Everything You Always Wanted to Know About BGPv4 in OMNeT++ But Were Afraid to Ask

Vladimír Veselý[1]*, Marcel Marek[2], Kamil Jeřábek[1], and Adrián Novák[3]

[1] Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic
{veselyv,ijerabek}@fit.vutbr.cz

[2] The Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway
marcelma@ifi.uio.no

[3] Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic
adriannovak94@gmail.com

Abstract

Border Gateway Protocol is one and only one exterior gateway protocol for routing between autonomous systems, which basically glues the Internet together. This paper outlines BGPv4 theory and its (re)implementations in OMNeT++ discrete event simulator. This effort extends IPv6 capabilities of INET4 framework and improves the accuracy of relevant simulation models.

1 Introduction

The Internet consists of autonomous systems (AS), where each one represents independent collection of devices and networks under the common administrative domain. AS are uniquely identified by autonomous system number (ASN, which is 16 or 32 bits long identifier [1]). Inside the AS, interior gateway protocols (such as distance-vector or link-state) are guaranteeing level of convergence, which reflects changes fast. Outside the AS, exterior gateway protocols (EGPs) maintains routing for default-free zone (DFZ, i.e., the backbone of the Internet). AS is treated as a single undivided entity from the perspective of global routing. EGPs consider the route as the list of ASN (hence, these protocols are often referred as path-vector routing protocols). EGP or path-vector are remarked as categories but they in fact currently contain only a single representative – Border Gateway Protocol (BGP).

Even though RFC 2460 [2] is more than 20 years old, IPv6 adoption is still taking time to gain significant momentum. It depends on the point of view [3] how to quantify IPv6 penetration (8.5% of web sites, 15.8% of mail servers, please see [4] for details), but it is safe to say that IPv6 availability in 2019 is probably lower than authors (and other proponents) of IPv6 would hope for. However, IPv6 is still the one and only widely accepted option that is going to substitute IPv4 and its addressing limitations. Hence, various service providers (including ISPs, content cachers, service deliverers and many others) are working on necessary changes in order to prepare their infrastructure and applications for IPv6. A significant part of this effort is invested into the routing of IPv6 traffic, which would be impossible without appropriate routing protocols. Many protocols support IPv6 address family – OSPFv3, EIGRP, IS-IS, Babel, RIPng, M-BGP. However, only some of them – namely OSPF and BGP – receive recognition as a standard tools of LAN/WAN design [5].

General idea driving the development of simulation software is to create a tool which: a) imitates a system behavior under various conditions; and b) subsequently generates accurate

*Corresponding author

and reliable results. We believe that OMNeT++ is such a tool and its INET framework is great for simulations of computer networks. Primary motivation of this paper is to enhance IPv6 support within INET4 framework. We planned to achieve this goal by improving functionality of widely used routing protocol BGP. This effort is a part of long-term running project (called ANSA) at Brno University of Technology, which focuses on implementation of various computer networking protocols for OMNeT++.

The paper has the following structure. Section 2 explains basic theory behind BGP protocols. Section 3 describes the current state in the frame of INET4 and outlines design and implementation notes of the source contribution. Section 4 demonstrates correct behavior of simulation models on scenarios also compared with referential Cisco implementation of BGP. Section 5 summarizes achievements of this paper together with a brief overview of future plans.

2 Principles

This section briefly overviews existing BGP protocol and its operational principles.

BGP is a path-vector routing protocol, which distributes classless routing information (mostly) between AS. BGP is useful for dual-homed or (dual-)multi-homed AS, where routing information needs to be propagated to the DFZ. BGP offers summarization and authentication (special TCP option with MD5-HMAC).

BGP peers or neighbors are routers running instance of BGP and exchanging routes with each other. BGP peering comes with two modes: a) external BGP (eBGP), where routers are in different autonomous systems; and b) internal BGP (iBGP), where both routers are in the same AS. In the case of eBGP, peers are usually adjacent (one hop away) and exchange all routing information. However, iBGP peers exchange only directly connected networks (i.e., prefixes originating in the same AS). This restriction works as built-in protection against routing loops for iBGP that also yields full-mesh connectivity between iBGP peers.

BGP transfers messages reliably with the help of TCP as the transport protocol. Hence, BGP was assigned with well-known port 179 as the application over TCP. Despite the existence of multiple versions of BGP, only the newest one BGPv4 is being used (see RFC 4271 [6]).

