PERFORMANCES EVALUATION OF BGP-4+ IN IPV4/IPV6

István Attila Katona, Virgil Dobrota, Tudor Blaga, Gabriel Lazar
Technical University of Cluj-Napoca, Department of Communications
Tel: +40-264-401226, 401264, 401816, Fax: +40-264-597083
E-mails: katonaistvanattila@yahoo.com, Virgil.Dobrota@com.utcluj.ro,
Tudor.Blaga@com.utcluj.ro, Gabriel.Lazar@com.utcluj.ro

Abstract: *BGP-4 (Border Gateway Protocol version 4) is the current EGP (Exterior Gateway Routing Protocol) used in Internet, being defined by IETF in 1995. The study is based on Network Simulator `ns-2.26`, with BGP++ 1.03a beta extension, under Linux RedHat 9.0/ Fedora Core 3 or later. There were three main objectives of this paper. The first one was the understanding the complex mechanism of establishing a logical BGP connection (finite state machine, messages, routing information) and evaluation of setup time depending on `ns-2` parameters (propagation delay and transfer rate). The second aim envisaged the configuration of Hold Time and KeepAlive interval for BGP connection state detection. After that, the paper evaluates the data traffic and link utilisation (about 70 % of bytes are IP and TCP Headers, the remaining being BGP messages).*

I. INTRODUCTION

Inside autonomous systems, the intra-domain routing is performed by IGPs *(Interior Gateway Routing Protocols).* Some examples are the following: RIP *(Routing Information Protocol)*, OSPF *(Open Shortest Path First)*, IS-IS *(Intermediate System to Intermediate System)*, EIGRP *(Enhanced Interior Gateway Routing Protocol).* They are optimized in accordance with the technical requirements, OSPF being recommended to be used whenever possible. Between autonomous systems, the inter-domain routing is realised by EGPs (Exterior Gateway Routing Protocols). The first protocol was called EGP but later on BGP *(Border Gateway Protocol)* took its place. BGP-4 *(Border Gateway Protocol version 4)* is the recommended inter-domain routing protocol nowadays, as its previous versions (BGP-1, BGP-2, BGP-3) or EGP are considered obsolete. Note that inter-AS routing usually reflects the business and political relationships between the networks and the companies involved rather than technical aspects [Hal00].

The paper presents an overview of the main concepts used in BGP. Messages, attributes, finite state machine and timers are briefly described in the first two paragraphs. The transition from IPv4-based networks to IPv6-enabled is currently requesting several, including inter-domain routing, This topic is discussed within the third paragraph, reminding the problem of multiprotocol extensions for BGP. Although the paper is concentrated on understanding the mechanism of establishing a logical BGP connection in IPv4, most of the work could be used for IPv6 too. However a comparative evaluation of network parameters (such as: setup time, HoldTime, KeepAlive interval, link utilisation) may offer a better point of view regarding the use of BGP in the new coming IPv6-based Internet. The study is based on Network Simulator `ns-2.26`, with BGP++ 1.03a beta extension, under Linux RedHat 9.0/ Fedora Core 3 or later.

II. BGP-4 MESSAGES AND ATTRIBUTES

BGP is an inter-autonomous system routing protocol that relies on IGPs for routing an AS. Its primary role is to exchange network reachability information with other BGP entities, including the list of autonomous systems that reachability information traverses. This information is enough to build a graph of connectivity, in which routing loops could be eliminated and routing policies at the AS level enforced [RFC1771]. BGP sees the Internet as a graph of autonomous systems, uniquely identified by their ASN *(AS Number).* A collection of path information associated with a given destination forms a loop-free route [Hal00]. According to [RFC1774] BGP cannot be classified as a pure distance vector protocol, neither as pure link state. Generally, it is considered a path vector routing protocol. This is similar to a distance vector protocol, but instead of measuring the distance, the entire path to the destination is given. Therefore, path vector protocols eliminate the well-known count-to-infinity problem of the usual distance vector protocols. For exchanging routing information, BGP connections are established over TCP *(Transmission Control Protocol)* at port 179. It is recommended to set PSH flag within TCP header, so that BGP data is delivered promptly to the Application Layer. Routers that run a BGP routing process are referred to as BGP speakers. Two BGP speakers that establish a TCP connection to exchange routing information are called peers or neighbours. When two BGP routers establish a connection, all BGP routes are exchanged between them. After the initial route exchange, only incremental updates are sent, in case network information changes. A BGP session established between two BGP speakers in different Autonomous Systems is called EBGP *(External BGP).* BGP speakers in the same

