OSPF extensions for the support of multi-area networks with arbitrary topologies

Miguel Gonçalves
Instituto Superior Técnico
Lisboa, Portugal
miguel.d.goncalves@tecnico.ulisboa.pt

ABSTRACT
Routing protocols are a major part of the Internet, providing information about the network to IP routers in order to allow them to forward received packets. Link state routing protocols, such as OSPF and IS-IS, are the most used nowadays, solving the count-to-infinity problem found in distance vector routing protocols by providing the complete network view to every router. Link state routing networks can be divided in areas to facilitate the scaling of the network. However, in current OSPF versions, inter-area routing uses a distance vector approach, bringing limitations to multi-area networks. In particular, they must have a two-level hierarchical structure, and optimal routing is not guaranteed for inter-area routing. This MSc Dissertation describes the development of two extensions of the OSPF protocol that overcome the mentioned limitations, one for OSPFv2 (IPv4) and the other for OSPFv3 (IPv6). They provide a link state approach to inter-area routing, allowing arbitrary multi-area topologies and optimal inter-area routing. An implementation in Python of the base OSPF protocol was also developed to support the development of the extensions, which were created on top of the base implementation. GNS3 networks were created and used to test the implementation, showing that both versions of the base implementation and the OSPFv2 extension are operational. This dissertation gives some background on OSPF and its multi-area extensions, describes the software architecture and the developed implementation, and presents the experiments performed and the respective results. This MSc Dissertation was supported by the Instituto de Telecomunicações.

Author Keywords
OSPF; OSPFv2; OSPFv3; Routing; Link state routing; Routing protocols.

INTRODUCTION

Motivation

Information travels through the Internet contained in IP packets, having a source device and destined to another device. Routers are the pieces of equipment that allow IP packets to reach destinations not directly connected to their sources. Routers allow it by performing packet forwarding, i.e. selecting the interface through which a received packet will be sent in order to reach its destination. In order to forward packets, routers must possess information about the network, which is learnt and disseminated in the network using routing protocols. Routing protocols are therefore a vital mechanism of the Internet. They can be either inter-domain, when routing is performed across multiple routing domains, or intra-domain when routing occurs inside a single routing domain. There are two main types of intra-domain routing protocols: DVR protocols, and LSR protocols.

DVR protocols, such as the RIP, operate by exchanging messages called distance vectors between directly connected routers in a network. Distance vectors bind a network destination to a path cost. In RIP, usually a hop receives a cost of 1. By receiving the distance vectors of all of the neighbor routers, a router is able to calculate the shortest path to reach any destination in the network, and the respective cost.

However, the approach brings problems, in particular a situation known as count-to-infinity. It occurs when a router becomes unreachable, for example by crashing. Its neighbor routers detect that they cannot reach it, however they keep receiving distance vectors from other neighbor routers in the network containing paths to the now unreachable router. Since routers running DVR protocols do not have a complete network view, the new paths are accepted and then disseminated with increased costs, leading to an infinite cycle where the failure of the router is never discovered and instead paths to it are successively disseminated with higher path costs. RIP solves the situation by declaring a path cost of 16 as infinity, which brings limitations to the size of RIP networks.

LSR protocols provide the complete network view to the routers, preventing the count-to-infinity problem. OSPF and IS-IS are two of such protocols. OSPF currently has 2 different versions, OSPFv2 and OSPFv3, respectively for IPv4 and IPv6. The main difference between the versions is the sepa-

ration of topological and addressing information in OSPFv3 when disseminating routing information in the network.

OSPF networks (i.e. networks with routers running OSPF) can be divided in areas to facilitate scaling. Current OSPF versions use a distance vector approach for inter-area routing. Such approach brings two limitations for multi-area networks in OSPF: the networks are limited to a two-level hierarchical structure with a single area in the top level, and there are cases where it is not guaranteed that the selected path to a destination is the shortest (i.e. optimal routing is not guaranteed).

Objectives

The general objective of the current MSc Dissertation is to develop two OSPF extensions, one for OSPFv2 and another for OSPFv3, which can overcome the previously mentioned restrictions by applying a LSR approach to inter-area routing in OSPF, allowing arbitrary multi-area topologies and optimal inter-area routing. The extensions are described in [16].

The extensions have been designed to take advantage of the OSPF base mechanisms, in order to only require ABRs to be updated so as to accommodate the extensions. The extensions are therefore transparent to area internal routers.

Achievements

Both OSPF versions have been implemented according to the current specifications, with the OSPF extensions being created on top of the base OSPF implementation. It is an objective of the current dissertation to create a proof of concept of the extensions, where readability is preferred over execution speed. For that reason, the base OSPF implementation and the extensions have been written in Python, and not in faster but less readable languages such as C++. Functional tests were performed on the base OSPF implementation and on the respective extensions using GNS3 and Wireshark. Another objective of the current dissertation is to create an implementation of the base OSPF and of the extensions which is compatible with Cisco routers, which therefore were used in functional and interoperability tests.

Structure

Section 2 will provide information about routing protocols. Section 3 will describe the developed OSPF extensions. Section 4 will describe the software architecture of the implementation of the base OSPF protocol and of the extensions. Section 5 will describe the implementation. Section 6 will describe the evaluation of the implementation. Section 7 will describe the conclusions and future work of the current dissertation.

ROUTING PROTOCOLS

Introduction to routing protocols

There are several types of physical devices connected to the Internet. From the point of view of computer networking, they can be classified in three categories: hosts, network devices, and communication media. Hosts, such as servers and mobile phones, are sources and destinations of the information that travels through the Internet. Network devices perform routing and forwarding of information, allowing it to reach destinations not directly connected to the information sources. Hosts and network devices are physically connected by communication media such as copper wires and Ethernet cables, forming networks that can be of several types.

In the network layer, information travels through the network in packets which are forwarded by IP routers. Forwarding is performed according to forwarding tables, which tell the outgoing interface of a packet according to its destination IP address. Each router maintains its own forwarding table, which can be created either statically by network administrators or maintained dynamically using routing protocols such as RIP or OSPF. Routing protocols have the advantage of being able to automatically update the forwarding tables of the routers in reaction to failures of links and routers in the network, or to the appearance of new ones.

