Implementation of BGP in a Network Simulator

Tony Dongliang Feng, Rob Ballantyne, and Ljiljana Trajković
Simon Fraser University
Vancouver, British Columbia
Canada
{tdfeng, ballanty, ljilja}@cs.sfu.ca

Abstract

Border Gateway Protocol (BGP) is the inter-domain routing protocol currently employed in Internet. Internet growth imposes increased requirements on BGP performance. Recent studies revealed that performance degradations in BGP are due to the highly dynamic nature of the Internet. In this paper, we describe the design of the ns-BGP model and its implementation in the ns-2 network simulator. We describe a validation test for route reflection and provide a scalability performance analysis of the ns-BGP model.

Keywords: inter-domain routing, BGP, scalability, route reflection, ns-2.

1. INTRODUCTION

The Internet consists of thousands of interconnected Autonomous Systems (ASs) loosely defined as networks and routers under a single administrative control. Routing in the Internet is performed on two levels (intra-domain and inter-domain) implemented by two sets of protocols. Interior gateway protocols (IGP), such as RIP, IS-IS, OSPF, IGRP, and EIGRP, route packets within a single AS (intra-domain). Exterior gateway protocols (EGP), such as EGP and BGP routes packets between ASs (inter-domain).

Although the Border Gateway Protocol (BGP) has been analyzed in detail, routing instability [1], [2], inefficient routing [3], [4], and scalability issues [5] still remain. Theoretical analysis and empirical measurements have been employed in the past, albeit with certain limitations [6]. Simulations allow more realistic experiments with fewer simplifications than the theoretical approach and with enhanced flexibility than empirical studies permit. We implemented a BGP-4 [7] model, the current version of BGP, in the network simulator ns-2 [8] by porting the BGP-4 implementation from SSFNet [9].

The rest of the paper is organized as follows. In Section 2, we introduce background on BGP, ns-2, and SSFNet. The design and implementation of ns-BGP are described in Section 3. A validation example for route reflection is presented in Section 4. We analyze the scalability of ns-BGP in Section 5 and conclude with Section 6.

2. BGP IMPLEMENTATIONS

A short introduction to BGP, ns-2, and the SSFNet network simulator follows.

2.1 Border Gateway Protocol

BGP-4 is the de facto inter-domain routing protocol [7]. It is used by BGP routers in ASs to exchange reachability information and to determine the end-to-end path for packets traversing multiple ASs.

BGP employs TCP as its transport protocol, which ensures transport reliability and eliminates the need for BGP to handle retransmissions. 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 the session management [7].

2.2 ns-2 network simulator

We implemented ns-BGP as an extension to the latest version of ns-2 network simulator (ns-2.26) [8]. ns-2, one of the most popular network simulators, 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 than performance. A graphical animator *nam* is used to visualize simulation results.

2.3 BGP implementation in SSFNet

SSF.OS.BGP4 is the BGP-4 model [6] in the SSFNet [9] 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, was designed with a purely object-oriented approach. We ported to ns-2 the

class hierarchy that implements the BGP-4 model in SSF.OS.BGP4.

2.4 Related work in BGP implementation

OPNET [10], 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 [11], supporting BGP [12] and other routing protocols. The Zebra BGP daemon has been recently ported to ns-2 [13]. Our project has been developed in parallel. We preferred the SSF.OS.BGP4 implementation because of its object oriented paradigm.

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 follows.

3.1 ns-2 unicast routing structure

The ns-2 unicast routing structure consists of the forwarding and the control plane [8], as shown in Figure 1.

The forwarding plane is responsible for classifying and forwarding packets to the destination nodes. It includes various types of connected *classifiers* and *routing module*s. Classifiers deliver the incoming packets either to the correct 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 c*lassifier_* 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. *dmux_* forwards the packet to an agent corresponding to the packet's destination port number.

The control plane handles route computation, 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 *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 [14].

