BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

MODELING OF OSPFV3 LINK-STATE ROUTING PROTOCOL

MODELOVÁNÍ LINK-STATE SMĚROVACÍHO PROTOKOLU OSPFV3

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. MICHAL RUPRICH**
AUTOR PRÁCE

SUPERVISOR **Ing. VLADIMÍR VESELÝ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2017

Zadání diplomové práce/19348/2016/xrupri00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů   Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Ruprich Michal, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Modelování link-state směrovacího protokolu OSPFv3**
Modelling of OSPFv3 Link-State Routing Protocol
Kategorie: Počítačové sítě

Pokyny:

1. Analyzujte směrovací protokoly pracující na principu link-state, konkrétně protokoly OSPFv2 a OSPFv3.
2. Zjistěte stav implementace link-state protokolů v OMNeT++.
3. Prostudujte dostupnost a chování OSPFv3 na Cisco zařízeních.
4. Podle doporučení vedoucího implementujte podporu OSPFv3 protokolu v prostředí OMNeT++ a na příkladech demonstrujte činnost.
5. Ověřte chování modelu vůči reálné topologii a analyzujte výsledky.

Literatura:

- A. Varga, "OMNeT++ Discrete Event Simulation System", User Manual, 2004.
- J. Moy, "RFC 2328 - OSPF Version 2", IETF, 1998.
- R. Coltun, "RFC 5340 - OSPF for IPv6", IETF, 2008.
- J. Moy, "RFC 1584 - Multicast Extensions to OSPF", IETF, 1994.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese
http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Veselý Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstract

The thesis deals with simulation of routing protocols. The aim is to create a functioning model of OSPF link-state protocol in the simulation framework OMNET++. OMNET++ is a discrete simulation environment which was created to provide means to build models of various network protocols and technologies. Chapters in the first part of the thesis focus on the theoretical foundation of OSPFv2 and OSPFv3 and their differences. Important data structures, finite state automata and communication techniques are described and the information is later used to implement the model itself. The chapters in the second part deal with the implementation of the model in C++. The created model reflects the functionality of OSPF on Cisco devices.

Abstrakt

Tato práce se zabývá tvorbou simulací směrovacích protokolů. Cílem práce je vytvořit fungující model směrovacího protokolu OSPF v simulačním prostředí OMNET++. OMNET++ je diskrétní simulační prostředí, které bylo vytvořeno za účelem tvorby modelů různých síťových protokolů a technologií. Kapitoly první části práce se zabývají teoretickým základem fungování protokolů OSPFv2 a OSPFv3 a jejich rozdíly. Jsou zde detailně rozebrány důležité datové struktury, konečné automaty a komunikační prostředky, na jejichž základě je pak implementován samotný model. V kapitolách druhé části je popsán postup při implementaci modelu v programovacím jazyce C++. Vytvořený model odpovídá funkcionalitě protokolu OSPF na zařízeních společnosti Cisco.

Keywords

OSPF, OSPFv3, network modeling, OMNET++

Klíčová slova

OSPF, OSPFv3, modelování sítí, OMNET++

Reference

RUPRICH, Michal. *Modeling of OSPFv3 Link-State Routing Protocol.* Brno, 2017. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Vladimír Veselý, Ph.D.

Modeling of OSPFv3 Link-State Routing Protocol

Declaration

Hereby I declare that this master's thesis was prepared as an original author's work under the supervision of Ing. Vladimír Veselý, Ph.D

. . . .
Michal Ruprich
May 23, 2017

Acknowledgements

I would like to thank my supervisor Ing. Vladimír Veselý, Ph.D for his very helpful notes and suggestions.

As a thank you I would like to share a recipe for my favorite Japanese dish. It is called Oyakodon. It is simply a bowl of rice with chicken and eggs on top. Oyako means parents and children, here symolized by chicken and egg, and don(donburi) means a bowl. This recipe is for one serving.

We need $\frac{1}{4}$ cup of Dashi(or at least a chicken broth since Dashi is sometimes hard to get), $\frac{1}{2}$ tablespoon(tbsp) of sugar, $\frac{1}{2}$ tbsp of Sake, $\frac{1}{2}$ tbsp of Mirin, 1 tbsp of soy sauce, $\frac{1}{4}$ of onion, thinly sliced, 1 chicken thigh, cut into bite size pieces, 1 egg, $\frac{1}{2}$ green onion, thinly sliced, steamed rice.

Put Dashi, sugar, Sake, soy sauce and Mirin in a pan. Heat until it is boiling. Add onion and cook for a couple of minutes on medium heat. Add chicken pieces to the pan and cook until the meat is cooked through. Beat the egg in a small bowl and pour over the chicken. Cover and cook for 30 seconds to a minute.

Put the steamed rice in a bowl. Carefully slide the egg with the chicken and sauce onto the rice. Sprinkle with the green onion and serve.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Computer simulation plays a very important part in the process of building any system. It is a very powerful tool which 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 network to interconnect its employees, departments and even distant branches together to improve communication and cooperation between them. To design any such network 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.

OMNET++ is a C++ library and framework primarily used for creating network simulations. It provides a component-based architecture. Each component can be created in C++ and then put together to form more complex models. This approach allows users to create a model or a simulation of almost anything they could think of. This thesis aims at creating an authentic model of OSPF routing protocol. The protocol will be represented as a separate module which can be used by a router to simulate OSPF processes and communication.

The first part of this thesis consists of four chapters. Chapters two and three focus on describing in detail how OSPF works. Chapter four specifies differences and new capabilities in OSPF for IPv6. Even though the core of the protocol stays basically the same, there are some differences due to the use of IPv6. The fifth chapter shortly describes OSPFv3 capabilities on Cisco routers.

The main focus of the second part is the implementation and testing. Chapter six is a brief insight into the developmnet environment. Last two chapters describe implementation and testing details.

Chapter 2

Dynamic Routing

This chapter briefly describes dynamic routing and places the topic of this thesis in a wider context. Open Shortest Path First (OSPF) is a dynamic routing protocol. For the purpose of the future analysis, it is convenient to be aware of its categorization, as it will later be used for comparison with other routing protocols. Later we will use this knowledge to compare it to other routing protocols.

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 the routes in a network manually, dynamic routing offers much greater network sustainability and scalability. The main tasks of every dynamic routing protocol are as follows:

- 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

Routing protocols can be divided into two main categories: Interior Gateway Protocols(IGP) and Exterior Gateway Protocols(EGP).

2.1 IGP

An IGP works within the bounds of an autonomous system (AS). An AS is a collection of routers under a common administration. An IGP could be further divided into two groups based on methods the protocols use to determine the best paths: distance-vector protocols and link-state protocols.

Distance vector protocols advertise known routes to their neighbors as vectors of distance and direction. Distance is defined in terms of a metric and direction is simply the next hop router on the way to the destination network. The core of these protocols is usually built around Bellman-Ford algorithm, which is used to calculate the best path route. As this algorithm only knows the routing information received from its neighbors, it cannot build its own topology of the whole network. This is the main reason distance-vector protocols are used in small and flat networks. The most widely used distance-vector protocols are RIP, RIPv2, IGRP and EIGRP.

Link-state protocols are more sophisticated and more difficult for an administrator to maintain, 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 - use the same algorithm for best path calculation, i.e. the Dijkstra algorithm also known as the shortest path algorithm.

2.2 EGP

EGP are protocols that exchange routing information between autonomous systems. The best-known protocol and basically the only one which is widely used is Border Gateway Protocol (BGP). Most internet service providers (ISPs) use BGP to exchange information about areas they administer. The way BGP works may be demonstrated on a system of multiple autonomous areas, each using OSPF to provide routing information within its borders. Such system is on the whole too big to be scalable with OSPF itself, BGP would be used to link these areas into one huge network.

Chapter 3

Basic Principles of OSPF

This chapter focuses on describing the OSPF protocol and all of its functions and capabilities. It is crucial to understand these procedures in order to be able to create a working simulation of this protocol.

The OSPF has two versions - version 2 (OSPFv2), which was created to be used with network protocol IPv4, and version 3 (OSPFv3), which was created as an extension of OSPFv2, so that it could support newer version of IP network protocol IPv6. OSPFv2 is defined in RFC 2328[8] and OSPFv3 in RFC 5340[6]. However, the basic principles described in this chapter apply to both versions. The main differences, which are mostly connected with IPv6 and were introduced in OSPFv3, are described in chapter 4.

3.1 OSPF Process

As was mentioned in 2.1, OSPF is a link-state IGP based on the Shortest Path First (SPF) technology. It 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 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. OSPF also supports Variable Length Subnet Masks (VLSM) and route summarization, uses areas for scalability and possesses many more features that will be described in this chapter.

Every router running the OSPF process must go through the following three basic stages:

1. Neighbor discovery
2. Topology exchange
3. Route computation

After these three stages, every router should have established the best paths to distant networks in the routing table. Provided the OSPF is set correctly, the network should be converged. Each of these stages will be described in detail below.

3.2 OSPF Packets

OSPF runs directly above the IP and uses the number 89 in the protocol field in the IP packet header. Every OSPF packet starts with a 24 byte header. The information included in the header is important for determining, whether the router should even consider processing this packet. The structure of the packet is shown in the picture 3.1.

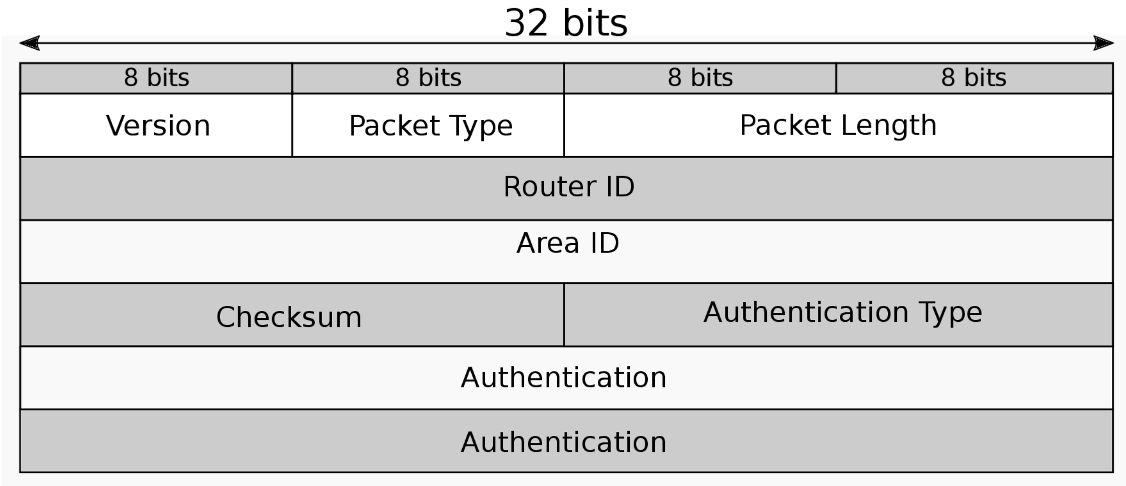

Figure 3.1: OSPF Packet Header

Version – determines whether OSPFv2 or OSPFv3.

Type – determines one of five different packet types.

Packet Length – the length of the packet in bytes, including the header.

Router ID – a unique ID of the router, which originated the packet.

Area ID – a 32-bit number identifying the area the router belongs to.

Checksum – standard IP checksum. The authentication field is excluded from the calculation.

AuType – the type of authentication procedure used for the packet.

Authentication – a 64-bit authentication field.

There are five different types of OSPF packets, each with a specific function:

1. Hello
2. Database Description
3. Link State Request
4. Link State Update

5. Link State Acknowledgment

Hello packets are described in 3.4.1. These packets are used to exchange initial information about routers in the routing domain. The remaining four types, which are used to exchange topology information, will be described in 3.5.

3.3 Multiple Areas in an AS

As mentioned in 3.1, OSPF enables the AS to be split into multiple areas. With the use of multiple areas, it does no longer apply that every router in an AS has the same link-state database. Each router holds a separate copy of a link state database for each area it is connected to. Routers inside an area are unaware of other areas' topologies, merely possessing information about the collection of routes reaching these areas. This results in a significant reduction in routing traffic. The amount of routing information that each router has to process is decreased.

3.3.1 Router Classification

OSPF uses a number of terms to describe routers regarding their location and function in a multiple area AS:

Internal Router – a router only belonging to one area.

Area Border Router (ABR) – a router connected to multiple areas. It is responsible for creating and distributing routes reaching other areas; any other information from these areas is filtered.

Autonomous System Border Router (ASBR) – a router connected to multiple ASs. These ASs may run other routing protocols, and the ASBR is responsible for redistributing routing information among multiple protocols.

Backbone Router – a router that has at least one interface in the backbone area (3.3.2).

Various types of routers may overlap. For example, an ABR might also be an ASBR and a backbone router. Furthermore, a backbone router does not necessarily need to be an ABR or an ASBR, but only an internal router. Each router generates specific types of link state advertisements. These will be described in 3.5.1

3.3.2 Backbone Area

When linking multiple areas together, there always needs to be a special area called the backbone area. The backbone area ID is always set to 0 (or 0.0.0.0). Routers in this area are responsible for distributing routing information between other areas (non-backbone areas). All ABRs have interfaces in the backbone area. This does not necessarily mean that all areas need to be physically contiguous. To establish connectivity to the backbone area, OSPF uses virtual links.

This does not mean that the backbone area must be used every time. For instance, we can create a OSPF topology containing a single area. This area's ID does not need to be the backbone ID.

Virtual link is created in a situation, when a direct connectivity between a non-backbone area and the backbone area is physically difficult or impossible. Virtual link may be also used in a situation, when the backbone area itself is partitioned. The virtual link is created between two or more ABRs in a transit area. The transit area must have full routing information and it cannot be a stub area (definition of stub area is in 3.5.2). When a virtual link is created, OSPF treats a pair of routers as if they were connected through the backbone area.

3.4 Neighbor Discovery

Before any data exchange takes place, every router must establish adjacencies with neighboring routers. The discovery of neighbors is either dynamic, using the Hello Protocol, or static, where every neighbor is manually configured.

3.4.1 Hello Protocol

The Hello Protocol is responsible for establishing neighbor relationships. Hello packets are generated periodically every 10 seconds by default and work as a keep-alive mechanism. The structure of the Hello packet is shown in the picture 3.2. The router starts the discovery process by sending Hello packets out all interfaces that participate in the OSPF process. In order for two routers to establish an adjacency, they need to agree on multiple values:

Subnet – both routers must be connected to the same subnet.

Hello and dead intervals – the values of hello and dead timers must be identical.

Area ID – both routers must share a common area.

Type of area – the type of the area (a stub area or a normal area) must correspond.

Authentication – if authentication is used, both the type and the password must match.

MTU – both routers must have the same MTU on the link, a mismatch might result in improper operation of topology exchange.

Router ID – the router ID must be unique in the routing domain.

If all the values described above match on both sides, routers would become neighbors and an exchange of topology data may begin.

The router ID is a 32-bit number that uniquely identifies a router within a routing domain. If two neighboring routers have the same ID, they will not establish an adjacency.

3.4.2 Dead Interval

Dead interval states the time during which at least one Hello packet must be received from the neighbor. If no packet is received during this period, all neighboring routers on this link are perceived as unavailable. A dead interval is set to four times the value of a hello interval by default. Dead timer values influence the time of convergence after a link failure.

