BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2022
http://excel.fit.vutbr.cz

Modeling of Border Gateway Protocol

Jan Zavřel*

Abstract
Border Gateway Protocol (BGP) was first sketched on two napkins over a conference lunch in 1989 by two gentlemen, and it would become one of — if not the most — essential routing protocols of our time in just a few years. It creates the global routing system of the Internet. Billions of people rely on BGP every single day without even knowing about it. A recent example of our collective reliance on BGP takes us back a few months ago when Facebook engineers misconfigured their BGP instances, effectively cutting themselves off from the outside [1]. The global importance of this protocol led to the creation of this paper, which discusses improvements to the BGP simulation model. Such a model, if of high quality, could help network engineers and others test the stability of their topologies and configurations inside a safe discrete environment. The model, written in C++, is improved and extended in several directions with new features, such as full support for the IPv6 address family, Cisco-like configuration, the BGP table, TCP improvements, and many more. The quality of the model is ensured by a close comparison of all aspects of the model to Cisco's implementation of BGP.

Keywords: BGP — simulation — routing protocol — OMNeT++ — INET

Supplementary Material: Author's version of BGP simulation model, INET version of BGP simulation model, Novák's version of BGP simulation model

*xzavre10@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

With BGP being the only EGP protocol to tie tens of thousands of Autonomous Systems (AS) together, deep knowledge of this protocol can be significantly beneficial to most network engineers. Especially when even a small mistake can render services, which millions of people rely on, completely unavailable. Minor BGP configuration errors occur daily and cause global impacts of a wide variety of severity [2]. And that is just the tip of the iceberg, as there are many topology design decisions, which are made by ISPs and other organizations, that could cause unnecessary service outages [3]. BGP forwarding information base (FIB) table is still steadily growing, as shown in Figure 1, with each entry representing one advertised prefix by some public AS. One way of getting closely familiar with a complex protocol such as BGP is through experimentation on a simulation model. If such a model reliably reflects reality, behavior patterns during critical scenarios can be observed and analyzed within a safe and risk-free environment. The results can be taken into account during the next topology design iteration, leading to a more stable and optimal design. However, simulation is not the only way to experiment with BGP.

There are currently three paths (two comfortable and one problematic) that can be taken if one is interested in conducting experiments with BGP. Let us

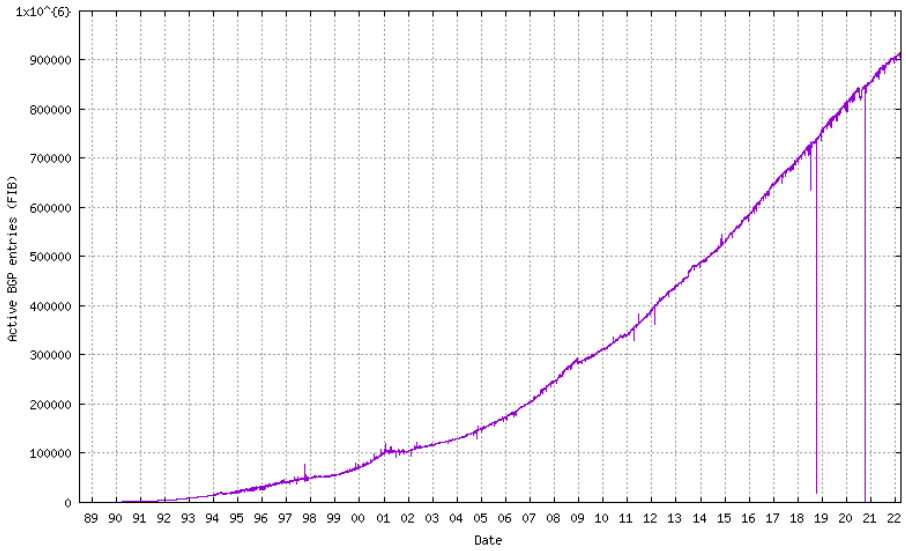

Figure 1. Active IPv4 BGP entries over years, currently sitting at 916,649. IPv6 is currently at 154,022 entries. Image and data sourced from [4, 5].

