Technische Universität München

Department of Informatics

Master's Thesis in Informatics

Informed Route Selection Strategies for Multipath Routing

Benedikt Engeser

Technische Universität München

Department of Informatics

Master's Thesis in Informatics

Informed Route Selection Strategies for Multipath Routing

Informationsbasierte Pfadselektionsstrategien in Multipfadnetzwerken

Author	Benedikt Engeser
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Dr. Heiko Niedermayer and Sree Harsha Totakura, M. Sc
Date	September 12, 2016

Informatik VIII
Chair for Network Architectures and Services

I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, September 12, 2016

Signature

Abstract

This thesis introduces Informed Route Selection. The technique enables end systems with a full network view to select routes based on information paths through an underlying information graph. The work elaborates this on the example of improving resilience in the internet using an overlay network, that is aware of the underlying Internet structure. It explaines the necessity and the concept of Informed Route Selection, develops a prototype and evaluates the performance.

Zusammenfassung

wird not übersetzt... wenn es passt im Englischen!!

do what stands the-
re in german

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis introduces Informed Route Selection as an alternative take on Internet routing. The approach will take a user's perspective on how to optimize paths through the Internet in the presence of an absolute network view. The thesis does not suggest to put the control over routing paths in the hands of an ordinary user. But should rather be seen as an experiment in thought and software, on how, for instance, a large content provider or a company network could improve resilience by using different router locations and an dynamic, underlay aware routing algorithm to increase its service quality.
As we explain from section 2.1, the Internet in its evolved and current structure is at the same time a communication system of unique possibilities and size and a patchwork system of tens of thousands of subsystems, that mostly try to run on the fine line between a minimized contribution and a maximized profit.
In this course we explain how these systems collaborate to create a routing graph, that reflects these objectives and how this can lead to problems caused by human and by technical failure. This leads to the main concern of this work: Resilience. We elaborate different views on this in section 2.2 and discuss the problems that emerge when trying to improve it in section 2.3. This leads to the presentation of other scientific works, that already approached this topic or a related one from section 2.3.2 onwards.
In chapter 3, we elaborate the pitch and the general idea of the contribution of this work. We will therefore explain the general idea in section 3.1 and give several reasons for it by explaining shortcommings of the BGP in section 3.2.1 and different requirements on paths in section 3.2.2 and the following.
From section 3.3 on, we address the topic of information quality, which is particularly important as also explained. We do this on the example of link exploration, using traceroute traces. In section 3.5, we explain how the optimization of routes for the profit of ac subsystem affects the overall structure of the routing path of the Internet and the conflict with endpoint interests.
In chapter 4 we develop an algorithm for IRS and eventually explain design and implementation of a prototype for IRS routing over the Internet. To achieve that goal, we

start to define an optimization goal for paths. After that, there is section 4.2 in which we introduce a information model that is used to model data in a structure for the prototype. Afterwards, we elaborate the prototype's algorithm in section 4.4. The second part of the chapter will start in section4.5 and address the design of the software.
The last part of the thesis will be the evaluation of IRS in chapter 5. For this, we will test IRS on generated internet-like topologies to experiment with different graph setups and finally deploy the prototype in the Planetlab infrastructure to compare the performance in the Internet with the laboratory conditions of generated graphs.

Chapter 2

Background and Related Work

In this chapter we provide background information about topics, which influenced this work. For that purpose we first look at the structure of the Internet and resilience mechanisms built in it. We will then discuss various attempts to influence the resilience and close with a quick review of theoretical information necessary for a better understanding of this work.

2.1 A Network of Autonomous Systems

The Internet is a collection of individual networks, build, operated and maintained by organizations of different size and orientation. The most prominent types might be Internet Service Providers (ISP), data center operators, educational and scientific networks, content distribution networks and internet exchanges. Those organization networks are referred to as Autonomous Systems (AS). The name already implies that organization operating the AS works as an autonomous sub unit of the Internet and is only accessible using gateways. The communication protocol common to all those networks is the Internet Protocol (IP). An AS is the home of one or more IP prefixes, which means that a certain range of IP addresses are associated and operated by this network. The ASes are free to use any underlying technique to transport IP data, residing on layer three of the OSI model, from one gateway to the other, in case of a pure traversal, or to a endpoint within the AS.
On this layer three however, the ASes have to agree upon a protocol to make IP transport possible. One of the most important of this protocol is to find a route to the destination host for each packet. Since there are - at the time of this writing - about 76.000 ASes, manual configuration is not feasible. At this point Exterior Gateway Protocols like the Border Gateway Protocol (BGP) are used. Those protocols however work on the basis of information provided by their respective AS administrator and are therefore, in most cases, selected to fit the economical or technical needs of the AS operator, as mentioned