Routing protocols can be inter-domain, when the routing is performed across multiple routing domains, even if inside the same AS; or intra-domain, when routing occurs inside a single domain. The BGP is the currently used inter-domain routing protocol. Intra-domain routing protocols can be further subdivided; the most important categories are distance vector and link state. RIP is an example of the former, while OSPF and the IS-IS protocol are examples of the latter.

The Open Shortest Path First protocol

DVR protocols have been largely supplanted by LSR protocols such as OSPF, as stated in [12]. LSR protocols provide routers with a complete network view, preventing the count-to-infinity problem of DVR protocols and removing the limitation in the number of hops of a network path. OSPF for IPv4, known as OSPFv2, is currently defined in RFC 2328 [13], while OSPFv3, for IPv6, is currently defined in RFC 5340 [10].

Section 2 of [17] describes both types of routing information and the OSPF synchronization mechanisms. From the point of view of routing, 3 types of information must be known:

- Topological information: Provides information about the routers, the costs assigned to their interfaces, and the links between routers;
- Addressing information: Provides information about the routable address prefixes assigned to routers and links;
- Link information: Provides information about the addresses needed to carry packets to the next-hop router.

OSPF stores the network information in a LSDB, created in every router. LSAs contains specific parts of the routing information, and are the building blocks of a LSDB. Except for the link information in OSPFv3 and assuming the network only has one area, every router in the network must have the same LSAs in its LSDB.

The main differences between OSPFv2 and OSPFv3 lie in the way the routing information is stored: while OSPFv2 mixes topological and addressing information in its LSAs (with link information being implicit in their contents), OSPFv3 separates each type of information in different LSAs, creating for the effect new types of LSAs not present in OSPFv2.

OSPF uses 3 synchronization mechanisms to keep the network view consistent and updated across all routers. The mechanisms are the following:

- Hello protocol: Allows routers to discover active neighbor routers in a subnet they are attached to;
- Reliable flooding procedure: Allows the dissemination of network information across a network, ending when all routers have received it;
- Initial LSDB synchronization process: Allows synchronization of the network information known by routers that become neighbors, such as when a new router joins a network.

OSPF has 5 types of links between routers: point-to-point links, point-to-multipoint links, Non-Broadcast Multi-Access links, broadcast (shared) links, and virtual links. The most relevant are the point-to-point links and the broadcast links, which will be the ones focused on in this work.

Routers and links have identifiers. Transit shared links, which can connect 2 or more routers at the same time, are the most complex case. To identify them, a router known as the DR must be elected using the Hello protocol to provide the identifier of the link. A BDR is also elected to replace the DR in case it fails.

To facilitate scaling, OSPF routing domains can be divided in areas. As described in Section 5.1.3 of [9], some restrictions apply. There must be a special area called backbone area directly connected to all others, and all inter-area traffic must pass through the backbone area even if it not the shortest path to the destination. The structure of an OSPF multi-area network is illustrated in Figure 1. A router interface always belongs to one and only one area; a router therefore always belongs to one or more areas.

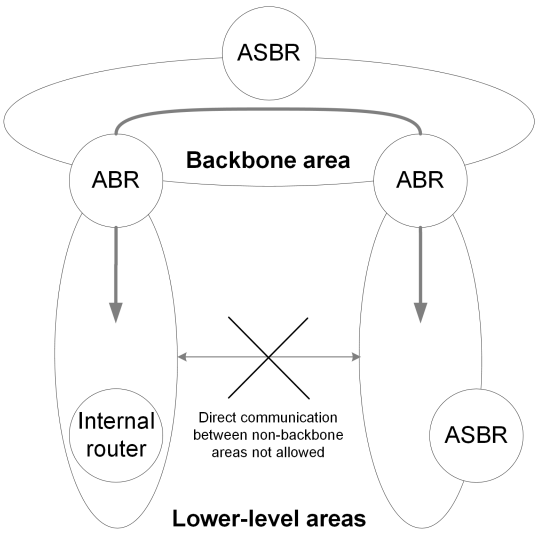

Figure 1. Topology of an OSPF hierarchical network
Source: Adapted from [17]

Regarding the structure of an OSPF area, there are 3 types of routers. ABRs connect the backbone area with at least one other area: they have at least one interface assigned to the backbone area, and at least one interface assigned to every other area they are connected to. Direct connection between 2 non-backbone areas is not possible.

ASBRs connect the OSPF routing domain with other routing domains or ASes. Routers that do not belong to the previous two types (i.e. routers whose interfaces belong all to the same routing domain and area) are called internal routers.

OSPF EXTENSIONS

General description

The main goal of the current MSc Dissertation is to implement two OSPF extensions, one for each version, that will address and overcome the restrictions in shortest path calculation and selection of network topologies currently present in OSPF hierarchical networks. The extensions are described in [16], which will be referred to as the *base article*, and the implementation supporting the MSc Dissertation, present at [11], is based on it with a few changes.

The use of a DVR approach by OSPF in inter-area routing imposes restrictions in the selection of the shortest path to a network destination, and limits the network topologies to a two-level hierarchy with only one area at the top level. The OSPF extensions described in the base article, one for each OSPF version, propose a LSR approach to inter-area routing to overcome the restrictions. Such approach to inter-area routing is currently not implemented by any LSR technology, as stated in Section 3.2 of [17].

Given the role of ABRs as gateways between areas, they form a routing overlay over an OSPF hierarchical network, similar to having a single-area network running on top of the hierarchical network. In a graph representing the ABR overlay, the nodes are the network ABRs, while the arcs are the shortest intra-area paths between two ABRs connected to the same area. The arc costs correspond to the cost of the shortest path to the neighbor ABR. Internal routers do not form part of the graph, including as well ASBRs that are not ABRs, just like layer-2 equipment such as switches can be abstracted from graph representations of OSPF networks. Figure 2 shows a multi-area OSPF network with a topology not supported by the base protocol. In the network, R1 is connected to the address prefix ap1, while R2 is connected to the address prefix ap2.