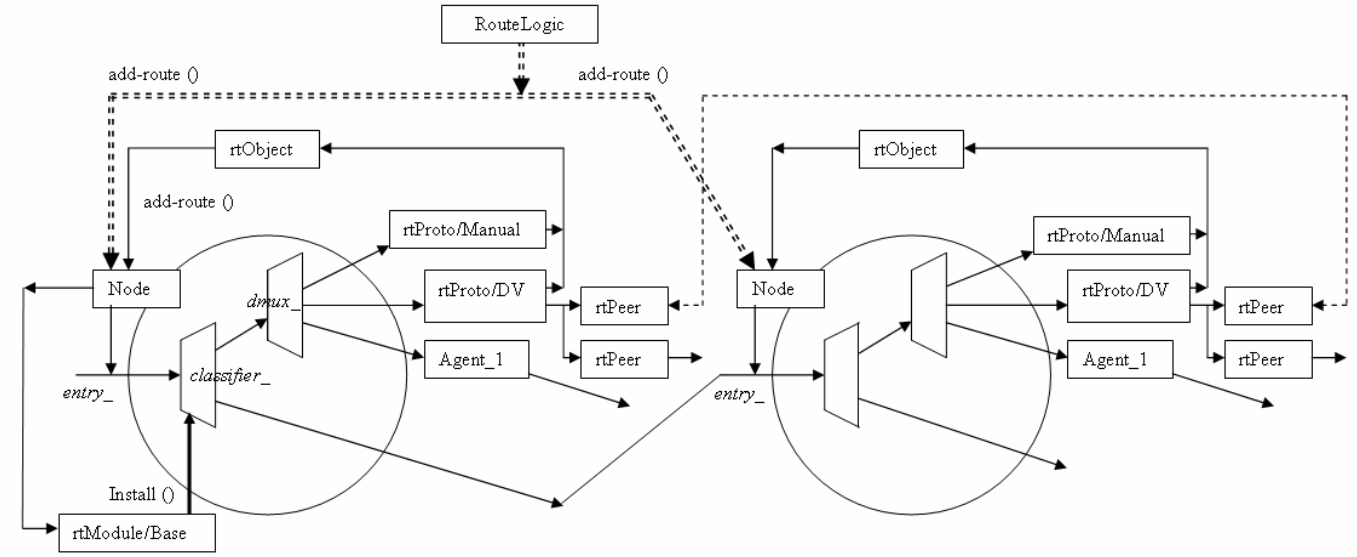


Figure 1. ns-2 unicast routing structure. Address classifier (*classifier_*) and port classifier (*dmux_*) are two types of classifiers (trapezoids) in an ns-2 unicast node. *Classifier_* forwards a packet to the *dmux_* or sends it to a downstream node. *dmux_* forwards the packet to the corresponding agent.

3.2 Unicast routing architecture of ns-BGP

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 and the 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* [8], the TCP agent for *TcpSocket*.

Figure 2 shows the unicast structure of ns-BGP. Address classifier *classifier_* is an *IPv4Classifier*. A new routing module *rtModule/BGP* manages the *IPv4Classifier* and replaces 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. *rtProtoBGP* 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 [7]. *BGP_Timer* provides support for the BGP timing features (timers).

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

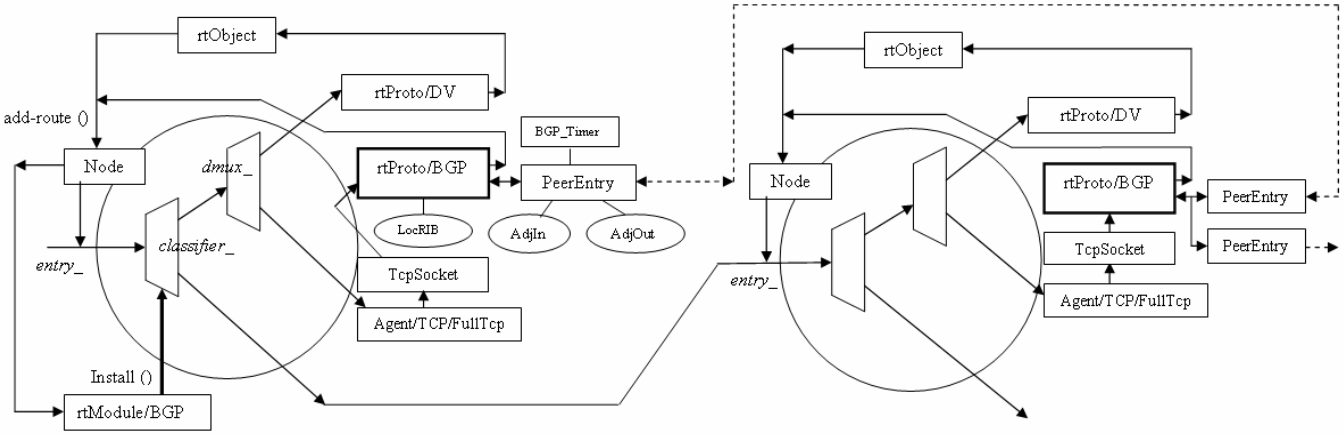


Figure 2. Unicast structure of ns-BGP. *rtModule/BGP* manages *classifier_*. *TcpSocket* resides on top of *Agent/TCP/FullTcp*, while routing protocol *rtProto/BGP* is introduced on top of *TcpSocket*. *rtProto/BGP* has one *PeerEntry* for each peer.