start with the problematic one: experiments on real devices. This solution lacks scalability and is also very expensive. Software on these devices, including the implementation of routing protocols, is often not open-source, and because of that, such device is more like a black box with the user hoping that it behaves accurately according to its specification. The benefit of choosing real devices is that their implemented protocols are often effective and, most importantly, one does not have to worry about the quality of any model or emulation discrepancy. The other two ways of conducting experiments are leveraging either emulators or simulators. Emulators such as GNS3 [6] or EVE-ng [7] can unpack and run binary firmware images found also on real devices. This solution, in comparison to real devices, substantially cuts costs, but still lacks transparency. Simulators, on the other hand, can often offer transparency together with source code-level customizability, which greatly encourages its users to become invested and experiment on such models even more. Together, users can improve the simulation models and share their ideas with the community without specific hardware requirements. The downside of this approach is the direct reliability on the accuracy of the model. Because the model has to be implemented from scratch, there is a lot of space for errors. The simulator that this paper is working with is OMNeT++ [8], but there are other similar projects and there are also simulation models which are self-contained and do not require a complex simulator environment.

One of the most popular well-known simulators is Cisco Packet Tracer [9], but its implementation of BGP is hugely limited as it completely lacks the iBGP part of BGP.

This paper talks about a model of BGP specifically tailored to be run in OMNeT++ simulator with the INET framework [10]. INET contains a set of implemented network protocols (e.g., Ethernet, IP, TCP, UDP, etc.) that can be used by other protocol models.

2. History of the BGP Model in OMNeT++

BGP4 protocol model has been included in INET for a long time and was originally developed by OMNeT++ developers [11]. At this time, BGP had many shortcomings: BGP `UPDATE` messages did not contain all necessary routes, the states of the finite-state machine were inconsistent, some message types were unsupported, and the model had trouble recovering during changes to the topology. All of these shortcomings are described in the master's thesis by Adrián Novák [12]. He started his work in 2018. At the same time, contributor Mani Amoozadeh worked on his improvements for the BGP protocol model that were merged into INET later on in 2018 [13]. However, the changes made by Adrián Novák were not merged, as this implementation requires another rewrite in order to be compatible with the new INET and also synchronized with the changes and features that were made by Mani Amoozadeh and OMNeT++ developers.

During my rewrite of the model, a lot of issues of Novák's approach became apparent. Support for IPv6, which was one of the most prominent features that Novák added, was rewritten from the ground up with the mentality to avoid address-family specific methods as much as possible. Other improvements are mostly regarding the router's main operation loop, so that routes are processed in bulk, node operability that allows nodes to handle common operations during the

simulation, TCP improvements, so the connection can be reliably established, and Cisco-like configuration, which is more realistic and versatile and easily derived from configuration of real devices.

The ultimate goal is for this greatly improved model to be accepted and merged into INET itself. Because this new improved version would replace the old versions, there are a lot of requirements set by the INET maintainers. From their point of view, this new version should be a clear improvement that introduces exactly zero new problems, so that all users experimenting with BGP would recognize this version, and no forking or branching happens.

3. Basics of BGP

Border Gateway Protocol is used as a common language between Autonomous Systems. With the usage of BGP, these ASes exchange network prefixes. The information about these prefixes is then used by routers to route traffic to the desired destination. To be able to fully appreciate BGP, one has to understand the basic theory of AS.

3.1 Autonomous Systems

The Internet is divided into smaller and more easily manageable networks called Autonomous Systems. These smaller parts are often operated by different administrative entities and use a custom set of rules and policies. Routing inside an AS is usually managed by IGP (e.g., OSPF, RIP, EIGRP, IS-IS) or static routing, while connectivity between ASes is typically achieved with BGP. Every public AS is identified by a unique public AS number. One public AS can contain multiple private ASes. These are only used for internal subdivisions and are not advertised via BGP. Public AS numbers are assigned by the respective Regional Internet Registries, e.g., RIPE NCC for Europe. Every AS can be connected to other ASes in different ways:

- **Single-Homed AS** (a.k.a. Stub AS) - connects to a single external AS;
- **Multihomed Nontransit AS** - connects to multiple other ASes, does not allow transit traffic;
- **Multihomed Transit AS** - connects to multiple other ASes, allows transit traffic.

In 2006, it was estimated that around 20-30% of all ASes are transit [14]. The total number of ASes worldwide is currently sitting at 109 996 [15]. Tools such as DB-IP [1] can be used to access publicly available pieces of information about ASes, their registered organizations, and IP prefixes.

[1] https://db-ip.com/

3.2 eBGP vs. iBGP peering

BGP recognizes two different types of peering (i.e., prefix exchanging): External BGP (eBGP) and Internal BGP (iBGP). External BGP peering is established with peers that belong to a different AS. The only necessity is TCP connectivity and pre-shared IP addresses between the organizations, as BGP does not use dynamic neighbor discovery. In contrast, Internal BGP is established between peers within the same AS. iBGP requires a full-mesh peering in regard to other routers in the same AS.