Each ABR computes its local view of the network using the information stored in the LSDBs of the areas it is connected to. Then, the information is stored and shared between ABRs using 3 new types of LSAs: ABR-LSAs, Prefix-LSAs, and ASBR-LSAs, called *overlay LSAs* in the base article. An ABR will update and flood new overlay LSAs every time it detects a relevant change in one of its LSDBs.

Since area internal routers are unaware of the details of inter-area routing and rely solely on the information supplied by ABRs to know about prefixes and ASBRs outside the area, the OSPF extensions are transparent to them and no modification of the internal routers is required. Both versions of OSPF

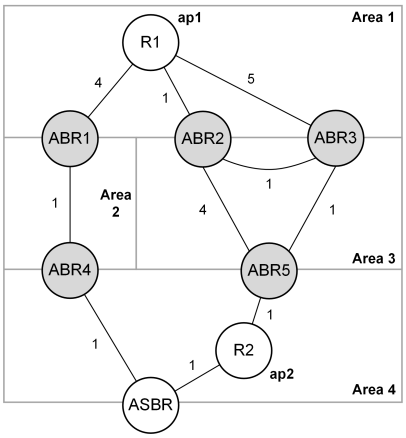

Figure 2. Topology of an OSPF multi-area network
Source: Adapted from [16]

already support new types of LSAs. In particular, since overlay LSAs must reach the entire network regardless of the area, they will have AS-scope. Area internal routers will store and flood overlay LSAs as if their LS Type was known, even if they do not use the information stored in the LSAs. The OSPF synchronization mechanisms ensure the reliable distribution of the overlay LSAs and therefore the correctness of the network routing information.

Both extensions operate in the same way. The only design difference is in the Prefix-LSA, which has different content in each OSPF version.

Overlay LSAs

All of the proposed overlay LSAs advertise the shortest path cost from the Advertising Router, which is always an ABR, to another ABR, a prefix, or an ASBR, located in the same areas the source ABR is connected to. ABR-LSAs, Prefix-LSAs and ASBR-LSAs advertise costs to ABRs, network prefixes and ASBRs, respectively. The advertised path is always intra-area i.e. it will be calculated using only the Router-LSAs, Network-LSAs and Intra-Area-Prefix-LSAs (only in OSPFv3) present in the area LSDBs, even if the actual shortest path crosses other areas. It is possible for the path to cross an intermediate ABR, as long as the inbound and outbound interfaces belong to the same area as the Advertising Router and the destination.

The content of the overlay LSA bodies only differs between OSPF versions for the Prefix-LSA. The Advertising Router is always the source ABR. In OSPFv2, the overlay LSAs are type 11 Opaque-LSAs (also known as Opaque-AS LSAs), with the LS Type field always set to 11. Considering the IANA assignation of Opaque Types available at [7], which states that values up to 10 are registered (the registration of value 10 is temporary), in our implementation ABR-LSAs, Prefix-LSAs and ASBR-LSAs received a value of 11, 12 and 13, respectively.

OSPFv3 already supports new LSA types. The first byte of the LS Type (the U-bit) is set to True to indicate that the LSA should be stored and flooded just like a regular LSA, while the second and third bits (the S1 and S2 bits) are set to 10 to indicate a AS-scope. IANA has registered LSA Function Codes (which denote the LSA Type in OSPFv3) up to 16, with the registration of value 16 being temporary, as seen at [8]. Therefore, in our implementation, the ABR-LSA, Prefix-LSA and ASBR-LSA are given a value of 17, 18 and 19 as LSA Function Code. This information is summarized in Table 1.

	ABR-LSA	Prefix-LSA	ASBR-LSA
OSPFv2 LS Type	11	11	11
OSPFv2 Opaque Type	11	12	13
OSPFv3 LS Type	17	18	19

Table 1. LS Type and Opaque Type of overlay LSAs

An overlay LSA can store information on as many network elements of the same type as required, as long as all of them are associated with the same ABR. The set of parameters that describes a network element inside an overlay LSA is repeated for every element being described. For that reason, each ABR produces at most one overlay LSA of each type.

For each described network element, all overlay LSAs store a metric value which consists of the shortest intra-area path cost from the source ABR to the element, either another ABR, a prefix or an ASBR. Prefix-LSAs in OSPFv3 also store the number of prefixes in the LSA. Additionally, each overlay LSA contains the following fields for each element:

ABR-LSA The Router ID of the neighbor ABR

Prefix-LSA In OSPFv2, the subnet address and subnet mask. In OSPFv3, the prefix length and the address prefix, along with its options

ASBR-LSA The Router ID of the neighbor ASBR

Differences to base article

There are a number of differences between the the OSPF extensions as described in the base article and the implementation supporting the current MSc Dissertation.

In the base article, ABR-LSAs can store information on multiple ABRs, while Prefix-LSAs and ASBR-LSAs can store information on only one prefix or ASBR. For the implementation, it was decided to make all overlay LSAs consistent by allowing all 3 types to store information on multiple network elements on a single LSA. This behavior is seen in the current OSPFv3 specification, in the Intra-Area-Prefix-LSAs and Link-LSAs, which can store multiple prefixes in a single LSA. It was also decided to add a Prefix Number field to the Prefix-LSA in OSPFv3, present as well in the two LSA types of the current specification.

Analysing RFC 2328[13] for OSPFv2 and RFC 5340[10] for OSPFv3, it is possible to conclude that Router-LSAs and Intra-Area-Prefix-LSAs use 2 bytes to store the metrics. AS-External-LSAs, Summary-LSAs of both types, Inter-Area-Prefix-LSAs and Inter-Area-Router-LSAs use 3 bytes to store metrics. In the first case the metric is an interface cost, while in the second case the metric is the sum of interface costs. In overlay LSAs, the metric is a sum of interface costs as well. For that reason, it was decided to increase the length of the metric field from 1 to 3 bytes in all overlay LSAs for the implementation.