Autonomous System can also establish BGP sessions between them, in order to exchange routing information received from different external ASs. This option is used by transit autonomous systems and is called IBGP *(Internal BGP)*. It is important to mention that EBGP peers must be directly connected (except when using the EBGP Multi-hop option). On the other hand IBGP peers does not need to be directly connected, as IBGP is routed by the intra-domain routing protocols. Note that even IBGP and EBGP have different purposes, there is no difference in their implementation. Before establishing a BGP peer connection, two BGP speakers must perform the standard three-way handshake to open a TCP connection to port 179. There are four types of BGP unicast messages defined by [RFC1771]: OPEN, KEEPALIVE, UPDATE and NOTIFICATION. Later on [RFC2918] defined ROUTE-REFRESH.

A BGP message is processed only after it has been entirely received. Each BGP message (up to 4096 bytes) consists of a message header (19 bytes) and a data portion (could be optional). BGP path attributes are a set of parameters to keep track of route-specific information, such as path, degree of preference, next hop, and aggregation information. Attributes are used in the route decision and filtering process. BGP path attributes fall into four separate categories: well-known mandatory, well-known discretionary, optional transitive and optional non-transitive. Well-known attributes must be recognized by all BGP implementations and must be passed along to other BGP peers (possibly after updating). Well-known mandatory attributes must be included in every UPDATE message. Well-known discretionary attributes may, or may not be included in an UPDATE message. Optional attributes may be added to each path in addition to well-known attributes. It is not required that all BGP implementations recognize all optional attributes. If a BGP peer receives an unrecognized optional attribute, it acts depending on the Transitive Bit from the Attribute Flags octet. If the attribute is transitive, it should be accepted and passed along to other peers, with the Partial Bit set to 1. If the attribute is non-transitive, it should be ignored and not passed along to other peers.

Table 1. BGP path attributes

Type Code	Attribute Type	Category	Value Code	Attribute Value	Defined by
1	ORIGIN	Well-known mandatory	0	IGP	[RFC1771]
			1	EGP	
			2	Incomplete	
2	AS_PATH	Well-known mandatory	1	AS_SET	[RFC1771]
			2	AS_SEQUENCE	
			3	AS_CONFED_SET	[RFC3065]
			4	AS_CONFED_SEQUENCE	
3	NEXT_HOP	Well-known mandatory	-	Next-hop IP address	[RFC1771]
4	MULTI_EXIT_DISC	Optional non-transitive	-	4-octet MED	[RFC1771]
5	LOCAL_PREF	Well-known discretionary	-	4-octet LOCAL_PREF	[RFC1771]
6	ATOMIC_AGGREGATE	Well-known discretionary	-	None	[RFC1771]
7	AGGREGATOR	Optional transitive	-	ASN and IP address of aggregator	[RFC1771]
8	COMMUNITIES	Optional transitive	-	4-octet community identifiers	[RFC1997]
9	ORIGINATOR_ID	Optional non-transitive	-	4-octet ROUTER_ID of originator	[RFC2796]
10	CLUSTER_LIST	Optional non-transitive	-	List of CLUSTER_IDs	[RFC2796]
11	DPA	Optional non-transitive	-	Destination Preference Attribute	expired draft
12	ADVERTISER	Optional non-transitive	-	BGP/IDRP Route Server	[RFC1863]
13	RCID_PATH/ CLUSTER_ID	Optional non-transitive	-	BGP/IDRP Route Server	[RFC1863]
14	MP_REACH_NLRI	Optional non-transitive	-	Multiprotocol Reachable NLRI	[RFC2858]
15	MP_UNREACH_NLRI	Optional non-transitive	-	Multiprotocol Unreachable NLRI	[RFC2858]
255	Reserved for development	-	-		[RFC2042]
-	WEIGHT	Local	-	2-octet integer value	Cisco

