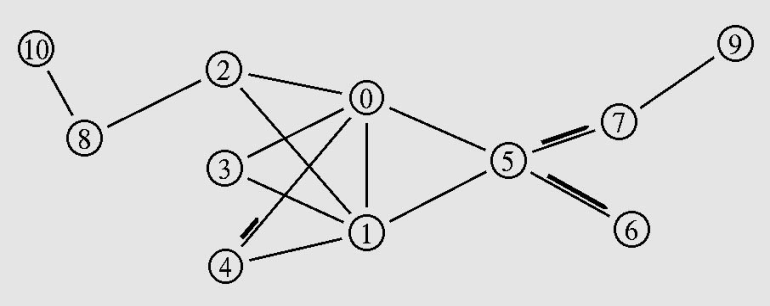
(b) Node 8 originates a route (0.2505 s).



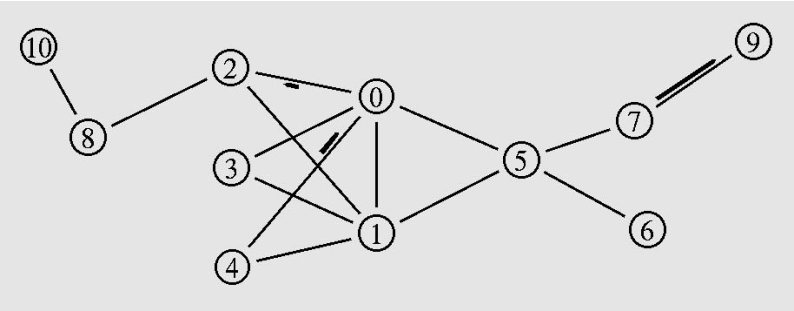
(c) Node 2 propagates the route to nodes 0 and 1 (0.2525 s).



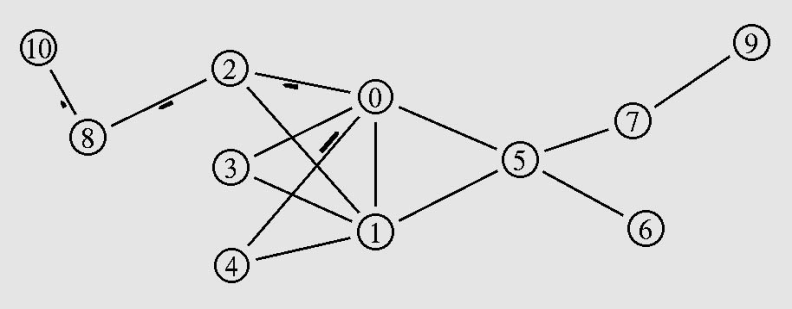
(d) Nodes 0 and 1 reflect the routes to nodes 3 and 4 (0.2561 s).



(e) Node 4 sends a UDP segment to node 10. Node 5 reflects the route to nodes 6 and 7 (0.2568 s).



(f) Node 4 sends the second UDP segment. Node 7 propagates the route to node 9 (0.2578 s).



(g) Four UDP segments are being delivered to node 10 (0.2580 s).

Figure 4.6: Snapshots of simulation results for route reflection test.

By the end of the simulation run, every BGP node knows routes to 10.1.10.0/24 and 10.2.9.0/24. Routing tables for BGP agents at 39.0 s are:

BGP routing table of node0
BGP table version is 2, local router ID is 10.0.0.1
*Status codes: * valid, > best, i - internal.*

	Network	*Next Hop*	*Metric*	*LocPrf*	*Weight*	*Path*
*>	*10.1.10.0/24*	*10.0.2.1*	-	-	- 1	*i*
*>	*10.2.9.0/24*	*10.0.7.1*	-	-	- 2	*i*

BGP routing table of node1

BGP table version is 2, local router ID is 10.0.1.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.0.2.1	-	-	- 1	i
*>	10.2.9.0/24	10.0.7.1	-	-	- 2	i

BGP routing table of node2
BGP table version is 4, local router ID is 10.0.2.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.1.8.1	-	-	- 1	
*>	10.2.9.0/24	10.0.7.1	-	-	- 2	i

BGP routing table of node3
BGP table version is 4, local router ID is 10.0.3.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.0.2.1	-	-	- 1	i
*>	10.2.9.0/24	10.0.7.1	-	-	- 2	i

BGP routing table of node4
BGP table version is 4, local router ID is 10.0.4.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.0.2.1	-	-	- 1	i
*>	10.2.9.0/24	10.0.7.1	-	-	- 2	i

BGP routing table of node5
BGP table version is 2, local router ID is 10.0.5.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.0.2.1	-	-	- 1	i
*>	10.2.9.0/24	10.0.7.1	-	-	- 2	i

BGP routing table of node6
BGP table version is 2, local router ID is 10.0.6.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.0.2.1	-	-	- 1	i
*>	10.2.9.0/24	10.0.7.1	-	-	- 2	i

BGP routing table of node7
BGP table version is 2, local router ID is 10.0.7.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.0.2.1	-	-	- 1	i
*>	10.2.9.0/24	10.2.9.1	-	-	- 2	

BGP routing table of node8
BGP table version is 3, local router ID is 10.1.8.1
*Status codes: * valid, > best, i - internal.*

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	self	-		-	-
*>	10.2.9.0/24	10.0.2.1	-		-	- 0 2

BGP routing table of node9
BGP table version is 3, local router ID is 10.2.9.1
Status codes: * valid, > best, i - internal.

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.10.0/24	10.0.7.1	-		-	- 0 1
*>	10.2.9.0/24	self	-		-	-

CHAPTER 5:
MODEL SCALABILITY

As the size and complexity of simulated networks grow, it is important to address the scalability properties of simulation models. Such properties include execution speed and memory requirements of a simulation experiment [29]. The ns-BGP model should scale both with respect to the number of peer sessions and the size of routing tables. Our simulation experiments were performed on a 1.6 GHz Intel Xeon host with 2 GBytes of memory and a RedHat Linux 9.0 operating system.

5.1 Model Configuration

In validation tests, we verified the ns-BGP model using three small scale networks. In contrast, the experiments performed in the scalability analysis are quite larger in terms of the network topology and its size. Some experiments used in the scalability analysis contain up to 10,000 nodes and 10,000 peer sessions. Experiments with such large scale networks require further configuration of the ns-BGP model.

5.1.1 Topology families

Our scalability analysis is based on several simpler topologies that are similar to subgraphs of the Internet's AS graph. BGP's behavior of these subgraphs is expected to be an important indicator of its behavior in more general topologies. Finding a closer behavioral relationship between the individual components and the Internet topologies would require further study [34].

We first introduce several definitions to simplify the explanation of the experiments. A *simple AS* is an AS containing only one BGP router and no host [34]. Every AS in the experiments of scalability analysis is a *simple* AS. A topology of size *n* has *n simple* ASs and,

therefore, n BGP routers referred as R_0, R_1, …, R_{n-1}. eBGP connections are established between each router R_i and its neighbors that are connected to R_i by physical links.

The first family of experiment is performed on a *line topology*. A line topology of size n is a topology with n *simple* ASs with routers R_0, R_1, …, R_{n-1}, such that there is link between R_i and R_{i+1} for $i = 0, 1, \ldots, n\text{-}2$. Figure 5.1 shows a *line topology* of size 6.

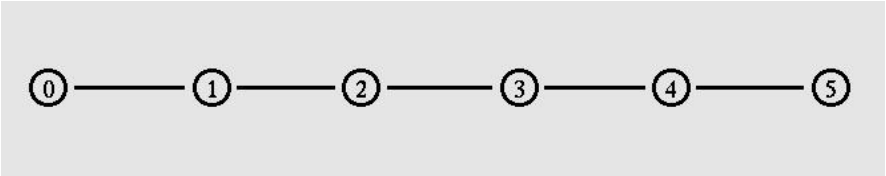


Figure 5.1 A *line topology* of size 6.

A *ring topology* of size n is a topology with n *simple* ASs with routers R_0, R_1, …, R_{n-1}, such that there is a link between R_i and R_{i+1} for $i = 0, 1, \ldots, n\text{-}2$, as well as a link between R_0 and R_{n-1}. Figure 5.2 illustrates a *ring topology* of size 6.

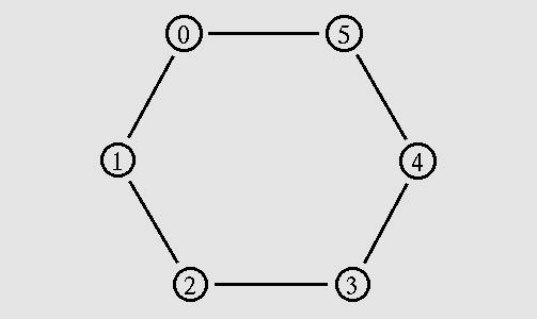


Figure 5.2 A *ring topology* of size 6.

A *binary tree topology* of size n ($n = 2^m\text{-}1$, where m is the tree height) is a topology with n *simple* ASs with routers R_0, R_1, …, R_{n-1}, such that there are links between R_i and R_{2i}, as well as R_i and R_{2i+1} for $i = 0, 1, \ldots, 2^{m-1}\text{-}2$. Figure 5.3 illustrates a *binary tree topology* of size 15.

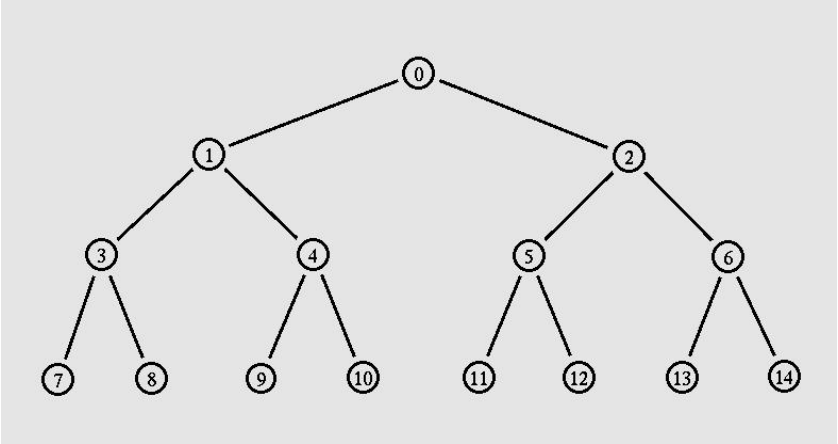


Figure 5.3 A *binary tree topology* of size 15.

