



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Institut für
Technische Informatik und
Kommunikationsnetze

Christelle Aline Gloor

Chronos: Finding the configurations recipe for fast convergence

Semester Thesis 227-1572-02L
March 2017 to June 2017

Tutor: Prof. Dr. Laurent Vanbever Co-Tutor: Ahmed Elhassany

Abstract

The goal of this thesis is the comparison of the convergence times of two different routing protocols (OSPF and BGP) in a datacenter fat-tree topology. It describes the many challenges that can arise in trying to accurately compare convergence times of different protocols. Seven different types of failures were induced for each experimental setup and the resulting convergence times analysed. Intermediate routing paths were observed during convergence for the different setups. It was found that OSPF generally scales poorly with the network size while for BGP the convergence times are far more dependent on the locality of the failure. For larger network sizes BGP might be the superior protocol, but more research is required to clarify how a larger number of prefixes advertised in the BGP network might affect the duration of individual path calculations.

Contents

1	Introduction	9
1.1	Motivation	9
1.2	The Task	9
2	Theory	10
2.1	Datacenter Topologies	10
2.1.1	Fat-Tree	10
2.2	Routing	11
2.2.1	Link State Routing Algorithm	11
2.2.2	Distance Vector Routing Algorithm	12
2.2.3	Comparison: LS & DV Algorithms	13
2.2.4	Propane	13
2.3	Measuring Convergence Time	14
3	Experiment Design	16
3.1	Flow Generation	16
3.2	Measurement	16
3.3	Failures	16
3.3.1	Upstream Edge-Aggregate Link	17
3.3.2	Downstream Edge-Aggregate Link	17
3.3.3	Upstream Aggregate Router	18
3.3.4	Downstream Aggregate Router	18
3.3.5	Core Router	19
3.3.6	Link Failure Methodology	19
3.3.7	Router Failure Methodology	19
3.4	Logs	20
3.5	Path Analysis	20
3.6	OSPF	20
3.7	BGP	22
3.8	Timers	22
3.8.1	OSPF	22
3.9	Additional Experiments	23
3.9.1	Larger Destination Pod	23
3.9.2	Larger Destination Pod With Less Connections	23
3.9.3	More Prefixes	24
3.10	Procedure	24
4	Results and Analysis	26
4.1	Log Analysis	26
4.2	Path Analysis	26
4.3	Main Discussion	27
4.3.1	General Tendencies	27
4.3.2	Individual Experiments	28
4.3.3	Core Router	42
4.4	Comparisons	44
4.4.1	OSPF: Enabled vs. Disabled Logs	44
4.4.2	OSPF Single Area	48

4.4.3	OSPF Multiple Areas	54
4.4.4	BGP	59
4.4.5	Adding More Prefixes to BGP	66
5	Conclusion and Outlook	72
5.1	Conclusion	72
5.1.1	Fat-Tree-Connections	72
5.1.2	OSPF With Multiple Areas	73
5.2	Considerations for similar work	73
5.2.1	Bidirectional Forwarding Detection	73
5.2.2	Logging	73
5.2.3	Flow Observation	74
5.2.4	Incremental SPF Algorithm	74
5.2.5	Statistical Validity	74
5.2.6	Real Hardware	74
A	Setup	75
A.1	Linux	75
A.2	Mininet	75
A.3	MiniNExT	75
A.4	Minigenerator	75
A.5	Quagga	76
A.5.1	ECMP	76
B	Declaration of Originality	77

List of Figures

2.1	Common data center interconnect topology. Host to router links are GigE and links between routers are 10 GigE [21].	10
2.2	The fat-tree topology constructed for this thesis for $k = 4$	11
2.3	Traditional setup to measure convergence time in a network [19].	14
3.1	Depiction of a failure of an upstream edge-aggregate link, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.	17
3.2	Depiction of a failure of a downstream edge-aggregate link, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.	18
3.3	Depiction of a failure of an upstream aggregate router, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.	18
3.4	Depiction of a failure of a downstream aggregate router, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.	19
3.5	Depiction of a failure of a core router, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.	19
3.6	Area organisation on the fat tree for $k = 4$	21
3.7	The division of cores into the different areas and the additional links in the backbone for $k = 6$	21
3.8	Distribution of <i>hello packets</i> over a second with the <i>hello multiplier</i> set to 5. The blue arrow indicates the point of failure. The red arrows indicate the last <i>hello packet</i> it will have seen by the time the <i>dead-interval</i> timer has run out. One second minus this time is the time it takes to notice the router has failed.	22
3.9	Hello packets over a second with the <i>hello multiplier</i> set to 10. The blue arrow indicates the point of failure. The red arrows indicate the last <i>hello packet</i> it will have seen by the time the <i>dead-interval</i> timer has run out. One second minus this time is the time it takes to notice the router has failed.	22
3.10	The adapted topology of the fat-tree for $k = 4$	23
3.11	Comparison of the routing decisions made when failing a downstream edge-aggregate link.	24
4.1	Convergence Times after a failure of a link between an edge and an aggregate router upstream on the standard topology.	28
4.2	The path evolution from the failure until a new route to the node is found for BGP and $k=4$. The behaviour is equivalent for $k=6$ and $k=8$ for this protocol.	29
4.3	The path evolution from the failure until a new route to the node is found for OSPF with a single area for $k = 6$. The behaviour is the same for all the iterations and for all k	29
4.4	The two longer paths OSPF with multiple areas converges to in rare cases for $k = 4$	30
4.5	Convergence Times after a failure of a link between an edge and an aggregate router downstream on the standard topology for the different protocols.	30
4.6	Path evolution for BGP and $k = 4$ after a downstream failure of an edge aggregate link.	31

4.7	Path evolution for BGP and $k = 6$ after a downstream failure of an edge aggregate link.	31
4.8	Most observed path evolution for BGP and $k = 8$ after a downstream failure of an edge aggregate link.	32
4.9	One of the paths observed in which a routing loop occur for BGP and $k = 8$ after a downstream failure of an edge aggregate link.	33
4.10	Path evolution for OSPF with one area and $k = 4$ after a downstream failure of an edge aggregate link. One can nicely observe the longer path that is taken until the upstream edge router has finished its SPF calculation.	33
4.11	Convergence Times after a failure of a link between an aggregate and a core router upstream on the standard topology for the different protocols.	34
4.12	Path evolution for BGP on a fat-tree with $k = 4$ after an upstream failure of an aggregate core link. The behaviour is identical for the higher k values.	35
4.13	Convergence Times after a failure of a link between an edge and an aggregate router downstream on the standard topology for the different protocols.	35
4.14	Path evolution for BGP on a fat-tree with $k = 6$ after a downstream failure of an aggregate core link. The behaviour is the same for $k = 8$	36
4.15	Possible path evolution for OSPF with a single area on a fat-tree with $k = 6$ after a downstream failure of an aggregate core link.	36
4.16	Possible path evolution for OSPF with multiple areas on a fat-tree with $k = 8$ after a downstream failure of an aggregate core link. A routing loop is formed between a core and an aggregate router as they do not finish their SPF calculation simultaneously.	37
4.17	Convergence Times after a failure of an upstream aggregate router on the standard topology for the different protocols.	37
4.18	Path evolution for BGP on the fat-tree with $k = 4$ for a failure of an upstream aggregate router. One can clearly observe the 40ms it takes for the Quagga process on the aggregate router to terminate. The behaviour for $k = 6$ and $k = 8$ are identical.	38
4.19	Path evolution for OSPF configured with a single area on the fat-tree with $k = 4$ for a failure of an upstream aggregate router. One can clearly observe the 40ms it takes for the Quagga process on the aggregate router to terminate. The SPF calculation is only triggered once the OSPF hello timer has ellapsed (950ms). The behaviour for $k = 6$ and $k = 8$ are identical.	39
4.20	One of the intermediate paths observed for OSPF with multiple areas on a $k = 4$ topology for an upstream aggregate router failure.	40
4.21	Convergence Times after a failure of an downstream aggregate router on the standard topology for the different protocols.	40
4.22	Some observed routing loops for BGP on the downstream failure of an aggregate router.	41
4.23	Convergence Times after a failure of a core router on the standard topology for the different protocols.	42
4.24	The convergence process for BGP on a fat-tree with $k = 4$ for a core router failure. This behaviour is also observed for $k = 8$	43
4.25	The convergence process for BGP on a fat-tree with $k = 6$ for a core router failure.	43
4.26	Comparison of the convergence times for OSPF with and without logs enabled on the standard topology for an upstream link failure between an edge and an aggregate router.	44
4.27	Comparison of the convergence times for OSPF with and without Logs enabled on the standard topology for an downstream link failure between an edge and an aggregate router.	45
4.28	Comparison of the convergence times for OSPF with and without logs enabled on the standard topology for an upstream link failure between an aggregate and a core router.	46
4.29	Comparison of the convergence times for OSPF with and without Logs enabled on the standard topology for an upstream failure of an aggregate router.	46
4.30	Comparison of the convergence times for OSPF with and without Logs enabled on the standard topology for an downstream failure of an aggregate router.	47

4.31	Comparison of the convergence times for OSPF with and without Logs enabled (WOL) on the standard topology for a failure of a core router.	48
4.32	Comparison of the convergence times for OSPF with a single area for a failure of an upstream link between an edge and an aggregate router.	49
4.33	Comparison of the convergence times for OSPF with a single area for a failure of a downstream link between an edge and an aggregate router.	49
4.34	Comparison of the convergence times for OSPF with a single area for a failure of an upstream link between an aggregate and a core router.	50
4.35	Comparison of the convergence times for OSPF with a single area for a failure of a downstream link between an aggregate and a core router.	51
4.36	Comparison of the convergence times for OSPF with a single area for a failure of an upstream aggregate router.	51
4.37	Comparison of the convergence times for OSPF with a single area for a failure of a downstream aggregate router.	52
4.38	Comparison of the convergence times for OSPF with a single area for a failure of a core router.	53
4.39	One of the oscillating paths on the topology with an enlarged destination pod for OSPF with a single area and $k = 4$. This was observed for a failure of an upstream edge aggregate link where no packet loss was observed. This is why the path logging script never stopped recording as the system was in a valid state from the beginning and it gave a full view of the oscillation throughout the 30 seconds of the experiment.	54
4.40	Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream link between an edge and an aggregate router.	55
4.41	Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream link between an edge and an aggregate router.	55
4.42	Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream link between an aggregate and a core router.	56
4.43	Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream link between an aggregate and a core router.	57
4.44	Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream aggregate router.	57
4.45	Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream aggregate router.	58
4.46	Comparison of the convergence times for OSPF with multiple areas for a failure of a core router.	59
4.47	The most observed path evolution for BGP and $k = 4$ for a downstream failure of an edge aggregate link.	60
4.48	The backwards fault propagation observed for one iteration of BGP with the larger destination pod on $k = 4$	61
4.49	Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream link between an edge and an aggregate router.	62
4.50	Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream link between an edge and an aggregate router.	62
4.51	Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream link between an aggregate and a core router.	63
4.52	Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream link between an aggregate and a core router.	64
4.53	Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream aggregate router.	64
4.54	Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream aggregate router.	65
4.55	Comparison of the convergence times for OSPF with multiple areas for a failure of a core router.	66
4.56	Upstream failure of an edge aggregate link for BGP on the topology with $k = 4$ for different amounts of additional prefixes.	67
4.57	Downstream failure of an edge aggregate link for BGP on the topology with $k = 4$ for different amounts of additional prefixes.	68

4.58	Upstream failure of an aggregate core link for BGP on the topology with $k = 4$ for different amounts of additional prefixes.	68
4.59	Downstream failure of an aggregate core link for BGP on the topology with $k = 4$ for different amounts of additional prefixes.	69
4.60	Upstream failure of an aggregate router for BGP on the topology with $k = 4$ for different amounts of additional prefixes.	70
4.61	Downstream failure of an aggregate router for BGP on the topology with $k = 4$ for different amounts of additional prefixes.	70
4.62	Core failure for BGP on the topology with $k = 4$ for different amounts of additional prefixes.	71
5.1	An alternative connection scheme between the aggregate and the core routers. For flows originating at the outer pods and going into the centre pods and vice versa this lessens the path the error needs to propagate for a failure of a downstream link between an edge and an aggregate router.	73

List of Tables

4.1	Detection times at router failure for OSPF in milliseconds.	26
4.2	Median values for Figure 4.1 in seconds.	29
4.3	Median values for Figure 4.5.	30
4.4	Median values for Figure 4.11.	34
4.5	Median values for Figure 4.13.	35
4.6	Median values for Figure 4.17.	37
4.7	Median values for Figure 4.21.	41
4.8	Median values for Figure 4.23.	42
4.9	Median values for Figure 4.26.	45
4.10	Median values for Figure 4.27.	45
4.11	Median values for Figure 4.28.	46
4.12	Median values for Figure 4.29.	47
4.13	Median values for Figure 4.30.	47
4.14	Median values for Figure 4.31.	48
4.15	Median values for Figure 4.32.	49
4.16	Median values for Figure 4.33.	50
4.17	Median values for Figure 4.34.	50
4.18	Median values for Figure 4.35.	51
4.19	Median values for Figure 4.36.	52
4.20	Median values for Figure 4.37.	52
4.21	Median values for Figure 4.38.	53
4.22	Median values for Figure 4.40.	55
4.23	Median values for Figure 4.41.	56
4.24	Median values for Figure 4.42.	56
4.25	Median values for Figure 4.43.	57
4.26	Median values for Figure 4.44.	58
4.27	Median values for Figure 4.45.	58
4.28	Median values for Figure 4.46.	59
4.29	Median values for Figure 4.49.	62
4.30	Median values for Figure 4.50.	63
4.31	Median values for Figure 4.51.	63
4.32	Median values for Figure 4.52.	64
4.33	Median values for Figure 4.44.	65
4.34	Median values for Figure 4.54.	65
4.35	Median values for Figure 4.55.	66
4.36	Median values for Figure 4.56.	67
4.37	Median values for Figure 4.57.	68
4.38	Median values for Figure 4.58.	69
4.39	Median values for Figure 4.59.	69
4.40	Median values for Figure 4.60.	70
4.41	Median values for Figure 4.61.	71
4.42	Median values for Figure 4.62.	71

Chapter 1

Introduction

1.1 Motivation

The internet is without a doubt one of mankind's greatest creations. The ability to exchange information cheaply and quickly has enriched our culture, facilitated research, and made education and knowledge available to the whole world.

Since the beginning of the internet routing has been a core field of research. How do we make sure that the data we send to a specific destination actually reaches it? This is particularly important now that there are more destinations, from phones to smart appliances to industrial robots, connected to the internet than ever before. Meanwhile the sheer amount of data transferred increases every day as corporations rush to gather data to train their AIs, social eclipses all other forms of media, and the internet of things connects ever more devices. As such, efficient routing has become one of the main operational challenges of the net.

Big companies like Google and Facebook need to be able to host a vast amount of data in order to implement their business models. This is typically done by stacking servers together in what is called a datacenter. These structures can host hundreds of thousands of interconnected servers [5] tending to the needs of algorithms and costumers accessing websites all over the world.

While a lot of research has been done on routing in general, its role in datacenters with their specialised networks is still a relatively new field with many opportunities for novel insights.

1.2 The Task

In 2016 a compiler called *Propane*, which compiles high level policies for the BGP routing protocol from an abstracted language into individual router configurations, was released in a collaboration between Microsoft, Princeton, and UCLA. This facilitates the normally complex, time consuming, and error prone process of manually configuring routers. However, the choice of BGP as the designated protocol might come with some drawbacks. Specifically, as a distance vector protocol, BGP might not converge as quickly as an alternative link-state protocol, like OSPF, when a failure in the network occurs.

The goal of this thesis is to compare the OSPF and BGP routing protocols and their respective convergence times in a datacenter topology.

Chapter 2

Theory

2.1 Datacenter Topologies

Classically datacenters organized their networks by bridging the routers which were directly connected to the data hosts, called *edge* routers, in a tree like fashion with an *aggregate* router on top, which were in turn bridged to a set of *core* routers which served as the gateways between the different host clusters.

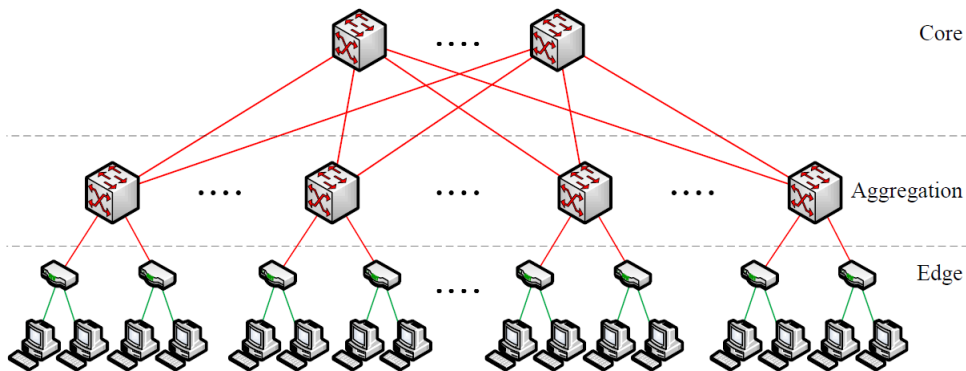


Figure 2.1: Common data center interconnect topology. Host to router links are GigE and links between routers are 10 GigE [21].

This design had some significant drawbacks. The higher up in the hierarchy the router is, the more load it needed to be able to carry. This leads to bandwidth bottlenecks at the core routers. In order to counter this non-commodity solutions were used for the crucial routers or oversubscription of some communication patterns in the network was accepted. Typically it was a balancing act between the cost of specialized routers and the optimal communication bandwidth. One way network operators avoid this now is to change the topology itself, build a large scale network from many small commodity-routers into a *fat-tree*. A fat-tree solves the problem of bottlenecks by having more bandwidth between routers higher up in the tree. Constructing a fat-tree can be achieved by organising multiple routers with multiple links between them for each branching, increasing the number of available paths between levels of the tree, which increases available bandwidth the higher up you go. Such a structure does not require specialised routers, so can be built using all identical, cheap routers. A fat tree is also *rearrangeably non-blocking* which means that for any communication pattern there will be a set of multiple paths so as to utilise all the bandwidth available between the hosts in the topology. [21]

2.1.1 Fat-Tree

The fat-tree used in this paper is structured in the following way:
A parameter k parametrizes the size of the tree. It must be an even number larger or equal to

four for the network to have the desired properties of redundant paths. K also describes the number of pods the aggregate and edge routers are grouped in.

The bottom most layer consists of k^2 hosts. Each pair of hosts is connected to one of $k^2/2$ edge routers in the layer above.

The second layer consists of k pods. Each pod consists a set of $k/2$ edge routers each with a link to each of a set of $k/2$ aggregate routers. A pod then forms a complete bipartite graph between sets of aggregate and edge routers.

The third layer consists of $(k/2) * (k/2)$ core routers. These connect the different pods to each other. Every aggregation router in each pod will connect to exactly $k/2$ core routers. This results in every core router having exactly one connection to each pod. Let n be the index of each aggregation router relative to its position in the pod $[0, 1, \dots, k/2 - 1]$ and m be the index of the core routers relative to each other $[0, 1, \dots, (k/2) \times (k/2) - 1]$. All the aggregation routers with index n will connect to the core routers with indices $m = n * k/2, n * k/2 + 1, \dots, n * k/2 + k/2 - 1$.

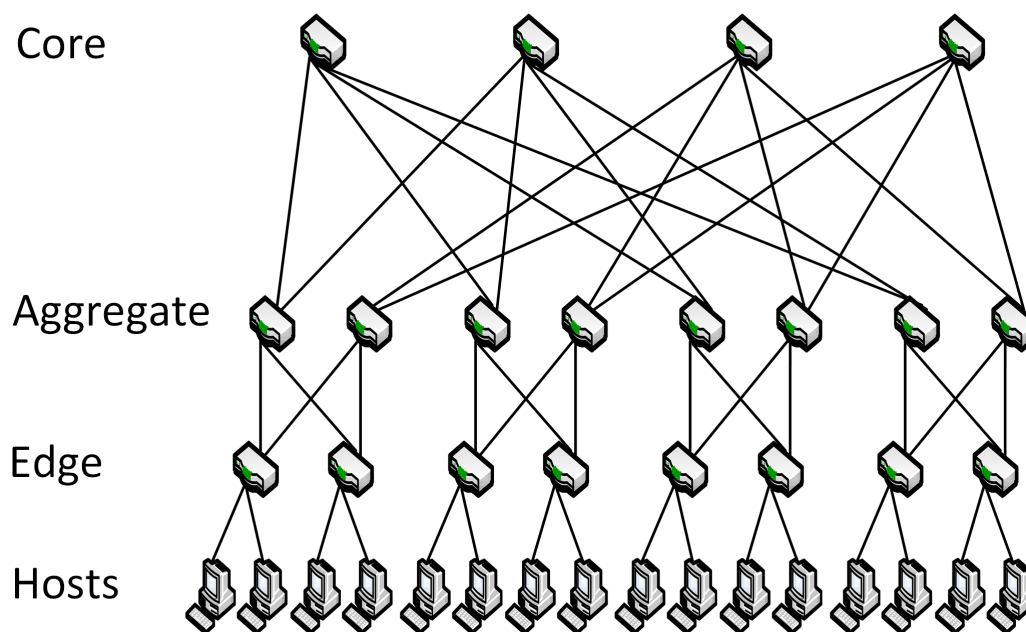


Figure 2.2: The fat-tree topology constructed for this thesis for $k = 4$.

2.2 Routing

Routers forward packets according to their internal forwarding table. This table matches IP addresses to a certain output port. Typically IP addresses are summarized according to a prefix of a certain length. The port whose entry has the longest prefix match for a given IP address is the one the router chooses to forward the packet to.

The forwarding table does not come pre-installed on the router. Networks are dynamic and there must be a mechanism for the routers to learn about changes as routers are added, removed, or other failures occur.

The mechanism responsible for this updating process are the routing algorithms. Such algorithms may behave in different ways; they can be global or decentralised, static or dynamic, load-sensitive or load-insensitive, etc..

A routing protocol is an implementation of the chosen routing algorithm running on the routers.

A routing protocol is an implementation of a specific routing algorithm run on the routers. Two main algorithms and their most popular implementations will be discussed in this paper.

2.2.1 Link State Routing Algorithm

These types of algorithms use globally disseminated information to calculate the best routes. They require every node in the network to have complete information about all the links in the

network and their costs. Link costs are usually set by the network administrator and can either have some relationship to the physical bandwidth of the link or be chosen due to other factor like link stability. If the costs are uniformly set the shortest path will be calculated.

Typically the global view is accomplished by having each node broadcast link-state messages to all other nodes in the network with the relevant information. This happens by using a link-state broadcast algorithm. The result is a network in which all the nodes have an *identical* map of the network, and so all the nodes will find the exact same least cost paths through the graph.

One of the most well known link-state algorithms is *Dijkstra's algorithm*. It computes the least-cost path from one specific node to to all other nodes in the network. It is iterative and after the j th iteration the least-cost paths are known to j destination nodes.

Open Shortest Path First (OSPF)

OSPF is a widely used protocol that uses flooding of link state information and Dijkstra's least-cost path algorithm to build the routing tables. Its most recent version is defined in RFC 2328 [18]. A router constructs a graph of its network (or AS, short for Autonomous System) and then runs Dijkstra locally with itself as the starting node.

Link state information is broadcast both whenever a change in the link state occurs and also periodically even if the state has not changed to make the algorithm more robust against failures. Link state information is broadcast both whenever a change in the link state occurs and at set intervals in order to make the algorithm more robust against failures. OSPF checks if links are operational by periodically exchanging so called *hello packets*. The protocol *dead-interval* is the time a router will wait between *hello packets* before declaring the link down.

To reduce the computational burden and messaging load on the network OSPF allows for hierarchical structuring into multiple areas. Nodes within an area limit their broadcasts and SPF calculations to other nodes within the same area. *Area border routers* (or ABRs) are specially designated routers which belong to more than just one area and are responsible for routing packets between them. Area 0 is the so called *backbone area* and contains all the ABRs.

2.2.2 Distance Vector Routing Algorithm

The distance vector algorithm is a iterative asynchronous and distributed algorithm. The nodes receive information only from their directly attached neighbours, this information is then used to calculate routing costs, which are then redistributed to the next neighbours. The process continues until no more information is exchanged between neighbours.

Each node starts by evaluating the cost to its directly attached neighbours. For all the other nodes in the network the initial cost is set to infinity to signify that the node does not yet know how to reach them. If at the end of the process a node still has infinite cost it is unreachable, as some nodes may be in the case of a partitioned network. The costs are stored in a *distance vector* and are sent to all direct neighbours. Upon receiving a neighbour's DV a node can now update its table by adding the cost to reach that neighbour to the costs in the DV. The neighbour with the lowest cost path to any given node is installed as the next hop in the forwarding table. This can potentially change multiple times until the optimal paths are found.

Generally the DV is updated if the node sees a cost change to its directly connected neighbours or when it receives an updated neighbouring DV.

Boarder Gateway Protocol (BGP)

The border gateway protocol is the standard inter-AS routing protocol in today's internet. It is specified in RFC 4271 [17]. Pairs of routers exchange routing information. This happens over a TCP connection which is atypical for a routing protocol. In the Internet BGP is typically used to establish communications between the different ASes. In a datacenter this model needs to be adapted as we want the different routers which would typically be placed in the same AS. To circumvent this the routers can each be placed in one AS containing only themselves. A BGP session which connects two routers in different ASes is called an external or eBGP session. Otherwise it is identified as an internal or iBGP session.

