BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2021
http://excel.fit.vutbr.cz

Modeling of Enhanced Interior Gateway Routing Protocol

Jan Zavřel*

Abstract
This paper deals with the implementation and the evaluation of a simulation model of a modern dynamic routing protocol designed by Cisco Systems, Inc. called Enhanced Interior Gateway Routing Protocol (EIGRP) in a discrete simulator OMNeT++ implemented in C++. The resulting simulation model can be used to conduct various experiments which allow network designers and alike to explore the protocol's behavior in different situations inside a safe discrete environment. In order to produce a trustworthy simulation course, the protocol model must be as accurate as possible to the real implementation and thoroughly tested. This paper provides a basic overview of both protocol EIGRP and simulator OMNeT++. It also discusses the state of the model before and after the integration into a newer version of the INET framework, showcases improvements, and outlines the testing methodology.

Keywords: EIGRP — Simulation — Routing — OMNeT++

Supplementary Material: INET's Github repository

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

1. Introduction

With the number of people using the Internet quadrupling since 2005 [1], it is now more important than ever to push for high availability and stability. These properties can be partially achieved with optimal route selection by network devices on the path to the desired destination. A combination of static routes and dynamic routing protocols is usually used. This approach enables the router to select optimal paths to various destinations even after the topology layout changes while having control over specific cases with static routes. Usage of these protocols allows routers to dynamically exchange routing information and update their routing tables accordingly when and if necessary. In order to verify the protocol behavior on larger topologies in critical scenarios, it is essentially a good practice to conduct various experiments in a network simulator. All observed effects of these events on the topology can be thoroughly investigated and analyzed. It is preferred to simulate such events inside safe environments rather than testing on active devices as it is cheaper, faster, safer, and more convenient overall. This however requires simulation models to be as accurate as possible.

OMNeT++ is a discrete, modular C++ simulation library and framework [2]. The functionality of this simulator can be extended with first or third party frameworks. The most popular OMNeT++ framework is INET [3]. It provides a set of implemented protocol simulation models for the link layer (`Ethernet`, `802.11`), the network layer (`IPv4`, `IPv6`), the transport layer (`TCP`, `UDP`) and the application layer (`RIP`,

`OSPF`, `HTTP`). These simulation models can be used to create and visualize network simulations and even implement custom simulation models of network protocols. The most current release version at the moment is `4.3.0`. INET is an open source project that is being developed by OMNeT++ developers and its community. An extension for INET called Automated Network Simulation and Analysis (ANSA) is being maintained at Brno University of Technology and it implements a number of network protocols such as `HSRP`, `IS-IS`, `BABEL`, `OSPFv3` and many others. It utilizes INET protocol models and greatly expands the number of available protocols to simulate. Some of these popular models are sometimes later integrated into INET itself (such as `OSPFv3`). ANSAINET models often use outdated INET modules to some degree or they rely heavily on ANSAINET's own modules which prevents a simple integration into the current INET.

Currently, the simulation model of EIGRP within OMNeT++ requires the user to use ANSAINET and in extension to also use much older versions of INET and OMNeT++. This solution is firstly inconvenient. Downloading outdated and completely separate software for a specific simulation model can completely discourage some users from experimenting. Moreover, any model in ANSAINET can not be simulated in conjunction with other models from the current INET. Additionally, simulations can also yield inaccurate results as the EIGRP simulation model has some now known bugs which are outlined in section 4.1. My work aims to make the EIGRP model better and easily accessible to the OMNeT++ userbase.

Another widely available and used EIGRP simulation model is included in the Riverbed network simulator. Riverbed is only available in a commercial version since Riverbed's Modeler Academic Edition has been discontinued as of September 01, 2020. Another well-known simulator is NS-2/3 which lacks EIGRP. Network emulators can also be used to conduct experiments. An example of such software is GNS3. Running an IOS image on an emulated hardware can be sometimes very unstable and it lacks the opportunity to customize protocols' behavior as well as being very non-transparent.

The goal of my work is to take the EIGRP simulation model from ANSAINET 3.4.0 as created by Ing. Jan Bloudíček, Ing. Vít Rek and Ing. Vladimír Veselý, Ph.D. [4], remove all dependencies on ANSAINET and fully integrate it with INET 4.2, so it can be publicly available to all INET users. Additional improvements and continuous support are expected.

2. Theory of Routing Protocols

The Internet can be viewed as a collection of disjoint sets of routers and there are two main types of dynamic routing protocols. One type provides routing within the set itself (IGP) and the other provides routing between different sets (EGP). Each set is called an Autonomous System (AS). These ASes are relatively small components of the Internet and each is under the control of a single administrative entity, usually a large organization such as Internet Service Provider. Protocols such as EIGRP, OSPF, IS-IS, or RIP are examples of the IGP type and BGP is the currently used EGP.

