Analysis Of OSPFv2-BGP4 Interactions Using The SSF-Net Simulator

Diplomarbeit

Angefertigt unter der Leitung von
Prof. Anja Feldmann, Ph.D.
am Fachbereich Informatik
der Universität des Saarlandes

von

Hagen Böhm

Saarbrücken, 28. Oktober 2003
Hiermit erkläre ich an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Hagen Böhm
Saarbrücken, 28. Oktober 2003
Abstract

In recent studies about inter-domain routing message exchange, it was found out that the amount of routing updates sent by BGP routers was several times greater than theoretically expected. Coming along with the high number of updates, Internet routing instabilities, rapid fluctuations of network reachability and even the loss of internal connectivity in wide-area networks could be observed, which resulted in packet loss and increased latencies [17]. Further examinations have revealed that a serious part of such pathological inter-domain routing behaviour can be attributed to increasing use of intra-domain routing elements in the BGP routing process [23] [14]. This thesis examines the impact of the integration of OSPF shortest path metrics in the BGP routing process. Our work is based on network simulations which we performed using the SSFNet [31] network simulator. We used a BGP implementation that both applies IGP metrics as a tie-breaking rule for selecting best routes and announces prefixes with OSPF metric mapped MED attributes to neighboring ASes. Via single, AS internal link and router failures we generated OSPF routing changes and observed the effects on BGP. We show that the impact on global inter-domain routing depends strongly on number, connectivity and BGP policy filters of the border routers located in the AS where intra-domain routing changes caused OSPF-BGP interactions.
## Contents

1 Introduction .................................................. 1  
   1.1 Motivation ................................................. 1  
   1.2 Outline ................................................... 3  

2 Background ................................................... 5  
   2.1 Routing in the Internet ..................................... 5  
   2.2 Deployed Internet Routing Protocols ...................... 9  
      2.2.1 An Introduction to OSPF .............................. 10  
      2.2.2 An Introduction to BGP ............................... 15  
   2.3 Routing Simulations ....................................... 18  

3 Topologies .................................................... 21  
   3.1 What is a “typical” topology? ............................ 22  
   3.2 Approximation ............................................. 24  

4 Analysing OSPF ............................................... 29  
   4.1 The SSFNet Implementation ............................... 31  
   4.2 Parametrical Impacts On Convergence Time ............ 38  
      4.2.1 The Hello Protocol ................................ 40  
      4.2.2 Routing Table Calculation Intervals .............. 41  
      4.2.3 LSA Advertisement and Installation Restrictions ... 42  
      4.2.4 Optimal Parameters? ................................ 42  
      4.2.5 Summary ............................................. 44  
   4.3 Topological Impacts On Convergence Time ............ 45  
      4.3.1 The Hello Protocol ................................ 46  
      4.3.2 Routing Table Calculation Intervals .............. 48  
      4.3.3 Link Propagation Delays ............................ 51  
      4.3.4 Summary ............................................. 53  
   4.4 Conclusion ................................................ 54
5 Simulating OSPF-BGP Interactions

5.1 Causales and Effects ........................................ 56
5.2 Additional SSFNet Enhancements .......................... 61
5.3 Interaction Analysis ......................................... 63
  5.3.1 An OSPF-BGP Scenario ................................. 64
  5.3.2 Analysis of Single Link Failures ....................... 65
  5.3.3 Analysis of Single Router Failures ................. 73
  5.3.4 Future Work ........................................... 78
5.4 Conclusion .................................................. 79

6 Conclusion .................................................... 81
Chapter 1

Introduction

1.1 Motivation

From the early beginnings at the end of the 1960s where the Internet ancestor ARPANET was born, until today's worldwide accessible Internet the world has seen a tremendous growth and evolution of worldwide interconnected computer networks, especially in the last two decades. With the number of nodes in the networks and the density of links, connecting the nodes and the networks with each other, the complexity of routing\(^1\) has grown. In the early days when networks were small and manageable, routing was static and determined manually by system administrators. In the 1980s inter-networks achieved a complexity that made it necessary to deploy routing protocols that dynamically respond to topology changes by adjusting routing information. Soon it turned out that the enormous growth of destination hosts and networks led to exhausting resource consumption by the routing processes. To get under control exploding routing tables and updates, the global routing was subdivided into demarcated routing domains, so called Autonomous Systems (AS), which are doing relatively independent routing and aggregation of routing information when exchanging with other domains at their borders. The ASes are again subdivided into subdomains, also separating and filtering routing information and so forth. A hierarchy of several coexisting routing protocols and protocol implementations has been developed, keeping a sensitive balance between optimal routing and resource exploitation. However studies about routing behaviour in the last few years have revealed that this is not the case by far! They detected a lot of routing instabilities and rapid fluctuation of network reachability information in the Internet [17]\(^1\). It was found that the amount of exchanged routing information was many times over their theoretically expected number, and that most of them were useless and did not reflect the real network topological changes. Furthermore those routing instabilities and information fluctuations

\(^1\)When speaking of routing we mean the process of finding a path from a particular source to any reachable destination within the interconnected networks. For a more detailed description see chapter 2.1
can propagate from node to node throughout several network domains and thus may cost data loss in several data connections, delays in the routing process and additional resource overhead when filtering and matching unnecessary routing information. But what are the reasons for such a disastrous development?

Recently more and more people began with a closer examination of the origins of these phenomena. [2] for example found out that the hierarchical organisation of the Internet is disintegrating by continuously setting up new exchange points and peering relations between the Autonomous Systems. But also new route attributes like BGP communities [9] [6] and especially the upcoming connection between inter and intra-domain routing hierarchy have undermined the relative strongly demarcated routing domains and cause more and more routing update traffic [23]. Besides some bugs in routing software implementations, misconfigured protocols and oscillating hardware failures, the coupling of intra-domain routing elements, like IGP path lengths (metrics), to inter-domain routing elements like the BGP MED attribute seems to be one of the main responsible party for the occurring routing anomalies [23]. By directing inter-domain traffic based on intra-domain information, Internet Service Providers (ISP) found a powerful alternative to speed up traffic flows and reduce traffic load transiting their ASes. But via the connection of intra-domain with inter-domain routing, otherwise locally limited routing changes now get visibly to global routing. The more ISPs pursue such an approach, the more likely it is that a single change in intra-domain routing will induce a global chain reaction in inter-domain routing.

The interactions between routing protocols like OSPFv2 (intra-domain) and BGP4 (inter-domain) haven’t been studied sufficiently yet. In addition there are many open questions concerning future developments and usage of routing protocols that may intensify the interaction effects. Traffic Engineering for example is a topic of highly interest. Although it is not clear whether a combined routing and traffic engineering protocol will be ever deployed in the future, there are however serious attempts to do Traffic Engineering on top of traditional routing protocols [36]. In this context it is of particular concern, to use optimal intervals when applying link changes and to know what impacts those little changes may have on global routing.

In this thesis we are going to show potential effects that interactions between a local and a global routing protocol may trigger. We analyse different interaction causales on their impact and try to answer the question how to reduce or to avoid these impacts. Our work is based on simulated network environments including specific implementations of an intra and inter-domain routing protocol. The protocol interactions were simulated in a single complex network topology, which is a simplified adaption of some U.S. ISPs backbone networks. To better understand the influence of intra-domain routing, we attached particular importance to the analysis of such a protocol’s behaviour. The next section gives a more detailed outline of our work.
1.2 Outline

We begin in chapter 2 with an introduction of several routing specific terms, which we use in the course of this thesis. Subsequently we give a short description of the intra-domain routing protocol OSPFv2 and the inter-domain routing protocol BGP4. Our analysis has concentrated on these two routing protocols. The chapter ends with a few words about the network simulator SSFNet [31] that was used to perform the simulations. Since routing behaviour is extremely dependent on the underlying network topology, we present in chapter 3 the topologies we used to study OSPF’s routing behaviour and its interaction with BGP. We explain the difficulties of finding an adequate topology for the OSPF-BGP interactions and justify the design. In chapter 4 we finally begin with the actual analysis of our work. The chapter deals exclusively with the routing behaviour of the Interior Gateway Protocol OSPF. We present in detail the individual parametrical and topological impacts on load and convergence time of this protocol. The findings from these examinations are used later to understand the context of today’s and possible future OSPF-BGP interactions. Thereafter we study the actual OSPF-BGP interactions in chapter 5. Our work has concentrated on the use of OSPF shortest path metrics in tie-breaking rules of the BGP best path selection algorithm. Besides the regular use of IGP metrics we also deployed IGP metric mapped MED attributes. Triggered by single, AS interior link and router failures, we studied sort and scale of OSPF-BGP interaction effects. Chapter 4 and 5 both end with a conclusion where we summarised our findings. The thesis finishes with a conclusion in chapter 6 where we briefly resume our work and point out the main results.
Chapter 2

Background

In this chapter we provide explanations and descriptions of terms, protocols and tools to understand the background of our work. We begin with a series of terms used in computer networking. Thereafter we are going to introduce the routing protocols OSPF and BGP which we used to study interactions between intra and inter-domain routing. In the last section we give a brief description of SSFNet, the network simulator we used to generate our network scenarios and the protocol data that we analyse in chapter 4 and 5.

2.1 Routing in the Internet

When exchanging data between end-systems in computer networks, one question that arises is: How does data travel from a sender A to a receiver B? In a network like today's Internet, that contains more than 170 millions of such end-systems called hosts [15], which are distributed around the world, efficient navigation is a quite complex task. Due to the geographical distances between the hosts, the bulk of them is not directly connected. In fact a large mesh of special-purpose computers called routers is deployed to ensure connectivity between the hosts. Routers and hosts form the nodes in a huge network being interconnected via various media, which we simply refer to as “links”. Apart from some other requirements the routers have two general tasks:

1. They do data transit between hosts by forwarding incoming data hop by hop, i.e. router by router until it reaches the destination host.

2. They have to provide correct, loop-free routes through the network in order that data traffic can flow via these routes from any source host to any reachable destination.

The first task is called data forwarding the second routing, and both are achieved separately by special software algorithms called protocols which are running distributed and

\[1\] When speaking of routers we mean “level three Internet Protocol packet switches” [27]
decentralised on all routers. Currently the most widespread data forwarding protocol deployed in the Internet is the Internet Protocol (IP) [16]. For the routing however, due to its higher complexity in comparison with data forwarding and the different demands on it, there are several protocols in use today, like for example RIPv2 [29], OSPFv2 [27], IS-IS [21], Appletalk [4], BGP4 [1] and others. But although these processes, data forwarding and routing, are executed by separate protocols, they are yet correlated in the way that when doing data forwarding the router tries to send data to a destination by use of information gained from the routing process. For that reason a common interface is used by both which is the *routing or forwarding table*\(^2\). After calculating, the routing protocol stores the best routes to any reachable destination in this table. Those routes or paths consist of a set of path describing attributes, like for instance the address of the next node (next hop) on the path to the destination, the name of the interface card that connects to the link leading to the node and so on. The sets form route entries and they are keyed by the corresponding route’s destination address. So when a router receives a data packet that is not destined for a local attached host, then it’s data forwarding protocol instance (mostly IP) performs a table lookup for the packets destination and hopefully gets back the necessary information to forward the data to the next hop router on the best path towards the receiving host. If so, the packet is sent to that next hop. This process repeats on each router (hop) on the way to the destination until the data packet is delivered to the receiver host.

Although this is an easy mechanism for transporting data from \(A\) to \(B\), it doesn’t scale for some reasons: As mentioned before the Internet is a huge network, containing more than 170 million hosts and it grows constantly. Storing a single route entry for each host on each router, would therefore be impossible to manage efficiently (access times, storage capacities). Moreover routes are not static! Due to its dimensions, changes in the Internet’s structure appear quite often. Hard- and Software used in networks may fail or may be shut down for maintenance or upgrades and then again get back operating. In addition, new hosts, routers, links and other components are added by and by. That leads to a permanently changing network topology and of course changing best paths. To maintain the paths, routing protocol instances must respond quickly to such changes by communicating them to all routers via topology updates and then recalculating the best paths. But a single routing protocol running distributed on all routers, doing the entire Internet’s routing process would be affected by such a high frequency of changes, amount of updates sent in responds to the changes, update propagation times and of course path calculation overhead on the routers, that sensible routing wouldn’t be possible anymore. In addition the Internet is not a homogeneous network under the control of one authoritative administrator. Instead it is composed of many networks of different sizes, owned and administrated by several Internet Service Providers (ISP), universities, private companies, and others. They all have their own idea of directing traffic within their domain

\(^2\)In SSFNet the forwarding table is called IP FIB (IP *Forwarding Information Base*).
2.1. ROUTING IN THE INTERNET

and transiting their domain.

![Simple Networks Diagram](image)

Figure 2.1: Examples for simple networks

So how then is routing in the Internet performed? To answer that question we first have to ask for the structure of the Internet. Today's Internet is often called a network, but it isn't just a set of routers and hosts that are somehow interconnected. Its structure is rather described as a network of networks organised more or less hierarchically. At the bottom, we have small subnetworks (subnets) linking only a few hosts and routers. The smallest possible subnet is a simple link connecting two network nodes, and it's called a point-to-point link (ptp-link) or point-to-point network (ptp-network). When more than two nodes are connected via one common link then it is often done by using a so-called broadcast media like Ethernet. Figure 2.1 shows graphs, sketching examples for both kinds of networks. Circles depict routers or hosts and bold lines represent links. All network nodes are identified by a particular, unique address which must be specified as destination address when data is sent to the node. To simplify matters, we assume here that hosts are the only nodes, data can be sent to. Such host addresses are encoded in a special format which is a 32-bit String divided into two parts. The first part is the network part, which takes usually the first 8, 16 or 24 bits of the String and addresses the network to which the destination host is attached. The rest denotes the host part that addresses the destination host itself relative to the network. Those addresses are commonly known as IP addresses.

Via the features of “subnetting”, that is the division of an IP address' host part into a subnet and a host ID [18] and “classless inter-domain routing” (CIDR), that is the aggregation of multiple IP addresses, sharing a particular portion of high-order bits, into a single IP address [3] [8], a hierarchy of subnets was built, where each of the subnets forms an isolated address domain of a common address space. At the top of this hierarchy there are, so called Autonomous Systems (AS), i.e. large networks under the administration of a single ISP, university or other organisations. These ASes, interconnected mostly by fast, high bandwidth ptp-links, form the Internet. Due to competition between the operators, the

---

3 They differ from non-broadcast media by their special broadcast address. Data sent to this address is delivered to all routers and hosts attached to the broadcast link.
ASes not only represent isolated address domains but also isolated routing domains. That means, routing is performed by different kinds of protocols. On a high level it is realized outside, i.e. between the ASes by an Exterior Gateway Protocol (EGP). We call this kind of routing inter-domain or inter-AS routing. On a local level, various Interior Gateway Protocols (IGP) are deployed to perform the routing process inside the ASes. Routing on IGP level is called intra-domain or intra-AS routing. Via subetting and address aggregation (CIDR) it is possible to describe the whole address space of an AS by a single address prefix matching the addresses of all nodes located within the AS. At the AS’ borders these prefixes are exchanged with neighbouring ASes via so called AS border routers. These routers belong to a particular AS, yet are connected to one or more neighbouring ASes. Thus instead of announcing a single route for each host located in any subnet of the AS, a single prefix is sufficient to describe reachability of all intra-domain locations. Thus the size of routing tables can be reduced by several orders of magnitude and ISPs are able to hide the details of their AS-internal network structure. To propagate those reachability information among the ASes and to provide a prefix based inter-domain routing, a single EGP called Border Gateway Protocol (BGP) is deployed in the Internet. Within an AS, due to the fact that ASes are under the administration of a single organisation, a shortest path routing to single hosts and subnets is usually deployed. To reduce the amount of routing updates, the propagation times and the size of routing tables in intra-domain routing, sometimes more than one IGP is used. In combination with the subetting feature the AS-interior routing domain thus may be subdivided into isolated address and routing domains exchanging aggregated routing information at their borders. On the other hand there are IGPs like the Open Shortest Path First (OSPF) routing protocol, which can be configred to set up isolated routing domains (areas) that are yet under the control of the same IGP. We come back to that in section 2.2.1.

This entire design breaks the problematic down into pieces, simplifies it and enables a relatively stable and efficient routing. Isolated routing domains limit the impact of topology changes to local domains. Address aggregation and prefix based routing saves routing table space. Autonomous Systems hide and protect network structures from competitors and the more or less strict separation of inter and intra-domain routing reduces the amount of routing traffic, update propagation times and finally the routing calculation overhead that is generated when the topology of the network has changed somewhere. The last points are extremely important. The number of updates, the propagation times and the calculation overhead have a big impact on the convergence time of routing protocols. By convergence we denote the process that starts at the time, a change in the network’s topology occurred and ends after the last router in the routing domain detected the change, calculated the new best paths and applied the changes to any forwarding table. We call the time period where the network passes through that process the convergence time or convergence phase. So when a routing protocol converges, it is in the process of updating its view of the network’s topology and calculating the new best paths. But what exactly is a “best path” or “best route”? There is no precise answer to this question since there
are different demands on routing protocols and thus on the paths. We will deal with that in more detail in the next section where we describe how intra and inter-domain routing is accomplished using OSPF and BGP. In this section we present the basic knowledge that is needed to understand the interactions between an IGP like OSPF and the inter-AS routing protocol BGP. The core of this interaction is based on the BGP routing strategies *hot potato routing* [14] and *cold potato routing*. These strategies both tie BGP to the AS-interior IGPs in the way that BGP selects, among multiple “equally good” routes to a destination prefix, the one having the shortest distance in the sense of the IGPs in the respective AS\(^4\). By an increased utilisation, these strategies become a powerful tool to reduce data traffic load within Autonomous Systems, since data traffic now takes the shortest paths through the relative ASes. But it also leads to otherwise isolated intra-domain routing information infiltrating into the inter-AS routing domain. With it, local intra-domain routing instabilities can get globally visible [23]. This again causes a higher amount of routing update traffic, longer propagation times, more calculation overhead and therefore longer and more frequent convergence phases. At the end a high measure of instability in the global routing process may result from that, causing losses in data traffic and increased network latency [17].

## 2.2 Deployed Internet Routing Protocols

We yet addressed before that the Internet is subdivided into ASs and that routing is done more or less separately outside and inside of those ASes. But due to the increased utilisation of hot and cold potato routing strategies, dynamics between BGP and IGPs can arise. To closer examine these dynamics we have to understand the functionality of the involved protocols first to get the basic knowledge about their character. Our work has concentrated on OSPF-BGP interactions, i.e. the impact that OSPF routing information might have on BGP. Therefore we are going to give a short outline about bodywork and functionality of those protocols in this section. In case of OSPF we restrict our descriptions to the components that contribute to convergence time and the background to understand these components. For a detailed specification of the OSPF architecture please see [24] and [27]. Concerning BGP we only give an abstract, since we are going to study the impact that OSPF has on BGP and not BGP itself or BGP-OSPF interactions, i.e. the impact that BGP information may have on OSPF. For more information about BGP see [1] and [7].

\(^{4}\)An exact description of hot and cold potato routing will be given later on when we have introduced the necessary terms.
2.2.1 An Introduction to OSPF

In today's Internet, there are two general categories of Interior Gateway Protocols. One differentiates between distance-vector\(^5\) and link-state routing protocols. OSPF is a link-state routing protocol and was expressly designed for the TCP/IP protocol stack [24]. Running distributed, it exchanges routing information between the routers of its routing domain. To transport this information, OSPF makes use of IP as “transport” protocol. Since IP's forwarding is based on OSPF's routing, OSPF only sends data to routers that are known before or detected via its Hello Protocol (see next paragraph). The routing information again consists of small pieces of data, describing the current state of particular network parts. On the one hand this includes the local environment of a router on which OSPF is running, i.e. the state of its locally attached links. This is where the name link-state comes from. On the other hand that might also include subnet descriptions learned from other areas within the routing domain (see paragraph Hierarchical Routing), culminating in information about prefixes learned from other routing protocols. We denote these pieces of routing information as Link State Advertisements (LSA). Each router generates at least one LSA, a router-LSA that describes the state of its locally attached links. OSPF stores LSAs in so-called Link-State Databases (LS Databases).

Via LSA distribution throughout the routing domain\(^6\) each router is able to accumulate LSAs generated by other routers in its LS Database. As it is the case with a puzzle, OSPF then assembles a network topology from the information gained from the installed LSAs. Figure 2.2 shows a sample topology composed of four routers all supposed to run OSPF. The left side shows the situation when OSPF was just started. Annotated to each router is the figuratively represented, individual router-LSA this router contributes by describing its local environment. In the further process the routers try to establish bidirectional communication with their directly attached neighbours. If bidirectional communication could be established, the LSAs stored in the respective LS Databases are exchanged. When a new LSA or a new instance of an already known LSA, a so-called LSA update is received from a neighbour, it is propagated to the other neighbours. This mechanism ensures that each router receives a copy of this LSA. At the end all routers have stored all available LSAs in their LS Databases, which means that the contents are exactly the same. When this state is reached the databases are said to be synchronised. The right picture of figure 2.2 shows the situation when all LSAs are exchanged. Annotated to each router is the illustrated content of its synchronised database. We assume here that bidirectional communication could be established between all pairs of neighbouring routers.