Each BGP router, also known as BGP speaker, advertises only the best route for each destination to configured peers. BGP can learn about prefixes in two ways:

- insertion into BGP locally via a static insertion or via redistribution by other routing protocols;
- receipt of Update Message advertising prefix as reachable by a BGP peer.

Routes received via BGP are generally advertised to other BGP peers — unless they are received from iBGP peers, in which case they are never advertised to other iBGP peers. Routes are withdrawn from the BGP if they become unresolvable or if an Update message is received, which requests withdrawal of routes.

3.3 Route representation

Each route embodies a reachable destination network prefix and is accompanied by a set of attributes. All these routes are stored in a data structure called BGP Table with the best route for each destination being highlighted or inserted into BGP Routing Table. These routes are the candidates for installation into the router's routing table, but can be overruled by route with a lower administrative distance. BGP does not use a single number to represent the quality of a route and instead uses a set of attributes to assess the desirability of a route. Each AS individually sets the weight of each attribute and so the path selection is influenced by the policy of the given AS. Mandatory attributes for each route are the following:

- `ORIGIN` - source of the information;
- `AS_PATH` - vector of AS number the route passed through;
- `NEXT_HOP` - IP address that should be used as a next-hop for the destination.

3.4 BGP Messages

BGP uses just four types of messages. All are sent reliably through TCP on port 179. It uses a peer-to-peer model with both BGP speakers being equal.

Table 1. Summary of basic features implemented in the INET Version and Novák's Version

Features	INET Version	Novák's version
Cisco-like configuration	✗	✓
IPv4	✓	✓
IPv6	✗	✓
Independent on OSPF	✗	✓
Initial Prefix Exchange	✓	✓
UPDATE message with multiple NLRIs	✗	✗
UPDATE message with Withdrawn routes	✗	✓
NOTIFICATION Message	✗	✗
BGP Table	✗	✗
Start/Stop/Crash Handlers	✗	✗
TCP Closed Handler	✗	✓
Interface State Change Handler	✗	✗
Routing Table Change Handler	✗	✗
Local Pref Attribute	✗	✗
MED Attribute	✗	✗

- `OPEN` - First message after TCP connection is established. Announces version of BGP, capability extensions, and AS number which determines eBGP or iBGP.
- `KEEPALIVE` - BGP's ping-pong mechanism.
- `UPDATE` - Advertises available prefixes with their attributes or withdraws prefixes as they become unavailable.
- `NOTIFICATION` - Carries information about detected errors. BGP peering is closed right after this message is sent.

4. State of the Implementation

As the first step, both available versions of the BGP model (current INET version and Novák's version) were analyzed and their capabilities tested. Both version are directly compared in Table 1.

4.1 INET version

The INET version was found to contain a few different bugs and dirty workarounds and a lot of missing features, among which are the following:

1. dependency on the OSPF module;
2. decentralized and unique declarative style of configuration;
3. support for IPv4 only;
4. missing withdraw route functionality;
5. limited interactivity with the ScenarioManager;
6. missing `NOTIFICATION` message type.

With this model, the simulation is very limited. Firstly, one has to learn the specific way of configuration, and even then, only the initial route exchange between BGP speakers can be observed. No simulation scenarios can be tested on this BGP implementation as it lacks the ability to withdraw routes. Some path selection steps are omitted, popular attributes like `MED` are not implemented. Furthermore, because no such structure like BGP Table is present, switching to a backup route when the primary route fails is not possible. The model also lacks the ability to recover after a link is reconnected, or the ability to be shut down or started up during the simulation. Because there is a bug with the TCP module which prevents a simultaneous TCP connect operation by two simulation entities, the simulation introduces increasing delay for each new connection.

4.2 Novák's version

A lot of downsides of the INET version were also observed by Ing. Adrián Novák. In his version, he created a new configuration scheme that closely follows Cisco's configuration, added withdrawn route functionality, and introduced support for IPv6. While a lot of the aforementioned problems were resolved, the resulting code was not merged into INET codebase. The readability and structure of the code suffered quite a bit and many of the problems with the INET version remained. IPv6 support was added by excessive and unnecessary copying of IPv4 methods without leveraging any of the powers of object-oriented programming. Advertisements of prefixes of an address-family that is different than the address-family of the architecture are also not supported. What was essentially created was a very hard-to-maintain code with still only very limited interactivity with the `ScenarioManager`.