Section 3 of the base article proposes a mechanism similar to the OSPF initial LSDB synchronization process, with new messages, where an ABR that has just joined the network can request its overlay LSAs to the ABRs in the same areas. As described in Section 4.2.2 of [15], since the overlay LSAs are Opaque-AS-LSAs which are recognized by OSPF, they are already exchanged during the initial LSDB synchronization process between the new ABR and its neighbor routers even if they are area internal routers. Given that in OSPFv3 new LSAs can be stored and flooded like regular LSAs if their U-bit is set (which happens for the overlay LSAs), it can be concluded that there is no need to implement an additional mechanism to perform the initial LSDB synchronization process. However, OSPFv2 routers must signal their support of Opaque-LSAs by setting the O-bit in the Options field of DB Description packets. Otherwise, they will not receive Opaque-LSAs during the initial LSDB synchronization process.

Practical example

Table 2 shows the overlay LSAs for the network from Figure 2. It considers the address prefix ap1 connected to R1, in area 1, and the address prefix ap2 connected to R2, in area 4. All costs shown in the overlay LSAs are costs of intra-area paths, including the costs of paths between ABRs.

ABR1	**ABR2**	**ABR3**	**ABR4**	**ABR5**
ABR2: 5	ABR1: 5	ABR1: 9	ABR1: 1	ABR2: 2
ABR3: 9	ABR3: 1	ABR2: 1	ABR5: 3	ABR3: 1
ABR4: 1	ABR5: 2	ABR5: 1		ABR4: 3

ABR-LSAs

ABR1	**ABR2**	**ABR3**	**ABR4**	**ABR5**
ap1: 4	ap1: 1	ap1: 5	ap2: 2	ap2: 1

Prefix-LSAs

ABR4	**ABR5**
ASBR: 1	ASBR: 2

ASBR-LSAs

Table 2. Overlay LSAs for the network from Figure 2
Source: Adapted from [16]

ABRs 1, 2 and 3 are connected to area 1, with all intra-area paths going through R1. ABR1 shares only area 1 with the other 2 ABRs, so the path costs to reach ABR2 and ABR3 that are introduced in its ABR-LSA are the ones that go through R1. On the other hand, ABR2 and ABR3 are both connected to area 1 and to area 3, with the shortest intra-area path between the 2 routers going through area 3 with cost 1. Therefore, for the ABR-LSAs of both routers, a cost of 1 to reach the other ABR is inserted. ABR3 does not insert in its ABR-LSA the shortest path cost of 5 to reach ABR1 by going through areas 3, 4 and 2, since it is an inter-area path, neither the path cost of 6 that can be obtained by going through areas 3 and 1. On the other hand, ABR2 puts in its ABR-LSA a shortest path cost of 2 to reach ABR5 despite being directly connected to it, since the path that goes through ABR3 is still an intra-area path even if it goes through an ABR.

In the network, ap1 and ap2 are being advertised inside area 1 by R1 and inside area 4 by R2, respectively. ABRs 1, 2 and 3 are connected to area 1, and therefore create and advertise Prefix-LSAs stating that ap1 is in a directly connected area, with the shortest intra-area path cost being the one that directly reaches R1. The same happens for ABR4 and ABR5 regarding ap2 and R2.

The network has a single ASBR, an area internal router in area 4. Given that only ABR4 and ABR5 are connected to the area, they are the only routers to create and flood an ASBR-LSA with the shortest intra-area path cost (which is also the overall shortest path cost) to reach the ASBR from them.

After the network stabilizes and all ABRs know all of the network overlay LSAs, they can compute the overall shortest path costs to reach the network prefixes and the ASBR. By using the network ABR-LSAs to build the overlay graph, each ABR will know the overall shortest path cost to reach all other ABRs in the network. For example, ABR1 will know that its overall shortest path costs to reach ABR2, ABR3, ABR4 and ABR5 are 5, 5, 1 and 4, respectively. Prefix-LSAs and ASBR-LSAs tell ABRs which prefixes and ASBRs are associated with each ABR, respectively, along with the intra-area shortest path costs from the ABR to the network element. ABR1 learns that ABR4 can reach the ASBR with cost 1 and ap2 with cost 2, while ABR5 can reach the ASBR and ap2 with cost 2 and 1, respectively. By adding the overall shortest path cost to each ABR with the intra-area shortest path cost from the ABR to the network element, ABR1 can obtain the overall shortest path cost from itself to the network element. To reach ap2 and the ASBR from ABR1, the next hop router will be ABR4.

With the shortest path costs calculated, ABRs create Network-Summary-LSAs in OSPFv2 and Inter-Area-Prefix-LSAs in OSPFv3, for prefixes, and ASBR-Summary-LSAs in OSPFv2 and Inter-Area-Router-LSAs in OSPFv3, for ASBRs, containing the overall shortest path costs to reach them. ABR1, ABR2 and ABR3 will tell R1 that the shortest path cost from them to ap2 is 3, 3 and 2, respectively, and 2, 4 and 3 to reach the ASBR, respectively. By summing the costs advertised by the ABRs with the intra-area shortest path costs to reach each of the area ABRs, R1 can conclude that the shortest path cost to reach ap2 and the ASBR through ABR1 is 7 and 6, through ABR2 is 4 and 5, and through ABR3 is 7 and 8, respectively. With the information, R1 can now select ABR2 as the next hop router to reach ap2 and the ASBR, knowing that the overall shortest path cost to reach ap2 and the ASBR is 4 and 5, respectively.

SOFTWARE ARCHITECTURE

General description

The program that supports the current MSc Dissertation consists of an OSPF router, able to be interconnected with other machines running either the same router program or current OSPF implementations. The program runs both OSPF versions as defined in the current specifications and supports Opaque-LSAs as well, with the OSPF extensions being created on top of the base implementation.

Its software architecture is based on the OSPF specifications as described in RFC 2328 [13] and RFC 5340 [10], and also on the OSPF implementation described in [14]. The architecture

is object-oriented, with an hierarchy of four software layers which from top to bottom are: the router, the area, the interface, and the neighbor. The architecture is illustrated in Figure 3. Grey lines represent the interactions between layers, and arrows indicate the direction of the interaction.