later in section 3.5. AS peering, the agreement between two ASes to exchange traffic mutually between them or in another form connect in a customer-provider relationship for transit traffic, has many more facets, that are not covered here.
This structure of the Internet is both it's strength and weakness. It is a strength, that there are almost always several different theoretical paths from one endpoint to the other. If one AS suddenly stops to transport traffic, there is another path to connect the two sites. The nature of the Exterior Gateway Protocols (EGP) makes sure, that the routing tables will be adjusted in the next iteration and the outage will only affect a small part of the Internet and only for a limited period of time. A comment on the Border Gateway Protocol, the de-facto standard for EGPs can be found in section 3.2.1.
On the other side, it is a weakness that this system is highly trust dependent. Each AS informs its neighbors about the destinations reachable through it. If one AS has several neighbors who offer the same destination, it usually takes the cheaper[1] one. However, this means an AS sending out wrong information - by accident or on purpose - is able to redirect traffic into its own network. As a prominent examples, this happened on the 8th of April in 2010 when 15% of all Internet traffic was routed through China for 18 Minutes and further back Pakistan Telecom (AS 17557) began advertising a network assigned to the video streaming service YouTube in response to a government order on 22.02.2008 to block a video. In total it is estimated that almost 50% of the earths population is connected through the Internet [2]. This should make it obvious, that improving the robustness is desirable, not only for economic reasons, but to bring people closer together.

2.2 Resilience in Networks

The Internet is not only a network for exchange of scientific data, but also a platform for a huge number of small to extremely large commercial applications and infrastructure for telephony, e-health and military applications. Depending on a third party to fix a broken routing scenario can therefore be an inconvenience for one party and a major financial loss for the other. For such an influential element of modern life and business, there is a need for resilience and robustness.
Resilience however is not easy to achieve. Threats to connectivity are not only given by the structure of the Internet but much more also by environmental circumstances, for example natural disasters or human cause like when an excavator destroys a cable. Also, as shown in [3], it is hardly possible to exactly know where a the course of a packet actually runs. This means, events, no operator could foresee, could influence packet flows. In addition to all of this there are problems that come with the regular use of the Internet like congestion induced by traffic peaks, either because of local patterns (entertainment like video streaming after work) or malicious traffic caused by

[1]Here "cost" can be path length or it can be an actual price on the traffic sent through an AS

amplification/network exhaustion attacks.

2.2.1 Resilience and Reliability

While resilience and reliability are often used as if they were synonyms, it is worth noting, that they do refer to different properties of a system: Reliability etymologically comes from something that can be fastened, something that you can trust is available at any time. Resilience comes from latin *resilire*, which means to rebound or spring back. While the two words are obviously related, this observation makes it possible to understand the different intensions. The resilience property of a system therefore describes the ability to fully recover after a (partial) system breakdown.

2.2.2 ResiliNet [1]

Facing many potential threads, J. P. Sterbenz et al. founded the ResiliNet Initiative to "to understand and progress the state of resilience and survivability in computer networks, including the Global Internet" [1].
Their definition of resilience divides into further subtopics:

- **Challenge Tolerance**
  - **Survivability** is the ability to function properly in the presence of threats
  - **Disruption Tolerance** is the ability to work in the presence of connectivity disruption
  - **Traffic Tolerance** is the ability to handle unforeseen traffic peaks caused by a flash crowd or an attack
- **Trustworthyness**
  - **Security** is the protection against unauthorized access and the fulfillment of common security goals
  - **Dependability** includes availability and reliability of a system and therefore has overlapping topics with **Security**
  - **Performability** is the property to deliver the performance required by the specification

Since this work is considering connections through the Internet, and how to counter problems regarding the routing protocols, it will mostly be located in the fields that fall under challenge tolerance.