Now that each router has a consistent view of the network topology, it is able to individually compute best paths to each reachable destination (i.e. known by an LSA). As it is denoted by the name, OSPF is a shortest path routing protocol. That means, in OSPF terms a best path is a path that has the shortest distance to its destination. To determine the

---

\(^5\)An example for a distance-vector protocol is RIP. A specification of RIP can be found in [29]

\(^6\)Some LSAs are subjected to restrictions concerning distribution. See [27].
2.2. DEPLOYED INTERNET ROUTING PROTOCOLS

Figure 2.2: Example topology with annotated LS Database contents

distance, each link being part of the OSPF routing domain is assigned a particular metric. We will also refer to as cost or weight. These link metrics are positive natural numbers assigned by the network administrators\(^7\). Usually their values are geared to the link delays or transmission times [24]. OSPF now defines the distance of a path, sometimes also called the path length, as its metric. Since a path is a series of links leading from a source to a destination, its according path metric is the sum of the metrics of all of its composing links. Therefore the shortest path to a destination is the path with the shortest path metric. OSPF calculates loop-free, shortest paths using Dijkstra’s Shortest Path First (SPF) algorithm [19]. Setting itself as root node and running the SPF algorithm on the topology graph provided by the LSAs in the database, each router’s OSPF session computes an individual Shortest Path Tree (SPT) that provides the desired routes to all reachable destinations. Figure 2.3 shows, on the left side, the sample topology of figure 2.2 with annotated link weights. The right side depicts the resulting shortest path graph that represents the individual SPT of each router. From the shortest paths, the next hop routers on the way to the destinations and the appropriate interface cards to forward data to those next hops (forwarding interface) can be derived. Together with some other attributes, route entries are built from these information and the IP forwarding table is updated accordingly.

Since each router computes its routes in the same manner based on the same data, all routers have a consistent view of reachable destinations and the proper routes. Thus data traffic follows paths that are exactly predetermined from the first hop on. To guarantee that this is the case indeed, two things must be assured:

- Link-State information gained from the LSAs must be kept up to date.
- Link-State Databases must be kept synchronised.

\(^7\)Actually metrics are not assigned to links but to interfaces connecting the routers to the links. This is done, to allow for asymmetric paths, but since we don’t work with asymmetric paths in this thesis, we may simplify the matter
Otherwise it could not be ensured that sensible routing tables are computed. For this reason, OSPF is equipped with mechanisms to establish and maintain bidirectional connections with other OSPF speaking routers and to reliably distribute LSAs via those connections. Given that these mechanisms make substantial contributions to convergence time, we are going to discuss them in more detail in the following paragraphs.

**The OSPF Hello Protocol**

To send data reliably from a network node A to a network node B, a bidirectional connection between A and B is required firstly, so that A’s data may reach B and B’s acknowledgement for the received data may reach A. As we suggested before, there is no direct communication between all OSPF instances in the routing domain. LSAs are rather propagated from router to router. Therefore OSPF reduces the problem of bidirectional communication to directly linked, neighbouring routers. Consequently, two arbitrary OSPF routers have a bidirectional connection, if all links composing this connection are bidirectional. The mechanism that establishes and maintains bidirectional connectivity between neighbours is the OSPF Hello Protocol. Via the router’s interface cards it periodically sends out special OSPF packets, the hello-packets (hellos). The interval at which such packets are sent out is referred to as hello-interval and its default value is 10 seconds, according to [27]. If a router receives a hello from a neighbour, then it knows that there is an, at least unidirectional connection to this neighbour via the receiving interface. To make sure that this connection is bidirectional too, a hello carries with it a list on which each neighbour is recorded of whom a hello was heard recently. Thus dependent on the interface on which it is sent out, the hello-packet carries the identifiers of all neighbours whose hellos have been seen recently on that interface. If a router receives a hello from a neighbour and sees its own identifier on the list, then it knows that there is a bidirectional connection to that neighbour. This mechanism is part of a state machine that determines the state of the relationship to each discovered, neighbouring router. It’s called, the Neighbour State Machine (NSM). Figure 2.4 shows an excerpt of the NSM, confining on the
part of the Hello Protocol. Each state of this automaton represents a particular state in a neighbour relationship. The states are computed individually for each neighbour to keep track about the connection and exchanged routing data. They are recorded among other information in Neighbour Data Structures [27]. Each time an OSPF router detects a new neighbour on one of its interface cards, i.e. the router received a hello from it via that interface for the first time, a new Neighbour Data Structure is set up. If the router couldn’t find itself on the neighbour list of a hello (1-way-received), the state of the relationship is turned to “Init” (initial), meaning that the connection is supposed to be unidirectional. Similarly transition “2-way-received” is traversed when the connection was found to be bidirectional as defined before. In state “Init” or higher, the router adds the neighbour’s identifier to the neighbour list of its hellos sent out the interface. In state “Down” the connection to the neighbour is supposed to be broken off completely and the identifier is removed from the list.

Because hellos are sent and thus received periodically, it is easy to determine whether a connection is still bidirectional, went down from a higher state to unidirectional (Init) or again became bidirectional. But when is a neighbour relationship declared to be “Down”? OSPF estimates that time by assigning an inactivity-timer to each neighbour relationship. When the first hello from the neighbour was received, the timer is started. It is restarted every time a new hello was heard of it. If it expires, no hello has been received for a period of time and the connection is declared to be “Down”. This period is called the router-dead-interval or simply dead-interval. Usually it is set against the hello-interval, that is three or four times larger than it, taking into account that packet losses may occur. The default value, according to [27] is 40 seconds.

Reliable Flooding

Having detected a change in the network’s topology, for example a failing link or a new router joining the routing domain, the detecting OSPF routers generate new LSAs describing the changed topology part and begin with the distribution of these LSAs to the other
OSPF routers in the routing domain. As mentioned before, OSPF uses IP as transport protocol to exchange routing information. Since IP is a best effort data forwarding protocol, without guaranteeing correct data transmission, OSPF must implement an adequate mechanism to assure reliable transportation. This mechanism is called reliable flooding process or just flooding. In this process each router, that received a new LSA or LSA update, is responsible for transmitting this LSA reliable to all of its adjacent neighbours. A neighbouring OSPF router is called adjacent, if it is directly attached via a single link and the relationship to this router is in a particular state. In this state, the connecting link was found to be bidirectional and it is allowed to exchange LSAs with the neighbour\(^8\). With this mechanism it is ensured that LSAs are propagating or flooding router by router though the routing domain. To get a reliable flooding, the transmission of LSAs between two adjacent OSPF neighbours must be made reliable. OSPF provides two kinds of packets for that purpose. In a link-state update packet (LS Update) the LSAs to be flooded are stored and sent out to the adjacent neighbour. Via a link-state Acknowledgement packet (LS Ack) the receiver sends back acknowledgements for each received, non-corrupted LSA. If the sender receives an LS Ack, it extracts the acknowledgements and checks which of the LSAs sent in the last LS Update have been acknowledged. LSAs that were not acknowledged must be retransmitted. For that reason the sender provides a link-state retransmission list for each Neighbour Data Structure and a retransmission timer. On the link-state retransmission list it keeps all LSAs that must be transmitted and are not yet acknowledged. In case an LSA was acknowledged, it is removed from that list. The retransmission timer ensures that the remaining LSAs are retransmitted at constant intervals until the list is empty. The interval at which LSAs are retransmitted is determined by the configurable rxmt-interval parameter and has a default value of 5 seconds according to [27]. Acknowledgements are sent back almost immediately after receipt of the LSA. To avoid sending to much LS Acks, acknowledgements are sometimes delayed, such that they can be bundled in a single LS Ack. The delay time is referred to as delayed-ack-time and should be smaller than rxmt-interval to prevent unnecessary retransmissions. Via the reliable flooding process, OSPF exchanges and distributes almost all of its LSAs. An exception is the process when exchanging LS Databases between two routers that are establishing an adjacency. This process is more or less uncoupled from reliable flooding and uses additional packets like Database Description and LS Request packets. Its mechanism of reliable data exchange is however similar.

### Hierarchical Routing

We mentioned before that dividing routing in isolated routing domains is a good strategy to hide parts of the network’s infrastructure, to reduce the memory consumption of routing tables and to limit the impact of topology changes, like the link-bandwidth consumption

\(^8\)For a detailed explanation when which router may become or is adjacent please see [27]
by routing updates and the routing table calculation overhead on the routers. Within an AS the need for hiding the network structure might be not so high since an AS is usually under the control of a single network administrator. The containment of resource requirements however is of great interest, since ASes usually comprise large networks. OSPF is an IGP that is capable to reduce its resource requirements by subdividing its routing domain into so called areas. Those areas are relatively isolated routing domains running a separated copy of the basic OSPF routing algorithm. That means that each router provides a separate link-state database for each area it is connected to and also calculates separate SPTs for its areas. The separated routing information is exchanged between areas via a special backbone area. To do so, all areas have to be somehow connected to the backbone area, physically or virtually by so called area border routers (ABR). The ABRs usually aggregate routing information into IP address prefixes before forwarding it into the backbone. Thus similar to BGP, routing between areas is mostly prefix based. The backbone again makes these prefixes available to the other areas. In so doing, the backbone area is determined to provide transit for data traffic originated in any non-backbone area, destined for any other. To be able to direct data traffic to destinations outside its routing domain, OSPF can import routing information. Special OSPF routers, the AS boundary routers (ASBR) learn address prefixes from other routing protocols, like for instance BGP, and announce them in the OSPF routing domain. To protect against a flood of external destination prefixes, inflating the routing tables, OSPF may be configured to declare areas to be “stub”. Such “stub areas” refuse all routing information coming from outside the AS and instead rely on default routes leading to routers which know about the destination. This sophisticated routing hierarchy enables a resource-saving but still efficient IGP routing. Moreover the principle of a backbone area was adopted by many network operators when they were organising their ASes [22]. Usually an AS is designed hierarchically. At the top there is a backbone network consisting of powerful routers very densely interconnected via high-speed and high-bandwidth links, connecting to other ASes and providing transit for all kinds of traffic (intra-AS, intra-to-inter-AS and also inter-AS). Beneath that level, attached to the backbone sub-networks connecting to customers are located. It is self-evident that routing within those subnets is usually somehow separated from the backbone. We come back to the topological aspects in chapter 3.2 where we describe our design decisions for our simulated network environments.

2.2.2 An Introduction to BGP

The Border Gateway Protocol is the de facto standard inter-Autonomous-System routing protocol deployed in todays Internet. Its primary function is the exchange of AS reachability information with other BGP running systems (BGP speakers). We call a unit of this information a route or route information. It consists of a destination prefix, aggregating the address domain of an AS, and a number of path attributes describing the path to that
destination. By collecting those routes and applying some filtering rules \((\text{policies})\), BGP constructs a loop-free graph of AS connectivity. Similar to OSPF, BGP provides mechanisms for establishing and maintaining connections between BGP speakers, exchanging routing information databases, detecting errors in connections and other requirements to accomplish its task. But in contrast to OSPF, BGP was not developed to do shortest path routing in first place, although it is capable to do that. It was rather designed to be able to direct traffic according to the needs and policies of the network operators. A best path in BGP is therefore not inevitably a shortest path. Its selection usually depends on many of its path attributes, not only on its path length. When choosing among several routes having in common a particular destination prefix, the path attributes of these routes are successively compared with each other. The route whose path attributes fulfil best certain criteria, becomes the preferred best route. We will come back to the selection process later on when we have introduced some of the path attributes. We refer to it as the BGP best path selection algorithm.

As mentioned before BGP exchanges routes between Autonomous Systems. To do so, it forms peering sessions to other BGP speakers, similar to the OSPF adjacencies. In a peering session, two BGP speakers, the peers, exchange routing information via a common connection. Peers don’t have to be directly connected. To transmit data they rely on local routing and the Internet’s popular reliable data transport protocol TCP \([37]\). Thus BGP can dispense with any precautions against data loss and corruption. If an AS has multiple BGP speakers having peering sessions \((\text{peering})\) with BGP speakers located in different ASes, thereby making available transit service for other ASes, a consistent view of the BGP routes within that AS must be ensured. This is usually done by establishing peering sessions between each pair of AS-interior BGP speakers and configuring a common set of policies within the AS. In addition, when transit service for other ASes is provided, particular care must be taken to ensure that the AS-interior routers are aware of the routes, to which transit is enabled. A simple but not so efficient method to guarantee this, is to set up BGP speakers on each router.

When a BGP speaker receives routes from its peers, it usually filters these routes before they are stored in \textit{Routing Information Bases} (RIB) and further propagated to other peers. The filtering is done, because routing on inter-domain level depends strongly on business agreements made between the organisations which administer the ASes. Using special filtering policies, configured individually on the BGP speaking systems, it is ensured that only those routes participate in the routing process that do not breach any agreement when data traffic is directed via these routes. After filtering, the remaining routes are deposited in RIBs called Adj-RIBs-In. Thereupon, for each accepted destination prefix, a single route is selected from the Adj-RIBs-In via the BGP best path selection algorithm and installed in the local RIB \((\text{Loc-RIB})\). Each BGP speaker has exactly one Loc-RIB. Routes stored here are also installed in the IP forwarding table to be used in the forwarding process and may be propagated to other BGP peers. The propagation depends again on filtering policies. A BGP speaker therefore maintains for each peer an Adj-RIB-Out
2.2. DEPLOYED INTERNET ROUTING PROTOCOLS

where it stores the individually filtered Loc-RIB routes which are to be announced to that peer. Concerning an interaction between OSPF and BGP, the interesting part here plays the BGP best path selection algorithm, since the selection not depends on locally configured policies but on path attributes that may be set according to IGP path metrics. In the following we numerate some important BGP path attributes, which we refer to in this thesis and describe their meaning and use:

- **AS_PATH**: This attribute consists of a sequence of AS identifiers (AS numbers) indicating the Autonomous Systems that the according route information has traversed so far. Since BGP is an inter-AS routing protocol, a path is described by a sequence of ASes leading from the source AS to the destination AS. A BGP route’s path length is therefore defined as the length of the AS_PATH attribute. If a BGP speaker advertises a route to a peer located in another AS, the speaker modifies the attribute by adding the AS number of the AS it resides.

- **NEXT_HOP**: A BGP speaker that advertises a route to one of its peers, informs that peer via this attribute about the IP address of the next hop (next router) where data is to be sent to when destined for the route’s destination. Often this attribute is set to the advertiser’s own address [7]. But it may be also the case that it is modified only when advertised to peers located outside the own AS [1].

- **MED**: The MULTI_EXIT_DISCRIMINATOR (MED) attribute is a metric that is set on border routers, i.e. BGP routers having a direct link to a peer located in a neighbouring AS. In case there are two neighbouring ASes connected via multiple exit and entry border routers, each border router may assign a different MED to a certain route that is advertised to the neighbouring AS. The BGP speakers in the neighbouring AS may then discriminate between the different exit and entry points by choosing the route leading via the border router that assigned the lowest MED. The MED attribute may be propagated throughout the neighbouring AS but not beyond it.

- **LOCAL_PREF**: The LOCAL_PREFERENCE attribute is a metric like the MED. It is individually computed on each BGP speaker for each route that it advertises to a peer located in the same AS. The attribute is intended to direct inter-AS traffic within an AS. If it is received from peers outside the own AS, it is ignored.

There are yet lots of other path attributes. Some of them are specified in Requests for Comments, like for example in [1] and [6]. Others are individually added in BGP implementations [5] by vendors. Which of the attributes are finally used in the BGP best path selection algorithm depends on the implementations.

In general the algorithm consists of a list of rules called tie-breaking rules or tie-breakers which are applied to a set of routes describing different paths to a common destination.
Each rule selects the route that fulfils best its criteria. If more than one route do so, the next rule in the list is applied until there is only one route left. The number of rules and the order in which they are applied is vendor specific. A common proceeding is to firstly compare the LOCAL_PREF attributes. The route having the highest degree of preference is selected. If this rule could not select a unique route the algorithm continues. Although the AS_PATH attribute is not specified in [1] to be used in the BGP best path selection algorithm, it is however common practice to define a shortest AS_PATH as tie-breaker. Determined in [1] are rules which selects routes having the lowest MED and the shortest IGP path (lowest IGP metric) to the next hop router, if their NEXT_HOP attributes differ. They are in any case inherent parts of all BGP4 conform implementations. To guarantee that the algorithm effectively selects a single best route, the last rule usually selects the route being advertised by the BGP speaker with the lowest BGP identifier. Since these identifiers are unique, a successful termination is ensured.

The individual implementation of the whole best path selection process is vendor specific. Therefore we cannot give an exact description of it. We will however describe in chapter 5.2 the implementation that we used in our OSPF-BGP interaction studies.

### 2.3 Routing Simulations

In this work we want to understand and analyse certain aspects of routing interactions between intra and inter-domain routing protocols. The core of this interaction are the “hot” and “cold potato” routing strategies. In the last years, Internet Service Providers (ISP) began to increase the utilisation of these strategies by an increased incorporation of intra-AS shortest path metrics in the inter-AS routing process. Before, the BGP best path selection algorithm made use of IGP shortest path metrics only within an AS. A single criterion, having a relative low rating in the list of tie-breakers, preferred the route with the shortest IGP path to the route’s next hop. Its application was triggered by route updates only and not when the IGP path metric changed. When a route is selected according to this criterion, we call this hot potato routing. Nowadays some ISPs extended this criterion to the BGP selection processes outside of their ASes. Routes that were learned via an AS-interior border router from a neighbouring AS, get their MED mapped on the metric of the IGP shortest path to that border router when they are further announced to the next neighbouring AS [23]. This resulted in neighbouring ASes, selecting routes based on the shortest IGP path through the announcing AS, when the selection of these routes was actually based on the MED attribute value. This kind of route selection is usually called cold potato routing. We will deal with the consequences resulting from these strategies in more detail in chapter 5.1.

[23] analysed the exchange of BGP routing messages, logged over a period of more than 28 months at five major U.S. exchange points, to find out among many other things about this kind of interactions. When we are going to study the impacts of such interactions,
their different causales and effects, we need access to a large network that provides intra as well as inter-domain routing. Instead of collecting data at a few points, we need the total control over this network to observe the effects in all details and to generate individual topology changes, resulting in the respective routing changes. This includes also the configuration of the protocols.

Since we neither have the equipment for building such a network and setting up an adequate configuration, nor do we have access to the facilities of an ISP, the only way to realize our studies was to simulate a network environment that provided all requirements, necessary to generate representative interaction conditions. With SSFNet, a collection of Network Models written in Java and built on Renesys [30] Scalable Simulation Framework (SSF) [31], we found a powerful network simulator that provides all basics to build up network scenarios meeting our demands. Based on a discrete-event simulator, the SSF, those models provide implementations of several protocols used in the Internet like IP, TCP, UDP, OSPF, BGP, HTTP and others. SSFNet consists of several packages. The most important are SSF.OS which contains basic classes defining the framework that any protocol model must concur with, and SSF.Net which implements the “Hardware” environment like classes for routers and hosts, Network Interface Cards (NIC) and link media.

On top of the SSF.OS package, the Internet protocol models are built, such as SSF.OS.IP, SSF.OS.sOSPF and SSF.OS.BGP4. With SSF.OS.BGP4, a very detailed model of the Border Gateway Protocol, implemented according to [1], we had the functionality of the Internet’s de facto standard inter-domain routing protocol. SSF.OS.sOSPF on the other hand, a quite limited, static version of the OSPFv2 intra-domain routing protocol, decided the issue of selecting an IGP. We yet could not use these protocol models as they were. SSF.OS.sOSPF did not meet our demands and SSF.OS.BGP4 did not at all use IGP metrics in its best path selection process. A recreation of OSPF, leading to the new package SSF.OS.OSPFv2 (see chapter 4.1) was therefore necessary. Concerning BGP, we had to apply only minor changes (see chapter 5.2).

SSFNet’s OSPFv2 and BGP4 both implement their own routing tables where they store their individual routing information. A quintessence of these information is finally installed in the IP Forwarding Information Base (IP FIB), an out-of-core IP forwarding table for common access by the routing protocols. This enables a clean and clear exchange of routing information between routing protocols. We use the FIB, in order to grant BGP4 access to the IGP metrics of the routes, OSPFv2 stored there.

Now that we have presented the background to our work and an introduction to our methodology and the tools, we can start with our studies. But before creating OSPF and BGP scenarios in SSFNet, we have to answer a particular question. The question for an adequate network topology.
Chapter 3

Topologies

When simulating the Internet, or even when simulating parts of it subjected to certain aspects as we are going to do, one performs a complex task. To get representative simulation results, a simulation environment is required that is most typical for the component of which behaviour is examined. We therefore need to discuss, what parts of an environment we have to consider in particular and which are negligible. Due to the heavy heterogeneity of the Internet, there are many attributes that may affect the behaviour of single Internet components and so the simulation results. [10] gives elaborate food for thoughts which details one has to pay particular attention to when dealing with simulations. There are, for example, the wide range of link technologies that are deployed in todays Internet, the interplay of protocols on multiple layers, the contribution to the amount of traffic that different applications make, the varying capacities of routers, the adaptive congestion control mechanisms and much more.