A *grid topology* of size n ($n = m^2$, where m is the grid length) to be a topology in which there are n *simple* ASs with routers R_0, R_1, …, R_{n-1}, such that there is link between R_x and R_y (where $x = im + j$, $y = km + l$ and $\{|i-k|, |j-l|\} = \{0,1\}$) for $x,y = 0, 1, \ldots, n-1$, $x ? y$, Figure 5.4 illustrates a *grid topology* of size 16.

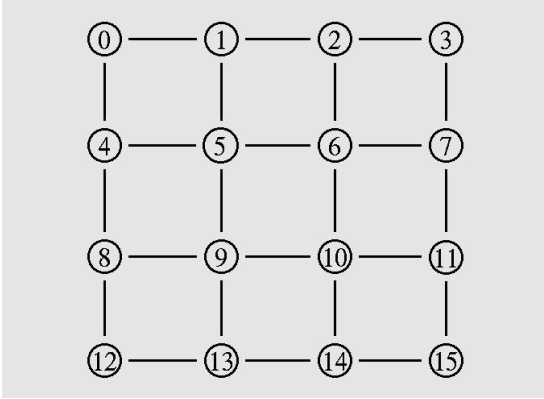


Figure 5.4 A *grid topology* of size 16.

A *clique topology* of size n is a topology with n *simple* ASs with routers R_0, R_1, …, R_{n-1}, such that there is link between R_i and R_j, for $i, j = 0, 1, \ldots, n-1$, $i ? j$. Figure 5.5 illustrates a *clique topology* of size 6.

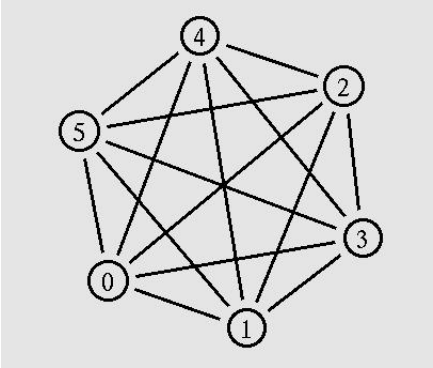


Figure 5.5 A *clique topology* of size 6.

5.1.2 Experiment parameters

BGP's behavior can be changed by a set of parameters. The default values for the parameters that are applied in all experiments are list in Table 5.1:

Table 5.1: Default values of parameters used in experiments.

Parameter description	**Default**
hold-time interval	90 s
keep-alive interval	30 s
MRAI	30 s
jitter keep-alive interval	yes
jitter MRAI	yes
jitter start-up timer interval	yes

The first three parameters are interval values for the BGP timers. They are set to the default values suggested in [37]. We introduce jitter factors to the keep-alive interval, the MRAI,

and the start-up timer interval to avoid performance degradation of the ns-2 scheduler, as described in Section 5.1.4 with more details.

5.1.3 ns-BGP simulation phases

Each BGP simulation described in this chapter contains seven phases. An ns-2 simulator instance is created during phase 1. The execution time and memory usage of this phase are identical for every simulation experiment and are small enough to be ignored. Nodes and links are created in phases 2 and 3, respectively. All BGP agents are enabled to be auto-configured by the ns-BGP model during phase 4. In phase 5, initialization of the simulator, such as the creation of the central routing table *(Route Logic)*, is performed. The time and memory usage of phase 4 and 5 are also negligible and are ignored. During phase 6, each BGP agent establishes peer sessions with its neighbors. In phase 7, after all peer sessions are established successfully, BGP messages (*keep-alive* and/or *update* messages) are exchanged between the peering BGP agents. A sample Otcl script containing information of simulation phases is given in Appendix B.

5.1.4 ns-2 Calendar Scheduler

ns-2 is an event-driven simulator. The scheduler runs by selecting the next earliest event, executing it to completion, and returning to execute the next event [30]. The *Calendar Queue* data structure used by the default *Calendar Scheduler* is described by Brown [7]. It is a priority queue specially designed for the event set problem.

Performance of the *Calendar Scheduler* is affected by the distribution of event times. As the network topology grows, more synchronous BGP agents schedule events for the same time instance. Large number of events scheduled at the few time instances can cause the *scheduling time* (cumulative execution time of the scheduler) to increase exponentially, as shown in Figure 5.6. In order to reduce the synchronization, we scatter the events by introducing random jitter factors to the BGP *start-up*, *keep-alive*, and *MRAI* timers. Figure 5.7 shows the effect of jittered timers on the distribution of event times. While the jittered *scheduling times* no longer increase

exponentially with the number of peer sessions, they are affected by the introduced random factors as shown in Figure 5.8.

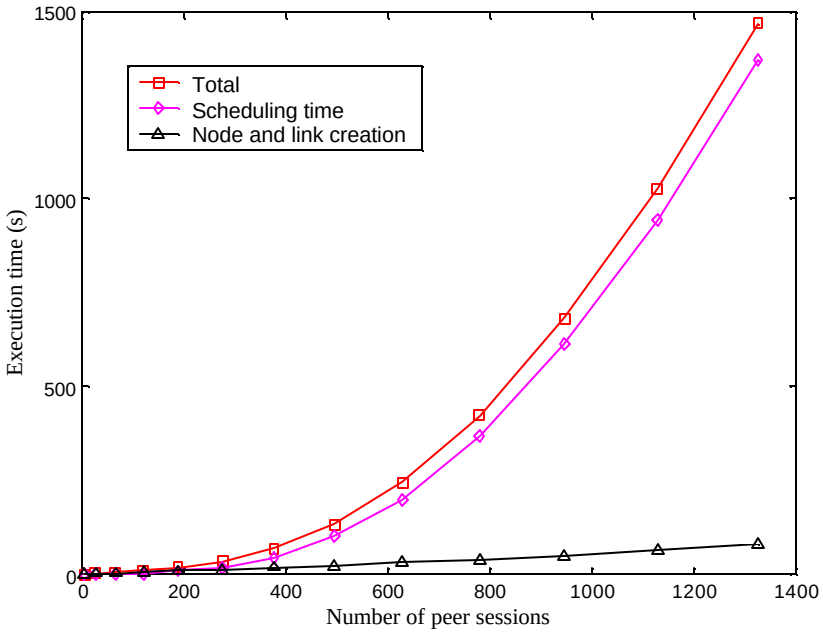


Figure 5.6 Execution times of *clique topologies* (without jittered timers).

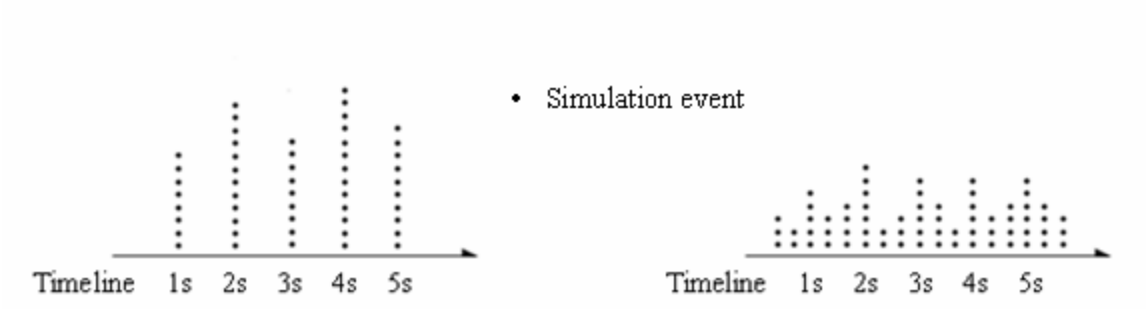


Figure 5.7 Scattering events (left) along the time line by jittering the timers (right).

5.1.5 Measurements

We measured execution time and memory usage of the ns-BGP model in every simulation phase. Besides collecting the statistics for an individual phase, we also calculated *total* sum of the statistics from different phases. The sum of phases 2 (node creation) and 3 (link creation) is named *node and link creation. ns-BGP* is the sum of phase 6 (session establishment) and phases 7 (message exchange). The *total* time is almost equal to the sum of *node and link*

creation and *ns-BGP*, since the statistics of phases 1 (simulator instance creation), phase 4 (enabling auto-configuration), phase 5 (initialization of the simulator) are small enough to be ignored.

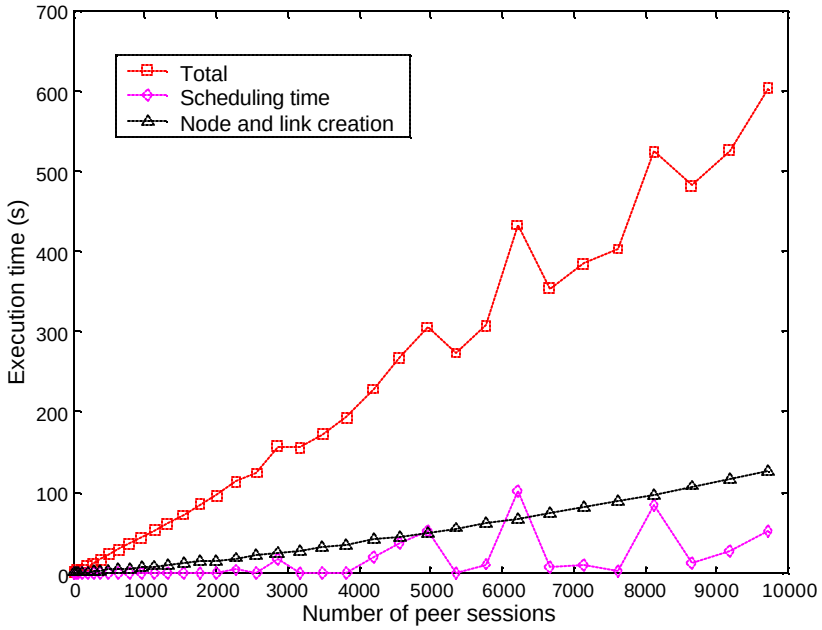


Figure 5.8 Execution times of *clique topologies* (with jittered timers).

The *clock* OTcl command was used to retrieve timestamps with millisecond precision at the beginning of each phase and at the end of simulation. These timestamps were used to calculate the execution time of each phase. The *malloc* C library call was employed to calculate the dynamic memory utilization using a modification of the approach proposed in [29].

5.2 Scalability: number of peer sessions

In this section, we illustrate the ns-BGP model's scalability with respect to the number of peer sessions, by examining the execution time and memory usage of BGP.

5.2.1 Line topology

Execution times of different phases for networks with the line topology are shown as functions of the number of peer sessions in Figure 5.9. The *total* execution time increases

nonlinearly with the number of peer sessions. The *total* execution time, including the *scheduling time* as one of its components, is affected by the randomness (jitter) introduced to the BGP timers.