III. FINITE STATE MACHINE AND TIMERS

The operation of a BGP peer connection can be described in terms of a Finite State Machine (FSM) with six states. The transitions between the states are triggered by events, which cause certain actions to be performed and possibly a message to be sent. A simplified diagram of the BGP Finite State Machine is shown in Fig.1.

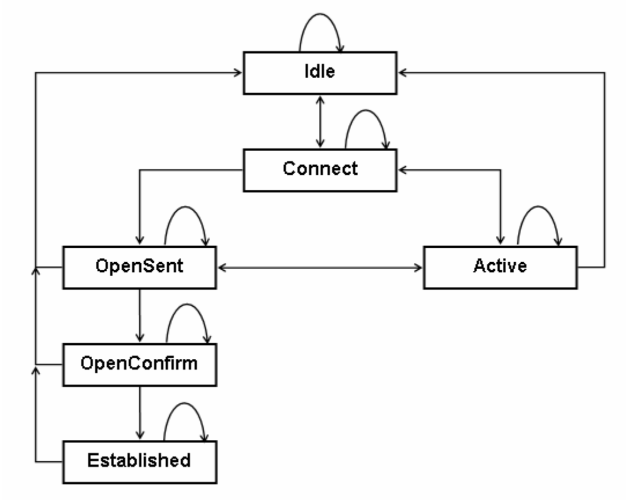


Fig.1 BGP Finite State Machine diagram

The BGP specifications describe three timers related to a peer connection:

- **ConnectRetry timer:** is used to measure the time spent trying to establish a TCP connection between BGP peers. When a BGP speaker tries to open a TCP connection to its peer, the ConnectRetry timer is started. If the connection is not established during the ConnectRetry interval, the timer is restarted and the speaker tries again to establish the TCP connection. When the TCP connection succeeds, the ConnectRetry timer is cleared and is no longer used. The value of the ConnectRetry timer should be large enough to allow establishing a TCP connection. The suggested value is 120 seconds [RFC1771].

- **Hold Timer**: the Hold Time is the maximum amount of time that may elapse between the receipts of successive KEEPALIVE or UPDATE messages. The Hold Timer is a timer that increments from 0 to the Hold Time. When a KEEPALIVE or UPDATE message is received, the Hold Timer is reset to 0. If the Hold Time for particular neighbour is exceeded, the connection is closed. The Hold Time for a particular peer connection is negotiated in the OPEN messages. The minimum acceptable value is 3 seconds. The value of 0 means that the Hold Timer is not used. [RFC1771] suggests the value of 90 seconds for the Hold Time.

- **KeepAlive timer**: the KeepAlive timer specifies the rate at which KEEPALIVE messages are sent. Its value should be chosen to ensure that the Hold Timer would not expire. When a BGP peer sends a KEEPALIVE or UPDATE message, it restarts the KeepAlive timer. The recommended KEEPALIVE rate is one-third of the negotiated Hold Time. If the Hold Time is 0, KEEPALIVE messages are not exchanged. [RFC1771] suggests the value of 30 seconds for the KeepAlive timer.

- **Start Timer**: While not required by the specifications, some BGP implementations (including the Zebra's `bgpd` and BGP++) use a **Start Timer** to automatically generate the Start event if a connection is in the Idle state. When the Start Timer expires, the Start event is generated. The Start Timer usually has a value of a few seconds, and its value is randomized (jittered) to avoid two peers generating the Start event at the same time and thus form two parallel connections.

