BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

Modelling of OSPFv3 Link-State Routing Protocol

Michal Ruprich*

Abstract
This paper deals with a simulation of routing protocols. Specifically with the OSPFv3 link-state routing protocol. OSPFv3 is a modern multi-address family protocol which means it supports both IPv4 and IPv6 routing. The resulting model may be used to demonstrate routing mechanisms in real networks. It is implemented in OMNeT++ Discrete Event Simulator and will become a part of INET framework. A contribution of this work is that a working model of OSPFv3 has not been yet implemented in any other simulators that are similar to OMNeT++.

Keywords: OMNeT++ — OSPFv3 — INET — Network Simulation

Supplementary Material: Github repository

*xrupri00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Computer simulation plays a very important part in the process of building any system. It allows us to analyze complex systems, to evaluate new ideas and concepts and to identify major problems in any design before spending huge amounts of money on implementation.

Every larger company uses a computer network to interconnect its employees, departments and even distant branches together to improve communication and cooperation between them. To design any such network and its routing schematics without creating a model first would be a very challenging task and in case of any misconduct, it could lead to unnecessary expenditures to fix the problem.

This work focuses on implementing a functioning model of the Open Shortest Path First (OSPF)[1] link-state routing protocol specifically its latest version the OSPFv3[2]. The aim is to create an independent module representing the OSPFv3 process which may be used as a part of various network devices. When used in a simulation representing a real network, the module will be capable of demonstrating how the routing mechanisms work.

The main advantage of this work is that it will provide a tool to simulate dual stack routing with both IPv4 and IPv6 on a single device. Since the IPv6 network protocol is being used more frequently, this is a huge benefit. The model may be used to simulate various real scenarios. For instance we might be able to simulate a transition from IPv4 to IPv6 while at the end of the transition both protocols are used in the network.

The model is being implemented in OMNeT++ [3] discrete event simulator and later it will become a part of INET framework. The INET framework [4] is a collection of packages that provide various models of internet stack protocols (TCP, UDP, IPv4, IPv6...) along with many other wired or wireless technologies. This work is part of the Automated Network Simulation and Analysis (ANSA [5]). ANSA is a project which aims at researching possibilities to extend the INET framework. It also focuses on creating a tool which could be used to automatically create realistic models of real networks.

There is an OSPFv2 model implemented in INET. This version only supports IPv4 network protocol routing and it only supports one OSPF process per router. It also lacks the support of the Not-So-Stubby Area (NSSA) option [6] present in OSPF. The OSPFv3 routing protocol simulation is not currently present in any other simulator similar to OMNeT++.

2. Theoretical Background

2.1 Dynamic Routing

Dynamic routing is a process that plays an essential role in every network. Its role is to exchange and distribute routing information in a network. In contrast to static routing, where the administrator needs to specify all routes in a network manually, dynamic routing offers greater network sustainability and scalability. The main tasks of every dynamic routing protocol are the following:

- discovering remote networks
- maintaining up-to-date routing information in the routing table
- choosing the best paths to destination networks
- reacting to any changes in the network

2.2 Link-State Protocols

Link-state protocols such as IS-IS and OSPF are more sophisticated and more difficult for an administrator to maintain than distance vector protocols such as RIP or EIGRP, but provide much better control over the routing process. Each router running a link-state protocol builds its topology of the whole network. The main aspect, which enables a router to "see" the structure of the network, is the fact that every router works with information not only from its neighbor but also from each and every router in the network. Such knowledge of the network allows every router to calculate best paths to distant networks rather, than using only information provided by its neighbors. The two best-known protocols - OSPF and IS-IS(Intermediate System to Intermediate System) - use the same algorithm for best path calculation, i.e. the Dijkstra algorithm also known as the shortest path algorithm. A pseudo code of Dijkstra algorithm is shown in 1.

2.3 OSPF

OSPF was designed to work with TCP/IP internet environment and is inherently classless. Routing packets can be secured with a variety of authentication methods, therefore only trusted routers can exchange routing information. OSPF uses only a small amount of network traffic and is designed to have very short

```
function DIJKSTRA(Graph, Source)
    for all vertex v in Graph do
        distance[v] ← infinity
        previous[v] ← undefined
    dist[Source] ← 0
    Q ← all nodes in Graph
    while Q ≠ ∅ do
        u ← node in Q with smallest distance[]
        remove u from Q
        for all neighbor v of u do
            alt ← distance[u] + dist_between(u, v)
            if alt < distance[v] then
                distance[v] := alt
                previous[v] := u
        return previous[]
```

Figure 1. Dijkstra algorithm pseudo code

convergence times when a change in the topology occurs. Routing information received from other routers is stored in a link-state database, which is later used by SPF to calculate the best paths to destination networks.