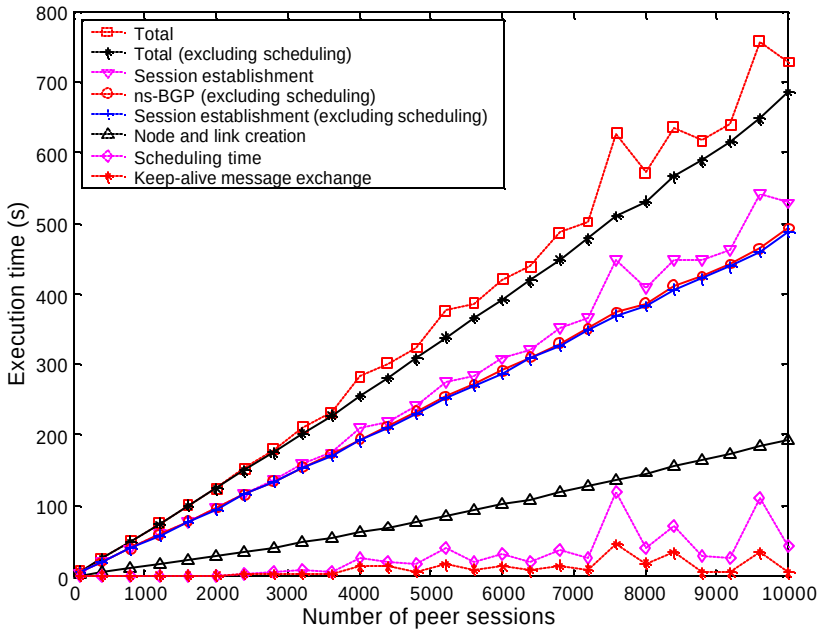


Figure 5.9 Execution times for line topologies. Simulated time is 100 s.

In order to exclude the effect of the randomized *scheduling time*, we examine the *total (excluding scheduling)* execution time, which is calculated by subtracting the *scheduling time* from the *total* execution time. As shown in Figure 5.9, the *total (excluding scheduling)* execution time increases smoothly, but still shows a slight exponential trend.

The *total (excluding scheduling)* execution time mainly consists of two parts: the *node and link creation* and *ns-BGP (excluding scheduling)* execution times. Because nodes and links are created before the simulation begins, the *node and link creation* time is not affected by the scheduler performance. However, the slight exponential trend shown by the *total (excluding scheduling)* execution time results from the *node and link creation* time. The node and link creation are not a part of the ns-BGP model. Since the performance degradation it caused is not severe, we have not attempted to improve the node and link creation processes in ns-2.

Similar to the *total (excluding scheduling)* time, the *ns-BGP (excluding scheduling)* time is calculated to exclude the affect of the randomized *scheduling time*. The *ns-BGP (excluding scheduling)*, which is the actual contribution of the ns-BGP model to the execution time, increases linearly. The *session establishment (excluding scheduling)* and *keep-alive message exchange* execution times are also measured. The *keep-alive message exchange* execution time fluctuates with the *scheduling time* as shown in Figure 5.9. The *session establishment (excluding scheduling)* execution time increases linearly and is very close to the *ns-BGP (excluding scheduling)* execution time. This implies that the ns-BGP model spent most of its execution time in session establishment and only a small portion in exchanging keep-alive message.

Memory utilizations of different simulation phases and their linear dependence on the number of peer sessions are show in Figure 5.10. We calculated a *total* memory usage of 54.21 Kbytes per peer.

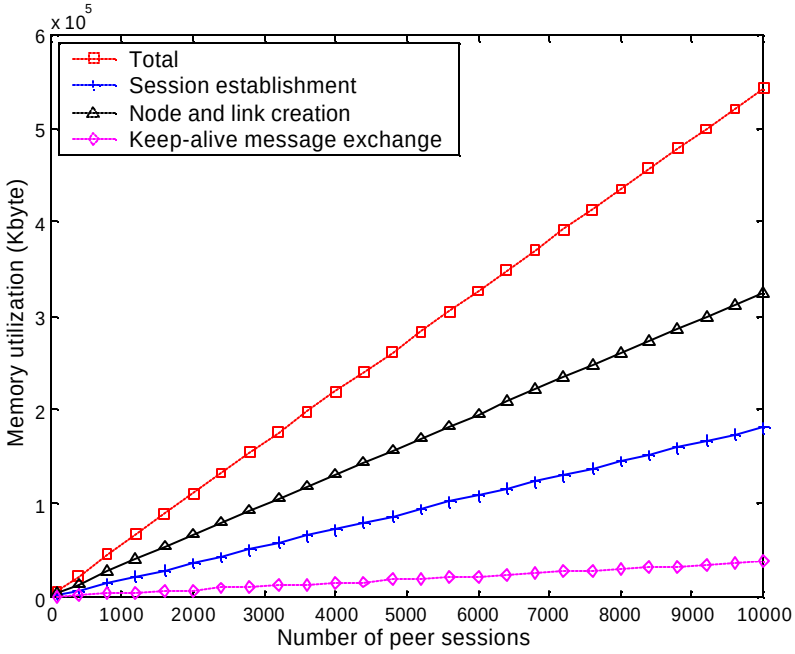


Figure 5.10 Memory utilization for line topologies. Simulated time is 100 s.

5.2.2 Ring topology

The *ns-BGP (excluding scheduling)* execution time and the *total* memory usage of the ring topologies increase linearly in the number of peer sessions, as shown in Figures 5.11 and 5.12. We calculated a *total* memory usage of 54.21 Kbytes per peer.

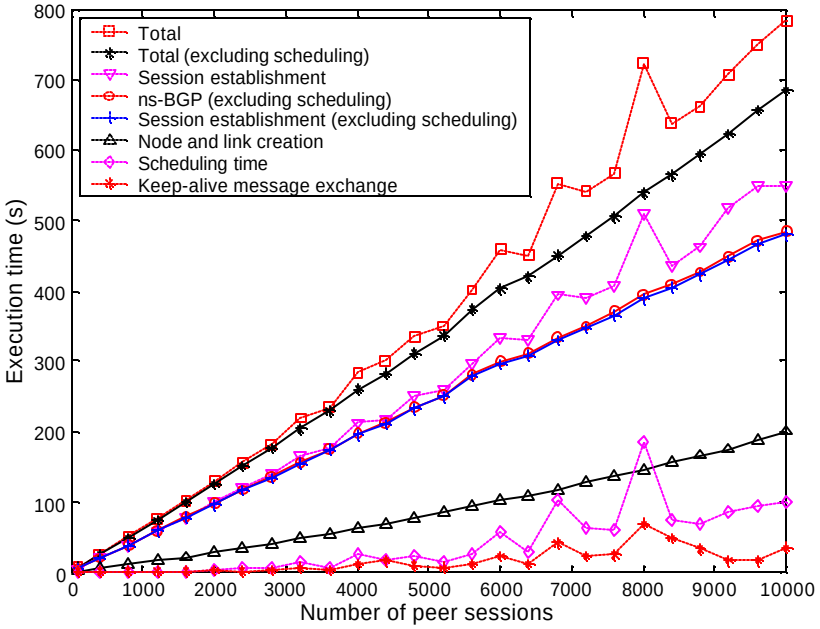


Figure 5.11 Execution times for ring topologies. Simulated time is 100 s.

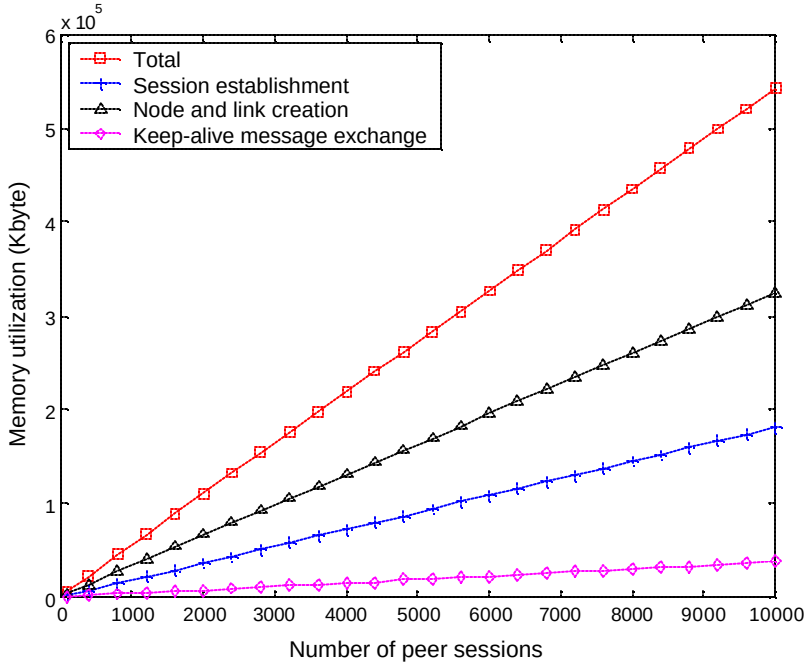


Figure 5.12 Memory utilization for ring topologies. Simulated time is 100 s.

5.2.3 Binary tree topology

The *ns-BGP (excluding scheduling)* execution time and *total* memory usage of the binary tree topologies increase linearly in the number of peer sessions, as shown in Figures 5.13 and 5.14. We calculated a *total* memory usage of 54.3 Kbytes per peer.

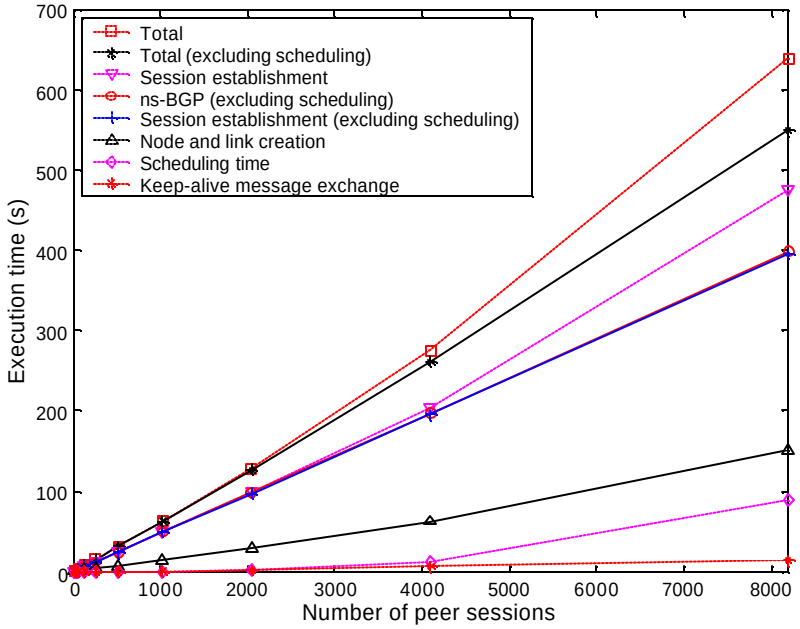


Figure 5.13 Execution times for binary trees. Simulated time is 100 s.