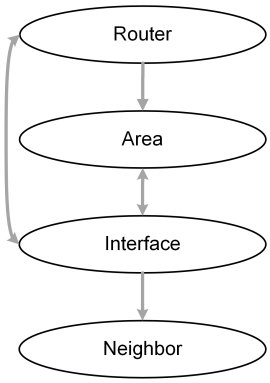

Figure 3. Software architecture of the supporting program

Router neighbors are directly associated with router interfaces, to the point where a router can have two or more different neighbor relationships with the same neighbor router through the same number of interfaces. For that reason, the neighbor layer can be placed under the interface layer.

Organizing the router, area and interface layers is less direct. Routers have at least one interface, which will belong to one and only one area. Areas also have at least one interface, regardless of the routers those interfaces belong to. Therefore, a router can belong to a single area while having several interfaces, or at most belong to as many areas as the interfaces it has. Given that, the router layer can be placed at the top of the hierarchy, with the area layer immediately below and the interface layer below it, leaving the neighbor layer as the bottom layer of the hierarchy.

Despite the differences between OSPFv2 and OSPFv3, the software architecture is suitable for both OSPF versions. In particular, state machines, mechanisms, and the OSPF base concepts such as the area, the interface and the neighbor router remain unchanged. The main consequences of the separation of topological and addressing information in OSPFv3 are the creation of new types of LSAs, and changes in the structure and content of existing LSA types. These changes are supported by the software architecture and only affect the implementation.

The top software layer contains four classes: the router, the OSPF routing table, the interface to the routing table stored in the kernel of the machine running the program, and the part of the LSDB that stores the overlay LSAs. The area layer contains the area class and the remainder of the LSDB, where the regular LSAs are stored. One router instance is associated with at least one area instance. Both the interface and the neighbor layers contain a single class, which respectively represents an OSPF interface and an OSPF neighbor router from the point of view of an interface. An area instance is associated with at least one interface instance, while the latter may have no neighbor instances associated.

A few classes support the program operation without belonging to a specific layer, being instantiated by classes in more than one layer. Both router and interface instances are associated with at least one instance of the socket class, while both interface and neighbor instances are associated with at least one instance of the timer class.

Software layers

The router layer contains the OSPF top-level data structures, as described in Section 5 of [13], such as the Router ID, the list of area data structures for the areas the router belongs to, and the OSPF forwarding table. The layer controls directly or indirectly all layers of the program.

The area layer contains the OSPF area data structures, as described in Section 6 of [13]. Each area data structure contains elements such as the Area ID, the interfaces connected to the area, and most of the LSAs defined in the base OSPF specifications in use by the supporting implementation. In particular, Router-LSAs, Network-LSAs, Network-Summary-LSAs in OSPFv2, and Intra-Area-Prefix-LSAs and Inter-Area-Prefix-LSAs in OSPFv3, are stored in the area layer.

The interface layer contains the OSPF interface data structures, as described in Section 9 of [13]. Each interface data structure contains elements such as the IP address, the network DR and BDR, the current interface state, the interface cost, the list of OSPF neighbors associated with the interface, and the list of the interface Link-LSAs for OSPFv3. Unlike the upper software layers, the interface layer only has one class, the interface class. The interface layer stores the states machines of the interface and the neighbor.

The neighbor layer contains the OSPF neighbor data structures, as described in Section 10 of [13]. Each neighbor data structure contains elements such as the current neighbor state, the Router ID and the IP address of the neighbor, and the neighbor inactivity timer. Like the interface layer, the neighbor layer has only one class, the neighbor class.

OSPF extensions

Changes in the router and the interface layers were required to accommodate the OSPF extensions implemented for the current MSc Dissertation. Most changes were performed in the router layer.

A new class was added to the router layer, a data object for storing the overlay LSAs. The class is similar to the LSDB class of the area layer, storing overlay LSAs and allowing the addition, update and removal of elements from its instances. The router class creates a single instance of the overlay LSDB class. It was decided to store the overlay LSAs in the router layer since the current OSPFv3 specification, described in Section 4.1 of [10], assigns unknown LSAs with their U-bit set and AS-scope to the top-level data structure, which matches the characteristics of the overlay LSAs.

The process of building the OSPF routing table has been updated. After the intra-area OSPF routing table is generated,

the self-originated overlay LSAs are updated using the routing table content, and are flooded to the network. Then, the OSPF routing table is updated using only the information contained in the overlay LSAs. Unlike in current OSPF specifications, it is possible for the intra-area shortest path costs to be replaced by inter-area shortest path costs to the same destination.

The information of the updated OSPF routing table is then transferred to the kernel routing table. Network-Summary-LSAs and Inter-Area-Prefix-LSAs are created, installed in the respective area LSDB and flooded with the information of the updated OSPF routing table.

The interface layer required less changes. As the overlay LSAs in both OSPF versions are stored and flooded like regular LSAs, the processing of incoming OSPF packets could remain intact. Only the access to the LSDB was updated. In order to add, remove or update an overlay LSA, an interface instance will directly access the extension LSDB in the router layer. Regular LSAs continue to be added to, and fetched and deleted from the area LSDB.

IMPLEMENTATION

Overview

The implementation supporting the current MSc Dissertation is available at [11], and was developed using Python 3.6 [6] in machines running Ubuntu 18.04 [4]. The structure of the program follows the software architecture defined in the previous section. In particular, the root directory of the project contains a Python package for each software layer defined previously. Each class is stored in a different module named after the class, and stored inside the corresponding package. The implementation has a total of eight packages.

Each class contains the code and the parameters for both OSPF versions, when required. The program can run both OSPF versions at the same time, each version in a separate process with different instances of the same classes. No state is shared between the processes.

The interaction with the user is provided by a command-line interface, which is available at every moment after the router program is started. The different commands available allow to access the content of the kernel routing table, the content of the router LSDBs, and the information about the router neighbors, among other functionalities. The commands are the same for both OSPF versions, and fetch the content from the two router processes if both OSPF versions are running at the same time.