2.2.3 Resilience in other contexts

The National Infrastructure Advisory Council of the USA defined in its final report in 2009 [4]:

> **Infrastructure resilience** is the ability to reduce the magnitude and/or duration of disruptive events. The effectiveness of a resilient infrastructure or enterprise depends upon its ability to anticipate, absorb, adapt to, and/or rapidly recover from a potentially disruptive event.

2.3 Improving Resilience

To improve the resilience property in networks there are measures like: redundant cable connections, forward error correction, load balancing and failover lines. But on the layer of resilient end-to-end IP connections, the structure of the Internet introduces a problem: AS coordination mechanisms like the use of Exterior Gateway Protocols makes sure, that usually there is a route from every AS to a packet's destination. These routes however are not optimized for the endpoint's needs but for those of the participating AS operators. In particular there is **exactly one** path negotiated. In cases like those described at the end of section 2.1, but also in cases of infrastructure breakdown in an intermediate AS, there is nothing a normal endpoint can do to correct this path except for waiting for the re-negotiation of a working route. The IP simply does not consider alternative routing. Hence, an effective resilience solution for the Internet needs some influence on the routing behavior of the systems between packet source and destination.
In this section, we first broach a problem that so-called middleboxed bring into the Internet. We then introduce different resilience solutions from the past to create a picture of the research conducted in that area. Common to all resilience solutions is of course, that there is no recovery for a single point of failure as often seen in endpoint Internet connections. For this kind of problem, multihoming systems can be used.

2.3.1 Middleboxes

When talking about resilience solutions it is important to acknowledge, that the straight forward development and deployment of a path aware network protocol is not easily possible. This has to do with the realization of networks. As described in [5], there are a lot of specialized applications, called middleboxes, that offer ”valuable benefits“ on the one hand, but influence packets in certain ways. Such middleboxes are for example firewalls, that treat packets based on predefined rules. If a packet has an unknown protocol and therefore doesn't match any rule, it might be dropped. This violates the end-to-end principle that belongs to the architectural principals of the internet [6]. Other middleboxes can be VPN Concentrators, NATs, Security appliances, content engines, ssl terminators or similar appliances. [5] also states that there are almost as many middleboxes as IP routers. An effective protocol rollout would need to be supported by many of those devices.

2.3.2 Resilient Overlay Networks

Resilient Overlay Networks (RON) is one of the earlier prominent works. David Anderson describes in [7] how he uses an overlay network to implement its own routing environment. His system, similar to the Internet, uses single hop decisions based on the destination address of a packet. But additionally a tag is introduced, which allows the so-called *entry node* to preselect a route under some constraints. This route however, can be changed by intermediate nodes in case of link failures. Link failures and the overall network state is aggressively probed over the whole time. Anderson explains, that the design decision to limit the size of RONs make this approach possible.

2.3.3 Resilient Routing Layers and Multiple Routing Configurations

Even though it is not easily possible to roll out this method over the Internet, it is worth mentioning, what Audun Fosselie Hansen et al. and Amund Kvalbein et al. describe in [8] [9]. The idea of resilient routing layers is to generate several routing topologies in a special way. If the failure of a node does not disconnect the network, it is considered *safe* regarding this particular node. When a node fails, the network can easily use a routing topology, safe for this node. This methodology used for recovery of whole networks after single failures. In that scenario one administrative unit has full control over all routers. In [9] the authors introduce a similar technique for Interior Gateway Protocols (IGP) to shorten the recovery time after failures from both inside and outside[2] the AS by having precomputed alternative routing layers and avoiding the need for an IGP to converge.

2.3.4 Multi-Topology Routing

In 2005, Menth et al. [10] developed a system called Multi-Topology Routing. The idea is that every node computes shortest paths to all other nodes in different routing topologies, creating those routing topologies from the real network topology by leaving out edges. Packets sent through the network are tagged with the corresponding *MT ID*. That way, if there is a link failure detected, a *MT ID* which leaves the failed edge out can be selected and there is no need for a costly recalculation of paths.

[2]When changes require traffic to be sent through another gateway router

2.3.5 Path Splicing

In [11] the authors Murtaza et al. introduce a system called path splicing. Its main idea is summoned as follows: Run several instances of a routing protocol (such as BGP) with different configurations. Those configurations should lead to as edge-disjoint topologies as possible. The origin of the traffic can then add some information to switch between topologies at certain routers. In [11] they give a method as example to how to implement this with just a few bits. This method gives a lot of freedom to the end system.