5. OMNeT++ Implementation

The initial goal was to push the Novák version into a usable state and then create a pull request. But as

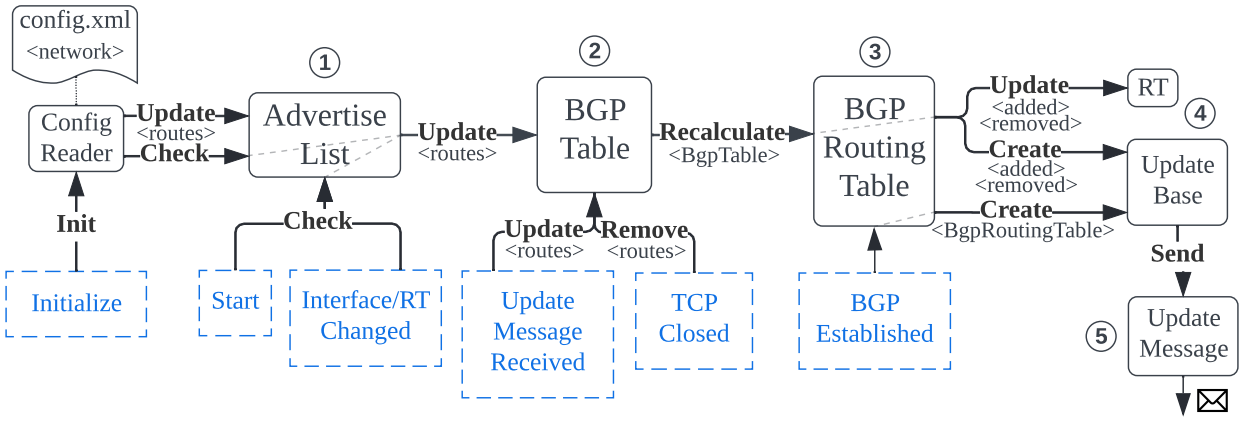

Figure 2. Simplified diagram of the new operation loop of the BGP simulation model. The diagram shows how the main route data structures interact with each other and the different input events (highlighted in blue) are processed.

more and more problems with this version became apparent, most of the model had to be rewritten (around 60%) and in the end, the Novák's version was used only as a reference with the INET version being used as a new base. A non-exhaustive list of changes includes configuration changes, OSPF dependency removal, introduction of IPv6 with the usage of generics as much as possible, the addition of BGP Table, rewrite of node's main operation loop, and addition of multiple mechanisms for handling changes of interfaces or TCP connections. Highlights of the changes are in the following subsections.

5.1 Configuration

The new configuration file structure is loosely based on the Novák's version [12] which itself is based on MP-BGP configuration on Cisco devices. It allows for normal IPv4 configuration, IPv6 configuration, their combination and even advertisement of IPv4 prefixes over IPv6 infrastructure or advertisement of IPv6 prefixes over IPv4 intrastructure, which was not possible in either of the existing versions. Furthermore, the new configuration allows for interface IP address assignments and static routes insertions. The previously mentioned TCP bug was fixed following a discussion with INET's developers and as such, all code related to the calculation of TCP connect delays could be cleaned up.

5.2 Support for Multi-Address-Family

Because the old model was not exactly designed and coded with IPv6 in mind, numerous changes were required. While code from Novák's BGP repository is no doubt functional, it contains a lot of duplicated code, which, inadvertently and unnecessarily, bloats the codebase. The goal was to have as little of address-family specific methods as possible. First, the IPv4 specific class `BgpRoutingTableEntry`, which represents a single BGP route, was changed to utilize the generic `L3Address` class instead. This enabled its usage even for IPv6 routes. Helper methods, which work with `BgpRoutingTableEntry` were rewritten to work with the abstracted address. Additionally, separate internal structures to store IPv6 routes were created, but because of the abstaction, no IPv6 specific methods were needed. `OPEN` message was extended by address-family capability field and `UPDATE` message was implemented for IPv6 routes. During the implementation, a bug in INET Neighbor Discovery Protocol (NDP) model was discovered and reported to INET maintainers [16].

5.3 Rewriting the Operation Loop

There were several missing key features of the BGP protocol that were addressed with the redesign of the main operation loop. The major issues included:

- Received `OPEN` messages were not validated.
- BGP Table was missing. Routes that were not installed in the BGP Routing Table were discarded, and so they could not be used as a fallback route if the necessity arised.
- Event handlers for various RT/interface signals or TCP events were missing, greatly limiting the interactivity of the model while ignoring the events of TCP connection.
- Routes were handled individually instead of in bulk, resulting in unnecessary additional processing and generation of redundant `UPDATE` messages.