The Shortest Path First(SPF) calculation is the core of the routing protocol. It determines best possible routes to distant networks and populates the routing table. The SPF algorithm is used to find the shortest path in a graph by creating a shortest path tree. In this case the tree is a representation of the OSPF domain and the nodes are reachable networks with the router as the root.

The OSPFv3 Process running over IPv4 [7] and IPv6 [8] has the same core algorithms and structures. These are the SPF algorithm, neighbor data structure and state machine and interface data structure and state machine. There are some changes introduced in OSPFv3.

2.4 OSPFv2 and OSPFv3 differences

IPv6 allows us to configure more than one IP address on an interface. This is why we no longer use the term subnet to describe a network segment connecting two or more routers. The term introduced in IPv6 is link. OSPFv3 runs per-link instead of per-IP-subnet.

OSPFv3 doesn't use its own authentication anymore. The authentication fields were left out because OSPFv3 relies on IP Authentication Header and the IP Encapsulating Security Payload.

OSPF uses Link State Advertisements (LSAs) to distribute information about reachable networks. OSPFv3 modifies previous LSAs and introduces new ones. Every LSA now has a flooding scope which is used to determine how to redistribute each LSA.

OSPFv3 run over IPv6 but it is capable of routing IPv4 as well [9]. This feature is implemented in form

of instances. Each interface is associated with one process but each process may have multiple instances. Each instance is identified by instance ID (unique number which identifies this instance across all routers in the network) and address family (this may be either IPv6 or IPv4).

OSPFv3 introduces new options field in the OSPF packet header. These options bring some interesting routing capabilities. It allows us to handle demand circuits or set the router to inactive mode in which it still participates in routing but doesn't forward packets which are not on the link-local scope (this is very useful for multi-homed hosts).

3. Design

The model should reflect the functionality of OSPFv3 process on network devices from Cisco. It is designed to support multiple processes running on a single device as well as two address families per process. Each process is designed as a separate module. Inside the process there are two instances available, one instance per address family. Each router's interface which participates in the routing process is associated with exactly one process thus allowing it to run maximum of two instances, one for IPv4 and the other for IPv6.

4. Implementation

As mentioned above, the model is implemented in OMNeT++. OMNeT++ uses component based architecture. Each component is programmed in C++, larger and more complex components are put together using a high-level language NED [3].

The structure of the module is shown in figure 2. The Splitter module and the processes modules are simple components and their internal logic is implemented in C++. The routing module is a compound module defined in NED language and it puts all the modules together. For anyone who would like to use the OSPFv3 routing module, it would appear as a "black box" which is capable of simulating the routing processes.

The Splitter module is responsible for the initial creation of each process. During the simulation, it provides an interface between the network layer and the process modules inside. It delivers each message to the appropriate process according to the router's interface on which the message is received.

Each process works with its own list of areas as shown in figure 3. Each area may be spread across multiple interfaces and each interface may be associated with multiple neighbors (this is typical for networks

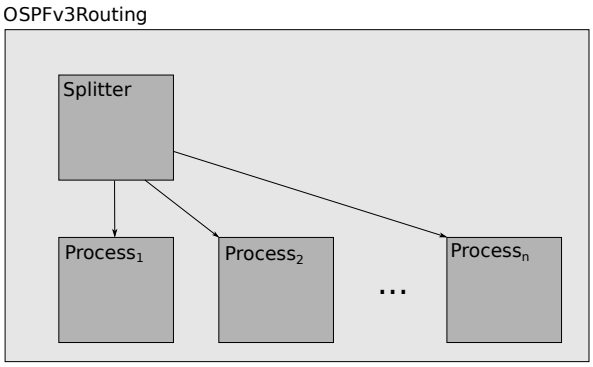

Figure 2. The OSPFv3 Routing module structure

with broadcast capabilities). Each interface and neighbor have its own finite-state automaton reflecting their current state.

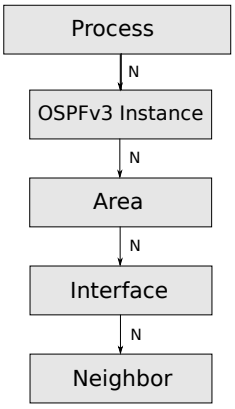

Figure 3. Hierarchy of inner classes

5. Testing

Each event in OMNeT++ is represented by a message. This could be a packet delivery, timer expiration or any other event which requires the module to react. This creates a bit difficult situation when it comes to comparing the outcome of the simulation with a real network. There are two methods I'm using to validate the model.