Since we are dealing with interactions between routing protocols, where we don’t need to generate a great amount of data traffic, we may refrain from all traffic relevant aspects. In addition, since we are interested in behaviour and convergence times of routing protocols, we can also refrain from a complex link layer and instead make allowances for this fact by reducing the complexity of its impact to a configured link delay. The only point that remains and which has a great impact on our work is the arrangement of routers and links in the deployed network, that is the network’s topology. So before we may begin with routing studies, we need to find at first representative topologies that meet our requirements without being too complex to simulate and analyse. On the one hand that should be simple topologies that trigger protocol features which have an impact on particular routing aspects which play an important role in the OSPF-BGP interactions, like for example OSPF’s convergence time. On the other hand, to analyse the interactions we need an AS-level topology that is complex enough to watch OSPF routing changes, but abstract enough to follow possible BGP updates and typical enough for the real Internet to be representative.
CHAPTER 3. TOPOLOGIES

3.1 What is a “typical” topology?

Following [10] it is hard to achieve an adequate topology when simulating the Internet on AS-level. We are even far away from a definition of a “typical” topology. The reasons for that can be found in the continuous expansion of the Internet over time, which results in constantly changing and growing networks [15]. But even if we limited our studies to a certain network size on a single point of time, we had to find a common used scheme that gives us an idea how to construct our topology. Yet the fact is, that for competitive reasons the most ISPs don’t share insights into the structure of their ASes. We are therefore strongly dependent on routing protocol observations to get data which can be used to model networks. The Oregon route server [38] for example, collects BGP routing tables which are used in many studies about modelling the Internet [34]. Via tracerouting, directed probing, path reductions and other techniques [22] one can derive some degree of network connectivity on AS-level as well as AS interior. But following [34] many Internet models designed on the basis of such collected data have turned out to be incomplete. They argue that, due to the capability of policy based routing, BGP forwards “only a portion of all existing AS connections” from AS to AS.

Although routing protocol measurements are quite a good tool to get the general idea of the Internet’s topology, we finally come to the conclusion that there is no chance to generate an exact image of an adequate sized part of the Internet to study OSPF-BGP interactions. There are too many factors coming into play when modelling a topology. [32] assume that expansion and shape of networks depends much on geographical environments and business interests. When regarding the history of the Internet then this is not surprising. Born in the middle of the 1960ies, when the cold war drove the fear of a nuclear attack to its peak, it was at first just a project of the U.S. Air Force. The idea was to install a packet switched data network among the missile and bomber bases so that command and control could be maintained even if the central of the country was destroyed by a nuclear strike. If this case arised, it should be possible via redundant links to keep up communication between the west and east cost. Eventually in 1969 the idea was realized by the ARPANET network. Figure 3.1 shows the ARPANET’s topology in September 1971. The shape of this early ancestor of todays Internet was strongly dependent on the location of some of the most prestigious universities in the U.S. and by the request of a redundant east-west connection. This example supports the assumption that geographical aspects play a role in modelling topologies.

In the early 1990ies, when the HTTP protocol came up and established the World Wide Web, the actual Internet was born. Telephone companies became providers for Internet access and began to install appropriate hardware in their telephone networks. During the 90ies more and more commercial ISPs came into being and Internet topologies were adopted from telephone networks. Smaller ISPs rented equipment from bigger ISPs whereas big, long-established telephone companies restructured and augmented their networks according to business interests. Placing network nodes and links became dependent
3.1. WHAT IS A ‘TYPICAL’ TOPOLOGY?

Figure 3.1: The ARPANET topology in September 1971

on proximity to customers, the technological feasibility and of course money! But besides geographical and business conditions, the social, technological and political circumstances in the individual countries had also a big influence on the growth and density of their topological contributions to the Internet. The University of Wisconsin-Madison provides a nice history of development in form of world maps documenting international connectivity from September 1991 until June 1997 [12]. While in 1991 the Internet was a phenomena that appeared almost only in technological high advanced countries with stable and free elected governments, it arrived over the years at less technological developed states until it reached even most of the less-developed countries. Although you can’t derive from those maps how and how far connectivity has progressed in the individual countries and whether a wide class of population gets access to the media, it is remarkable that countries like Sahara, Libya, Syria, Iraq, Afghanistan, North Korea and Somalia have refused the Internet all the time [13]. Because the actual stage of technology in these countries was very low at that time or their government was quite restrictive concerning freedom of opinion, this is an evidence for the supposition that formation of topologies is also dependent on political stability and freedom and of the technical and financial situation within a particular region.

Regarding all those influences, we are forced to accept that there is no typical topology. Hence we need to approach the problem in another way.
3.2 Approximation

The Internet is a world wide phenomena. Almost every country in the world contributes at least a few hosts, routers and links to its shape. The number of those network nodes and links, their types and locations depend much on the cultural, technological, financial and political conditions of those countries. We also mentioned that due to the active competition between the companies engineering the Internet’s topology, insights into their strategies how to build up network topologies or even information about existing topologies are mostly refused. The rapid growth and changes over short time come along yet too. Since all these facts making it impossible to pick out “the right” topology, we have to decide how much realistic our topology has to be. That raises the question for our demands and which topology is simple but sufficient enough to meet these demands. The question to answer we are most interested in is, how and how much does OSPF impact BGP? We are not going to show how an interaction between these protocols may be improved nor do we develop a future routing protocol. We therefore don’t want to test special features that require special topologies. Since we do not make architectural changes to the protocols, the only way for OSPF and BGP to interact is via changes in the IP forwarding table. And because the OSPF Hello Protocol detects changes in routing via very small messages, data traffic has a negligible impact on our studies so that our topology must not be designed according to special traffic aspects like for example bottleneck links, size of router queues or even least-cost-multi-pathing and asymmetric routes.

One of OSPF’s key features is its support for hierarchical routing (see chapter 2.2.1). AS topologies may be subdivided into a backbone area and several other areas all doing more or less separated routing, but under control of a single OSPF instance. Since the backbone area was designed to provide data transit traffic and to be the only one responsible for distributing routing information within an OSPF routing domain, routers importing AS-external routing information (AS boundary routers) are usually placed here. Because of that and due to the fact that routing information is aggregated and sometimes even refused at area borders (stub areas), BGP traffic is usually observed in backbone areas only, when passing an AS. Therefore routing changes in OSPF areas other than the backbone actually don’t affect BGP and vice versa, and we gain the advantage that we might concentrate our work on AS backbone areas. These areas consist mostly of high-bandwidth and high-speed ptp-links to give good performance and usually don’t deploy broadcast media. For that reason and for the fact that OSPF and BGP reacts on topology changes in magnitudes of seconds we may abstain from exact link delays, since really big delays were observed at connections to satellites in geosynchronous orbits and had orders of hundreds of milliseconds [10]. Nevertheless we incorporate link delays in our OSPF-BGP scenarios.

According to [10], “for some topics [...] the simplest scenario that illustrates the underlying principles is often the best” and so our first topology will be the one depicted in figure 3.2. It consists of five routers connected in mesh form via ptp-links. We will use this
3.2. APPROXIMATION

Figure 3.2: Simple topology for studying parametrical impacts on OSPF convergence
topology in chapter 4.2 to study parametrical impacts on OSPF convergence time. The
network was composed in a way that on the one hand it clearly shows these impacts and
on the other hand it remains simple. We will yet return to this.
On the other hand, we can’t ignore the factor topology completely. Since we are deal-
ing with routing protocols, and taking into account what we mentioned concerning our
demands, network size (i.e. the number of routers and links) and network structure (i.e.
how routers are interconnected) play an important role. To get a feeling for what mag-
nitude these factors might affect OSPF convergence time we are going to regard extreme
cases. We analyse networks with extreme low redundancy in connectivity and extreme
long paths\(^1\). On the other hand we compare them with networks that have high re-
dundant connectivity along with short paths. Examples for the former case are networks
consisting of in-line connected routers like chains or rings. For the latter case we use
fully meshed networks. Figure 3.3 shows examples for ring and fully mesh topologies.
The results we obtain from these scenarios will give us an opinion between which values
the overall OSPF convergence time in a network may range. We will substantiate these
understandings in chapter 5.3.2, where we compare them with OSPF convergence time
measurements from simulations made in a more complex and “realistic” topology. These
measurements then should fit somewhere in the range of convergence times determined
before. With this procedure we follow an advise from [10], which recommends to show
that results gained from simple scenarios still apply when using complex scenarios. The
“realistic” topology is characterised by the fact that it is not composed following special
demands like the simple structured but extreme topologies accounted before. Instead it is
generated in a more random fashion although we shaped it following information about
some U.S. ISP’s backbone networks. In the following we are going to call it the “US
scenario”.

Having studied OSPF’s routing behaviour in more detail we will be able to understand
possible changes in BGP routing triggered by OSPF and may start with simulating those

\(^1\)Path length is usually measured via a hop count. In this case a hop is an intermediate router on the way
from a source to a destination.
interactions. This brings us back to the question for an appropriate topology for these studies. So far we have seen that we don’t need a special purpose topology showing particular aspects or new features of routing protocols. For reasons described before we may also restrict our topology to a backbone network. Future work may show whether more detailed networks produce further aspects in interactions, we didn’t deal with. Since we cannot replicate existing backbone networks of big ISP’s like AT&T or Sprint, due to complexity and missing data, we decided to emulate one of those backbone networks to approximate the reality. From information gained from [22] we finally created our “US scenario” depicted in figure 3.4. In the middle you see a backbone network which is a simplified adaption of ISP backbones showed in [22] figure 7. Characteristical for those topologies is that they deploy POP networks in major cities (in parts in smaller cities too), since there they establish the connections to their customers. The POPs are of different sizes depending probably on several factors at their locations like the number of customers, the companies headquarters, economic shares, etc. A generic POP consists of a few routers interconnected via a dense mesh of links. We created twelve of those POP networks in our backbone demarcated by dashed circles from the rest of the network. They consist of two to four routers fully meshed among each other with ptp-links. These formations symbolise the locations of major cities in the U.S. Similar to the Sprint backbone described in [22], we connected these “cities” well with each other via the blue coloured “inter-city” ptp-links (40-60). Together with the POPs they form an AS, which we will call the “US AS” or AS 1. There we are going to simulate the OSPF-BGP interactions as well as to measure OSPF convergence times which help us by the interpretation of our collected data.

To study possible impacts from those interactions on other ASes we attached several, so called “dummy ASs” \(^2\) (AS 2 till AS 7) to the “US AS” via multiple ptp-links (thick red lines, numbered 61-76), consisting of one router only. Green lines indicate their borders.

\(^2\)“Dummy AS” is not an official term. We use it to indicate that these ASs are only used to form a BGP topology
The dummy ASes are not connected among each other since we don’t examine propagation effects on AS-level. For that we had to create several ASes of similar complexity than the “US AS”. This is left for future work. However the “US AS” together with the dummy ASes build a BGP routed network. Within the “US AS” each router runs OSPF to do intra-AS routing as well as BGP to provide each router with external BGP routing informations. How the protocols and the network’s infrastructure are set up in particular we discuss in chapter 5.3, where we describe and analyse the interactions of the routing protocols.

Figure 3.4: The US scenario
Chapter 4

Analysing OSPF

In the last chapters we outlined open questions in routing and mentioned that interactions between intra and inter-domain routing protocols may play a major role when providing answers to these questions. We also described the background of the fields we are going to research into and we gave an introduction to the topologies that we use when doing so. Now we begin with our actual work, that is we examine the effects that local routing changes may have on global routing. When speaking of local routing changes we mean changes that are detected and propagated by IGP's like OSPF. In this chapter we will deal with those IGP factors that have a major impact on its interaction with the inter-domain routing protocol BGP. The IGP we examine is OSPF and we already gave a description of its functionality in chapter 2. Here we like to provide answers to the question in what orders of magnitude particular OSPF factors might be. We thereby concentrated our investigations on two of them, the processor load that OSPF may cause during a phase of convergence and the convergence time. Concerning processor load, we can’t determine exact values since hardware is changing rapidly and we don’t have access to all kinds of route processor or at least to the most widespread ones. Therefore we need to abstract and to measure load in another way. According to [11] the factor that is most time-consuming when doing OSPF computations is an update of the Forwarding Information Base (IP forwarding table). Preceding to the update is always an SPF calculation which can be as time-consuming as a routing table update, depending on the number of nodes in the network [35]. Both factors are independent from the protocol design since they are principle tasks of routing. For these reasons the number of routing table changes is a good unit to measure processor load generated by OSPF.

In chapter 2 we briefly introduced the term convergence time. Since this term is not exactly defined, we need to provide our understanding of it. For the OSPF measurements in the course of this paper we define convergence time as:

The time period, beginning at a network component’s state change that changes the network’s topology and ending at the time where all routers have realized
the change, computed the new shortest paths and got a stable view of the new network’s topology.

In our simulations we measured this time period by observing the OSPF routing table changes. The convergence time was therefore determined by the difference of the time, the last routing table change on an OSPF router was seen and the time, where a state change in the network took place.

But why do we analyse OSPF in respect of these factors, and why do we measure them in this way? Since local routing changes can have an impact on higher routing levels we need to know when these local changes may be realized in order to estimate at what time they might induce routing changes on higher level (BGP). By determining the worst case of convergence time with respect to the entire OSPF routing domain, we are able to estimate when the effect appears at the latest. This is important since we are interested in convergence phases that are as short as possible (see section 1.1). On the other hand the more and the faster OSPF converges the more load on processors it might generate. When doing Traffic Engineering it is therefore important to know how load evolves from shortening convergence time and intervals. The reasons for our definition of load were already demonstrated before.

Having introduced our research interests for this chapter so far, we yet need to classify the different factors that have a major impact on their behaviour. As it is neither useful nor possible to determine exact values for convergence time and load, due to the use of different hardware components, evolving technologies and different protocol implementations, we are more interested in getting a feeling for their range. In doing so two components play a major role. On the one hand there are the parametrical impacts that are direct affecting the protocols behaviour. We define parametrical impacts as the influences on routing behaviour when changing both protocol’s configurable parameters and architectural constants. On the other hand we study by means of different network compositions indirect influences on routing which we regard in a separate section about topological impacts. All these effects were analysed in the network simulator SSFNet.

In the course of this chapter we will first give a brief outline about the implementation of OSPFv2 in SSFNet. We are going to describe its capabilities and the improvements we put on to potentiate our experiments. The next two sections deal with the actual simulations. In section 4.2 we examine parametrical impacts. We introduce the OSPF parameters and architectural constants which are relevant for our work in their respective context. After that we show in a step by step fashion how OSPF convergence time can be decreased and which parameter is responsible for which degree of decrease. We will also illustrate where we came up against limits in our efforts and discuss the problems when trying to overcome them. Dependent of the change of some parameters, the number of routing table calculations (RTC) will increase or fall. This will be another part of our investigations. In section 4.3 we eventually study the impact of different topologies on OSPF routing behaviour. With respect to different types and sizes of networks we again observe
4.1 THE SSFNET IMPLEMENTATION

the changes of load and convergence time. In doing so, we will concentrate especially on
the coaction of different topologies with different protocol parameters. But also propa-
gation delays and their effect on convergence time are examined in greater detail, since
networks are composed in most instances of different media types which have different
propagation times.

4.1 The SSFNet Implementation

When we first used SSFNet there was just a static version of OSPF available (sOSPF). This implementation was not intended to study the protocol’s routing behaviour nor to simulate its effects on other routing protocols. sOSPF is rather a partial implementation of the IETF’s OSPFv2, designed to quickly compute routing tables for arbitrary topologies within a single OSPF area only, and to flood the area with external route announcements induced by BGP. The protocol retrieved the adjacency information once directly from the underlying network topology, and each router formed a single link-state database from the link properties gained from these information. Two major drawbacks made a use of sOSPF impossible: The implementation did not support any dynamics, besides the induced BGP routes, so that mechanisms like dynamic neighbour discovery, database synchronisation and link-state updates in response to dynamic topology changes were simply not available. Secondly, SSFNet at that time did not provide mechanisms to dynamically create link failures or recoveries. sOSPF was therefore not at all suitable for our purposes. We needed a dynamic protocol to study the impacts of topology changes on intra-domain routing. In addition we had to provide the infrastructure for dynamic topology changes. Since sOSPF did not provide enough basics to begin with dynamic extensions, we decided to entirely rebuild OSPF from the bottom-up becoming SSF.OS.OSPFv2. This implementation provides all dynamics specified in [27] that are necessary to run on point-to-point networks. A fully functioning Neighbour State Machine, triggered by incoming OSPF packets, manages the relationship to a neighbouring router by passing through a series of states from DOWN to FULL (see [27]). We implemented a complete, dynamic OSPF routing process consisting of a Hello Protocol, a Database Exchange Process, a Flooding Procedure, Dijkstra’s Shortest Path Algorithm and a routing table computing and updating routine. This process supports loopback as well as point-to-point interfaces and makes use of all types of OSPF packets specified in [27]. OSPFv2’s work is however limited by the restricted capabilities of SSFNet beneath the transportation layer in the following way: Because of the lack of a detailed link layer implementation, the Interface State Machine is just rudimental developed. On the network layer, some components of OSPFv2 like the Hello Protocol rely on multicast. Since the IP implementation of SSFNet does neither support multicast nor broadcast we had to simulate multicast by unicast which restricted the functionality. Hello Protocol tasks for example, like the discovery of neighbouring routers and the establishment of neighbour relationships thus could not be simulated. For
CHAPTER 4. ANALYSING OSPF

Figure 4.1: The SSFNet OSPFv2 core classes

this reason we also had to restrict our implementation to the point-to-point part of the protocol, since OSPF running on broadcast media utilises IP multicast.

To get a view of the important parts of the implementation, figure 4.1 shows an abstract representation of the java classes implementing the protocol’s core and the structure of their dependencies. In the following we give a short introduction to the classes. A documentation can be found in the source code or the javadoc available at [31].

At the top there is OSPF which is the main protocol class, extending the abstract class SSF.OS.ProtocolSession. It is responsible for the global part of the protocol’s work. An instance of this class is running on each OSPF router. It sends out and receives the various OSPF protocol packets via SSF.OS.IP and is responsible for managing the RTC (routing table calculation) process. That includes updating the IP Forwarding Information Base as well as the computation of a separate OSPF routing table and the triggering of SPF tree computations in the various area data structures representing the areas an OSPF router might be connected to.

Since OSPF has its own representation of routes, we decided to implement a separate OSPF routing table. Each OSPF protocol session references a single java class called HMRoutingTable where it stores best routes to all known destinations. This became necessary as OSPF distinguishes between routes to networks and routes to routers. Router routes come into play when doing hierarchical routing. Since they represent just intermediate routes, they are not stored in the IP table.

In chapter 2 we briefly described the notions for hierarchical routing and how OSPF performs it. Because it is fundamental to the architecture, we accounted for it from the very first. The class AreaData represents all information that is necessary to do routing on area level. That includes the following: It maintains a single link-state database (represented
by class LSDatabase) where it stores and removes all those LSAs, own and received ones, that were created for the area the AreaData object is associated with. It is also responsible for the initiation of the flooding process and the creation and destruction of self-originated LSAs within its area. When triggered by OSPF, AreaData computes an SPF tree from the topological information of its database by executing Dijkstra. Since an OSPF router may be connected to more than one area, the OSPF class holds as many AreaData objects as attached areas. This is indicated by a dotted line in figure 4.1.

Although it is quite unusual that a router has connections to many areas, each OSPF router is however connected to at least one area, via one or more network interfaces. For each network interface card (NIC in SSFNet) OSPFv2 is running on, it creates an Interface object. These objects represent the OSPF view of a NIC. They contain certain values like the type of the underlying network to which the NIC connects to (e.g. broadcast, ptp), the configured OSPF output cost, the intervals at which hellos are sent out this interface and neighbours are declared to be DOWN when no “hello” was heard (hello-interval, router-dead-interval) and some other. OSPF packets received on a particular NIC are processed in the associated Interface object. We will describe packet processing later in more detail, considering a Hello Packet as example.

Since OSPF is communicating with direct attached neighbours only, the Interface objects are also responsible for managing the relationships to those neighbours. That means that for each neighbour, OSPF locates on a particular NIC, the according Interface object creates a Neighbour object that implements all necessary data and routines to maintain an OSPF relationship, the Neighbour Data Structure. The core of the Neighbour objects are routines which together implement the Neighbour State Machine (NSM). Interface provides the logic that triggers these NSMs. Since OSPFv2 implements only the point-to-point part of [27], each Interface references only a single Neighbour object.

Among these tasks, the Interface class is also responsible for simulating multicast. We mentioned before that due to the lack of a multicast protocol in SSFNet, OSPF must send its multicast packets via unicast to each single recipient. Interface therefore initiates the unicast sending process of those packets in the Neighbour objects. In addition, it also implements the Timers which send multicast packets, like the SENDNEWHELLO_TIMER that is sending out hellos at regular intervals.

Managing a neighbour relationship, the Neighbour class provides, besides an NSM and certain informational data, the processing routines for the content of incoming OSPF packets. This ranges from the simple processing of a Hello Packet’s neighbour list to the more complex treatment of LSA headers and their bodys, received in database description and link-state update packets. Dependent on the content, Neighbour causes some actions to be performed. The sending of unicast packets is also part of it. We come back to this, when we describe how an RTC is triggered in OSPFv2.