Instead of putting the routers and the cost to reach them in the announced distance vector BGP uses IP prefixes with each prefix representing a subnet. The routers then exchange their distance vectors with their neighbours as described in the previous section.

BGP evaluates the optimal routing path mainly by looking at the AS-path attribute. Whenever a prefix is passed through an autonomous system the autonomous system number is added to the AS-Path attribute. This is how BGP calculates the distance of a route. Also it can be used to avoid looping advertisements. If a router sees its own AS in the path it will reject the advertisement.

BGP has a lot more attributes which can be used to influence the routing decision. Specifically there is the possibility to add preference metrics to the routes.

Since in the internet BGP directly influences how the traffic flows between the different ASes there are a lot of possibilities to implement specific economic relationships. For example one AS can behave like a customer to another provider AS or two ASes might implement a peer relationship. To realize those policies BGP also allows to filter specific advertisements. A customer might for example not want to be the transit node to its provider and have to pay for traffic that does not originate or end in its AS.

In summary BGP is a very complicated protocol which allows for very specific steering of routing behaviour.

2.2.3 Comparison: LS & DV Algorithms

There are three main characteristics for which we want to compare the two algorithms:

- **Message complexity:**

Link state protocols requires full knowledge of the whole graph on each node. A change (or multiple cumulative changes) in the network will always require that the network be flooded with the new information. The volume of messages required to keep the network updated for such protocols is therefore very high and increases quickly with network size.

Distance vector protocols only exchange messages between direct neighbours and therefore scale much better with the network size.

- **Robustness:**

In the link state protocol each node computes its own forwarding table. So, even if bad information were to be broadcast the fault will stay somewhat localised since the SPF calculations are separated from each other.

For DV protocols bad information will be propagated much further along the network making it less robust in the case of faults.

- **Convergence Time:**

Common implementations of link state algorithms have a complexity of $O(n^2)$ where n is the number of nodes in the network. It requires the exchange of $O(|n||e|)$ messages where e is the number of edges in the network.

The convergence of a distance vector algorithm is much more dependent on the specific routing path and the position of the failure in the path. Routing loops can occur during convergences and it can suffer from the *count-to-infinity problem* which may have severe negative effects on the convergence time.

Exploring the convergence times of these protocols for the fat-tree topology described above is the goal of this paper.

[20]

2.2.4 Propane

Configuring networks router by router can be very error prone. Especially when complicated policies and rules are applied, since such network-wide policies need to be decomposed into device behaviours.

Propane aims to alleviate these issues. The behavioural rules are formulated as a collection of policies written in an abstracted high level language which then gets compiled to eBGP configuration files for each router. This abstracted language intentionally does not make any references to BGP-specific features.

Propane requires a description of the topology in XML using the *topology*, *node* and *edge* tags. The topology tag has an attribute to specify the AS number.

The node tags have a *name* attribute, used to identify the nodes to the edges and for the generated configuration files, and an *internal* attribute, which specifies if the node is part of the AS (*internal="true"*) or part of an external AS (*internal="false"*). One can also optionally specify the AS number for each node as an attribute, if omitted the generated eBGP configurations place each internal router in its own AS.

```
<!-- example topology described in XML -->
<topology asn="100">
  <node internal="true" name="A"></node>
  <node internal="true" name="B"></node>
  <node internal="true" name="C"></node>
  <node internal="true" name="D"></node>
  <edge source="A" target="B"></edge>
  <edge source="A" target="C"></edge>
  <edge source="B" target="C"></edge>
  <edge source="B" target="D"></edge>
  <edge source="C" target="D"></edge>
</topology>
```

The policies are written in a very intuitive way. A set of prefixes are matched using rules. Those rules can be combined with the boolean operators *and*: (&) and *or*: (+) or be given preference in respect to each other (*»*). E.g. *through(router1) » any* would mean that the path should prefer to flow over *router1* but if that is not possible it would not be dropped but routed over any available path. Some typical constraints are *end(router)*, *start(router)*, *path(router1, router2,...)*, *reach(router1, router2)* etc.

[10][23]

2.3 Measuring Convergence Time

Convergence is the process in which a network restores a stable routing state after a failure or general network change. The convergence time is defined as the time from set change until the routing paths have converged into error free paths. This means that if a flow is running through some router or link that fails, packet loss will occur while the network is converging. Observing this packet-loss can be used to measure the convergence time.

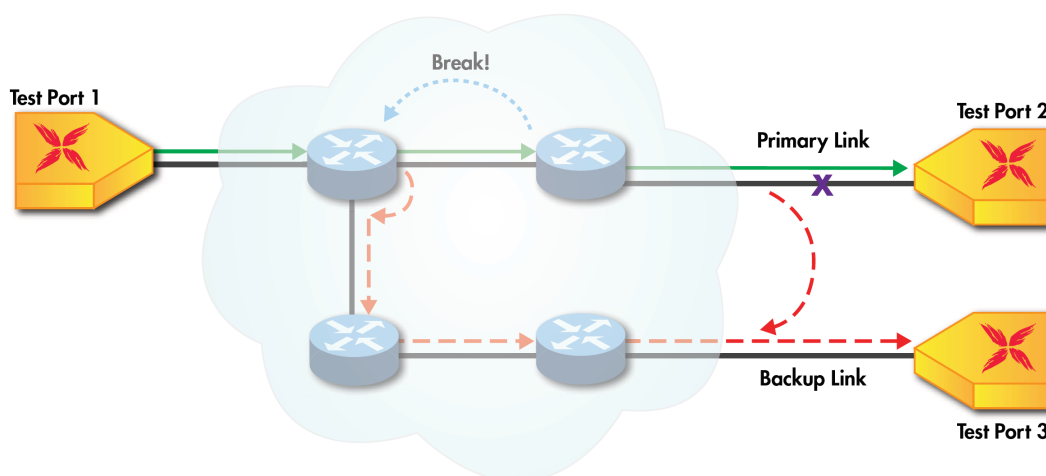


Figure 2.3: Traditional setup to measure convergence time in a network [19].

A flow between two hosts is chosen which results in a path of interest. Somewhere on this path a link or a router is failed. This will result in packet loss on the destination host while the network converges. As soon as the network has converged the flow will be rerouted to a stable path and the packets will start arriving at the destination host again. This moment can be recorded and the convergence time inferred [19].

Most routing protocols support some mechanism of load balancing which splits flows into multiple paths. When failing part of a flow to measure convergence time such mechanisms must be deactivated, otherwise a flow may have an immediate fallback and packet-loss would not necessarily occur.

It is also crucial to have a good time-resolution for this method. This can be achieved by sending the packets of the flow at short and stable intervals.

Chapter 3

Experiment Design

3.1 Flow Generation

UDP was chosen over TCP for the network protocol as the experiments only require a steady flow of packets and no additional features like flow control or error handling. The experiment duration was fixed to 30s and the failure is triggered after 3s to provide a comfortable margin around the ~1s startup time of the traffic generator.

During the experiment packets are transmitted by the source host at 1ms intervals, thus the resolution of the convergence measurements is approximately 1ms. Each packet is assigned consecutive sequence numbers that are included in their payload. The source and destination hosts log the send/receipt time and the sequence number of each packet. The destination host also logs the time elapsed between each received packet.

The convergence time was determined by processing the packet data on the destination host.

3.2 Measurement

The convergence time was calculated from the packet dumps using the following process: Packet dumps on the destination hosts are sorted by sequence number in order to avoid issues emerging from potential reordering of the packets. Periods of packet loss are obtained by looking for gaps in subsequent sequence numbers. The convergence time is equivalent to the duration of such periods and is calculated by taking the time from the last packet received before the loss to the first packet received after. To look at the convergence right after the failure in case of re-occurring events of packet drops which can happen in case of a routing loop, the drop around the failure time was chosen.

3.3 Failures

There are several distinct ways a failure can occur without partitioning the network. Such failures allow for convergence and may occur on:

- **Edge-Aggregate Links**
- **Aggregate-Core Links**
- **Aggregate Routers**
- **Core Routers**

Failing edge routers, or edge-host links, will separate hosts from the rest of the network completely and are therefore not relevant cases in this study since convergence is the measure of the time for host-to-host flow to be restored.

For the links and the aggregate routers there is a distinction between an *upstream* and a *downstream* failure. An upstream failure occurs when the failing router or link lies between the source

host and the core router in the routing path. Whereas a downstream failure occurs when the failing router or link is between the core and the destination host. When failing core routers there is no such distinction.

3.3.1 Upstream Edge-Aggregate Link

In this case a link between the edge router directly connected to the source host and an aggregate router fails. The expected behaviour will be for the affected edge router to switch to a different path provided by the protocol and direct the flow to an alternative aggregate router.

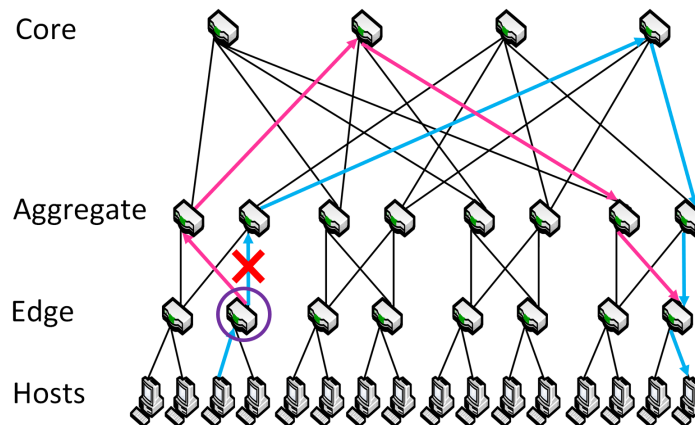


Figure 3.1: Depiction of a failure of an upstream edge-aggregate link, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.

3.3.2 Downstream Edge-Aggregate Link

Failing a downstream edge-aggregate link breaks the second to last step of the route, the edge router affected directly connects to the destination host and so an alternative path must include it still. The aggregate router on the link will attempt to find a new path to the edge router, from its position it must add at least two more hops to the path (ie. by going through another edge and aggregate router in the pod) and so should not chose it as an alternative path.

The core router will not be able to find an improved path since it only connects to the affected aggregate router in that pod. The upstream aggregate router has links to other core routers, however, those cores also only connect to the affected aggregate router, so it too is unable to find an improved path. Instead the error is expected to propagate all the way back to the source-adjacent edge router, which can push traffic to an alternative aggregate router which will have a shortest path equal in length to the original.

Due to the topology of the network this should be the the costliest possible link failure.

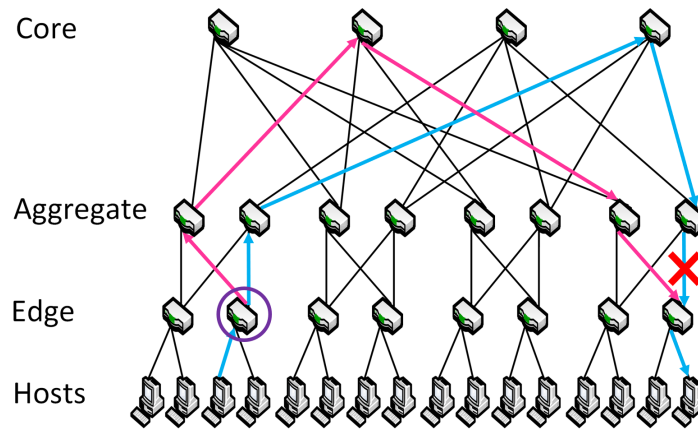


Figure 3.2: Depiction of a failure of a downstream edge-aggregate link, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.

3.3.3 Upstream Aggregate Router

This failure type causes a behaviour nearly identical to the upstream edge-aggregate link failure. The edge router can switch to an alternative route as soon as it is notified that the router is down. The only difference is the delay between the router going down and the adjacent routers realising.

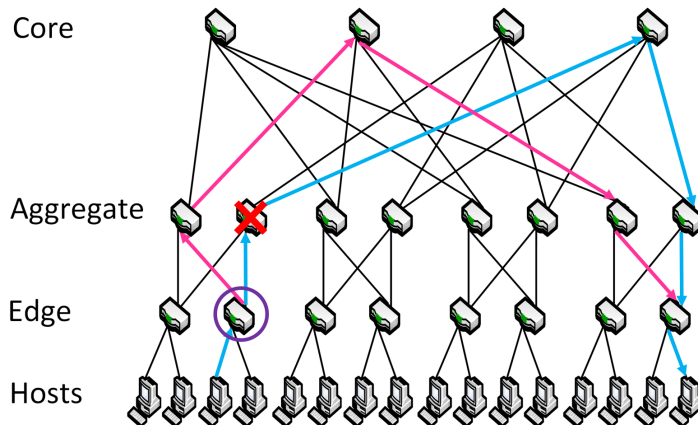


Figure 3.3: Depiction of a failure of an upstream aggregate router, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.

3.3.4 Downstream Aggregate Router

This is another failure that shares behaviour with its equivalent link. Again there is a delay between the router going down and adjacent routers realising. This should be the costliest possible router failure.

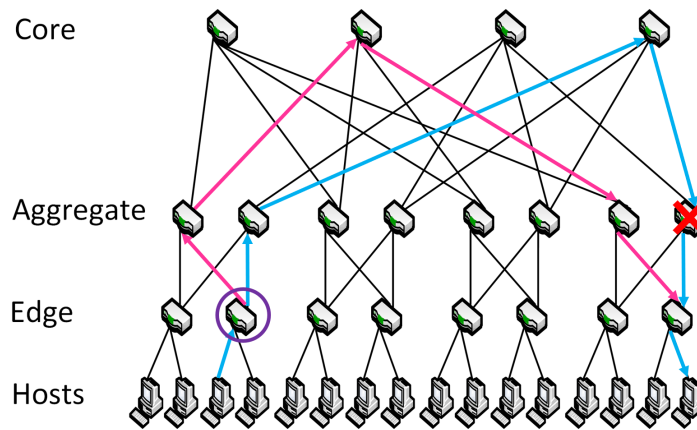


Figure 3.4: Depiction of a failure of a downstream aggregate router, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.

3.3.5 Core Router

Since the aggregate routers are connected to more than one core, and all cores are connected to all pods, the behaviour in this case is expected to be quite simple. Once the upstream aggregate router realises the core router is down it can switch to an alternative core router, each of which have a viable path equal in length to the original.

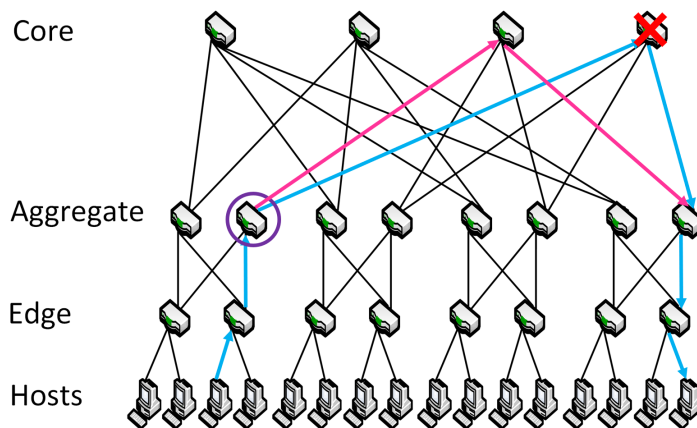


Figure 3.5: Depiction of a failure of a core router, for $k = 4$. The original route is shown in blue and the new route in pink. The purple circle indicates the decision router.

3.3.6 Link Failure Methodology

The links were failed using mininet's `configLinkStatus` function. This runs an `ifconfig` command on the virtual link which sets the link's status to `up` or `down`. The router on which this command is run is therefore immediately aware of the change in link status and triggers the appropriate response according to the routing protocol.

3.3.7 Router Failure Methodology

Failing a router is simulated by stopping and restarting the Quagga process. Depending on the protocol this may lead to a termination notification being distributed to the neighbouring routers. This process usually took about 40 milliseconds to complete.

3.4 Logs

In order to analyse the events and behaviour on the hosts around fail time the Quagga debug option was turned on for some of the experiments.

For BGP the logs are relatively sparse, even with all logging options enabled, yet clearly describe the events on the hosts. For OSPF, however, logging caused some issues. First of all the sheer volume of log messages is much higher than for BGP, and there is less granularity in the logging options. After evaluating the different options for logging, the *ospf event* option would log the relevant information and was enabled. Even with limited logging the disk space on the VMs running the experiments would quickly fill up, keeping the experiments from completing.

Some methods were attempted to avoid these issues. Dynamically enabling the logs only during the failure events seemed like a solution and works for BGP, however, the command would just quietly fail for OSPF. Increasing the disk space helped for the smaller topologies, but was not viable for larger ones. Clearing the logs periodically significantly improved the situation, but was still not enough. This was because of the variable duration required for the network to return to a stable state after routers and links were restored, during which time the disk would fill up once more. This duration would vary due to the way the network connectivity was checked. A built in mininet method for pinging the nodes was given a list of all the hosts. The return value of this function, the percentage of pings that didn't reach their destination, was used as a measure of the network's convergence. With more time this could have been optimised, by clearing the logs more frequently for example, to potentially solve the problem. Finally the experiments were modified to always start and end in the same pods so that only the routers in those pods, plus the core routers, would be required to log.

Even with all the measures considered in place, for $k > 8$ the system would still sometimes fill up its disk. This is why the decision was made to only log for smaller k values.

A concern that all the IO operations required for logging might slow down the system and affect the observed convergence times was also brought up. So OSPF logs were only enabled for parallel runs of experiments to identify the different behaviours of link and router failures. A detailed discussion of the potential impact of logging on the convergence time will follow in the analysis section. 4

3.5 Path Analysis

The logs were helpful in describing the routers' individual behaviours during the convergence process. It was, however, very difficult to derive any meaningful description of global behaviour from this data. This is why a method of logging the changing route paths during convergence was introduced. The paths were computed by using the *ip route get dstIP* command on the source host to find the next router in the path, this process would then be repeated on each router in the path until either the destination host or a black hole was reached. The paths were computed at 5 to 50ms intervals until a path was found to the destination, the length of these intervals would depend on the current processing load on the system.

3.6 OSPF

Two different configurations were chosen to test OSPF. One with all the routers in a single area and one where the network gets grouped into different areas.

Single Area For this configuration every router is placed in *area 0*.

Multiple Areas The OSPF Protocol requires that all areas are directly connected to the *backbone* or area 0. This area is responsible to inform the other areas about possible routes outside of their area by sending out so called summary LSAs. The naïve choice for a fat-tree is therefore to define the *backbone* as the area composed of all the core routers, and for each pod to be its own area. A few key factors prevent this approach:

Each area must internally take the form a connected graph. In the fat-tree topology there are

no connections between routers on the same level. This is not a problem for the pods, which include two levels and form bipartite connected graphs. Between the cores, however, there are no links, and so the *backbone* would not be a well defined area. Additionally, design guidelines for OSPF configurations suggest avoiding placing a router in more than three areas in order to maximize the stability of the protocol [4]. This makes this topology even less desirable as each core router connects to each pod and would therefore be in $k + 1$ areas with this approach.

To work around those requirements the topology had to be slightly adjusted:

For $k = 4$ the two core routers in the centre were connected with an additional link and chosen as the *backbone* area. The first core router and the first two pods form *area 1* while the the last core router and the last two pods form *area 2*.

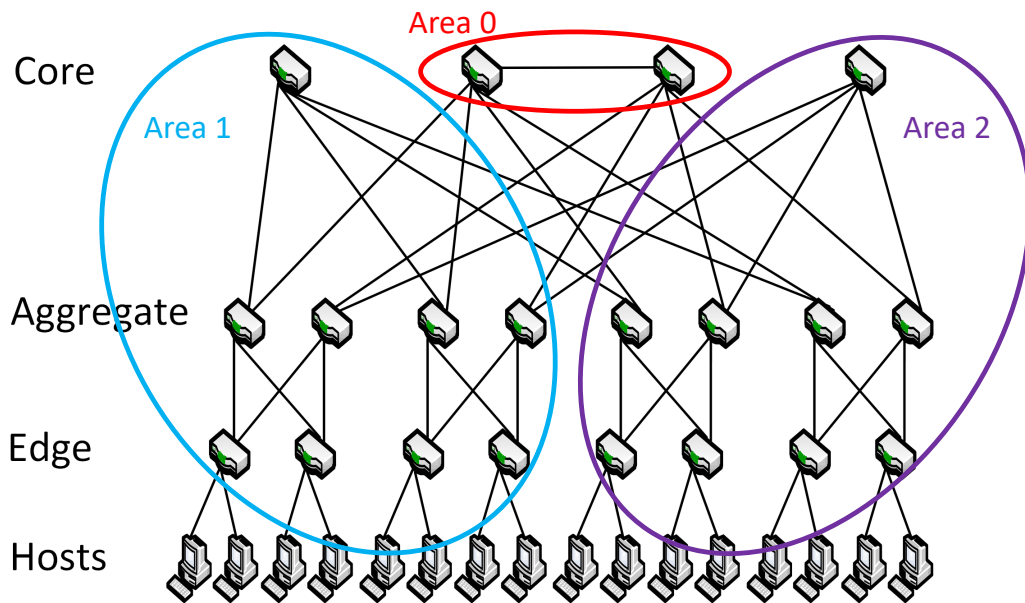


Figure 3.6: Area organisation on the fat tree for $k = 4$.

For the higher k values two core routers are each picked to be part of *area 1* and *area 2*. Thus if a core router in one of those areas fails the area does not get partitioned. This is unavoidable in this scheme for $k = 4$. The rest of the core routers, which form the *backbone*, are connected circularly to avoid a partitioning of the area when a core fails.

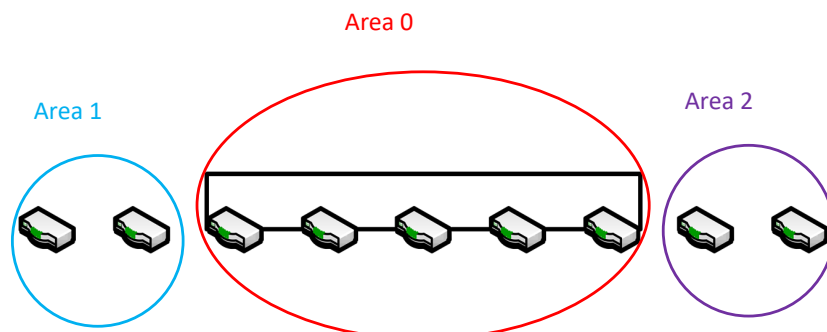


Figure 3.7: The division of cores into the different areas and the additional links in the backbone for $k = 6$.

3.7 BGP

The BGP configurations are generated using Propane. A very simple policy was chosen for the networks which only indicates which prefix ends on which destination router. This corresponds to the *dc-small* example that is on the VM provided by Propane for compiling the configurations [11]. A second example on their VM, *dc-large*, was also considered. This policy places half the routers in a local group and the other half in a global group. The prefixes originating from the local group are not visible outside the datacenter while the ones originating from the global group is. There are also additional rules preventing transit traffic through the datacenter from outside and setting a preference order of the core routers for exiting traffic. Since only east-west experiments were conducted (in which all traffic originates and remains inside the datacenter) the two configurations behaved nearly identically and so the simpler one was chosen for the actual evaluation.

3.8 Timers

When measuring convergence the values used for the different timers on the protocols are very important. Ideally the timers would be equivalent, or cause equivalent behaviour, for both protocols. However, due to the different nature of the protocols this was not always possible. For the experiments minimum timer values were used in order to optimise performance and simplify the comparison of the protocols.

3.8.1 OSPF

There are two relevant timers that can be set for this protocol:

Dead-interval and hello multiplier The *dead-interval* is the delay from the last received *hello packet* from a neighbour before that neighbour is declared non-operational and the appropriate response is triggered [9]. With Quagga the minimum value for the *dead-interval* is 1s [14]. The *hello multiplier* indicates how many *hello packets* should be sent per second. This has a direct influence on how soon the *dead-interval* timer is triggered after a failure.

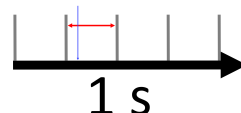


Figure 3.8: Distribution of *hello packets* over a second with the *hello multiplier* set to 5. The blue arrow indicates the point of failure. The red arrows indicate the last *hello packet* it will have seen by the time the *dead-interval* timer has run out. One second minus this time is the time it takes to notice the router has failed.

The longer the delay between *hello packets* the shorter the minimum time between the failure occurring and the end of the *dead-interval*. Maximising the *hello multiplier* minimises this time. For Quagga 0.99.22.4 the maximum *hello multiplier* is 10.

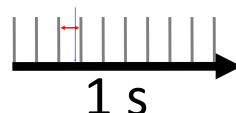


Figure 3.9: Hello packets over a second with the *hello multiplier* set to 10. The blue arrow indicates the point of failure. The red arrows indicate the last *hello packet* it will have seen by the time the *dead-interval* timer has run out. One second minus this time is the time it takes to notice the router has failed.

SPF throttle timers These timer values define the behaviour of the protocol in scheduling the *SPF* recalculations after a network change. They also specify the boundaries of an adaptive hold time that separates consecutive *SPF* calculations. Three values are defined when setting this timer:

- **Delay:** This value indicates how long in milliseconds the router waits until it triggers a response to a changed network state. This value was set to 0ms.
- **Initial-Holdtime:** The first value the hold time will be set to. Events occurring within the hold time of the previous *SPF* calculation will double hold time. This value was set to 50ms.
- **Max-Holdtime:** This value caps the hold time to a maximum even as more events are registered. This value was set to 250ms.

[15]

3.9 Additional Experiments

3.9.1 Larger Destination Pod

For this variation $k/2$ aggregate routers were added to the destination pod to increase the length of the shortest possible paths. The goal of this experiment was to add an additional layer between deep downstream failures and the decision routers. The assumption was that BGP would be more affected by this change than OSPF since its distance vectors need to propagate further back until a decision leading to convergence can be made.