Interior Gateway Protocols (IGP) is a class of dynamic routing protocols and it is used to automatically manage routing table entries between routers in the same AS. Usage of these protocols however does not make static routes obsolete as specific cases can only be handled with static routes. One AS can also utilize multiple IGP protocols if such usage is favorable. These routing protocols have four main functions:

- discovery of remote networks;
- advertisement of known networks;
- calculation of best paths;
- recalculation of best paths when a change in topology occurs.

Other favorable characteristics are low resource utilization, fast convergence, good scalability, support for authentication, and easy extensibility. There are two main types of IGP protocols: *Distance Vector* and *Link-State*.

Distance Vector is a type of IGP that advertises routes as a vector of distance and direction. The distance can vary from a simple hop-count up to a composite metric with multiple different factors. Direction is next-hop or exit interface. IGRP, EIGRP, and all versions of RIP belong to this type.

Link-State is a type of IGP which, in contrast with Distance Vector, advertises link-states. Each router keeps track of its neighbors and generates messages with all necessary information about its directly connected links. This information is distributed throughout the desired area. Each router is able to build the exact same graph of the topology and then independently calculate the best paths to all destinations. Only incremental updates are sent if a topology change occurs. IS-IS and OSPF both belong to this type. Both of these protocols use Dijkstra's algorithm for finding the shortest path to the destination.

More about this topic can be learned in Cisco's materials [5].

3. Enhanced Interior Gateway Routing Protocol

EIGRP is an IGP dynamic routing protocol. It is officially classified as a distance-vector type but it is also referred to as being a hybrid between link-state (OSPF, IS-IS) and the distance vector (IGRP, RIP) by the community. This is because it uses many features found in link-state protocols such as non-periodical partial routing updates and the establishment of neighborships. Even though EIGRP has these beneficial traits, it remains a distance vector and it should be viewed as a modern take on a distance vector routing protocol. EIGRP was designed and developed by Cisco Systems, Inc. as an improved version of their previous proprietary routing protocol Interior Gateway Routing Protocol (IGRP), thus the 'Enhanced' in its name. EIGRP provides classless addressing, support for Variable Length Network Mask, and Classless Interdomain Routing. EIGRP uses a composite metric calculation which consists of up to 6 different components. It operates directly on top of the network layer in ISO/OSI model and supports IPv4, IPv6, AppleTalk, and IPX with the usage of Protocol Dependent Modules (PDMs). It also employs a unique transport protocol called Reliable Transport Protocol (RTP) to provide lossless message exchange. Unlike other distance vector protocols, it uses dynamic neighbor discovery and partial routing update messages which reduces convergence time and saves resources. It uses the Diffusing Update Algorithm (DUAL) to calculate the shortest loopless route to the destination. EIGRP supports authentication, equal and unequal load balancing, stub routing to limit the overhead traffic, and division of topology into multiple EIGRP domains. EIGRP was submitted as an open standard in 2013 and it is most currently described in the informational RFC 7868 [6]. The first section of this RFC contains some of the terminologies that are used in the following three sections.

3.1 Packet types

- **Hello Packet** - used to discover neighbors on EIGRP enabled interfaces. The same format is also used for Acknowledgement messages.
- **Query Packet** - used by the DUAL algorithm for diffusing computation of the best path to the destination. Routers use this message to advertise a transition of a route to the Active state as it is missing a feasible successor on the path to the destination.
- **Reply Packet** - used as response to the Query. It advertises the availability of a destination with a metric.
- **Update Packet** - used to advertise available destinations.

3.2 DUAL

DUAL is a mechanism for handling topology changes and it manages each route individually. If a DUAL receives a *local* event for a route in a passive state, it tries to perform a local computation. In that case, the route remains in the passive state, while its metric and/or successor changes. If the feasible distance has changed, an Update message for all neighbors is generated. If there is no feasible successor available, a local computation is not possible and the route transfers to the Active state. When a route transfers to the Active state, diffusing computation begins. It starts with sending Query messages to EIGRP neighbors and ends with the reception of the Reply message from each neighbor. DUAL uses a Finite State Machine (FSM) to keep track of the route diffusing computation status. If a given route in the Passive state receives a Query message and a feasible successor for the given route is available, the route remains in the Passive state while a Reply is generated. If there is no feasible successor, the router begins its own diffusing computation. Diffusing computation is finished when all sent Queries are answered with a Reply. If a route computation was triggered with a Query message, a Reply message is generated. If the feasible distance for the route changed, an Update message for all neighbors is also generated.