Having introduced the most important classes of OSPFv2 so far, we will yet demonstrate how they are interacting. Since we can not describe all the mechanisms of OSPFv2
in detail, we focus on two examples which are giving an overview. For more information please read the documentation of SSFNet OSPFv2 [31]. In a first example we demonstrate packet processing by means of a Hello Packet. Figure 4.2 illustrates the procedure via big red arrows. In step one, OSPF receives a packet from SSF.OS.IP. After having performed some tests, the packet is either rejected (dropped) or accepted. We assume the packet passed the tests, then OSPF determines the area from which the packet was received and hands it on to the appropriate AreaData object. In step two, the packets type is determined (type 1: hello) and the Interface object of the receiving NIC is selected. Via the type, AreaData may call the correct Interface routine and passes the Hello Packet over in step three. Interface now handles the actual packet processing. After the content of the Hello Packet passed several tests, the sending neighbour’s associated Neighbour object is selected and an accordant event on the NSM is triggered (step four). Dependent on the actual state, the Neighbour object resets several timers\(^1\), matches and probably adjusts some data and, if necessary updates the state of the relationship. According to the new state, a new NSM event is triggered. For example, when the state moved to EXSTART the database exchange process is started by calling \texttt{NEIGHBOR.INITDBEXCHANGE()}. The packet processing procedure is similar for each type of OSPF packet. While classes OSPF and AreaData do a rough processing, mostly on the packet surface like header checks and type and origin determination, the classes Interface and Neighbor do a content based processing with direct effects on the protocol’s behaviour. We demonstrate one of these effects in a second example were we describe how an incoming update packet

\footnote{One of them, that is always reseted when receiving a hello, is the \texttt{DEAD\_TIMER}. It simulates the router-dead-interval}
triggers a routing table calculation. The processing of a link-state update packet can be illustrated by figure 4.2 as well. In the first three steps the packet is received, accepted, classified and passed to the according Interface object. Here the state of the NSM in the sending neighbour’s Neighbor object is examined and according to it, a new event on the NSM is triggered (step four). Neighbor eventually unpacks the LSAs stored in the packet and applies several tests to them. Among other things, the LSAs are for example tested whether they are originated by the router itself, usable or duplicates. We assume here that at least one LSA is usable and carries new information. In this case Neighbor triggers an installation process via its responsible AreaData object to install the LSA in the according LSDatabase. This is indicated in step five in figure 4.3. Since the packet came from the represented Neighbor router, Neighbor yet sends a direct acknowledgement back or schedules a delayed one via its managing Interface. Subjected to the architectural constant $MIN_{LS\_ARRIVAL}$ which we explain in more detail in section 4.2, LSDatabase installs or rejects the LSA and checks whether the routing table must be recalculated and if so, which parts. If a positive decision was made, LSDatabase triggers a routing table calculation (RTC) in step 6. OSPF thereupon begins a complex RTC process. If not already started or delayed by configurable constants (see section 4.2), OSPF calls all of its AreaData objects to compute the individual shortest path trees of their domains via Dijkstra. From the different SPTs it gets in return (step 7), OSPF then may compute a new routing table (RT). This is the actual complex part of the calculation since probably reachability of ABRs and ASBRs must be tested, inter-area routes must be recomputed, resulting in updates and removals of LSAs and by comparison of the routes just calculated with entries in the old RT, it is determined which routes in the IP table must be updated, deleted

**Figure 4.3:** Triggering a routing table calculation
or simply added. At the end in step eight, the old RT is replaced by the new RT and in a
ninth step OSPF may update the FIB of SSF.OS.IP.

In version 0.2.0 all these computations were yet made in zero time and at once. Delays
resulting from computation time or configured parameters in real OSPF implementations
were simply not allowed for. Part of this diploma thesis was to improve OSPFv2 in this re-
spect, to gain a more realistic behaviour when carrying out our simulations. We therefore
upgraded OSPFv2 by two improvements:

1. An Update Packet-Pacing Timer (PACINGTIMER) according to Cisco’s timers pac-
ing flood command specified in [26].

2. A Routing Table Calculation Control Timer (RTCCONTROLTIMER) used to man-
age the routing table calculation process.

The Update Packet-Pacing Timer controls the rate at which LS Update packets are trans-
mitted out of a NIC. It reduces the likelihood that a single new LSA instance is sent out in
a separate LS Update Packet. Instead the generation of LS Update Packages is delayed at
intervals to accumulate LSAs before they are placed in a new update, so that the packet’s
“space” can be better utilised and the number of LS Update Packages flooded through the
network is reduced. The time interval of PACINGTIMER is configurable via the OSPFv2
pacing_flood_time attribute. Its configurable range is from 5 milliseconds to 100 millisec-
onds. The default value is 33 milliseconds.

The Routing Table Calculation Control Timer synchronises the RTC calls from the LS-
Databases. In addition it simulates delays caused by the RTC process. This includes
processing delays like the SPF computation time and the time it takes to update the IP
table as well as configurational delays concerning RTCs, that is an spf-delay and an spf-
holdtime according to Cisco [25]. To reduce the amount of RTCs triggered by lots of
back-to-back installed LSAs, Cisco implemented an spf timer that delays SPF calcula-
tions. It is however not part of the OSPF specifications in [27]. The timer makes sure
that an SPF calculation is executed not before spf-delay time after a topology change was
received. Additionally a minimum of spf-holdtime delay must be met between two con-
secutive SPF calculations. The default value for spf-delay is 5 seconds, for spf-holdtime
10 seconds. Both values are configurable via their respective OSPFv2 attributes and may
range between 0 and 65535 seconds.

Processing delays simulated by the RTC Control Timer were implemented according to
a paper of Aman Shaikh and Albert Greenberg. In [11] they developed the following
formula to approximate the SPF calculation time for fully meshed networks of arbitrary
size

\[ 0.00000247 \times x^2 + 0.000978 \text{ seconds} \]
were \( x \) is the number of nodes in the network. All delays caused by SPF calculations in AreaData objects are calculated according to this formula. If a router is attached to more than one OSPF area, the calculation delay of the biggest area, i.e. the area with the most nodes, is chosen as the SPT calculation delay. [11] also examines update times for FIBs (Forwarding Information Bases). But the results could not be described by a formula. Nevertheless they stated that a FIB update lasts between 100 and 300 milliseconds dependent on the router architecture and little on the number of nodes in the network. Since there was no exact description or approximation for this delay, we decided to assume a delay between 100 and 300 milliseconds which is randomly chosen each time the IP FIB must be updated.

To upgrade OSPFv2 in terms of these features and to implement the timers introduced before, we had to make changes to the following classes of the SSF.OS.OSPFv2 package: OSPF, AREA_DATA, LSDATABASE, INTERFACE and NEIGHBOR. In the following we give a short abstract.

A PACING_TIMER class derived from SSF.OS.TIMER implementing the features of the Update Packet-Pacing Timer was created in class INTERFACE. The timer inherits from AREA_DATA the decision process (specified in RFC2328 chapter 13.3) to which neighbour of the associated NIC an LSA should be flooded out. Depending on the result of this process PACING_TIMER stores the LSA in PACING_FLOODLISTs created in the NEIGHBOR objects. When the timer fires, it triggers the LS Update generation and sending process in the NEIGHBOR objects. A single update is sent out to each concerned neighbour and the LSAs are transferred to the link-state retransmission lists of the according NEIGHBOR objects. If more LSAs must be sent out than fitting in a single LS Update Packet, PACING_TIMER is restarted. The timer’s firing interval is specified globally in OSPF since it is identical for each NIC.

In class OSPF an RTCCONTROL_TIMER was created to model and synchronise the RTC process. All calls triggering an RTC, whether they are delayed or not are now directed to this timer. The timer passes through four different phases indicated by boolean flags: In one phase it simulates spf-delay, a second phase is simulating spf-holdtime, a third SPT calculation delay and a fourth the IP table update time. By means of the boolean flags, the timer is able to determine which delay has expired and which action to take upon. For example, if the IP table update time expired, then the IP FIB is updated by the changes in HMROUTING_TABLE, debug output is given probably, and if an RTC call came in during the RTC process, a new process is started. RTCCONTROL_TIMER distinguishes between RTC calls due to changes in the SPTs and changes in inter-area destinations. In the latter case the timer makes a decision to remove or to build an inter-area route and schedules its execution time. It stores the associated LSA on a remove- or install-list to remember which routes to recalculate when the timer expires and sets appropriate management flags. Otherwise an entire RTC is prepared. The operation sequence of a calculation is the following: First RTCCONTROL_TIMER simulates spf-delay. After expiration, it checks whether spf-holdtime is met. If not, the difference to spf-holdtime is simulated. When
spf-holdtime is finally met, a new HMRoutingTable is calculated and the IP routes that must be installed or removed in the FIB are remembered on install- and remove-lists. In addition the calculation routines return the computation time which is taken as calculation delay for the timer. The computation is accomplished before the delay is simulated since the number of nodes in an SPT calculation is determined during the computation and, as mentioned before, this number is needed to calculate the computation delay in case of an entire RTC.

After the timer expired again, possibly debug output is given and RTCConrolTimer is restarted with a random IP FIB update delay. Which updates to make, after the update delay expired is determined by the IP route install- and remove-lists.

4.2 Parametrical Impacts On Convergence Time

The behaviour of the OSPF routing protocol depends in a great measure on parameter values, configured individually on OSPF routers by administrators. Concerning convergence time and the number of RTCs we try to give answers to the following questions in this section:

In what magnitudes do particular parameters affect the convergence behaviour of OSPF? How far may parameter changes speed up convergence time? Is there a lower bound for it and if so, what does account for this limit? Regarding convergence time, we will go far beyond the lower limits of todays OSPF implementations. [35] showed for ISIS that convergence time can be decreased into orders of milliseconds when making minor changes to the protocol specifications. Is this possible too when using OSPF instead and what would be the changes here?

Let us start now with a presentation of the OSPF parameters that impact the routing process after a topology change occurred. As described in section 2.2.1 the OSPF Hello Protocol is responsible for detecting new connections or the fail of established ones. Two parameters determine the time interval at which this can be accomplished: the hello-interval and the router-dead-interval. They were already introduced before.

Having detected a topology change so far, some routers have to produce new (instances of) LSAs to describe the change and then flood them throughout the routing domain. To protect against network elements that are changing very rapidly and routers that are reacting on these changes too often, since they would otherwise cause too much LSA update traffic, the protocol defines two architectural constants: MIN_LS_INTERVAL and MIN_LS_ARRIVAL. For MIN_LS_INTERVAL time an OSPF router is not allowed to create a new instance of an LSA since the last instance was created. On the receiver side, the router is not allowed to accept updates for a particular LSA on intervals shorter than MIN_LS_ARRIVAL time. The values of MIN_LS_INTERVAL and MIN_LS_ARRIVAL are set to 5 respectively to 1 seconds. They are part of the protocol specifications and can’t be configured.
4.2. PARAMETRICAL IMPACTS ON CONVERGENCE TIME

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Configurability</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello-interval</td>
<td>10 seconds</td>
<td>configurable</td>
</tr>
<tr>
<td>router-dead-interval</td>
<td>40 seconds</td>
<td>configurable</td>
</tr>
<tr>
<td>spf-delay</td>
<td>5 seconds</td>
<td>configurable</td>
</tr>
<tr>
<td>spf-holdtime</td>
<td>10 seconds</td>
<td>configurable</td>
</tr>
<tr>
<td>MIN_LS_INTERVAL</td>
<td>5 seconds</td>
<td>fixed</td>
</tr>
<tr>
<td>MIN_LS_ARRIVAL</td>
<td>1 second</td>
<td>fixed</td>
</tr>
<tr>
<td>rxmt-interval</td>
<td>5 seconds</td>
<td>configurable</td>
</tr>
<tr>
<td>delayed-ack-time</td>
<td>1 second</td>
<td>fixed in SSFNet</td>
</tr>
</tbody>
</table>

Table 4.1: OSPFv2 parameters, their default values and configurability

The flooding of LSA updates in turn is affected by the interval at which LS Update Packets are retransmitted in case of packet corruption or loss and the time, acknowledgements for these packets are sent back. [27] defines the router interface parameter `rxmt-interval` for configuring the retransmission interval of Database Description, LS Request and LS Update packets and proposes a value of 5 seconds for it. The acknowledgements that are sent back on received LSAs (see section 2.2.1) can be delayed in some cases, but there is neither a default value for this delay, nor does [27] enumerate it as a configurable or architectural constant. It is only recommended to keep it smaller than `rxmt-interval`. We therefore decided to introduce a new router interface parameter `delayed-ack-time` that accounts for this delay with a fixed value of 1 second, which is the smallest value among the configurable ones.

Having received and successfully installed the updated LSA, OSPF has to recompute its routing table. The time when the computation is executed, depends on the spf timer and its configurable parameters `spf-delay` and `spf-holdtime`. It was implemented in the OSPF.RTCCTRL_TIMER and its functionality and the default values for its parameters were already introduced in section 4.1. Table 4.1 summarises again all parameters examined in this section, their default values and configurability.

![Figure 4.4: Simple OSPFv2 topology with two failing links](image)

The effects of changes to these parameters were studied in a simple network, depicted
in figure 4.4. All routers in this network are running OSPFv2, all interface output costs were set to 1. The red dashed lines between router 3 and 2, respectively 2 and 4 indicate links which will simultaneously fail and recover during a simulation run. These links and the topology itself were set up that way to make parametrical effects visible. We will yet discuss this in the subsections below. All simulation runs on this topology are executed in the same way: The duration of a single simulation run is 300.0 seconds. At time 100.0, both red dashed links fail. At time 200.0, they recover. During the fail and recovery time we measure convergence time and the number of RTCs.

In a first run, we configured OSPF with the default parameter values from table 4.1. The results are summarised in the following table:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>convergence time after fail:</td>
<td>49.30353 seconds</td>
</tr>
<tr>
<td>convergence time after recovery:</td>
<td>24.26988 seconds</td>
</tr>
<tr>
<td>number of RTCs after fail:</td>
<td>10</td>
</tr>
<tr>
<td>number of RTCs after recovery:</td>
<td>9</td>
</tr>
</tbody>
</table>

We see that it takes about two RTCs for each router on average to find stable best routes after a change. The convergence time after the fail is about twice the time it took to converge after the links came up again. The explanation for that is up to the Hello Protocol as we will see in the next subsection.

### 4.2.1 The Hello Protocol

Configured with default values, each router sends out a *hello* every 10 seconds. The SSFNet OSPFv2 implementation incorporates asynchronously running interfaces by delaying the Hello Protocol on each OSPF interface at simulation start. The delay is determined randomly from a time interval between 0 and 10 seconds at a granularity of seconds. So when *router-dead-interval* is set to 40 seconds, it takes between 30 and 40 seconds for an arbitrary interface to detect a connection fail. On the other hand, a connection recovery is detected much earlier. It takes at most 10 seconds to determine a bidirectional link since at least one neighbour detects bidirectionality within 10 seconds, due to the asynchronously firing interfaces. This neighbour will then exchange into state EXSTART and immediately start sending Database Description packets. The other neighbour treats the receipt of such a packet as a “2-way-received” (see chapter 2.2.1) and declares the connection to be bidirectional.

We come to the conclusion that, due to the conservative default values for the Hello Protocol parameters, about 70% of the convergence time is needed to detect a connection fail and still 40% to notice its recovery. Because of the tremendous rise of link and processor capacities in the last years, there is no reason anymore to abide by the defaults. We therefore may offhand configure much smaller intervals at which hellos are sent. In a second run, we decreased *hello-interval* to 1 second and *router-dead-interval* to 3 seconds, all other parameters were left unchanged. The results are given here:
4.2. PARAMETRICAL IMPACTS ON CONVERGENCE TIME

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>convergence time after fail:</td>
<td>17.28553</td>
</tr>
<tr>
<td>convergence time after recovery:</td>
<td>17.26015</td>
</tr>
<tr>
<td>number of RTCs after fail:</td>
<td>8</td>
</tr>
<tr>
<td>number of RTCs after recovery:</td>
<td>9</td>
</tr>
</tbody>
</table>

We see that the number of RTCs is nearly unchanged, whereas the convergence times could be decreased a lot, which is of no surprise. But it might be surprising that it now takes nearly as long to detect a connection fail as it takes to find it recovered. Since the simulation scenario is the same than the one before, besides a change to the Hello Protocol the explanation is quite simple: In the same way as derived before, we expect the time for the detection of the failing links to be between 2 and 3 seconds. Since the granularity of the delay which adds asynchronous behaviour is 1 second and the hello-interval is also set to 1 second, the interfaces now send synchronous hellos. That means, it takes at least 2 seconds to detect the recovery. The marginal deviation between the two convergence times is also explained by the randomly determined IP table update time (see section 4.1).

4.2.2 Routing Table Calculation Intervals

Having reduced the Hello Protocol component of convergence time to a minimum so far, we may now examine the proximate biggest parameter value which is, according to our defaults in table 4.1, \(\text{spf-holdtime}\) and the respective \(\text{spf-delay}\). As seen in the results before, it takes two RTCs on average for each router to calculate the new best paths after a topology change. The time interval between the changes is 100.0 seconds, so that, when the first LSA update indicates an RTC, \(\text{spf-holdtime}\) is definitely met at that time. From these facts we can derive that the \(\text{spf-delay}\) time is altogether 15 seconds since it takes \(\text{spf-delay}\) until the execution of the first RTC and again \(\text{spf-holdtime}\) till the second is performed. So, if we additionally switch off both delays, we can expect both convergence times to be between 2 and 3 seconds. The next table represents our results of measurements when setting hello-interval to 1, dead-interval to 3 and additionally \(\text{spf-delay}\) and \(\text{spf-holdtime}\) to 0 seconds.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>convergence time after fail:</td>
<td>7.30959</td>
</tr>
<tr>
<td>convergence time after recovery:</td>
<td>7.27182</td>
</tr>
<tr>
<td>number of RTCs after fail:</td>
<td>15</td>
</tr>
<tr>
<td>number of RTCs after recovery:</td>
<td>15</td>
</tr>
</tbody>
</table>

Interestingly the results don’t meet our predictions. Both convergence times could in fact be reduced, but only by 10 seconds which fits exactly \(\text{spf-holdtime}\). We thus observe that there must be another factor which in parallel adds circa 5 seconds to convergence time. We will deal with it in the next subsection. On the other hand the number of RTCs increased strongly as expected. An RTC is now performed immediately after an LSA was updated. Since the Hello Protocols are running almost synchronously, each router detects
the change simultaneously and recomputes its routing table for exactly three times. For the first time after the first of the router-LSA updates from router 2, 3 and 4 arrived, for the second time to account for the other LSA updates arrived after the first and for a third time, when router 2 updates its router-LSA once again, 5 seconds later to react on the change of its second link. This 5 second delay, seeming strange at the first sight, is caused by another parameter examined next.

4.2.3 LSA Advertisement and Installation Restrictions

We mentioned before that OSPF was equipped with a mechanism to reduce update traffic when reacting on topology changes. To demonstrate this mechanism we constructed our topology that way it is shown in figure 4.4. We needed a router having three links. Two of them should fail and recover to create at least two updates of the router’s router-LSA, the third is used to announce the updates. Due to the fixed architectural constant MIN_LS_INTERVAL the second update of router 2’s router-LSA is created 5 seconds after the first. If we could reduce MIN_LS_INTERVAL and MIN_LS_ARRIVAL to 0 seconds in addition to the parameter changes made before, we would get the following results:

<table>
<thead>
<tr>
<th></th>
<th>2.45323 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>convergence time after fail:</td>
<td></td>
</tr>
<tr>
<td>convergence time after recovery:</td>
<td>2.4925 seconds</td>
</tr>
<tr>
<td>number of RTCs after fail:</td>
<td>10</td>
</tr>
<tr>
<td>number of RTCs after recovery:</td>
<td>10</td>
</tr>
</tbody>
</table>

Since router 2 is now able to update its router-LSA immediately after realizing a change, only two RTCs per router are yet necessary\(^2\). The convergence times now only comprise of the Hello Protocol contribution to detect the change and some computational delay. As possible additional parametrical delays, we have only packet retransmissions left. These are in orders of rxmt-interval and depend on many factors like hardware capacities and traffic volume. We are not going to examine them here since they are not part of this thesis. We will also not deal with delayed-ack-time because it does not contribute to convergence time according to our definition. But it plays a role when we analyse the interaction of parameter changes with respect to optimal values, whatever optimal means in doing so. We therefore listed it in table 4.1.