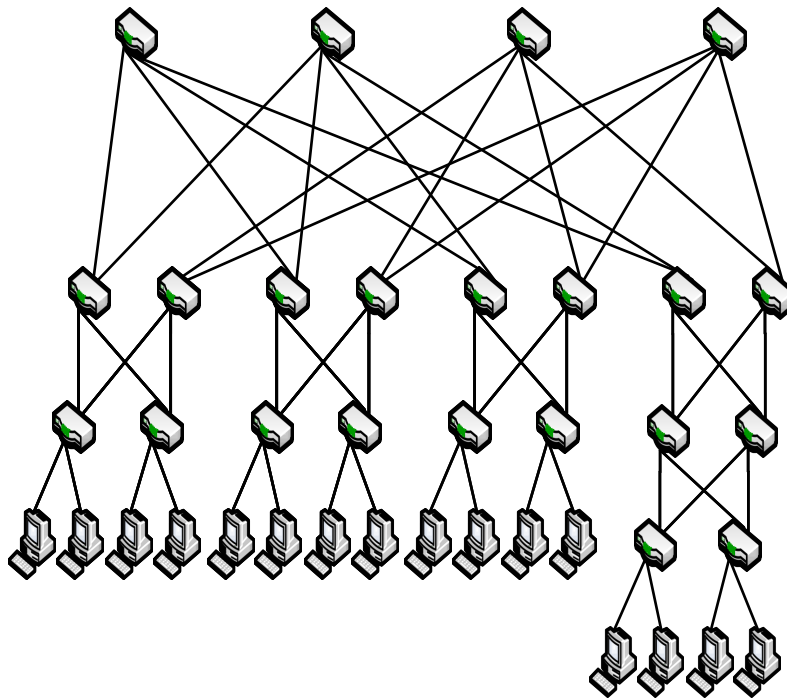


Figure 3.10: The adapted topology of the fat-tree for $k = 4$.

3.9.2 Larger Destination Pod With Less Connections

The assumption made in using the topology in 3.9.1 was undermined by the fact that the additional routers each had $k/2 * k/2$ equal length paths to all the hosts that connect to the pod and so became the natural decision routers when links failed downstream of them. This is why

the topology was slightly modified to observe the previously predicted behaviour by removing the connecting links between the additional routers and the aggregate routers. This leaves the additional routers simple extensions of the links without adding any redundant paths to the network.

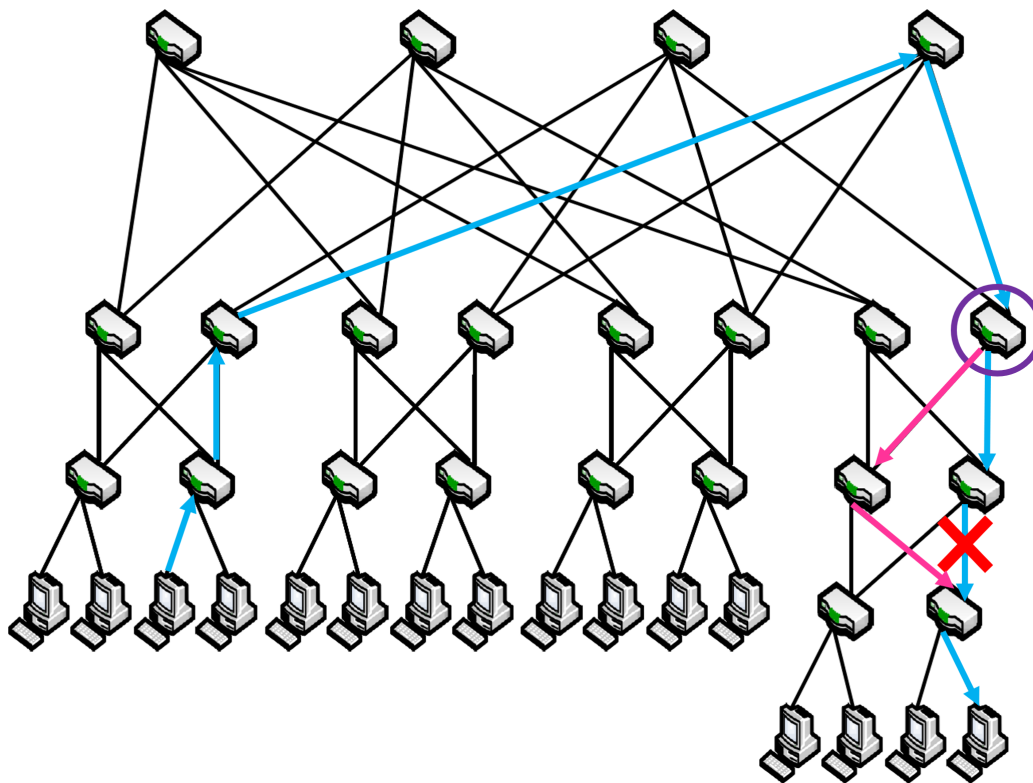


Figure 3.11: Comparison of the routing decisions made when failing a downstream edge-aggregate link.

3.9.3 More Prefixes

In this variation the original topology was used. The only thing that was added was the number of prefixes the edge routers in the destination pods advertise to the network. The assumption was again that this would have a greater impact on the convergence time for BGP since this protocol recalculates routes on a prefix basis. OSPF, on the other hand, scales with the number of links and nodes in the network and so would not see much of a difference. For OSPF the router ID was set manually for the edge routers where the additional prefixes were advertised since they had to be configured to the loopback address to actually get advertised through the network.

Running this experiment every edge router in the destination pod advertised ten times the original number of prefixes.

3.10 Procedure

The network gets set up for the chosen k value with modifications depending on which experiment is run. The flows always go from the first to the last pod, intermittently picking between three distinct source and destination hosts. If logging is enabled, it is enabled only on the routers in those pods and on the core routers. The logs get cleared periodically during the bootstrapping process. To choose the proper router or link to fail the current path is calculated by using the *ip route get dstIP* command on the source host to find the next router in the path, this process is then repeated on each router in the path until the destination host is reached. The flow is started, the failure is triggered, and the time of failure recorded. While the experiment is running

the source and target hosts record data about the package flow. After the network has converged the relevant log files get stored for analysis. Between the experiments a ping between all hosts is used to determine the converged state of the network. There is a delay to let the network settle prior to the post-processing.

Chapter 4

Results and Analysis

The following plots show the results measured for the different experiments. The median over all the iterations of the experiments are plotted along with the 5th and 95th percentiles. For some experiment iterations there was no package loss at all. An explanation for this could be that the protocol may have reconverged onto another path before the failure for some unrelated reason (e.g. an expired LSA triggering a new SPF calculation). Whatever the reason, those specific iterations did not produce any relevant data and were therefore not included in the analysis.

4.1 Log Analysis

Sample logs were evaluated by tracing the behaviour of the protocols. For BGP the logs showed clearly that the protocol response was triggered immediately upon failing a link or router. It seems that the semi-permanent TCP connections BGP utilizes notifies the affected routers immediately upon going down.

For link failures under OSPF a similar behaviour was observed. Since the SPF calculation delay timer was set to 0 an LSA was sent out immediately after the router noticed the failure which then propagated through the network. For router failures the situation was different. The *hello packet* and *dead-interval* mechanisms result in a delay before neighbouring routers are aware of the failure. All relevant logs were evaluated and the time it took the neighbours to realize that a router was down was statistically analysed.

Table 4.1: Detection times at router failure for OSPF in milliseconds.

95th	5th	median
904	995	950

For some experiments the median convergence times were lower than the median router failure detection time. Therefore the 95th percentile value was chosen as the correction factor, making for a more conservative estimate. Correcting for the detection time adds another source of uncertainty to the convergence times when a router is failed.

4.2 Path Analysis

Analysis of the recorded paths was the main tool to better understand the differences in convergence times across failure types. For any given failure type there would often be more than one set of paths recorded during convergence. Due to the resolution of the path sampling this does not necessarily mean different behaviour, as paths may have been attempted between sampling times. This also means these logs are less descriptive when convergence happens quickly, ie. faster than the sampling intervals. This is why for OSPF routing loops are observed more often in the larger networks where convergence time was generally larger. The number of routers in the system also increase the effect of asynchronicity. New routing paths typically occur when some routers have already converged to the new state while others are still in the pre-failure

state and the distance vectors or the view of the graph are not uniform across the network any more.

In the figures showing the paths failed links are drawn in red, failed routers as red crosses, and the routing paths are shown in blue, with links traversed more than once in magenta. Once a path to the destination host was found the path logging would end. In retrospect it would have been better to log the paths for the entirety of the experiment in order to see the path evolution until the protocol has converged to a shortest path. There was unfortunately no time to rerun everything with the path logs enabled till the end.

4.3 Main Discussion

4.3.1 General Tendencies

BGP Convergence times under BGP scale with the distance from the failure to the decision router. This means the convergence time for BGP remains quite stable with increasing network size since the larger k values only increase the number of routers in each pod and not the actual path length from source to destination host.

When failing a router the convergence times were much lower than expected. The *keepalive* and *dead-interval* timers, set to one and three seconds respectively, should have caused convergence times to be in the order of seconds as with the *hello timer* and *dead-interval* for OSPF. However, due to the way Quagga terminates a router, there was only a 40ms delay between the termination command and the neighbouring routers being notified. It is assumed that when this router is down so is the semi-permanent TCP connection BGP utilizes for communication between the routers and this is what triggers the protocol response. The termination notifications immediately triggers BGP's response without waiting for the *dead-interval* to expire. Therefore there was no need to correct for detection time for the BGP experiments.

A difference in convergence times can generally be observed by comparing the upstream and downstream experiments. This is due to the different path lengths the failure must propagate. An analysis of the logs showed that there is always a gap of about 50ms between receiving a withdrawal update on a router and sending the complementary unreachable update.

An interesting observation about how the Propane configurations influence the routing choices could be made:

The propane configurations do not take any measure of load balancing. This means that the normal BGP decision process will follow its predefined order. For our purposes these are the relevant steps:

1. **Shortest-AS-Path**
2. **BGP-Identifier**

[2]

The local preference is never set by the simple BGP configurations Propane generates, nor is the multi exit discriminator. It is not necessary to consider any process which relates to the interaction of different protocols since only eBGP is running on our network. This leaves us with the two main steps that affect the decision process.

The first step ensures that the path chosen once the entire network has reached a stable state will always have the shortest possible length. With the chosen topology there are always same-length alternative shortest paths unless the network is actually partitioned.

The second step ensures that the upstream routes always converge onto one specific path. The higher BGP-Identifiers are given to the routers ordered by type and number ascending (ie. *r_c_1* has the highest id, *r_c_2* has the next highest, *r_a_1* has the highest id of the aggregate routers, etc.). This does not mean that the flows will always take the same path. Since links are failed and restored during experiments the paths will change and remain changed for the next experiment given that the protocol does not change paths unless invalidated.

For example: a link on the main route is failed and the protocol converges to a different route. When the link is restored the protocol has no reason to change the current route since it is still valid.

The route chosen due to the propane configurations does have an effect on the number of prefixes advertised on one path. An edge router will always choose the same router to forward

all the upstream packets to, the one with the highest router number and therefore the lowest BGP-Identifier. The aggregate router will in turn always send all its upstream traffic to the core router with the lowest BGP-Identifier. Aggregate routers have fewer upstream paths than edge routers as they can reach all the prefixes of the edge routers in the pod downstream, while edge routers can only reach directly attached hosts downstream. A core router only knows downstream paths and therefore has an even distribution of paths on all its interfaces. This will have an effect on the duration it takes the protocol to recalculate the minimal paths for the different prefixes. The more paths there are on a specific route which is now invalidated the longer it will take. Nonetheless, it seems that for the considered network sizes the propagation length has a much greater effect on convergence than this time difference does.

OSPF For both the single and multiple area configurations one generally observes that the larger the network, the more time it takes the protocol to converge. This is due to the nature of the shortest path algorithm whose complexity scales directly with the number of edges and nodes in the network.

The difference in convergence times between upstream and downstream experiments are not as pronounced as for BGP for the single area configurations. While it is still true that the fault is further away from the decision router in the downstream cases, it doesn't matter because Quagga does not implement an incremental SPF algorithm. Despite the fault being considered in an earlier iteration of the algorithm for upstream failures, the routing table is only updated after the SPF calculations for the entire network have finished, so there is little to no difference between the two cases.

The configuration with multiple areas is not generally faster than the one with a single area. This could be because the networks are still too small, thus the overhead of handling multiple areas is not offset by the shorter SPF calculations.

4.3.2 Individual Experiments

Upstream Link Failure Edge-Aggregate

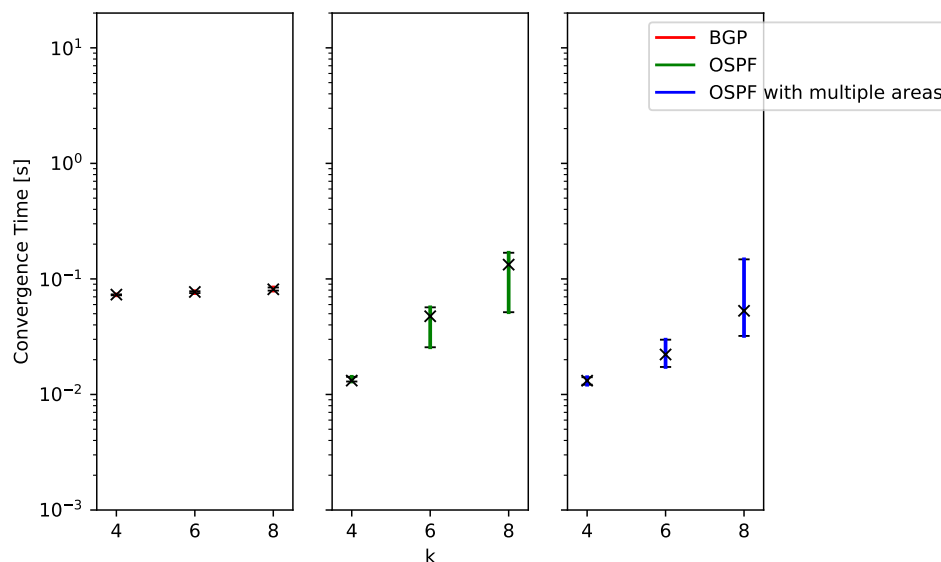


Figure 4.1: Convergence Times after a failure of a link between an edge and an aggregate router upstream on the standard topology.

Table 4.2: Median values for Figure 4.1 in seconds.

k	4 (s)	6 (s)	8 (s)
BGP	0.073	0.077	0.081
OSPF	0.013	0.048	0.133
OSPF with multiple areas	0.013	0.022	0.053

In figure 4.1 BGP behaves very stably. This behaviour is as expected since the failure is directly adjacent to the decision router. It needs only notice that the link is down, send the unreachable updates, and converge to a different aggregate router. This is also what is observed by the path plots for this experiment (Figure 4.2).

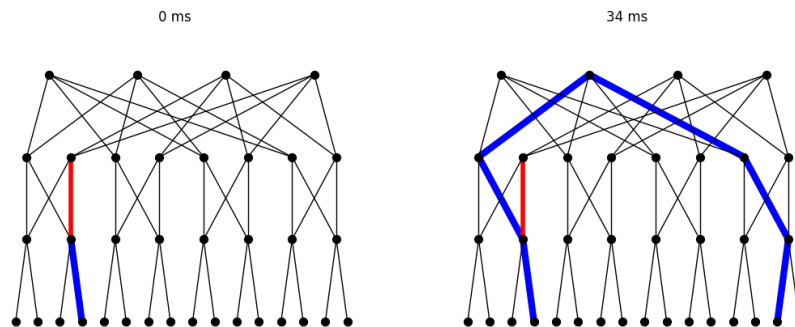


Figure 4.2: The path evolution from the failure until a new route to the node is found for BGP and $k=4$. The behaviour is equivalent for $k=6$ and $k=8$ for this protocol.

For OSPF the convergence time increases with k as SPF calculations take longer for bigger networks.

The configurations with a single area behave as expected. For all the iterations and k values the network is recomputed and a new path with the correct minimized number of hops is found (Figure 4.3).

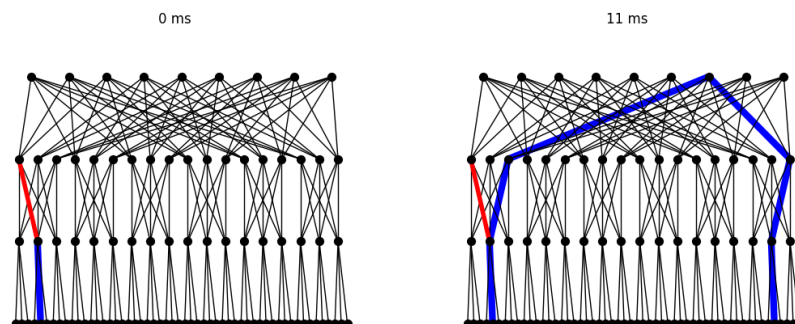


Figure 4.3: The path evolution from the failure until a new route to the node is found for OSPF with a single area for $k = 6$. The behaviour is the same for all the iterations and for all k .

The OSPF configuration with multiple areas behaves in the same way as the one with only one area for $k = 6$ and $k = 8$. For $k = 4$, the smallest topology, there was some unexpected behaviour. In four of the iterations the protocol converged to a path that was two hops longer than the actual path.

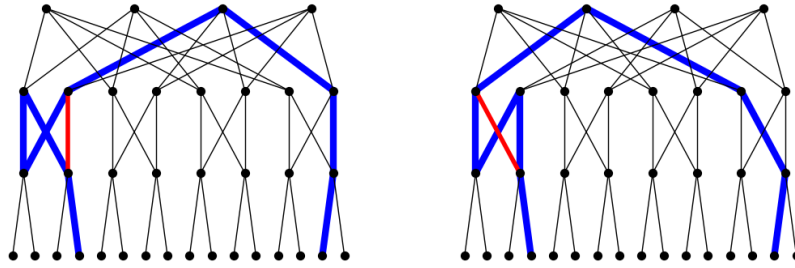


Figure 4.4: The two longer paths OSPF with multiple areas converges to in rare cases for $k = 4$.

Downstream Link Failure Edge-Aggregate

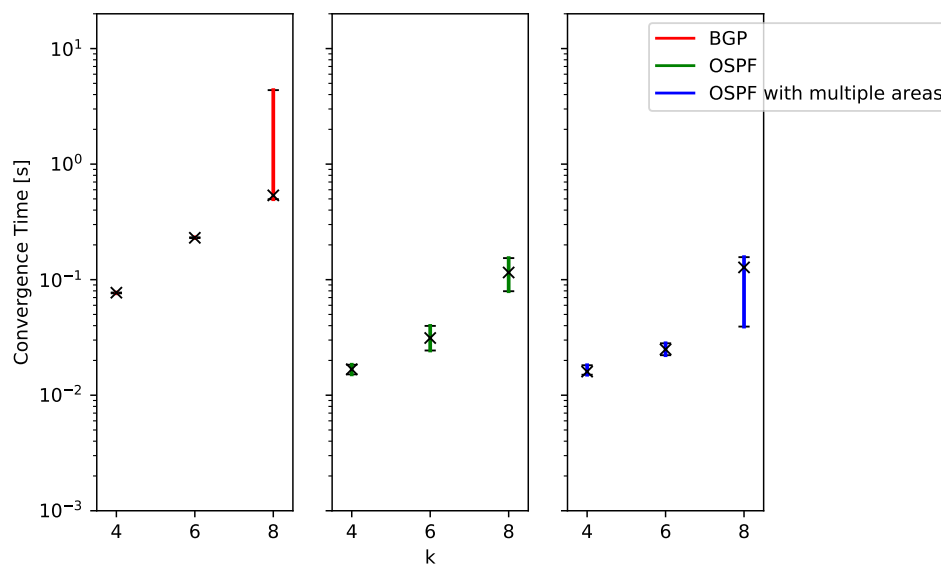


Figure 4.5: Convergence Times after a failure of a link between an edge and an aggregate router downstream on the standard topology for the different protocols.

Table 4.3: Median values for Figure 4.5.

k	4 (s)	6 (s)	8 (s)
BGP	0.077	0.23	0.537
OSPF	0.017	0.031	0.115
OSPF with multiple areas	0.016	0.025	0.128

The failure of a downstream edge-aggregate link is clearly the worst case scenario for BGP. The failure is the furthest possible from the decision router in this case, as shown in 3.3.2. The expected behaviour would predict a convergence time of at least 150ms since the failure needs to propagate back three hops, until it reaches the upstream edge router. This was only observed for $k > 4$. Three different sets of behaviours were observed for this failure depending on the network size:

For $k = 4$ the downstream aggregate router routes the flow to the other edge router as soon as it notices the failure. The edge router in turn routes it back up to the second aggregate router which can finally reach the destination router. The downstream aggregate router reacts the quickest as it is directly adjacent to the failure and sends the flow towards the new shortest path it knows. This leads to a similar convergence time as for the upstream failures, about 75ms (figure 4.6).

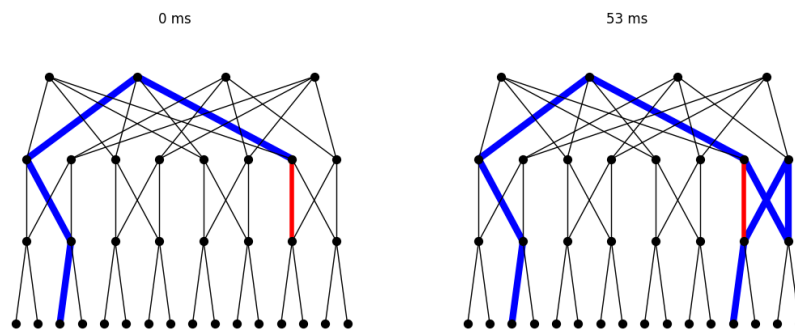


Figure 4.6: Path evolution for BGP and $k = 4$ after a downstream failure of an edge aggregate link.

For $k = 6$ this behaviour is no longer observed. Instead one can clearly see the originally expected fault propagation back to the upstream edge router (figure 4.7).

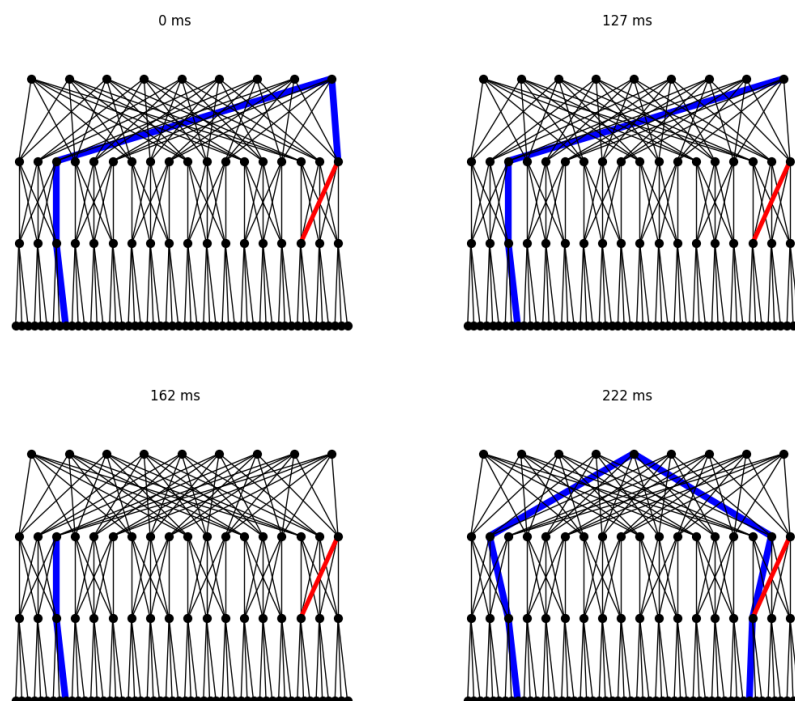


Figure 4.7: Path evolution for BGP and $k = 6$ after a downstream failure of an edge aggregate link.

For $k = 8$ the most often observed behaviour (24 iterations) is similar to the one observed for $k = 6$. There is a clear failure propagation back to the upstream edge router. In the meantime, the core router tries to route the flow to different pods. The core routers know that their connection to the destination pod is no longer valid. But since the failure has not propagated through the entire network yet they still see the other aggregate routers advertising the prefix over the same faulty link but via a different core router. Therefore the core router tries to route the flow there until the withdrawal messages from all the core routers reach the aggregate routers and they in turn stop advertising the prefix. (Figure 4.8).