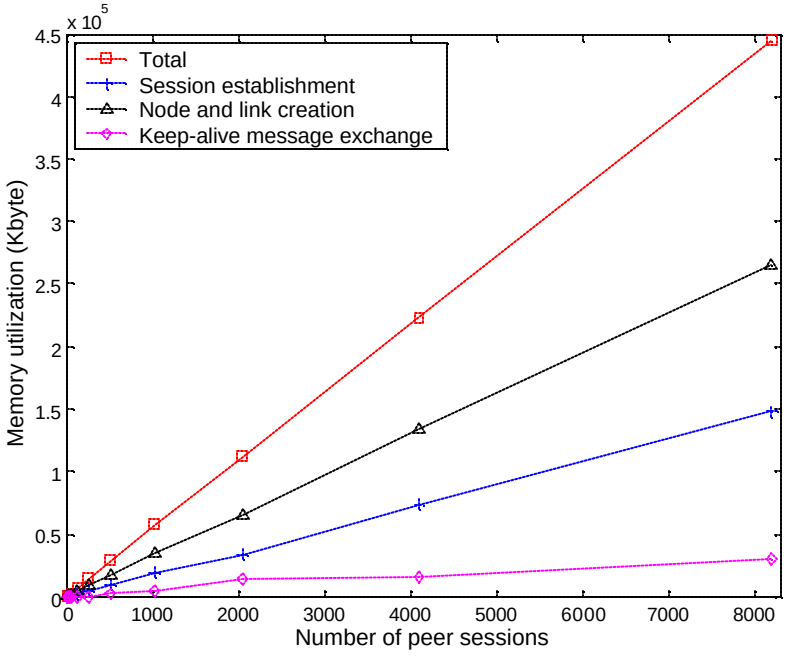


Figure 5.14 Memory utilization for binary trees. Simulated time is 100 s.

5.2.4 Grid topology

The *ns-BGP (excluding scheduling)* execution time and *total* memory usage of the grid topologies increase linearly in the number of peer sessions, as shown in Figures 5.15 and 5.16. We calculated a *total* memory usage of 47.7 Kbytes per peer.

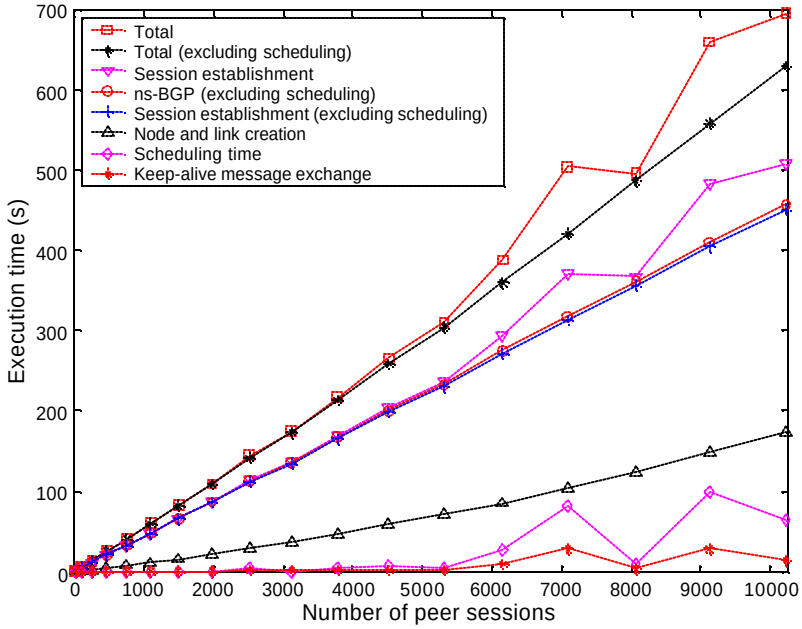


Figure 5.15 Execution times for grid topologies. Simulated time is 100 s.

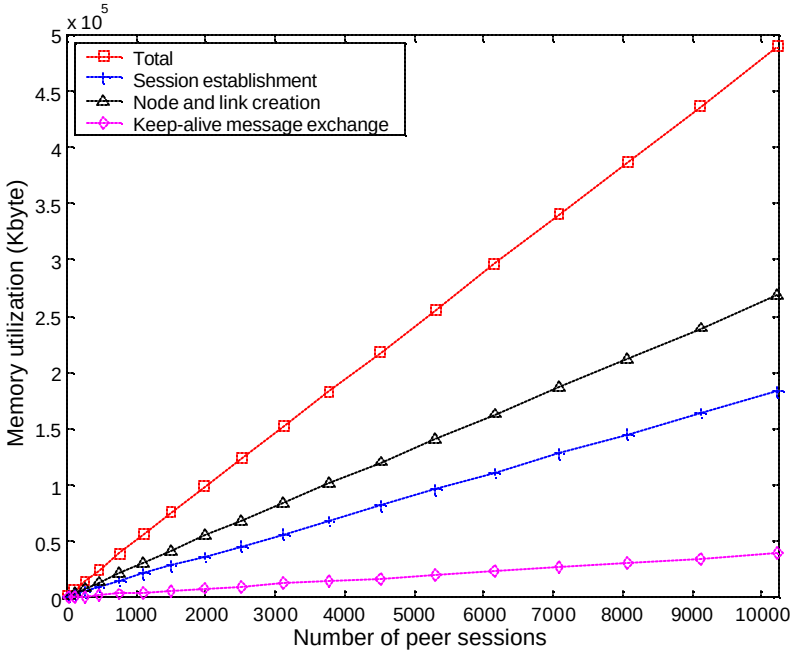


Figure 5.16 Memory utilization for grid topologies. Simulated time is 100 s.

5.2.5 Clique topology

The *ns-BGP (excluding scheduling)* execution time and *total* memory usage of the clique topologies increase linearly in the number of peer sessions, as shown in Figures 5.17 and 5.18. We calculated a *total* memory usage of 43.8 Kbytes per peer.

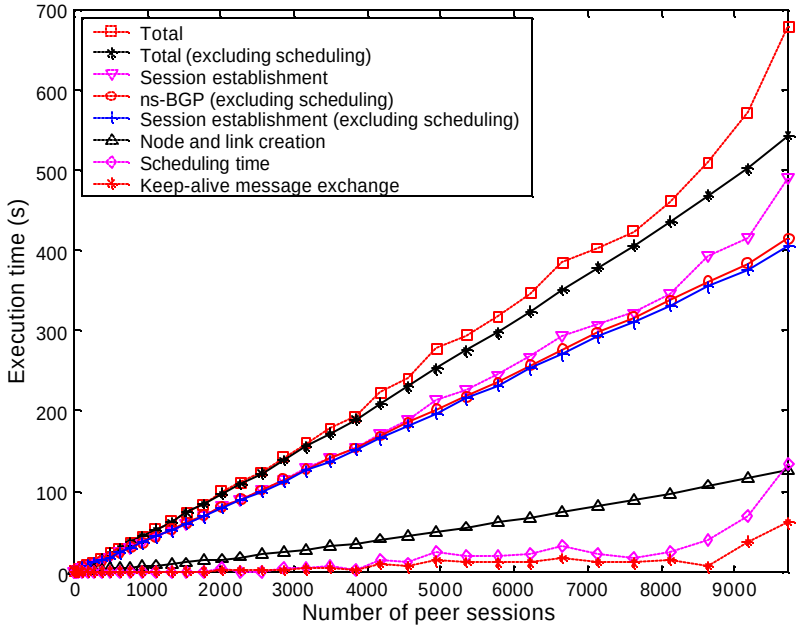


Figure 5.17 Execution times for clique topologies. Simulated time is 100 s.

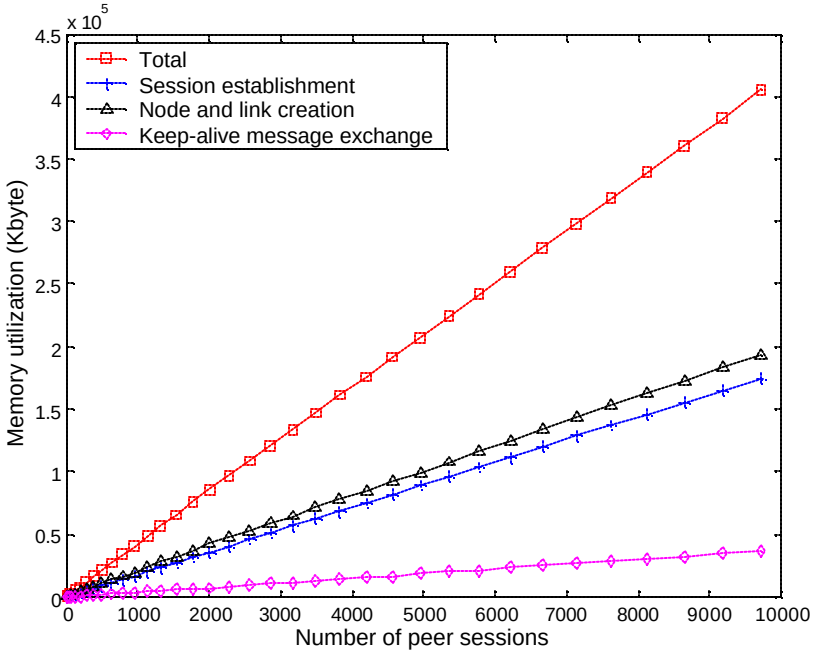


Figure 5.18: Memory utilization for clique topologies. Simulated time is 100 s.

5.3 Scalability: size of routing tables

In this section, we analyze the scalability of the ns-BGP model with respect to the size of routing tables. We examine the execution time and memory usage of BGP simulations with five topologies. The experiments in this section are performed on topologies with static sizes unlike the experiments shown in Section 5.2. The five static topologies used are: *line*, *ring*, *grid*, and *clique* topologies of size 16 and a *binary tree* topology of size 15. In order to analyze the model's scalability with respect to the size of routing tables *M*, each node is configured to send *M*/16 (*line*, *ring*, *grid*, *clique* topologies) or *M*/15 routes (*binary tree* topology) to its peers. After the process converged, the routing table of each node should contain *M* routes.