3.4.3 Network Type

There are three types of networks, each type behaving differently in relation to Hello messages and neighbor discovery:

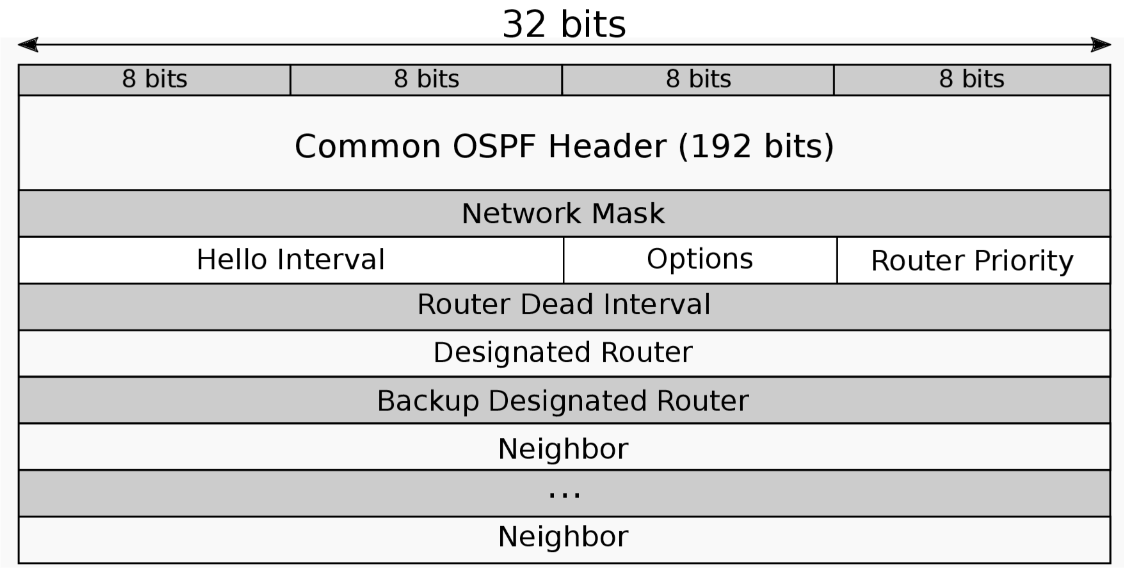

Figure 3.2: OSPF Hello Packet Structure

Broadcast network – a network joining multiple routers together with the capability to send a message to all attached routers (broadcast).

Non-broadcast network – a network joining multiple routers together but lacking the broadcast capability.

Point-to-Point network – a network with a single pair of routers.

In a Broadcast network, every router periodically sends Hello packets to a multicast address 224.0.0.5. Every router running the OSPF process should receive packets on this address, thus allowing a dynamic discovery of neighbors in the network. Bidirectional communication is established when a router sees itself in the neighbor field of the Hello packet received from a neighbor.

In a Point-to-Point network, Hello packets are used to exchange the information described in 3.4.1, but the neighbors must be discovered by other means, such as the Inverse ARP; they may also be configured manually.

In a non-,broadcast network, the Hello packet cannot be multicasted. Therefore, other mechanisms need to be exploited to overcome this issue. There are two modes that are used with non-broadcast networks when running the OSPF. In the first mode, called multiple-access non-broadcast, or NBMA, Hello packets are sent to each of the neighbors one at a time, rather than be multicasted. In the second mode, called Point-to-Multipoint, the non-broadcast network is perceived as a collection of Point-to-Point links.

3.4.4 Designated Router and Backup Designated Router

The concept of a Designated Router (DR) is used to decrease the amount of traffic during topology exchange in Broadcast and NBMA networks. Whenever there are at least two routers in a network, they elect a DR. The election process is described in 3.4.5. Every DR has two main objectives:

- It becomes adjacent to every router in the subnet. Each router in the subnet only exchanges routing information with the DR. The DR distributes topology information to every router in the subnet. This results in a decrease of traffic generated during the topology exchange and the link-state database of each router is therefore much smaller.
- It generates a network-LSA. This LSA lists all routers that are attached to the subnet.

The Backup Designated Router (BDR) becomes adjacent with every router in the subnet, just like the DR; however, it does not generate any LSAs. The primary function of the DBR is to become the new DR, when the current DR fails. Since the BDR is already adjacent with all of the routers, in the event of DR failure, the new DR merely needs to send out LSAs announcing the new DR, rather than exchange the whole topology.

3.4.5 DR and BDR election

The DR and BDR are elected during the Neighbor Discovery phase; the election is based on the information included in the Hello packet. The values used for the election are router ID and router priority. Router priority is an 8-bit unsigned integer. When the priority is set to 0, the router is ineligible to become the DR.

The election process goes through the following steps:

1. Choosing the router with the highest priority to become the DR.
2. In the event of there being two or more routers with the same priority, choosing the one with the highest router ID.
3. Choosing the BDR as the router with the second highest priority or the second highest router ID.

If the router with the highest priority is connected to the network and DR and BDR are already elected, it will not enforce any new election until the DR or BDR fails. If the DR is down, the BDR will become the new DR, even should a router with higher priority be added to the network within the time frame between the last election and the failure. If the BDR becomes the new DR or fails, a new BDR is elected.

3.4.6 Neighbor Data Structure

For each neighbor, the router stores a neighbor data structure, which is used to describe the conversation between routers.

State – one of 8 states that serve as indicators of the adjacency conversation progress.

Inactivity Timer – a timer activated every time the Dead Interval expires.

Master/Slave – only the master can send a Database Description packet (DD), the slave can only respond.

DD Sequence Number – the sequence number of the DD packet which was last sent to the neighbor.

Last Received DD packet – used to determine, whether a received DD packet is a duplicate or not.

Neighbor ID – the ID of a neighboring router learned from the Hello packet.

Neighbor Priority – neighboring router priority learned from the Hello packet.

Neighbor IP Address – the IP address of the neighboring router's interface.

Neighbor Options – optional OSPF capabilities supported by a neighbor.

Neighbor's DR – the RID of the router, which is identified as a DR by the neighbor.

Neighbor's BDR – the RID of the router, which is identified as a BDR by the neighbor.

The information in the structure reflects the progress of adjacency establishment. The state information is directly connected to the finite state automata which is described in appendix B.

3.5 Topology exchange

After the initial neighbor establishment, every router starts to generate link state advertisements (LSAs), in order to distribute local routing topology to all the other routers in the OSPF domain. Each router floods the LSAs it creates, as well as the LSAs it has received from its neighbors. The flooding process is reliable, ensuring that all routers in an OSPF area have the same data. Each LSA received is stored in a link-state database (LSDB). When all routers have the same topology data, the information stored in the LSDB is then used to calculate the best route for each reachable subnet.

This section will begin with description of each LSA type. The second part focuses on giving a detailed explanation of the packet types used in topology exchange and the process of LSDB data exchange.

3.5.1 LSA Types

There are up to five different types of LSAs generated in the OSPF process. Each type has its specific function in the SPF calculation. The LSA types are described in table 3.5.1. There are six more types of LSAs. These types are not covered in the implementation part of this thesis but are mentioned for completeness in figure 3.5.1.

Each LSA has a common header shown in figure 3.5.1. Every LSA is identified by a 32-bit link state identifier (LSID), which is used to determine the source of the LSA.

LSA Type	Name	Description
1	Router	Type 1 of LSA contains the states and costs of the router's links to its neighbors and the neighbors' RIDs.
2	Network	Generated for each broadcast and NBMA network by the DR. It describes all routers attached to the network, including the DR itself.
3	Summary	Created by ABRs to represent subnets from other areas.
4	ASBR Summary	Advertises routes to reach the ASBR.
5	AS external	Created by the ASBR to distribute routing information injected into the OSPF by other routing protocols.

Figure 3.3: Standard LSA types

LSA Type	Name	Description
6	Group Membership	Defined for MOSPF - multicast extension of OSPF.
7	NSSA External	Similar to LSA type 5 but used in NSSA (3.5.2).
8	External Attributes	Used for BGP and OSPF interoperation.
9-11	Opaque	Created for future upgrades of OSPF. For example, LSA type 10 has been adapted for MPLS traffic engineering.

Figure 3.4: Non-standard LSA Types

3.5.2 Area Types

Area types are used in OSPF to control the amount of external routing information distributed through areas. By configuring the ABR's interface to the area as a stub interface, we suppress LSA types 4 and 5 from being passed through the ABR. Every LSA type 5 is converted to type 3. An area is usually configured as a stub when there is a single exit point from the area. In such case it would be unnecessary to flood all external routing information to all routers in the area. Instead, the ABR advertises itself as the default gateway for these LSAs.

The RFC 2328[8] defines only the stub area type as was described above. However, Cisco devices support three other area types as an extension of the stub area:

Totally Stubby Area - rejects LSA types 3,4 and 5. Every LSA type 3 and 5 is converted as default route and flooded as LSA type 3.

Not-So-Stubby Area (NSSA) - it has similar functionality as the stub area but it allows certain external routes to be transited through the area. Any external routing information is carried as LSA type 7. When the LSA leaves the NSSA, it is converted to LSA type 5. The NSSA is defined in [9].

Totally NSSA - it rejects LSA types 3,4 and 5 (similar to the Totally Stubby Area) but it converts the LSA type 5 to type 7 (similar to NSSA). Therefore, the external routing information is able to transit through the area.

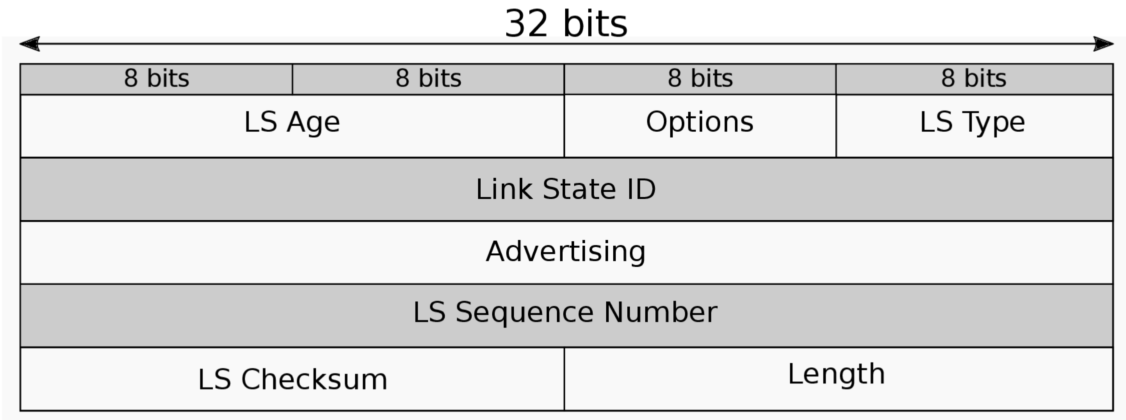

Figure 3.5: Common LSA header structure

3.5.3 Link State Database

LSDB is simply a collection of all LSAs the router has received from other routers in the domain. Each router has a separate instance of LSDB for each area it belongs to. All routers in a domain have the same LSDB after the exchange is completed. As was mentioned in section 3.2, there are four types of packets used to exchange topology data among adjacent routers.

3.5.4 Database Description

Database Description packet (DD) describes a set of LSAs belonging to the routers database. DD packets are sent on master/slave basis; one of the routers in an adjacency is the master, the other is the slave. The master sends DD packets to the slave (polling) and the slave acknowledges them by sending its own DD packets. Multiple DD packets may be sent to describe one LSDB. The responses are linked via the DD sequence field in the header. Structure of DD packet header is in figure 3.5.4. There are a few header fields which may need an explanation:

0 – these fields are reserved and must always be set to 0.

Options – The optional capabilities supported by routers. Some of these options are mandatory and some are optional. However, if there is a capability mismatch between two neighboring routers, they are usually unable to form a neighborship or a router performing SPF calculation will not include the router with different capabilities in the SPF Tree.

I-bit – the Init bit indicates that this packet is the first DD packet.

M-bit – the More bit indicates that more DD packets will follow.

MS-bit – the Master/Slave bit expresses which router is the master and which is the slave. Setting this bit to 1 indicates that the router is the master.

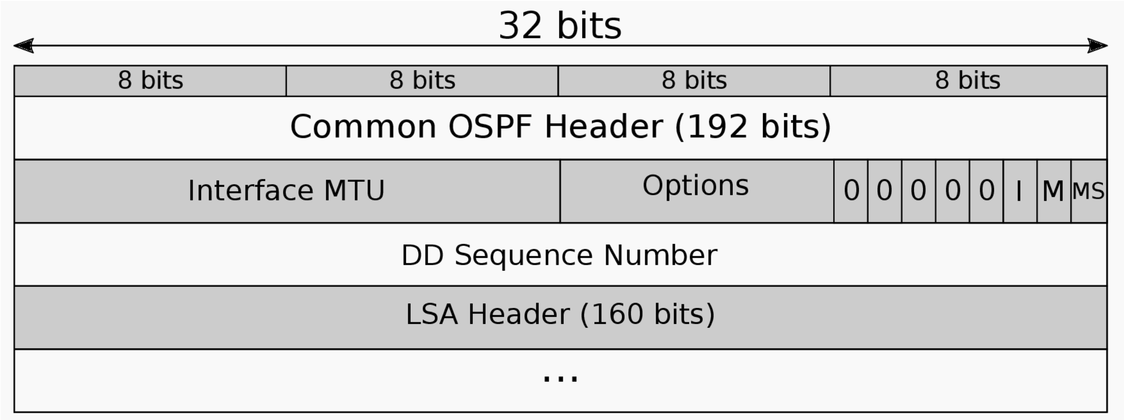

Figure 3.6: Database Description packet header

3.5.5 Link State Packets

OSPF packet type 3 is a Link State Request (LSR). After receiving a DD packet from a neighbor, router sends LSRs to request LSAs which are more up-to-date than LSAs in his own LSDB. The requested LSA is specified by LS Type, LS ID and the Advertising Router. This unique specification of a particular LSA is understood as a request for its latest instance. The LSA received from the neighbor after a request may be already newer than the LSA describer in the DD packet.

Link State Update packet (LSU) is OSPF packet type 4. It is used by the flooding process to distribute LSAs to the other routers. LSU packets are multicasted on those networks that have multicast capabilities. Each LSU carries a collection of LSAs one hop further from their origin.

Link State Acknowledgement packet is OSPF packet type 5. Each LSA received by the router is acknowledged by including its header in the Link State Acknowledgement header. Acknowledgements may be sent either as a multicast to AllSPFRouter address or AllDRouters, or they may be sent as unicast. Multiple LSAs may be acknowledged in one packet.

3.6 Route Computation

This section will describe the shortest path tree construction on an example LSDB. The SPF Tree is constructed using the Dijkstra's Algorithm which is beyond the scope of this thesis. What follows is a very brief description of the basic mechanisms used by the router to populate the routing table. An example network is introduced in figure 3.6. On this network we will demonstrate how the SPF algorithm works and how it populates the routing table with the information gathered from other routers in the area.

3.6.1 Directed Graph

The information stored in LSDB is used to calculate paths to distant networks. The LSDB in an AS describes a directed graph with vertices consisting of routers and networks.