2.3.6 One Hop Source Routing

Gummadi et al. pursued another approach in [12]. They put it into the responsibility of a packets source to avoid failed links. The source can, after detecting a transport failure, chose a set of detour nodes. Using those nodes it tries to avoid the broken link. They claim, that a random selection of detour nodes brought almost maximum benefit in their test scenario using 67 PlanetLab nodes in 2004. The approach is especially interesting for this work, since the detour decision is at the source node.

2.3.7 Path Diversity via Routing Deflections

In [13] Yang et al. worked out a system, that introduces precomputed paths as a result of deflecting traffic to a router other than the one providing the shortest path. They give three rules on how to select the deflection routes and prove liveness and safeness. In their model the end systems do not need to know the full topology. Additionally, the system is deployable one node at a time since the authors claim it is compatible with shortest path routing.

2.3.8 Path Diversification

In [14] [15] Rohrer et al. introduce a method to improve resilience and moreover define a set of metrics for the *diversification* of paths. The diversification metrics are embedded into a process. These metrics are especially important for this thesis, since they help defining the goal of resilience and can be applied to measure one aspect this property.

2.4 Graphs and Node Disjoint Paths

2.4.1 Hyperbolic Graphs

In [16], the authors show, that if a network has a metric structure, which means, that there is a defined distance between all pairs of members in it, and a heterogeneous degree distribution, which the Internet both has, this network has a hyperbolic geometry. They developed a generator for this kind of graphs that will be used in the analysis section of this thesis.

2.4.2 Survivability in Communication Networks and Disjoint Paths

In [17], general objectives for the survivability, and therefore part of resilience, are discussed. [17] and [14] discusses the necessity to have node- **and** edge-disjoint paths. The overview in [18] compares various disjoint path problems and helps to understand the properties of each problem, which lead to the conclusion that the Maximum Disjoint Path problem (NP hard) is the best choice to represent the needs for resilience in real Internet application.

2.5 Structure of and Information about the Internet Topology

When selecting paths based on information, it is of special relevance to understand the quality of that information, used in the process. Especially as there are understandable security concerns that make AS operators hide the real network structure from the public. On the other hand, there is a necessary curiosity to understand the real structure. This may help detect abnormal network behavior or develop new network concepts.

2.5.1 Invisible Hops

In [19], Sommers et al. discuss Multi Protocol Layer Switching (MPLS), a technique used in the underlying structure of ASes. This technique can lead to a state, where an IP hop displayed by a traceroute trace leads over an unknown number of other hops. This happens when MLPS is used in, what is called, *pipe mode*, as opposed to the more insightful *uniform mode*. In combination with [3], which analyses the long haul fiber optic network in the USA this is particularly interesting. The combination suggests that a packet could traverse the atlantic ocean several times in between two IP hops.

2.5.2 Peer to Peer Tracing

In the work [20] of Chenet al., they started a large-scale analysis of traceroute traces, using the plugin Ono for the popular Azure BitTorrent client. That way, they were able to collect traces from over 992000 IP addresses in over 40000 routable prefixes. That way, they claim to have found almost 24000 new links which contained 26 AS numbers, that were not available through the public BGP data. The paper shows, that there can be huge differences between the *publicly advertised* structure and the real, measurable structure of the Internet.

Chapter 3

Informed Route Selection

In the previous chapter, we sketched an overview on the background of this thesis, including an abstract perspective on the Internet as well as the related work in the field of resilience, the ways to improve it, some graph theory and possible pitfalls while gathering information.
In this chapter we will first explain what *Informed Route Selection (IRS)* is in section 3.1 and the motivation for it in more detail in section 3.2. Afterwards there will be an explanation which kind of information can and should be considered for routing in section 3.3. At the end of the chapter, from section 3.5 onwards, there will be an outlook on the implications IRS has in different surroundings.

3.1 IRS in Networks

A route in the Internet can be seen as a path in a graph, representing its topology. Communication endpoints like source and endpoint nodes, communicate through the network over other nodes, called hops. Neighboring nodes are connected over links. Route selection is therefore the process of selecting a sequence of nodes, each two succeeding nodes connected by a link, where the first node is the source and the last is the destination.
IRS is the process of selecting a routes while taking into account the information collected from the network and other indirect sources.