Source code

In our implementation, most constants are stored in the *conf.py* file, in the configuration package. The file does not contain any classes, consisting in a list of static parameters. The static parameters defined by the current OSPF specifications, such as the time to wait after sending each Hello packet, the time to wait until a neighbor is declared dead if no Hello packets are received from the neighbor, and the types of the LSAs and the packets, are stored in the file.

The general package contains classes that do not belong to a specific software layer of the program, and are used by classes in more than one layer. The socket and the timer classes are stored in the general package, along with a third class containing utility functions used throughout the code.

In our implementation, LSAs and OSPF packets are represented using data objects. The LSA and the packet headers are represented as different classes. Each LSA body type and each packet body type are represented as different classes as well. Each data object can be packed into a byte stream in order to be sent to the network, or unpacked from a byte stream received from the network. Although the LSA and the packet classes are used by more than one software layer, given their particular role and their number, it was decided to place the classes in separate packages and not in the general package.

The router package contains the classes of the router layer. The router class controls directly or indirectly all layers of the program. It is one of the largest classes of the program, with more than 1.000 lines of code. During the program startup, a router main process is started, which controls the operation of the corresponding OSPF version in the program. The kernel routing table is also set by the router main loop, using a separate thread.

No loops are run inside the area layer classes, which act as data objects to store information. In particular, the area class is used to start all underlying interfaces during the router startup, and provide the initial content of the respective LSDB.

The interface class, the only class of the interface package, is the largest class of the code with more than 1.500 lines of code. The class contains the code of all of the events defined in the OSPF specifications that change the states of the interface and the neighbor state machines. Each event is represented as a different method. Additionally, the code of the DR election is also contained in the interface class, with each step of the election being represented as a different method. In our implementation, OSPFv3 interface instances can store Link-LSAs, supporting the addition, deletion and getting of stored LSAs, the same way it is done in the LSDB class. The interface class also contains a loop, similar to the router main loop.

The neighbor class is the only class of the neighbor package. It is instantiated by the interface class when an Hello packet from a new neighbor router is received. Similar to the area class, no loops are run inside the neighbor class.

OSPF extensions

Most of the code written for the OSPF extensions was placed in the router layer. The interface layer suffered minor changes in order to accommodate the OSPF extensions.

A new class was created in the router layer, the extension LSDB class. The class is very similar to the LSDB class of the area layer, storing the overlay LSAs of the router. Overlay LSAs of the same type are stored in the same list, and each LSA list is protected by a thread-safe lock. The class contains methods to get, add and delete LSAs stored in the class instance. The class contains as well a method that returns the shortest path tree of the ABR overlay, in the same format as the area shortest path tree returned by the base LSDB class.

The procedure of updating the kernel routing table has been updated in the router class. The new steps are only executed if the router is an ABR. The router main process counts the number of areas that the router is connected to, according to the initial configuration. If the router is connected to more than 1 area, then the router is an ABR.

For ABRs, a copy of the overlay LSAs is obtained together with the copies of the area LSDBs. An OSPF routing table is created, and the table receives the paths to the prefixes inside the directly connected areas, which allows the router to update its overlay LSAs. The updates are written to the extension LSDB class and to the copy created by the update procedure, to avoid fetching a new copy of the overlay LSAs.

The router then updates the OSPF routing table using the information of the overlay LSAs, which will always provide the shortest path costs to each prefix in the network. If a prefix already has an entry in the OSPF routing table but with an higher shortest path cost, then the entry is updated. With the OSPF routing table updated, the router updates the kernel routing table. Finally, the router creates Network-Summary-LSAs in OSPFv2, and Inter-Area-Prefix-LSAs in OSPFv3, using the information of the updated OSPF routing table. The new LSAs are stored in the corresponding area LSDBs and flooded to the network.

The overlay LSAs are represented in the program by new body classes in the LSA package. The new classes implement the body abstract class, and can be packed into byte streams and unpacked from byte streams. The LSA interface class contains methods that call the methods of the overlay LSA body classes, and the class supports the creation and manipulation of overlay LSAs. For that reason, overlay LSAs can be handled by the program like regular LSAs.

During the program startup, each interface instance receives a reference to the extension LSDB class, along with the reference to the area LSDB, in order to allow the interface instances to access and manipulate the overlay LSAs.

EVALUATION

GNS3 networks

In order to test the implementation supporting the current MSc Dissertation, a total of three virtual networks have been created. GNS3 [3] was used to create the virtual networks, since it supports Cisco [1] virtual routers.

It is an objective of the current dissertation to perform interoperability tests on our implementation, using Cisco routers. Two versions of the Cisco IOS were used in the virtual networks: the version 12.2(15)ZJ (the Cisco 3725 router was used), and the version 15.7(3)M.

In order to virtualize the machines running our implementation, Docker containers [2] were used. Not only GNS3 supports Docker containers, but also containers are much faster to start and stop than virtual machines, and use less memory. While VMware [5] was used during the initial months of the development, it became impractical once the number of virtual machines in the network started to grow, leading to the virtual machines being replaced by Docker containers in the networks.

The test networks are divided in two groups. The first group contains two single-area OSPF networks, and the second group contains a multi-area OSPF network with a topology not possible with the current OSPF specifications. The networks inside the first group have the same topology and addresses. The only difference between the networks lies in the routers: in the first network a router can be represented by a Cisco router, while in the second network the same router is replaced by a Docker container running our implementation.

All of the networks are configured for both OSPFv2 and OSPFv3. The IP addresses for the networks always have the form 222.222.X.Y/24, in OSPFv2, and 2001:db8:cafe:X::Y/64, in OSPFv3, where X identifies the subnet inside the network and Y identifies the interface inside the subnet. For every network interface, the values for X and Y are the same for both OSPF versions. For example, if an interface has the IPv4 address 222.222.1.3/24, then the interface will also have the IPv6 address 2001:db8:cafe:1::3/64.