However, numerous RFCs are extending protocol capabilities (see RFC 4760 [7]) of this version, and they offer MPLS, VPN, multi-address family, or multicast support. The BGP models network as a graph, where vertices represent autonomous systems and edges represent connections between them. These vertices include a list of destination networks contained within AS. Series of edges then represent suitable paths to reach destinations, where paths include various attributes such as next-hop, AS-path (a list of autonomous system numbers), local preference (announced exit point from AS), multi-exit discriminator (preferred entry point to AS) and others. By (re)configuring these attributes, BGP can enforce inbound and outbound routing policies that affect how traffic is routed to/from/via the AS.

BGP protocol uses the following message types:

- *Open* as the initial message, which is sent after TCP connection is established between peers;
- *KeepAlive* that is periodically sent as a heartbeat, which signals that peer is working correctly. *KeepAlive* is sent every 60 s by default and neighbor is willing to wait generally three times more (so-called *HoldDown* timer, which is 180 s by default) before declaring the router down;
- *Update* message carries routing information for a single path (unique route across AS), which includes attributes (mentioned above), the list of reachable destination networks

(using this path) and optionally the list of withdrawn destinations;
- *Notification* is generated whenever some problem with BGP occurs and terminates BGP peering. *Notification* contains error code and more verbose error description.

After the initial synchronization, BGP peers send only incremental updates (i.e., new or withdrawn routes) because this approach is more scalable than a periodic exchange of the whole routing tables (which in the case of DFZ router with many peers may be GBs of routing data).

3 Implementation

This section discusses current development state of INET framework pursuant to above-mentioned routing protocols. Moreover, the section briefly covers the design of enhanced simulation model for BGPv4.

3.1 INET

Open-source framework INET helps academia and industry to simulate computer networks. INET model library spans through the whole network stack and includes models for Ethernet, WiFi, IPv4/IPv6, TCP/UDP/SCTP and plethora of routing and application protocols. ANSAINET is a fork of INET that further extends capabilities and available models of the original framework. ANSAINET feature-set focuses on the simulation of wired enterprise networks.

We are aware of Mani Amoozadeh contributions to BGPv4 and OSPFv2 in INET 4.1 [8] that happened in January 2019. We coordinated our work with him on dedicated teleconference during winter 2018. We understood that his major interest was features related to split-horizon, next-hop and couple of protocol fixes. Whereabouts, our main concern was multi-address family support and major revisit of BGP FSM plus protocol details (including not only checksum but also missing message types). We believe that both his and our work can be successfully merged.

Work on BGP implementation described in this paper has started in October 2018, thus, our starting point was source base of INET 4.0 (more specifically inet-4.0.0-576b2dc). This INET still includes a lot of code provided by Helene Lageber that dates back to 2010. We identified the following list of problems in above-mentioned version of INET4:

P1) Peer is unable to advertise more than one network prefix to its neighbor. *Update* message contains only one network prefix, and a neighboring peer is unable to process more prefixes. Hence, it is currently impossible to announce more network prefixes between peers.
P2) The current implementation does not support IPv6 prefixes, only IPv4.
P3) AS Path appending adds ASN even in cases of iBGP, which should not happen. Moreover, current iBGP peers are willing to exchange all routing information (both learned and directly connected prefixes), which violates an important operational condition of BGP and leads to routing instabilities in certain topologies.
P4) If a router has active eBGP peering, then it is impossible to form any iBGP with other potential routers.
P5) *ConnectRetry* timer is not being reset correctly. When this timer expires, BGP routers lose TCP connection and stop being peers.
P6) Implementation (namely the state of BGP peering) is unable to react on link flapping. It is neither able to tear down the neighborship, nor to re-establish it.

P7) *Notification* message is not implemented at all, and there are inconsistencies in the finite state machine of BGP operation.

The main goal of our contribution is to support IPv6 in BGPv4 implementation in OMNeT++. Nevertheless, we have also decided to amend all issues and problems discovered during the assessment of source code base.

3.2 Design

BGPv4 is designed as a simple module named `BGP` and it is available in the following folder `src/inet/routing/bgpv4`.

We extended appropriate message definition files with new headers and fields - namely multiprotocol capability indicators, network (un)reachability and address-family specifiers, in order to address P1, P2 and P7. Moreover, we have added appropriate version of all BGP messages for IPv6 address-family (if IPv6 is used as a carrier protocol for peering).

BGP tracks the status of TCP connection and neighborship with every peer in `BGPSession`. We extended this abstract data structure with information whether the neighborship leverages IPv4 or IPv6 (resolving problem P2) and which type of peering is employed (resolving P4).