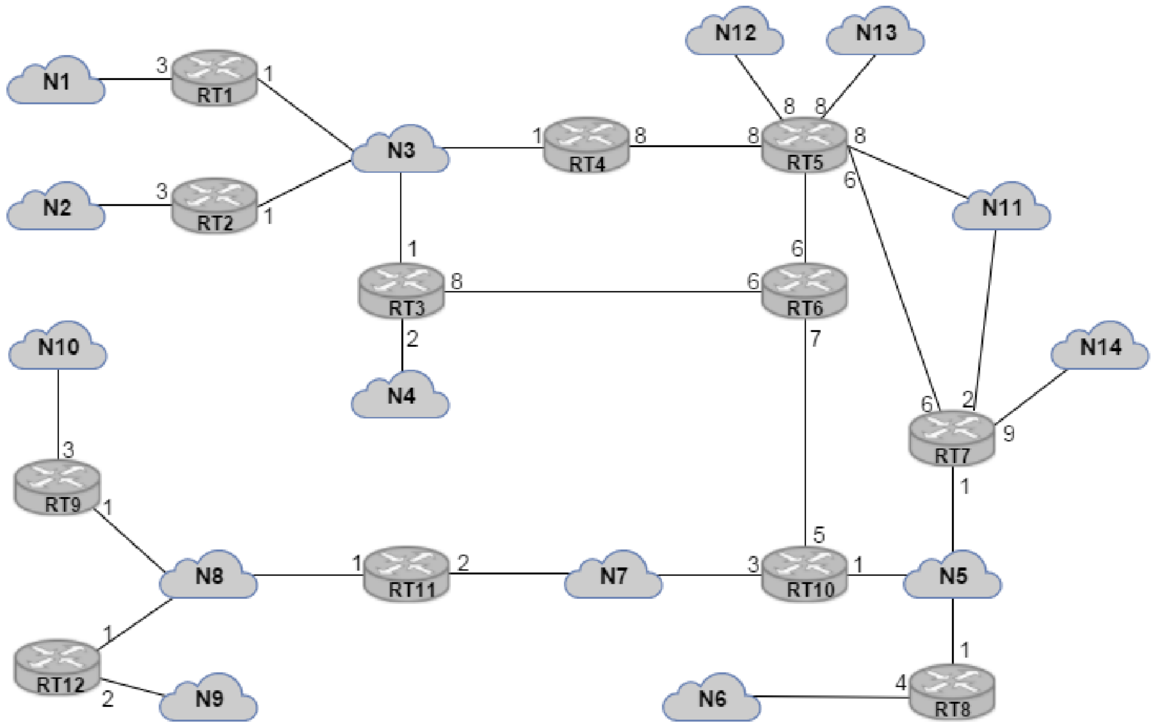

Figure 3.7: Example of network for LSDB structure visualization

A graph edge connecting two routers indicates that they are connected by a physical point-to-point interface. If there is an edge between a router and a network, it demonstrates that the router has an interface in this network. If a network has only one router's interface connected, it is a stub network. If there are more than one routers connected to a network, it is a NBMA or a broadcast network. Each interface is evaluated with an integer value representing its cost. This cost is either configured by an administrator on a per-interface basis or it is calculated from interface bandwidth value. The lower the cost, the more likely the interface is chosen to forward data traffic.

The figure 3.8 depicts a directed graph created from gathered LSAs represented by a table. Each intersection evaluated by a number is the cost on a connection from a router or a network in the corresponding column to a router or a network in the corresponding row. If there is no value in the intersection, it means that there is no direct connection between the two elements. A cost from each network is zero because they only represent network segments but not any existing interface which could be associated with a cost.

3.6.2 Shortest Path Tree

When the directed graph is constructed, it is used to create a Shortest Path Tree (SPF Tree) using the Dijkstra's algorithm. The tree represents every possible path to any destination network or host. Each router calculates its own SPF Tree with itself as a root. The tree is used to populate the routing table with next-hop addresses to distant networks. There is a separate SPF Tree for each area. SPF Tree, with the RT6 router as the root, constructed from example network is shown in figure 3.9.

	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	N3	N6	N7	N8
RT1													0			
RT2													0			
RT3						6							0			
RT4					8								0			
RT5				8		6	6									
RT6			8		7					5						
RT7					6									0		
RT8														0		
RT9																0
RT10						7								0	0	
RT11															0	0
RT12																0
N1	3															
N2		3														
N3	1	1	1	1												
N4			2													
N5							1	1		1						
N6								4								
N7										3	2					
N8									1		1	1				
N9												2				
N10									3							
N11					8		2									
N12					8											
N13					8											
N14							9									

Figure 3.8: Example network transformed to a directed graph represented by a table structure

3.6.3 Next Hop Calculation

The next hop calculation is invoked each time a shorter path is found to reach the destination. It can happen in any stage during the SPF Tree construction or when a change occurs in the topology and the tree needs to be recalculated. The resulting routing table will contain next hops to destinations reflecting the absolute shortest paths in the tree. The candidate next hop is either the destination itself or the parent node between the destination and the root. If there is at least one router between the root and the destination, the destination inherits the set of next hops from the parent. If the parent is the root it means that the destination is a router or a network directly connected to the root. The next hop in this case is simply the OSPF interface connecting the root to the router or the network. The routing table of RT6 router from the example above is shown in figure 3.6.3

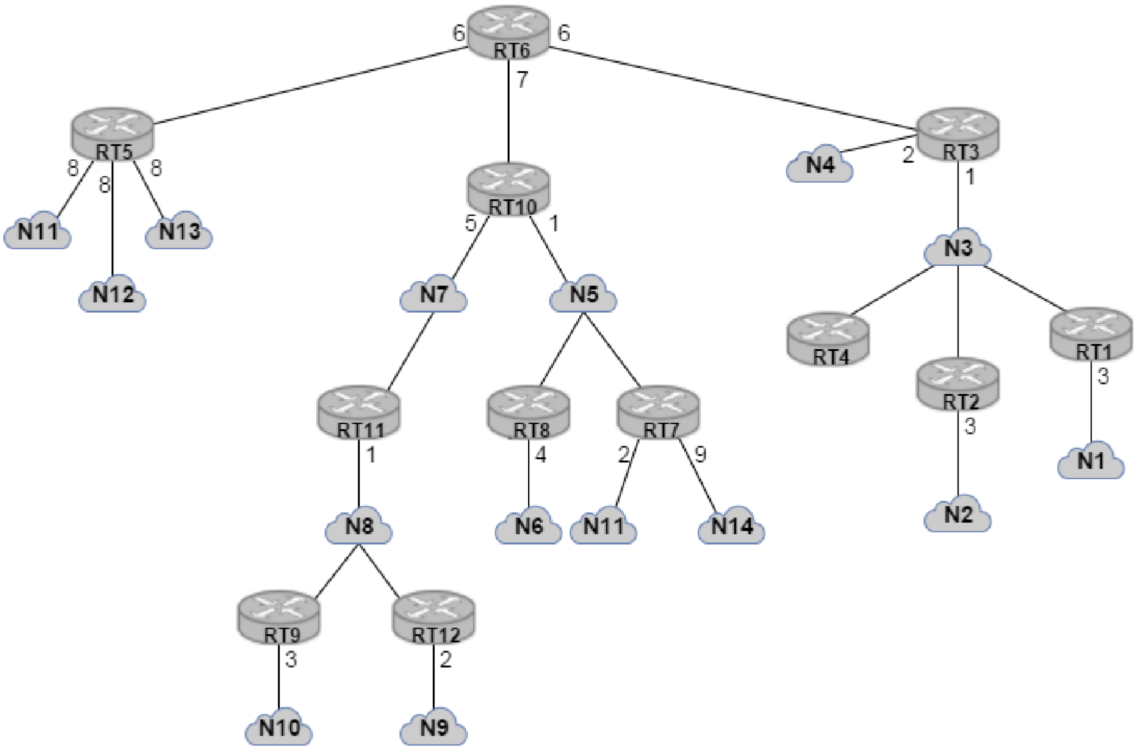

Figure 3.9: The SPF Tree constructed from example network in 3.6

Destination	Next Hop	Cost
N1	RT3	10
N2	RT3	10
N3	RT3	7
N4	RT3	8
N5	RT10	8
N6	RT10	12
N7	RT10	10
N8	RT10	11
N9	RT10	13
N10	RT10	14
N11	RT10	10
N12	RT5	14
N13	RT5	14
N14	RT10	17
RT5	RT5	6
RT7	RT10	8

Figure 3.10: The routing table on RT6 router from the example network

Chapter 4

OSPF for IPv6

The OSPF Process running over IPv4 and IPv6 is, in most parts, the same. Slight changes have been introduced in OSPFv3 due to different semantics of IPv4 and IPv6 protocols and because of the increased size of IPv6 packet header. This chapter describes main differences which are essential for correct implementation of OSPFv3.

IPv6 described in [7] introduces the term „link“ which indicates „a communication facility or medium over which nodes can communicate at the link layer.“ This means the terms „subnet“ and „network“ used in OSPFv2 should be replaced by the term link. With IPv6 there may be multiple subnets assigned to a single link and two nodes can communicate over a link, even if they do not share the same subnet. OSPFv3 thus runs per-link instead of per-IP-subnet.

IPv6 as it is implemented in OMNeT++ is described in [11].

4.1 Packet Format Changes

This section illustrates main differences introduced in packet structures in OSPFv3. Fields which are new, changed or were left out in the new version of OSPF will be described in detail. Fields which are the same as in the previous version were already described in chapter and will not be mentioned here.

Figure 4.1 shows changes of the common OSPFv3 Packet header. The authentication fields were left out because OSPFv3 relies on IP Authentication Header and the IP Encapsulating Security Payload.

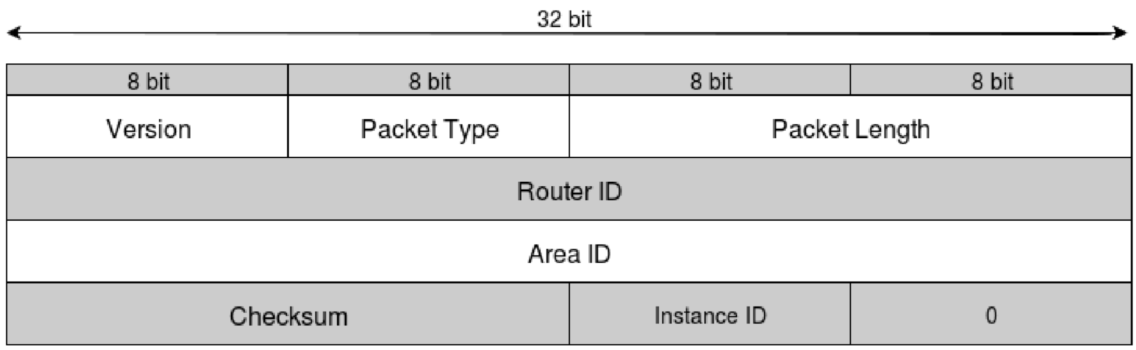

Figure 4.1: The structure of OSPFv3 common packet header

Because OSPFv3 may run multiple instances on a single link, each instance needs its own ID. The Instance ID is a new 8-bit field and it holds a value assigned to a single instance of OSPF. The value has only link-local significance.

The figure 4.2 shows the structure of Hello Packet used in OSPFv3. Because multiple IPv6 subnets may be assigned to a single link, even if they share a common subnet, the Network Mask field is no longer needed and it has been removed. Any information about the address has been removed; rather the new Hello Packet contains Interface ID field. The Interface ID is a 32-bit number which uniquely identifies router's interface connected to a particular link. This value is also used as network-LSA's Link State ID in a situation when the router becomes the DR.

The Router Dead Interval field's size has been reduced from 32 bits to 16 bits. Also the Options field's size has been increased from 8 bits to 24 bits and it is described below.

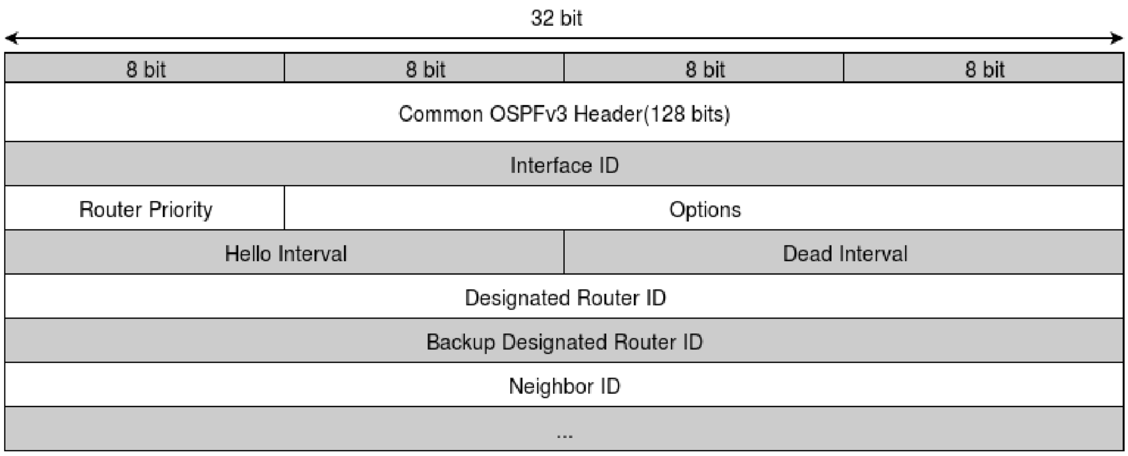

Figure 4.2: The structure of Hello Packet in OSPFv3

The figure 4.3 shows the structure of the new DD packet. Except for new OSPF Header and new 24-bit Options field, it is almost the same as it was in previous version.

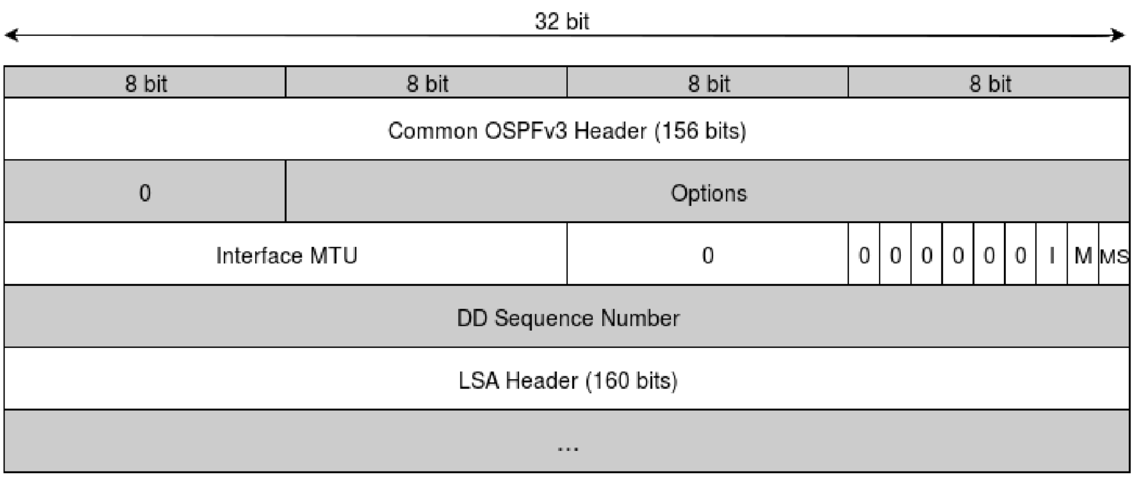

Figure 4.3: The structure of DD Packet in OSPFv3

The new longer Options field is described below. This field enables OSPF routers to indicate whether they are able to support additional capabilities. If two adjacent routers support different capabilities, it may result in variety of behaviours, depending on the particular option mismatch. Seven bits of the Options field have been assigned and each unrecognized bit should be reset by the router. The meaning of each bit is described below:

V6-bit – clearing this bit indicates that the router or link should be excluded from the route computation.