4.2.4 Optimal Parameters?

Although there are several parameters which add delays of different sizes to the overall convergence time, all of them are nevertheless important to balance the extent of time,

---

\(^2\) Runs where only MIN_LS_INTERVAL was set to 0, generated a second more delay, caused by router 5 when installing the update. This was however enough to force a third RTC.
4.2. PAREMERTICAL IMPACTS ON CONVERGENCE TIME

RTCs and routing messages to be sent when responding on route changes. It thus would be a bad idea to simply set some of them to 0 to speed up convergence time. Since the granularity of the configurable parameters is one second and some parameters are even fixed, there is not much leeway, apart from the Hello Protocol parameters, for reducing convergence time by reducing parameter values. We hence conclude that convergence time in hundreds of milliseconds for example, can not be accomplished without changes to the protocol itself. These changes would incorporate making all of the parameters of table 4.1 configurable and extending their configurable range to milliseconds. [27] states no reason why some of them must have fixed architectural values and we don’t see any problems in making them configurable. The same holds for extending the range except for hello-interval and router-dead-interval. This is because both parameters are transmitted to neighbouring routers via Hello Packets, to turn down a relationship immediately if it is detected that the involved routers use different values for hello-interval and router-dead-interval. Since these values are 16- respectively 32-bit Integers and taken by seconds, we cannot configure milliseconds here without running into semantic errors. Therefore the protocol should be altered in one of two ways: Either the Hello Protocol parameters are taken by milliseconds and the according timers are adjusted. In this case the range for hello-interval would be from 1 millisecond to 65,536 seconds. Or we must redesign the Hello Packet to allow for a bigger range of Hello Protocol parameter values. Whatever the solution would be, in the next simulation run we made OSPFv2 capable to accept our changes and configured the following values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello-interval</td>
<td>0.01 seconds</td>
</tr>
<tr>
<td>router-dead-interval</td>
<td>0.03 seconds</td>
</tr>
<tr>
<td>spf-delay</td>
<td>0.05 seconds</td>
</tr>
<tr>
<td>spf-holdtime</td>
<td>0.10 seconds</td>
</tr>
<tr>
<td>MIN_LS_INTERVAL</td>
<td>0.05 seconds</td>
</tr>
<tr>
<td>MIN_LS.ARRIVAL</td>
<td>0.01 seconds</td>
</tr>
<tr>
<td>rxmt-interval</td>
<td>0.05 seconds</td>
</tr>
<tr>
<td>delayed-ack-time</td>
<td>0.01 seconds</td>
</tr>
</tbody>
</table>

Since these values are exactly one-hundredth of the values configured in section 4.2.1, we could assume that both convergence times are likely to be about 170 milliseconds. Indeed the results deviate quite a bit as depicted in the next table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>convergence time after fail:</td>
<td>0.698706 seconds</td>
</tr>
<tr>
<td>convergence time after recovery:</td>
<td>0.641968 seconds</td>
</tr>
<tr>
<td>number of RTCs after fail:</td>
<td>9</td>
</tr>
<tr>
<td>number of RTCs after recovery:</td>
<td>9</td>
</tr>
</tbody>
</table>
The number of RTCs remained roughly the same, but the convergence times are much bigger than expected. But even if we reduced the parameter values once again, we would not get much lower convergence times. The additional delay we see here, comes from the execution time of Dijkstra’s SPF algorithm and the time it takes to update the IP tables (see section 4.1). This is indeed a lower bound for convergence time when regarding parameters only. So if we are interested in convergence times in link propagation time scales, we have to follow the advice in [35] to switch to modern SPF calculation algorithms and more efficient data structures to store IP routes. But this is not part of this thesis.

Similar to [35], we however demonstrated that it is possible to reduce convergence times to magnitudes of hundred milliseconds by just making marginal changes to the protocol specifications. Whether the parameter values we used for that purpose were optimal or not cannot be stated, since routing does not solely depend on convergence time. It is also a question of other factors like the amount of routing data generated to detect and propagate changes (e.g. Hello Packets), the number of additional RTCs to react more quickly on those changes, the size and shape of the network and its propagation delays and last but not least the effects that local changes may have on global routing. With one of these factors, the impact of different topology sizes and shapes, we are going to deal in the next paragraph.

4.2.5 Summary

In this paragraph we demonstrated the impact of different protocol parameters on OSPF’s routing behaviour. We have seen that convergence time is mostly determined by the configured values of these parameters. Due to the fact that not all of them are configurable and that they are in orders of seconds, OSPF is not able to react on topology changes in range of milliseconds. Nevertheless we showed that some of the default values are too conservative and may be reduced a lot. We suppose for example to set the hello-interval to 1 second and router-dead-interval to 3 or 4 seconds to detect network changes more quickly. Furthermore the spf-delays, which are responsible for the amount of RTCs could be reduced, but we didn’t study the effect of smaller delays here, since the number of RTCs is also dependent on the network’s propagation delay, which depends much on its size and shape. The topic is therefore postponed to the next paragraph.

Today the Internet is more and more used by applications which react sensitive on transmission delays and packet loss. During a convergence phase routes become unstable, leading to wrong routing decisions which again leads to additional transmission delays and packet losses. To reduce those disturbances of traffic flow, it is proposed to reduce convergence time to magnitudes of link propagation delays, which is currently tens of milliseconds. We showed that with little changes to the protocol design, OSPF can be made capable to converge in hundreds of milliseconds when configuring appropriate parameters. Admittedly we did not study the consequences from those changes. A single
Hello Packet on a point-to-point link for instance, is composed of a 24 byte body (we assume that it stores a 4 byte neighbour IP address), a 24 byte OSPF header and a 24 byte IP header. If we configure default values, we can expect 72 bytes/second of Hello-Data sent by each interface. The parameter values from section 4.2.4 however would produce about 7 kilobytes/second. On broadcast media we yet had to expect a hundred times more. Nevertheless we have demonstrated with which OSPF convergence times we deal today and which may be expected in the future. It remains to show whether a simple, arbitrary network topology is sufficient to produce representative results in this respect.

4.3 Topological Impacts On Convergence Time

In the last paragraph we observed that, dependent on the configured protocol parameters, the convergence phases of OSPF last between a few seconds and about 50 seconds. Since all tests were accomplished using a simple network topology, we can’t say to what extend these results can be regarded as realistic. Hence we are now going to examine how far more complex topologies may influence the spectrum of convergence times. As we have seen in chapter 3, there is no typical topology which we could use to demonstrate each network feature that can have an impact. Instead we decided to use extreme topologies to determine the limits, between the topological impact ranges. As elucidated in chapter 3 and shown in figure 3.3 these will be ring-shaped and fully meshed networks.

In the following we are going to extend our examinations from paragraph 4.2 to ring- and fully-meshed-shaped topologies of different sizes. We will thereby concentrate on the configurable parameters. Similarly as before, we analyse at a time a particular factor in rings as well as in full meshes. The respective results are incorporated in the next simulations. At first we deal with different Hello Protocol configurations, thereafter we are going to examine different RTC intervals and try to give recommendations for their settings. Since networks are usually composed of different link technologies, we are also going to analyse the impact of different link propagation delays on OSPF’s convergence behaviour in a last section.

In all of these studies the routing domain was not subdivided, i.e. we configured the respective ring and full mesh networks as OSPF backbones. Our hard- and software allowed for rings in dimensions of 3 up to 500 routers. Between 100 and 500 routers we created scenarios in steps of 50, below 100 the rings had sizes of 3, 10, 25, 30, 50 and 75 routers. Concerning fully meshed networks we could simulate scenarios consisting of 5 up to 50 routers in steps of 5. Due to limitations in hard- and software, it was not possible to go beyond these limits since for example a single full mesh scenario of 50 routers needed more than 1 Gigabyte of memory to terminate. But we are concerned that these numbers reflect the actual situation in real networks.

All simulations were accomplished in the same way: After a setup phase in which the routers learned the topology and computed the forwarding tables accordingly, a partic-
ular link was turned down for a time interval, long enough for OSPF to converge. We measured load and convergence time after the link was turned down and after it came up again respectively. The scenario was thereupon repeated twice with other random seeds to prevent misinterpretations due to possible extreme cases. In each scenario we provided all interfaces with a bitrate of 10 Megabits per second and a latency of 0.1 milliseconds. In the following we present a cutting of our work. In the course of this paragraph we abbreviate the phrase “convergence time after the link was turned down/came up again” by “convergence-time-down/up”.

4.3.1 The Hello Protocol

Similar to section 4.2.1 we begin our topology analyses with a closer examination of the OSPF Hello Protocol. Figure 4.5 shows the progression of convergence time for both ring-shaped (left side) and fully meshed networks (right side) when configuring different values for hello-interval and router-dead-interval. By “default Hello Protocol” we denote that these parameters were configured with the default values from table 4.1. Similarly “1/2 default Hello Protocol” denotes that hello-interval and router-dead-interval were set to the half of their default values and so on. All other parameters were left to their defaults.

When regarding the left side of figure 4.5 we notice that convergence-time-down is growing slightly linear and proportionally to the size of the rings. But at a particular point, it suddenly makes a jump of approximately 8 seconds. This can be explained by the fact that the disconnected interfaces detect the fail at different times and so they generate their updates. In addition the propagation delay for these updates grows proportional to the ring size, so that at a particular size the accumulated delays between the two updates of the fail detecting routers is bigger than spf-delay. Thus on some routers, a second RTC must
be scheduled \textit{spf-holdtime} after the first, which results in a serious rise of convergence-
time-down. This is substantiated by the RTC growth progression in figure 4.6. We further
notice that the phenomena moves rightwards the more the Hello Protocol parameters are
decreased. This can be explained by the fact that due to the shortened intervals, the dis-
connected interfaces detect the fail almost synchronously and the propagation delay now
must fill a big part of the accumulated delay to exceed \textit{spf-delay}. We also observe in figure
4.6 that with increasing ring size the number of second RTCs increases too, because the
second LSA update reaches even less router in \textit{spf-delay} time. Convergence-time-up on
the other hand, behaves completely different from that. The only thing in common with

\begin{figure}[h!]
\centering
\includegraphics[width=0.5\textwidth]{rtc_growth_ring.png}
\caption{Growth of RTCs in Ring Topologies when changing Hello Protocol parameters}
\end{figure}

convergence-time-down is its slightly linear growth, caused in both cases by the increas-
ing propagation delay. Regarding convergence-time-up we observe a kind of oscillation.
Since the amplitudes of this oscillation shrink proportional to the decreased Hello Proto-
col parameters, they can be ascribed to the hello-interval. As indicated in paragraph 4.2
it takes at most 10 seconds to detect a bidirectional link when configuring default Hello
Protocol parameter values. So due to the asynchronously hello sending interfaces we can
expect a detection delay between 2 and less than 10 seconds when using default values.
When cutting them in half, this should be between 2 and less than 5 seconds. With a
\textit{hello-interval} set to 1 second, we expect only 2 seconds of detection delay. This is ac-
cording to the mechanism described in section 4.2.1. Including an \textit{spf-delay} of 5 seconds
this explains the convergence-up-times.

It is remarkable that we don’t see a second RTC on any router (see figure 4.6). This is
because after the routers have detected the bidirectional link, which happens due to the
design nearly simultaneously, they determine that their databases are already synchro-
nised and so they are able to update their router-LSAs at the same time. Since the ring
structure is now closed again, both LSAs arrive at each router almost synchronously so
that one RTC is sufficient to converge.
We conclude that in ring-shaped networks the topological delay has an impact on OSPF routing behaviour, but does not affect convergence time significantly if we go without particular link propagation delays. Our configured bitrates and latencies for example add up to approximately 1 second of additional convergence time per 100 routers.

Regarding the right side of figure 4.5 we can state that convergence time in fully meshed networks is independent of the network size. Since each router is connected to each other directly, there is no appreciable propagation delay. But it is noteworthy that we can save about 80% of convergence-time-down when assigning minimal values to the Hello Protocol. We already explained in section 4.2.1 the big time difference between convergence-time-up and convergence-time-down when using default Hello Protocol values and why both converge when decreasing the values. It remains to explain the big amplitudes in convergence-time-down when using default Hello Protocol values and the spike at 30 routers where the associated convergence-time-up lasts more than 21 seconds.
The amplitudes at 5, 10 and 20 routers in the former case come from a second RTC. Although one would believe that due to the network’s fully meshed structure, a single RTC is sufficient to led OSPF converge, it is again the amount of time between the different fail detection times of the involved interfaces, i.e. their asynchronous operating, that may exceed spf-delay. Taking a hello-interval of 10 seconds, an amount of 5 seconds and more difference is quite possible. The spike in convergence-time-up in the 30 router network however is explained by an accidental synchronicity of the hello sending processes on the involved interfaces. 6 seconds after the recovery of the link, both interfaces send a hello and turn to state INIT after the receive. 10 seconds later, the link is declared to be bidirectional by both and after spf-delay additional seconds the RTC is accomplished.
We conclude that, without particular link delays, in fully meshed topologies the network size has neither an impact on OSPF’s convergence behaviour nor on its convergence time, because of the especial network structure.

4.3.2 Routing Table Calculation Intervals

We have seen that a big part of convergence time is caused by the OSPF Hello Protocol if we configure default values. When setting hello-interval and router-dead-interval to minimal values like a tenth of their default we do not only reduce convergence time a lot, but we also eliminate spikes coming from different detection times due to asynchronously operating interfaces which cause additional, delayed RTCs.
Bearing in mind this, we now examine how changes to the RTC delay intervals spf-delay and spf-holdtime affect convergence time as well as load. To concentrate on delays caused by the spf-timers we set the Hello Protocol parameters to a tenth of their defaults in all scenarios of this section. The rest of the parameters are left to their defaults. Hereupon we reduce, in five steps, spf-delay and spf-holdtime proportionally from their defaults to 0. Figure 4.7 shows the results for ring-shaped topologies. On the left side we begin
with “default RTC scheduling” which corresponds to the “1/10 default Hello Protocol” scenario from section 4.3.1, that is only the Hello Protocol parameters were set to a tenth of their defaults, all other especially the RTC delay intervals were left on the values from table 4.1. In a first step, we reduced \(spf\)-delay and \(spf\)-holdtime by 20\% of their defaults to 4 and 8 seconds respectively (“0.8 default RTC scheduling”). Then we repeated this step until both parameters were set to 0, denoted by “no RTC scheduling” on the right side of figure 4.7. We notice that, the more we reduce the RTC delay intervals the earlier and more frequently a second RTC must be executed when a link went down. This is also confirmed by the progression of RTC growth shown in figure 4.8. On the other hand we save up to 5 or 8 seconds of convergence-time-down depending on the network size and
CHAPTER 4. ANALYSING OSPF

the configured values.
We see thus, choosing good values for spf-delay and spf-holdtime is not trivial, since
dependent on the topology, one has to balance between load and convergence time. In case
of our ring-shaped topologies we observe that reducing the default RTC delay intervals
down to 60% we can save more than 2 seconds of convergence-time-down at each step in
big networks. If we reduce these parameters anymore we soon have bad load performance
and gain only little time. Therefore an spf-delay of 3 and an spf-holdtime of 6 seconds
seem to be a good compromise between load and time.
Concerning convergence-time-up we observe a reduction of approximately 1 second per
step, due to the prevailing configured spf-delay time, besides in the last step where 2 RTCs
must be performed per router which may add up to nearly a second. In all other cases each
router performs exactly 1 RTC since the two LSA updates arrive almost synchronously at
each router as explained in section 4.3.1, that is within spf-delay time.
Figure 4.9: Convergence time progression when changing RTC scheduling in Full Mesh Topolo-
gies
we yet show how these changes will affect fully meshed networks. For that purpose we
applied the same configurations as in the ring scenarios. Figure 4.9 presents the results.
Since we resolved any spikes by setting the Hello Protocol parameters to minimal values,
we get constant convergence times, shrinking proportionally to the decreasing spf-delay
time. Due to the structure of fully meshed networks and the almost synchronously sending
interfaces, a fail or recovery is detected and propagated throughout the whole network
very quickly. Thus the second LSA update, generated by one of the detecting routers
arrive always before spf-delay time is elapsed, and so each router needs 1 RTC only to
converge, except for case “no RTC scheduling”.
Although we did not specify specific link propagation delays, it can be assumed that, if not
all links of at least one router in a full mesh network have a significant propagation delay,
4.3. TOPOLOGICAL IMPACTS ON CONVERGENCE TIME

i.e. at least a few seconds, we won’t see any changes to the previous convergence time and processor load results. In this case and presumed that the involved interfaces detect a change almost at the same time, due to a quickly acting Hello Protocol, \textit{spf-delay} and \textit{spf-holdtime} may be set to 1 and 2 seconds respectively without increasing the number of RTCs. But if the link propagation delays are in the order of seconds, or if the detection times differ too much, this does of course not hold anymore. Anyway we can state that in full mesh networks OSPF’s convergence times remain constant in comparison to the network size whereas in ring-shaped networks we can expect a linear growth. We will be closer concerned with this topic in the next section.

4.3.3 Link Propagation Delays

Up to now we have seen how far OSPF’s convergence time can be influenced by different parameters, topologies and network sizes. In all of our simulations however, we did not specify particular link propagation delays. To better bring out parametrical features without adulterating them by accumulated propagation delays, only a bitrate of 10 Megabit/second and a latency of 0.1 milliseconds was configured per interface. But the impact of a network’s topology and size on convergence time is immensely bigger if appropriate delays are taken into account. In the following we provided our ring and full mesh networks with link propagation delays. Two scenarios were created for it: In a first scenario we configured today’s common delays, ranging between 1 and 100 milliseconds. The second scenario added delays between 1 millisecond and 1 second. In both cases the individual link delays where determined as follows: The maximum delay was divided by the number of links in the network and the result was taken as an interval. To the first link we assigned 1 millisecond of delay. Each other link got the delay of the one before plus the determined interval. So on average each link had a propagation delay of 50 or 500 milliseconds dependent on the scenario. In ring-shaped networks we assigned link delays hop by hop. In fully meshed networks the delays were assigned router by router backwards, so that we had always one router which was connected to each other router via the slowest links, thus serving as the “bottleneck” in routing. The link with the smallest delay was determined to fail and recover during all simulation runs.

In the course of this chapter we have seen that a lot of convergence time can be saved if some OSPF parameters are set to small values instead to the defaults. We could also regard some characteristics in the progression of convergence time. But we did not examine how far these savings and characteristics are dependent on propagation delays. For that reason we used two different OSPF configurations in each scenario to closer investigate this relation. In “default OSPFv2” all OSPF parameters were set to the defaults specified in table 4.1, “1/10 Hello & 0.6 RTC scheduling” signifies that \textit{hello-interval} was set to 1, \textit{router-dead-interval} to 4, \textit{spf-delay} to 3 and \textit{spf-holdtime} to 6 seconds, the values we determined to give good performance. In figure 4.10 we see the results for both configurations when running in ring-shaped and full mesh topologies provided with link delays.
between 1 and 100 milliseconds. Regarding the ring-shaped topologies on the left side we notice that all characteristics discussed before can be clearly recognised. But due to the additional delays we see that, after a link went down, a second RTC is already necessary in much smaller networks (at a size of 50 routers) than before. The convergence time is of course still growing linear but much faster now. On the other hand, if we regard the full meshes on the right side of figure 4.10 our assumption from the last section is confirmed. Since each router is directly connected with each other, and because none of their links has more than 100 milliseconds propagation delay, there is no discernible difference to the results before.

We conclude that, if we configure common delays between 1 and 100 milliseconds, the link propagation delay might have an impact on convergence time and the number of RTCs, but this impact is nevertheless minor than the effects that are caused by parameter changes. If we however extend the range to 1 second, things may change completely. Figure 4.11 shows the convergence times for rings and full meshes for this case. We see that, for ring-shaped networks on the left side, the link propagation delay definitely dominates convergence time. Due to the big delays, almost all networks need two RTCs to compute stable routes after the link went down. But after it came up again the second update still propagates fast enough to be in spf-delay time, independent of the OSPF configuration, and thus one RTC is still sufficient to converge. The difference between “default OSPFv2” and “1/10 Hello & 0.6 RTC scheduling” is however small concerning convergence-time-down and even negligible concerning convergence-time-up in comparison with the entire convergence time.

Regarding the fully meshed topologies on the right side, we detect that at first sight, there is no difference in comparison to the results in figure 4.10. A closer examination however reveals that convergence times indeed increased marginally. This is comprehensible since the average link propagation delay only increased from about 50 milliseconds to 500 mil-
4.3. TOPOLOGICAL IMPACTS ON CONVERGENCE TIME

Figure 4.11: Convergence time progression when adding link delays between 0.001 and 1.0 seconds

milliseconds and due to the network structure each router is reachable via a single hop. Although we know that link delays of 1 second are quite unusual, they demonstrate at least the fact that OSPF’s convergence behaviour is almost independent from propagation delay. We can also state that OSPF’s convergence time is approximately between 5 and 50 seconds plus a factor for distributing routing informations, preconditioned that the OSPF parameter values don’t exceed the defaults.

4.3.4 Summary

In paragraph 4.3 we examined the impact of network topologies of different sizes on the convergence behaviour of OSPF. For that purpose we made use of extreme cases, i.e. ring-shaped and fully meshed networks, to primarily study the limits of convergence time. We can state that the topological impact is by far less than the impact that parameter changes have. Only on extreme conditions, like in ring-shaped networks consisting of hundreds of routers with link propagation delays of up to 1 second, convergence time is highly dependent on the topological impact. The reason for that is in the order of magnitudes of the OSPF parameters and the propagation delay. The former are in seconds, the latter usually in tens of milliseconds. Thus it really needs extreme cases to exceed for instance an SPF-holdtime of 10 seconds by accumulated propagation delay. There is also the fact that convergence time is determined by the number of RTCs. For example if a fail is detected almost synchronously, the probability for a delayed, second RTC can be reduced a lot, which of course reduces again the convergence time. Since interfaces normally don’t operate synchronously, this can be accomplished by setting hello- and router-dead-interval to the smallest possible and sensible values, i.e. 1 and approximately 3 or 4 seconds respectively. Another aspect is the size of the RTC delay intervals. Dependent on the topology it is recommendable to decrease or even to slightly increase the values for
spf-delay and spf-holdtime to save convergence time by scheduling the first RTC earlier or by preventing a second RTC schedule by extending the schedule for the first. Last but not least, OSPF features by the little amount of routing traffic it generates, which does not contribute any significantly delay to the convergence process when sent.