We have also implemented all missing parts of the finite state machine (FSM) in `BgpFsm` as well as event listeners in order to deal with P2, P3, P5 and P6. Our contribution covers all standard BGP states and appropriate transitions between them:

- *Idle* – Beginning state of any neigborship and a fallback state for BGP related errors.
- *Connect* – Router creates TCP connection with its peer, or more accurately, the peer with higher IP address initiates TCP connection while the other peer listens for it.
- *Active* – If TCP connection can not be established, the router resets *ConnectRetry* timer during this state and repeats connection attempt. This state is used as a fallback for TCP related errors.
- *OpenSent* – TCP connection was successfully established (i.e., communicating peers conducted 3-way handshake). The *Open* message is sent to validate the basic settings of BGP peering. Following properties must match between potential peers: BGP version, unique router identifier, autonomous system numbers, peering IP addresses, optionally authentication or time-to-live value.
- *OpenConfirm* – The router received *Open* also from a neighbor and now it waits for first *KeepAlive*, which states that basic settings is fine. To avoid endless waiting for response, *HoldDown* timer is started. If *KeepAlive* is received, then the peer is ready to exchange network prefixes.
- *Established* – Peers exchange routing information using *Updates*. *HoldDown* is reset with every *KeepAlive* or *Update* to monitor peer's heartbeat.

Detail description of the implemented functionality and BGP is beyond the scope of this paper. Nevertheless, we can recommend the official CCIE certification book [9], which we consider as a great compendium of BGP knowledge.

Since `omnetpp.ini` has limitations in passing more structured configuration to models, we achieved this by creating dedicated XML structure (which can bootstrap abstract data structures). This XML is initiated via a reference in `omnetpp.ini` and parsed upon startup of the simulation. Apart from configuring simple values to certain parameters (e.g., HoldDown timer, KeepAlive), it allows robust setup of router's state (e.g., route filters containing lists of networks, which can be denied/accepted during inbound/outbound traffic engineering, session parameters for neighboring routers).

4 Testing

In this section, we focus on testing of our code extension of BGPv4 in INET framework. The extension covers missing functionality as described above (most notably IPv6 support, more prefixes in *Updates*, reaction to the failure/recovery). The chosen testing scenario primarily demonstrates and tests *multi address-family* routing functionality that covers both IPv6 and IPv4 routing. The behavior of the new simulation model is compared with the referential Cisco implementation (IOS version 15.4), which is compliant with the standard and outlined in [9].

4.1 Topology

The testing topology depicted in Figure 1 contains four routers `R1-R4`. Each router has configured loopback interface `Loopback0` with specific IPv4 and IPv6 addresses that can be seen in the Figure 1 above each router. The router `R1` is part of AS 100. The routers `R2` and `R3` are part of AS 200, and internal neighborship is established between them. We use static routing inside AS 200 to simplify the scenario. The router `R4` is part of AS 300. External neighborship is established between routers `R1-R2` and `R3`-R4. All routers negotiate their `Loopback0` IPv4 and IPv6 addresses.

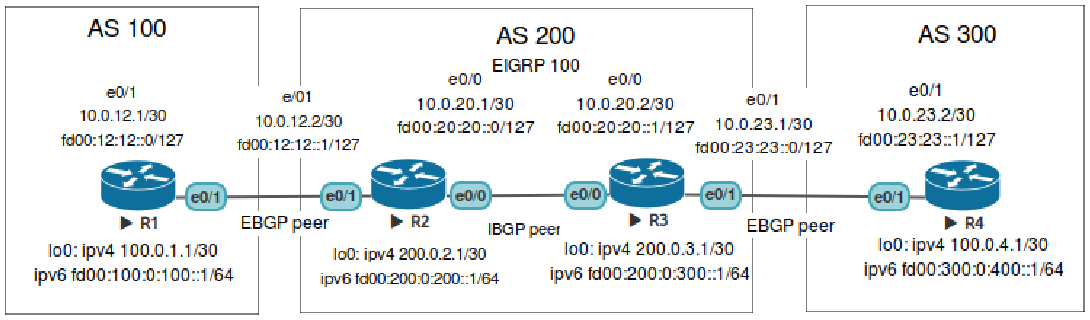

Figure 1: BGP testing topology.

The example can be found in the directory `examples/bgpv4/BgpEx1Loopback` and it contains three simulation configurations, where for the cause of this paper is relevant the one called *LinkDownAndUp*.

4.2 Connection Establishment

In this part of the scenario, we cover neighborship establishment from the perspective of router `R1` communicating with router `R2`. It is the first connection made in the scenario/experiment. The messages produced during the connection establishment phase are depicted in Table 1. The table shows comparison of message exchange in simulation and in referential network. The table shows network simulation message exchange in comparison to real network message exchange. The order of messages in the referential implemenetation differs because TCP connections for IPv4 and IPv6 are initiated in parallel.