The first group of networks contains the networks 1-1 and 1-2. Both networks are single-area, and have 6 routers and 6 network prefixes. Four of the routers, R1, R4, R5, and R6, are connected to the same subnet. R1 connects the subnet with the rest of the network, and forms a loop with R2 and R3. A VPCS is connected to R1, and another VPCS is connected to R2. The network 1-1 contains one Docker container, the R4, while the remainder of the routers are Cisco routers. The network 1-2 replaces every Cisco router with a Docker container running our implementation. The networks use the Cisco IOS release 12.2(15)ZJ.

The second group of networks contains a single network, the network 2-1. The network contains 6 routers and 5 network prefixes, and is divided in 4 areas: the area 1 at the top, the area 2 in the center on the left, the area 3 in the center on the right, and the area 4 at the bottom. All of the network routers, except for R6, are ABR. R1, R2 and R3 are connected to the area 1 through a switch, which also connects R6 to the network. R1 is also connected to the area 2, while R2 and R3 are also connected to the area 3. The area 2 contains a single link, connecting R1 to R4. The area 3 contains 2 links, connecting R5 to R2 and R3, respectively. R4 and R5 are connected through the area 4, using two switches. R6 is a Cisco router, while the remaining routers are Docker containers. The network uses the Cisco IOS release 15.7(3)M.

Manual testing and results

During the implementation of the OSPF extensions, it was noticed that the Cisco 3725 routers did not correctly flood stored OSPFv3 LSAs with unknown LS Type during the initial LSDB synchronization process. The LSAs were flooded with the LS Function Code set to 0, preventing the identification of the respective LS Type by the receiving routers. An attempt was made to overcome the problem by replacing the Cisco routers of the network 2-1 with other Cisco routers running the Cisco IOS release 15.7(3)M, more recent than the Cisco IOS release executed in the Cisco 3725 routers.

While the problem disappeared, a new apparent bug was found: the Cisco routers being used seemed to treat all OSPFv3 LSAs with unknown LS Type as if all of them had the same LS Type. If a Cisco router received a LSA with unknown LS Type, and later it received another LSA with a different unknown LS Type and a lower sequence number, then the Cisco router would discard the second incoming LSA and send the first LSA back to the sending router, as if it was a more recent instance of the second LSA. To the best of our knowledge, the problem was not caused by a bug in our implementation or by a configuration error.

Considering the apparent bugs in the Cisco routers regarding LSAs with unknown LS Type in OSPFv3, it was decided to perform experiments only with the OSPFv2 extension.

In order to test the correct implementation of the OSPFv2 extension, two experiments have been performed in the network 2-1. The network has a multi-area topology that is not supported by the current OSPF specification, being divided in 4 non-hierarchical areas.

The first experiment tested the correct functioning of the OSPFv2 extension, including the generation and flooding of the overlay LSAs and the update of the kernel routing table based on the respective information. A second objective was to test the correct creation and flooding of Network-Summary-LSAs, in order to allow the area internal routers to know the network prefixes outside the area and how to reach them. The network 2-1 has one area internal router, a Cisco router, placed in the area 1.

For the experiment, the five network ABRs were started at the same time. The network was started and allowed to converge a total of three times, in order to measure the network convergence times. Table 3 shows the measured convergence times for each router in each execution, in seconds. The time was measured since the router startup until the last update of the respective kernel routing table, therefore including the 40 seconds that each router remains in the Waiting state after the respective startup.

Execution	R1	R2	R3	R4	R5	Average
1	189,2	196,2	193,4	187,1	191,7	191,5
2	190,5	201,2	200,2	187,8	194,6	194,9
3	171,9	175,3	173,1	169,3	165,5	171,0
Average	183,9	190,9	188,9	181,4	183,9	185,8

Table 3. Convergence time for each ABR of the network 2-1 during 3 executions, in seconds

On average the routers took 186 seconds to converge, a value 258% higher than the average convergence times of the Cisco routers R1, R5 and R6 in the network 1-1, and 151% higher than the average convergence times of the routers R1, R2 and R3 running our implementation in the network 1-2.

The increase in the number of routers converging at the same time, the additional processing required by the OSPFv2 extension, and the exchange of overlay LSAs along with regular OSPF LSAs, can be considered causes of the increment of the convergence times. However, it can also be considered that further work can be performed on the OSPFv2 extension to improve and optimize it.

Figure 4 shows the overlay LSAs generated by the network ABRs. Each router in the network generates one Prefix-LSA with the different prefixes of the areas that the ABR is connected to, and one ABR-LSA with the different ABRs connected to the same areas, along with the costs to reach the prefixes and the ABRs, respectively. Every interface cost was set to 10 for the experiment.

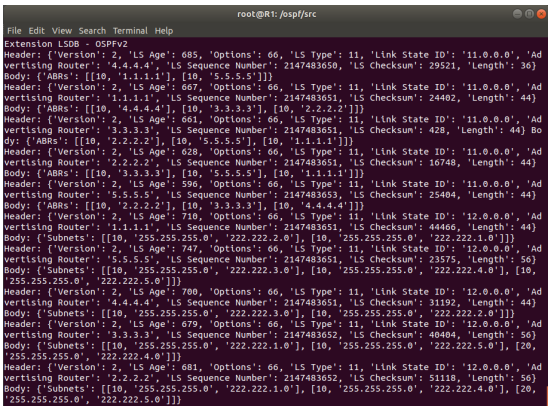

Figure 4. Overlay LSAs of the network 2-1 as seen in the command-line interface of R1

After the network converged and the convergence times were measured, R6 was started. When the router stabilized, the kernel routing table of the router contained paths to the 5 subnets of the network. In particular, the subnet with the address 222.222.3.0 could be reached through three different shortest paths, each path having a cost of 30 and going through a different ABR of the area 1.

The second experiment was similar to the first experiment, with the difference of setting the costs of the interfaces of R1 from 10 to 1. The remaining interface costs were not changed. The goal of the experiment was to test the correct functioning of the OSPFv2 extension in a network with different interface costs, and the correct transmission of the respective routing information to the area internal routers.