5.3.1 Line topology

Execution times for different simulation phases for the line topology as functions of the size of routing tables are shown in Figure 5.19. Given the small topology size, the *node and link creation* and *session establishment* execution times are close to zero. On the other hand, the *total* and *message exchange* execution times are similar, which implies that the simulator spent most of its time in exchanging BGP messages (*keep-alive* and *update*). The *total* and *message exchange* execution times increase linearly.

The topologies used for this scalability analysis have small number of peer sessions. For an instance, the line topology of size 16 has 15 peer sessions. Hence, very few events are scheduled for the same time instance and, thus, the scheduler performs well. The scheduling times are very small, less than 0.5% of the *total* execution times. Therefore, we only show the *total*, *session establishment*, *node and link creation*, and the *message exchang*e execution times.

Memory utilizations for different simulation phases and their linear dependence on the size of routing tables are show in Figure 5.20. We calculated a *total* memory usage of 20.88 Kbytes per route.

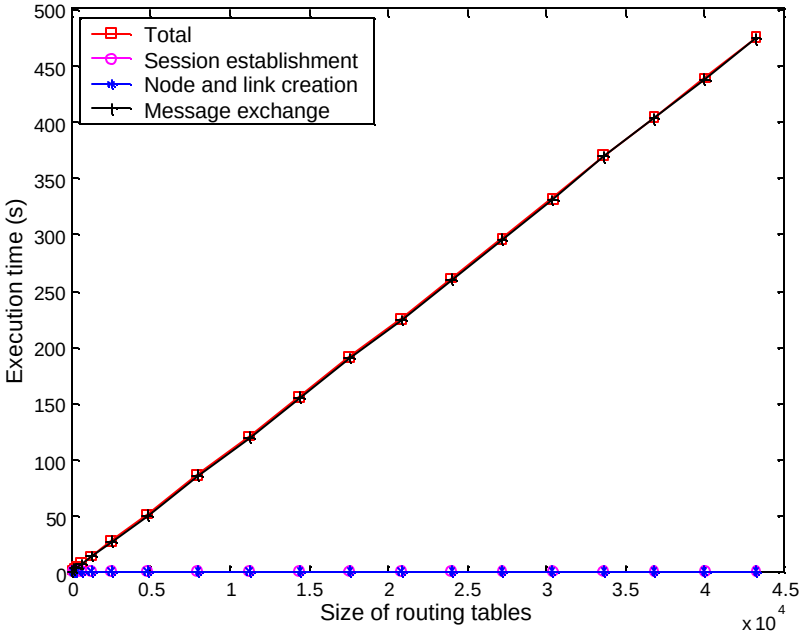


Figure 5.19 Execution times for the line topology. Simulated time is 10,000 s.

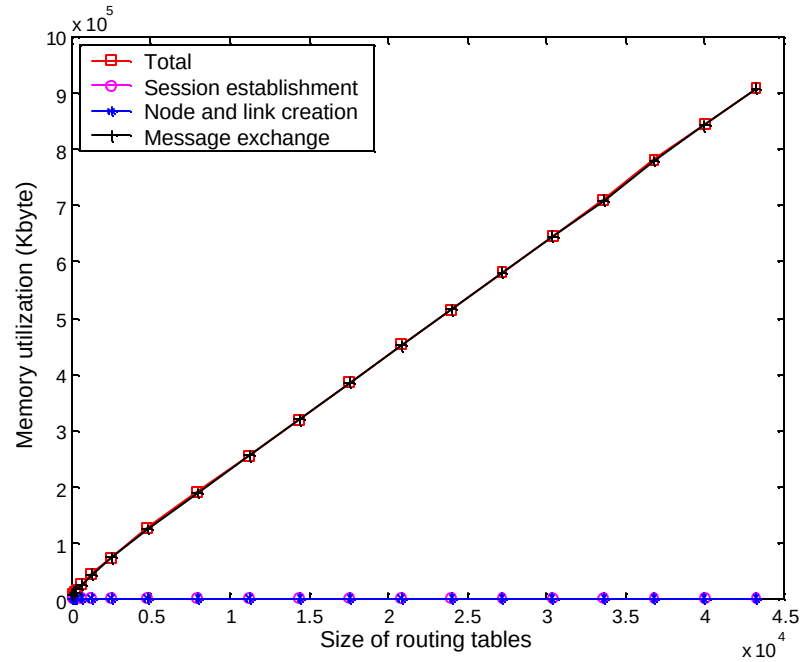


Figure 5.20 Memory utilization for the line topology. Simulated time is 10,000 s

5.3.2 Ring topology

Similar to the line topology, we found that the *ns-BGP (excluding scheduling)* execution time and memory usage of the ring topology both increase linearly in the size of routing tables, as shown in Figures 5.21 and 5.22. We calculated a *total* memory usage of 24.97 Kbytes per route.

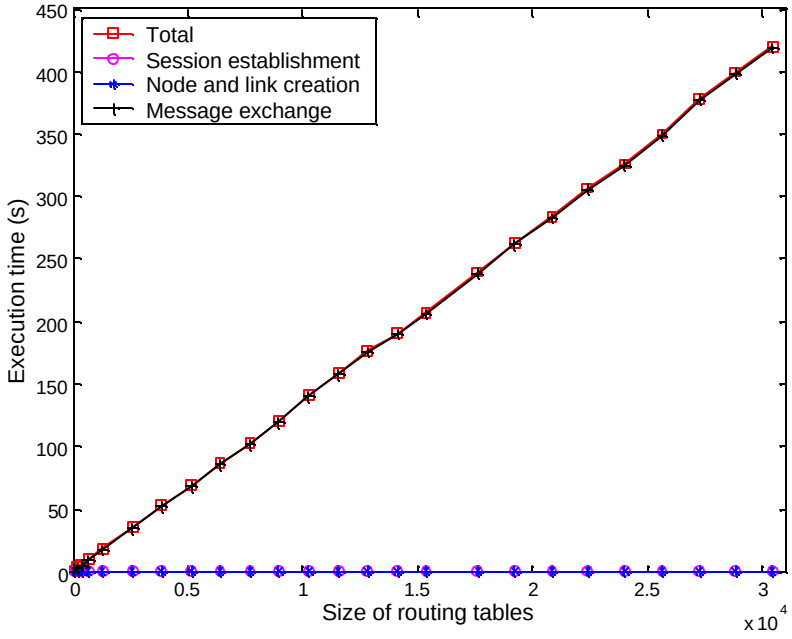


Figure 5.21 Execution times for the ring topology. Simulated time is 10,000 s.

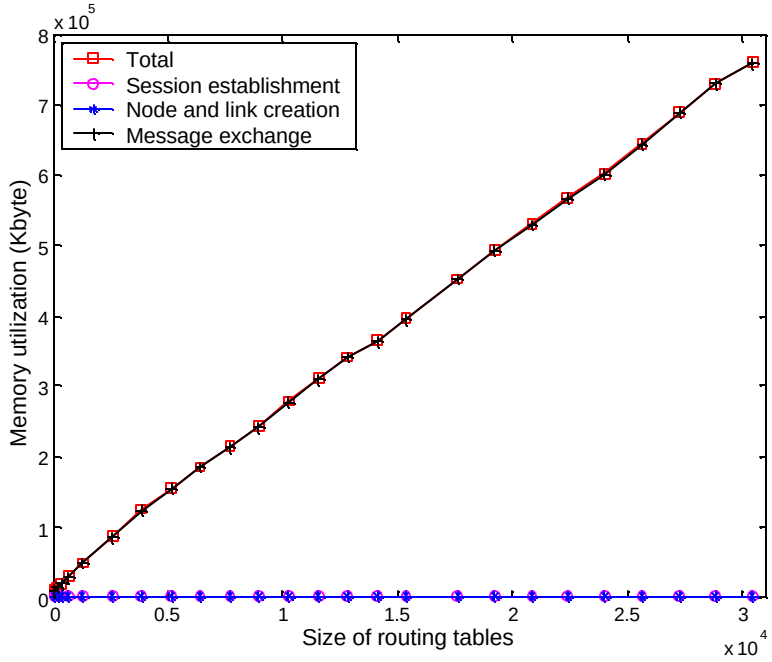


Figure 5.22 Memory utilization for the ring topology. Simulated time is 10,000 s

5.3.3 Binary tree topology

The *ns-BGP (excluding scheduling)* execution time and *total* memory usage of the binary tree topology increase linearly in the size of routing tables, as shown in Figures 5.23 and 5.24. We calculated a *total* memory usage of 19.28 Kbytes per route.

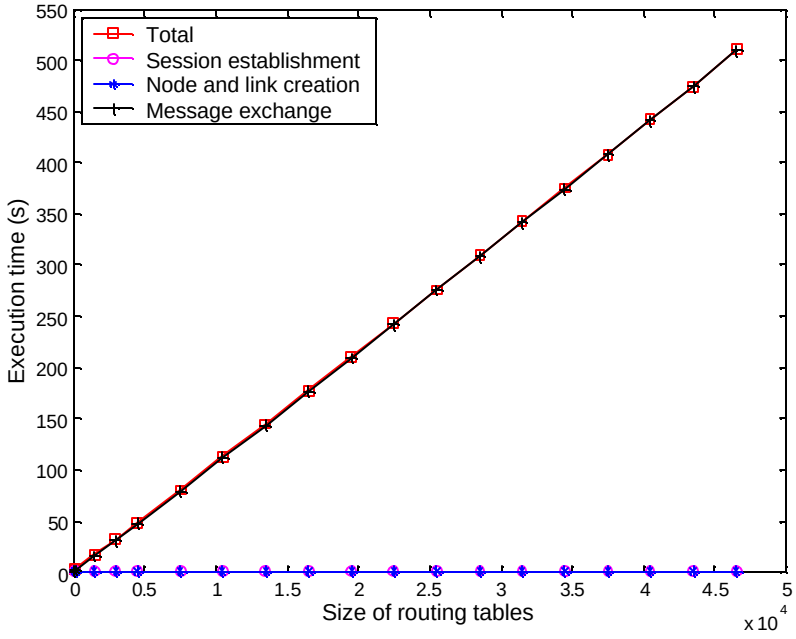


Figure 5.23 Execution times for the binary tree. Simulated time is 10,000 s.