3.2 Why IRS?

In section 2.1 we explained the structure of autonomous systems and how they are organized in the modern Internet. In the following sections we will illustrate further, how this raises the necessity of advanced routing schemes. Likewise, we will explain the coherence with multipath networks and how the application of IRS can lead to improved end to end resilience.

3.2.1 BGP and Convergence

The Border Gateway Protocol[1] (BGP) is the de-facto standard for inter AS route negotiation. It belongs to the Exterior Gateway Protocol category and experience shows that it works well. BGP implementations are capable to negotiate everything an AS operator can expect from a routing software. It scales with even in times of increasingly bigger, more complex topologies [21] and gets steadily improved to fit the modern Internet's needs.
In case of link failures, however, a route re-negotiation is necessary, that involves a converging process. Link failures are detected after one BGP partner does not receive a Keep Alive message during the timeout of the Hold Timer, normally in the order of 90 seconds, as ([22, chapter 10]) suggests. The convergence process has widely been acknowledged to be slow and target of a lot of analyses as seen in [23], [24] and others. But, as many of them state: the solution to this problem seems far away since it would require a large portion of the AS operators to act; and, depending on the proposed solution, require them to reveal more information about the inner structure of their network.
BGP finds one *optimal* path from one AS to another. What the term optimal means, will be discussed in section 3.5. For now it is relevant, that at any given time, there are zero or one established and working routes from any source to any destination in the Internet. And if this route is failing, through any circumstance, there is no other route left, until the BGP detects it and converges again.

3.2.2 Avoiding ASes

Related to the property that there is only one working path at a time, but more general, is the question of whether it is possible to avoid certain ASes. This could be the requirement in a range of use cases: A company might want its traffic to not leave ASes which are bound to EU legislative. Some networks might be suspected to be not trustworthy in regards to traffic delivery, which makes it attractive to avoid them. Partners in a secret negotiation might know the value of meta- or connection data ([25]) and want to avoid a spy. In such cases, the method IRS, by selecting a route around such ASes, and by selecting a route according to the individual requirements enables a way to satisfy the restrictions and still communication over the Internet. When only the BGP routes can be used, there is no way for an endpoint to avoid an AS by choice.

[1]in the context of this work we will be talking about the EBGP, which stands for External Border Gateway Protocol

3.2.3 Multipath Networking

One other prominent application that suffers from only having one working path at a time, is multipath networking.
Multipath networking, in general, means that within the same communication, the packets from source to destination take different paths through the network. This can be attractive for various sub applications such as resource pooling and resilience through fallback paths or Forward Error Correction (FEC). It is understandable that, if there is only one selectable path, there can not be real multipath networking. IRS however can not only make multipath possible, but also select routes for the particular cases. Resource pooling might want to split the traffic just around a bottleneck node and enjoy the perks of an optimal route over the rest of the communication route, resilience applications might want a path, that is as edge disjoint and as node disjoint as possible ([14]). Moreover, as noted in [1], there might be some requirement that forbids to route through too much of a detour.

3.2.3.1 Fallback Paths

One way to achieve a better resilience through multipath networking is to utilize only one path to send data from source to destination until there is suspicious behavior. In case of an impending timeout or warning signs like the triggering of the fast retransmission mechanism of TCP [26], one of the other established paths is already there as a fallback. This avoids overhead in both time and data that go along with a reconnect procedure. For such an application however, certain things have to be considered, especially the question which part of such a protocol detects suspicious behavior.

3.2.3.2 Forward Error Correction

Forward Error Correction (FEC) is a mechanism that can correct transmission errors without retransmission of data. FEC encoded data therefore carries some redundancy that makes it possible to verify the integrity of data or, if that fails, recover the original data. FEC can be used inside one packet and therefore detect wrong bits and correct them. In a multipath scenario another possibility is, to send packets over one path that have correction information about packets on another path. A simple case would be just cloning a packet, sending it down both paths and if one of them fails, there is still the other path through which the packet is delivered.
In an ideal scenario with FEC, link failure can be handled unnoticed by higher network layers. For this to be possible, it is necessary, that the failing entity is not part of more than one of the selected paths. The multipath system can then, after detecting the failure in a path, replace it by another. To raise the likelihood of this scenario we will introduce a metric called *node diversity* in section 4.1 and take it into account when using IRS.