E-bit – this bit influences the way the AS-external-LSAs are processed.

x-bit – this bit was previously used with MOSPF but this capability is deprecated with the OSPFv3 and this bit should be set to 0.

N-bit – this bit indicates whether the router is attached to an NSSA area.

R-bit – this bit indicates whether the originating router is an active router. Clearing this bit is useful for multi-homed hosts which want to participate in routing, but they do not want to forward packets which are not addressed in local scope.

DC-bit – this bit describes handling of the demand circuits.

***-bit** – these bits are reserved for migration of OSPFv2 capabilities.

4.2 Protocol Structures Changes

This section describes changes in main data structures such as Interface, Neighbor and Protocol data structures. Most parts remain the same except for a few adjustments due to new capabilities of OSPFv3.

The Interface data structure described in C has been modified so that it supports Interface ID and Instance ID information appearing in OSPFv3 Packet header or the Hello Packet header.

The Neighbor data structure described in 3.4.6 has been modified to contain information about neighbors in the form of IDs instead of IP addresses. The Neighbor's ID thus becomes the Neighbor's Interface ID and IP addresses are no longer used to identify the DR and BDR. To identify the DR and BDR we need to use the Routers' IDs instead. The Neighbor's IP address of the neighboring router's interface will now be the IPv6 link-local address of the neighbor.

4.3 Flooding Scope

The flooding scope was changed and now it is coded into LSA's LS Type field. How the flooding scope affects the processing of LSAs is described in 4.4. We recognize three new flooding scopes:

Link-local scope – as the name indicates, these LSAs are used only on a link-local scope.

Area scope – these LSAs are flooded only through a single area and no further.

AS scope – LSAs are flooded through the routing domain. They are originated from ASBRs.

4.4 New LSA Types

The structure of the remaining LSA packets remains almost the same except of course for the new OSPF Header. In the LSR packet, the LS Type field size has been reduced from 32 bit to 16 bit. In the LSA header the LS Type field size has been increased to 16 bits and it has replaced the original Options field. The upper three bits of the LS Type field now specify the flooding scope.

The first bit of the LS Type field is the U-bit and it specifies how the router should handle unknown LSAs. If the bit is set to 0, unknown LSAs will be treated as if they have only link-local flooding scope. If the bit is set to 1, any unknown LSA is stored and then flooded. The other two bits have the following meaning:

S1	S2	Description
0	0	Link-local flooding
0	1	Area scope flooding
1	0	AS scope flooding
1	1	Reserved

Figure 4.4: The second and third bits in the LS Type field of every LSA

OSPFv3 uses the same 7 types of LSAs as the OSPFv2. However some of them have been repurposed and also two new types have been introduced. The following table shows all 9 types of LSAs used in OSPFv3 with appropriate LS Type according to flooding scope described above:

OSPFv2		OSPFv3	
1	Router LSA	0x2001	Router LSA
2	Network LSA	0x2002	Network LSA
3	Network Summary LSA	0x2003	Inter-area Prefix LSA
4	ASBR Summary LSA	0x2004	Inter-area Router LSA
5	AS-External LSA	0x4005	AS-External LSA
6	Group Membership LSA	0x2006	Group Membership LSA
7	NSSA External LSA	0x2007	Type-7 LSA
		0x0008	Link LSA
		0x2009	Intra-area Prefix LSA

Figure 4.5: Comparison of LSA Types in OSPFv2 and OSPFv3

The Inter-area Prefix LSA is the equivalent of OSPFv2 type 3 LSA. It is originated by ABRs and it describes routes to address prefixes belonging to other areas. The prefix is described by the Prefix Length, Prefix Options and Address Prefix in the LSA body. As was mentioned at the beginning of this chapter, the network mask is no longer used. Link-local addresses should never appear in Inter-area Prefix LSA.

The Intra-area Router LSA is an equivalent of the OSPFv2 type 4 LSA. It advertises a route to an ASBR (a router which might be external to the area but it is internal to the AS). Each such LSA describes route to a single ASBR.

Link-LSA describes router's attached physical links; there is one LSA for each. These LSAs are never flooded beyond the associated link. Link-LSAs have three main purposes:

1. They provide link-local addresses to all other routers which are attached to this link.

2. They provide other routers with the information about all IPv6 prefixes which are associated with this link.

3. They allow the router to distribute a collection of Option bits in the network-LSA originated by the DR on a broadcast or NBMA network.

Intra-area Prefix LSA is used to advertise one or more IPv6 address prefixes that are associated with the router. The prefixes associated with a local router address and an attached stub network were previously advertised by using router-LSA. Attached transit network has been advertised via network-LSA. Since all the addressing semantics have been removed from all LSAs in OSPFv3, the Intra-area Prefix LSA is used for these purposes.

Chapter 5

Support of OSPF on Cisco Devices

This chapter describes OSPF capabilities supported on Cisco devices. It is important to see the difference between the standard defined in RFC and a real implementation used on routing devices. When implementing OSPFv3, I was following the RFC as well as a Cisco configuration guide [3].

5.1 Basic Configuration

Before configuring OSPFv3, IPv6 routing has to be enabled in the configuration mode:

```
Router(config)# ipv6 unicast-routing
```

Configuring OSPFv3 itself comes in two steps. The first step is to define the OSPFv3 process:

```
Router(config)# router ospfv3 process-id
```

This opens up the router configuration mode. In the router mode it is possible to define address family for the process:

```
Router(config-router)# address-family [ipv4 | ipv6]
```

And for each address family, the router ID may be defined here. The router ID may be specified either for the whole process or each address family separately:

```
Router(config-router)# router-id ip-address
```

```
Router(config-router-af)# router-id ip-address
```

The second step is to configure an interface that belongs to the process. In OSPFv2 this is done in the router configuration by specifying network address range that should be included in the process. Any interface that belongs to the range and is running is automatically included in the routing process.

In OSPFv3 each interface is configured separately in the interface configuration. First and foremost IPv6 protocol needs to be enabled on the interface:

```
Router(config-if)# ipv6 enable
```

After assigning an IPv6 address to the interface, the OSPFv3 itself may be configured as follows:

```
Router(config-if)# ospfv3 process-id address-family
area area-id instance instance-id
```

It is important to note that the process-id has lost the global meaning that it had with OSPFv3. Instead, the instance-id is used to separate OSPFv3 processes among other routers. If the instance-id is not explicitly given, it is assigned based on the address-family.

5.2 Interface Configuration Mode

Besides from basic configuration described in 5.1, the interface configuration mode may be used to configure other important aspects of the process. Following list describes the most important commands:

ospfv3 cost - Set a fixed cost of sending packets on an interface.

ospfv3 dead-interval - Sets the value of the dead interval on an interface.

ospfv3 hello-interval - Sets the value of the hello interval on an interface.

ospfv3 network - Allows user to configure OSPF network type(broadcast, NBMA, etc.) on an interface.

ospfv3 neighbor - When the interface is in NBMA network, this command allows user to explicitly define the neighbors on this interface.

ospfv3 priority - Sets the priority value on an interface. This value is used in the DR and BDR election.

ip ospf retransmit-interval - Allows user to specify the time between LSA retransmissions on an interface.

5.3 Troubleshooting

Troubleshooting commands are used in privileged exec mode. These are used mainly for testing purposes described in 8. They are either `debug` or `show` commands.

The `debug` commands are used as follows:

```
Router# debug ospfv3 process-id command
```

There are lots of commands that can be used for debugging but only a couple of them are used in this thesis for testing:

hello - Allows watching how Hello Packets are being received and sent. It shows the source and destination of the packet along with area and router ID that originated it. This is particularly useful in 8.1.

adj - All adjacency events are shown. This is mostly used to see when and if routers become neighbors and what are the changes in the neighbor state machine.

events - Other OSPFv3 related events are shown.

The `show` commands serve to troubleshoot general OSPFv3 settings. These commands are mostly used in the testing chapter 8 specifically in sections 8.2 and 8.4. Every `show` command may be used simply as:

```
Router# show ospfv3 process-id command
```

For the testing purposes, these commands are used in this thesis:

database - Shows complete OSPFv3 database for this router.

interface - Used to troubleshoot interface settings. This includes timers and priority settings.

neighbor - Shows all neighbors of this router along with their states.

More OSPFv3 commands are described in appendix D.

Chapter 6

Descrete Event Simulator OMNeT++

This chapter briefly describes development and simulation environment OMNeT++[10], INET[4] and the ANSAINET[2] frameworks.

6.1 OMNeT++

OMNeT++ is a modular, component-based C++ simulation library and framework. It is used to create discrete network simulation models. OMNeT++ is directly embedded into Eclipse and it uses its IDE.

Each simulation module's behaviour is implemented in C++. OMNeT++ uses its own language NED to describe the module's structure. Using NED language, it is possible to interconnect multiple simple modules and create more complex modules or networks. Modules can communicate with each other by sending messages.

Each simulation is described by another NED file, a configuration file (omnetpp.ini) and other XML files setting parameters to modules in the network.

6.2 INET

The INET project aims to provide a basic set of modules that may be used in OMNeT++ to create TCP/IP simulations quickly. It includes implementations of basic protocols like UDP, TCP, RTP, IP, ARP, Ethernet, PPP and more.

6.3 ANSAINET

ANSA(The Automated Network Simulation and Analysis) project is an extension of INET framework. It is being developed at the Faculty of Informatics at Brno University of Technology. The aim of this project is to provide tools for formal analysis of real networks.

Chapter 7

Design and Implementation

This chapter focuses on design and implementation details. I am describing the main OSPFv3 module, C++ classes hierarchy and what each class represents. Differences between the RFC standard and an actual Cisco implementation of the protocol are an important part of this chapter as well.

7.1 OSPFv3 Module

`OSPFv3Routing` is a compound module consisting of simple modules `OSPFv3Splitter` and `OSPFv3Process`. It is a part of `ANSA_Router`, and it is connected directly to `ANSA_MultiNetworkLayer` since it operates at the network layer.

The `OSPFv3Splitter` module is responsible for parsing configuration files and creating necessary data structures and objects. It examines every packet from the network layer and passes it to the right `OSPFv3Process` based on the incoming interface.

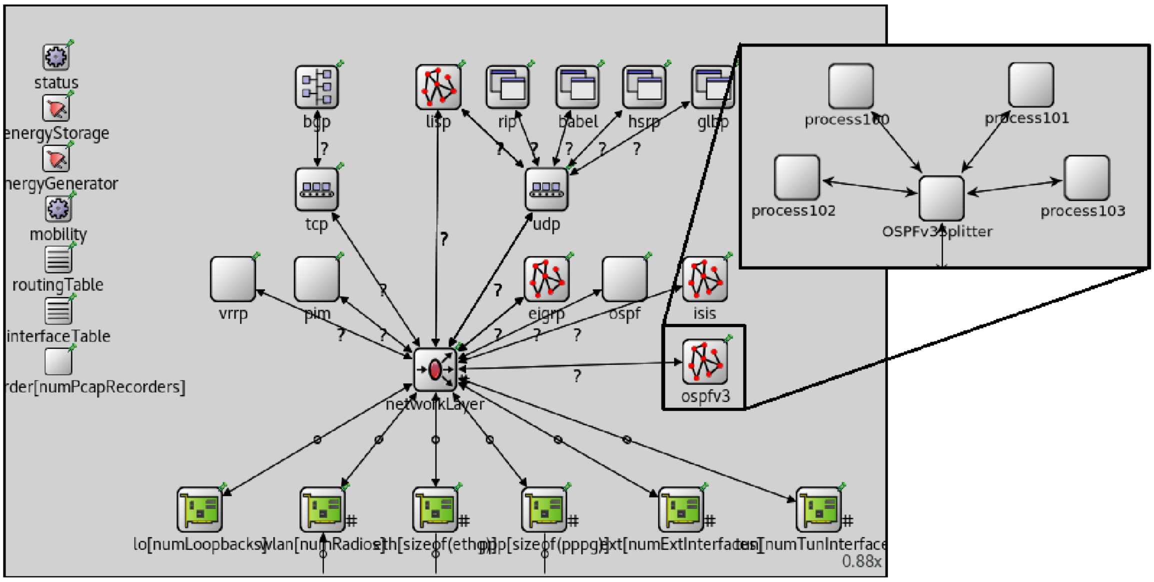

Figure 7.1: The OSPFv3 Module Inside ANSA_Router

The top-level structure providing OSPFv3 routing capabilities on a Cisco router is a process. A simple `OSPFv3Process` module represents each process. There may be up to 32 processes running on a Cisco router, and they are created dynamically by `OSPFv3Splitter` based on configuration specifications.

7.2 OSPFv3 Classes

A top-level class is the `OSPFv3Process` as it reflects the top-level structure on Cisco routers. There may be up to two processes for each interface as long as each of them is for a different address family. RFC 5340 states that there may be multiple instances on a single link. Cisco, on the other hand, allows only a single instance per process and address family. This model meets somewhere in the middle. It allows a maximum of two processes per interface, but each process may have multiple instances.

Each instance is represented by `OSPFv3Instance` class and has an integer ID. This ID appears in each OSPFv3 packet, and it is used to determine whether this packet should be processed or not. Since there is no information about the process in any packet, the `OSPFv3Splitter` duplicates every packet on arrival in case there are two processes configured for a single interface. The process itself then determines whether there is an instance with ID set in the packet. The scope of process loses its global importance as it used to have in OSPFv2 and has only local significance.

An instance may be further separated into multiple areas. `OSPFv3Area` class represents each area. Because the SPF tree is calculated for each area separately, each area has a separate database for LSAs generated by routers belonging to the area. One area is usually spread accross multiple interfaces on a single router.

Every interface that belongs to an area is implemented as an `OSPFv3Interface` class. Each interface may have multiple neighbors represented by `OSPFv3Neighbor`.

7.3 Configuration

The configuration file format is based on how the OSPFv3 protocol is configured on a Cisco router([3], [1]). Usual configuration of OSPFv3 routing takes two steps:

1. Setting OSPFv3 process along with address families and router ID for each family in the global configuration mode.

2. Separating OSPFv3 configuration on each interface. This involves area, address family and instance configuration.

The parameters in `config.xml` file used for each simulation have a very similar format. The router process is set in the `<Routing6>` section. This section may include a configuration for other routing protocols as well.

Each interface is set in the `<Interfaces>` section. A process, instance with address family and area, this interface belongs to, have to be specified.

```xml
<Router id="R1">
    <Routing6>
        <OSPFv3>
            <Process id="1">
                <RouterID>10.10.10.1</RouterID>
            </Process>
        </OSPFv3>
    </Routing6>

    <Interfaces>
        <Interface name="eth0">
            <Process id="1">
                <Instance AF="IPv6">
                    <InterfaceType>Broadcast</InterfaceType>
                    <Area>0.0.0.0</Area>
                </Instance>
            </Process>
            <IPv6Address>fe80::a8bb:ccff:fe00:100/64</IPv6Address>
            <IPv6Address>2001:db8:a::1/64</IPv6Address>
        </Interface>
        <Interface name="eth1">
            <Process id="1">
                <Instance AF="IPv6">
                    <InterfaceType>Broadcast</InterfaceType>
                    <Area type="stub">0.0.0.1</Area>
                    <RouterPriority>10</RouterPriority>
                </Instance>
            </Process>
            <IPv6Address>fe80::a8bb:ccff:fe00:110/64</IPv6Address>
            <IPv6Address>2001:db8:1::1/64</IPv6Address>
        </Interface>
    </Interfaces>
</Router>
```

Figure 7.2: Sample OSPFv3 Configuration file

Chapter 8

Testing

This chapter focuses on testing the OSPFv3 module and comparing it with a L3 Cisco devices. As a testing environment, I am using EVE-NG Virtual Environment[5]. Each Router is using `I86BI-LINUX-L3-ADVENTERPRISEK9-M, version 15.4(2)T4, DEVELOPMENT TEST SOFTWARE`.