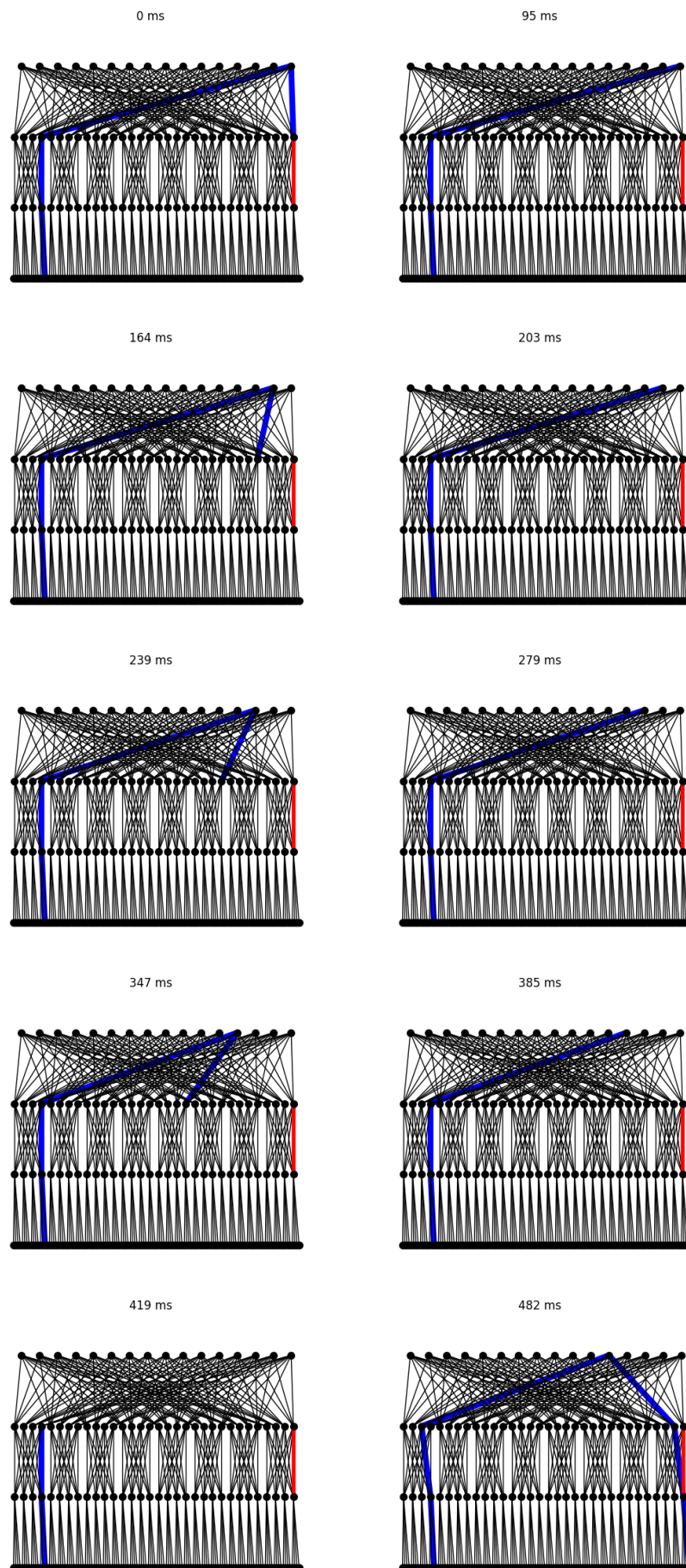


Figure 4.8: Most observed path evolution for BGP and $k = 8$ after a downstream failure of an edge aggregate link.

For some iterations the routing loops would occur. Again the aggregate router would sometimes route the flow to a different edge router, as observed for $k = 4$. (Figure 4.9).

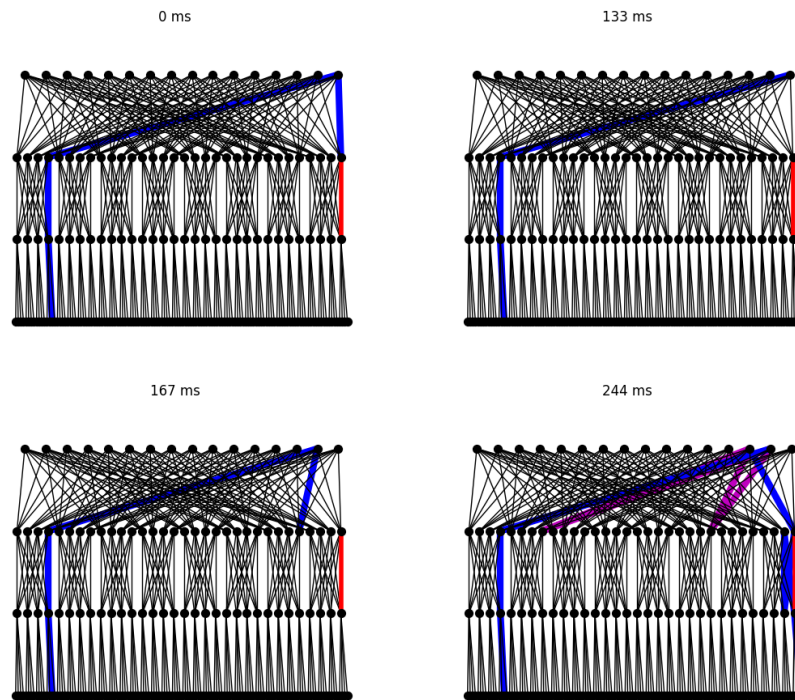


Figure 4.9: One of the paths observed in which a routing loop occur for BGP and $k = 8$ after a downstream failure of an edge aggregate link.

One possible explanation for the higher rate of route loops occurring for $k = 8$ could be that the added number of routers increases the asynchronicity of the network. There are more possibilities that neighbouring routers find themselves in conflicting states when there are more of them, leading to more routing loops being observed.

OSPF has a much easier time dealing with this failure.

OSPF deals with this failure much better. For a single area the protocol behaves as expected. The stable state is reached once the edge router has finished its SPF calculation and routes the flow over a different aggregate router. The convergence time is reduced when the protocol picks a longer path before the decision router is done with its calculation. The routers closer to the failure will finish their SPF calculation first and change the way they forward the packets of the flow until the flow stops going through them. This was observed in certain cases (Figure 4.10).

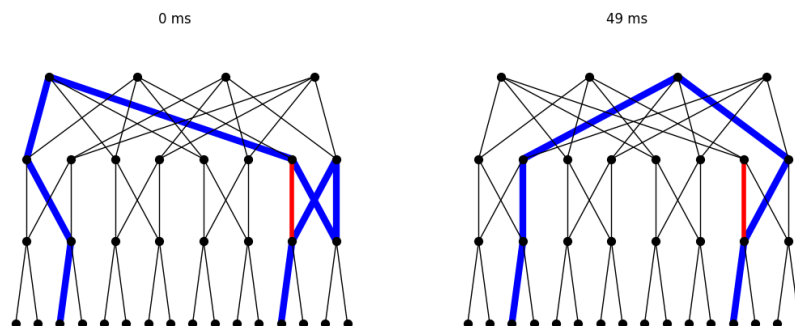


Figure 4.10: Path evolution for OSPF with one area and $k = 4$ after a downstream failure of an edge aggregate link. One can nicely observe the longer path that is taken until the upstream edge router has finished its SPF calculation.

The same behaviour is observed for the OSPF configuration with multiple areas. Sometimes the longer paths that are attempted while the decision router is still working on its SPF calculation can be observed in the path plots. In the end the edge router will change its mind and choose a different aggregate router as next hop for this flow.

Upstream Link Failure Aggregate-Core

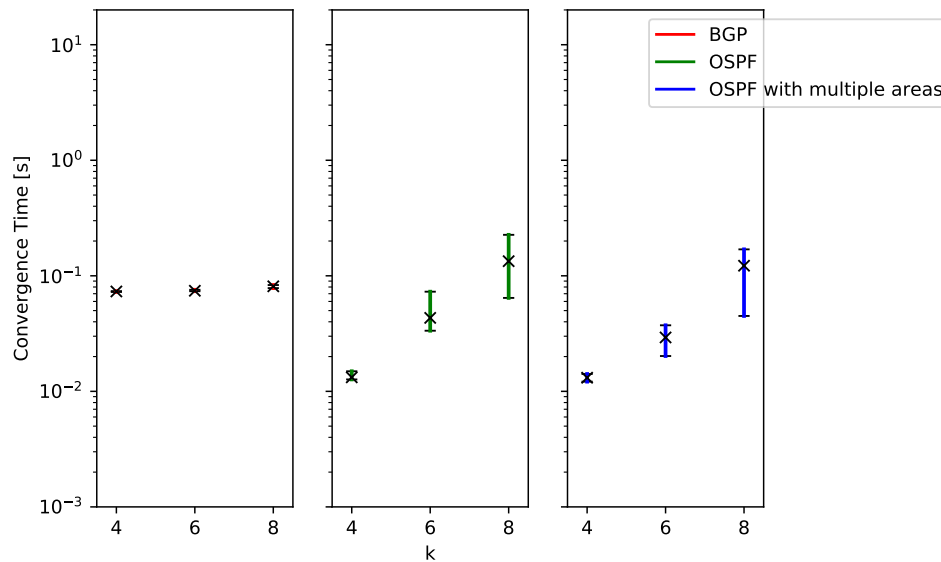


Figure 4.11: Convergence Times after a failure of a link between an aggregate and a core router upstream on the standard topology for the different protocols.

Table 4.4: Median values for Figure 4.11.

k	4 (s)	6 (s)	8 (s)
BGP	0.073	0.074	0.081
OSPF	0.013	0.043	0.134
OSPF with multiple areas	0.013	0.029	0.122

In figure 4.11 we see a very similar behaviour to the failure of an upstream edge-aggregate link. This makes sense as we basically have the same situation, just between an aggregate and a core router instead of an edge and an aggregate router.

For BGP the aggregate router needs to notice the failure and send an unreachable update for the prefixes converged on this link before it can re-converge to a path over a different core router. The convergence time is very similar to that of the upstream edge-aggregate link, since the failure is again directly adjacent to the decision router. This interpretation is confirmed by the path plots (Figure 4.12).

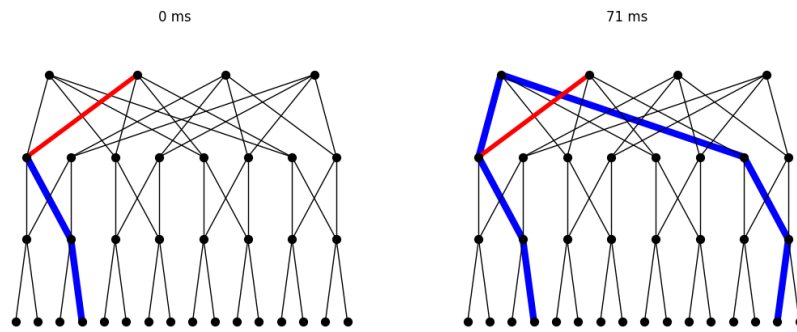


Figure 4.12: Path evolution for BGP on a fat-tree with $k = 4$ after an upstream failure of an aggregate core link. The behaviour is identical for the higher k values.

OSPF also shows a similar behaviour for aggregate-core link failures as for edge-aggregate. The path plots show the exact same behaviour as for BGP. In contrast with the upstream edge-aggregate link failure no paths longer than the shortest path were observed during convergence.

Downstream Link Failure Aggregate-Core

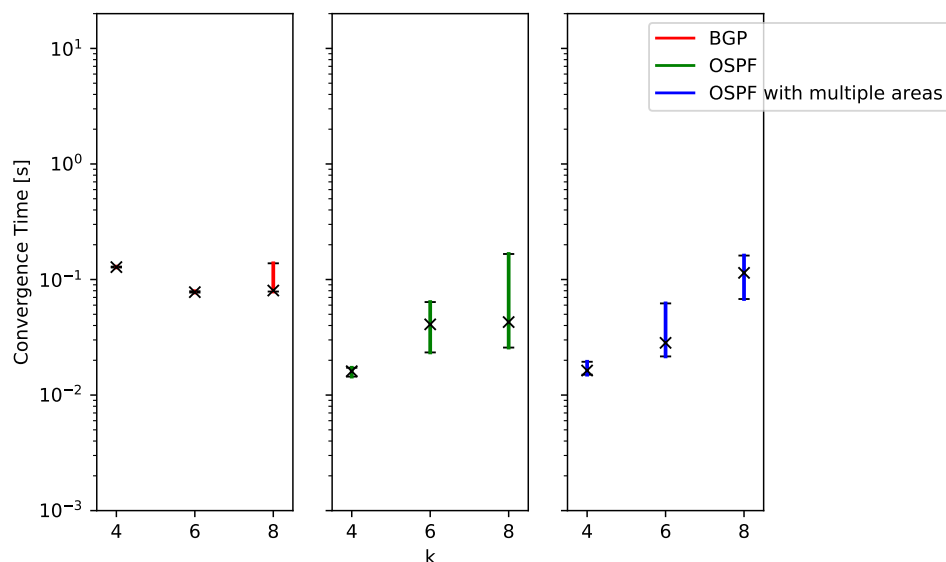


Figure 4.13: Convergence Times after a failure of a link between an edge and an aggregate router downstream on the standard topology for the different protocols.

Table 4.5: Median values for Figure 4.13.

k	4 (s)	6 (s)	8 (s)
BGP	0.128	0.078	0.08
OSPF	0.016	0.041	0.043
OSPF with multiple areas	0.016	0.028	0.114

For the downstream failure of an aggregate core link (Figure 4.13) BGP is generally slower than OSPF. Here the failure must propagate along two hops in the path until the upstream aggregate router can converge to a new path over a different core router. While this behaviour is observed for $k = 4$. For the higher k values the core router instead pushes the traffic down to a different aggregate router which in turn can reach a different core router with no defect links. This is

also why the packet loss ceases faster for higher k values. The core router chooses to push the traffic over a longer path earlier than the upstream aggregate router notices the failure and can converge to the shortest path (Figure 4.14).

One possible explanation for why this is observed more for larger k values could be path redundancy. For $k = 4$ every aggregate router is only connected to two different core routers. It therefore has two next hops to choose from to reach other pods that result in a shortest path. The likelihood that the aggregate routers connected to the core router affected by the failure choose the same core router as their next hop is higher than for higher k values, for which there are simply more choices. If the next hop of all other aggregate routers connected to the affected core router point back to it those paths will not be chosen by the protocol to avoid routing loops.

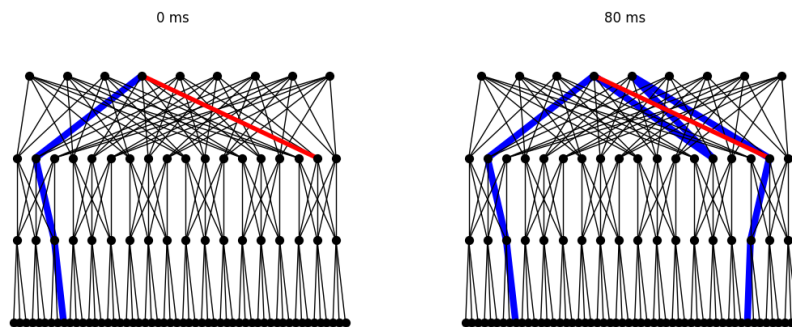


Figure 4.14: Path evolution for BGP on a fat-tree with $k = 6$ after a downstream failure of an aggregate core link. The behaviour is the same for $k = 8$.

For the single area OSPF configuration the expected behaviour is observed.

As soon as the SPF calculation is done on the upstream aggregate router it routes the flow over a different core router. Occasionally the core router routes the flow over a different aggregate router, resulting temporarily in a longer path, as an intermediate step occurring when the core router finishes its SPF calculation while the upstream aggregate router is still computing. This is similar to BGP's behaviour, but happens on a much smaller time frame since the SPF calculations take in the order of 10ms while a BGP failure propagation is in the order of 100ms (Figure 4.15).

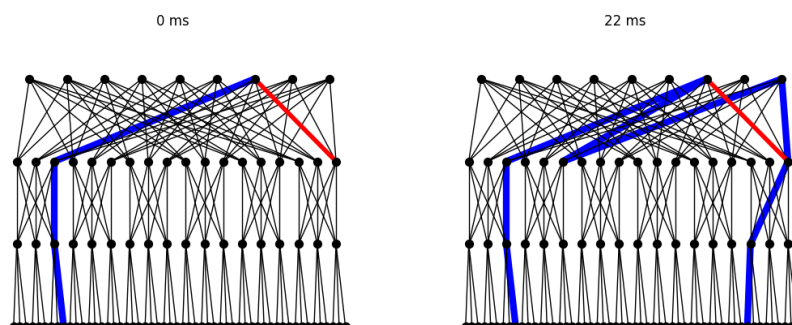


Figure 4.15: Possible path evolution for OSPF with a single area on a fat-tree with $k = 6$ after a downstream failure of an aggregate core link.

For the multiple area OSPF configuration with $k = 4$ and $k = 6$ the intermediate paths are the same as for the single area. For $k = 8$ a routing loop is sometimes observed. The aggregate router the core chooses to route the flow over routes it back until it is done computing and chooses a different router to route the flow to (Figure 4.16).

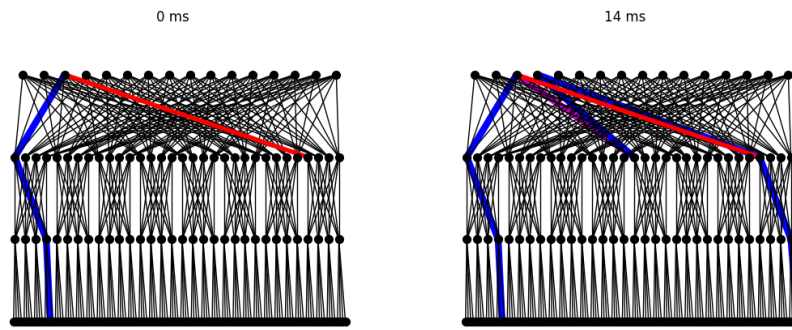


Figure 4.16: Possible path evolution for OSPF with multiple areas on a fat-tree with $k = 8$ after a downstream failure of an aggregate core link. A routing loop is formed between a core and an aggregate router as they do not finish their SPF calculation simultaneously.

Upstream Aggregate Router

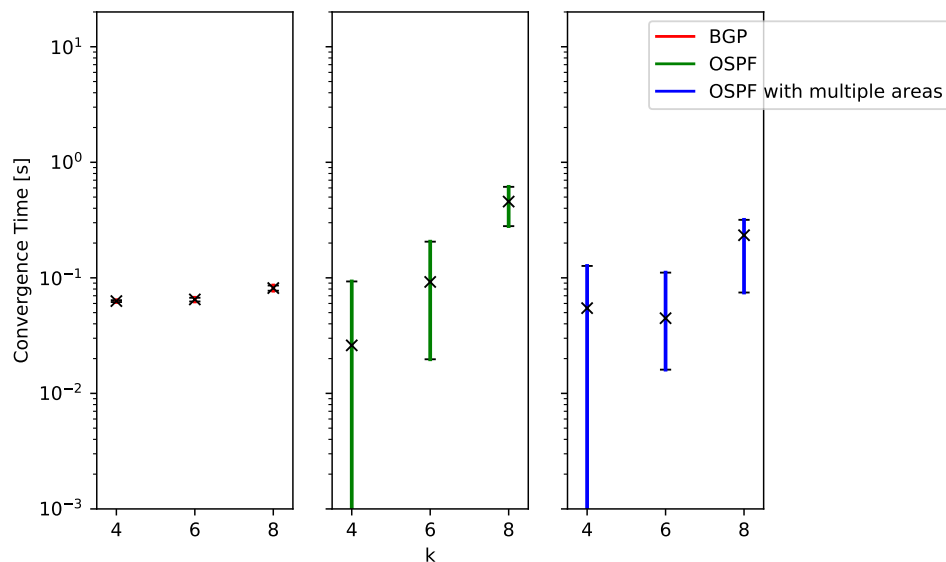


Figure 4.17: Convergence Times after a failure of an upstream aggregate router on the standard topology for the different protocols.

Table 4.6: Median values for Figure 4.17.

k	4 (s)	6 (s)	8 (s)
BGP	0.063	0.065	0.082
OSPF	0.026	0.092	0.457
OSPF with multiple areas	0.055	0.045	0.234

The failure of an upstream aggregate router shows very similar behaviour to that of an upstream aggregate-core link. The BGP path plots show the exact expected behaviour. As soon as the upstream edge router notices the aggregate router is down it converges to a different path via another aggregate router which is still up. The path plots also show very clearly that the Quagga process takes about 40ms to terminate a router (Figure 4.18).

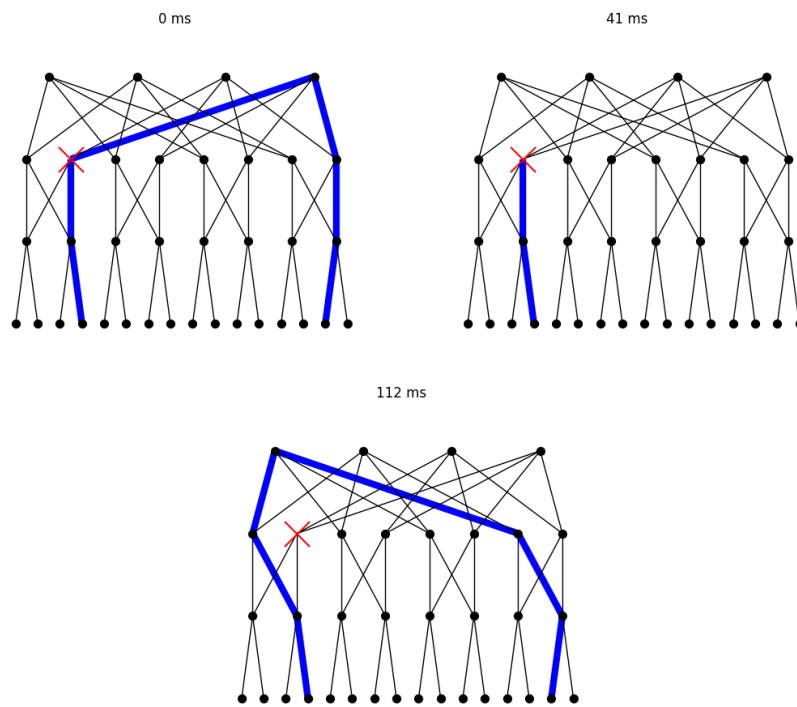


Figure 4.18: Path evolution for BGP on the fat-tree with $k = 4$ for a failure of an upstream aggregate router. One can clearly observe the 40ms it takes for the Quagga process on the aggregate router to terminate. The behaviour for $k = 6$ and $k = 8$ are identical.

For single area OSPF the same behaviour is observed for all k values. The effect of the *hello multiplier* and *dead-intervals* can also be clearly observed from the path plots (Figure 4.19).

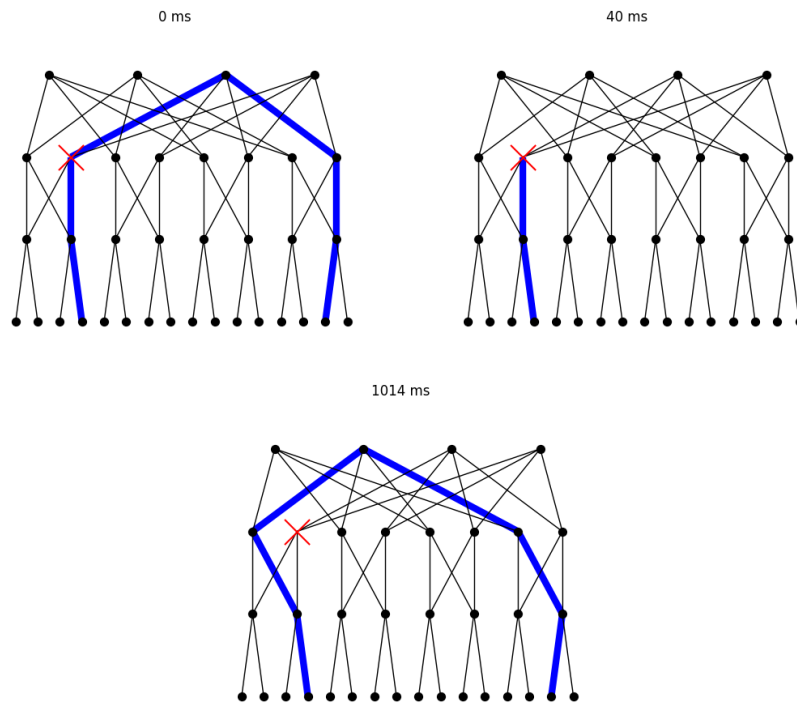


Figure 4.19: Path evolution for OSPF configured with a single area on the fat-tree with $k = 4$ for a failure of an upstream aggregate router. One can clearly observe the 40ms it takes for the Quagga process on the aggregate router to terminate. The SPF calculation is only triggered once the OSPF hello timer has elapsed (950ms). The behaviour for $k = 6$ and $k = 8$ are identical.

For multiple areas and $k > 4$ we observe the same behaviour as for a single area. For $k = 4$ intermediate routes are again observed which are longer than the actual shortest paths (Figure 4.20). This behaviour is the same as observed for the failure of an upstream edge-aggregate link, but the router is killed instead of a single link, so in this case this path does not lead to convergence but serves as an intermediate step until the aggregate router routes the flow to the core.

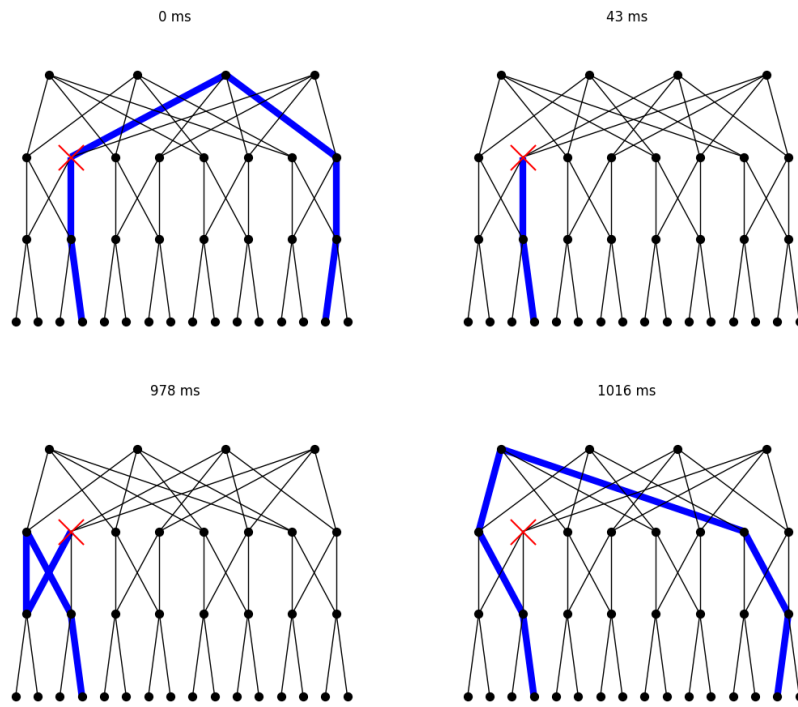


Figure 4.20: One of the intermediate paths observed for OSPF with multiple areas on a $k = 4$ topology for an upstream aggregate router failure.