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.

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 a route that has 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 most 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.

3.3 Supported features

The implementation of the ns-BGP is compliant with the BGP-4 specification RFC 1771 [7]. Nevertheless, it currently does not support the multiprotocol extensions for BGP-4 [15]. 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.

4. VALIDATION TEST: ROUTE REFLECTION

SSF.OS.BGP4 includes 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 [6]. We implemented most of these validation tests in ns-2 and tested the same network topologies as employed in the SSFNet validation tests [9]. We also introduced a new validation test for route reflection [16].

4.1 Network topology

Figure 3 shows the network topology used for simulation of route reflection. The network consists of three AS's: 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 addresses space associated with each node is shown in Table 1.

Table 1. IP address space associated with network nodes.

Nodes: 0 through 7	10.0.0.0/24 though 10.0.7.0/24
Nodes: 8 and 10	10.1.8.0/24 and 10.1.10.0/24
Node: 9	10.2.9.0/24

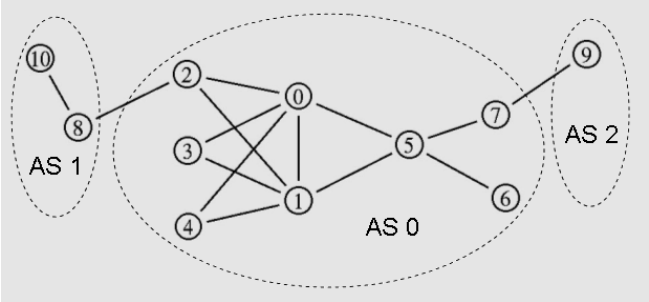


Figure 3. Network topology used in the route reflection validation test.

4.2 BGP configuration

The goal of the simulation test was to validate the behavior of multiple reflectors inside a BGP cluster [14]. AS 0 contains two clusters. The first cluster contains two reflectors: nodes 0 and 1. 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 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 10 bytes every millisecond to the IP address of node 10 (10.1.10.1).

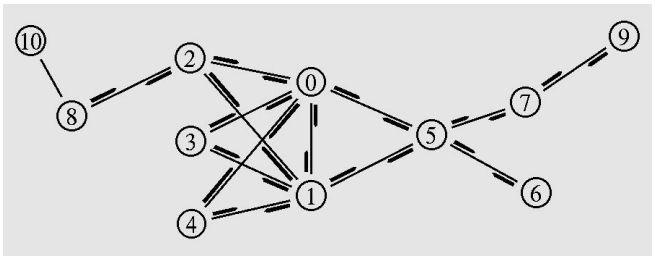
The traffic source begins sending UDP segments at 0.23 s and stops sending them at 20.0 s. At 0.25 s, the BGP agent in node 8 sends a route advertisement for a network 10.1.10.0/24 that is within its AS (AS 1). At 0.35 s, the BGP agent in node 9 sends a route advertisement for network 10.2.9.0/24 (AS 2). At 39.0 s, ns-2 displays all routing tables for BGP agents. The simulation terminates at 40.0 s.

4.4 Simulation results.

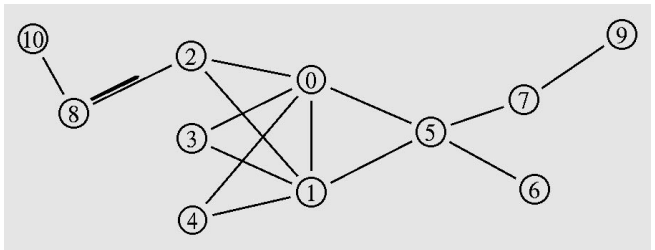
The simulation sequence of events is shown in Table 2. Simulation results displayed by *nam* are shown in Figures 4(a)–(g).

Table 2. Sequence of simulation events.

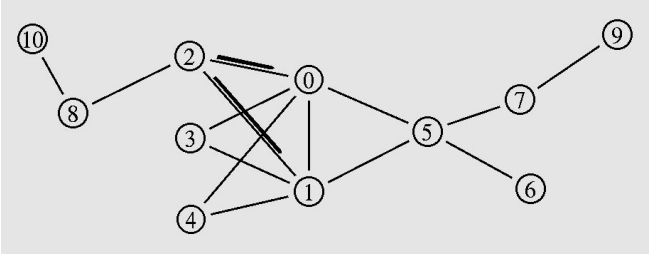
0.05 s	Figure 4(a): TCP SYN segments are exchanged between BGP peers, establishing the underlying TCP connections.
0.2505 s	Figure 4(b): node 8 originates an update message advertising the route for network 10.1.10.0/24.
0.2525 s	Figure 4(c): node 2 propagates the route advertisement to nodes 0 and 1.
0.2561 s	Figure 4(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(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.
0.2578 s	Figure 4(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(g): UDP segments are delivered to node 10.