The establishment phase in the simulation environment goes as follows:

#1A) At first, router `R1` creates a TCP connection for IPv4 to router `R2`. Router `R1` sends *Open* message with initial BGP information. Router `R2` sends a positive response to *Open*. The *KeepAlive* message exchange follows.

#2A) Router `R1` then sends *Update* message with known routes (in this case only the route to the network of the `Loopback0` is negotiated). In response, the router `R2` sends *Update* with its known routes.
#3A) Router `R1` creates another TCP connection for IPv6 to router `R2`. The BGP message exchange for IPv6 is similar to the IPv4 one.
#4A) Router `R1` receives only one *Update* for each TCP connection with information regarding the rest of the network, because the rest of the routers exchange BGP information later then `R1` and `R2`.

More *KeepAlive* messages follow after but they are not included in Table 1. Timestamps included in this and subsequent tables represent real timestamps as recorded by the simulation log or PCAP file. Therefore, you can observe slight discrepancies because it is hard to align events of the simulated and real control plane. If you focus on message order, then you check the same transition through BGP finite state machine, which ends with the same content of routing tables. Message order may be slightly different because of packet pacing and control plane delay processing in case of the referential implementation.

Simulation			**Referential implementation**		
Phase	**Time**	**Message**	**Phase**	**Time**	**Message**
#1A	17.000	→ *Open*	#1A	54.621	→ *Open*
#1A	17.000	← *Open*	#3A	54.622	→ *Open6*
#1A	17.000	→ *KeepAlive*	#1A	54.623	← *Open*
#1A	17.000	← *KeepAlive*	#1A	54.623	→ *KeepAlive*
#2A	17.000	→ *Update*	#1A	54.623	← *KeepAlive*
#2A	17.000	← *Update*	#3A	54.626	← *Open6*
#3A	17.500	→ *Open6*	#3A	54.626	← *KeepAlive6*
#3A	17.500	← *Open6*	#2A	54.629	→ *Update*
#3A	17.500	→ *KeepAlive6*	#3A	54.630	→ *KeepAlive6*
#3A	17.500	← *KeepAlive6*	#2A	54.630	← *Update*
#3A	17.500	→ *Update6*	#3A	54.635	→ *Update6*
#3A	17.500	← *Update6*	#3A	54.635	← *Update6*
#4A	21.000	← *Update*			
#4A	26.000	← *Update6*			

Table 1: BGP connection establishment messages captured from router `R1`.

4.3 Connection Failure

In this section, we demonstrate the behaviour after a link failure. The link interconnecting router `R1` and `R2` is removed. Table 2 provides message exchange from the perspective of router `R2`.

#1B) *KeepAlives* from both `R1` and `R2` do not reach each other. *HoldDown* timer is set to 180 s (which is recommended value). *KeepAlive* messages normally reset the timer. When the timer expires, the router `R2` removes peer routes from its routing table. The router `R2` immediately generates *Update* messages and sends them to its peer router `R3`.
#2B) The router `R3` receives *Update* for IPv4, removes the route from its routing table and sends *Update* to router `R4`. Similar process repeats also for IPv6.

In the simulation, the last *KeepAlive* message from `R1` to `R2` gets through at $t = 17.0$ s.

HoldDown timer expired three minutes (i.e., 180 s) since then at $t = 197.0$ s. R2 sends *Update* to R3 for IPv4 and 0.5 s later *Update6* for IPv6. In the referential implementation, the link shuts down at around $t = 35$ s relative to the packet capture start. The time gap between *Update* and *Update6* message is that the experiment was running for some time and the timers diverged.

The reason why there is a gap of 50 s (which is roughly equal to the value of *KeepAlive* timer) between the two *Update* is because IPv4 *KeepAlive* was received just before the link was shut down. Therefore, the *Update6* was sent sooner at $t = 165$ s and *Update* at $t = 214$ s.

Simulation			**Referential implementation**		
Phase	**Time**	**Message**	**Phase**	**Time**	**Message**
#1B	197.000	→ *Update*	#1B	165.287	→ *Update6*
#1B	197.500	→ *Update6*	#1B	214.641	→ *Update*

Table 2: BGP connection failure messages captured from router R2.

4.4 Connection Recovery

In this section, we demonstrate the behaviour after a link becomes available again. The link between router R1 and router R2 is recovered. The router R1 and R2 are in the *Idle* state. The neighborship establishment between those routers proceeds as described in section 4.2. Table 3 depicts the message exchange both for simulation and referential implementation.