Downstream Aggregate Router

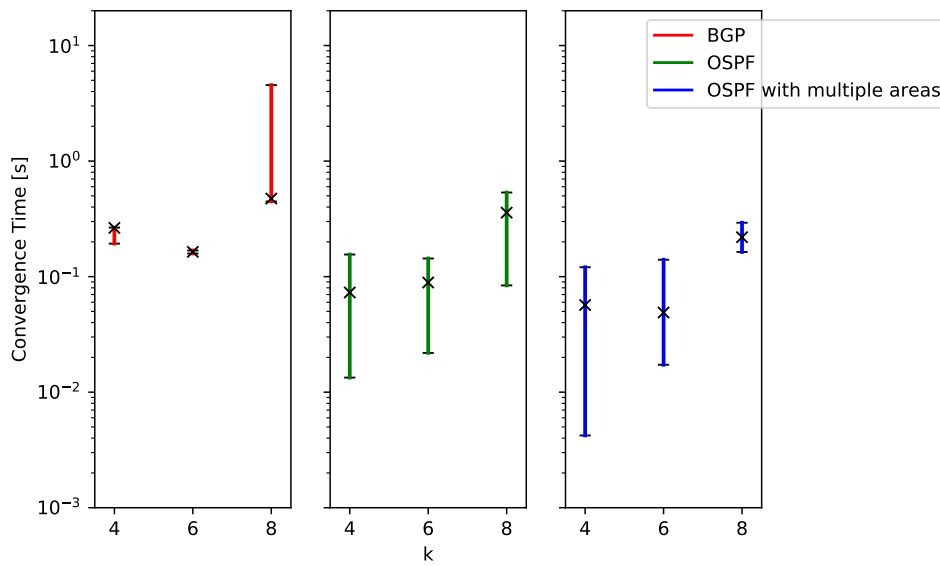


Figure 4.21: Convergence Times after a failure of a downstream aggregate router on the standard topology for the different protocols.

Table 4.7: Median values for Figure 4.21.

k	4 (s)	6 (s)	8 (s)
BGP	0.264	0.164	0.474
OSPF	0.073	0.089	0.358
OSPF with multiple areas	0.057	0.049	0.219

The trends seen in 4.21 are similar to the ones seen in the failure of a downstream edge-aggregate link (4.5) since the failure must similarly propagate all the way back to the upstream edge router before the system can converge.

For all k values the BGP protocol shows the behaviour that was already observed for $k = 8$ during a downstream failure of an edge aggregate link (Figure 4.8). The core router will push the traffic to the other aggregate routers it can reach which are not down until the fault has propagated to the edge router which can then converge to the right path. Since the router is killed instead of just one link one can no longer observe the aggregate router choosing a longer path in the destination pod, as was observed for $k = 4$ (Figure 4.6). For $k = 8$ some routing loops can be observed right before convergence (Figure 4.22).

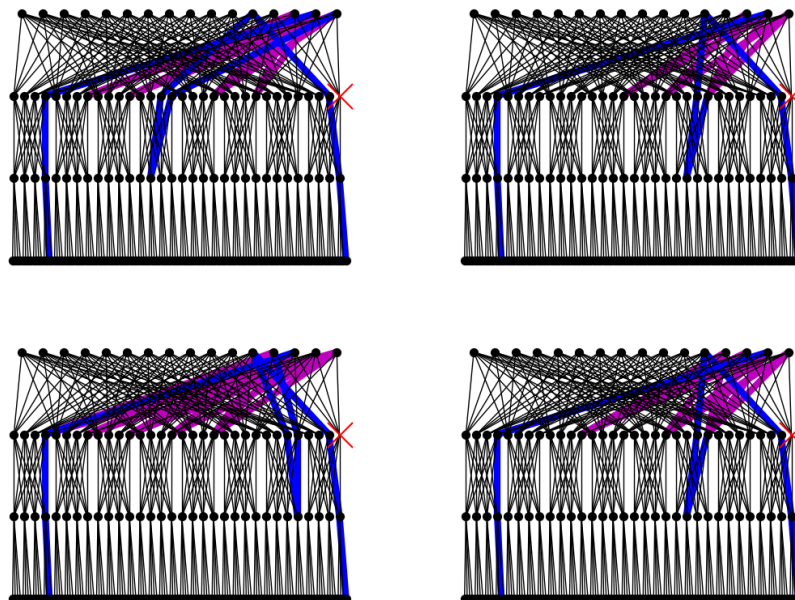


Figure 4.22: Some observed routing loops for BGP on the downstream failure of an aggregate router.

For both OSPF configurations routing loops could be observed much more often than when a single edge is failed. Again the upstream aggregate routers try to push the traffic to the cores who have already finished their SPF computations and that in turn push the traffic down to aggregate routers outside the destination pod. Since a router is failed and not just a link no paths originating from routing in the destination pod can be observed, as opposed to for downstream edge-aggregate link failures.

The long propagation path slows BGP significantly when compared to the upstream aggregate router failure. In this case OSPF is still faster for the k values used even with the conservative *dead-interval* correction. Unfortunately, the additional error introduced by this correction makes it hard to draw meaningful conclusions concerning the convergence time comparing the single area configuration to the multiple area one.

4.3.3 Core Router

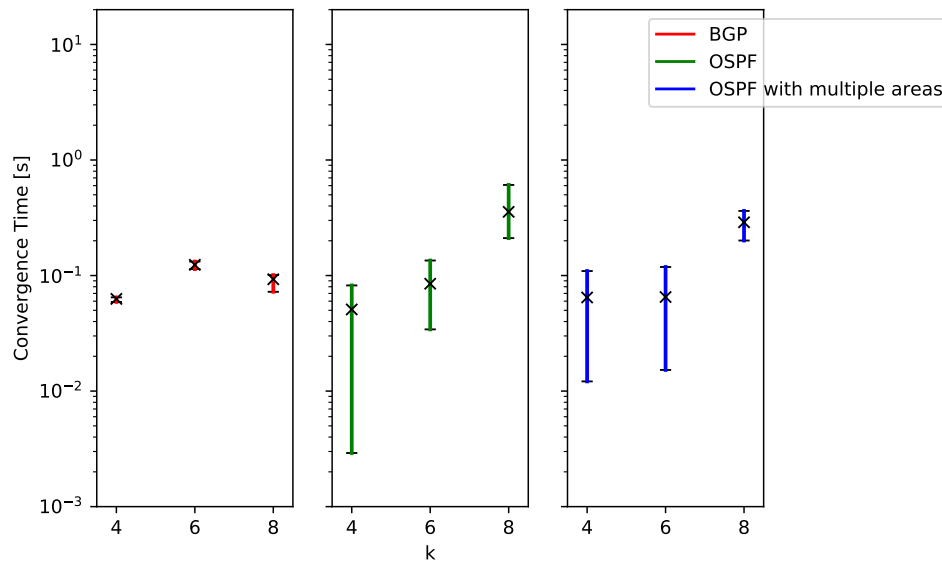


Figure 4.23: Convergence Times after a failure of a core router on the standard topology for the different protocols.

Table 4.8: Median values for Figure 4.23.

k	4 (s)	6 (s)	8 (s)
BGP	0.063	0.124	0.093
OSPF	0.051	0.085	0.356
OSPF with multiple areas	0.065	0.065	0.289

The failure of a core router is generally similar to upstream failures. The router which has to switch paths in this case is the upstream aggregate router, which is why the times are again generally low for BGP. This behaviour is observed for $k = 4$ and $k = 8$ (Figure 4.24).

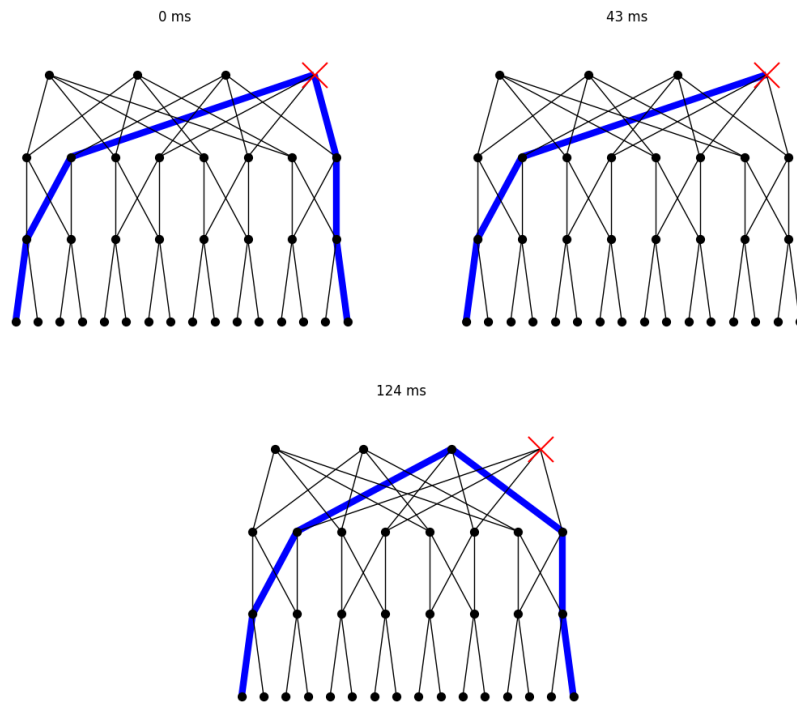


Figure 4.24: The convergence process for BGP on a fat-tree with $k = 4$ for a core router failure. This behaviour is also observed for $k = 8$.

For $k = 6$ the observed behaviour is different. The upstream aggregate router does not simply change the core router it routes the flow to, but also converges onto a different upstream aggregate router (Figure 4.25). This is why the convergence time is higher here. Since the failure propagates one hop further we see a convergence time which is around 50ms higher.

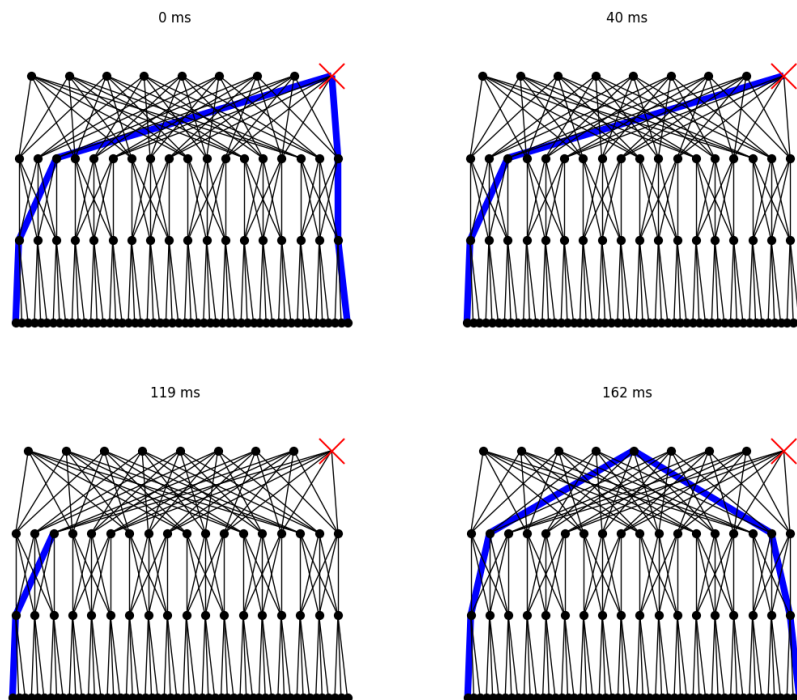


Figure 4.25: The convergence process for BGP on a fat-tree with $k = 6$ for a core router failure.

For both OSPF protocols the paths evolve the same way as for the upstream aggregate-core link failure (except for the *dead-interval* delay). The upstream aggregate router simply converges to route over a different core router as it does for BGP with $k = 4$ and $k = 6$.

4.4 Comparisons

Herein follows a comparison of the different protocols for the various adjustments to the topology and logging.

For the larger destination pod topologies only the one with the single link between original and new aggregate routers is considered in the analysis, as it were the one with the relevant topological difference.

4.4.1 OSPF: Enabled vs. Disabled Logs

The process of logging had a significant effect on the convergence times. The additional IO operations slowed down the individual routers' SPF calculations. This was primarily observable and problematic for the OSPF protocol. There the logs for every router for an experiment duration of about 30s were in the worst cases around 800KB ($k = 4$) and 3MB ($k = 6$) for the single area configuration and 3.5MB ($k = 4$) to 12MB ($k = 6$) for the multiple area configuration. In contrast, BGP never logged much more than 250 KB for $k = 8$. OSPF was therefore run with the logs disabled for the three topologies compared in the thesis and those values were chosen to be analysed.

The OSPF configuration with a single area on the standard topology were chosen to be shown for comparison. They representative of the general convergence behaviour with and without logs for OSPF. Since, for $k = 8$, it was not possible to run a full set of experiments with logging enabled on this data is now shown in the plots. Also the downstream failure of an aggregate core link is missing from the data. This is because there was an error in the experiments run for this specific failure. By the time this was noticed it was already clear that the logs were having too great of an effect on the convergence time and that they were only going to be used to statistically evaluate the effect of the *dead-interval* for router failures.

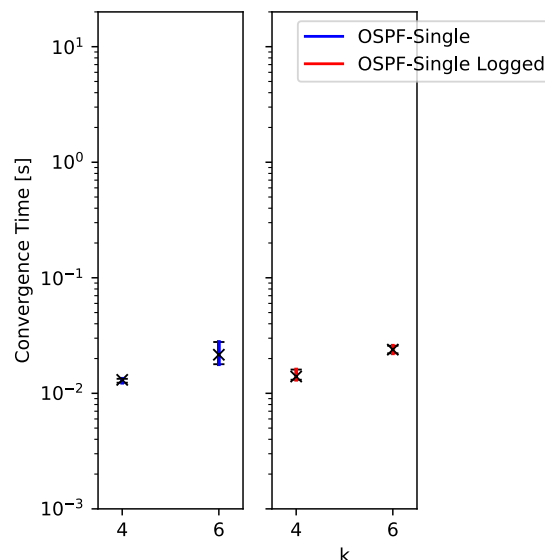


Figure 4.26: Comparison of the convergence times for OSPF with and without logs enabled on the standard topology for an upstream link failure between an edge and an aggregate router.

Table 4.9: Median values for Figure 4.26.

k	4	6
OSPF-Single	0.013	0.022
OSPF-Single Logged	0.014	0.024

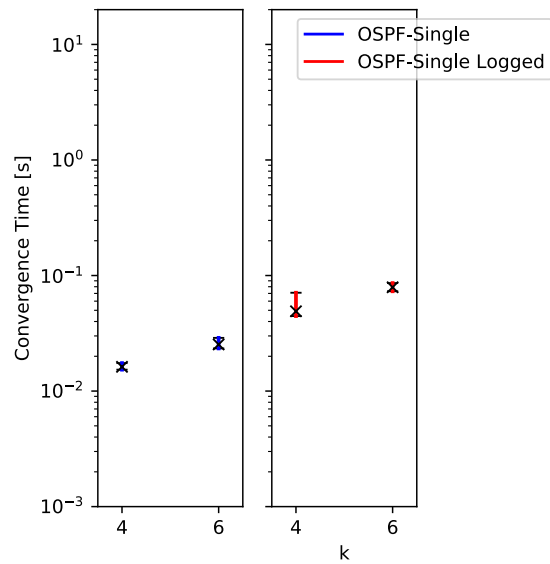


Figure 4.27: Comparison of the convergence times for OSPF with and without Logs enabled on the standard topology for a downstream link failure between an edge and an aggregate router.

Table 4.10: Median values for Figure 4.27.

k	4	6
OSPF-Single	0.016	0.025
OSPF-Single Logged	0.049	0.079

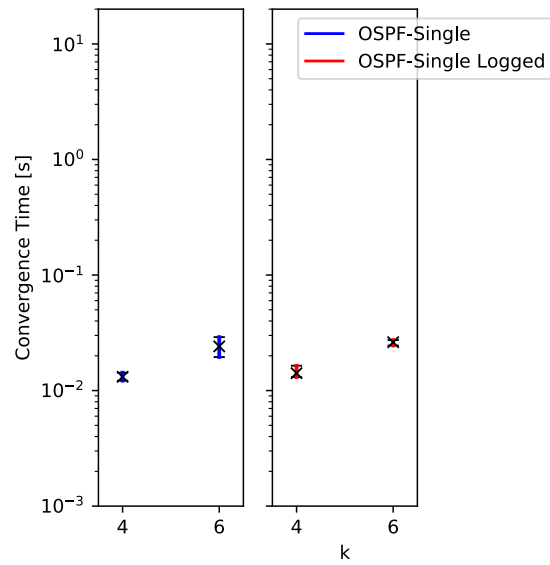


Figure 4.28: Comparison of the convergence times for OSPF with and without logs enabled on the standard topology for an upstream link failure between an aggregate and a core router.

Table 4.11: Median values for Figure 4.28.

k	4	6
OSPF-Single	0.013	0.024
OSPF-Single Logged	0.014	0.026

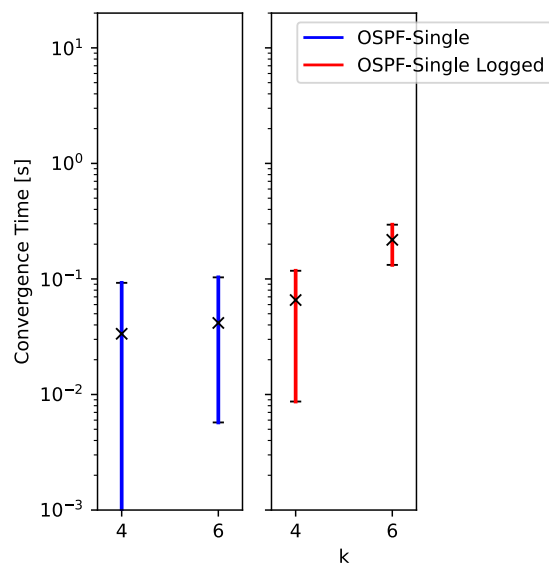


Figure 4.29: Comparison of the convergence times for OSPF with and without Logs enabled on the standard topology for an upstream failure of an aggregate router.

Table 4.12: Median values for Figure 4.29.

k	4	6
OSPF-Single	0.034	0.042
OSPF-Single Logged	0.066	0.218

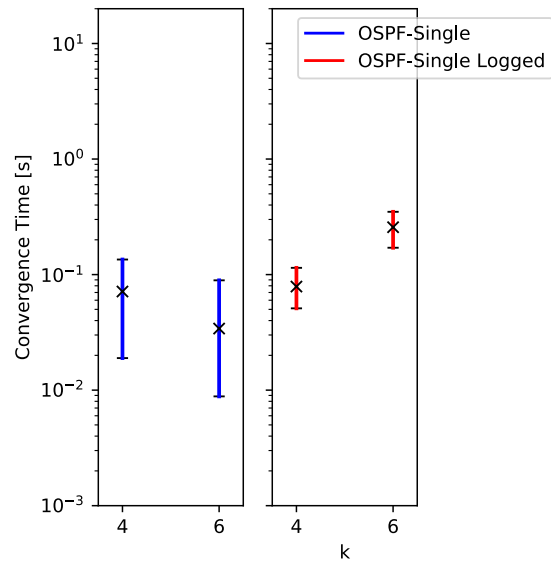


Figure 4.30: Comparison of the convergence times for OSPF with and without Logs enabled on the standard topology for an downstream failure of an aggregate router.

Table 4.13: Median values for Figure 4.30.

k	4	6
OSPF-Single	0.071	0.034
OSPF-Single Logged	0.079	0.257

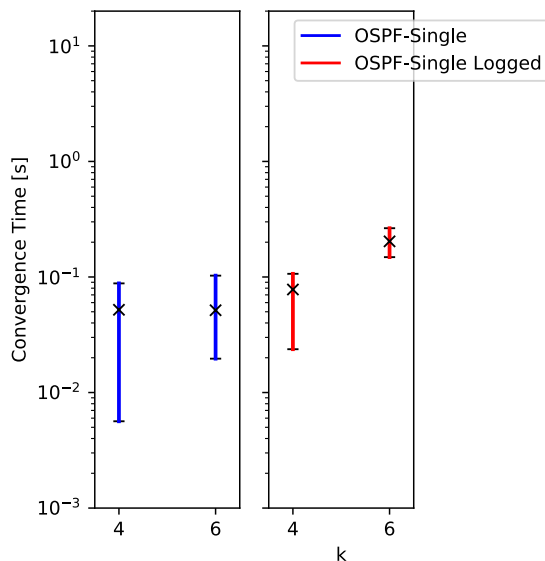


Figure 4.31: Comparison of the convergence times for OSPF with and without Logs enabled (WOL) on the standard topology for an failure of a core router.

Table 4.14: Median values for Figure 4.31.

k	4	6
OSPF-Single	0.052	0.052
OSPF-Single Logged	0.078	0.203

All the plots show that, generally, logging slowed down convergence. It is interesting to note that this effect is always greater for the downstream failures. One possible explanation could be that the flooding algorithm and the forwarding of link state advertisements were affected by the processing load on the routers since the message of the updated link state needs to propagate through more routers to reach the decision router for downstream failures.

4.4.2 OSPF Single Area

As expected for the single area OSPF configuration the protocol behaved very similarly for the different experiments. The shortest path algorithm should not be effected at all by the additional prefixes advertised through the network since the SPF calculations are based on the links. Some differences were found for $k = 8$ for the failure of a downstream aggregate router, but the errors introduced by the *dead-interval* correction might account for this difference (Figure 4.37). Another significant difference were found also for $k = 8$ for the downstream failure of a core aggregate link but the 5th and 95th percentile are near identical there, so it is likely due to the low iteration count (Figure 4.35). The path plots did not indicate a significantly different behaviour for these cases.

The additional links and nodes introduced in the larger destination pod topology have a very slight effect on the total size of the network and therefore have little effect on the convergence times. There was one exception:

Often, for $k = 4$ on this topology, a much higher variance was found for the convergence times throughout the experiments. The problem described at the beginning of this section, iterations with no observed convergence, occurred unusually often for this specific configuration, which reduced the number of valid iterations to about 15 from 30.

The reason for this is that for $k = 4$ the change in topology leads to an oscillating behaviour in the protocol. The path logs show a very high rate of oscillation between different paths. Over 40 different paths were recorded for each type of failure, in contrast to just a few distinct paths during other experiments. Due to these oscillations, for many of the iterations, the flow will have

reconverged onto a different route between the original path calculation and the failure along that path being triggered.

The convergence times are shown in figures 4.32 to 4.38.

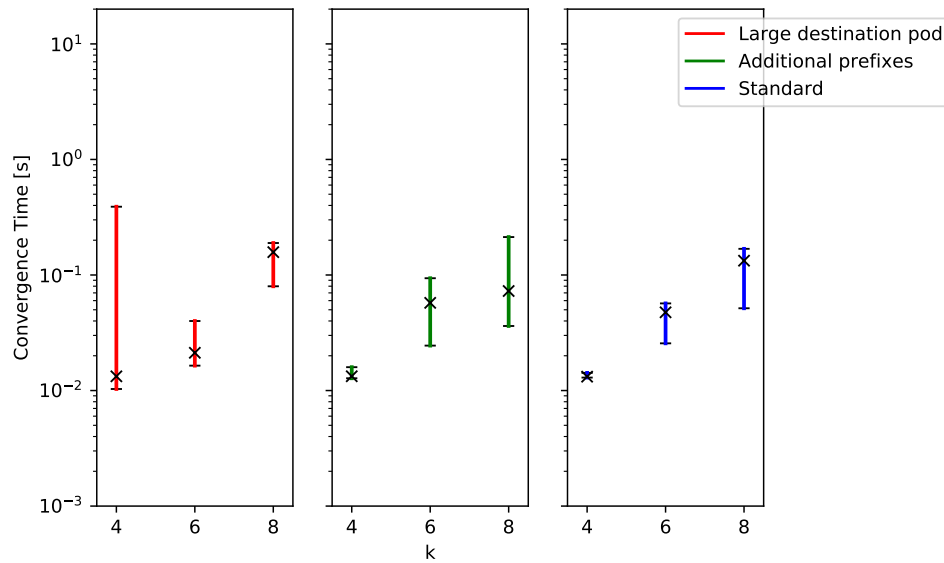


Figure 4.32: Comparison of the convergence times for OSPF with a single area for a failure of an upstream link between an edge and an aggregate router.