The chapter is divided into five sections. Each section describes the particular phase of OSPFv3 operation.

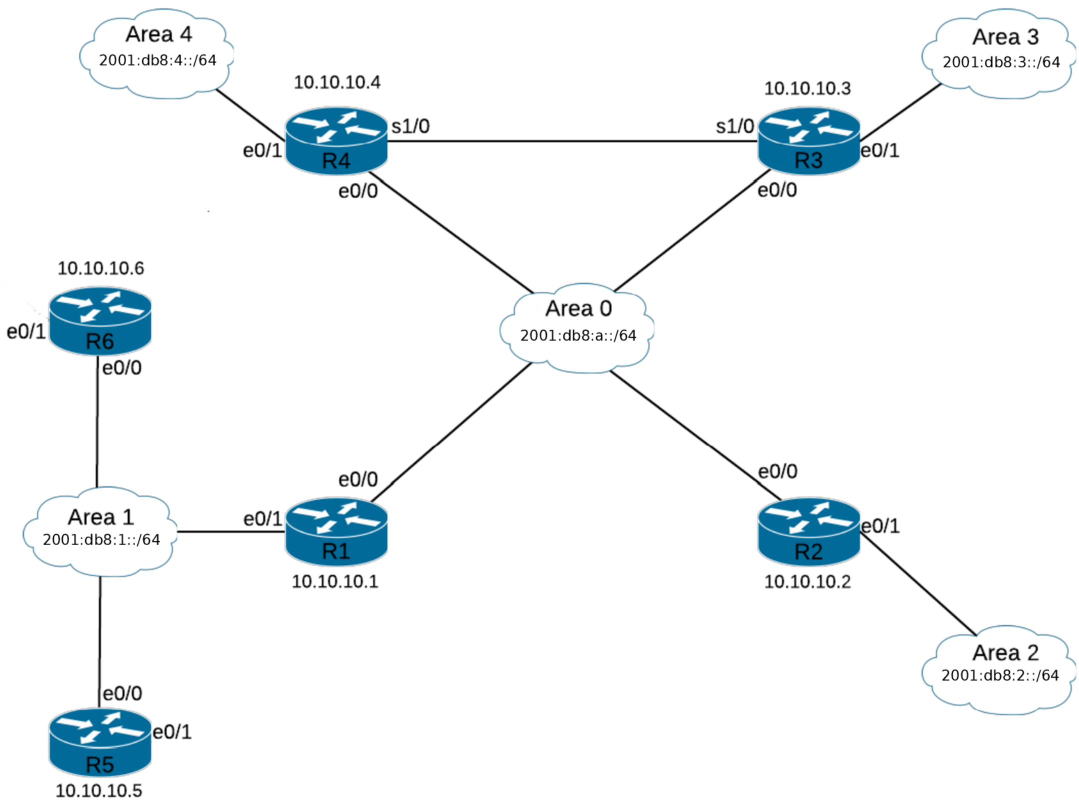

Figure 8.1: Testing Topology with Multiple Areas

The first phase focuses on the Hello Packet exchange and format. The second phase shows how neighbors are established and how the DR and BDR are elected. In the third phase, a database exchange takes place. The fourth phase shows the exchanged database after the routing processes are converged. In the final phase, I briefly describe what happens when a router or an interface fails in the topology.

Figure 8.1 shows a topology chosen for the actual testing. It consists of six routers and five different areas. Area 0 is the Backbone area used for LSA exchange. Area 1 is a stub area, other areas are normal.

In most demonstrations, I am using router R1 as a reference router. R1 is an ABR on the edge of a backbone and a stub area. It is responsible for originating a default prefix LSA for the stub area and for Inter-Area-Prefix LSA redistribution between different areas. The R1 is set to be the DR in Area 1 and DROTHER in Area 0. This makes the R1 router a perfect reference point for each of the testing phases.

8.1 Hello Protocol

8.1.1 The Hello Packet Format

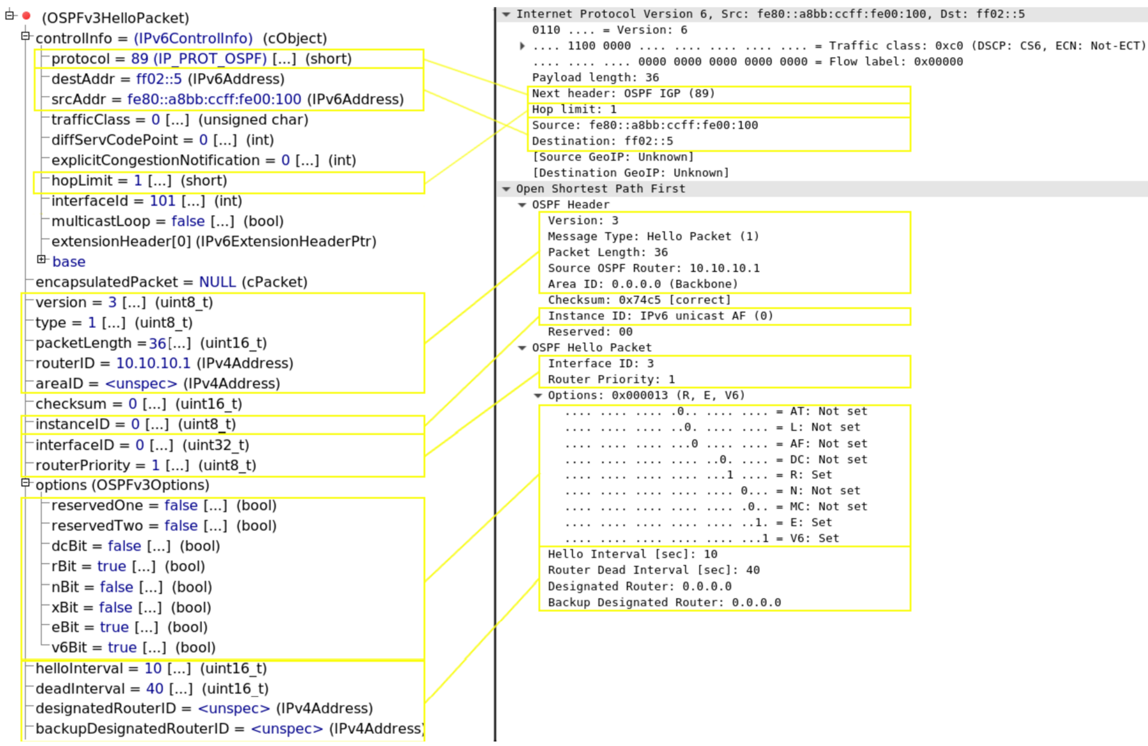

Figure 8.2: Comparison of the Hello Packet Content

The figure 8.2 shows a comparison of an OMNeT++ message on the left and a packet captured using Wireshark on the right. The important parts are highlighted in yellow color. Both the message and the packet are captured on Ethernet 0/0 interface on router R1.

Link-local address is used as the source address, and the hop limit is set to 1. This ensures that neighbors will be discovered on the local network only. The destination address in the Hello Packet is always ff02::5 and all OSPFv3 capable routers have to be prepared to receive them.

The `<unspec>` value in OMNeT++ message means 0.0.0.0 IPv4 address. In the example, the `<unspec>` value is set in area ID field, meaning that this area is the backbone, and in the DR and BDR fields, meaning that this is a beginning of communication and the DR and BDR are not yet elected.

The options field and the hello and dead intervals need to be the same among different neighbors for them to even become adjacent in the first place. The checksum is set to 0 in the OMNeT++ message. In a real network, the checksum has an important role in identifying correct contents of the packet. In OMNeT++ this is not the case since the channel between devices is not prone to create any errors.

8.1.2 The Hello Packet Exchange

Figure 8.3 shows a message traffic in OMNeT++. The output is restricted to R1 communication only. The S0 and S1 in the output are switches connecting the routers in the Area 0 and Area 1 respectively.

The figure 8.4 shows a result of `debug ospfv3 1 hello` command from R1 in EVE-NG. Each Cisco router sends an immediate message to any new neighbor discovered on the network for the first time. This is not described anywhere in RFC 2328 nor RFC 5340. The RFCs only state that two routers become neighbors when they see themselves in the Hello Packet from the neighbor. This ensures that the communication is bidirectional. I have implemented an immediate response in the model, so that is corresponds to the Cisco implementation.

```
0.00000000  R1 --> S0    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::5
0.00000000  R1 --> S1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:110 > ff02::5
0.00000922  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:200 > ff02::5
0.00000922  S1 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:500 > ff02::5
0.00001748  R1 --> S0    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:200
0.00001748  R1 --> S1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:110 > fe80::a8bb:ccff:fe00:500
0.00001834  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:300 > ff02::5
0.00001834  S1 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:600 > ff02::5
0.00002660  R1 --> S0    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
0.00002660  R1 --> S1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:110 > fe80::a8bb:ccff:fe00:600
0.00002746  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
0.00002746  S1 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:500 > fe80::a8bb:ccff:fe00:110
0.00003604  R1 --> S0    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:400
0.00003658  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:200 > fe80::a8bb:ccff:fe00:100
0.00003658  S1 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:600 > fe80::a8bb:ccff:fe00:110
0.00004570  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
0.00005482  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:400 > fe80::a8bb:ccff:fe00:100
10.0000000  R1 --> S0    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::5
10.0000000  R1 --> S1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:110 > ff02::5
10.0000098  S1 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:500 > ff02::5
10.0000101  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:200 > ff02::5
10.0000196  S1 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:600 > ff02::5
10.0000202  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:300 > ff02::5
10.0000303  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
20.0000000  R1 --> S0    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::5
20.0000000  R1 --> S1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:110 > ff02::5
20.0000098  S1 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:500 > ff02::5
20.0000101  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:200 > ff02::5
20.0000196  S1 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:600 > ff02::5
20.0000202  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:300 > ff02::5
20.0000303  S0 --> R1    OSPFv3HelloPacket --- IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
```

Figure 8.3: Hello Packet Exchange in OMNeT++ on Router R1

The timestamps between the two figures are different because each OMNeT++ simulation starts simply in a time zero. The time in the virtual environment on the other hand is

```
12:52:04.419: OSPFv3-1-IPv6 HELLO Et0/0: Send hello to FF02::5 area 0 from FE80::A8BB:CCFF:FE00:100 interface ID 3
12:52:04.421: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.3 area 0 from FE80::A8BB:CCFF:FE00:300 interface ID 3
12:52:04.421: OSPFv3-1-IPv6 HELLO Et0/0: Send Immediate hello to nbr 10.10.10.3, src address FE80::A8BB:CCFF:FE00:300
12:52:04.421: OSPFv3-1-IPv6 HELLO Et0/0: Send hello to FE80::A8BB:CCFF:FE00:300 area 0 from FE80::A8BB:CCFF:FE00:100 interface ID 3
12:52:04.423: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.4 area 0 from FE80::A8BB:CCFF:FE00:400 interface ID 3
12:52:04.423: OSPFv3-1-IPv6 HELLO Et0/0: Send Immediate hello to nbr 10.10.10.4, src address FE80::A8BB:CCFF:FE00:400
12:52:04.423: OSPFv3-1-IPv6 HELLO Et0/0: Send hello to FE80::A8BB:CCFF:FE00:400 area 0 from FE80::A8BB:CCFF:FE00:100 interface ID 3
12:52:04.423: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.3 area 0 from FE80::A8BB:CCFF:FE00:300 interface ID 3
12:52:04.429: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.4 area 0 from FE80::A8BB:CCFF:FE00:400 interface ID 3
12:52:04.658: OSPFv3-1-IPv6 HELLO Et0/1: Send hello to FF02::5 area 1 from FE80::A8BB:CCFF:FE00:110 interface ID 4
12:52:04.671: OSPFv3-1-IPv6 HELLO Et0/1: Rcv hello from 10.10.10.6 area 1 from FE80::A8BB:CCFF:FE00:600 interface ID 3
12:52:04.671: OSPFv3-1-IPv6 HELLO Et0/1: Send Immediate hello to nbr 10.10.10.6, src address FE80::A8BB:CCFF:FE00:600
12:52:04.671: OSPFv3-1-IPv6 HELLO Et0/1: Send hello to FE80::A8BB:CCFF:FE00:600 area 1 from FE80::A8BB:CCFF:FE00:110 interface ID 4
12:52:04.672: OSPFv3-1-IPv6 HELLO Et0/1: Rcv hello from 10.10.10.5 area 1 from FE80::A8BB:CCFF:FE00:500 interface ID 3
12:52:04.672: OSPFv3-1-IPv6 HELLO Et0/1: Send Immediate hello to nbr 10.10.10.5, src address FE80::A8BB:CCFF:FE00:500
12:52:04.672: OSPFv3-1-IPv6 HELLO Et0/1: Send hello to FE80::A8BB:CCFF:FE00:500 area 1 from FE80::A8BB:CCFF:FE00:110 interface ID 4
12:52:04.803: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.2 area 0 from FE80::A8BB:CCFF:FE00:200 interface ID 3
12:52:04.803: OSPFv3-1-IPv6 HELLO Et0/0: Send Immediate hello to nbr 10.10.10.2, src address FE80::A8BB:CCFF:FE00:200
12:52:04.803: OSPFv3-1-IPv6 HELLO Et0/0: Send hello to FE80::A8BB:CCFF:FE00:200 area 0 from FE80::A8BB:CCFF:FE00:100 interface ID 3
12:52:04.854: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.2 area 0 from FE80::A8BB:CCFF:FE00:200 interface ID 3
12:52:13.523: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.3 area 0 from FE80::A8BB:CCFF:FE00:300 interface ID 3
12:52:13.573: OSPFv3-1-IPv6 HELLO Et0/0: Send hello to FF02::5 area 0 from FE80::A8BB:CCFF:FE00:100 interface ID 3
12:52:13.585: OSPFv3-1-IPv6 HELLO Et0/1: Rcv hello from 10.10.10.5 area 1 from FE80::A8BB:CCFF:FE00:500 interface ID 3
12:52:13.690: OSPFv3-1-IPv6 HELLO Et0/1: Rcv hello from 10.10.10.6 area 1 from FE80::A8BB:CCFF:FE00:600 interface ID 3
12:52:13.754: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.2 area 0 from FE80::A8BB:CCFF:FE00:200 interface ID 3
12:52:13.908: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.4 area 0 from FE80::A8BB:CCFF:FE00:400 interface ID 3
12:52:14.495: OSPFv3-1-IPv6 HELLO Et0/1: Send hello to FF02::5 area 1 from FE80::A8BB:CCFF:FE00:110 interface ID 4
12:52:22.802: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.3 area 0 from FE80::A8BB:CCFF:FE00:300 interface ID 3
12:52:22.803: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.2 area 0 from FE80::A8BB:CCFF:FE00:200 interface ID 3
12:52:22.879: OSPFv3-1-IPv6 HELLO Et0/0: Send hello to FF02::5 area 0 from FE80::A8BB:CCFF:FE00:100 interface ID 3
12:52:23.304: OSPFv3-1-IPv6 HELLO Et0/0: Rcv hello from 10.10.10.4 area 0 from FE80::A8BB:CCFF:FE00:400 interface ID 3
12:52:23.511: OSPFv3-1-IPv6 HELLO Et0/1: Rcv hello from 10.10.10.5 area 1 from FE80::A8BB:CCFF:FE00:500 interface ID 3
12:52:23.573: OSPFv3-1-IPv6 HELLO Et0/1: Rcv hello from 10.10.10.6 area 1 from FE80::A8BB:CCFF:FE00:600 interface ID 3
12:52:24.178: OSPFv3-1-IPv6 HELLO Et0/1: Send hello to FF02::5 area 1 from FE80::A8BB:CCFF:FE00:110 interface ID 4
```

Figure 8.4: Hello Packet Exchange in EVE-NG on Router R1

based on the system time of the device. But the most important is that the initial exchange of the first series of Hello Packets takes place at the same time. Other Hello Packets are sent or received every 10 seconds which is the time of the Hello Interval. These packets serve as a keepalive for the neighbors.