### 4.4 Conclusion

Our analysis of the Interior Gateway Protocol OSPFv2 revealed that its response time on changes in the network topology is in tens of seconds. The main reason for that are the configurable and architectural OSPF constants which are restricted to time scales of seconds. Since link propagation delays are usually in dimensions of tens of milliseconds, the OSPF constants remain the dominating factor. Among these constants the Hello Protocol parameters have turned out to be the most time consuming factors. A reduction of hello-interval and router-dead-interval to minimal values may save up to 80% of convergence time in some cases.

Another point of our examinations was the processor load generated by OSPF which we measured in terms of number of RTCs. This number highly depends on the configured values for the spf-timers, i.e. spf-delay and spf-holdtime, but also on the MIN_LS_INTERVAL and MIN_LS_ARRIVAL constants and the propagation delay of LSA updates. Due to the constants and configurable RTC delay intervals a lot of RTCs may be saved since changes can be accumulated before updates are formed and updates again can be collected before an RTC is executed, although the defaults and the restriction to seconds often make not much sense. Its remarkable that Cisco has replaced its timers spf command by a new timers throttle spf command in its IP Routing Protocols release 12.2(14)S [20], which allows, among other things, to set the spf timers to milliseconds. This is an indication that spf-delay and spf-holdtime might have generated too much time overhead in the past and that saving RTCs is not so important anymore than it was years ago. The latter might be due to the fact that processing capacities have evolved over the years, so that a computation of an OSPF routing table is less expensive anymore, and that the number of updates, triggering an RTC is relatively small (MIN_LS_INTERVAL, MIN_LS_ARRIVAL) so that an RTC more or less doesn’t preponderate. Concerning an interaction with BGP however, convergence time is the more important factor. We therefore abstain from measuring the number of OSPF RTCs in the following.
Chapter 5

Simulating OSPF-BGP Interactions

Our goal in this chapter is to understand what impact local IGP routing changes can have on the global routing process, performed by BGP. In particular we are interested in answering questions like: What happens with BGP if an intra-AS route changes or becomes unreachable? Is BGP routing affected by it and if so, in which cases and to what extent? But before we can approach these questions we first have to explain how BGP is linked with IGP to understand the interactions that may arise between those different types of routing protocols. To our best knowledge there haven’t been yet a direct interaction between an IGP and BGP. That means the routing processes work separately from each other. This is because BGP is an inter-Autonomous-System routing protocol whose primary task is to exchange network reachability information with other BGP speakers to form a policy-based graph of AS connectivity and not to do shortest path routing as it is the primary task of IGPs. Therefore an IGP can only affect BGP routing if BGP

- makes use of IGP metrics as a tie-breaking rule in the BGP best path selection algorithm (performs hot potato routing) or
- maps MED attributes to IGP shortest path metrics when announcing routes to neighbouring ASes, and the BGP speakers in the neighbouring ASes make their routing decisions based on these MEDs (perform cold potato routing).

Although its priority in the BGP path selection process might be quite low in some implementations [5], the use of IGP metrics as a tie-breaking rule however is well known and widely used, since it is described in [1] as a basic criterion for selecting among different routes for the same destination prefix. The “cold potato” routing strategy again, enables the MED announcing AS to direct transit traffic along an AS ingress point that has the shortest path to the AS egress point for the traffic’s destination. That means that traffic coming from a neighbouring AS, not destined for the own AS is kept as short as possible in the own AS. This strategy seems to be quite popular among ISPs [23]. Unfortunately IGP shortest paths and their metrics are not static and may change due
to AS internal link or router failures, traffic engineering, etc. Moreover the BGP protocol specification [1] only provides path reevaluations based on BGP updates. Current BGP implementations solve this problem by periodically checking best paths and IGP dependent attributes for all BGP prefixes and re-advertising prefixes in case their attributes changed. This is of course not done synchronously but each router checks its routes individually and makes its own decisions about which of them must be changed and how far. Consequentially each router can originate BGP instability information based on IGP routing changes.

In the course of this chapter we are going to study the causes for those interactions and their consequences and try to estimate the extent they may achieve and how likely they are. Our work has concentrated on OSPF-BGP interactions. We start with a description of the theoretical aspects of those interactions in chapter 5.1, where we explain in more detail what effects may occur and what the causes are. Moreover we specify our proceeding and the questions we like to answer when we are going to confirm our theory in the practical part. To better distinguish between effects on routers inside a particular AS and outside of it, we introduce the following naming conventions: We call changes in BGP routing IBGP (internal BGP) changes, if they are restricted to routers within that AS, and EBGP (external BGP) changes, if they were observed outside the AS. In the same way we denote the corresponding routers IBGP routers and external BGP routers. In chapter 5.3 we finally come to the practical part of our work, were we analyse the OSPF-BGP interactions which were generated in several simulation scenarios. These scenarios were performed in a single network environment, generated in the SSFNet simulator. But before we come to the analysis, we give a short overview in chapter 5.2 about the add-ons that had to be implemented to enable our examinations and the way the BGP best path selection algorithm was implemented in SSFNet.

5.1 Causales and Effects

Having defined our understanding of OSPF-BGP interactions, we are now going to deal with the impact these interactions have on BGP routing. The first question to answer is: What may change in BGP routing when IGP reacts on AS internal topology changes by adjusting its routes? Besides the fact that prefixes may be withdrawn or reannounced if prefix announcing BGP routers get dis- or reconnected, we are aware of four prefix’s path attributes that may change in case that IGP shortest path metrics change: The NEXT_HOP, AS_PATH, MED and COMMUNITY attributes. This is best explained by an example. Regarding figure 5.1 we see a network scenario of four ASes interconnected among each other. AS 1 learns a prefix A from AS 2 and AS 3 via routers R2, R3 and R4, and announces that prefix to AS 4 via router R1. Let us assume that router R1 and R5 decided to set the NEXT_HOP attribute for prefix A to R3’s address, because the BGP best path selection algorithm selected R3 as next hop due to the IGP shortest path metric tie-breaking
5.1. CAUSES AND EFFECTS

Figure 5.1: The context of IGP-BGP interactions

rule. Then traffic destined for prefix A is sent via R3 to AS 2. If the shortest path from R1 and R5 to R3 gets disconnected somehow, forcing the IGP to recalculate shortest paths, the periodically prefix checking mechanism on R1 and R5 may eventually detect that R2 or R4 have a better IGP metric for prefix A. In this case R1 and R5 switch the NEXT_HOP attribute value to R2’s or R4’s address. If R4 is the new next hop, they also have to change the AS_PATH attribute for prefix A, since R4 learned A from AS 3 instead of AS 2. In addition, if the system is configured to set MED attributes according to IGP metrics then we may also observe an update for prefix A with a changed MED attribute sent to AS 4 by router R1. Note that this update is only sent if the IGP metric changed, independent from any NEXT_HOP or AS_PATH attribute changes on other AS internal routers. That means although router R1 may have changed the next hop for A due to an increased IGP metric for R3, the new shortest path may have the same metric than the one before. On the other hand, there may be an alternative path from R1 to R3 having a better IGP metric than the paths to R2 and R4 but a worse metric than the former shortest path to R3. Anyway, if a MED change is not accompanied by a next hop or AS path change in some AS, it won’t have any impact on BGP routing. In the end we yet mentioned the BGP COMMUNITY attribute [6] that may be affected by IGP metric changes. [9] describes its utilisation as informational, transitive attribute to indicate the location where a prefix was learned from. This information is passed down to neighbouring ASes where it can be incorporated in the BGP best path selection process. Concerning our example network in figure 5.1, such a COMMUNITY attribute could be set by router R1 for prefix A when announcing the prefix to AS 4. Dependent on the type of the location (interconnection point (next hop), Autonomous System), R1 had to send an update to AS 4 if its next hop or AS path for pre-
fix $A$ changes. Thus changes that may be caused by communities are similar to changes caused by MEDs that were set according to IGP metrics, when IGP-BGP interactions effects them. In the following we therefore concentrate on the examination of next hop and AS path changes, since they are the main factor in these dynamics, and we will use IGP metric based MEDs to study the propagation of their effects into neighbouring ASes.

So lets now begin with a classification of NEXT_HOP and AS_PATH attributes. For two BGP speakers it is valid that, if they have identical NEXT_HOP attributes for the same route, they also share the same AS path for that route, but the reverse is not valid. Thus we can conclude that for an IBGP router a change of the AS_PATH attribute of one of its routes is always combined with a NEXT_HOP change for this route if the change was due to an IGP routing change. This assumption leads us to the following classification: For each prefix, learned from neighbouring Autonomous Systems, we can divide the IBGP routers of a given AS into *as-equivalence-classes* (ase-classes), such that for a particular route each router is associated with a class that exactly represents the neighbouring AS where the router sends traffic to when destined for that prefix. So if an AS learned a prefix from exactly one neighbouring AS, then all its IBGP speakers belong to the same ase-class. The ase-classes again can be subdivided into *next-hop-equivalence-classes* (nhe-classes) similar to the ase-classes. That means that for a particular prefix, each IBGP router belongs to the class that represents the AS border router to which traffic is forwarded when destined for the prefix. Thus if a neighbouring AS has more than one connection to the given AS, two IBGP router may be part of the same ase-class but they may have different nhe-classes. For an AS border router holds that it is part of the nhe-class of the IBGP routers that selected him as next hop. Hence each AS border router forms its own nhe-class and may be itself part of nhe-classes formed by other AS border routers within the same AS.

Having closer characterised the changes that BGP may encounter by assigning ase- and nhe-classes to the IBGP routers, the questions posed in the introduction to this chapter can be rephrased. Since we are now aware that BGP is indeed affected by IGP routing changes and how this is caused, we can examine the OSPF-BGP interactions on the basis of the equivalence classes. In the remaining part of this chapter we are therefore going to answer the questions: How and to what extent do the equivalence classes change? And to what extend does this influence BGP routing?

There are several causales for an alteration of equivalence classes produced by IGP routing. In figure 5.2 we show a schematic illustration of the events that lead to alterations. An IGP routing protocol like OSPF is always affected by link or router failures. Having detected failures that changes the current SPT, OSPF tries to compute alternative shortest paths which might have differing metrics. It is also possible that parts of the AS become completely disconnected. Somewhat later, BGP will notice the change. Dependent on its magnitude, more or less IBGP router have to change NEXT_HOP or even AS_PATH attributes for some prefixes due to disconnections or changed interior AS distances to AS border routers these prefixes were learned from. We mentioned before that periodically
5.1. CAUSALES AND EFFECTS

Figure 5.2: Causales of IGP-BGP interactions

checking best paths and IGP dependent attributes guarantees that changes of IGP metrics are considered when selecting best routes. But what happens, when an IGP like OSPF detected a topology change and is still in its convergence phase, while BGP tries to verify reachability of AS border routers and distances of their shortest paths by scanning the forwarding table? In this phase AS interior routes might be instable, i.e. due to the fact that not all OSPF routers may have up-to-date topology information, some of them could have compute incorrect shortest paths at the time a BGP scanner searches for IGP metric changes. In consequence of that, the IBGP instance on one or more routers might perform routing changes based on temporarily incorrect IGP metrics and route entries. These changes of course can affect BGP next hops or AS paths and might propagate through the network until they are again reversed when the next scanning cycle accesses the correct intra-domain routing information after OSPF finished its convergence phase. In the following we denote this particular event as **OSPF-BGP-clash**.

All those events described so far, result in changes in the equivalence class classification of the IBGP routers. In case these changes propagate to an AS border router called X, that learned the according prefixes from one of the affected IBGP border routers, then we possibly may also observe changes in BGP routing in neighbouring Autonomous Systems. But this is the case only if the relative neighbouring AS decided the AS border router X to be the next hop for the announced prefixes before. As response to the updates that X announces into a neighbouring AS, this AS may itself undergo equivalence class changes as indicated by the backward arrow in figure 5.2.

Up to now we explained which parts in the BGP routing process are affected by a change in AS interior routing. We furthermore demonstrated the causales for these interactions. It remains to deal in greater detail with the effects that the interactions may have on global routing. That means, how can a neighbouring AS or a chain of neighbouring ASes be influenced by local IBGP next hop or AS path changes if these attributes don’t serve as best path selection criteria? Figure 5.3 illustrates the effects that may propagate through a series of Autonomous Systems caused by IGP-BGP interactions. We always assume here that the local equivalence class changes propagate to an AS border router X, that learned at least one prefix from one of the affected IBGP border routers and that IGP shortest path
metrics are used as MEDs. Under these conditions we have to distinguish between two cases concerning the effects: In the first case, \( X \) may change its nhc-class for some prefixes without changing the ase-class. If the interior distance to \( X \)’s new next hop router has changed, compared to the distance to the old one, then \( X \) has to announce a new MED into the neighbouring ASes for all those prefixes learned from the old next hop router before. In case \( X \) is next hop router of a neighbouring AS \( A \) for such a prefix, then some IBGP routers in that AS may change their next hops (NH) and AS paths for that prefix (1a) in case they find an alternative best entry point to a third AS offering a better MED, or \( A \) is connected via multiple links to the former AS and there is an alternative best entry point to that AS, due to MED comparison, but it announces another AS path for that prefix. If an alternative best entry point to the former AS could be found that announces the same AS\_PATH attribute for the prefix (1b) then of course the next hop changes only.

In the second case, \( X \) may additionally change its ase-class for a prefix. That means that the next hop for that prefix changes which may or may not involve a MED change, depending on the interior distance to \( X \)’s new next hop router, and that the AS\_PATH attribute for that prefix changes too. Although the AS\_PATH attribute is not specified as best path selection criterion in [1], BGP developer like Cisco however make use of it. The selection algorithm in [5] for example defines a rule which prefers the route with the shortest AS\_PATH. If this rule is applied, a change of the AS\_PATH attribute may have extensive consequences. If \( X \) is next hop router of a neighbouring AS \( A \) for a prefix whose attributes changed, then there are three cases to distinguish: Some IBGP router of AS \( A \) may change the next hop as well as the AS path for that prefix (2a)) if there is a third AS offering better AS\_PATH or NEXT\_HOP attributes or, if the former AS is connected via multiple links to \( A \) and there is an alternative best entry point to the former AS offering a better but different path. On the other hand, if there is a second entry point to the former AS, offering the same as path, some IBGP routers of AS \( A \) may only change the next hop (2b)). In a last case (2c)) there may be routers in AS \( A \) that do not change their next hop but have to accept the changed AS path, announced by router \( X \), since there is no better alternative.

**Figure 5.3: Effects of IGP-BGP interactions on global routing**
5.2 ADDITIONAL SSFNET ENHANCEMENTS

All these effects can propagate through a series of attached Autonomous Systems even if these ASes don’t announce IGP metrics as MEDs since the most BGP routers use the AS-PATH attribute as a criterion in the best path selection algorithm. In the following we are therefore going to examine in which extent the causals and effects presented so far can occur. For that purpose we are concentrating our analyses on a single scenario, already introduced in chapter 3.2 and figure 3.4. Via the SSFNet simulator we are going to simulate link and router failures in the “US AS”, observe the resulting impacts that OSPF-BGP interactions will cause in this AS as well as in the dummy ASes, and try to estimate their magnitude. Particular attention we will pay for OSPF-BGP-clashes.

To perform our intention we had to make some changes in the SSFNet code before. These changes are described in the next section. To proceed with the documentation of our simulation results please go on with section 5.3.

5.2 Additional SSFNet Enhancements

We already mentioned that SSFNet featured no dynamic IGP before we developed SSF.OS. OSPFv2. For that reason SSF.OS.BGP4’s best path selection process did not cover the tie-breaking rule described in [1] chapter 9.1.2.1 b) which recommends to prefer the path with the lowest IGP metric to the BGP next hop. To add this rule to the selection algorithm and to assure that dynamic changes in IGP routing are eventually accounted for in BGP, at least at regular intervals, we made several changes to the BGP4 package and implemented a special timer.

First of all, an IGP metric had to be added to the BGP peer information. Since all data about a BGP peer is stored in class PEERENTRY we added the following variables to that class: An integer ospfv2Cost to keep the metric of the shortest IGP path to that peer and a boolean variable unreachable to indicate whether the peer is still reachable via IP or not. To make use of this information we had to augment the best path selection process. In SSF.OS.BGP4 this process is implemented in method compare() of class ROUTEINFO, a class that contains all information about a particular BGP route. Taking a ROUTEINFO as parameter compare() returns the decision of the process, preferring one of both route infos or declaring them to be identical. Having made little changes to it, the selection algorithm now operates as follows: First it compares the route’s degree of preference which is set to the configured LOCAL_PREF value or, if not specified, to 100 minus the length of the route’s AS_PATH attribute. The one with the higher value is preferred. In case both routes have the same preference and both were received from BGP speakers in the same AS, their MULTI-EXIT-DISCRIMINATOR (MED) attribute is compared. The route with the lower MED is preferred. In case that one route has a MED and the other doesn’t, SSF.OS.BGP4 prefers the one having a MED. The next tiebreaker was modified. Before, it was the announcing peer’s type. If one route was announced from an external peer and the other from an internal, the route received from the external peer was pre-
ferred. We instead added a combination of peer type and IGP metric comparison. Now the \texttt{ospfv2Cost} values in the according \texttt{PEERENTRY} classes are compared and the route with the lower value is preferred. This tiebreaker is also applied if one or both peers are external peers. Since OSPF metrics are defined to be greater than 0, we can assign 0 to \texttt{ospfv2Cost} in case the peer is external. Thus the preference of an external router and the preference of the router with the lowest IGP metric in case both routers are internal is guaranteed in a single query. But there is another reason why we assign an IGP metric to external peers which we discuss later. If at this time none of the two routes was preferred, the last tie-breaking rule is either the announcing peer’s id or a randomly chosen number. In our simulations we always used the peering routers id. The smallest id determines the route to prefer, otherwise both routes are declared to be identical.

In a last step, we had to assure that reachability and IGP costs are determined and maintained in the \texttt{PEERENTRY} objects during a simulation run. Therefore we firstly altered the \texttt{init()} method in \texttt{BGPSESSION}, the protocols main class, which establishes among other things \texttt{PEERENTRY} objects for the direct attached BGP neighbours. For each neighbour the IGP metric is assigned to \texttt{ospfv2Cost} in case an entry could be found in the IP FIB and the neighbour was not external. For external neighbours we also assigned an \texttt{ospfv2Cost} which is always 0. This is done in order that we can simply assign a MED, based on IGP metrics to each route. Since SSF.OS.BGP4 was implemented to favour routes having a MED over routes having not, we simply assure in this step that routes, learned from an external peer are still favoured over otherwise identical routes learned from an internal peer if both are forwarded to a third AS and MED values based on IGP metrics are assigned. In addition, if no entry could be found in the IP FIB and the neighbour is internal, the \texttt{unreachable} bit is set. Since the configuration of all other peers is done before the simulation is started, it does not make sense to determine IGP metric and reachability for those peers in method \texttt{config()}. Hence this process is postponed until a full relationship was established and a decision must be made whether a route, learned from such a neighbour should be installed in the local RIB (Loc-RIB). This decision process is implemented in method \texttt{decision_process2()} which was altered so that each time it is called, \texttt{ospfv2Cost} and \texttt{unreachable} are determined for each neighbour as in \texttt{init()}, and only those \texttt{ROUTE-INFOs}, when announced by internal peers, that are reachable at the time the method is called, are involved in the process.

Eventually it comes the time to send route withdrawals or advertisements to some peers. In case routes are to be advertised and the advertising router is an AS border router (the receiving neighbour is external), we assign a MED attribute to these routes having the value of their announcing peer’s \texttt{ospfv2Cost} field. We don’t need further checks here since all advertised routes come from reachable peers as determined before by the decision process and for external announcing peers it is guaranteed that their MED is 0 as explained before. Since now changes in IGP metrics may have an influence on route preference and MED values, a quick response on these changes is necessary. One way of maintaining IGP metrics and reachability of internal peers is to use a timer which scans the peer data at regular
5.3. INTERACTION ANALYSIS

We implemented such a timer as inner class of BGPSession called SCANNER-TIMER. At time intervals of 60 seconds it checks all peers whether they are still reachable via the IP FIB and whether their shortest path metric has changed in case they are internal peers. This process is the same as in methods init() or decision_process() described before. Thereafter the Loc-RIB is scanned route by route. For each route, each reachable peer’s Adj-RIB-In (see chapter 2.2.2) is searched for an alternative route and if one could be found the best path selection process decides whether the route in the Loc-RIB must be replaced due to a change of one or more best path selection criteria. If the route is replaced by an alternative or the advertising peer became unreachable and there was no alternative, SCANNER-TIMER removes the route from Loc-RIB and adds it to a list of Loc-RIB changes. If an alternative route could be selected, it is installed in the Loc-RIB and recorded in the list of Loc-RIB changes. This list then serves as parameter for method decision_process() which is responsible for applying the changes to the Adj-RIB-Out databases and disseminating updates to the concerned peers (see [1] chapter 9.1.3).

Finally we yet had to implement a “configurable link failure”. The current SSFNet release supports only a rudimentary link layer implementation. The links for example, can not be configured to fail or recover during a simulation run, and although there is an opportunity to configure packet dropping on NICs (Network Interface Cards) with a given probability, this probability is however static and can’t be changed during simulations. Thus it was not possible to simulate dynamic link failures and recoveries by setting packet dropping probability to 1 and 0 by turns. We therefore introduced a new subattribute fail for attribute link in the dml-language. fail has two atomic subattributes from and until specifying the link’s failure and recovery time. Each link attribute may have several fail subattributes specifying several fail intervals. These intervals get stored in the SSF.NET.LINK class. From there they can be called by any link type instance. Since in our scenarios ptplink links are used only, we only adjusted the ptplink type instance SSF.NET.PTPLINKLAYER. It now fetches the fail intervals one by one from LINK when pending. Each time a packet is supposed to be delivered to the peering NIC, the current simulation time is determined and checked against the current fail interval. Only if the simulation time is not within such an interval, the packet is delivered.