Table 4.15: Median values for Figure 4.32.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.013	0.021	0.158
Additional prefixes	0.013	0.057	0.073
Standard	0.013	0.048	0.133

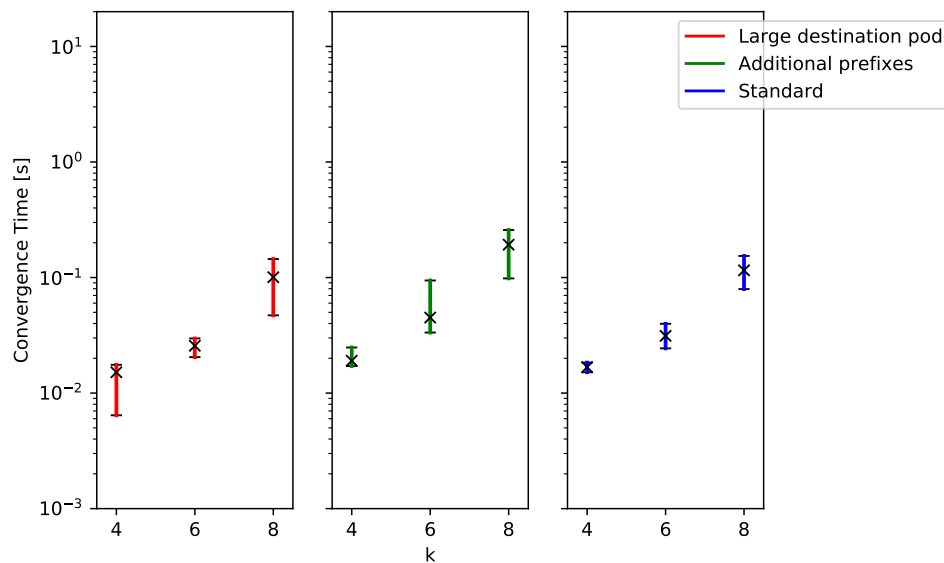


Figure 4.33: Comparison of the convergence times for OSPF with a single area for a failure of a downstream link between an edge and an aggregate router.

Table 4.16: Median values for Figure 4.33.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.015	0.026	0.101
Additional prefixes	0.019	0.045	0.192
Standard	0.017	0.031	0.115

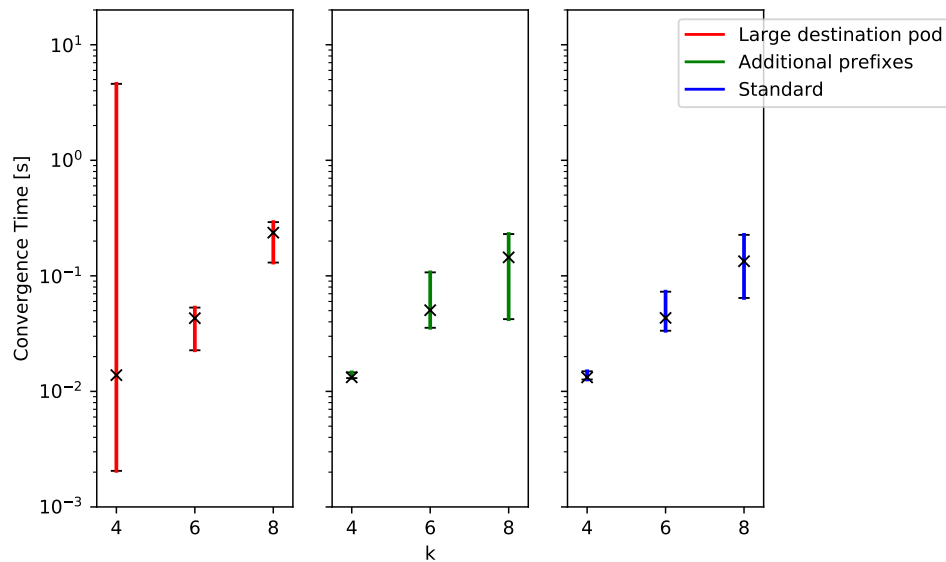


Figure 4.34: Comparison of the convergence times for OSPF with a single area for a failure of an upstream link between an aggregate and a core router.

Table 4.17: Median values for Figure 4.34.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.014	0.043	0.236
Additional prefixes	0.013	0.05	0.145
Standard	0.013	0.043	0.134

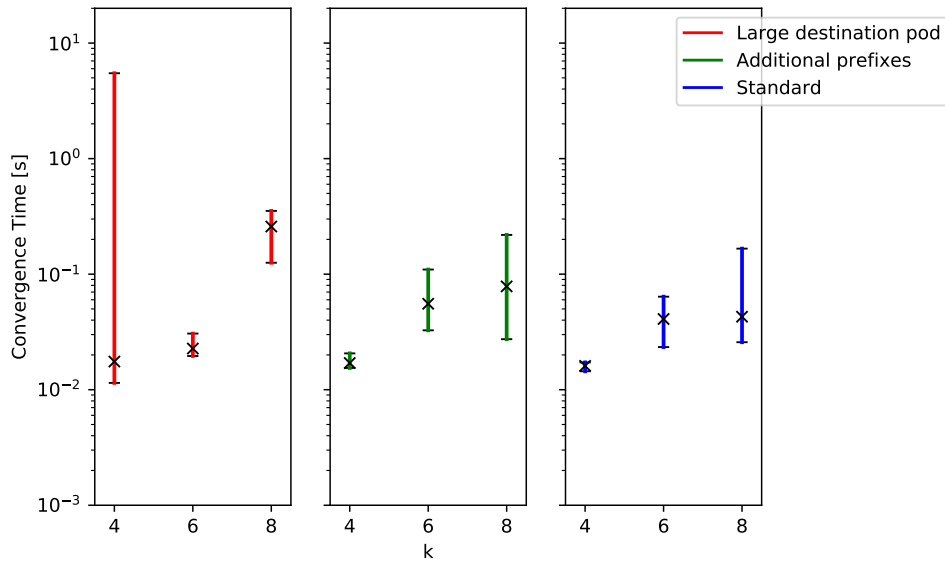


Figure 4.35: Comparison of the convergence times for OSPF with a single area for a failure of an downstream link between an aggregate and a core router.

Table 4.18: Median values for Figure 4.35.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.018	0.023	0.258
Additional prefixes	0.017	0.055	0.078
Standard	0.016	0.041	0.043

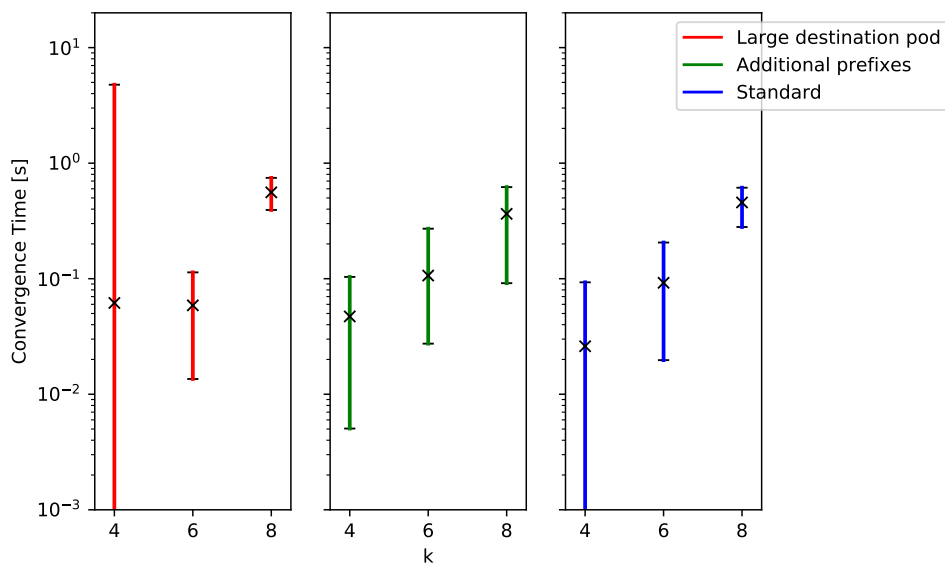


Figure 4.36: Comparison of the convergence times for OSPF with a single area for a failure of an upstream aggregate router.

Table 4.19: Median values for Figure 4.36.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.062	0.059	0.559
Additional prefixes	0.047	0.106	0.364
Standard	0.026	0.092	0.457

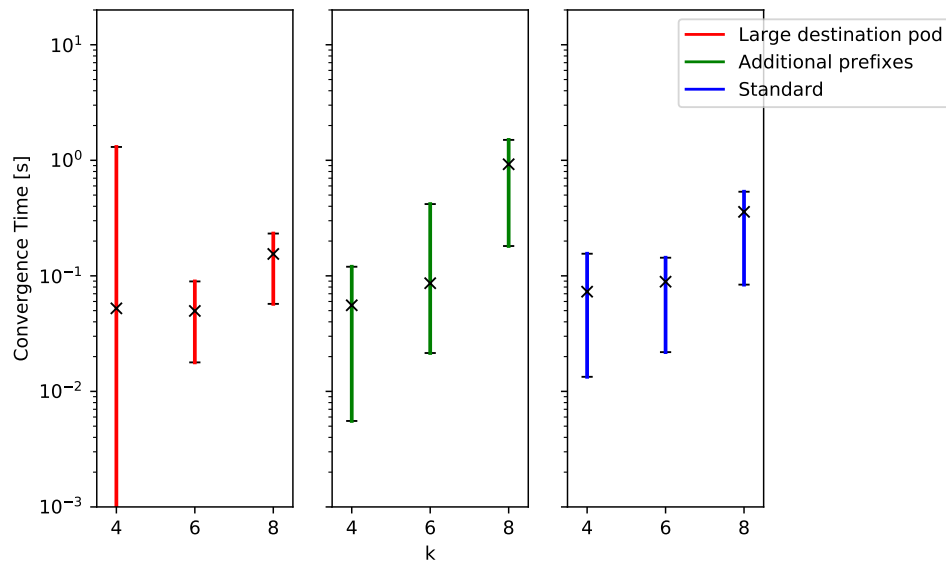


Figure 4.37: Comparison of the convergence times for OSPF with a single area for a failure of a downstream aggregate router.

Table 4.20: Median values for Figure 4.37.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.052	0.05	0.155
Additional prefixes	0.056	0.086	0.924
Standard	0.073	0.089	0.358

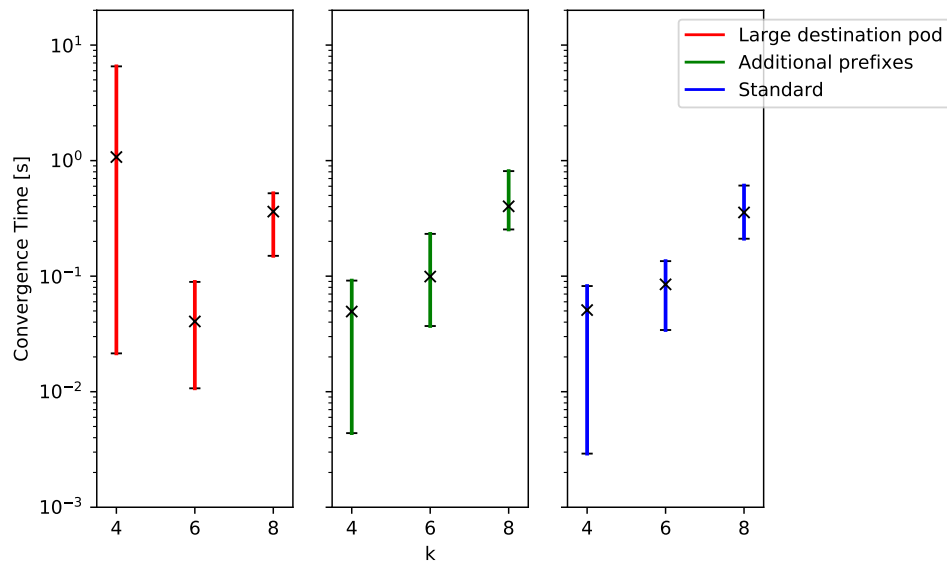


Figure 4.38: Comparison of the convergence times for OSPF with a single area for a failure of a core router.

Table 4.21: Median values for Figure 4.38.

k	4 (s)	6 (s)	8 (s)
Large destination pod	1.074	0.041	0.362
Additional prefixes	0.049	0.099	0.402
Standard	0.051	0.085	0.356

One of the oscillating paths found for $k = 4$ is shown in figure 4.39.

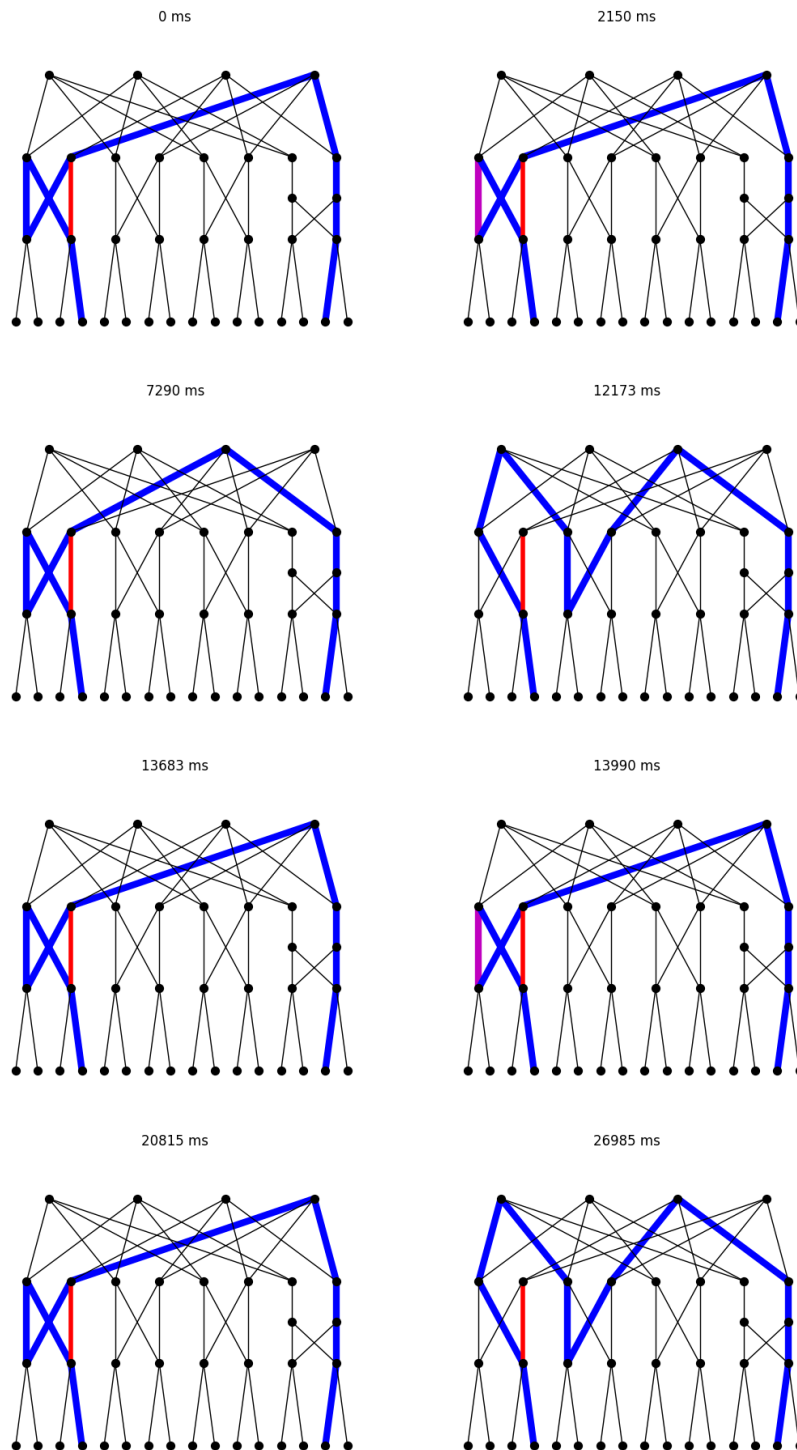


Figure 4.39: One of the oscillating paths on the topology with an enlarged destination pod for OSPF with a single area and $k = 4$. This was observed for a failure of an upstream edge aggregate link where no packet loss was observed. This is why the path logging script never stopped recording as the system was in a valid state from the beginning and it gave a full view of the oscillation throughout the 30 seconds of the experiment.

4.4.3 OSPF Multiple Areas

For this configuration the results are extremely similar across the different topologies. This is as expected for OSPF. For upstream link failures the additional prefixes seem to be stabilising for

$k = 8$, this effect could not be explained using the available analysis tools. The convergence times are shown in the figures 4.40 to 4.46.

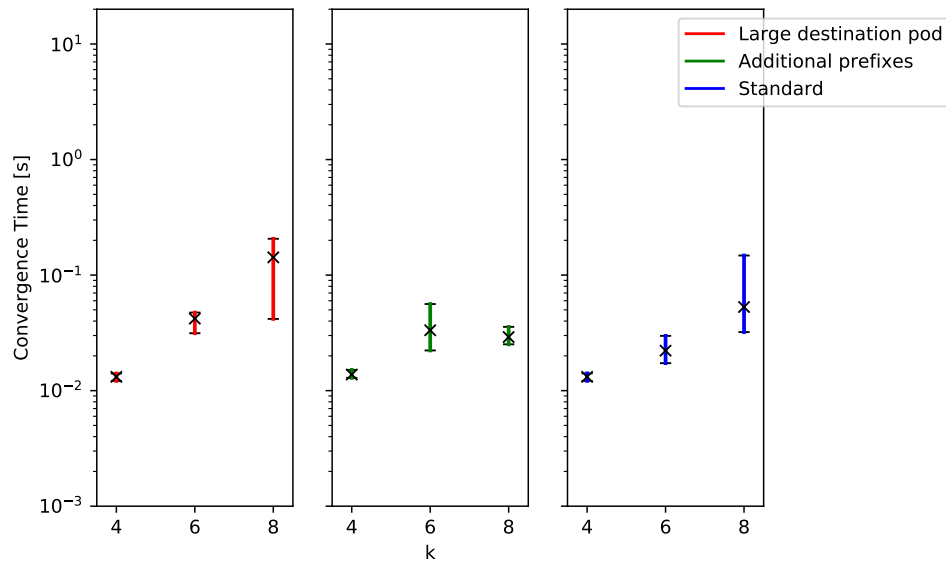


Figure 4.40: Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream link between an edge and an aggregate router.

Table 4.22: Median values for Figure 4.40.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.013	0.042	0.142
Additional prefixes	0.014	0.033	0.029
Standard	0.013	0.022	0.053

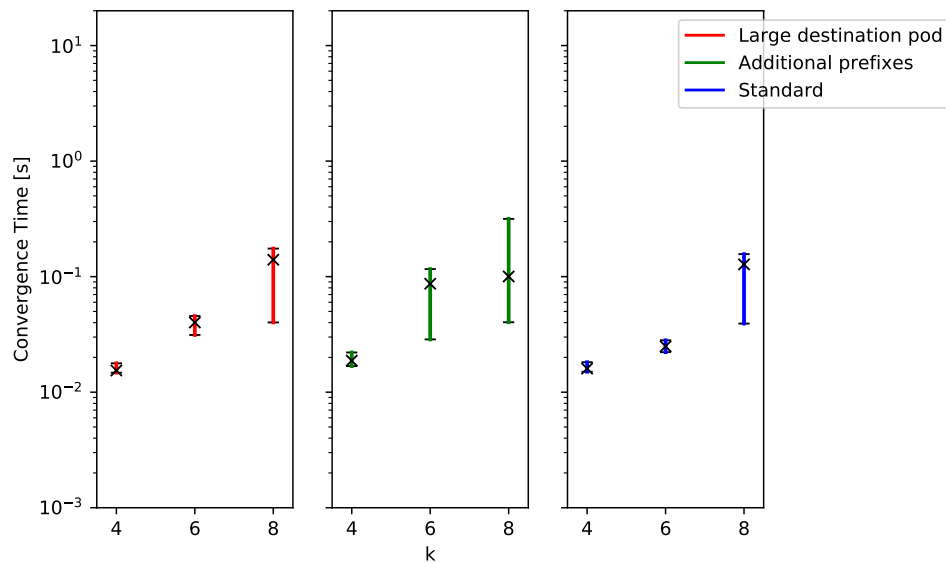


Figure 4.41: Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream link between an edge and an aggregate router.

Table 4.23: Median values for Figure 4.41.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.015	0.04	0.14
Additional prefixes	0.019	0.087	0.1
Standard	0.016	0.025	0.128

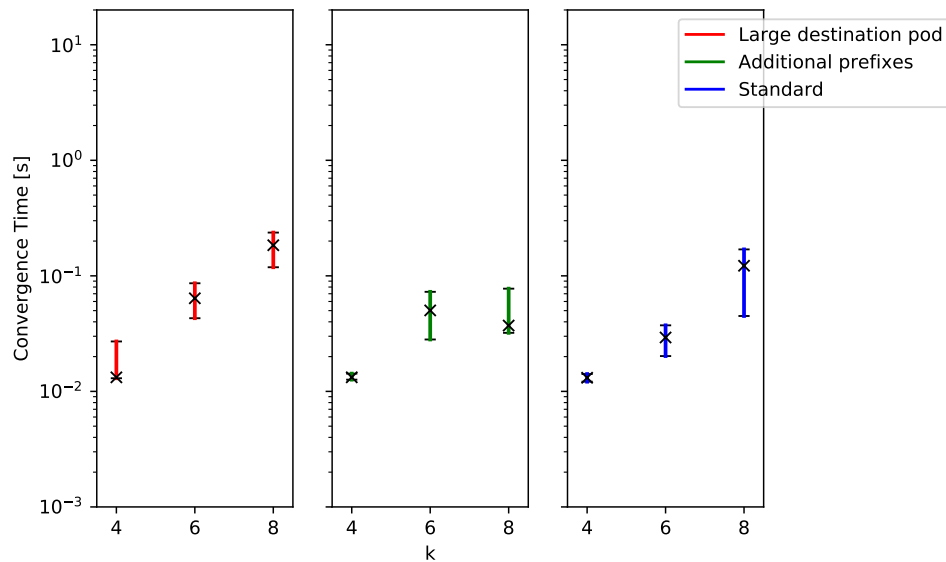


Figure 4.42: Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream link between an aggregate and a core router.

Table 4.24: Median values for Figure 4.42.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.013	0.064	0.184
Additional prefixes	0.013	0.05	0.037
Standard	0.013	0.029	0.122

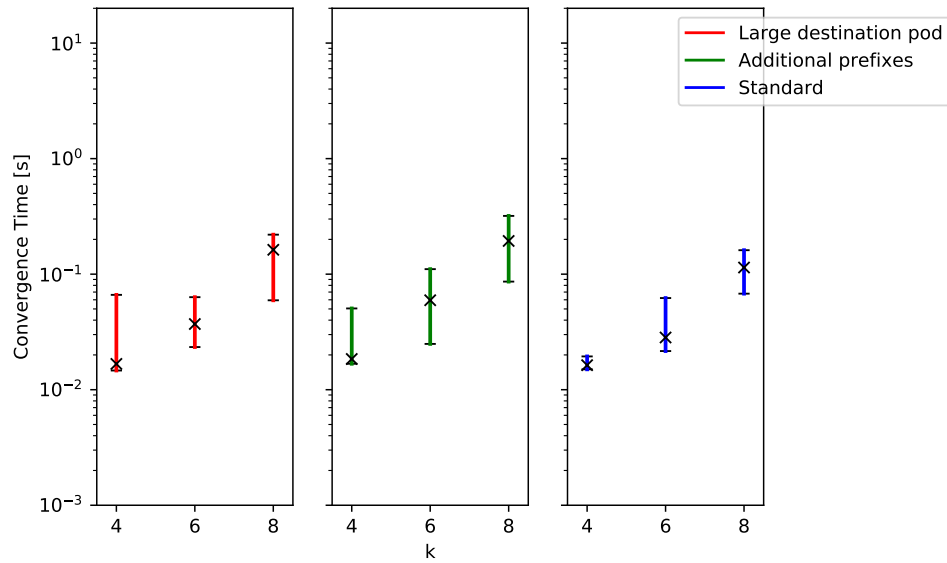


Figure 4.43: Comparison of the convergence times for OSPF with multiple areas for a failure of an downstream link between an aggregate and a core router.

Table 4.25: Median values for Figure 4.43.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.017	0.037	0.162
Additional prefixes	0.018	0.059	0.194
Standard	0.016	0.028	0.114

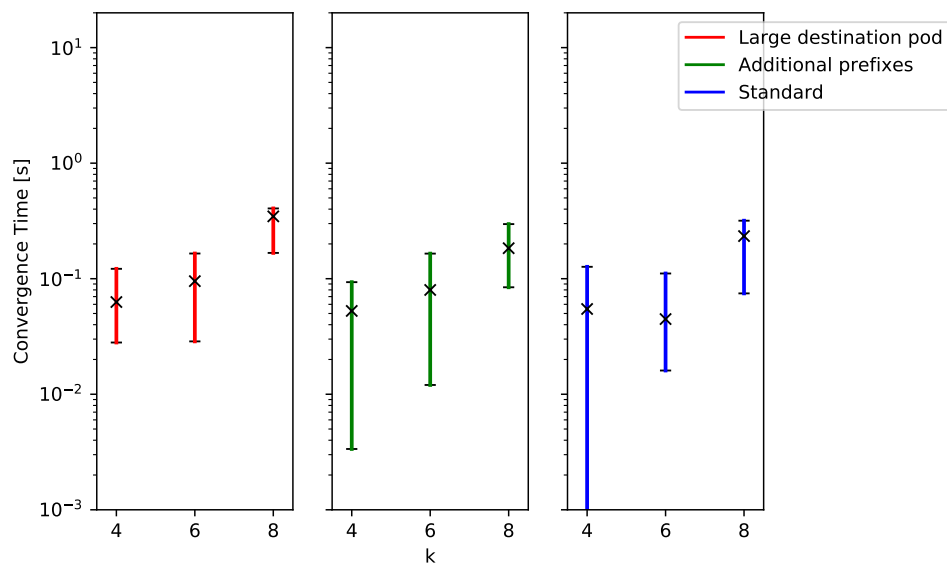


Figure 4.44: Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream aggregate router.