3.3 Metric Calculation

EIGRP uses a composite metric to evaluate known routes. It employs up to six components for its metric calculation. These components are called K-values or coefficients. Each coefficient has a value in the range of 0 to 254. The metric calculation can be customized by manipulation of these values. In order to ensure loopless routes, all EIGRP active devices in the same EIGRP domain have to share the same K-values. This is ensured by advertising them in Hello Packets. The final metric calculation has the following format and may include factors like throughput, latency, load, reliability, and extended attributes.

$$metric = (K_1 * Thr + \frac{(K_2 * Thr)}{256 - Load} + K_3 * Lat + K_6 * Ext) * \frac{K_5}{(K_4 + Rel)} \quad (1)$$

4. OMNeT++ Implementation

OMNeT++ is a discrete, modular C++ simulation library and framework mostly used for network simulations. It has a generic modular architecture allowing for a wide spectrum of simulation model types. Each simulation model is composed of modules. Modules communicate by messages and multiple modules can be combined into a single component module. Module nesting is not limited and each module is technically reusable. The behavior of each module can be easily customized as they are implemented in C++. Module hierarchy is defined in OMNeT++'s language called *NED* [2].

4.1 State of EIGRP in ANSAINET

Current EIGRP implementation is divided into several modules. The single-responsibility principle (SRP) is met as each module has narrowly defined responsibilities which are also summarized by the module's name. Source files also use the correct folder structure scheme. The core of the implementation are PDMs implemented in `EigrpIpv4Pdm` and `EigrpIpv6Pdm`. These modules handle all the input events and utilize other modules, especially data structure modules. Because IPv4 and IPv6 addresses are very similar, modules usually can accommodate both with the usage of `C++ templates`. Module versions for IPv4 and IPv6 are split only if bigger code divergences are needed. The configuration of the EIGRP process and IPv6 interfaces are specified in the `.xml` configuration file. However, the current implementation has several issues (i):

- **i1** - Flawed metric calculation. This issue occurs when a route combines interfaces of different speeds.
- **i2** - `NetworkTable`'s reference counter does not subtract the number of removed routes and thus proper entries do not get deleted in some cases.
- **i3** - Receiving a Query for a route that is being deleted crashes the simulation.
- **i4** - Input signals require a specific order otherwise protocol model could loop indefinitely.
- **i5** - `ANSAINET` dependencies - especially dependencies on `ANSA_InterfaceEntry`. This requires changes throughout all the modules.
- **i6** - `INET 3.4.0` dependencies - Old way of creating, sending and handling messages.
- **i7** - Interface IP configuration is loaded via ANSA specific class. IP configuration should be consistent with other simulation models which are implemented in INET.

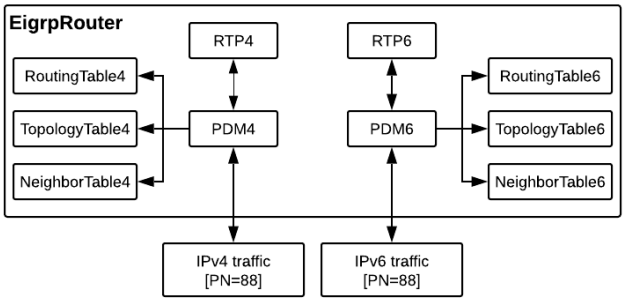

Figure 1. Structure of EIGRP Router in ANSAINET 3.4.0. `PN` stands for Protocol Number, `88` in case of EIGRP.

- **i8** - Classification of an address-family is fully handled by ANSA. The module should have direct access to the network layer.

The structure of EigrpRouter is shown in Figure 1.

4.2 Addressing Issues

During the integration of the EIGRP protocol model, the overall structure of the model was left almost untouched. Most focus was given to highlighted issues recognized in section 4.1.

In order to address issue **i8**, a traffic splitter was added in front of both PDMs. This splitter classifies the address family of each packet and forwards it to the correct PDM. A new structure is shown in Figure 2.

Issue **i7** was addressed with `IPv4NetworkConfigurator`. This is a somewhat consistent solution with the OSPFv2 model for example. IPv6 addresses must be specified in the configuration `.xml` file as there is no consistent way of the IPv6 configuration. This IPv4 solution is up for debate because the new way is more consistent with other models while the old way of configuration (parsing both IPv4 and IPv6 from the `.xml` configuration file) is more consistent with real devices.

Issue **i6** was addressed by updating method calls to the newer INET's API. Most prominent changes were made in the network layer method calls for operations with IP addresses and with message creation and dispatch procedures.

Issue **i5** required changing the usage of the `ANSA_InterfaceEntry` class to `NetworkInterface` class and moving all the necessary attributes of an EIGRP interface (delay, reliability, load, bandwidth) into `EigrpInterfaceEntry`.

Issue **i4** was addressed by not allowing the same signal to be processed multiple times. This area could use some improvements by subscribing to completely different and newer types of signals. This is also mentioned in the conclusion.