5.3 Interaction Analysis

Having explained the theory behind the interactions between an IGP like OSPF and BGP, and having prepared the SSFNet simulator to be capable to perform simulations of OSPF-BGP interaction causes and effects, we are now able to study the magnitude of those interactions considering a sample network. A simulation scenario that showed the effects in all of their details would imply to simulate a considerable part of the global Internet, because of their propagation. But that would have been a too big venture since we cannot
simulate such a huge network in all of its necessary details. We therefore restricted our scenario to a small, abstract fraction of a global routing system, instead of simulating a series of complex, neighbouring ASes, and we concentrated on the origin of the effects and not on their propagation.

This section has two main parts. In the one we study the impact of single, AS internal link fails on BGP (5.3.2), in the other we repeat this studies with the AS internal routers (5.3.3). Before, we give short outline about the scenario configuration (5.3.1). The section ends with a summary of aspects that would be worth to consider in future work.

5.3.1 An OSPF-BGP Scenario

At the end of chapter 3.2 we introduced a scenario that we are now using to simulate OSPF-BGP interactions: The “US Scenario” (see figure 3.4 on page 27). To better understand the simulation effects we first of all describe here shortly the scenario’s configuration. The network is composed of 7 Autonomous Systems. AS 2 till AS 7 are “dummy ASes” consisting of a single router only. They are connected via links 61-76, the inter-AS links, to AS 1, which we call the “US AS” since its shape is a simplified adaption of several U.S. ISP’s backbone networks. These inter-AS links are configured with a propagation delay of 10 milliseconds each and the NICs, connecting inter AS links with routers have a bitrate of 100 Megabytes/second. The remaining network structure build up the “US AS”. It consists of twelve “cities”, i.e. fully meshed POP subnetworks connected via links 0-39, the inner city links, which do not have any additional propagation delay. Since the distances between routers within a city are usually much smaller than inter-state and all the more inter-continental distances to other cities or ASes, the delay here can be seen as negligible. Each NIC, connecting to link 0-39 is provided with a bitrate of 1 Megabyte/second. Finally the city subnetworks are interconnected via links 40-60, the inter city links, configured with propagation delays, randomly chosen from a set of 1 to 6 milliseconds and NIC bitrates of 10 Megabyte/second. Even if those values might not reflect the real conditions exactly, their proportion however approximate the proportions in real life and we don’t need a millisecond time resolution since BGP and OSPF converge in seconds.

The protocol instances were set up as follows: On each router belonging to the “US AS” an SSF.OS.OSPFv2 protocol session was configured to run on the router’s NICs (Network Interface Cards) connecting to inner or inter city links. All sessions were configured with the default OSPF parameter values specified in table 4.1 on page 39 and to build a single OSPF backbone area. For the connecting links we chose OSPF metrics of 1 for each inner city link (0-39) and 10 for each inter city link (40-60) according to [24] which proposes to use metrics dependent on link delays or transmission time. On all routers, including those located in the dummy ASes, the modified (see section 5.2) SSF.OS.BGP4 protocol was set up. Insight the “US AS” the BGP routers were configured to peer with each other such that a fully meshed IBGP net was build. The EBGP peering sessions, that means the
5.3. INTERACTION ANALYSIS

Peering sessions between AS 1 routers and the routers in the dummy ASes, were built according to the inter AS connections. Concerning the configuration of the BGP timer intervals, we accepted for each router the suggested values described in [1] chapter 6.4. Having set up OSPF and BGP, we are now able to simulate BGP prefix’ NEXT_HOP attribute changes caused by OSPF routing alteration. However to also cause AS_PATH attribute changes in this way, we have to introduce prefixes announced by different ASes such that the “US AS” can select the best path for these prefixes from a choice of neighbouring ASes. For that reason we injected three different prefixes into AS 5, AS 2 and AS 3 respectively. This bundle of prefixes and attributes was the same for all these ASes so that the choice made by “US AS” routers, which AS to prefer when selecting the best path, only depended on intra-domain routing. Any external AS path changes caused by “US AS” intra-domain routing changes concerning these prefixes will be therefore observed in AS 4, AS 6 and AS 7 only. This is the startup configuration for all further simulations analysed in the next sections.

5.3.2 Analysis of Single Link Failures

Having prepared our simulation scenario as described in section 5.3.1, we begin with the examination of OSPF-BGP interactions caused by single link failures. For each internal link in our test AS (the “US AS”), that are links 0 - 60 in figure 3.4, we created a simulation run where the link failed for a particular period of time and then recovered again. The failure was initiated after a period of time, long enough to led the participating routing protocols finish their setup, that is the convergence phase after that the protocol instances on all routers are in a stable state for the first time. We estimated setup and failure time to be 500 seconds, so that the failure began at time 500 seconds and ended at time 1000 seconds, and measured the interaction impacts at both times. The results are depicted in figure 5.4. The thickness of the link lines have no meaning concerning the results, they were inherited from figure 3.4 to better illustrate the different types of links (inner city, inter city, ...).

Following the legend, we notice at first view that there are surprisingly many interior links (approx. 50%) whose failure lead to changes in BGP routing. But this is not really surprising, given that most of them affect NEXT_HOP attributes on IBGP routers only. Such a change is quite likely since an interior link failure always results in IGP shortest path changes which again are likely to result in distance changes between IBGP routers and their prefix advertising AS border routers. If the distance changes and there is an alternative AS border router, now having a better (lower) distance and announcing the same prefixes, then IBGP next hop changes can be observed. The effect of such changes is shown in table 5.1. Overall there have been 31 links which forced IBGP routers to change AS internal next hops for at least one prefix. The left column of table 5.1 indicates the number of IBGP routers who had to change NEXT_HOP attributes in response to a link failure. The next shows the percentage of these routers at the total number of routers.
Figure 5.4: Impact of each single link failure on BGP routing

forming the AS. In the last column you see how many links caused the amount of routers,

<table>
<thead>
<tr>
<th>affected routers</th>
<th>percentage of AS</th>
<th>quantity of causing links</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.702 %</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>5.404 %</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>8.106 %</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>10.808 %</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>13.510 %</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>16.212 %</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>21.621 %</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>27.027 %</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3.580</td>
<td>9.677 %</td>
</tr>
</tbody>
</table>

Table 5.1: The degree of IBGP next hop changes caused by single link failures

indicated in the left column, to change next hops. It is remarkable that less than 10% of the “US AS” routers are on average affected by BGP next hop changes caused in this way. The explanation for that is in the connectivity of the AS. The denser the AS is connected
to the inside and to the outside, the less routers are affected by a single link failure, since a link is part of less IBGP connection paths. Because the connectivity of our test AS is quite high, a value of 10% is thus realistic.

We explained in section 5.1 that under certain conditions, next hop changes can involve AS path changes. For links 16, 35, 41, 43 and 53 those conditions eventuate. This becomes clear when regarding figure 5.5 where we classified the test AS routers according to their ase- and nhe-classes for the prefixes we injected into AS 2, AS 3 and AS 5 before (see section 5.3.1). The ase-classes are denoted with X, Y and Z, and the corresponding nhe-sub-classes are numbered. In case link 16 or 35 fails, we observed only one router (H1) changing its next hop for the injected prefixes. Because H1 switches its next hop router from D1 to B2, it changes also the ase-class from X1 to Z2. But since it is not a border router this equivalence-class change will not propagate to other ASes. The like holds for link 41 where router E1 and E3 switch from class Z1 to X1. Yet in case of a failure of link 43 or 53 we can observe an AS path change on a border router for the first time. In this case routers H1 and H2 both change their equivalence-classes from X1 to Z2. But although AS border router H2 changes next hop as well as MED (next hop IGP metric) and AS path for the injected prefixes, there is no response from AS 4, 6 or 7 on that since they all are multi-linked to the “US AS” and prefer another entry router. Because of better
MEDs, AS 4 prefers border router C1 (MED is 0) and AS 6 and 7 prefer border router K2 (MED is 12). Since H2 announced a MED of 22 before the link failed (path-course: links 36,53,43,16) and a MED of 23 after (path-course: links 34,58,31,50,6), this explains why we don’t see any AS path changes outside our test AS. But in case of link 43, 53 and 55 we do see external next hop changes, that means one or more dummy ASes are switching entry points to the “US AS”. When link 43 or 53 fail, AS 4, 6 and 7 have to change their next hop for AS 2, since the previous next hop H2 changes to a more expensive shortest path to that AS (former path: 36,53,43,16. new path: 34,52,24,41,0), and K2 offers a better MED due to a shortest path (23,55,44,16), having the same metric than the one used before. Similar effects could be observed when link 55 failed. AS 4 and 6 also had to change their next hop for AS 5 from K2 to H2. In addition, AS 6 also changed, in the same way, the next hop for the injected routes since it learned them from AS 5.

Although we have seen that a lot of links cause IBGP next hop changes when failing, it is remarkable that only a few of them involve AS path changes too, or next hop changes in neighbouring ASes. There is no general explanation for these facts, since they are caused by several different factors. Three of these factors apply to the OSPF-BGP interactions and can be observed in our scenario:

- **The network’s topology:** How are ASes connected among each other and how are they connected internal? What is the degree of redundancy of these connections?

- **The use of IGP metrics as MEDs:** Whether or not the IGP metric of the shortest path to the border router of which a prefix was learned is used as MED if this prefix is announced to another neighbouring AS.

- **The configuration of link metrics:** According to which criteria are link metrics assigned? Which path becomes the shortest?

As described before, we only see IBGP AS path changes on routers E1, E3, H1 and H2. In chapter 5.1 we mentioned that the precondition for an AS path change is always a next hop change, so that according to figure 5.4 about 50% of the links are out of the question for such a change. If an IBGP router has to change its next hop router for a particular prefix, it may choose a router who learned that prefix from another AS. In this case the AS path for the prefix changes on the IBGP router, since traffic destined for that prefix is now sent to another AS via the new next hop. The decision whether to select such a next hop strongly depends on the network’s topology and the configuration of the IGP link metrics. In particular it depends on the following factors: The number of neighbouring ASes announcing the prefix, the number of AS border routers connecting to these ASes, the subset of the ASes that these border routers select as next hops when announcing the prefix to the interior BGP routers, the IGP distance between the border and the interior BGP routers and any best path selection rule following the IGP distance comparison in case the comparison yielded no preference (for instance the comparison of the routers BGP identifier).
5.3. INTERACTION ANALYSIS

The quantity of these factors and their interaction makes it hard to predict the number of IBGP routers that experience an AS path change due to a single link failure. In the “US Scenario” for example, there are three ASes (AS 2, 3 and 5) announcing an injected prefix, but only two of them (AS 3 and 5) are chosen by the “US” AS border routers to be next hops for that prefix as depicted in figure 5.5. This is explained by the BGP best path selection algorithm running on router A1 and D1 which applies the lowest-BGP-identifier rule to choose among AS 2 and AS3 respectively AS 5. Since the border router in AS 2 has coincidentally the highest BGP identifier, AS 2 is not selected as next hop. In case now that link 41 fails, router E3 changes its next hop for the injected prefixes from router A1 to router D1, and thus changes the AS path. It does so since D1 now has the shortest IGP distance to E3 among all other announcing border routers. The same holds for router E1 besides that router D1 was not chosen as next hop router due to the shortest distance but because of its BGP identifier. This is because after the failure of link 41, router E1 had the choice between D1 and B2 as next hop since both have a shortest distance of 22 to E1. The AS path change is therefore performed due to the coincidence that D1 has the lower BGP identifier which makes E1 prefer him. If B2 had a lower BGP identifier than D1, we would not have seen an AS path change on E1.

We see that there are different reasons for an IBGP router to change or not to change the AS path attribute for a prefix, in case it has to change its next hop. Thus it is not possible to simply reduce the likelihood of such changes to the IGP distance between IBGP routers and their prefix announcing border routers and the number of border routers announcing different AS paths for those prefixes.

It is comparably difficult to predict the number of links in an AS, which cause neighbouring ASes to change their next hops or AS paths for prefixes learned from that AS when those links fail. It depends on the network’s topology as well as on the configuration of link metrics and the use of IGP metrics as MEDs. AS 4, 6 and 7 for example all use the same path (via routers H2,H3,G1,D3,D1) through the “US AS” to get to AS 2. They do so, since they all have a connection to router H2 and H2 announces the best MED among all other possible candidates for becoming next hop router (C2, K2 and L3). The last is due to the use of IGP shortest path metrics as MEDs and due to H2 having the shortest path to AS 2. Thus the probability, that a single link failure in the “US AS” disconnects AS 4, 6 or 7 from AS 2 and forces them to change their next hops or even their AS paths is reduced, since they use a minimum of links to get to AS 2. On the other hand there are some cases where neighbouring ASes are not at all affected by internal link failures due to connectivity. This is the case for AS 4 concerning the injected prefixes which it learned from AS 3. Since AS 4 sends traffic to AS 3 via the external links 68 and 64, any “US AS” internal link failure does not affect this connection. But even in the case that an AS internal link failure leads to a prefix’s next hop change on an AS border router that announces the prefix into neighbouring ASes, it does not have to inevitably involve a next hop or AS path change in those ASes. Because firstly, an AS border router could not be a next hop router for any of its neighbouring ASes, as it is the case for the injected prefixes.
announced by H2 into AS 4, 6 and 7, and secondly the MED could not change at all or only slightly, i.e. not enough for a neighbour to change the next hop. This is for example the case for link 16 or 36, whose failure forces router H2 to increase the MED for prefixes learned from AS 2 by one. But after the change the MED remains still the best for the neighbouring ASes, since no other border router announces the same or a better MED to AS 4, 6 or 7. Similarly, if link 50 fails, router H2 still announces the best MED to AS 6 and 7 for any prefixes learned from AS 3, since the MED did not change although H2’s next hop did (from B2 to C1).

Last but not least we observe that there are no external AS path changes in our scenario. This is evident, now that we have analysed the IBGP AS path and external next hop changes in more detail. The external AS path changes are limited to the injected prefixes, since the scenario was composed in the way that each AS can reach each other only via the “US AS”. Since all of them are directly connected to it, there is no alternative. For the injected prefixes it is valid that AS 4 prefers router C1 as next hop router and therefore does not change its AS path if an “US AS” interior link fails as explained before. AS 6 and 7 both set their next hop to K2, but this router is never affected by an IBGP AS path change. Hence any “US AS” interior AS path change does not propagate to these ASes. So the only case that is left and that could raise an external AS path change in these ASes is case 1a) from figure 5.3. That means after a next hop change for the injected prefixes on router K2, the MED increases and AS 6 and 7 are forced to switch their next hop router and so may switch their AS path. But since the alternative next hop routers (H2 and L3) both are in the same ase-class (see figure 5.5) as the former next hop K2, there is no external AS path change. In addition, the next hop change on router K2 does not affect router H2 or L3.

Finally we can conclude that it is not possible to estimate the impact of single, AS interior link failures on EGP routing. All the effects described in section 5.1 may happen or not. Their occurrence depends much on the topological circumstances, the use of IGP metrics as MEDs and the configuration of the individual IGP link metrics as shown in the exemplifications before. The AS border routers thereby play an important role. Their number determines the number of alternative entry and exit points. It depends on them, how many alternative AS paths are advertised into an AS (see figure 5.5) and to how many ASes they provide these paths. Further more, the number of neighbouring ASes, using a particular AS border router as next hop is important when estimating the magnitude of BGP routing changes outside the causing AS.

Thus a possibility to reduce those changes would be to connect certain AS border routers within an AS via one or more redundant paths, having the same IGP metric than the best paths between these routers. In case of a single link failure, these paths would reduce the probability of a next hop switch or a MED change on the concerned border routers and would thus reduce also next hop and AS path changes outside the AS. Concerning our “US scenario” for example, we are able to prevent the external next hop changes by
5.3. INTERACTION ANALYSIS

simply adding an additional link between routers D3 and G1, G1 and H3 and J2 and K3 having OSPF cost 10. However there is yet another factor that influences the impact of an OSPF-BGP interaction. In this section we only dealt with separate, single link failures and we did not cover the impact of several links failing at the same time. Under the given conditions it is obvious that the more links failing within an AS the more likely are BGP changes outside that AS. We analysed multiple link failures by disconnecting single “US AS” internal routers from the network. The results are described in the next section. But before we come to single router failures, simulated in that way, there is one matter left which we did not go into yet. In section 5.1 we characterised a special cause for an equivalence-class change, the *OSPF-BGP-clash*. The fact that those clashes can occur, poses some questions which we like to answer before we terminate this section: With respect to a single link failure, how likely is an OSPF-BGP-clash? Are there any links in our scenario that induce such a clash? And what is the dimension of its impact? For single link failures these questions are simple to answer: The probability for an OSPF-BGP-clash is 0 and therefore no link failure induces a clash in our scenario. The explanation is not so simple. Initially we need to know, how much time OSPF requires to converge and which part of it is the critical time period where such a clash can occur. Figure 5.6 shows the overall OSPF convergence time in the “US AS” for each single link failure. We can see that our propositions made about OSPF convergence time in chapter 4 also apply to a more complex topology, that is OSPF converges within the estimated time of 50 seconds plus possible propagation delays. As explained in chapter 4.2.1, an OSPF protocol session configured with default parameter values (see table 4.1) requires between 30 and 40 seconds to detect a link failure. After that it determines the execution of the SPT-calculation and the IP FIB update to take place after a delay of 5 seconds (spf-delay). If there are routers at expiration of detection time and spf-delay, that didn’t get all topology information updates yet, due to propagation delay or the asynchronous link failure detection times, these routers may compute incorrect SPTs and routing tables. When they eventually get the missing updates, a second calculation is allowed to be executed unless 10 seconds have past since the first (spf-holdtime). If the BGP scanner checks the IP forwarding table within this 10 second time period, we might have an OSPF-BGP-clash. Usually the second wait period provides enough time to generate all topology updates and to propagate them throughout the whole network. If not, further calculations are performed again 10 or more seconds later at a time, depending on the time the updates arrive. For now we assume that two calculations, scheduled as described before, are sufficient to calculate stable routes. That means we determine the critical time period when the BGP scanner may access wrong route information to be the first spf-holdtime, which is 10 seconds. The period begins when the first IP table update terminated, because before the table did not change since the topology change occurred. This time is more than 35 seconds after the failure at the earliest (30 seconds detection time, 5 seconds spf-delay) and more than 45 seconds after it at the latest (40 seconds detection time). Thus we may be confronted with a clash if the overall OSPF convergence time exceeds 45 seconds (detection
CHAPTER 5. SIMULATING OSPF-BGP INTERACTIONS

time plus spf-delay plus holdtime). If we have a look on figure 5.6, we see that 21 links each cause at least one OSPF protocol instance to converge after more than 45 seconds when the link fails. Furthermore we see that our assumption about the number of SPT calculations is confirmed since OSPF converges always in less than 50 seconds. Concerning the 21 links, we can find from figure 5.4 that 11 of them don’t have an impact on BGP routing. For the remaining 10 (link 16,23,24,43,47,50,52,54,57 and 58) on some routers BGP scanners could in fact be observed, scanning the IP FIB during the critical period but this had no impact. The reason for that is the fact that during each simulation run only a single link failed. This failure will be detected by the two routers being connected via that link. But even if the OSPF instances on these routers detect the failure asynchronously, such that their updates force other OSPF routers to perform two SPT calculations, the IP table holds the correct route information already after the first calculation. This is because both routers send updates informing about the same link failure so that the second update has no new information and therefore does not alter the results from the first SPT computation. On the other hand a clash may of course have an impact on BGP if several links fail at the same time or if one or more routers fail. For the general case we can state that the critical period for an OSPF-BGP clash lasts at least 10 seconds if OSPF was configured with default values. For each simulation run the interval at which the BGP scanner timer expired was set to 60 seconds on each router. Thus we get a probability of at least 1/6 (16.6%) for an OSPF-BGP-clash on a router that temporarily calculated incorrect IGP route information during its convergence phase. The occurrence of such routers depends much on the network’s topology and on the sort and number of components that fail as we have seen. In the next section we are therefore going to simulate multiple link failures

![Figure 5.6: OSPFv2 convergence time in US topology, simulating single link failures](image)

5.3. INTERACTION ANALYSIS

to study this effect in more detail.