Table 4.26: Median values for Figure 4.44.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.063	0.095	0.346
Additional prefixes	0.053	0.08	0.184
Standard	0.055	0.045	0.234

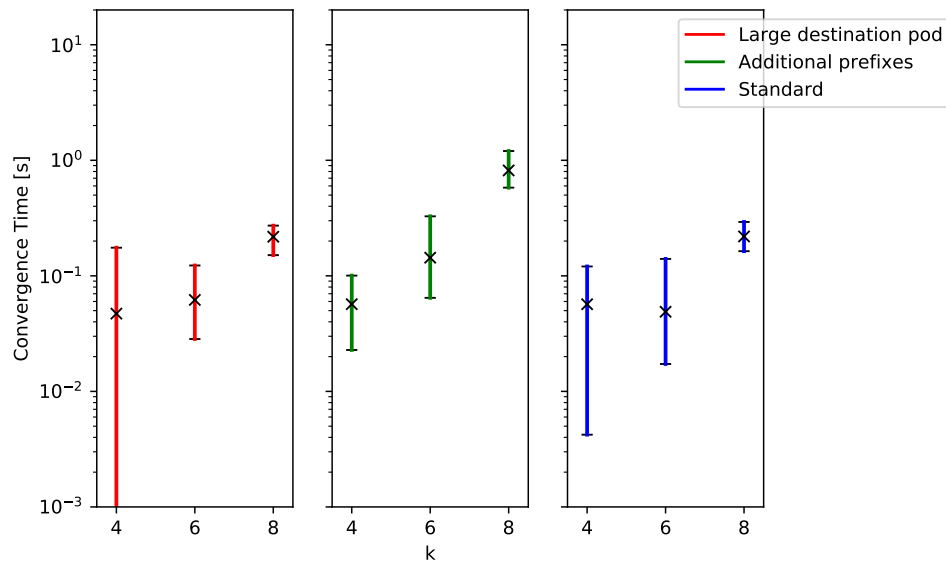


Figure 4.45: Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream aggregate router.

Table 4.27: Median values for Figure 4.45.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.047	0.062	0.218
Additional prefixes	0.057	0.144	0.817
Standard	0.057	0.049	0.219

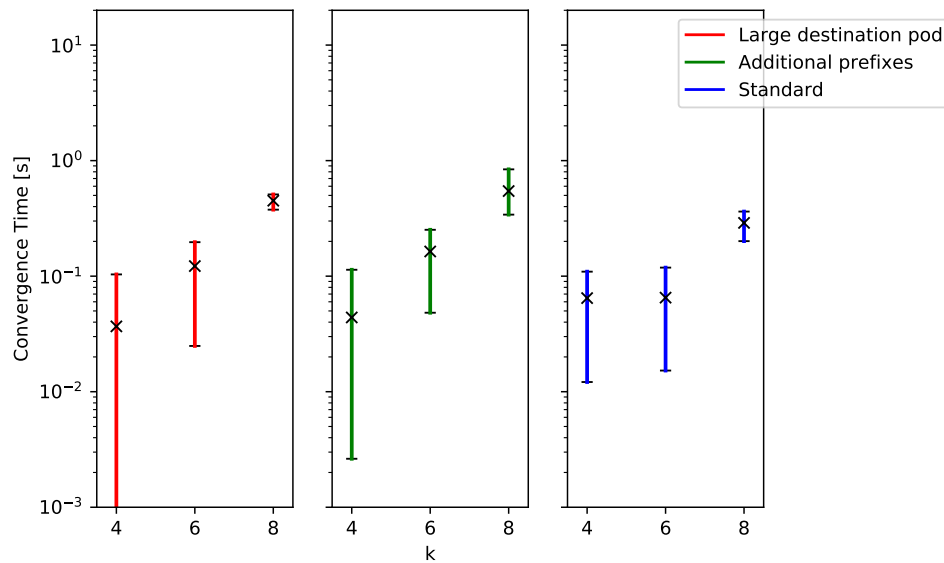


Figure 4.46: Comparison of the convergence times for OSPF with multiple areas for a failure of a core router.

Table 4.28: Median values for Figure 4.46.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.037	0.122	0.45
Additional prefixes	0.044	0.164	0.545
Standard	0.065	0.065	0.289

4.4.4 BGP

The additional prefixes did not result in any change in convergence time for $k = 4$ where the two edge routers in the destination pod advertise 400 prefixes instead of 4. For $k = 6$ where the three edge routers in the destination pod advertised 900 instead of 9 prefixes some changes were observed for certain types of failures. For $k = 8$ where the four aggregate routers in the destination pod advertised 1600 prefixes instead of 16 the difference was most prevalent when failing the aggregate routers. This effect was so large that the convergence time would generally exceed the 30s runtime of the experiments, and so there was too little comparable data to make it a relevant for comparison with the other experiments. This is why figures 4.53 and 4.54 are missing some data points. This happened less frequently when a core router was failed, but still too often for this data to be considered in the comparison (Figure ??). The next section (4.4.5) is a more detailed discussion on the problems encountered while running BGP with additional prefixes.

As expected the larger destination pod did not have any effect on the upstream failures for BGP. The failure which was most affected was the downstream failure of an edge-aggregate link (Figure 4.50). For $k = 4$ the path plots showed that, for 29 of 30 iterations, the downstream aggregate router changed its mind and converged to a longer path via the second downstream aggregate router (Figure 4.47).

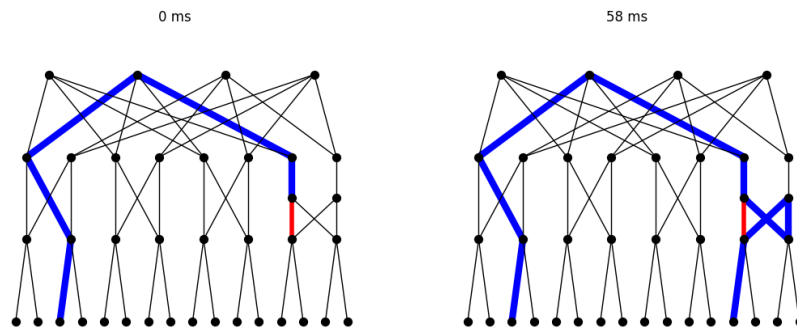


Figure 4.47: The most observed path evolution for BGP and $k = 4$ for a downstream failure of an edge aggregate link.

In only one iteration was the fault propagation back to the upstream edge router observed (Figure 4.48). For this iteration the path plots still show the upstream edge router attempting to route the flow to the second aggregate upstream router instead of choosing the first one, from which it should not have received a withdrawn message.

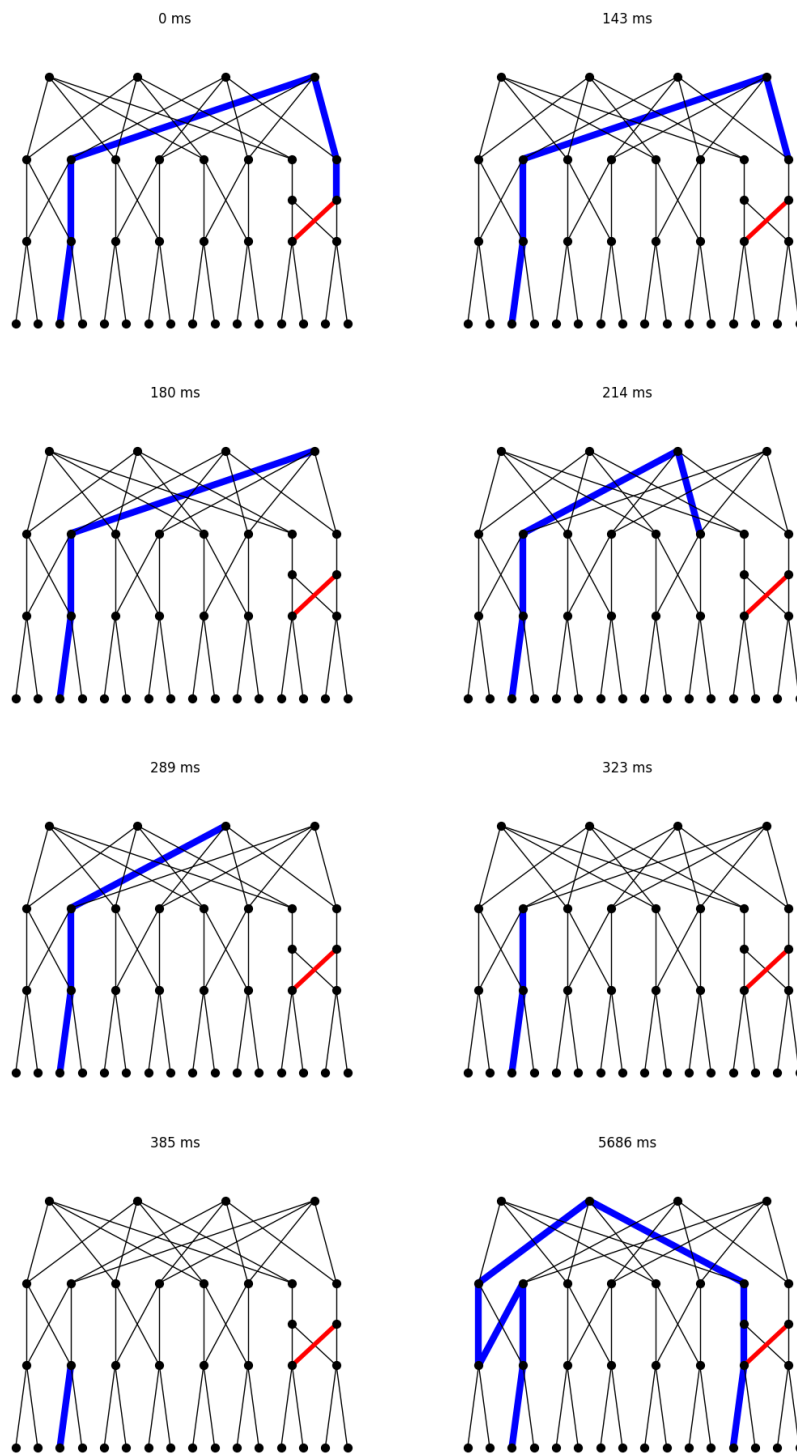


Figure 4.48: The backwards fault propagation observed for one iteration of BGP with the larger destination pod on $k = 4$

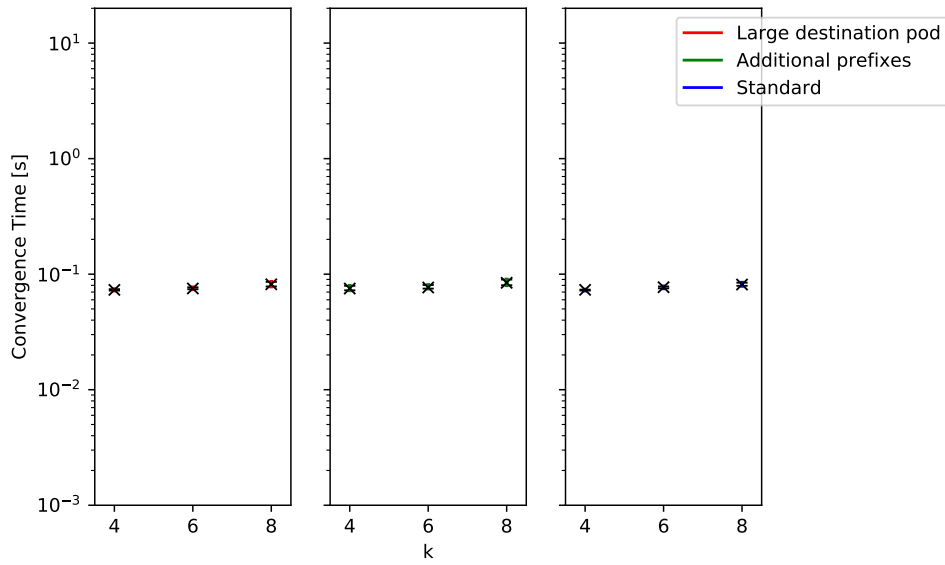


Figure 4.49: Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream link between an egde and an aggregate router.

Table 4.29: Median values for Figure 4.49.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.013	0.042	0.142
Additional prefixes	0.014	0.033	0.029
Standard	0.013	0.022	0.053

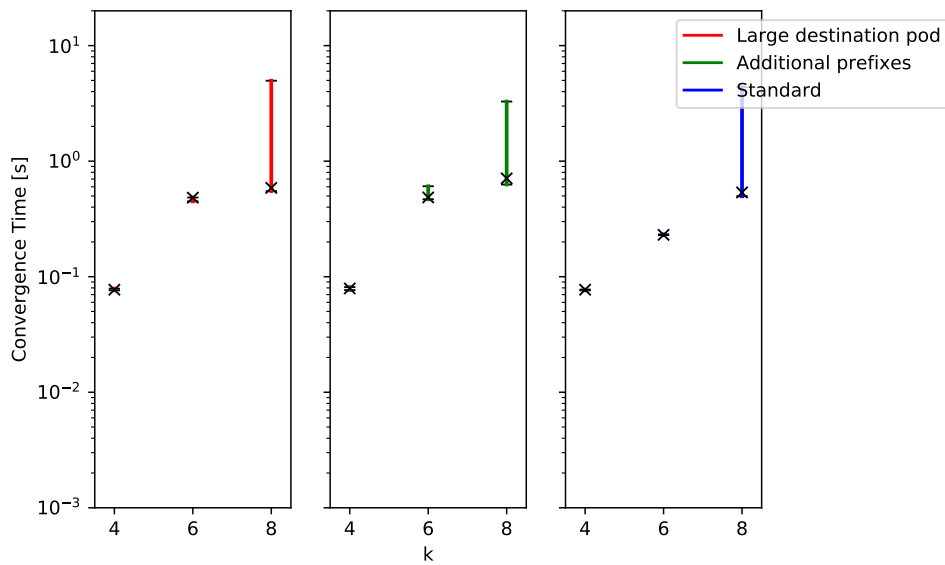


Figure 4.50: Comparison of the convergence times for OSPF with multiple areas for a failure of a downstream link between an egde and an aggregate router.

Table 4.30: Median values for Figure 4.50.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.015	0.04	0.14
Additional prefixes	0.019	0.087	0.1
Standard	0.016	0.025	0.128

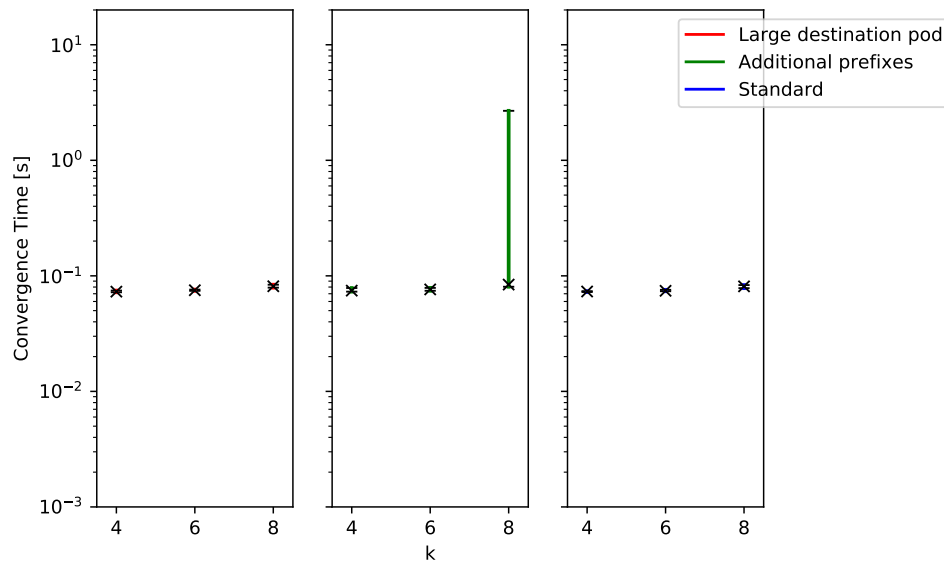


Figure 4.51: Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream link between an aggregate and a core router.

Table 4.31: Median values for Figure 4.51.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.013	0.064	0.184
Additional prefixes	0.013	0.05	0.037
Standard	0.013	0.029	0.122

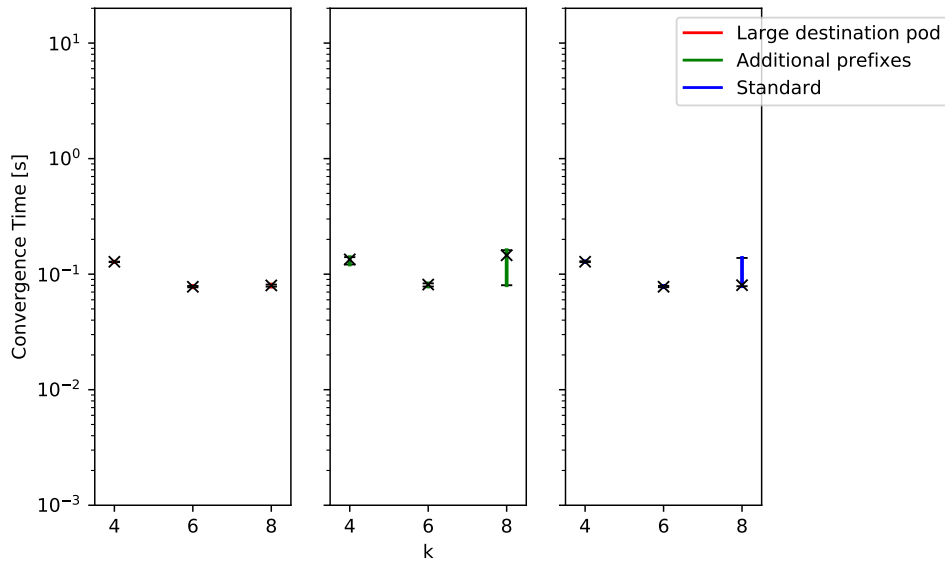


Figure 4.52: Comparison of the convergence times for OSPF with multiple areas for a failure of an downstream link between an aggregate and a core router.

Table 4.32: Median values for Figure 4.52.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.017	0.037	0.162
Additional prefixes	0.018	0.059	0.194
Standard	0.016	0.028	0.114

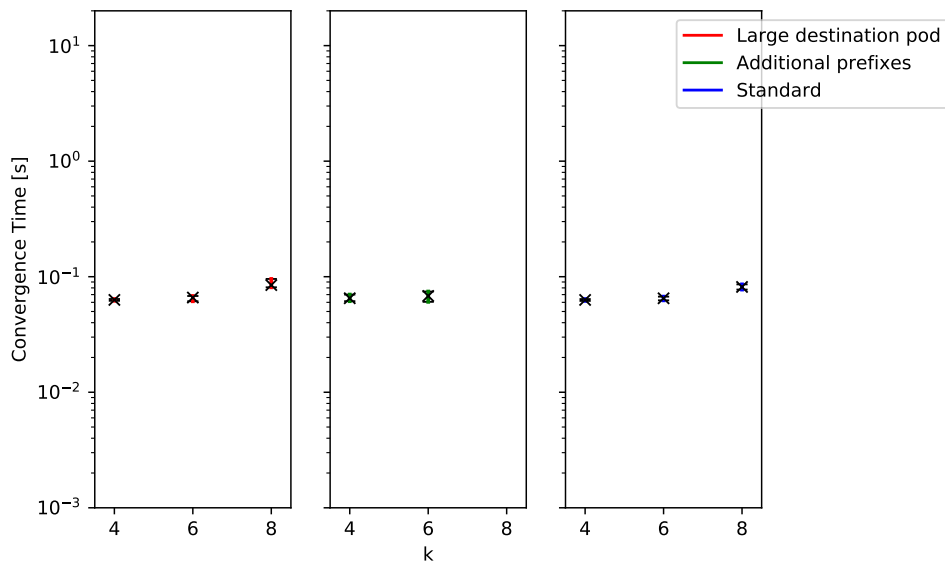


Figure 4.53: Comparison of the convergence times for OSPF with multiple areas for a failure of an upstream aggregate router.

Table 4.33: Median values for Figure 4.44.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.063	0.095	0.346
Additional prefixes	0.053	0.08	0.184
Standard	0.055	0.045	0.234

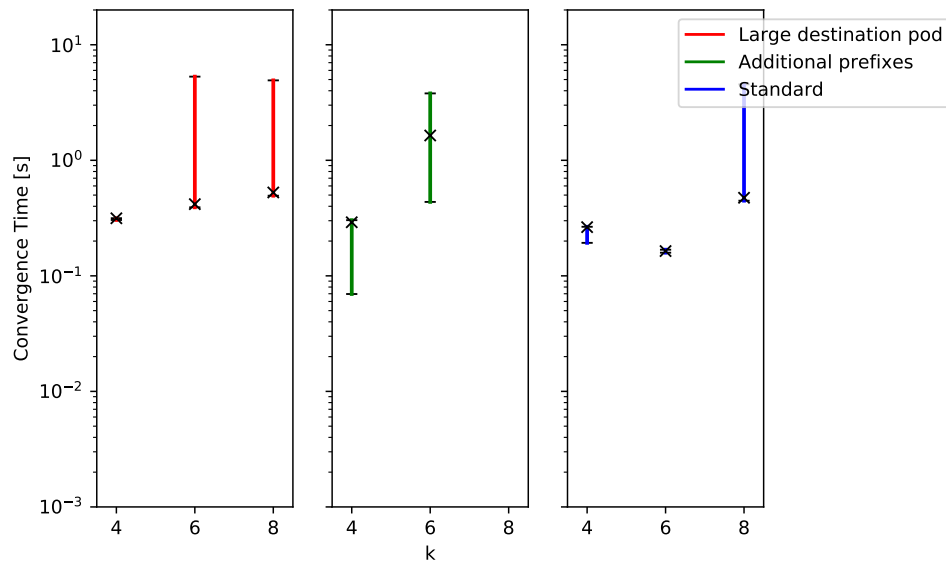


Figure 4.54: Comparison of the convergence times for OSPF with multiple areas for a failure of an downstream aggregate router.

Table 4.34: Median values for Figure 4.54.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.047	0.062	0.218
Additional prefixes	0.057	0.144	0.817
Standard	0.057	0.049	0.219

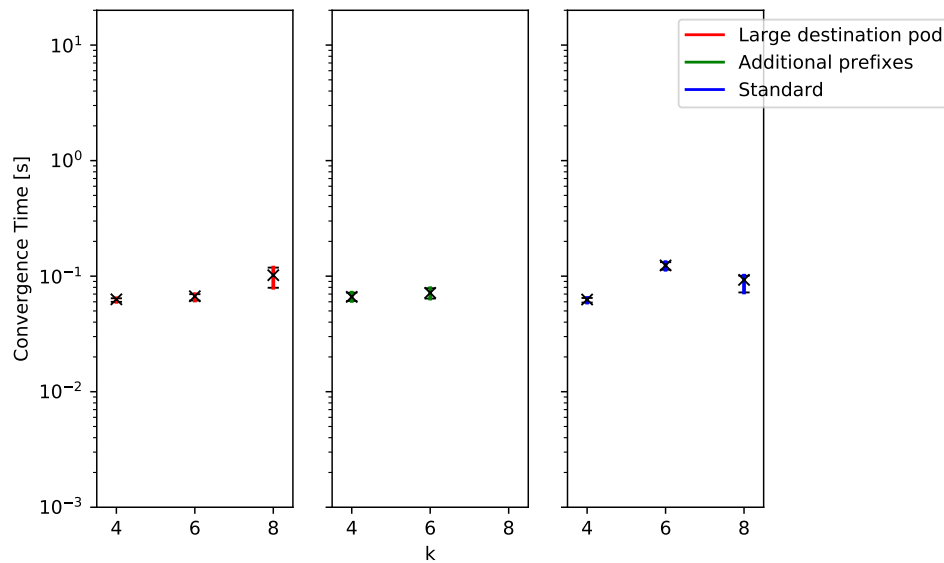


Figure 4.55: Comparison of the convergence times for OSPF with multiple areas for a failure of a core router.

Table 4.35: Median values for Figure 4.55.

k	4 (s)	6 (s)	8 (s)
Large destination pod	0.063	0.067	0.102
Additional prefixes	0.066	0.072	-
Standard	0.063	0.124	0.093

4.4.5 Adding More Prefixes to BGP

For experiments with the additional prefixes attempts were made to add very big numbers of additional prefixes to the last pod to run only for BGP with $k = 4$ in order to get an idea of how it would scale. In doing this a bug in the Propane compiler was discovered:

When adding more than 6550 prefixes the configurations the compiler produced were no longer valid. If left unspecified propane assigns unique router IDs by counting down from 65534, which is the highest allowed identifier. Propane saves every prefix on its own into a prefix list which is then added to a route map with the name *rm-in* which are differentiated by different sequence numbers and given the permit attribute. Those sequence numbers start at 10 and are incremented in steps of 10 for the total number of prefixes. When the sequence numbers and the BGP identifiers clash configs are no longer valid and applying them with Quagga's *vtys* client fails with the error message: *Unknown command*.

This resulted in an effective upper limit to the number of additional prefixes that could be added. It was attempted to set the prefix multiplier to 1600, 800, and 400. For $k = 4$ this means 6400, 3200, and 1600 prefixes on the two edge routers in the destination pods, since they advertise four prefixes on the standard topology. The 1600 multiplier kept the system from converging at bootstrap. The system with the multiplier set to 800 was able to converge after setting up but after the downstream edge aggregate link failure it failed to return to a stable state of convergence. For the multiplier set to 400 the experiments ran through but there was only time to run 10 iterations.