#1C) The router can send all prefixes in the first *Update* message. Then only *KeepAlive* messages are exchanged.

The difference between message exchange described in section 4.2 and here occurs because there is only one *Update* needed from the router R2. Router R2 has already all the information about the topology.

Simulation			**Referential implementation**		
Phase	**Time**	**Message**	**Phase**	**Time**	**Message**
#1A	217.000	→ *Open*	#1A	118.576	← *Open*
#1A	217.000	← *Open*	#3A	118.577	← *Open6*
#1A	217.000	→ *KeepAlive*	#1A	118.578	→ *Open*
#1A	217.000	← *KeepAlive*	#1A	118.579	← *KeepAlive*
#1C	217.000	→ *Update*	#1A	118.579	→ *KeepAlive*
#1C	217.000	← *Update*	#3A	118.579	→ *Open6*
#3A	222.500	→ *Open6*	#3A	118.581	→ *KeepAlive6*
#3A	222.500	← *Open6*	#3A	118.584	← *KeepAlive6*
#3A	222.500	→ *KeepAlive*	#1C	118.585	→ *Update*
#3A	222.500	← *KeepAlive6*	#1C	118.585	← *Update*
#1C	222.500	→ *Update6*	#1C	118.591	→ *Update6*
#1C	222.500	← *Update6*	#1C	118.591	← *Update6*

Table 3: BGP connection recovery messages captured from router R1.

5 Conclusion

In this paper, we thoroughly described widely deployed protocol for intra (and in some cases also inter) autonomous system routing – BGPv4. We had revisited and significantly improved simulation models of this protocol within INET framework. We created new test scenarios checking both old and new functionalities. We have validated the functionality of our simulation model against referential implementation. Compared message orders yield: a) same outcomes in routing table content; and b) same transitions through BGP finite state machines.

It is important to support IPv6 by providing implementation in computer network simulators despite scarce deployment of this network protocol in production. We believe that our contribution enhances capabilities of (ANSA)INET framework. It makes BGPv4 simulations more accurate and thus more reliable in cases when OMNeT++ is being used for network design validations.

More information about the ANSAINET project is available on the homepage [10]. All source codes (BGPv4 simulation models) can be downloaded from the GitHub repository [11]. We hope that these contributions can be submitted as pull request into official INET. We have also prepared results reproduction webpage, which includes additional supporting materials [12].

This work was supported by the Brno University of Technology organization and by the research grant FIT-S-17-3964 and as part of the OCARINA research project [13], which develops a combined congestion control and routing solution for the Recursive InterNetworking Architecture.

References

[1] John A. Hawkinson and Tony J. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930, March 1996.

[2] Bob Hinden and Steve E. Deering. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[3] Matěj Grégr, Tomáš Podermański, and Miroslav Švéda. Measuring quality and penetration of ipv6 services. *ICNS'14*, pages 96–101, 2014.

[4] Matěj Grégr and Tomáš Podermański. Live statistics — 6lab.cz. http://6lab.cz/live-statistics, accessed: July 2019.

[5] David Bauer, Murat Yuksel, Christopher Carothers, and Shivkumar Kalyanaraman. A case study in understanding ospf and bgp interactions using efficient experiment design. In *20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06)*, pages 158–165. IEEE, 2006.

[6] Yakov Rekhter, Susan Hares, and Tony Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006.

[7] Tony J. Bates, Ravi Chandra, Yakov Rekhter, and Dave Katz. Multiprotocol Extensions for BGP-4. RFC 4760, January 2007.

[8] Mani Amoozadeh. INET Framework for OMNEST/OMNeT++. https://github.com/ManiAm/inet/tree/INETex/src/inet/routing/bgpv4, accessed: August 2019.

[9] Vinit Jain and Bradley Edgeworth. *Troubleshooting BGP: A Practical Guide to Understanding and Troubleshooting BGP*. Cisco Press, 2016.

[10] Brno University of Technology. ANSA project. http://ansa.omnetpp.org, accessed: July 2019.

[11] Brno University of Technology. ANSAINET repository. https://github.com/ANSA/inet/tree/bgp-multi-address-family, accessed: July 2019.

[12] Brno University of Technology. OMNeT++ Summit 2019 results reproduction. `https://github.com/ANSA/results-reproduction/wiki/OMNeT---Community-Summit-2019`, accessed: July 2019.

[13] University of Oslo. OCARINA: Optimizations to compel adoption of rina - department of informatics. `https://www.mn.uio.no/ifi/english/research/projects/ocarina`, accessed: July 2019.