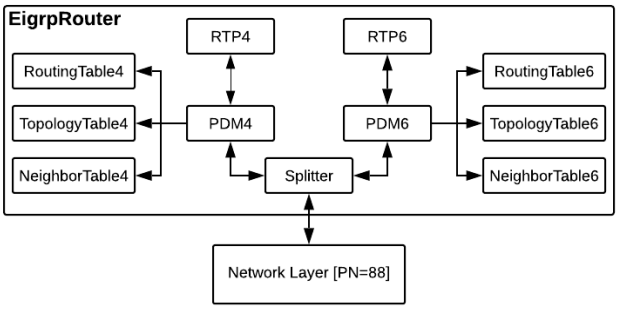

Figure 2. Structure of EIGRP Router in INET 4.3. `PN` stands for Protocol Number, `88` in case of EIGRP.

Issues **i3** and **i2** are actually related. Because `NetworkTable` entries are not managed properly, a received Query message can have its needed network information deleted before a proper Reply message is created, which results in the simulation crashing. This is addressed by properly counting the number of references for a given network.

Issue **i1** is caused by a simple bug in the metric calculation which was hard to find and easy to fix.

With the integration process being finished, the resulting model was tested on multiple topologies.

5. Testing

Tests of this EIGRP model were done against the implementation on real Cisco IOS images. All Cisco routers were running on version `15.4` with *Advance Enterprise Services* feature set. The model was tested on topologies of various sizes and each topology was also tested in multiple scenarios. These typically include topology-changing events. The EIGRP traffic that is being generated in the simulator during these scenarios is compared to the traffic captured on Cisco's referential implementation. The contents of Neighbor tables, Topology tables, and Routing tables are also compared. The following section contains one of the tested topologies.

5.1 Testing Showcase

This section showcases some of the methodologies used to verify the validity of this model. The topology is shown in the Figure 3. All routers have configured EIGRP and all interfaces are included in the EIGRP process. During the start-up of the topology, there is a huge amount of Hello messages as all routers are trying to discover EIGRP neighbors. When two routers become neighbors, they exchange the initial set of UPDATE packets in order to inform each other about available routes. The comparison between OMNeT++ traffic and real captured traffic can be sometimes problematic. Because of the discrete nature of the simulator, actions for a given event are processed immediately.

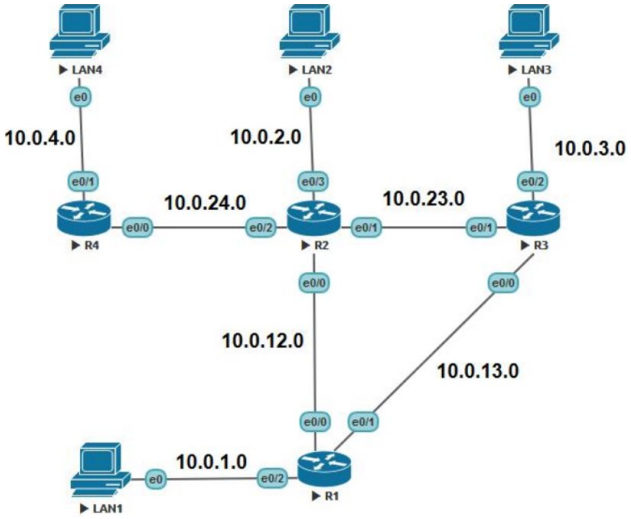

Figure 3. Simple IPv4 topology containing 4 LANs and 4 EIGRP Routers

There is also a certain race condition on real devices which causes each run of the topology to be somewhat unique regarding the exact timings and the order of the messages so the topology has to be tested multiple times and the weight is given to the syntax and semantics of the messages rather than their order.

Arguably more important are then the actual updates of the Routing tables. EIGRP creates an entry in the Topology table for every known route and only the best routes with the lowest metric are being propagated into the Routing table itself. Figure 4 shows a comparison between the topology table of `R1` on Cisco router and OMNeT++ simulation after the topology reached convergence. A closer inspection of these two tables together with an inspection of the traffic itself shows the validity of the model on this topology.

6. Conclusions

While EIGRP is not the most popular routing protocol it has some favorable properties. It has highly customizable metric calculation, niche features like unequal load balancing and address-family support while being reportedly easier to understand than OSPF. A recent article [7] found that EIGRP can have higher performance in convergence duration in some small-scale topologies than OSPF. Even though this study used the GNS3 emulator with some mentioned issues, EIGRP could potentially replace OSPF in some real-life scenarios with a performance benefit. A huge downside of EIGRP is its very scarce support by network vendors outside Cisco and some Cisco proprietary features.

The integration of the EIGRP simulation model into the current INET was finished in Autumn 2020. The results were presented by myself (and my supervisor) at the international OMNeT++ Community Sum-