The experiments which were most affected by the additional prefixes were the failure of an upstream edge aggregate link (Figure 4.56), the failure of a downstream aggregate router (Figure 4.61) and the failure of a core router (Figure 4.62). For the failure of a downstream edge aggregate and aggregate core link the percentiles were elevated while the median stayed constant (Figures 4.57 and 4.59). The only experiment which seemed unaffected was the upstream failure of an aggregate router and an aggregate core link (Figures 4.60 and 4.60).

Generally when outliers were observed the network showed irregular behaviour and often the convergence times exceeded the 30 seconds during which the experiment was run so the times shown in these plots could not be used to draw direct conclusions about the behaviour with more prefixes. The only thing that could be showed is that there is indeed an effect on the convergence for BGP when adding more prefixes to the network, but due to the nature of this effect it could not be well described. More research and a different experiment design would be required to actually study the behaviour of BGP with additional prefixes given this topology.

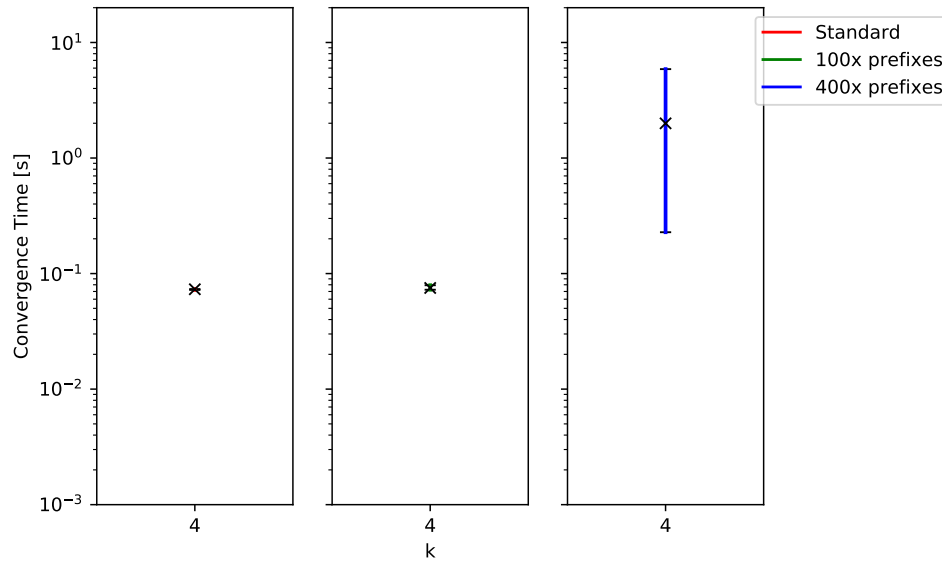


Figure 4.56: Upstream failure of an edge aggregate link for BGP on the topology with $k = 4$ for different amounts of additional prefixes.

Table 4.36: Median values for Figure 4.56.

k	4 (s)
Standard	0.073
100x prefixes	0.075
400x prefixes	1.996

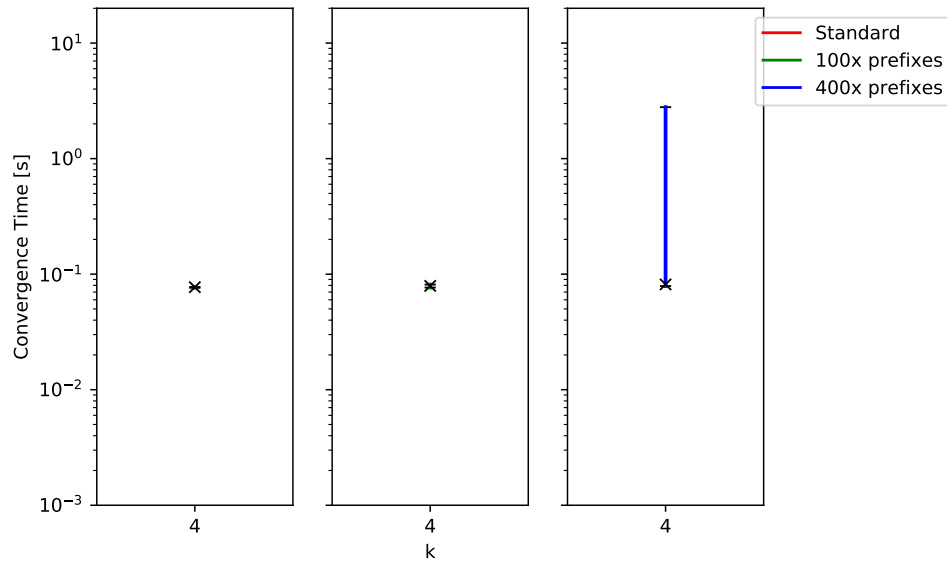


Figure 4.57: Downstream failure of an edge aggregate link for BGP on the topology with $k = 4$ for different amounts of additional prefixes.

Table 4.37: Median values for Figure 4.57.

k	4 (s)
Standard	0.077
100x prefixes	0.079
400x prefixes	0.082

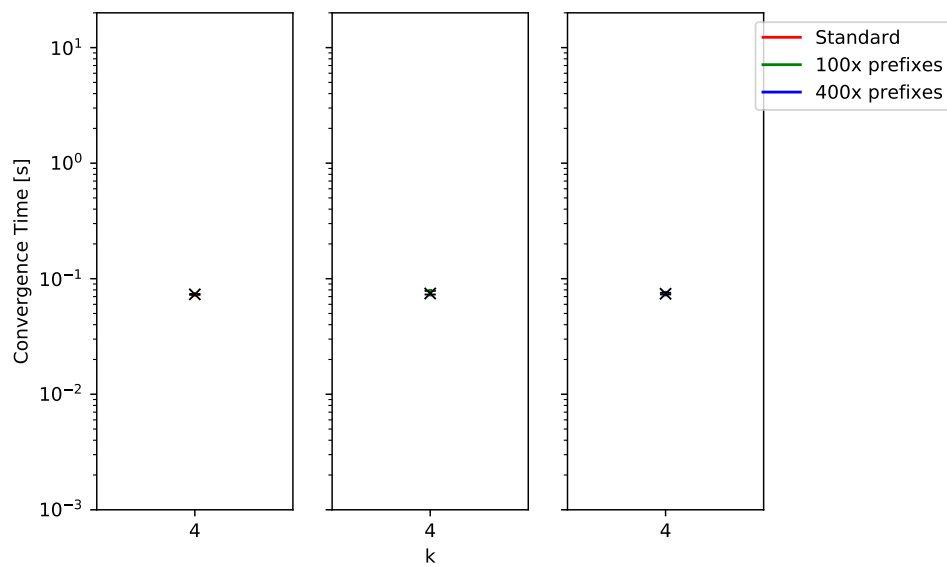


Figure 4.58: Upstream failure of an aggregate core link for BGP on the topology with $k = 4$ for different amounts of additional prefixes.

Table 4.38: Median values for Figure 4.58.

k	4 (s)
Standard	0.073
100x prefixes	0.074
400x prefixes	0.074

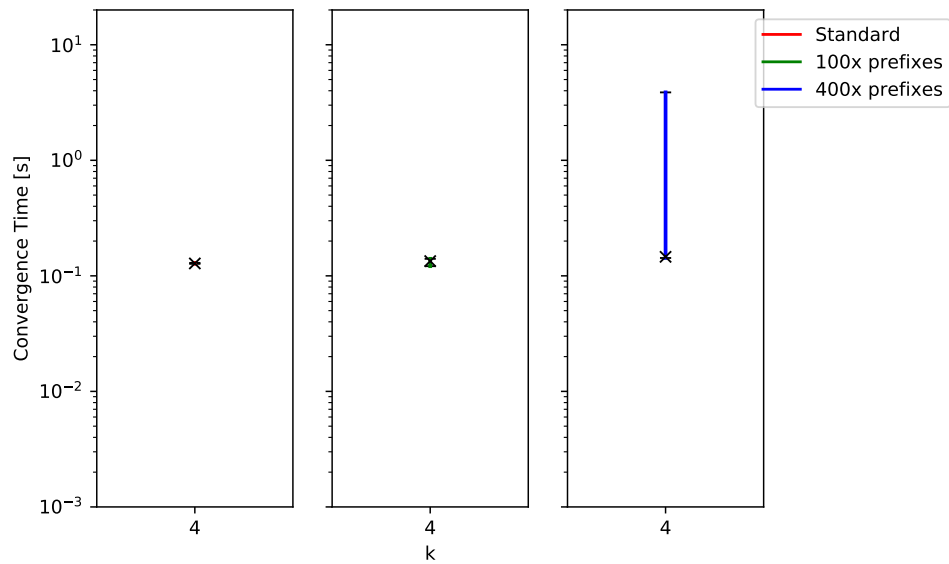
Figure 4.59: Downstream failure of an aggregate core link for BGP on the topology with $k = 4$ for different amounts of additional prefixes.

Table 4.39: Median values for Figure 4.59.

k	4 (s)
Standard	0.128
100x prefixes	0.134
400x prefixes	0.146

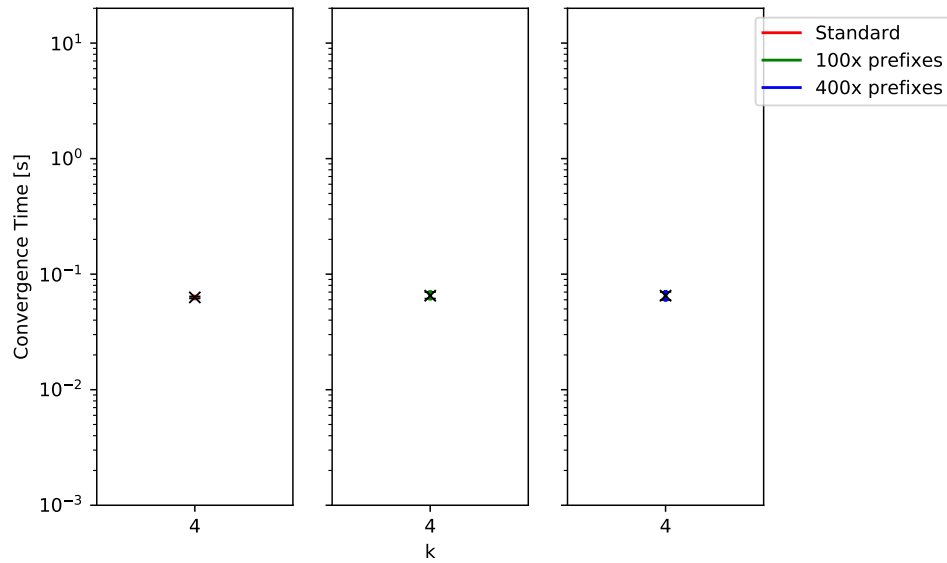


Figure 4.60: Upstream failure of an aggregate router for BGP on the topology with $k = 4$ for different amounts of additional prefixes.

Table 4.40: Median values for Figure 4.60.

k	4 (s)
Standard	0.063
100x prefixes	0.065
400x prefixes	0.065

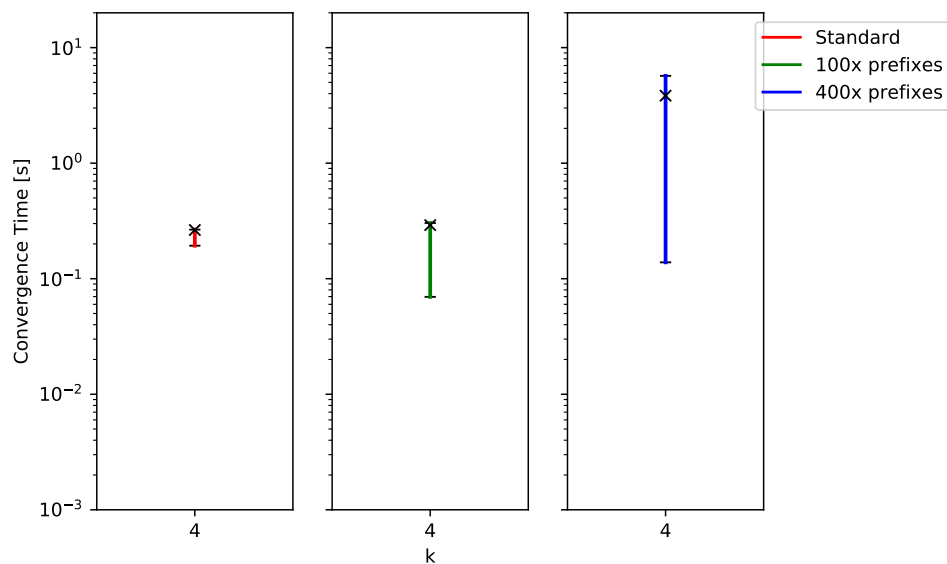


Figure 4.61: Downstream failure of an aggregate router for BGP on the topology with $k = 4$ for different amounts of additional prefixes.

Table 4.41: Median values for Figure 4.61.

k	4 (s)
Standard	0.264
100x prefixes	0.291
400x prefixes	3.836

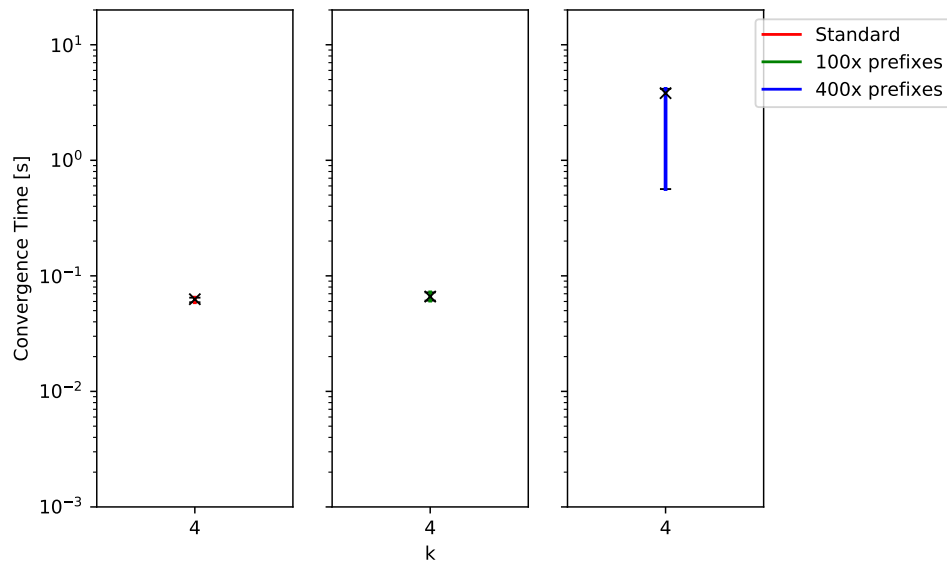
Figure 4.62: Core failure for BGP on the topology with $k = 4$ for different amounts of additional prefixes.

Table 4.42: Median values for Figure 4.62.

k	4 (s)
Standard	0.063
100x prefixes	0.066
400x prefixes	3.811

Chapter 5

Conclusion and Outlook

5.1 Conclusion

The main discussion led to a few key conclusions which could be drawn from the data. The OSPF protocol scales directly with the network size as expected. BGP on the other hand is very sensitive to the locality of the failure. Upstream experiments converged much quicker than downstream experiments simply because the failure did not have to propagate as far. The network sizes considered in this thesis were still relatively small compared to the scale of real datacenters. Extrapolating the results would lead to the conclusion that BGP is probably better suited for these of topologies than OSPF. One needs to consider that the number of prefixes BGP has to find shortest paths for would also increase in a real datacenter network. While the effect of this was minimal for the small topologies considered it could be that the effect will be much more pronounced in larger networks.

Furthermore the timers play a very important role. To find the ideal values more thorough evaluations of these settings would be required. No considerations were made during this paper relating to other adverse effects the minimization of these timers could have on convergence (like e.g. route flapping).

It would also be advised to test the behaviour of the protocols on a network facing real traffic burdens. The fact that only ever one flow was going through the network might have masked some problems that may arise in more realistic conditions.

5.1.1 Fat-Tree-Connections

One thing that became clear from evaluating the different behaviours of failure locations is that the severity of the worst case, the failure of a link between an edge and an aggregate router, could be reduced by connecting the fat-tree slightly differently. If the upstream aggregate router did not only have paths to specific pods across the the same downstream aggregate router it would become the decision router instead of the failure having to propagate all the way back to the upstream edge router. This would decrease the length of the propagation path and lead to lower convergence times.

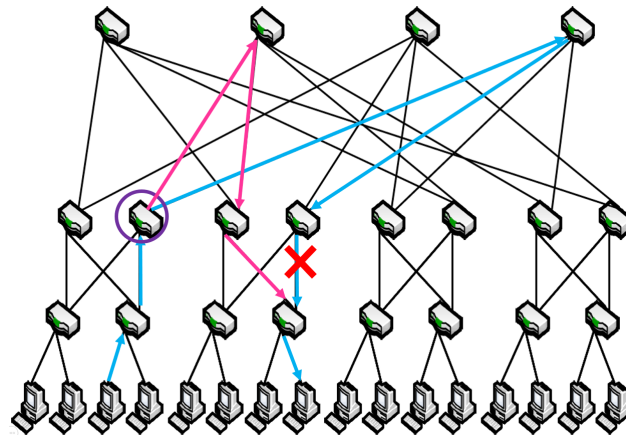


Figure 5.1: An alternative connection scheme between the aggregate and the core routers. For flows originating at the outer pods and going into the centre pods and vice versa this lessens the path the error needs to propagate for a failure of a downstream link between an edge and an aggregate router.

5.1.2 OSPF With Multiple Areas

The fat-tree topology made the grouping of areas difficult for this protocol. Restricting the number of areas a router may be placed in to three reduced the maximum areas possible to three since the core routers populated the backbone area and required connections to each pod. The way the topology was structured also meant that all of the aggregate routers were also Area Border Routers as they all had a connection to at least one core router in the backbone area.

5.2 Considerations for similar work

During this thesis a lot of insights were gained about potential obstacles with accurately comparing convergence times between different protocols on this particular setup were gained. Following are some suggestions on how these difficulties could be minimized or even avoided.

5.2.1 Bidirectional Forwarding Detection

To really compare the convergence times of two protocols to each other it is important to equalize as many parameters as possible. Especially the time a protocol needs to trigger a response to a failure or detect it in the first place can make up a big chunk of the convergence time as we have seen it for OSPF when killing a router. Unfortunately as we have seen during this paper just adapting timers and going through logs to try to balance out those subtle differences is not ideal.

Bidirectional Forwarding Detection is a detection protocol designed to lessen path failure detection time for all routing protocols [3]. If this could be properly enabled in Quagga it might be a possibility to unify the detection times over multiple protocol and really only observe the convergence time without the need of estimated corrections on top of the measurements.

kbfd [6] and *bfdd* [1] are both implementations of the protocol which could be enabled with Quagga but unfortunately there was not enough time to explore this possibility further during the Thesis.

5.2.2 Logging

Letting the routers print logs can be very useful to understand exactly what is happening in the background. In order to lessen the impact of over logging and to solve the problem of not having enough information in certain cases it could be considered to modify the way Quagga logs. Printing only the relevant messages instead of flooding all kinds of information would lessen the

impact the many IO operation have on convergence time. Also one could steer what exactly got logged and modify it to suit a specific purpose.

5.2.3 Flow Observation

It would have been interesting to see the path evolution until the end of each experiment instead of stopping when packet flow is restored. Some interesting observations could have been made on the time it takes the protocol to reach a stable state with a shortest path flow once more.

5.2.4 Incremental SPF Algorithm

To reduce the calculation time for OSPF it would be beneficial to choose an incremental SPF algorithm instead of a non-incremental to recompute the shortest paths. This could dramatically lower the convergence times for larger networks for OSPF and might make it a viable option for these kinds of topologies. More research would be needed to evaluate the performance of such an algorithm.

5.2.5 Statistical Validity

Often differing convergence times were observed for similar experiments where no difference was expected. The logs and path logs were often not enough to explain such differences. However, these differences were still within the variances seen for the experiments. It would therefore be beneficial to run more iterations of these experiments to reduce the potential for statistical errors and to focus on the differences that were still left unexplained.

5.2.6 Real Hardware

The simulation environment while having its advantages also comes with drawbacks. The times a router takes to realize a link or a neighbouring router is down is likely to be different than on actual hardware. A possibility would be to test this on actual hardware and combine these data with the time it takes for the protocol to converge afterwards to get a more accurate view of the subject.

Appendix A

Setup

A.1 Linux

The whole system was built on a Linux VM. The specifications are:
Linux version 3.13.0-24-generic (buildd@panlong) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1))
Section A.2 explains why this specific kernel version was chosen.

A.2 Mininet

Mininet is software for emulating networks. It creates a virtual network of hosts, switches, controllers, and links on top of Linux. Mininet version 2.1.0 was used instead of the latest version for reasons explained in A.3. There is a bug in this version when running on newer Linux kernels so it was necessary to downgrade to an older kernel [7].

A.3 MiniNEXt

MiniNEXt is an extension layer on Mininet. It was mainly used to enable the Quagga routing engine making it possible to run OSPF and BGP on the Mininet switches. It also allows for every router to have its own separate log files. MiniNEXt does not support the latest version of Mininet. [8]

A.4 Minigenerator

To generate the desired traffic Edgar Costa Molero's traffic generator was used. The code was designed to run on top of Mininet and send a flow (or multiple flows) of a certain rate and duration from a source host to a destination host via UDP or TCP. It was originally used in a paper dealing with Flow Control [22]. It was slightly modified by Edgar Moleros to work alongside MiniNEXt.

Additional Modifications: The original traffic generator would produce small bursts of traffic. This made it difficult to make precise measurements of convergence. This behaviour was implemented to avoid dropping of packets on links with a capped bandwidth. Since this experiment did not impose any such restrictions this behaviour was altered so as to generate a constant stream of packets at 1ms intervals.

The sending and receiving functions were also changed to record the packets and their payload data into files for post-processing as described in 3.1 and 3.2.

A.5 Quagga

Quagga is a routing software suite implementing various routing protocols for Unix platforms. It is open source and therefore often used in research. The routers were configured via its integrated cli for its daemons called *vtys* [12].

Quagga version *0.99.22.4* was installed using the Advanced Packaging Tool (APT).

A.5.1 ECMP

As explained in 2.3 equal cost multipath (ECMP) routing had to be disabled to get valid convergence time measurements. This can be done by either recompiling the Linux kernel or recompiling Quagga with the multipath flag changed from 64 to 1 when configuring: *-enable-multipath=1* [16].

This caused a few problems. When trying to compile Quagga from the source provided on their website [13] *make* produced errors after running the configure script with the multipath parameter. Attempts were made to build a few different recent Quagga versions, including 0.99.22.4, but each lead to different errors or had other incompatible issues when the system was set up. There was not enough time to isolate the issues in the *make* script, however, a workaround was found by using the source from the APT repositories instead of downloading it directly from Quagga. The appropriate flag was changed in the *.rules* file and the package rebuilt by running *debuild* and *dpkg -i*.

Appendix B

Declaration of Originality

Bibliography

- [1] bfd. <https://sourceforge.net/projects/bfd/>.
- [2] Bgp decision process. <http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html>.
- [3] Bidirectional forwarding detection. http://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fs_bfd.html#wp
- [4] Designing cisco network service architectures (arch): Developing an optimum design for layer 3 (ccdp). <http://www.ciscopress.com/articles/article.asp?p=1763921&seqNum=6>.
- [5] Google data centers. <http://www.artificialbrains.com/google/datacenters>.
- [6] kbfd. <http://kbfd.sourceforge.net/>.
- [7] Mininet kernel bug. <https://github.com/USC-NSL/miniNExT/issues/9>.
- [8] mininext. <https://github.com/USC-NSL/miniNExT>.
- [9] Ospf timers overview. https://www.juniper.net/documentation/en_US/junos/topics/concept/ospf-timers-overview.html.
- [10] Propane language. <http://www.cs.princeton.edu/~rbeckett/Propane/Tutorial/Background.html>.
- [11] Propane source github. <https://github.com/rabeckett/propane>.
- [12] Quagga. <http://www.nongnu.org/quagga/>.
- [13] Quagga downloads. <http://download.savannah.gnu.org/releases/quagga/>.
- [14] Quagga ospf interface. <http://www.nongnu.org/quagga/docs/docs-multi/OSPF-interface.html>.
- [15] Quagga ospf router. <http://www.nongnu.org/quagga/docs/docs-multi/OSPF-router.html>.
- [16] Quagga the configure script and its options. <http://www.nongnu.org/quagga/docs/docs-multi/The-Configure-script-and-its-options.html>.
- [17] Rfc a border gateway protocol 4. <https://tools.ietf.org/html/rfc4271>.
- [18] Rfc ospf version 2. <http://www.rfc-base.org/txt/rfc-2328.txt>.
- [19] Ixia. Measuring network convergence time.
- [20] K. W. R. James F. Kurose. *Computer Networking A Top-Down Approach*. Pearson, sixth edition, 2015.
- [21] A. V. Mohammad Al-Fares, Alexander Loukissas. A scalable, commodity data center network architecture.
- [22] E. C. Molero. Improving load-balancing decisions in data center networks using software-defined networking. Master's thesis, ETH Zürich, 2016.
- [23] T. M. J. P. D. W. Ryan Beckett, Ratul Mahajan. Don't mind the gap: Bridging network-wide objectives and device-level configurations. -, 2016.