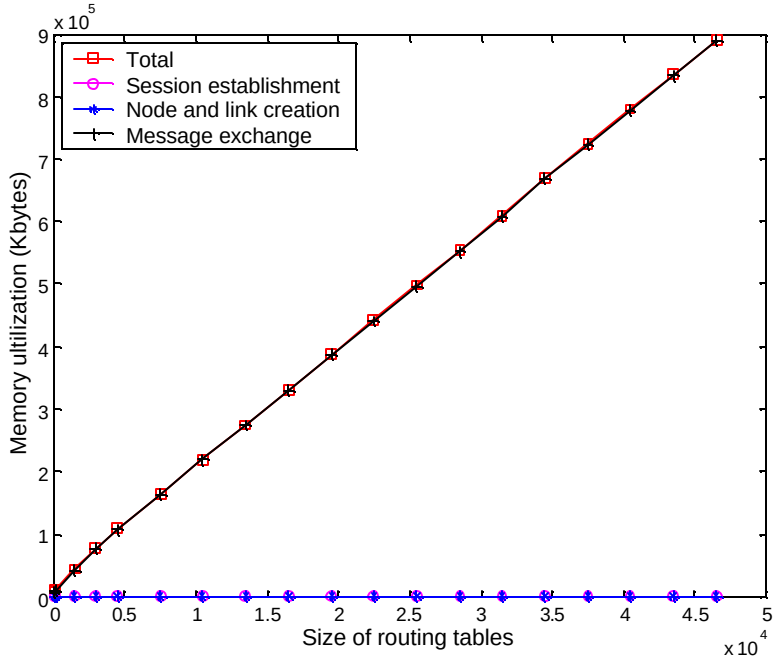


Figure 5.24 Memory utilization for the binary tree. Simulated time is 10,000 s

5.3.4 Grid topology

The *ns-BGP (excluding scheduling)* execution time and *total* memory usage of the grid topology increase linearly in the size of routing tables, as shown in Figures 5.25 and 5.26. We calculated a *total* memory usage of 60.54 Kbytes per route.

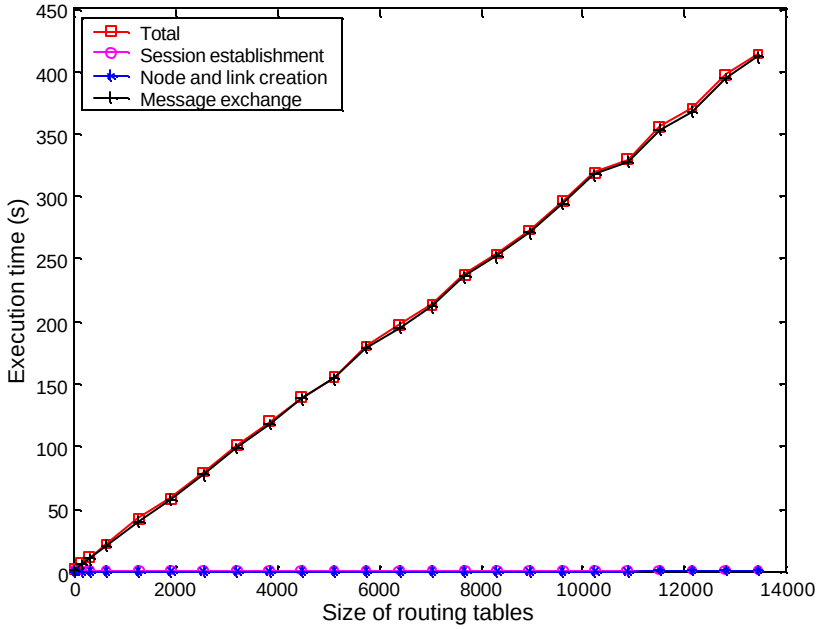


Figure 5.25 Execution times for the grid topology. Simulated time is 10,000 s.

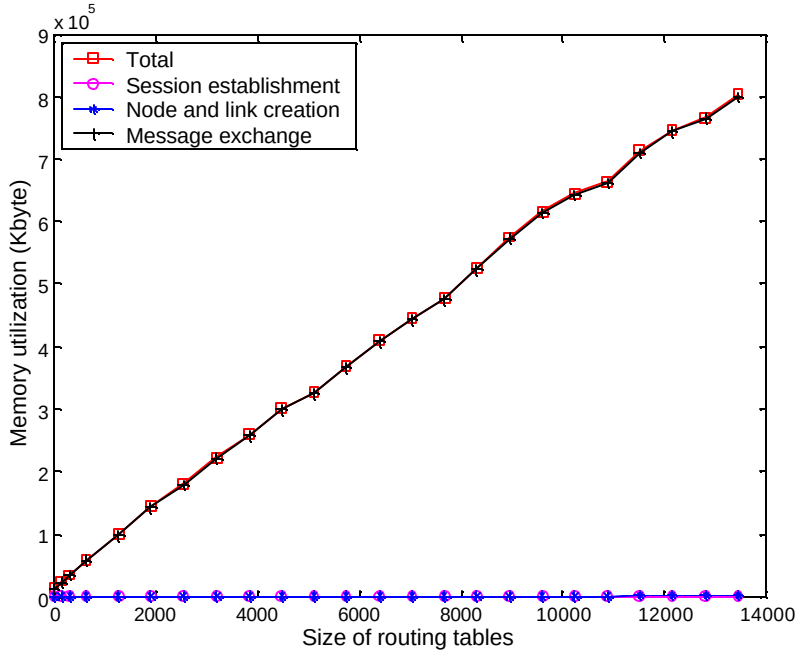


Figure 5.26 Memory utilization for the grid topology. Simulated time is 10,000 s.

5.3.5 Clique topology

The *ns-BGP (excluding scheduling)* execution time and *total* memory usage of the clique topology increase linearly in the size of routing tables, as shown in Figures 5.27 and 5.28. We calculated a *total* memory usage of 67.25 Kbytes per route.

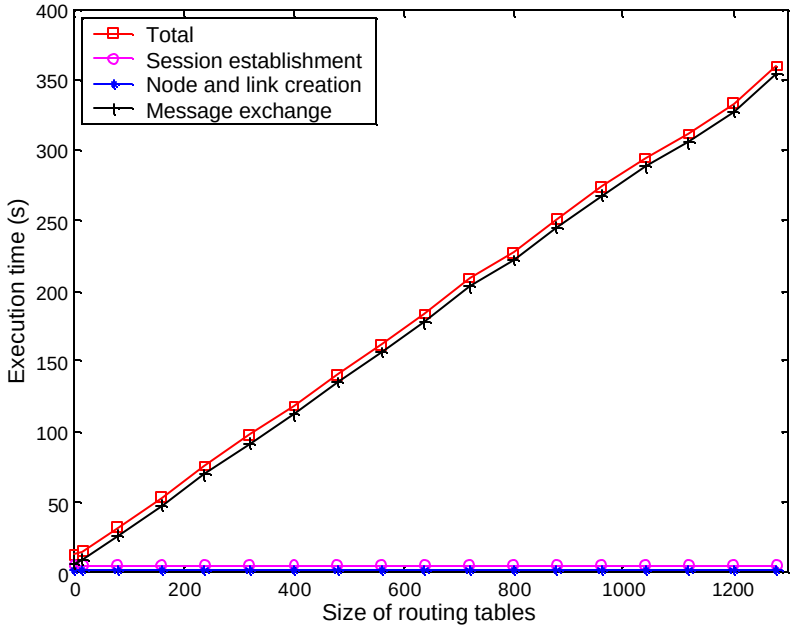


Figure 5.27 Execution times for the clique topology. Simulated time is 10,000 s.

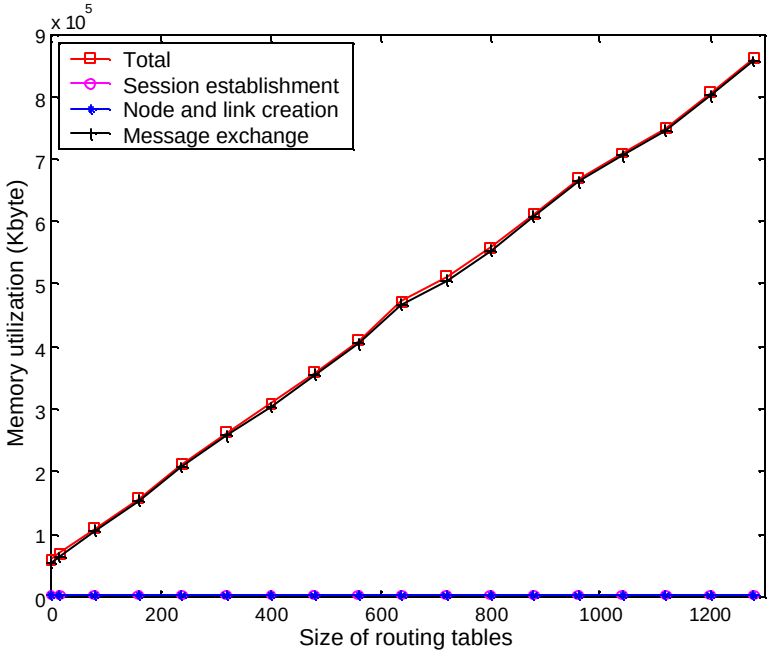


Figure 5.28: Memory utilization for the clique topology. Simulated time is 10,000 s.

CHAPTER 6:
CONCLUSIONS

In this thesis, we presented the architecture and implementation of ns-BGP, a BGP-4 model for the ns-2 network simulator. ns-BGP enables simulation and evaluation of BGP protocol and its variants. The validation test illustrated the validity of the ns-BGP implementation. Our scalability analysis was based on various network topologies. It shows that the internal data structures and employed algorithms are scalable in terms of the number of peer sessions and the size of routing tables. The ns-BGP implementation also includes several optional BGP features.

As for feature work, more realistic network topologies and routing polices can be employed to simulate genuine behavior of the Internet. Additional features, such as route flap damping, policy routing, and multiprotocol extension, may be added to the existing ns-BGP model. These features will help compare the performance of various algorithms for route flap damping, study the detailed behavior of BGP policy routing, and evaluate new technologies that are based on the multiprotocol extension, such as BGP/MPLS (Multiprotocol Label Switching) VPN (virtual private network).