Inspecting the IP addresses in both figures, we are able to see that truly only routers on the local network are able to communicate. For instance, there is no packet from router R6 to the router R4. Only routers from Area 0 communicate with R1 on interface Eth 0/0(link-local address fe80::a8bb:ccff:fe00:100) and only routers from Area 1 are able to contact R1 on Eth 0/1(link-local address fe80::a8bb:ccff:fe00:110).

8.2 Neighborship Establishment

The next phase in the OSPFv3 process is directly connected to the Hello Packet exchange. It is the neighborship establishment and DR and BDR election.

Figure 8.5 shows the state of neighbors on router R1. The top part shows the output from OMNeT++. The bottom part is the output from EVE-NG after issuing the `sh ospfv3 1 neighbors` command.

The output is actually from a period after the database exchange. But I am using it in this section to show which router becomes DR and BDR for different areas. During the exchange, the state of the neighbors is either `2WAY` or `LOADING` so there is no information about the election.

The router R1 has a default priority for Area 0 (default priority is 1) and priority set to 10 for Area 1. This results in R1 being a `DROTHER` for Area 0 because the DR election is based on the highest router-id value. For Area 0 the DR is router R4, and BDR is router R3. The relationship with router R2 stays in `2WAY` because R1 and R2 do not exchange any LSAs directly.

R1 is the DR in Area 1. Router R6 is the BDR because it has higher router ID than R5.

```
OSPFv3 1 address-family IPv6 (router-id 10.10.10.1)

Neighbor ID    Pri    State           Dead Time    Interface ID    Interface
10.10.10.2     1      2WAY/DROTHER    35           0               eth0
10.10.10.3     1      FULL/BDR        35           0               eth0
10.10.10.4     1      FULL/DR         35           0               eth0
10.10.10.5     1      FULL/DROTHER    35           0               eth1
10.10.10.6     1      FULL/BDR        35           0               eth1
```

```
          OSPFv3 1 address-family ipv6 (router-id 10.10.10.1)

Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
10.10.10.2        1   2WAY/DROTHER    00:00:35    3               Ethernet0/0
10.10.10.3        1   FULL/BDR        00:00:38    3               Ethernet0/0
10.10.10.4        1   FULL/DR         00:00:37    3               Ethernet0/0
10.10.10.5        1   FULL/DROTHER    00:00:39    3               Ethernet0/1
10.10.10.6        1   FULL/BDR        00:00:39    3               Ethernet0/1
```

Figure 8.5: Neighbors' Relationship Comparison on Router R1

```
Ethernet0/0 is up, line protocol is up
  Link Local Address FE80::A8BB:CCFF:FE00:100, Interface ID 3
  Area 0, Process ID 1, Instance ID 0, Router ID 10.10.10.1
  Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State DROTHER, Priority 1
  Designated Router (ID) 10.10.10.4, local address FE80::A8BB:CCFF:FE00:400
  Backup Designated router (ID) 10.10.10.3, local address FE80::A8BB:CCFF:FE00:300
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:06
  Graceful restart helper support enabled
  Index 1/1/1, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 0, maximum is 3
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 3, Adjacent neighbor count is 2
    Adjacent with neighbor 10.10.10.3  (Backup Designated Router)
    Adjacent with neighbor 10.10.10.4  (Designated Router)
  Suppress hello for 0 neighbor(s)
```

```
Ethernet0/1 is up, line protocol is up
  Link Local Address FE80::A8BB:CCFF:FE00:110, Interface ID 4
  Area 1, Process ID 1, Instance ID 0, Router ID 10.10.10.1
  Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State DR, Priority 10
  Designated Router (ID) 10.10.10.1, local address FE80::A8BB:CCFF:FE00:110
  Backup Designated router (ID) 10.10.10.6, local address FE80::A8BB:CCFF:FE00:600
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:05
  Graceful restart helper support enabled
  Index 1/1/2, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 2, maximum is 9
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 2, Adjacent neighbor count is 2
    Adjacent with neighbor 10.10.10.5
    Adjacent with neighbor 10.10.10.6  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
```

Figure 8.6: Interfaces Setting Output on Router R1 in EVE-NG

Figures 8.6 and 8.7 show a comparison of detailed information about each interface which is turned on for OSPFv3. The first figure shows `sh ospfv3 1 interfaces` command output on R1 from EVE-NG. In the second, there is a detailed output from each `OSPFv3Interface` in OMNeT++. Except for a few extra lines in EVE-NG output the results are identical.

We can see that R1 is adjacent with both routers R5 and R6 in Area 1. The DR is always fully adjacent with all other routers in an area. In Area0, on the other hand, R1 is adjacent with R3 and R4 only. R2 is `DROTHER` and as such does not create a full relationship with R1.

```
Interface eth0
Link Local Address fe80::a8bb:ccff:fe00:100, Interface ID 101
Area 0, Process ID 1, Instance ID 0, Router ID 10.10.10.1
Network Type BROADCAST, Cost: 0
Transmit Delay is 1 sec, State DROther, Priority 1
Designated Router (ID) 10.10.10.4, local address fe80::a8bb:ccff:fe00:400
Backup Designated router (ID) 10.10.10.3, local address fe80::a8bb:ccff:fe00:300
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
        Hello due in 5
Neighbor Count is 3, Adjacent neighbor count is 2
Adjacent with neighbor 10.10.10.3(Backup Designated Router)
Adjacent with neighbor 10.10.10.4(Designated Router)
Suppress Hello for 0 neighbor(s)
```

```
Interface eth1
Link Local Address fe80::a8bb:ccff:fe00:110, Interface ID 102
Area 1, Process ID 1, Instance ID 0, Router ID 10.10.10.1
Network Type BROADCAST, Cost: 0
Transmit Delay is 1 sec, State DR, Priority 10
Designated Router (ID) 10.10.10.1, local address fe80::a8bb:ccff:fe00:110
Backup Designated router (ID) 10.10.10.6, local address fe80::a8bb:ccff:fe00:600
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
        Hello due in 5
Neighbor Count is 2, Adjacent neighbor count is 2
Adjacent with neighbor 10.10.10.6(Backup Designated Router)
Suppress Hello for 0 neighbor(s)
```

Figure 8.7: Interfaces Setting Output on Router R1 in OMNeT++

8.3 Database Exchange

After adjacencies between neighbors are established, and DR and BDR are elected, the databases are exchanged. The figures 8.8 and 8.9 show how the exchange is carried out in EVE-NG and OMNeT++ respectively. Both figures show exchange in Area 0 on ethernet 0/0 interface on router R1. The EVE-NG figure is a sequence of packets captured using Wireshark.

The most distinct difference between both figures is the number of packets. There are more *LS Update* packets in EVE-NG and much less LS Acknowledge packets in OMNeT++.

In OMNeT++ there is one *LS Update* packet sent as a response to *LS Request* packet. But when a DR is flooding received updates, it sends one *LS Update* per received LSA. This does not violate any process described in the standard. The LS Acknowledgement packets in OMNeT++ are sent with a delay of one second. The standard clearly states that there may be multiple LSAs acknowledged in a single LS Acknowledgement packet. The one second is chosen so that all of the LSAs are already exchanged and the RxmtInterval defined in RFC 2328 does not expire in that time.

More important than the number of packets exchanged are the source and destination addresses being used. All DB Description and *LS Request* packets are exchanged between R1 and R4 or R3. *LS Update* and LS Acknowledgement packets are sent either directly between routers or to an appropriate multicast address. The R1 as a DROther is using ff02::6 as an address used by DR and BDR only. R4 and R3, on the other hand, use ff02::5 to deliver updates. This multicast address should be used by every OSPFv3 capable router in the area.

No.	Time	Source	Destination	Protocol	Length	Info
60	34.256044	fe80::a8bb:ccff:fe00:400	fe80::a8bb:ccff:fe00:100	OSPF	82	DB Description
65	37.551654	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:400	OSPF	82	DB Description
67	38.772360	fe80::a8bb:ccff:fe00:400	fe80::a8bb:ccff:fe00:100	OSPF	82	DB Description
68	38.772577	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:400	OSPF	162	DB Description
69	38.772950	fe80::a8bb:ccff:fe00:400	fe80::a8bb:ccff:fe00:100	OSPF	162	DB Description
70	38.773104	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:400	OSPF	118	LS Request
71	38.773119	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:400	OSPF	82	DB Description
72	38.773443	fe80::a8bb:ccff:fe00:400	fe80::a8bb:ccff:fe00:100	OSPF	234	LS Update
73	38.773458	fe80::a8bb:ccff:fe00:400	fe80::a8bb:ccff:fe00:100	OSPF	118	LS Request
74	38.773664	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:400	OSPF	234	LS Update
75	38.898317	fe80::a8bb:ccff:fe00:400	ff02::5	OSPF	234	LS Update
76	39.115795	fe80::a8bb:ccff:fe00:400	ff02::5	OSPF	234	LS Update
78	39.320888	fe80::a8bb:ccff:fe00:400	ff02::5	OSPF	302	LS Update
80	39.439178	fe80::a8bb:ccff:fe00:400	ff02::5	OSPF	146	LS Update
82	39.655982	fe80::a8bb:ccff:fe00:400	ff02::5	OSPF	146	LS Update
84	41.278187	fe80::a8bb:ccff:fe00:100	ff02::6	OSPF	390	LS Acknowledge
87	41.581686	fe80::a8bb:ccff:fe00:400	ff02::5	OSPF	190	LS Acknowledge
90	42.568541	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:300	OSPF	82	DB Description
91	42.568700	fe80::a8bb:ccff:fe00:300	fe80::a8bb:ccff:fe00:100	OSPF	82	DB Description
92	42.568872	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:300	OSPF	402	DB Description
94	42.569434	fe80::a8bb:ccff:fe00:300	fe80::a8bb:ccff:fe00:100	OSPF	362	DB Description
95	42.569564	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:300	OSPF	94	LS Request
96	42.569600	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:300	OSPF	82	DB Description
97	42.570083	fe80::a8bb:ccff:fe00:300	fe80::a8bb:ccff:fe00:100	OSPF	154	LS Update
99	42.603246	fe80::a8bb:ccff:fe00:100	ff02::6	OSPF	154	LS Update
101	42.603825	fe80::a8bb:ccff:fe00:300	fe80::a8bb:ccff:fe00:100	OSPF	110	LS Acknowledge
115	45.074123	fe80::a8bb:ccff:fe00:100	ff02::6	OSPF	150	LS Acknowledge
122	47.210468	fe80::a8bb:ccff:fe00:100	fe80::a8bb:ccff:fe00:400	OSPF	154	LS Update
123	47.211008	fe80::a8bb:ccff:fe00:400	fe80::a8bb:ccff:fe00:100	OSPF	110	LS Acknowledge
140	51.182283	fe80::a8bb:ccff:fe00:100	ff02::6	OSPF	110	LS Acknowledge

Figure 8.8: Database Exchange on Interface Ethernet 0/0 on R1 in EVE-NG

The exchange of packets in a real network will hardly ever be the same as in OMNeT++. Any delay in the network may change the order of packets. For instance, the DR may have a different database in the real network when it received *LS Request* from R1 because it has already received *LS Update* from R2. Much more important is whether the database is complete at the end of this process. This is described in section 8.4

8.4 Convergence

The convergence indicates a state when all the databases of all routers in an area are the same. This is a starting point for Dijkstra algorithm and route calculation.

```
40.0000000  R1 --> S0       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:400
40.0000385  S0 --> R1       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > fe80::a8bb:ccff:fe00:100
40.00005044 R1 --> S0       inet::OSPFv3DatabaseDescription:162 bytes   inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:400
40.00011924 R1 --> S0       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
40.0001201  S0 --> R1       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > fe80::a8bb:ccff:fe00:100
40.0002097  S0 --> R1       inet::OSPFv3DatabaseDescription:182 bytes   inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > fe80::a8bb:ccff:fe00:100
40.00022964 R1 --> S0       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:400
40.0002305  S0 --> R1       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.00024244 R1 --> S0       inet::OSPFv3LinkStateRequest:130 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:400
40.0002433  S0 --> R1       inet::OSPFv3LinkStateRequest:118 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > fe80::a8bb:ccff:fe00:100
40.00025908 R1 --> S0       inet::OSPFv3DatabaseDescription:162 bytes   inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
40.00027828 R1 --> S0       inet::OSPFv3LSUpdate:370 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::6
40.00032906 S0 --> R1       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.00037034 S0 --> R1       inet::OSPFv3LinkStateRequest:118 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > fe80::a8bb:ccff:fe00:100
40.00038516 R1 --> S0       inet::OSPFv3LSUpdate:370 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::6
40.00041802 S0 --> R1       inet::OSPFv3LSUpdate:410 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.0005905  S0 --> R1       inet::OSPFv3LSUpdate:146 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00060842 S0 --> R1       inet::OSPFv3LSUpdate:122 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00062442 S0 --> R1       inet::OSPFv3DatabaseDescription:162 bytes   inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.00064276 R1 --> S0       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
40.00064362 S0 --> R1       inet::OSPFv3LSUpdate:198 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00065556 R1 --> S0       inet::OSPFv3LinkStateRequest:118 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
40.0006657  S0 --> R1       inet::OSPFv3LinkStateRequest:118 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.00068052 R1 --> S0       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
40.00068138 S0 --> R1       inet::OSPFv3LSUpdate:126 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00073258 S0 --> R1       inet::OSPFv3LSUpdate:410 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.0007873  S0 --> R1       inet::OSPFv3LSUpdate:410 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.0007926  R1 --> S0       inet::OSPFv3LSUpdate:122 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::5
40.00082634 S0 --> R1       inet::OSPFv3LSUpdate:146 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00085706 S0 --> R1       inet::OSPFv3LSUpdate:122 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00087306 S0 --> R1       inet::OSPFv3LSUpdate:198 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00089514 S0 --> R1       inet::OSPFv3LSUpdate:126 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00091146 S0 --> R1       inet::OSPFv3LSUpdate:146 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00092938 S0 --> R1       inet::OSPFv3LSUpdate:122 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00094538 S0 --> R1       inet::OSPFv3LSUpdate:198 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00096746 S0 --> R1       inet::OSPFv3LSUpdate:126 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00098378 S0 --> R1       inet::OSPFv3LSUpdate:126 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00107978 S0 --> R1       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.00109172 R1 --> S0       inet::OSPFv3DatabaseDescription:382 bytes   inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
40.00109258 S0 --> R1       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.00128202 S0 --> R1       inet::OSPFv3DatabaseDescription:362 bytes   inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.00131636 R1 --> S0       inet::OSPFv3DatabaseDescription:82 bytes    inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
40.00132916 R1 --> S0       inet::OSPFv3LinkStateRequest:190 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > fe80::a8bb:ccff:fe00:300
40.00135786 S0 --> R1       inet::OSPFv3LinkStateRequest:202 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.0013794  R1 --> S0       inet::OSPFv3LSUpdate:902 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::6
40.0015985  S0 --> R1       inet::OSPFv3LSUpdate:854 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00171594 S0 --> R1       inet::OSPFv3LSUpdate:126 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00182346 S0 --> R1       inet::OSPFv3LinkStateRequest:202 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > fe80::a8bb:ccff:fe00:100
40.001845   R1 --> S0       inet::OSPFv3LSUpdate:902 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::6
40.00192042 S0 --> R1       inet::OSPFv3LSUpdate:998 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00204266 S0 --> R1       inet::OSPFv3LSUpdate:126 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.00210186 S0 --> R1       inet::OSPFv3LSUpdate:122 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
40.0021873  S0 --> R1       inet::OSPFv3LSUpdate:126 bytes      inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
41.00040178 S0 --> R1       inet::OSPFv3LSAck:410 bytes     inet::IPv6Datagram: fe80::a8bb:ccff:fe00:400 > ff02::5
41.0004562  R1 --> S0       inet::OSPFv3LSAck:470 bytes     inet::IPv6Datagram: fe80::a8bb:ccff:fe00:100 > ff02::6
41.00054686 S0 --> R1       inet::OSPFv3LSAck:490 bytes     inet::IPv6Datagram: fe80::a8bb:ccff:fe00:300 > ff02::5
```