The redesigned operation loop that addresses these shortcomings is shown in Figure 2. What was essentially created is a chain of consecutive subroutines (numbered 1 through 5 on the diagram) that takes an

incoming event together with its associated routes, recalculates the contents of these internal structures accordingly, updates the appropriate RT, and notifies relevant peers about the changes. The context of the event that triggered the computation is passed through the chain as well, allowing BGP to branch in different stages as needed. These stages are as follows:

① Validation of the Advertise List First, `advertiseList` structure is populated with prefixes entered by the `network` command once during node initialization. When the configuration file parsing process is complete, every prefix in the `advertiseList` structure is processed into BGP route entry and marked as valid or invalid, depending on whether the prefix can be exactly matched in the RT (*check* operation). A vector of these entries is passed to update `BGPTable`. *check* operation is executed either after the configuration file has been parsed, during the start of the node, or if a signal notification regarding Interfaces or RT is received.

② Update of the BGP table This subroutine takes a vector of routes as input. Each invalid entry is removed from `BgpTable` and each valid entry is verified to be present and added if necessary. The routes received in the `UPDATE` messages are processed in the same way; routes received in `UPDATE` message are processed into BGP route entries, while the withdrawn routes are also marked as invalid. If an active TCP socket closes, all routes received from that particular peer are invalidated. With this updated `BgpTable`, `BgpRoutingTable` can be recalculated.

③ Calculation of the BGP Routing Table This procedure selects only the best route to each destination. It leaves `BgpRoutingTable` in sync with `BgpTable` and also returns two vectors: one vector that contains routes removed from the BGP Routing Table during the computation, and one vector that contains routes newly added to the BGP Routing Table. Both vectors are used to update the local RT and are passed to the `updateSendProcess` method, which creates a base for the update messages.

④ Creation of the Update Base This procedure has two possible inputs: either it takes the two aforementioned vectors from the previous stage or it takes the entire current content of the BGP Routing Table. Its main purpose is to calculate attributes for each route and group them accordingly so that as few `UPDATE` messages are sent as possible.

⑤ Sending of the Update Message This procedure creates and sends the appropriate AF-specific `UPDATE` message with the content of the structure created in the previous stage.

This redesign required a significant rewrite of the `BgpRouter` class. To allow the model to receive events during the simulation, the inheritance hierarchy of several classes had to be changed and additional handler methods had to be implemented. After all of these changes, this new version of the model has all features listed in Table 1.

6. Testing

Since the majority of the model had been rewritten, thorough testing was required. Test suites of the previous model versions included a few different topologies, but since the models' support for `ScenarioManager` was fairly limited, usually, only the initial exchange could be observed and analyzed. With the model being extended to fully support TCP state changes, interface state changes, routing table changes caused by other protocols, and node manipulation by `ScenarioManager`, a completely new and much-extended set of testing scenarios was devised. In order to provide the tests with additional value, they were taken in an educational direction.

In a video conference based discussion, a serious interest in routing tutorials was expressed by INET developers and OMNeT++ community. It was agreed to create a set of BGP tutorials that could be published on INET website. A similar unfinished set for OSPF and RIP was already created as an inspiration [17]. In the BGP set of tutorials, each includes an animation of the simulation, accompanied simulation files, and a description.

6.1 Methodology

Each tutorial is focused on exactly one feature or aspect of BGP implemented in the simulation model. Its behavior and purpose are explained in detail. Testing topologies are kept as simple as possible. Every topology is compared to a topology built with Cisco BGP images inside the EVE-ng emulator. Reproduction package that allows the replay of resulting behavior is also included. Each tutorial is accompanied by the following:

1. animated imagery showcasing the topology and the exchange of the messages;
2. detailed explanation of the showcased feature and observed behavior;
3. captured traffic (`.pcap`) and RT changes in the simulation;
4. captured traffic (`.pcap`) and RT changes in the emulator with Cisco images;

5. reproduction package (simulation files and Cisco configuration files)

A detailed explanation of the testing process can be found in paper [18] which deals with the standardized methodology of simulation models.

6.2 Example

The creation of these tutorials is an ongoing process. A preview of the topics can be found on the INET website [17] and some of them are already available on ANSA wiki page [19]. However, this is only temporary version as the goal is to get the tutorials published on INET website itself. Detailed example of the methodology (executed on EIGRP) can be also found on ANSA wiki page [20].