IV. MULTI-PROTOCOL EXTENSIONS

BGP-4 was initially designed to carry routing information for IPv4 only. There are three pieces of information which are specific: NEXT_HOP (an IPv4 address), AGGREGATOR (an IPv4 address) and NLRI (IPv4 address prefixes). The version with multi-protocol extensions, sometimes referred to as MBGP *(Multi-protocol BGP)* or BGP-4+, was later defined by [RFC2283] and reviewed by [RFC2858]. It could handle routing information for other Network Layer protocols such as IPv6 or IPX. To provide backward compatibility while adding support for additional protocols to BGP-4, the existing fields and attributes have been preserved, and two new optional non-transitive attributes were defined (see also Table 1):

- MP_REACH_NLRI: used to carry a set of reachable destinations, along with the next hop information to be used for forwarding towards the given destinations

- MP_UNREACH_NLRI: used to carry a set of unreachable destinations. There is no need to carry next hop information in this case.

A BGP speaker that does not support Multi-protocol Extensions will ignore these attributes and not pass them along to its peers. Their use is negotiated between peer routers by the Capability Advertisement feature. Even if a BGP router supports Multi-protocol Extensions, it must have an IPv4 address, which is used (among other things) for the AGGREGATOR attribute. By the time the transition towards IPv6 will be complete, a new version of BGP might be issued. For the moment the Multi-protocol Extensions are needed to carry routing information for IPv6 and inter-domain multicast routing [RFC2545]. A newer application is to distribute label information for MPLS *(Multiprotocol Label Switching)*, as described in [RFC3107].

V. BGP-4+ SIMULATOR

The first aim of this paper was to study the Border Gateway Protocol version 4 *(BGP-4)*, the de-facto standard inter-domain routing protocol in the Internet. The approach chosen was simulation, using ns-2 extended with the BGP++ package. Network Simulator with BGP++ was installed and used under the GNU/Linux operating system (Fedora Core 3 distribution. By the time the experiments were realised the releases were the following: ns-2 2.26 and BGP++ 1.03a beta. The simplest way to obtain NS is to download the so-called "all-in-one" package, which includes all the software tools required to build and use NS: Tcl *(Tool Command Language)*, Tk *(ToolKit for designing user interfaces)*, Tclcl *(Tcl/C++ interface)*, and OTcl *(Object Tcl)*. It also includes `nam` *(Network AniMator)*, a tool for visualizing network simulations, and other useful software for network simulation. After unpacking the archives, the BGP++ patch files `patch_bgp++1.03a_pdnsv2` was copied to the `ns-allinone-2.26/ns-2.26/` directory, and the patch was applied with the command:

```
patch -p2 < patch_bgp++1.03a_pdns_v2
```

This command installs the BGP++ source code into the NS source code. After applying the patch and making some required changes in the configuration files, `ns` with BGP support was compiled by executing the `install` script from the `ns-allinone-2.26` directory. When the compilation is completed, the `ns` executables can be accessed through the symbolic links in the `ns-allinone-2.26/bin` directory. The documentation suggests that after installation, NS should be validated by running the supplied test scripts, to verify the correct operation of the simulator. This was done by running the `validate` script in the `ns-allinone-2.26/ns-2.26` directory.

a) Simulation of Two BGP Peer Routers

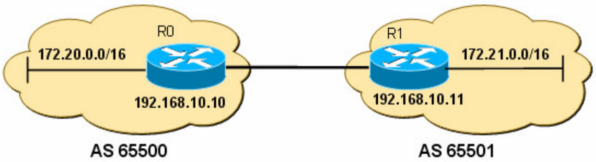


Fig.2 Two-Router scenario

The first step is to create a simulation `script 2peers.tcl` for the simplest study case: two BGP routers (called `router0` and `router1`) forming a peer connection. We suppose that the readers are already familiar with the use of ns-2. A brief introduction could be found in [Dob03]. Let us have a link to connect the two routers at 1.5 Mbps with 10 ms propagation delay and a DropTail queuing discipline, i.e. FIFO (*First In First Out*).

```
$ns duplex-link $router0 $router1 1.5Mb
10ms DropTail
```

The next step is to create BGP routers from the nodes. The following code creates a BGP instance called `bgp0`, and attaches it to the node `router0`:

```
set bgp0 [new Application/Route/Bgp]
$bgp0 register $r $bgp0 attach-node
$router0
```

The `bgp0` instance of BGP will use the `bgpd0.conf` configuration file from the directory specified by `opt(confdir)` variable. Note that R0 will be used to refer to both the node `router0` and the BGP instance `bgp0`. Similar, a BGP instance `bgp1` is created and attached to `router1`. By running the simulation, each BGP instance will create a log file, from which the operation of BGP can be examined. Network Simulator can create so-called `trace` files, containing the description of all packets processed during the simulation.