5.1 Wireshark

It is possible to create a real network (or a real time simulation in GNS3 for instance) and capture a communication between network devices running OSPF with Wireshark. The simulation creates a message log which represents the order of messages that were received by certain modules. The order of packets in wireshark should be more or less the same as the order of messages received by modules. An example of comparison is in figures 4 and 5.

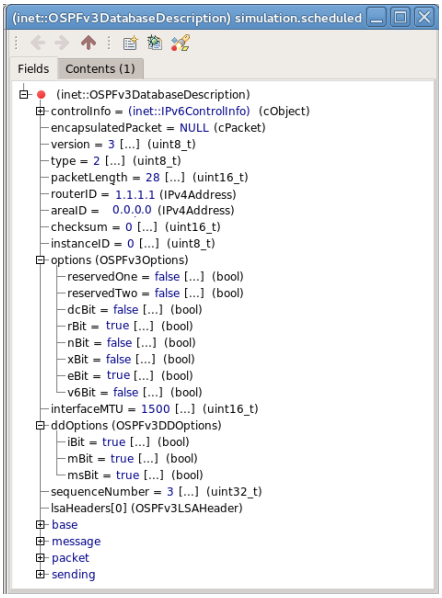

Figure 4. OSPFv3 packet in OMNeT++

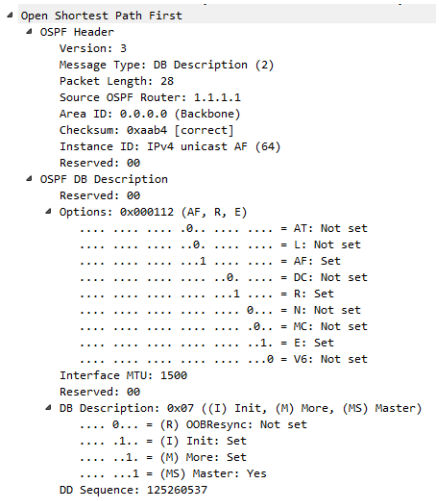

Figure 5. OSPFv3 packet in wireshark

5.2 Data Structures

After the real network has converged, it is possible to print all the data structures which are associated or in any way modified by the OSPF process. These include the routing tables, the neighbor and interface data structures as well as the OSPF data structures. These data structures should contain the same information both in the simulation and in the real network.

6. Conclusions

This work focuses on creating a simulation model of link-state routing protocol OSPFv3. It is being implemented in OMNeT++ simulation environment. It will improve the current OSPF model present in the INET framework.

The most important contributions are the capability to route both IPv4 and IPv6 internet protocols, the support of multiple processes running on a single device and the implementation of the NSSA option. The greatest benefit is that the OSPFv3 model has not yet been implemented by any other network simulation tool similar to OMNeT++.

Acknowledgements

I would like to express my gratitude to my supervisor Ing. Vladimír Veselý for his support and guidance.

References

[1] John Moy. Ospf version 2. RFC 2328 https://tools.ietf.org/html/rfc2328, April 1998. Updated by RFCs 5709, 6549, 6845, 6860, 7474.

[2] Rob Coltrun, Dennis Fergusson, John Moy, and Acee Lindem. Ospf for ipv6. RFC 5340 https://tools.ietf.org/html/rfc5340, July 2008. Updated by RFCs 6845, 6860, 7503.

[3] Omnet++ discrete event simulator. [online]. [cit. 2016-04-24].

[4] Inet framework. [online]. [cit. 2016-04-24].

[5] Ansa. [online]. [cit. 2016-04-24].

[6] Pat Murphy. The ospf not-so-stubby area (nssa) option. RFC 3101 https://tools.ietf.org/html/rfc3101, January 2003.

[7] Information Sciences Institute University of Southern California. Internet protocol. RFC 791 https://tools.ietf.org/html/rfc791, September 1981. Updated by RFCs 6845, 6860, 7503.

[8] Stephen E. Deering and Robert M. Hinden. Internet protocol, version 6 (ipv6) specification. RFC 2460 https://tools.ietf.org/html/rfc2460, December 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112.

[9] A. Lindem, Ed. Ericsson, S. Mirtorabi, A. Roy, M. Barnes, and R. Aggarwal. Support of address families in ospfv3. RFC 5838, April 2010.