7. Conclusion

In this paper, I have described the state of the BGP simulation models for OMNeT++ and I have highlighted their issues. Furthermore, I have solved the issues by re-writing a considerable part of the model. Nodes' main operation loop was completely redesigned and the interactivity of the model with the simulation was greatly improved. The resulting implementation combines features of both currently available models. Requirements of the INET maintainers were also taken into account during the implementation. I have described a testing methodology that is used to verify the model's functionality which also provides an educational value to the reader. The first section of the paper provides the reader with the necessary background about the basic BGP concepts and the theory behind Autonomous Systems.

To ensure the best chances for the resulting model to be merged into the INET code base, multiple code consultations occurred, which further narrowed the requirements for the product. A special 'Hackathon' session regarding the BGP model was held during the international OMNeT++ Summit in September 2021[2]. Bugs in TCP [21] and NDP [16] simulation models were also reported, with TCP being fixed quickly after the fact.

Further improvements to the simulation model could deal with a unified way of handling redistribution. This feature would be, in my opinion, another big step-up in the quality of the simulation.

8. Acknowledgments

I would like to thank my supervisor Ing. Vladimír Veselý, Ph.D. for his consistent stream of motivation.

[2] https://summit.omnetpp.org/2021/

References

[1] Santosh Janardhan. Meta: More details about the october 4 outage. https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/, October 2021. [Online].

[2] Ratul Mahajan, David Wetherall, and Tom Anderson. Understanding bgp misconfiguration. *ACM SIGCOMM Computer Communication Review*, 32, 09 2003.

[3] Lily Hay Newman. The infrastructure mess causing countless internet outages. https://www.wired.com/story/bgp-route-leak-internet-outage/, June 2019. [Online].

[4] Geoff Huston. AS65000 BGP Routing Table Analysis Report. https://bgp.potaroo.net/as2.0/bgp-active.html. [Online], Accessed: 2022-04-23.

[5] Geoff Huston. AS131072 IPv6 BGP Table Data. https://bgp.potaroo.net/v6/as2.0/index.html. [Online], Accessed: 2022-04-23.

[6] Jeremy Grossmann et al. GNS3. https://www.gns3.com/. [Online], Accessed: 2022-04-23.

[7] EVE-NG Ltd. EVE-NG. https://www.eve-ng.net/. [Online], Accessed: 2022-04-23.

[8] OpenSim Ltd. Simulator OMNeT++. https://omnetpp.org, 2022. [Online], Accessed: 2022-04-23.

[9] Cisco Systems. Packet tracer. https://www.netacad.com/courses/packet-tracer. [Online], Accessed: 2021-07-23.

[10] INET. INET Framework. https://inet.omnetpp.org, 2022. [Online], Accessed: 2022-04-23.

[11] INET. Initial BGP Implementation. INET commit 9763e6c, 2018. [Online].

[12] Adrian Novak. Modeling and simulation of bgp. Master's thesis, Brno University of Technology, Faculty of Information Technology, 2019.

[13] Mani Amoozadeh. Pull request #381 - Bgp improvements. https://github.com/inet-framework/inet/pull/381, 2018. [Online].

[14] Beichuan Zhang, Vamsi Kambhampati, Daniel Massey, Ricardo Oliveira, Dan Pei, Lan Wang, and Lixia Zhang. A secure and scalable internet routing architecture (sira). 2006.

[15] Patrick Maigron. World - Autonomous System Number statistics. https://www-public.imtbs-tsp.eu/~maigron/RIR_Stats/RIR_Delegations/World/ASN-ByNb.html. [Online], Accessed: 2022-04-23.

[16] ANSA. INET NDP Issue. https://github.com/inet-framework/inet/issues/746, 2022. [Online].

[17] INET. Tutorials. https://inet.omnetpp.org/docs/tutorials/, 2022. [Online].

[18] Vladimír Veselý and Jan Zavřel. Quality control methodology for simulation models of computer network protocols. *CoRR*, abs/2109.12854, 2021.

[19] ANSA. EIGRP V&V Result reproduction package. https://github.com/ANSA/results-reproduction/wiki/INET-BGP-Routing-Tutorials, 2022. [Online].

[20] ANSA. EIGRP V&V Result reproduction package. https://github.com/ANSA/results-reproduction/wiki/OMNeT-Community-Summit-2021, 2021. [Online].

[21] Rudolf Hornig. INET TCP Issue. https://github.com/inet-framework/inet/issues/92, 2022. [Online].