APPENDIX A: TEST SCRIPTS FOR VALIDATION TESTS

A.1 Route selection

```
#
# select.tcl
#

puts ""
puts "SELECT Validation Test: "
puts ""
puts " A \"triangle\" consisting of three ASes.  Each AS has one"
puts " BGP-speaking router.  Each router is connected directly to"
puts " the routers in each neighboring AS."
puts ""
puts "    AS----AS "
puts "     \\    /  "
puts "      \\  /   "
puts "       AS    "
puts ""


set nf [open select.nam w]
set ns [new Simulator]
$ns namtrace-all $nf

$ns node-config -BGP ON
set n0 [$ns node 0:10.0.0.1]
set n1 [$ns node 1:10.1.1.1]
set n2 [$ns node 2:10.2.2.1]
$ns node-config -BGP OFF

$ns duplex-link $n0 $n1 1Mb 1ms DropTail
$ns duplex-link $n0 $n2 1Mb 1ms DropTail
$ns duplex-link $n1 $n2 1Mb 1ms DropTail

set bgp_agent0 [$n0 get-bgp-agent]
$bgp_agent0 bgp-id 10.0.0.1
$bgp_agent0 neighbor 10.1.1.1 remote-as 1
$bgp_agent0 neighbor 10.2.2.1 remote-as 2

set bgp_agent1 [$n1 get-bgp-agent]
$bgp_agent1 bgp-id 10.1.1.1
$bgp_agent1 neighbor 10.0.0.1 remote-as 0
$bgp_agent1 neighbor 10.2.2.1 remote-as 2

set bgp_agent2 [$n2 get-bgp-agent]
$bgp_agent2 bgp-id 10.2.2.1
$bgp_agent2 neighbor 10.0.0.1 remote-as 0
$bgp_agent2 neighbor 10.1.1.1 remote-as 1

$ns at 0.25 "puts \"\n time: 0.25 \n n0 (ip_addr 10.0.0.1) \
                    defines a network 10.0.0.0/24.\""
$ns at 0.25 "$bgp_agent0 network 10.0.0.0/24"

$ns at 39.0 "puts \"\n time: 39 \
                  \n dump routing tables in all BGP agents: \n\""
$ns at 39.0 "$bgp_agent0 show-routes"
$ns at 39.0 "$bgp_agent1 show-routes"
$ns at 39.0 "$bgp_agent2 show-routes"

$ns at 40.0 "finish"

proc finish {} {
```

```
        global ns nf
        $ns flush-trace
        close $nf
        puts "Simulation finished. Executing nam..."
        exec nam select.nam
        exit 0
}

puts "Simulation starts..."
$ns run
```

A.2 Reconnection

```
#
# reconnect.tcl
#

puts ""
puts "RECONNECT Validation Test:"
puts ""
puts " Three ASes connected in a line, each with one router."
puts "       AS 1       AS 0       AS 2"
puts "       n1 }------{ n0 }------{ n2"
puts ""


set nf [open reconnect.nam w]
set ns [new Simulator]
$ns namtrace-all $nf

$ns node-config -BGP ON
set n0 [$ns node 0:10.0.0.1]
set n1 [$ns node 1:10.1.1.1]
set n2 [$ns node 2:10.2.2.1]
$ns node-config -BGP OFF

$ns duplex-link $n0 $n1 1Mb 1ms DropTail
$ns duplex-link $n0 $n2 1Mb 1ms DropTail

set bgp_agent0 [$n0 get-bgp-agent]
$bgp_agent0 bgp-id 10.0.0.1
$bgp_agent0 neighbor 10.1.1.1 remote-as 1
$bgp_agent0 neighbor 10.2.2.1 remote-as 2

set bgp_agent1 [$n1 get-bgp-agent]
$bgp_agent1 bgp-id 10.1.1.1
$bgp_agent1 neighbor 10.0.0.1 remote-as 0
$bgp_agent1 neighbor 10.0.0.1 keep-alive-time 200

set bgp_agent2 [[$n2 get-module BGP] get-bgp-agent]
$bgp_agent2 bgp-id 10.2.2.1
$bgp_agent2 neighbor 10.0.0.1 remote-as 0

$ns at 0.25 "puts \"\n time: 0.25 \n n0 (ip_addr 10.0.0.1) \
                      defines a network 10.0.0.0/24.\""
$ns at 0.25 "$bgp_agent0 network 10.0.0.0/24"
$ns at 0.35 "puts \"\n time: 0.35 \n n1 (ip_addr 10.1.1.1) \
                      defines a network 10.1.1.0/24.\""
$ns at 0.35 "$bgp_agent1 network 10.1.1.0/24"
$ns at 0.45 "puts \"\n time: 0.45 \n n2 (ip_addr 10.2.2.1) \
                      defines a network 10.2.2.0/24.\""
$ns at 0.45 "$bgp_agent2 network 10.2.2.0/24"

## Network converges at 27.25*.
$ns at 28.0 "puts \"\n time: 28 \
                    \n dump routing tables in all BGP agents: \n\""
$ns at 28.0 "$bgp_agent0 show-routes"
$ns at 28.0 "$bgp_agent1 show-routes"
$ns at 28.0 "$bgp_agent2 show-routes"
```

```
## At 90.35, HoldTimer of bgp_agent0 expired, bgp_agent0 will
## 1. drop peer with bgp_agent1,
## 2. withdrawl route that learned from bgp_agent1

## Connection closing finished at 90.36*.
$ns at 90.38 "puts \"\n time: 90.38 \
                    \n dump routing tables in all BGP agents: \n\""
$ns at 90.38 "$bgp_agent0 show-routes"
$ns at 90.38 "$bgp_agent1 show-routes"
$ns at 90.38 "$bgp_agent2 show-routes"

## Network converges at 117.50* again after reconnection.
$ns at 119.0 "puts \"\n time: 119 \
                    \n dump routing tables in all BGP agents: \n\""

$ns at 119 "$bgp_agent0 show-routes"
$ns at 119 "$bgp_agent1 show-routes"
$ns at 119 "$bgp_agent2 show-routes"

$ns at 120.0 "finish"

proc finish {} {
       global ns nf
       $ns flush-trace
       close $nf
       puts "Simulation finished. Executing nam..."
       #exec nam reconnect.nam
       exit 0
}

puts "Simulation starts..."
$ns run

#* These times are recorded with "jitter_factor_seed" set to 12345.
#  (Please see file bgp/global.h)
```

A.3 Route reflection

```
#
# reflection2.tcl
#

puts ""
puts "REFLECTION2 VALIDATION TEST:"
puts ""
puts " Three ASes(AS0, AS1 and AS2) connected in a line, the middle "
puts " one(AS0) containing eight BGP routers, the others just one each."
puts " AS0 has two clusters: cluster 1000 and 2000. Cluster 1000 has "
puts " two reflectors: n0 and n1. n2, n3 and n4 are reflection clients of "
puts " both n0 and n1. Cluster 2000 contains one reflector n5, which has"
puts " n6 and n7 as its reflection clients. "
puts ""
puts "        AS 1         AS 0         AS 2 "
puts "         n8 }------{ n0-7 }------{ n9 "
puts ""

set nf [open reflection2.nam w]
set ns [new Simulator]
$ns namtrace-all $nf

$ns node-config -BGP ON
set n0 [$ns node 0:10.0.0.1]
set n1 [$ns node 0:10.0.1.1]
set n2 [$ns node 0:10.0.2.1]
set n3 [$ns node 0:10.0.3.1]
set n4 [$ns node 0:10.0.4.1]
set n5 [$ns node 0:10.0.5.1]
set n6 [$ns node 0:10.0.6.1]
```

```
set n7 [$ns node 0:10.0.7.1]
set n8 [$ns node 1:10.1.8.1]
set n9 [$ns node 2:10.2.9.1]
$ns node-config -BGP OFF
set n10 [$ns node 1:10.1.10.1]

## SETUP INTER-REFRECTOR LINKS
$ns duplex-link $n0 $n1 1Mb 1ms DropTail
$ns duplex-link $n0 $n5 1Mb 1ms DropTail
$ns duplex-link $n1 $n5 1Mb 1ms DropTail

## SETUP REFRECTOR-CLIENT LINKS
$ns duplex-link $n0 $n2 1Mb 1ms DropTail
$ns duplex-link $n0 $n3 1Mb 1ms DropTail
$ns duplex-link $n0 $n4 1Mb 1ms DropTail
$ns duplex-link $n1 $n2 1Mb 1ms DropTail
$ns duplex-link $n1 $n3 1Mb 1ms DropTail
$ns duplex-link $n1 $n4 1Mb 1ms DropTail
$ns duplex-link $n5 $n6 1Mb 1ms DropTail
$ns duplex-link $n5 $n7 1Mb 1ms DropTail

## SETUP INTRA-AS LINKS
$ns duplex-link $n8 $n10 1Mb 1ms DropTail

## SETUP EBGP LINKS
$ns duplex-link $n2 $n8 1Mb 1ms DropTail
$ns duplex-link $n7 $n9 1Mb 1ms DropTail

## SETUP REFRECTORS
set bgp_agent0 [$n0 get-bgp-agent]
$bgp_agent0 bgp-id 10.0.0.1
$bgp_agent0 cluster-id 1000
$bgp_agent0 neighbor 10.0.2.1 route-reflector-client
$bgp_agent0 neighbor 10.0.3.1 route-reflector-client
$bgp_agent0 neighbor 10.0.4.1 route-reflector-client
$bgp_agent0 neighbor 10.0.1.1 remote-as 0
$bgp_agent0 neighbor 10.0.5.1 remote-as 0

set bgp_agent1 [$n1 get-bgp-agent]
$bgp_agent1 bgp-id 10.0.1.1
$bgp_agent1 cluster-id 1000
$bgp_agent1 neighbor 10.0.2.1 route-reflector-client
$bgp_agent1 neighbor 10.0.3.1 route-reflector-client
$bgp_agent1 neighbor 10.0.4.1 route-reflector-client
$bgp_agent1 neighbor 10.0.0.1 remote-as 0
$bgp_agent1 neighbor 10.0.5.1 remote-as 0

set bgp_agent5 [$n5 get-bgp-agent]
$bgp_agent5 bgp-id 10.0.5.1
$bgp_agent1 cluster-id 2000
$bgp_agent5 neighbor 10.0.6.1 route-reflector-client
$bgp_agent5 neighbor 10.0.7.1 route-reflector-client
$bgp_agent5 neighbor 10.0.1.1 remote-as 0
$bgp_agent5 neighbor 10.0.0.1 remote-as 0

## SETUP CLIENTS
set bgp_agent2 [$n2 get-bgp-agent]
$bgp_agent2 bgp-id 10.0.2.1
$bgp_agent2 neighbor 10.0.0.1 remote-as 0
$bgp_agent2 neighbor 10.0.1.1 remote-as 0
$bgp_agent2 neighbor 10.1.8.1 remote-as 1

set bgp_agent3 [$n3 get-bgp-agent]
$bgp_agent3 bgp-id 10.0.3.1
$bgp_agent3 neighbor 10.0.0.1 remote-as 0
$bgp_agent3 neighbor 10.0.1.1 remote-as 0

set bgp_agent4 [$n4 get-bgp-agent]
$bgp_agent4 bgp-id 10.0.4.1
$bgp_agent4 neighbor 10.0.0.1 remote-as 0
$bgp_agent4 neighbor 10.0.1.1 remote-as 0
```

```
set bgp_agent6 [$n6 get-bgp-agent]
$bgp_agent6 bgp-id 10.0.6.1
$bgp_agent6 neighbor 10.0.5.1 remote-as 0

set bgp_agent7 [$n7 get-bgp-agent]
$bgp_agent7 bgp-id 10.0.7.1
$bgp_agent7 neighbor 10.0.5.1 remote-as 0
$bgp_agent7 neighbor 10.2.9.1 remote-as 2

## SETUP EBGP'S
set bgp_agent8 [$n8 get-bgp-agent]
$bgp_agent8 bgp-id 10.1.8.1
$bgp_agent8 neighbor 10.0.2.1 remote-as 0

set bgp_agent9 [$n9 get-bgp-agent]
$bgp_agent9 bgp-id 10.2.9.1
$bgp_agent9 neighbor 10.0.7.1 remote-as 0

set udp0 [new Agent/UDP]
$udp0 set dst_addr_ [$n4 strtoaddr 10.1.10.1]
$udp0 set dst_port_ 0

set cbr0 [ new Application/Traffic/CBR]
$cbr0 set packetSize_ 20
$cbr0 set interval_ 0.001
$cbr0 attach-agent $udp0
$ns attach-agent $n4 $udp0

$ns at 0.23 "puts \"\n time: 0.23 \
                  \n cbr0 starts to send UDP segments to n10.\""
$ns at 0.23 "$cbr0 start"

$ns at 0.25 "puts \"\n time: 0.25 \n n8 (ip_addr 10.1.8.1) \
                     defines a network 10.1.10.0/24.\""
$ns at 0.25 "$bgp_agent8 network 10.1.10.0/24"

$ns at 0.35 "puts \"\n time: 0.35 \n n9 (ip_addr 10.2.9.1) \
                     defines a network 10.2.9.0/24.\""
$ns at 0.35 "$bgp_agent9 network 10.2.9.0/24"

$ns at 20 "puts \"\n time: 20 \n cbr0 stops.\""
$ns at 20 "$cbr0 stop"

$ns at 39.0 "puts \"\n time: 39
                  \n dump routing tables in all BGP agents: \n\""
$ns at 39.0 "$bgp_agent0 show-routes"
$ns at 39.0 "$bgp_agent1 show-routes"
$ns at 39.0 "$bgp_agent2 show-routes"
$ns at 39.0 "$bgp_agent3 show-routes"
$ns at 39.0 "$bgp_agent4 show-routes"
$ns at 39.0 "$bgp_agent5 show-routes"
$ns at 39.0 "$bgp_agent6 show-routes"
$ns at 39.0 "$bgp_agent7 show-routes"
$ns at 39.0 "$bgp_agent8 show-routes"
$ns at 39.0 "$bgp_agent9 show-routes"

$ns at 40.0 "finish"

proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        puts "Simulation finished. Executing nam..."
        exec nam reflection2
        exit 0
}

puts "Simulation starts..."
$ns run
```