Figure 8.9: Database Exchange on Interface Ethernet 0/0 on R1 in OMNeT++

Figures 8.10 and 8.11 show the state of the database on router R1 after it starts up. This is the state before any DR or BDR are elected. The router has only information about its surroundings. Only the Router LSA, Link LSA and Intra-Area-Prefix LSA are present.

Since the R1 is an ABR, it has to create Inter-Area-Prefix LSAs and distribute them between different areas. It creates Intra-Area-Prefix LSA for address prefix `2001:db8:1::/64` from Area 1 and places the LSA in Area 0 database. The same happens for prefix `2001:db8:a::/64` from Area 0. R1 is also responsible for creating a LSA with a default route for any stub area. Area 1 is a stub area, R1 creates Intra-Area-Prefix LSA with a default prefix `::/0`. There are no Network LSAs in the database, because these are created on DR after the election.

Figures 8.12 and 8.13 show a state of the database on router R1 after the databases between routers have been exchanged. Each area has only the Router LSAs of routers belonging to the area. There is one Network LSA for each area originated by the DR. The prefix from Area 1 is distributed in Area 0 and all the prefixes from Areas 2-4 are redistributed into Area 1. In this state, the router is ready to start the Dijkstra algorithm.

OSPFv3 1 address-family ipv6 (router-id 10.10.10.1)

Router Link States (Area 0.0.0.0)

ADV Router	Age	Seq#	Fragment ID	Link count	Bits
10.10.10.1	0	0x80000001	0	0	B

Inter Area Prefix Link States (Area 0.0.0.0)

ADV Router	Age	Seq#	Prefix
10.10.10.1	0	0x80000001	2001:db8:1::/64

Link (Type-8) Link States (Area 0.0.0.0)

ADV Router	Age	Seq#	Link State ID	Interface
10.10.10.1	0	0x80000001	0.0.0.0	eth0

Intra Area Prefix Link States (Area0.0.0.0)

ADV Router	Age	Seq#	Link ID	Ref-lstype	Ref-LSID
10.10.10.1	0	0x80000001	0.0.0.0	0x2001	0.0.0.0

OSPFv3 1 address-family ipv6 (router-id 10.10.10.1)

Router Link States (Area 0.0.0.1)

ADV Router	Age	Seq#	Fragment ID	Link count	Bits
10.10.10.1	0	0x80000001	0	0	B

Inter Area Prefix Link States (Area 0.0.0.1)

ADV Router	Age	Seq#	Prefix
10.10.10.1	0	0x80000001	2001:db8:a::/64
10.10.10.1	0	0x80000002	::/0

Link (Type-8) Link States (Area 0.0.0.1)

ADV Router	Age	Seq#	Link State ID	Interface
10.10.10.1	0	0x80000001	0.0.0.1	eth1

Intra Area Prefix Link States (Area0.0.0.1)

ADV Router	Age	Seq#	Link ID	Ref-lstype	Ref-LSID
10.10.10.1	0	0x80000001	0.0.0.0	0x2001	0.0.0.0

Figure 8.10: Initial State of LSA Database on R1 in OMNeT++

```
        OSPFv3 1 address-family ipv6 (router-id 10.10.10.1)

                Router Link States (Area 0)

ADV Router       Age         Seq#        Fragment ID  Link count  Bits
 10.10.10.1      21          0x80000001  0            0           B

                Inter Area Prefix Link States (Area 0)

ADV Router       Age         Seq#        Prefix
 10.10.10.1      12          0x80000001  2001:DB8:1::/64

                Link (Type-8) Link States (Area 0)

ADV Router       Age         Seq#        Link ID    Interface
 10.10.10.1      17          0x80000002  3          Et0/0

                Intra Area Prefix Link States (Area 0)

ADV Router       Age         Seq#        Link ID    Ref-lstype  Ref-LSID
 10.10.10.1      17          0x80000001  0          0x2001      0

                Router Link States (Area 1)

ADV Router       Age         Seq#        Fragment ID  Link count  Bits
 10.10.10.1      21          0x80000001  0            0           B

                Inter Area Prefix Link States (Area 1)

ADV Router       Age         Seq#        Prefix
 10.10.10.1      22          0x80000001  ::/0
 10.10.10.1      12          0x80000001  2001:DB8:A::/64

                Link (Type-8) Link States (Area 1)

ADV Router       Age         Seq#        Link ID    Interface
 10.10.10.1      17          0x80000002  4          Et0/1

                Intra Area Prefix Link States (Area 1)

ADV Router       Age         Seq#        Link ID    Ref-lstype  Ref-LSID
 10.10.10.1      17          0x80000001  0          0x2001      0
```

Figure 8.11: Initial State of LSA Database on R1 in EVE-NG

OSPFv3 1 address-family ipv6 (router-id 10.10.10.1)

Router Link States (Area 0.0.0.0)

ADV Router	Age	Seq#	Fragment ID	Link count	Bits
10.10.10.1	5	0x80000002	0	1	B
10.10.10.4	5	0x80000002	0	1	B
10.10.10.2	5	0x80000002	0	1	B
10.10.10.3	5	0x80000002	0	1	B

Net Link States (Area 0.0.0.0)

ADV Router	Age	Seq#	Link State ID	Rtr count
10.10.10.4	5	0x80000001	0.0.0.0	4

Inter Area Prefix Link States (Area 0.0.0.0)

ADV Router	Age	Seq#	Prefix
10.10.10.1	45	0x80000001	2001:db8:1::/64
10.10.10.4	45	0x80000001	2001:db8:4::/64
10.10.10.2	45	0x80000001	2001:db8:2::/64
10.10.10.3	45	0x80000001	2001:db8:3::/64

Link (Type-8) Link States (Area 0.0.0.0)

ADV Router	Age	Seq#	Link State ID	Interface
10.10.10.1	45	0x80000001	0.0.0.0	eth0
10.10.10.4	45	0x80000001	0.0.0.0	eth0
10.10.10.2	45	0x80000001	0.0.0.0	eth0
10.10.10.3	45	0x80000001	0.0.0.0	eth0

Intra Area Prefix Link States (Area0.0.0.0)

ADV Router	Age	Seq#	Link ID	Ref-lstype	Ref-LSID
10.10.10.4	5	0x80000001	0.0.0.0	0x2002	0.0.0.0

OSPFv3 1 address-family ipv6 (router-id 10.10.10.1)

Router Link States (Area 0.0.0.1)

ADV Router	Age	Seq#	Fragment ID	Link count	Bits
10.10.10.1	5	0x80000002	0	0	B
10.10.10.5	5	0x80000002	0	0	None
10.10.10.6	5	0x80000002	0	0	None

Net Link States (Area 0.0.0.1)

ADV Router	Age	Seq#	Link State ID	Rtr count
10.10.10.1	5	0x80000001	0.0.0.1	3

Inter Area Prefix Link States (Area 0.0.0.1)

ADV Router	Age	Seq#	Prefix
10.10.10.1	45	0x80000001	2001:db8:a::/64
10.10.10.1	45	0x80000002	::/0
10.10.10.1	45	0x80000003	2001:db8:4::/64
10.10.10.1	45	0x80000004	2001:db8:2::/64
10.10.10.1	45	0x80000005	2001:db8:3::/64

Link (Type-8) Link States (Area 0.0.0.1)

ADV Router	Age	Seq#	Link State ID	Interface
10.10.10.1	45	0x80000001	0.0.0.1	eth1
10.10.10.5	45	0x80000001	0.0.0.0	eth1
10.10.10.6	45	0x80000001	0.0.0.0	eth1

Intra Area Prefix Link States (Area0.0.0.1)

ADV Router	Age	Seq#	Link ID	Ref-lstype	Ref-LSID
10.10.10.1	5	0x80000001	0.0.0.0	0x2002	0.0.0.1

Figure 8.12: A Complete LSA Database on R1 in OMNeT++

```
          OSPFv3 1 address-family ipv6 (router-id 10.10.10.1)
                Router Link States (Area 0)
ADV Router       Age         Seq#        Fragment ID  Link count  Bits
 10.10.10.1      401         0x80000002  0            1           B
 10.10.10.2      403         0x80000002  0            1           B
 10.10.10.3      403         0x80000002  0            1           B
 10.10.10.4      402         0x80000002  0            1           B
                Net Link States (Area 0)
ADV Router       Age         Seq#        Link ID    Rtr count
 10.10.10.4      402         0x80000001  3          4
                Inter Area Prefix Link States (Area 0)
ADV Router       Age         Seq#        Prefix
 10.10.10.1      433         0x80000001  2001:DB8:1::/64
 10.10.10.2      435         0x80000001  2001:DB8:2::/64
 10.10.10.3      436         0x80000001  2001:DB8:3::/64
 10.10.10.4      436         0x80000001  2001:DB8:4::/64
                Link (Type-8) Link States (Area 0)
ADV Router       Age         Seq#        Link ID    Interface
 10.10.10.1      438         0x80000002  3          Et0/0
 10.10.10.2      440         0x80000002  3          Et0/0
 10.10.10.3      441         0x80000002  3          Et0/0
 10.10.10.4      441         0x80000002  3          Et0/0
                Intra Area Prefix Link States (Area 0)
ADV Router       Age         Seq#        Link ID    Ref-lstype  Ref-LSID
 10.10.10.4      402         0x80000001  3072       0x2002      3
                Router Link States (Area 1)
ADV Router       Age         Seq#        Fragment ID  Link count  Bits
 10.10.10.1      402         0x80000002  0            1           B
 10.10.10.5      403         0x80000002  0            1           None
 10.10.10.6      399         0x80000002  0            1           None
                Net Link States (Area 1)
ADV Router       Age         Seq#        Link ID    Rtr count
 10.10.10.1      397         0x80000002  4          3
                Inter Area Prefix Link States (Area 1)
ADV Router       Age         Seq#        Prefix
 10.10.10.1      443         0x80000001  ::/0
 10.10.10.1      433         0x80000001  2001:DB8:A::/64
 10.10.10.1      398         0x80000001  2001:DB8:4::/64
 10.10.10.1      398         0x80000001  2001:DB8:3::/64
 10.10.10.1      398         0x80000001  2001:DB8:2::/64
                Link (Type-8) Link States (Area 1)
ADV Router       Age         Seq#        Link ID    Interface
 10.10.10.1      438         0x80000002  4          Et0/1
 10.10.10.5      443         0x80000002  3          Et0/1
 10.10.10.6      444         0x80000002  3          Et0/1
                Intra Area Prefix Link States (Area 1)
ADV Router       Age         Seq#        Link ID    Ref-lstype  Ref-LSID
 10.10.10.1      402         0x80000001  4096       0x2002      4
```

Figure 8.13: A Complete LSA Database on R1 in EVE-NG

8.5 Failover State

The failover state is simulated by disconnecting interface ethernet0 on R4. The `ScenarioManager` module is used to conduct this experiment. This module parses the `scenario.xml` file present in the example directory. Based on parameters it is capable of dynamically changing certain aspects of the simulation(like disconnecting an interface).

At time `t=80` the ethernet 0/0 interface on R4 and ethernet 0/3 on switch connecting routers in Area 0 are disconnected. The result is not obvious immediately because there is a Dead Interval Timer running for each neighbor. After 40 seconds, when the Dead Timer expires, each of the routers in Area 0 removes the router R4 from its neighbors. Since R4 was the DR for this area, the new DR and BDR need to be elected.

R3 as the BDR for Area 0 is immediately elected as the new DR. R2 has the highest router ID after R3 so this is the new BDR.

R3 as the newly elected DR now creates new Network LSA and Inter-Area-Prefix LSA with the Area 0 network prefix and floods them throughout the area. Since all the routers in the area share the same database, there is no need to exchange the whole databases now. Only the neighbors' relationships are changed. R1 and R2 now become fully adjacent.

Chapter 9

Conclusion

During my work on this thesis, my aim was to create a model of widely used link-state routing protocol OSPFv3. Since the protocol is quite complicated and comprehensive my task was to create the model without the SPF Tree calculation. Even without the tree a huge amount of work went into this project. But of course a routing protocol which is incapable of routing is not complete. Hence, my journey does not end here.

There are three main sources of information that form the foundation blocks of this project. The first source is, of course, the standard. Both `RFC 2328` and `RFC 5340` describe the protocol in a very detailed way. Even though every real implementation of the protocol should follow the standard as much as possible, the real life usage might differ.

This is when the second source comes into play. It is the vendor specification and behaviour of the protocol as it is used in routing devices. Capturing traffic or watching protocol events on a router give more insight into what is actually happening on the network.

The last source is the implementation of older OSPFv2 protocol present in INET framework. A large number of parts are similar in the new version of the protocol but with the new features and changes basically the whole model had to be changed. All the three sources combined gave a great foundation for this work.

I would assess my effort as a success. As was shown in the chapter 8, the model's behaviour is truly comparable to a real routing device.

There is one last step ahead of me. I have consulted future development of this project with my supervisor, and I will add the SPF tree calculation to make this module complete. We will present the project at one of the official OMNeT++ Community summits afterwards. The project should become a permanent part of INET one day.