The simulator should be set to end the simulation after the specified number of seconds.

```
$ns at $opt(stop) "finish"
```

The above line tells the simulator to call the `finish` procedure at the number of seconds specified by `opt(stop)` variable. The role of the `finish` procedure is to stop the simulation, close the trace files, and run `nam`, specifying the name of the `nam` trace-file to use as input. Finally, the simulation must be started by specifying the following command at the end of the script: `$ns run`

Let us comment now the very first experimental results to understand the way the designed simulator has been used. The given scenario supposed that at moment 7.008645, R1's `Start timer` expires and tries to open a TCP connection to R0, changing from the `Idle` to the `Connect` state:

```
bgpd1.log:
7.008645 BGP: 192.168.10.10 [FSM] Timer
(start timer expire).
7.008645 BGP: 192.168.10.10 [FSM]
BGP_Start (Idle->Connect)
7.008645 BGP: 192.168.10.10 went from
Idle to Connect
7.008645 BGP: 192.168.10.10 [Event]
Connect start to 192.168.10.10
```

The first TCP segment sent to open the connection, as displayed in `nam` is shown in Fig.3. At moment 7.029072 the TCP connection was established, R1 changes to the `OpenSent` state and sends the OPEN message:

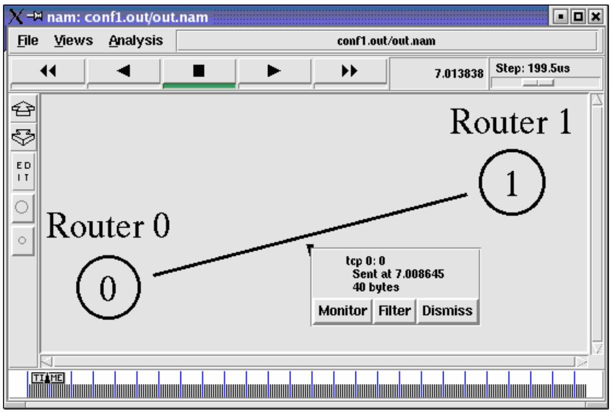


Fig.3. R1 initiates transport connection

```
bgpd1.log:
7.029072 BGP: 192.168.10.10 [FSM]
TCP_connection_open (Connect->OpenSent)
7.029072 BGP: 192.168.10.10 went from
Connect to OpenSent
7.029072 BGP: 192.168.10.10 sending
OPEN, version 4, my as 65501, holdtime
180, id 192.168.10.11
7.029072 BGP: 192.168.10.10 send message
type 1, length (incl. header) 45
```

R0 does not accept the connection, because its `Start timer` did not expire yet and it is still in the `Idle` state (the `Start` event was not generated):