APPENDIX B: SAMPLE SCRIPT FOR SIMULATION PHASES

```
set line_size 16           ##  Line topology with 16 nodes.
set route_number 1         ##  Each node announces one prefix.
set finish_time 10000      ##  Simulated time is 10000 s.

##
# Phase 1: ns-2 simulator instance creation
##
set ns [new Simulator]

##
# Phase 2: node creation
##
$ns node-config -BGP ON
for {set i 0} {$i < $line_size } {incr i} {
   set n($i) [$ns node $i:10.0.$i.1]
}
$ns node-config -BGP OFF

##
# Phase 3: link creation
##
for {set i 0} {$i < [expr $line_size -1] } {incr i} {
        $ns duplex-link $n($i) $n([expr $i + 1]) 1Mb 1ms DropTail
}
##
# Phase 4: enable each BGP agent to be auto-config
##
for {set i 0} {$i < $line_size } {incr i} {
set bgp_agent($i) [$n($i) get-bgp-agent]
        $bgp_agent($i) set-auto-config
}

##
# Phase 5: Other initialization
##

##
# Phase 6: BGP session establishment
##
$ns at 0.0 “puts \”Begin establishing BGP sessions. \””

##
# Phase 7: BGP message exchange
##
for {set i 0} {$i < $line_size } {incr i} {
     for {set j 0} {$i < $route_number } {incr j} {
        $ns at 20.0 “$bgp_agent($i) network $i.0.$route_number.0/24
}

##
# Simulation terminates at 10000s
##
$ns at $finish_time "finish"

proc finish {} {
        exit 0
}

puts "Simulation starts..."
$ns run
```

BIBLIOGRAPHY

[1] J. Banks, J. Carson II, B. Nelson, and D. Nicol, *Discrete-Event System Simulation*. Upper Saddle River, NJ: Prentice Hall, 2001.

[2] T. Bates, R. Chandra, and E. Chen, "BGP route reflection – an alternative to full mesh IBGP," RFC 2796, April 2000.

[3] T. Bates, Y. Richter, R. Chandra, and D. Katz, "Multiprotocol extensions for BGP-4," RFC 2858, June 2000.

[4] T. Bates, P. Smith, and G. Huston, CIDR Report: http://www.cidr-report.org. Accessed: April 10, 2004.

[5] I. Beijnum, BGP. Sebastopol, CA: O'Reilly, 2002.

[6] BGP++: http://www.ece.gatech.edu/research/labs/MANIACS/BGP++. Accessed: April 10, 2004.

[7] R. Brown, "Calendar queues: a fast O(1) priority queue implementation for the simulation event set problem," *Communication of the ACM*, vol. 31, no. 10, pp. 1120-1227, October1988.

[8] T. Bu, L. Gao, and D. Towsley, "On routing table growth," in *Proc. of Global Internet Symposium*, Taipei, Taiwan, November 2002.

[9] E. Chen and J. Stewart, "A framework for inter-domain route aggregation," RFC 2519, February 1999.

[10] N. Feamster and H. Balakrishnan, "Towards a logic for wide-area Internet routing," in *Proc. SIGCOMM*, Karlsruhe, Germany, August 2003, pp. 88-100.

[11] T. D. Feng, R. Ballantyne, and Lj. Trajkovic, "Implementation of BGP in a network simulator," to be presented at the *Applied Telecommunication Symposium*, ATS '04, Arlington, Virginia, April 2004.

[12] L. Gao, "On inferring autonomous system relationships in the Internet," in *Proc. GLOBECOM*, San Francisco, CA, November 2000, pp. 378-396.

[13] L. Gao and J. Rexford. "Stable Internet routing without global coordination," in *Proc. SGIMETRICS*, Santa Clara, CA, June 2000, pp. 307-317.

[14] T. Griffin and B. Premore, "An experimental analysis of BGP convergence time," in *Proc. ICNP*, Riverside, CA, November 2001, pp. 53-61.

[15] T. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proc. SIGCOMM*, Cambridge, MA, August 1999, pp. 277-288.

[16] T. Griffin and G. Wilfong, "A safe path vector protocol," in *Proc. INFOCOM*, Anchorage, Alaska, April 2001, pp. 490-499.

[17] T. Griffin, F. Shepherd, and G. Wilfong, "Policy disputes in path-vector protocols," in *Proc. ICNP*, Toronto, Canada, October 1999, pp. 21-30.

[18] T. Griffin, F. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE Transactions on Networking*, vol. 10, no. 2, pp. 232-243, April 2002.

[19] GNU Zebra: http://www.zebra.org. Accessed: April 10, 2004.

[20] GNU Zebra BGP daemon: http://www.zebra.org/zebra/BGP.html#BGP. Accessed: April 10, 2004.

[21] S. Halabi and D. McPherson, *Internet Routing Architectures*. Indianapolis, IN: Cisco Press, 2000.

[22] C. Huitema, *Routing in the Internet*. Upper Saddle River, NJ: Prentice Hall, 2000.

[23] N. Hutchinson and L. Peterson, "The x-kernel: an architecture for implementing network protocols," *IEEE Transactions on Software Engineering*, vol. 17, no. 1, pp. 64-76, January 1991.

[24] C. Labovitz, G. Malan, and F. Jahanian "Origins of Internet routing instability," in *Proc. INFOCOM*, New York, NY, March 1999, pp. 218-226.

[25] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja, "The impact of Internet policy and topology on delayed routing convergence," in *Proc. INFOCOM*, Anchorage, AK, April 2001, pp. 537-546.

[26] G. Malkin, "RIP version 2," RFC 2453, November 1998.
.

[27] Z. Mao, R. Govindan, G. Varghese, and R. Katz. "Route flap damping exacerbates Internet routing convergence," in *Proc. SIGCOM*, Pittsburgh, PA, August 2002, pp. 221-233.

[28] S. Murphy, "BGP security vulnerabilities analysis," Internet draft, June 2003.

[29] D. Nicol, "Scalability of network simulators revisited," in *Proc. CNDS*, Orlando, FL, February 2003.

[30] ns manual: http://www.isi.edu/nsnam/ns/doc/index.html. Accessed: April 10, 2004.

[31] OPNET BGP: http://www.opnet.com/products/bgp.html. Accessed: April 10, 2004.

[32] V. Paxson, "End to end routing behavior in the Internet," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 601-615, October 1997.

[33] B. Premore, SSFNet BGP User's Guide: http://www.ssfnet.org/bgp/user-guide-ps.zip. Accessed: April 10, 2004.

[34] B. Premore, *An Analysis of Convergence Properties of the Border Gateway Protocol Using Discrete Event Simulation*, Ph.D. thesis, Dartmouth College, May 2003.

[35] SSFNet: http://www.ssfnet.org/homePage.html. Accessed: April 10, 2004.

[36] J. Stewart III, BGP4: *Inter-Domain Routing in the Internet*. Reading, MA: Addison-Wesley, 1998.

[37] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," RFC 1771, March 1995.

[38] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," ISI Tech. Rep. 96-631, February 1996.

[39] C. Villamizar, R. Chandra, and R. Govindan, "BGP route flap damping," RFC 2439, November 1998.