5.3.3 Analysis of Single Router Failures

So far we have seen that a single, AS internal link failure might have a certain impact on BGP, AS internal as well as on neighbouring ASes. But since this change to the overall topology was quite little, so was the impact. We could not observe any AS path changes in the neighbouring ASes, and there were only a few external next hop and IBGP AS path changes. From table 5.1 we see that only about 10 % of the network’s composing routers were on average affected by IBGP next hop changes, and OSPF-BGP-clashes could not be observed due to the fact that the failure was limited to a single link. We therefore suppose that a bigger change to the topology, like several links failing at the same time can have a bigger impact on BGP. But before we examine such scenarios in more detail, it remains to determine how many and which components should fail. A common procedure for networks is to shut down or reboot routers for servicing. Such a topology change has a much bigger impact on routing than a single link failure. In fact SSFNet does not provide an opportunity to shut-down or reboot a router, but we can simulate a router failure by letting fail all links that connect to him. For these reasons we decided to simulate single router failures via multiple link failures instead of planless selecting links for a collective failure. To study those failures, we prepared our “US scenario” as described in section 5.3.1. For each router in the “US AS” (that are routers A1 to L3 in figure 5.5) we created a simulation run were this router gets completely disconnected from all its neighbors 500 seconds after simulation start and again reconnected 500 seconds after the “failure”. The proceeding was kept similar to the single link failure simulations. In figure 5.7 we present a description of the results. Each router is classified by the impact its failure has on the rest of the network. Following the legend and comparing with the link failure classification in figure 5.4, we see that the impact of the most router failures can be derived from the failing links that connect to them. For example, if a router has only egress links whose failure doesn’t change BGP routing, then the router also doesn’t affect BGP routing as well when failing. In addition, a router failure has no impact on BGP if all of its egress links only affect the router’s BGP instance when failing. This is the case for router F2 and its egress link 57, because the failure of link 57 causes IBGP next hop changes only on router F2, which is declared to be down and therefore doesn’t participate in the routing process anymore. For the majority of the remaining routers however it is valid that the degree of changes caused by their failure is the sum of the degrees of changes of its egress links. That means, if any of the egress links cause for example an IBGP AS path change when failing, then the router causes this change too when it fails. Excluded from that are again changes which only affect the router itself, as it is the case for router H1 and its link 35. We have seen in the last section that a failure of link 35 forced only H1 to change an AS path, and we therefore don’t see any AS path changes when router H1 fails. From this it follows that, besides of the “US AS” border routers, we described the
impact of each single router failure on BGP routing in section 5.3.2 already. The AS border routers of course present the more interesting part, since their failure also affects one or more EBGP connections. In table 5.2 we list the BGP changes and the number of routers that are affected when an individual border router fails. From the left to the right, the columns have the following meaning: The router that failed, the overall number of IBGP next hop changes on the remaining “US AS” routers, the overall IBGP AS path changes on these routers, the overall next hop changes on “US AS” external routers, the overall AS path changes on the external routers and last the number of routers that were affected by those changes. In the last column we distinguished IBGP from AS external routers. First the number of IBGP routers, thereafter, separated by a slash, the number of dummy AS routers affected by a change. On the first glance we see that router I3’s failure generates a very high rate of IBGP next hop changes. This number is greater than the number of all BGP routes installed on all “US AS” routers together. From that we can conclude that some of the routes must have changed for more than one time, and that possibly some changes were made due to OSPF-BGP clashes. To find out more about possibly redundant changes, we filtered our data in the way that we counted only the last change of each single route. If this last change set back the route entry in the BGP table to the values it had before the first change, then all changes made to that route entry
5.3. INTERACTION ANALYSIS

<table>
<thead>
<tr>
<th>router</th>
<th>IBGP next hop</th>
<th>IBGP AS path</th>
<th>ext next hop</th>
<th>ext AS path</th>
<th>aff routers</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>89</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>32/2</td>
</tr>
<tr>
<td>B2</td>
<td>115</td>
<td>42</td>
<td>39</td>
<td>12</td>
<td>16/4</td>
</tr>
<tr>
<td>C1</td>
<td>22</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>7/2</td>
</tr>
<tr>
<td>D1</td>
<td>282</td>
<td>49</td>
<td>45</td>
<td>6</td>
<td>36/6</td>
</tr>
<tr>
<td>H2</td>
<td>204</td>
<td>10</td>
<td>45</td>
<td>0</td>
<td>36/6</td>
</tr>
<tr>
<td>I3</td>
<td>499</td>
<td>218</td>
<td>75</td>
<td>18</td>
<td>36/5</td>
</tr>
<tr>
<td>J3</td>
<td>40</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>8/4</td>
</tr>
<tr>
<td>K2</td>
<td>28</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>12/3</td>
</tr>
<tr>
<td>L3</td>
<td>66</td>
<td>4</td>
<td>21</td>
<td>0</td>
<td>36/5</td>
</tr>
</tbody>
</table>

Table 5.2: The impact of single "US AS" border router failures on BGP

<table>
<thead>
<tr>
<th>router</th>
<th>IBGP next hop</th>
<th>IBGP AS path</th>
<th>ext next hop</th>
<th>ext AS path</th>
<th>aff routers</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>89</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>32/2</td>
</tr>
<tr>
<td>B2</td>
<td>33</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>9/1</td>
</tr>
<tr>
<td>C1</td>
<td>22</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>7/2</td>
</tr>
<tr>
<td>D1</td>
<td>107</td>
<td>9</td>
<td>11</td>
<td>0</td>
<td>36/2</td>
</tr>
<tr>
<td>H2</td>
<td>61</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>23/4</td>
</tr>
<tr>
<td>I3</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3/0</td>
</tr>
<tr>
<td>J3</td>
<td>40</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>8/4</td>
</tr>
<tr>
<td>K2</td>
<td>28</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>12/3</td>
</tr>
<tr>
<td>L3</td>
<td>7</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>7/1</td>
</tr>
</tbody>
</table>

Table 5.3: The in effect impact of single "US AS" border router failures on BGP

were redundant and therefore haven’t been counted. After the data was filtered, we got the adjusted results, listed in table 5.3. We see from that table that the in effect impact in two-thirds of the cases is much smaller than the real amount of changes depicted in table 5.2. Since that was a quite surprising result, we also checked the changes caused by the other, non-border routers. But differences in the numbers of real and in effect changes could only be detected on router D4. There, 8 of 54 IBGP next hop changes were redundant, the rest of the changes were all necessary. The question that arises from that is: Can the huge amount of needless changes and needless affected routers be explained by an OSPF-BGP-clash? We cannot answer this question in detail, since for that purpose an accurate observation of each OSPF and BGP routing change on all routers would be necessary. But we can estimate where a clash is quite possible. To do so, we use the method described in section 5.3.2. On the basis of the OSPF convergence times we can
estimate which router failure may have caused an OSPF-BGP-clash. The convergence times for each single “US AS” router failure are shown in figure 5.8. The routers are numbered in the way that the numbering is conform with the order of the labels in figure 5.7. As it is the case for the single link failures before, we see that OSPF converges within a time period of 50 seconds. That means on each remaining router, two SPT calculations are executed at most when a single router fails. Furthermore we see that the OSPF convergence time after a router failure lasts longer than 45 seconds for router B2 (5), D1 (11), D4 (13) and H2 (24), but it is less than 45 seconds for router I3 (28), J3 (31) and L3 (37). We therefore have to assume that the three routers last mentioned required only a single SPT calculation to update their IP tables with the correct new routes. A check of the corresponding log-files validates this assumption. The check also showed that neither BGP changes were made nor that BGP updates were sent during the OSPF convergence phase. From that it results that there was no OSPF-BGP-clash when these routers failed and that the redundant changes must be traced back to another reason. For the rest of the failure-scenarios where we could observe redundant BGP routing changes (failure of router B2, D1, D4 and H2) we cannot say exactly whether, and if so how much of such changes were generated due to a clash. But we can estimate and determine the number of routers that might be hit. In section 5.3.2 we calculated that the probability for an OSPF-BGP-clash on a single router is $\frac{16}{6}$% if its OSPF instance requires two SPT-calculations. After a closer analysis of the log-files we could isolate those routers. Table 5.4 indicates the results. For each router whose failure generated redundant BGP routing changes possibly by a clash, we determined the number of routers where OSPF was forced to execute a second SPT-calculation (second column). From these routers we calculated the average number of those who might be hit by a clash, supposing a probability of

![Figure 5.8: OSPFv2 convergence time in US topology, simulating single router failures](image-url)
5.3. INTERACTION ANALYSIS

<table>
<thead>
<tr>
<th>failing router</th>
<th>number of routers calculating 2nd SPT</th>
<th>possible clashes (estimated)</th>
<th>possible clashes (real)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2</td>
<td>36</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>D1</td>
<td>24</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>D4</td>
<td>36</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>H2</td>
<td>9</td>
<td>1.5</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5.4: Estimated and determined number of routers probably affected by an OSPF-BGP-clash

16.7% (third column). At last we counted the number of routers whose BGP scanner in fact scanned the IP table before the OSPF instance calculated the SPT for the second time (fourth column). Regarding table 5.4 we notice by far the most routers that scanned the IP table during OSPF’s critical routing phase when D4 failed. Therefore it is quite likely that BGP relied on incorrect IGP routing information in that simulation run. This assumption is substantiated by a look into the corresponding log-file which revealed that 4 of the affected routers changed 7 BGP routes during the critical phase. The other 4 yet changed 2 routes after all. Further investigations yielded that the other failure-scenarios not only hold considerably less affected routers but their routers also changed less routes during the critical time (between 1 and 5). If we assume now that there was in fact an OSPF-BGP-clash after router D4 had failed, then its impact was marginal since we only saw 8 redundant IBGP next hop changes of 54 altogether as already mentioned before. The remaining changes were all not redundant. Thus in case of the failures of B2, D1 and H2 we can assume that the fraction of the redundant BGP routing changes which were caused by OSPF-BGP-clashes is negligible. It is even possible that none of them come from a clash, because the quantity of the possibly affected routers is too small. But what are the origins of those redundant routing updates then? To answer this question, we had to dig into the details of the OSPF and BGP routing tables and to follow each single BGP update that was generated after the failure of the concerned routers in their particular simulation runs. We had to determine when, why and which updates were originated from which router, “US AS” internal as well as external, and what they triggered at which time and router to exactly explain the observed BGP routing behaviour. From these details we then could deduce the mechanism that is behind it. But the complexity of this work would make a description going far beyond the scope of this chapter. Anyway there might be many reasons for the strange BGP behaviour. It is for instance remarkable, that all routers whose failure caused redundant changes which can not, or just poorly be explained by OSPF-BGP-clashes are AS border routers. This might be an evidence for the possibility that the redundant changes are caused by external (inter-AS) link failures, since all AS border routers are connected to dummy ASes via those links and we simulate router failures by disconnecting the router from all of its direct attached neighbors. On the other hand, before we began our work, the SSFNet protocols were implemented to run
in networks with static topologies. Single link or even router failures were not allowed for and as far as we know, it had been never tested what happens with the IP, TCP and BGP protocol instances if all connections to a particular router failed. But these are speculations which deserve more intensive study.

5.3.4 Future Work

Although we have seen that AS internal routing changes may have an impact on BGP, we did not analyse to what extend this impact may affect the global routing. It therefore remains for future work to analyse OSPF-BGP interactions in a more complex AS-level network, where a greater number of Autonomous Systems is interconnected among each other in a more complex way than in our “US scenario”. In such scenarios one could study how far BGP routing changes, caused by an IGP, do propagate on AS-level, and what impact they induce in the ASes they ingress. However it would be necessary then to also use more complex AS topologies than our dummy ASes. In addition a single complex AS topology like our “US AS” reveals only a single aspect of link and router failure impacts. It remains to correlate our results with OSPF-BGP interactions observed in different AS topologies to classify the dimension of the impacts. On the other hand there are besides different topologies, different protocol implementations and configurations too. It remains to explain whether or not the redundant updates we have seen, are ascribed to some opaque OSPF-BGP interaction and how far protocol parameters and implementations are involved in that. It is possible for example to reduce the probability of an OSPF-BGP-clash by stronger synchronising the OSPF link failure detection times on the concerned routers. If we increase the sending rate of the Hello Packets (say Hello-Interval set to 1 second), the connection failures were detected almost synchronously (see chapter 4.2), and thus a single SPT-calculation could be sufficient on all OSPF routers to calculate the correct new routes. Concerning BGP we used the default parameter values to configure the various BGP instances (see section 5.3.1). Other values used here could have effects on an OSPF-BGP interaction and could provide more information about the reason for the redundant BGP changes. Is for example the BGP convergence process disturbed, if a BGP instance detects a connection failure before OSPF or the scanner timer detects it due to a low BGP Hold Time? Last but not least the different BGP implementations deployed in todays Internet can differ much in their routing behaviour due to additional features and concepts deviating from [1]. The SSFNet BGP implementation, which we used in our simulations, is quite conform with [1]. Therefore an implementation that better meets the behaviour of real world BGP implementations could provide more realistic information about IGP-EGP interactions.
5.4 Conclusion

In the course of this chapter we drew up some causales for IGP-BGP interactions and described the possible effects that those causales may generate. Subsequent to that we used a network simulator where we generated a network consisting of several ASes to study the extent that the effects may reach. Within a particular AS we created single, independent link and router failures and watched their impact on the BGP routing inside and outside of that AS. For the link failures we found out that although 50% of the AS internal links caused BGP route changes when failing, most of them only generated NEXT_HOP attribute changes on IBGP routers that had no effect on AS external routing. Furthermore due to a densely connected AS, only about 10% of the routers were on average affected by these next hop changes at each of those link failures. A little minority of the failing links also caused IBGP AS path changes or NEXT_HOP attribute changes on routers outside the AS. In addition, none of these AS external next hop changes generated AS path changes. Although the big amount of IBGP limited next hop changes could be easily explained by the IGP shortest path metric changes, however for the other effects we found out that it is hard to predict their occurrence, since they are determined by a wealth of factors, like for instance the network topology, the number and connectivity of AS border routers, the use of IGP metrics as MEDs and the configured IGP link weights. An AS path change for instance always implies a next hop change but not vice versa. Therefore there might be less AS path changes but never more than next hop changes. So when we see only a few next hop changes, it is quite likely that there are even less or none AS path changes. The AS border routers thereby play an important role. It depends on their number and their connectivity how many alternative egress and ingress points an AS has and how many alternative AS paths are advertised into that AS and provided to other ASes. A single AS interior link failure therefore can only affect BGP routing outside the AS if the link is part of a transit path through that AS. That means the link failure forces an AS border router to change a route for which this border router is also next hop of one or more neighbouring ASes. If several ASes use the same transit path or the transit path makes not use of any interior link (the AS border router is the only hop on the way through the AS), then the probability for an external next hop or yet AS path change is little. A single AS interior link failure furthermore cannot cause an OSPF-BGP-clash since all OSPF updates sent inform about the same link failure, and therefore each OSPF router already holds the correct new IGP routes after the first SPT-calculation and provides BGP with incorrect IGP route updates at no time.

The impact of OSPF-BGP interactions were more extensive when single router failures were simulated, but in most cases their effects did not deviate from the effects that single link failures caused. That means that they only caused BGP changes which were already caused by the failure of any of their attached links. This is obvious because a single router failure is tantamount to the failure of all of its egress links, and therefore its impact is the sum of all the effects that these links cause when failing. Since our link failure analy-
sis didn’t include inter-AS links, the impact of AS border router failures could not be predicted in that way. Although we expected a bigger impact on BGP than caused by single link or non-border router failures, the impact was quite surprising. We detected that in some cases the router failure caused an unusual high rate of redundant BGP routing changes, AS internal as well as external. A total of two thirds of the AS border routers caused redundant changes. Nearly all of the AS path changes, internal as well as external were redundant. The number of redundant next hop changes in some cases were several times over the number of in effect changes. Further investigations yielded that only one non-border router failure caused redundant BGP changes. But the number of these changes was little and they can possibly attributed to an OSPF-BGP-clash. Concerning the affected AS border routers, we could exclude an OSPF-BGP-clash from the possible causes for the redundant changes in half of the cases. For the rest of these cases such a clash could be assessed to be quite unlikely for a reason. It remains for future work to determine whether the redundant BGP routing changes can be attributed to any of the protocol implementations we used in the network simulator or whether there was an OSPF-BGP interaction that we have not traced. In any case the AS border routers remain a weak spot in routing. Only if BGP routes on those routers are affected by IGP routing changes, the changes may propagate through a series of neighbouring ASes. Otherwise the impact remains limited on the local AS. Since BGP prefixes are usually learned from AS border routers again, one could reduce the impact of IGP routing changes on the global routing system by connecting the AS border routers within an AS via multiple parallel paths. That means, if AS border router \( R_a \) learned a prefix from AS border router \( R_b \), located in the same AS, via a shortest path \( p_c \) and \( p_c \) becomes disconnected, then \( R_a \) may switch to a parallel path \( p_d \) connecting to \( R_b \) and having the same IGP metric as \( p_c \), without changing its next hop or AS path. Hence one could reduce the impact of AS internal link or non-border router failures on the global routing system to a minimum, if all AS border routers, belonging to the same AS, were connected among each other in that way.
Chapter 6

Conclusion

In this work we studied the effects of coupling intra-domain routing elements with inter-domain routing using a network simulator, considering OSPF and BGP as examples. An interaction between both is already given via IGP shortest path metrics which are calculated by OSPF and utilised by BGP. The core problem of this coupling are the differing objectives of OSPF and BGP. OSPF is an intra-domain routing protocol that is intended to discover shortest paths. BGP is an inter-domain routing protocol that emphasises reachability between ASes.

The traditional approach to interaction between the protocols is to deploy an update triggered hot potato routing in BGP according to [1] and [14]. However, AS transit traffic is often routed suboptimally through the AS that it traverses, since BGP is an inter-domain routing protocol and does not provide a globally optimal shortest path routing through an AS.

Therefore ISPs modernised this approach and extended it by the export of their IGP metrics to the neighbouring ASes [23]. This enables ASes to use optimal ingress points when sending transit traffic to neighbouring ASes. However, this now causes intra-domain routing information, that was formerly restricted to its AS, to spread throughout other ASes. Therefore local routing instabilities can propagate through the global routing domain.

Using a network simulator we showed the possibility of reaching OSPF convergence times in the range of milliseconds (see chapter 4.2.4). However, this presumed (a) changes to OSPF architectural constants, (b) changes to the OSPF configurational constants, and (c) a homogeneous deployment of this modified OSPF throughout the AS. This significantly reduces the possibility that intra-domain routing instabilities propagate outside the respective AS.

Moreover we examined failures of links and routers within an AS (see chapter 5). First, we showed that, within the simulated AS, the routing stability increases with the level of interconnectivity between the AS-interior routers. Second, when the interconnections between the AS-interior border routers are protected from failures and changed path
metrics, intra-domain routing changes are not propagated outside the AS anymore. We suggest redundant connections with equal IGP path metrics between the border routers within an AS.

Eventually, there is, on the one hand, a demand for optimal routing. On the other hand, resources in terms of routing table memory, i.e. link bandwidth for routing updates and route processor capacity, are scarce and therefore do not allow a non-hierarchical approach to routing. Our simulations showed that the current trade-off between both directions are well performing. However, we expect a possible implementation of the changes suggested above to show significant performance increases by effectively eliminating transient routing instabilities.
Bibliography


[9] Olivier Bonaventure: Common utilizations of the BGP community attribute, IETF Internet

[10] Sally Floyd, Vern Paxson:
**Difficulties in Simulating the Internet**, AT&T Center for Internet Research at ICSI (ACIRI), February 2001


[14] Renata Teixeira, Aman Shaikh, Tim Griffin, Jennifer Rexford:
**Hot Potatoes Heat Up BGP Routing**
paper in submission

[15] **Internet Domain Survey, Jan 2003:**

[16] Jon Postel:
Request for Comments 791: **Internet Protocol**, September 1981,
http://www.rfc-editor.org/rfcsearch.html

[17] Craig Labovitz, G. Robert Malan, Farnam Jahanian:
**Internet Routing Instability** in Proceedings of the ACM SIGCOMM ’97, Nice, France, MD, August 1997

[18] J.C Mogul, J.B. Postel:

[19] T. Cormen, C. Leiserson, R. Rivest:
**Introduction To Algorithms**, The MIT electrical engineering and computer science series, 1990

[20] **IP Routing Protocol Commands: show ip ospf through version:**
[21] D. Oran:

[22] Neil Spring, Ratul Mahajan, David Wetherall:
**Measuring ISP Topologies with Rocketfuel**, Computer Science and Engineering, University of Washington Seattle

[23] Craig Labovitz, G. Robert Malan, Farnam Jahanian:
**Origins of Internet Routing Instability**, Department of Electrical Engineering and Computer Science, University of Michigan

[24] John T. Moy:

[25] **OSPF Commands**:
http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/np1_cr/1rospf.htm#xtocid2457237

[26] **OSPF Update Packet-Pacing Configurable Timers**:
http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t4/ftospfct.htm#xtocid17

[27] John T. Moy:

[28] Dirk W. Jacob:
**OSPFv2 Testsuite for the SSFNet simulation framework**, http://www.d-jacob.net/diplom/index.html

[29] G. Malkin:

[30] **Renesys Corporation**:
http://www.renesys.com

[31] **Scalable Simulation Framework**:
http://www.ssfn.org/homePage.html

[32] Christos Gkantsidis, Milena Mihail, Ellen Zegura:
**Spectral Analysis of Internet Topologies**
[33] W. Richard Stevens:  
TCP/IP Illustrated, Volume 1: The Protocols, ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES, 1994

[34] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger:  
Towards Capturing Representative AS-Level Internet Topologies, Tech. Rep. CSE-TR-454-02, EECS Department, University of Michigan, 2002

[35] C. Alaettinoglu, V. Jacobson, H. Yu:  
Towards Milli-Second IGP Convergence, IETF Internet Draft, November 2000

[36] Bernard Fortz, Jennifer Rexford, Mikkel Thorup:  
Traffic Engineering With Traditional IP Routing Protocols

[37] J. Postel:  

[38] University of Oregon Route Views Project:  
http://www.antc.uoregon.edu/route-views/.