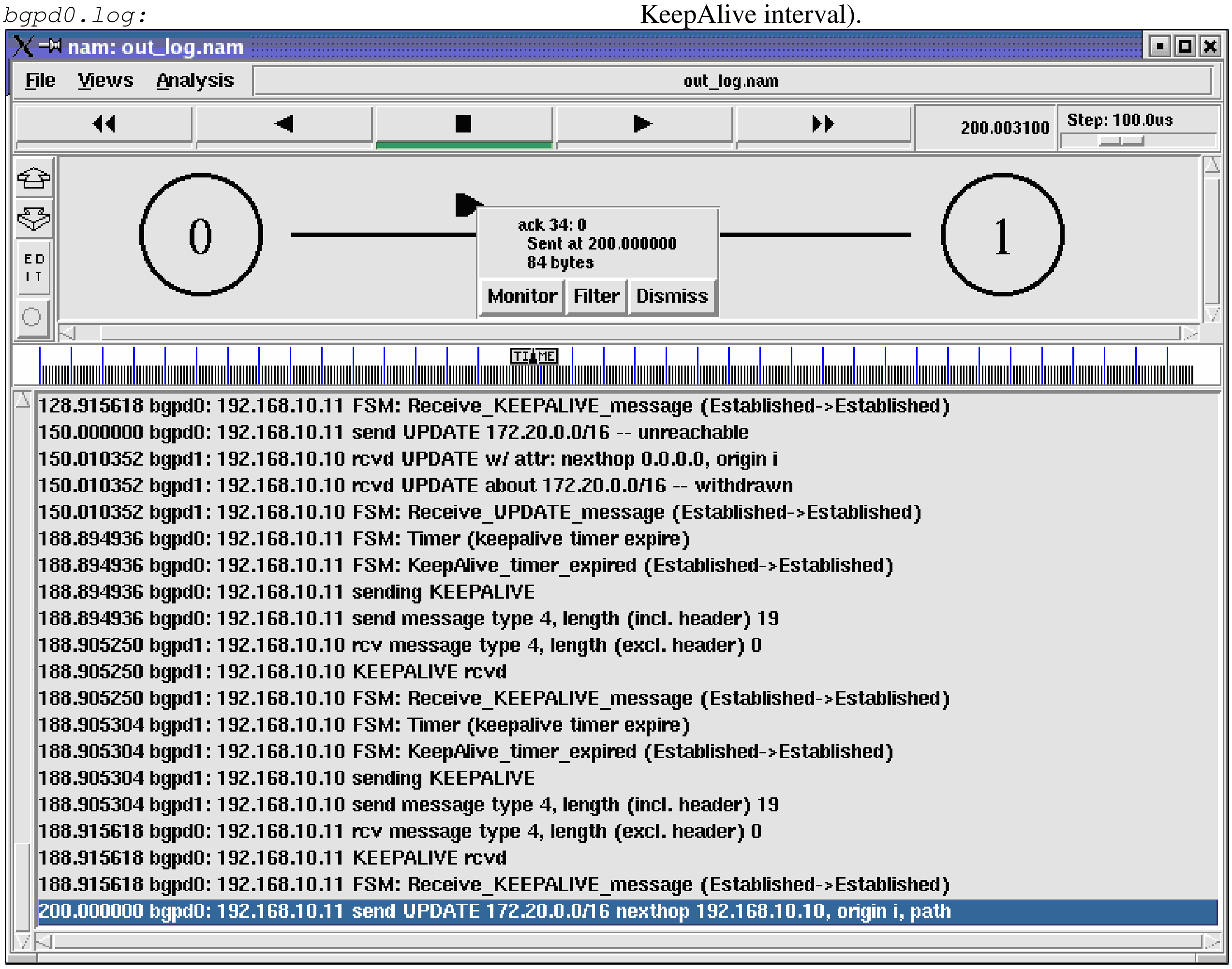
```
bgpd0.log:
7.039285 BGP: [Event] BGP connection
from host 192.168.10.11
7.039285 BGP: [Event] BGP connection IP
address 192.168.10.11 is Idle state
```

At moment 7.049499, R1 receives the `TCP connection closed` event; therefore it changes from `OpenSent` to the `Active` state. Retrying successful BGP connections, the influence of the link is presented within Table 2.