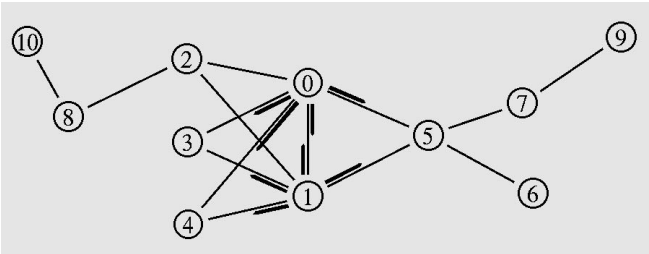
(a) Establishing TCP connections (0.05 s).



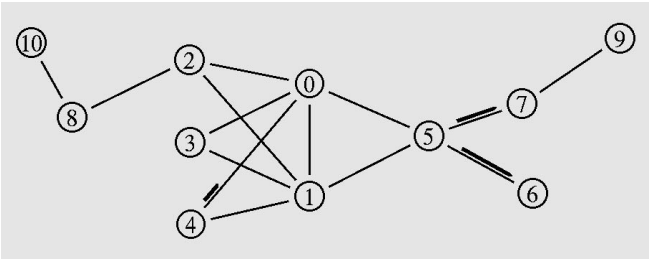
(b) Node 8 originates a route (0.2505 s).



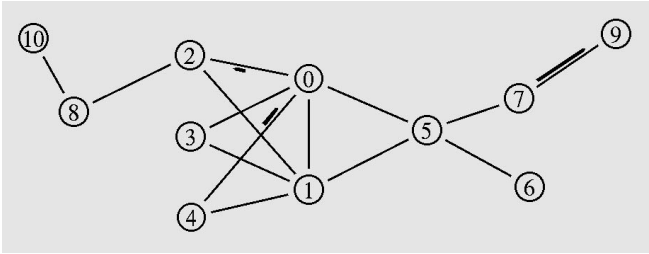
(c) Node 2 propagates the route to nodes 0 and 1 (0.2525 s).



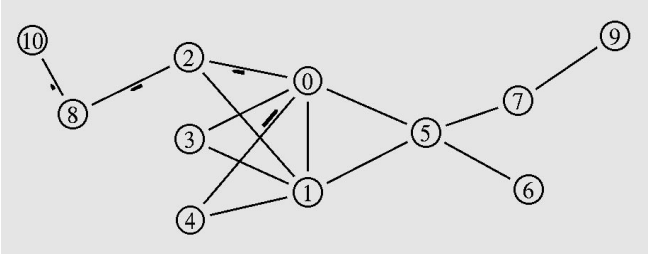
(d) Nodes 0 and 1 reflect the routes to nodes 3 and 4 (0.2561 s).



(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. Snapshots of simulation results.

By the end of the simulation run, every BGP node knows routes to 10.1.10.0/24 and 10.2.9.0/24. We printed routing tables for BGP agents at 39.0 s. Status codes are: * valid, > best, i – internal.

```
LocRIB dump of node 0, router ID: 10.0.0.1
     Network          Next Hop  Metric LP Weight Path
*>   10.1.10.0/24     10.0.2.1       -  -    - 1      i
*>   10.2.9.0/24      10.0.7.1       -  -    - 2      i
LocRIB dump of node 1, router ID: 10.0.1.1
     Network          Next Hop  Metric LP Weight Path
*>   10.1.10.0/24     10.0.2.1       -  -    - 1      i
*>   10.2.9.0/24      10.0.7.1       -  -    - 2      i
                                .
                                .
                                .
LocRIB dump of node 8, router ID: 10.1.8.1
     Network          Next Hop  Metric LP Weight Path
*>   10.1.10.0/24     self
*>   10.2.9.0/24      10.0.2.1       -  -    - 0 2
LocRIB dump of node 9, router ID: 10.0.9.1
     Network          Next Hop  Metric LP Weight Path
*>   10.1.10.0/24     10.0.7.1       -  -    - 0 1
*>   10.2.9.0/24      self
```

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 [17]. 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 the RedHat Linux 9.0 operating system.

5.1 Scalability: number of peer sessions

We used completely connected network topologies to analyze the scalability of the ns-BGP model with respect to the number of peer sessions. Each node is an individual AS containing one BGP instance and it is connected to every other node by eBGP sessions. Figure 5 shows the execution times of the ns-2 simulations as function of the number of peer sessions.