Similar to the first experiment, first the network ABRs were started and allowed to converge, and then R6 was started. With R1 having lower interface costs than R2 and R3, R6 inserted in the respective kernel routing table a single shortest path to the subnet with the prefix 222.222.3.0, with a cost of 21 and going through R1. The cost of the shortest path to the prefix 222.222.2.0, which already passed through R1 in the previous experiment, was reduced from 20 to 11. The shortest paths to the other prefixes were not changed.

CONCLUSION

LSR protocols are superior to DVR protocols, since they allow routers to have the complete network view, preventing the problem known as count-to-infinity that affects DVR protocols such as RIP. However, the current versions of OSPF use a distance vector approach to inter-area routing, bringing two important restrictions to OSPF multi-area networks: the networks must have a two-level hierarchical structure with a single area in the top level, and optimal routing between areas is not guaranteed.

For the current MSc Dissertation, two extensions to the base OSPF protocol were implemented, one for each version, which apply a link state approach to inter-area routing, providing the complete network view to all routers in the network even across areas. The extensions allow to create an ABR overlay, a logical network formed by the network ABR over the OSPF network, where all ABRs know the shortest paths to reach each other, and advertise as well the shortest path costs to reach the address prefixes located in their areas. New types of LSAs were created in order to carry the information between ABRs. As the current OSPF specification already supports the creation of new types of LSAs, the extensions are transparent to area internal routers.

An implementation of the current OSPF specifications was first created, in order to be later extended with the OSPF extensions. The software architecture contains four different layers: from the top to the bottom, the router, the area, the interface, and the neighbor, each layer corresponding to a data structure or a set of data structures defined in the OSPF RFCs. The router layer contains the top-level OSPF data structures, and is responsible for listening to OSPF packets in the network, coordinating the flooding of LSAs, and setting the kernel routing table. The area layer stores the area-scope LSAs. The interface layer manages the state machines of the interface and of the neighbor, elects the DR and the BDR of the subnet, and processes the incoming packets. The neighbor layer contains the neighbor data structure. The OSPF extensions required modifications in the router and in the interface layers, mainly a reformulation of the process of updating the kernel routing table of the machine running the program.

In order to test the implementation, three GNS3 networks were created, interconnecting Cisco routers and Docker containers running the implementation. Experiments were conducted in single-area OSPF networks, and in a multi-area network with a topology not possible with the current OSPF specifications. The single-area experiments were performed for both OSPF versions, and proved that the implementation can work correctly in single-area OSPF networks, being capable of interoperating with Cisco routers, and successfully forming networks only with machines running the program. Bugs were found in the Cisco routers that prevented the correct manipulation of LSA with unknown LS Type in OSPFv3. For that reason, the experiments in the multi-area network were conducted only for the OSPFv2 extension. The experiments proved that the extension is operating, being capable of generating and flooding overlay LSA across a multi-area network, and flooding the routing information to the area internal routers.

Future work

As a future work, our implementation of the base OSPF specifications should be improved and optimized, with the objective of fixing bugs in the implementation and decreasing the convergence times of networks with Docker containers running our implementation. Additionally, experiments should be performed on our implementation of the OSPFv3 extension, using networks with topologies not possible with the current OSPF specifications. The implementation of both OSPF extensions can then be improved and optimized like the base implementation, in order to fix bugs and reduce the respective convergence times. Another possibility of future work is the addition of support for ASBRs and domain-external prefixes, both in the base OSPF implementation and in the implementation of the OSPF extensions.

Finally, the OSPF extensions and their implementation can be described in an Internet-Draft, to be submitted to the IETF.

REFERENCES

[1] Cisco - Global Home Page. (????). https://www.cisco.com/

[2] Docker - Empowering App Development for Developers. (????). https://www.docker.com/

[3] GNS3 | The software that empowers network professionals. (????). https://www.gns3.com/

[4] Ubuntu. (????). https://ubuntu.com/

[5] VMware – Delivering a Digital Foundation for Businesses. (????). https://www.vmware.com/

[6] Welcome to Python.org. (????). https://www.python.org/

[7] 2020a. Opaque-LSA Option Types. (2020). https://www.iana.org/assignments/ospf-opaque-types/ospf-opaque-types.xhtml

[8] 2020b. OSPFv3 parameters. (2020). https://www.iana.org/assignments/ospfv3-parameters/ospfv3-parameters.xhtml

[9] Olivier Bonaventure. 2010. *Computer Networking : Principles, Protocols and Practice* (1 ed.). Université Catholique de Louvain. https://www.computer-networking.info/

[10] R. Coltun, D. Ferguson, J. Moy, and Ed. A. Lindem. 2008. RFC 5340 - OSPF for IPv6. (2008). DOI: http://dx.doi.org/10.17487/RFC5340

[11] Miguel Gonçalves. 2020. ospf-multiarea-arbitrary-topology. (2020). https://github.com/migueldgoncalves/ospf-multiarea-arbitrary-topology

[12] Gary Malkin. 1998. RFC 2453 - RIP Version 2. (1998). DOI:http://dx.doi.org/10.17487/RFC2453

[13] John T. Moy. 1998. RFC 2328 - OSPF Version 2. (1998). DOI:http://dx.doi.org/10.17487/RFC2328

[14] John T. Moy. 2001. *OSPF Complete Implementation* (1 ed.). Addison-Wesley, New York.

[15] André Pinheiro Pires. 2018. *OSPF extension to support multi-area networks with arbitrary topologies*. Technical Report. Instituto Superior Técnico. 76 pages. https://fenix.tecnico.ulisboa.pt/departamentos/dei/dissertacao/1972678479054624

[16] Rui Valadas. 2017. OSPF extension for the support of multi-area networks with arbitrary topologies. (2017). http://arxiv.org/abs/1704.08916

[17] Rui Valadas. 2019. *OSPF and IS-IS From Link State Routing Principles to Technologies* (1 ed.). CRC Press. DOI:http://dx.doi.org/10.1201/9780429027543