Table 2. BGP connection setup time versus link parameters

Rate [Mbps]	Connection setup time [sec] (1 ms delay)	Connection setup time [sec] (10 ms delay)	Connection setup [sec] (100 ms)
0.064	0.0627094	0.116	0.652842
0.128	0.034	0.088	0.628
0.256	0.02	0.074	0.614
0.512	0.013	0.067125	0.607
2.048	0.00775	0.06175	0.60175
5	0.0067168	0.0607168	0.6007166
10	0.0063582	0.0603586	0.6003584
50	0.0060718	0.0600718	0.6000718
100	0.0060358	0.060036	0.6000356

Fig.4 presents an example of a sucessful exchange of routing information (UPDATE) which could be used for performance evaluation (for example the proper KeepAlive interval).



```
128.915618 bgpd0: 192.168.10.11 FSM: Receive_KEEPALIVE_message (Established->Established)
150.000000 bgpd0: 192.168.10.11 send UPDATE 172.20.0.0/16 -- unreachable
150.010352 bgpd1: 192.168.10.10 rcvd UPDATE w/ attr: nexthop 0.0.0.0, origin i
150.010352 bgpd1: 192.168.10.10 rcvd UPDATE about 172.20.0.0/16 -- withdrawn
150.010352 bgpd1: 192.168.10.10 FSM: Receive_UPDATE_message (Established->Established)
188.894936 bgpd0: 192.168.10.11 FSM: Timer (keepalive timer expire)
188.894936 bgpd0: 192.168.10.11 FSM: KeepAlive_timer_expired (Established->Established)
188.894936 bgpd0: 192.168.10.11 sending KEEPALIVE
188.894936 bgpd0: 192.168.10.11 send message type 4, length (incl. header) 19
188.905250 bgpd1: 192.168.10.10 rcv message type 4, length (excl. header) 0
188.905250 bgpd1: 192.168.10.10 KEEPALIVE rcvd
188.905250 bgpd1: 192.168.10.10 FSM: Receive_KEEPALIVE_message (Established->Established)
188.905304 bgpd1: 192.168.10.10 FSM: Timer (keepalive timer expire)
188.905304 bgpd1: 192.168.10.10 FSM: KeepAlive_timer_expired (Established->Established)
188.905304 bgpd1: 192.168.10.10 sending KEEPALIVE
188.905304 bgpd1: 192.168.10.10 send message type 4, length (incl. header) 19
188.915618 bgpd0: 192.168.10.11 rcv message type 4, length (excl. header) 0
188.915618 bgpd0: 192.168.10.11 KEEPALIVE rcvd
188.915618 bgpd0: 192.168.10.11 FSM: Receive_KEEPALIVE_message (Established->Established)
200.000000 bgpd0: 192.168.10.11 send UPDATE 172.20.0.0/16 nexthop 192.168.10.10, origin i, path
```

Fig.4 Example of simulation: R0 sends UPDATE message

b) Simulation of Three BGP Peer Routers

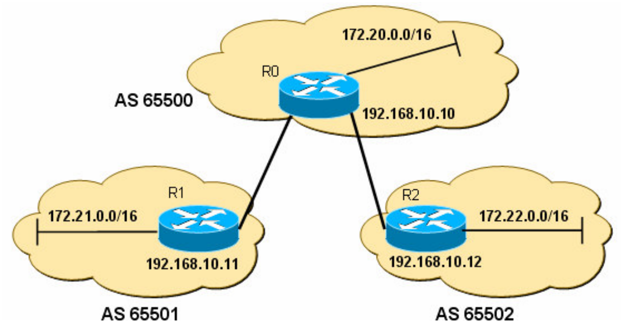


Fig.5 Three-Router scenario

Three-router experiments envisaged the understanding of BGP route selection based on attributes (NEXT_HOP, AS_PATH, WEIGHT etc.) and on routing policies. Taking into account the previous simulation, with the configuration R0 (AS 65500) provides a transit service between AS 65501 and AS 65502 (R1 and R2). Supposing that AS 65500 does not want to offer transit for traffic coming from AS 65502 and going towards the network `172.21.0.0/16` (in AS 65501), a routing policy must be implemented. Since this decision is taken in AS 65500 (R0), only R0's configuration file should be modified. The above described policy can be implemented at R0 by not advertising the network address `172.21.0.0/16` to R2. A routing policy, either referring to incoming or outgoing advertisements, is implemented with a route map. Supposing that the route map implemented at R0 is called `RM1`, the following command is added to `bgpd0.conf` (after the `neighbor` commands):

```
neighbor 192.168.10.12 route-map RM1 out
```