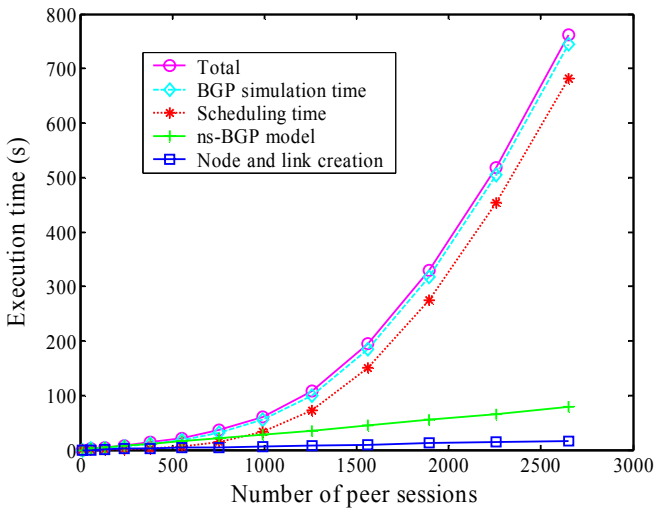


Figure 5. Execution times for completely connected networks. Simulated time is 100 s. BGP timer intervals are the default values suggested in RFC 1771 [7].

The total ns-BGP execution time increases nonlinearly with the number of peer sessions. The nonlinear increase is due to the execution time spent by the scheduler that is responsible for scheduling simulation events. The contribution to execution time due to ns-BGP model increases linearly in the number of peer sessions. This contribution is no larger than the difference between the total execution time and the execution time spent by the scheduler.

The *malloc* C library call was employed to calculate the dynamic memory utilization per peer session using a modification of the approach given in [17]. Each peer session required 50.6 Kbytes of memory.

5.2 Scalability: size of routing tables

We use the network topology shown in Figure 3 to analyze the scalability of the ns-BGP model with respect to the size of routing tables. In the simulation experiment, nodes 8 and 9 send to their peers a number of routes (half the number of routes contained in their respective routing tables). The execution times and their linear dependence on the size of routing tables are shown in Figure 6.

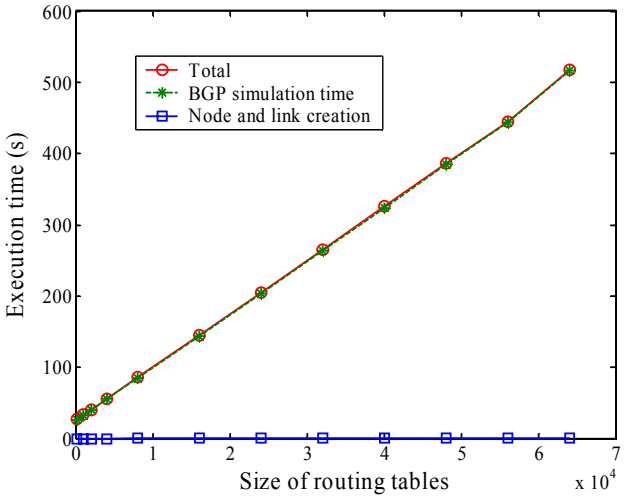


Figure 6. Execution time for various sizes of routing tables. Simulated time is 100,000 s. BGP timer intervals are default values as suggested in RFC 1771 [7].

We also measured the memory utilization of the ns-BGP model. We found that the total memory utilization grows linearly and calculated a memory usage of 23.1 Kbytes per route.

6. CONCLUSIONS

In this paper, 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 described ns-BGP implementation includes several optional BGP features. Other features, such as confederation and policy-based filtering, could be added in the future. The validation test illustrated the validity of the ns-BGP implementation. Our scalability 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.

7. ACKNOWLEDGEMENT

We thank Zheng Wang from SFU for developing the socket layer of the ns-BGP. We also thank anonymous reviewers for valuable comments that helped improve the manuscript.

8. REFERENCES

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

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

[3] 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.

[4] 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.

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

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

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

[8] ns manual: http://www.isi.edu/nsnam/ns/doc/index.html.

[9] SSFNet: http://www.ssfnet.org/homePage.html.

[10] OPNET BGP: http://www.opnet.com/products/bgp.html.

[11] GNU Zebra: http://www.zebra.org.

[12] GNU Zebra BGP daemon: http://www.zebra.org/zebra/BGP.html#BGP.

[13] BGP++: http://www.ece.gatech.edu/research/labs/MANIACS/BGP++.

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

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

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

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