BGP Simulator Documentation

Group 8
Niko Hellgren, 505174

Protocol Processing, Spring 2017

Contents

0.1 Introduction 1
0.2 Code overview 2
0.3 Trust implementation 2
0.4 General notes from the project 2

0.1 Introduction

The simulator software is implemented in Java programming language using just the base libraries. For visualization purposes, a graph plotting library, GraphStream [1], was utilized. A high-level overview of the program structure is presented in 0.2. The code is available in `https://github.com/nipehe/BGP-simulator` for the time being.

The simulation has the following basic technical features and limitations:

- The simulation is strongly threaded, with each router and client working as its own thread and `Timer` threads utilized in testing and `KEEPALIVE` messages.
- Layer 1 and Layer 2 functionalities are simulated by Java's `PipedInputStream` and `PipedOutputStream` that are made to transit `byte[]` arrays between threads. The output streams are `synchronized` to make sure the sent packets do not mix up.
- Layer 3 functionality is implemented according to the IPv4 protocol. Although the constructed packets contain and transmit all the fields defined in [2], fields *Type of Service*, *Identification*, *Flags*, *Fragment Offset*, and *Protocol* are filled with default values in all cases, and they are not used for anything. This is partly enabled by the simulation not supporting packet fragmentation.
- A globally available static class provides the simulated routers and clients with DNS-like functionality, since implementing an actual DNS functionality is not essential in this simulation.
- The simulated network uses BGP-4 messages to transmit information between routers.
- TCP functionality is not implemented, and BGP packets are transmitted as ordinary IP packet payloads. This removes the possibility to test or simulate security and stability issues caused by TCP, but simplifies the implementation due to the complicated functionality of TCP [3].

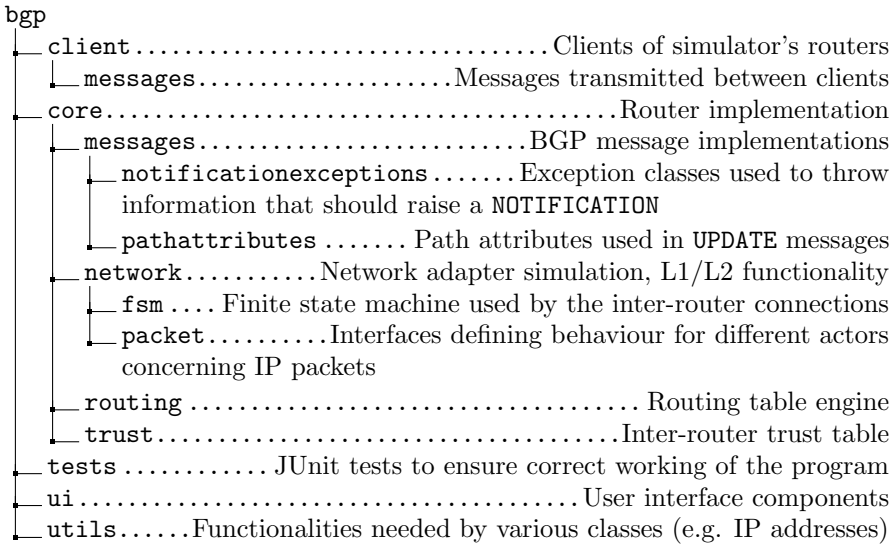

Figure 1: Package structure of the code

- Routing information is transmitted using `UPDATE` messages after connection initialization and all routing table changes.
- Additional BGP message type, `TRUST` has been added for trust voting between routers. To avoid Man-in-the-Middle attacks (usually the router being voted on is on the transmission path of the vote), the trust votes are encrypted using 1024-bit RSA and signed using RSA with SHA-1. To avoid making key transmission overly complicated, a PKI-like functionality was made available as a globally available class, providing routers with other routers public keys. The trust implementation is discussed in more detail in Section 0.3.
- Behaviour in error situations has been implemented comprehensively, and both L1/L2 breakage, missing `KEEPALIVE` messages, and erroneous BGP messages cause the routers to drop the link and inform their neighbourhood of this. Possible issues effect the trust rate of the misbehaving router.

0.2 Code overview

0.3 Trust implementation

0.4 General notes from the project

- When Java interprets bytes as integers, they are shifted to range -128..127. Doing bitwise shifts also easily converts the values to a larger data size (e.g. 32-bit integers) instead of dropping the overflowing bits. This caused multiple errors in initial implementations of the bit-level manipulations, and adding bitmasking (e.g. `<value>&0xFF` to force a value to 8 bits) was

necessary in most of the places to avoid issues caused by this. Lack of unsigned numbers also made value comparisons difficult at some points, since the interpretations of bytes easily flowed over to the negative numbers.

- Great built-in support for threads, timers and task executors in Java 8 made some otherwise difficult tasks (e.g. `KEEPALIVE` message sending and checking) really easy.
- Following object-oriented paradigm in development was both intuitive and helpful, due to the software being a simulator.

Bibliography

[1] GraphStream Team, “Graphstream - a dynamic graph library,” Mar. 2017.

[2] Information Sciences Institute, “Internet Protocol.” RFC 791, Sept. 1981.

[3] Information Sciences Institute, “Transmission Control Protocol.” RFC 793, Sept. 1981.