The command specifies that for peer `192.168.10.12` (R2) the route map for outgoing advertisements is `RM1`. The next step is implementing the route map itself, by creating an access list. It was denoted with the number `1`, and specifies that the IP address `172.21.0.0/16` is not accepted, but any other address prefix is permitted.

```
access-list 1 deny 172.21.0.0/16
access-list 1 permit any
```

Now the access list `1` can be attached to the route map

```
route-map RM1 permit 1
match ip address 1
```

VI. CONCLUSIONS

Several simulations based on Network Simulator `ns-2.26`, with BGP++ 1.03a beta extension, under Linux RedHat 9.0/ Fedora Core 3 or later were carried out. During both establishing phase and in case of a route oscillation BGP-4 routing information exchanged included network number, autonomous systems' path and attributes. Two neighboring routers established a TCP connection before sending BGP updates. Several OTcl and Linux shell scripts were written, as well as router configuration files in order to obtain the results from `log` files. The paper studied the BGP connection setup time depending on `ns-2` parameters (propagation delay and transfer rate). The simulation of BGP proved that the Hold Timer is used to detect failures. Preliminary tests showed that the connection setup time is at least three times greater than RTT (Round-Trip Time). The most important attribute in selecting the best route is by default AS_PATH. Measurement of data traffic and link utilisation revealed that about 70 % of bytes are related to TCP/IP headers, the remaining being BGP messages.

Practical implementations of BGP-4+ involving `bgpd` under Linux are under progress.

REFERENCES:

[Bat00a] T. Bates, R. Chandra, E. Chen, "BGP Route Reflection – An Alternative to Full Mesh IBGP", *RFC 2796*, 2000

[Bat00b] T. Bates, Y. Rekhter, R. Chandra, D. Katz, "Multiprotocol Extensions for BGP-4", *RFC 2858*, 2000

[Cha96] R. Chandra, P. Traina, T. Li, "BGP Communities Attribute", *RFC 1997*, August 1996

[Che00] E. Chen, "Route Refresh Capability for BGP-4", *RFC 2918*, September 2000

[Chu04] J. Chung & M. Claypool, *NS by Example*. Worchester Polytechnic Institute, http://nile.wpi.edu/NS/

[Cis03] ***, Internetworking Technologies Handbook, Cisco Systems, 2003

[Dob03] V. Dobrota, *Digital Networks in Telecommunications. Volume 3: OSI and TCP/IP*, Second Edition, Mediamira Science Publishers, Cluj-Napoca 2003 (in Romanian)

[Doy01] J. Doyle & J. Dehaven Caroll, *Routing TCP/IP. Volume II*. (CCIE Professional Development). Cisco Press, 2001

[Fal03] K. Fall, K. Varadhan, *The ns Manual.* The VINT Project, 2003, http://www.isi.edu/nsnam/ns/ns-documentation.html

[Fly03] C. Flynt, *Tcl/Tk. A Developer's Guide*. Second Edition. Morgan Kaufman Publishers, 2003

[Hal00] S. Halabi & D. McPherson, *Internet Routing Architectures*. Second Edition. Cisco Press, 2000 [RFC1771] Y. Rekhter & T. Li , "A Border Gateway Protocol 4 (BGP-4)", *RFC 1771*, March 1995

[Man97] B. Manning, "Registering New BGP Attribute Types", *RFC 2042*, January 1997

[Mar99] P. Marques, F. Dupont, "Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing", *RFC 2545*, March 1999

[McP02] D. McPherson, V. Gill, D. Walton, A. Retana, "Border Gateway Protocol (BGP) Persistent Route Oscillation Condition", *RFC 3345*, August 2002

[Rek01] Y. Rekhter, E. Rosen, "Carrying Label Information in BGP-4", *RFC 3107*, May 2001

[Tra95] P. Traina, "BGP-4 Protocol Analysis", *RFC 1774*, 1995

[Tra01] P. Traina, D. McPherson, J. Scudder, "Autonomous System Confederations for BGP", *RFC 3065*, 2001