Bibliography

[1] OSPFv3 Commands. [Online]. Accessed: 2016-03-02.
Retrieved from: `http://www.cisco.com/c/en/us/td/docs/routers/xr12000/software/xr12k_r4-2/routing/command/reference/b_routing_cr42xr12k/b_routing_cr42xr12k_chapter_0101.pdf`

[2] Automated Network Simulation and Analysis. [Online]. 2012. Accessed: 2015-09-20.
Retrieved from: `https://ansa.omnetpp.org/`

[3] Configuring OSPFv3. [Online]. 2015. Accessed: 2016-02-10.
Retrieved from: `http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/configuration/15-mt/iro-15-mt-book.html`

[4] INET Framework. [Online]. 2016. Accessed: 2015-09-20.
Retrieved from: `https://inet.omnetpp.org/`

[5] The Emulated Virtual Environment - Next Generation. [Online].. 2017. Accessed: 2017-03-25.
Retrieved from: `http://www.eve-ng.net/index.php/documentation`

[6] Coltrun, R.; Fergusson, D.; Moy, J.; et al.: OSPF for IPv6. RFC. July 2008. updated by RFCs 6845, 6860, 7503.
Retrieved from: `https://tools.ietf.org/html/rfc5340`

[7] Deering, S. E.; Hinden, R. M.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460. December 1998. updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112.
Retrieved from: `https://tools.ietf.org/html/rfc2460`

[8] Moy, J.: OSPF Version 2. RFC 2328. April 1998. updated by RFCs 5709, 6549, 6845, 6860, 7474.
Retrieved from: `https://tools.ietf.org/html/rfc2328`

[9] Murphy, P.: The OSPF Not-So-Stubby Area (NSSA) Option. RFC 3101. January 2003.
Retrieved from: `https://tools.ietf.org/html/rfc3101`

[10] Varga, A.: OMNeT++ Simulation Manual. [Online]. 2015. Accessed: 2015-09-20.
Retrieved from: `https://omnetpp.org/doc/omnetpp/manual/`

[11] Černý, B. M.: *Modelování IPv6 v prostředí OMNeT++*. Master's Thesis. 2011. Accessed: 2016-02-10.

Appendices

Appendix A

Enclosed CD Content

/xrupri00.pdf	Electronic version of the master's thesis in PDF.
/readme.txt	Manual describing how to get this project running.
/install/*	Files needed to install OMNeT++.
/tex/*	Source text of the master's thesis in .tex format.
/src/*	Source files of this project.

Figure A.1: Content of the Enclosed CD

Appendix B

Neighbor State Machine

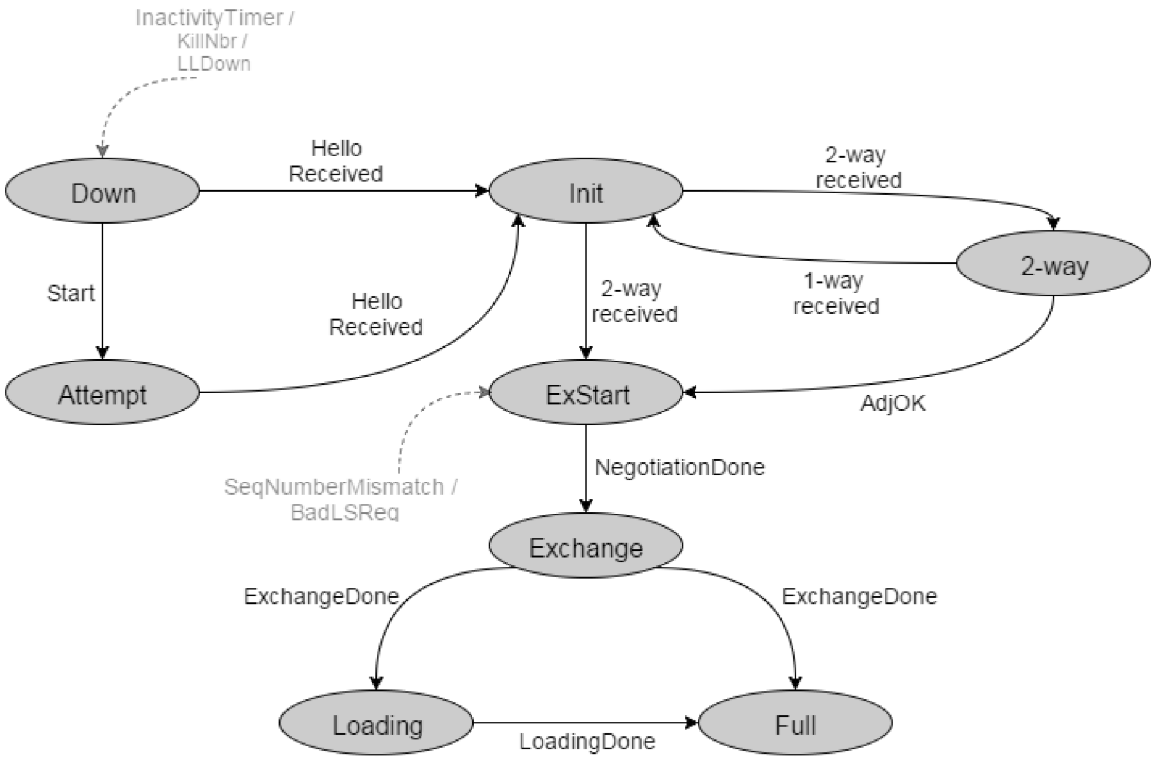

Figure B.1: Neghbor State Machine

Neighbor states reflect the adjacency progress with a neighbor on an interface. States Down, Attempt, Init, 2-way and ExStart are established on the basis of receiving and sending Hello packets. The other states represent the LSDB exchange process. The first part describes each state, the second part describes some of the transitions between states.

Down - initial neighbor state. No information about any neighbor has been received on this interface yet.

Attempt - the interface is sending Hello packets on this interface. This state is only valid on NBMA networks.

Init - Hello packet has recently been received on this interface. All neighbors in this state are listed in the Hello packet originated on the router but the router has not seen itself in Hello packets from neighbors. No bidirectional communication has been established yet.

2-way - bidirectional communication has been established. The DR and BDR are elected in this state or greater.

ExStart - adjacency is beeing created and negotiated in this state. The master and initial DD sequence number are chosen to start the database exchange process.

Exchange - routers exchange their LSDBs in this state. The process is described in detail in section 3.5.

Loading - LSR packets are sent to neighbor because more recent LSAs have been discovered during the exchange process.

Full - in this state, two neighbors are fully adjacent. Their LSDBs are synchronised and Hello packet is used to indicate that they are up and running.

Transitions between states in figure B.1 have names that reflect events causing state changes. Lines between states indicate direct transition between states. Dashed lines indicate events that may happen in more than one states but the result is a transition to a lower state. This usually indicates some error between neighbors has occured.

Start - Hello packets should be sent to the neighbor. It has significance only for NBMA networks.

Hello Received - a Hello packet has been received.

2-way Received - router is aware of its neighbor, because it has seen itself in Hello packet. Bidirectional communication can be establihed.

1-way Received - router received Hello packet but cannot see itself in it.

AdjOK - a decision has to be made wheter to establish an adhjacency with the neighbor.

Negotiation Done - initial exchange information has been established. This indicates that the exchange of databases may begin.

Exchange Done - LSDB has been successfully exchanged. Router is now aware of any LSAs which are out-of-date and may ask for them by sending LSRs.

Loading Done - all out-of-date LSAs have been updated. Both routers now have the same LSAs.

Inactivity Timer - no Hello packets have been received from the neighbor recently. Firing this timer always causes transition from any state to Down state.

KillNbr - all communication with the neighbor is impossible. This causes the transition from any state to the Down state.

LLDown - lower layer protocol indicates that the neighbor is unreachable. This causes the transition from any state to the Down state.

SeqNumberMismatch - wrong DD sequence number or some options mismatch in a DD packet is received. This causes the transition from Exchange state or greater to the ExStart state.

BadLSReq - LSR has been received for an LSA not present in the LSDB. This causes the transition from Exchange state or greater to the ExStart state.

Appendix C

Interface State Machine

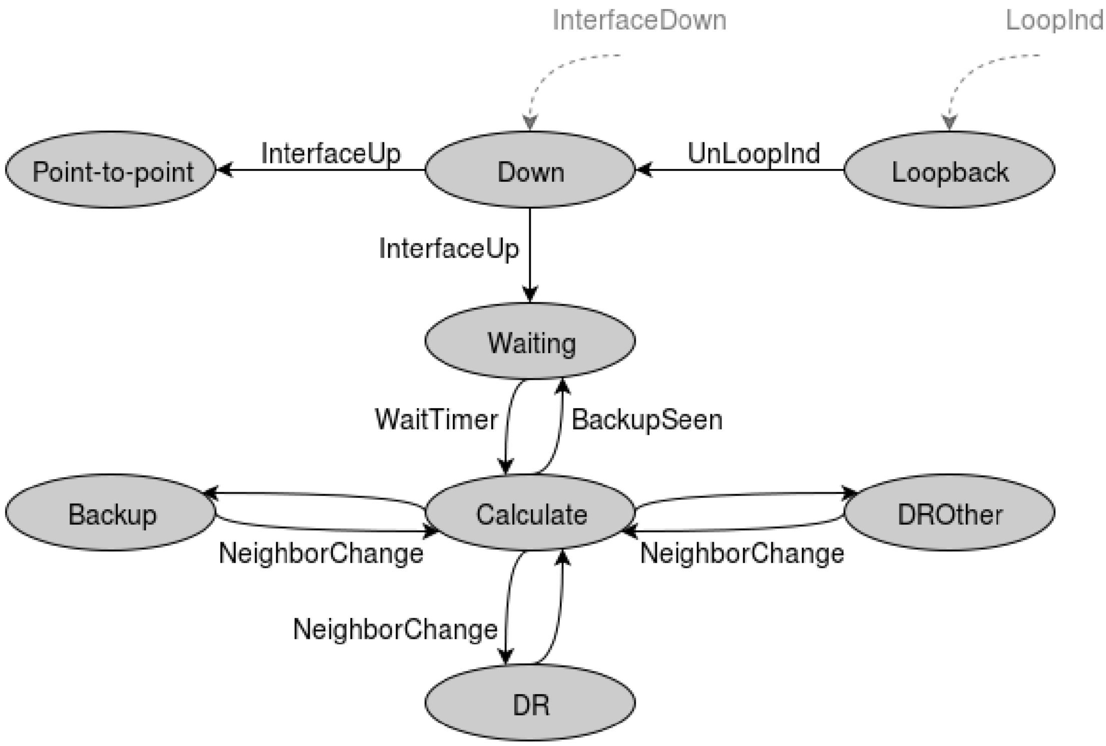

Figure C.1: Interface State Machine

Interface data structure holds information about an interface participating in OSPF process. It contains information about authentication, timers, IP address and mask, DR and BDR, cost and some others. The state of interface reflect the functional level of an interface. The state machine in figure C.1 demonstrates transitions between states caused by OSPF or other configuration changes. The states are described as follows:

Loopback - the interface is configured as loopback interface.

Down - initial interface state. No traffic is sent or received, no adjacencies are established.

Waiting - the router monitors Hello packets and it is trying to determine the DR and BDR. No ellection is held until the router leaves the Waiting state.

Point-to-point - the interface is connected to either physical point-to-point interface or a virtual link. An attempt to establish neighborship will be made by sending Hello packets.

Calculate - only auxiliary state to determine whether the router becomes DR, BDR or a DROther.

DROther - the interface is on a network segment, where the DR and BDR are elected, but this router is neither.

Backup - this state indicates, that the router is the BDR on this network segment. It will be promoted to DR in case the DR is down.

DR - this state indicates, that the router has been elected as the DR.

The transitions between states indicate events that cause a change in interface status. Lines between states indicate direct transition between states. Dashed lines indicate events usually caused by lower level protocols or by issuing some administration command.

InterfaceUp - network interface is operational. Transition to another state depends on the type of network in which the interface operates.

WaitTimer - the wait timer has expired. This indicates, that the DR should be elected.

BackupSeen - a Hello packet is received, indicating the existence or non-existence of BDR. This event signals the router, that the Waiting state is over.

NeighborChange - a change occured in the network and the DR or BDR needs to be recalculated.

LoopInd - interface has changed to a loopback interface. This event causes the interface to transit from any state to Loopback state.

UnLoopInd - interface is no longer a loopback interface.

InterfaceDown - indication from lower level protocols, that the interface is not working. This event causes the interface to transit from any state to Down state.

Appendix D

OSPFv3 Commands

D.1 Global Configuration Mode

ip ospf name-lookup - Routers' IDs are in the form of IP addresses. This command enables the DNS lookup, therefore routers are displayed by names rather than their IDs.

router ospf - This command allows user to enter the Router Configuration Mode(D.2). In this mode it is possible to configure the OSPF process. The process needs to be identified by its pid.

D.2 Router Configuration Mode

area authentication - Allows user to enable authentication on a per-area basis. Two authentication methods are available. The first is a simple password, the second is MD5 authentication.

area default-cost - Allows user to define a cost for default summary route sent into a stub or NSSA areas.

area nssa - Allows user to configure an area as NSSA.

area range - Allows user to configure a summarize route for a range of IP addresses. This command is used on ABR.

area stub - Allows user to configure an area as a stub area or a totally stubby area.

area virtual-link - Command is used to configure a virtual link.

auto-cost - Allows user to modify the reference bandwidth value used for cost calculation on OSPF interfaces.

compatible rfc1583 - Summary route cost calculation method that was introduced in obsolete RFC 1583 will be used.

D.3 Address-Family Configuration

default-information originate - The router will advertise a default route into the OSPF domain.

default-metric - Sets a default-metric value for redistributed routes.

discard-route - Sets a discard route entry in the routing table of the ABR or ASBR. The discard route is used to prevent routing loops.

distance ospf - Modify the administrative distance of intra-area, inter-area and external routes. The domain tag is usually used in route maps for policy decisions or to prevent loops when redistributing routes.

log-adjacency-changes - Allows user to send syslog messages informing about a neighbour going up or down.

router-id - Set a fixed router ID.

neighbor database-filter - Allows user to filter outgoing LSAs to an OSPF neighbor. It has similar function as the ip ospf database-filter all out command in D.4.

network area - Used to define interfaces on which OSPF runs and in which area it belongs. If not used, the lowest loopback or interface IP address is used instead.

summary-prefix - Creates a IPv6 summary prefix for a range of IP addresses learned from other routing protocols.

D.4 Interface Configuration Mode

ospfv3 authentication - Allows user to specify authentication method for an interface.

ospfv3 authentication-key - Sets a password used by neghboring routers to authenticate the OSPF traffic. This option works only for simple password authentication method.

ospfv3 cost - Set a fixed cost of sending packets on an interface.

ospfv3 database-filter all out - Allows user to filter LSA on an interface. It has similar function as the neighbor database-filter command in D.2.

ospfv3 dead-interval - Sets the value of the dead interval on an interface.

ospfv3 demand-circuit - Suppresses hello messages and LSA refresh functions on an interface.

ospfv3 flood-reduction - LSAs will not be flooded in stable topology.

ospfv3 hello-interval - Sets the value of the hello interval on an interface.

ospfv3 mtu-ignore - Disables the MTU mismatch detection on an interface.

ospfv3 network - Allows user to configure OSPF network type(broadcast, NBMA, etc.) on an interface.

ospfv3 neighbor - Static configuration of neighbors in networks without broadcast capabilities.

ospfv3 priority - Sets the priority value on an interface. This value is used in the DR and BDR election.

ospfv3 retransmit-interval - Allows user to specify the time between LSA retransmissions on an interface.

ospfv3 transmit-delay - Sets an estimated time required to send a LSU on an interface. This command is used on very low-speed links.