3.2.3.3 Pooling of Resources

Resource pooling can be a good usage of multipath networks. Use cases can be of different nature. Either there can be several connections with the same properties, to be combined, or there are different connections that can be used different kind of applications. An example for the first case would be using two connections to have a combined bandwidth for high throughput applications and for the second case, there would be the combination of a low latency but low throughput connection and a high throughput but high latency, in such a way, that time-sensitive applications take one connection and high bandwidth applications use the other.

3.2.4 Requirement Types

From the previous sections, we extract two types of requirement types:

- **Boolean requirements**, that are either fulfilled or not, like the avoidance of an AS. We sometimes call those requirements constraints.
- **Metric requirements**, that can be rated and optimized, like, as we'll describe in section 4.1, the diversity of paths

3.3 Information Quality

Garbage in, garbage out, the principle, that the output of any computation can only be as good as the input, is as true in IRS as in other algorithms. Therefore, when selecting routes, it is important to carefully inspect the information we use. While a careful selection of information is of course always advisable, for the scope of this work, we focused on traceroute trace data. We decided so, because traceroute data is a well researched field and example information is easy to gather. In the following sections, we will explain how a trace can be gathered and what causes traces to be distorted.

3.3.1 Path Tracing

does this say "tracer- utes is the easiest ossible way for to do it?"

Getting detailed hop by hop information about a path can be done by tracing it. We found, there is currentlyno way to get up-to-date route information between two nodes in the Internet, because keeping a database up to date would take a tremendous effort since it would require constant, extensive probing or the cooperation of many AS operators. This section is about the technique used to trace a path.
IP is a protocol to deliver packets from one host to another. For that purpose every packet that is sent through the Internet, is prefixed with a header. This header has in both of the current versions, IPv4 and IPv6, a field that defines a maximum number of

hops on a path to the destination, which is necessary to detect stale packets and prevent them from circling forever through the network and cause congestion. It is named Time To Live[2](TTL) in IPv4 and Hop Limit in IPv6. In the scope of this work, we will refer to it as TTL.
The packet's source initially sets the field to a value[3] and every routing node on the path decreases it by one. When the value reaches zero, the packet is discarded and a notification in form of an Internet Control Messaging Protocol (ICMP) message is sent to the packet's source.
This error mechanism can be exploited to trace a packet's path. For that purpose, a test program, for instance the Linux command line tool traceroute, sends a packet with a TTL value 1 to the destination. In order to properly identify the response later, in IPv6 an identifier like a random number should be the content of the first 64 bits of payload. The first router will decrease the TTL field and notice it is zero. As a reaction it will send an ICMP packet to the source, including header and the first 64 bits of the discarded packet. The test program will recognize the packet and associate the first hop with the source address of the ICMP message. This process is repeated several times with an increasing TTL value until the source of the answer packet matches the destination address. This last message however is no TTL exceeded message but rather, an appropriate response for the sent packet. This is the case because the last router will not send a packet with TTL equal to zero to the destination and the destination will simply accept a packet with a TTL greater than zero and not decrease it first.
Traces can be done in several variations: the ICMP has a subtype called traceroute, that will make the destination respond with an ICMP packet. When using UDP or TCP packets, the responses will be according to the protocol specifications. This can make it harder to get proper responses. They are however useful to exploit established rules in stateful middleboxes.

3.3.2 Understanding AS Infrastructures

One problem of interpreting information about the topology of the Internet is that the AS operators tend to keep their router level topology in confidence [28]. This makes it more difficult to interpret the data gathered in traceroute traces. The problem with that can be seen when considering one routing site with two routers, that share the burden of routing of traffic. Two traces, gathered using the method described in section 3.3.1, that are both routed through this site, might not have common node in this hop, if the TTL exceeded messages originate each from another router. If the traces take different routes after this site, this could create a false sense of different routes. Because of the same location and same environment, those routers might be subject to the same problem,

[2]According to [27, page 14], in IPv4 this was actually meant to be a time in seconds, that had to be reduced by at least one per hop
[3]common initial values, depending on the implementation are 48 and 64

like power outage, a link break, misconfiguration or similar problems. The path traces are hence not as independent as one could think. A proper understanding of those topologies could therefore enhance the traceroute data a lot.

3.3.3 Multi Protocol Label Switching

Multi Protocol Label Switching (MPLS) is a technique, which makes it possible to handle AS internal routing particularly efficient. It works as described in [29] by prefixing a data packet with a label that is associated to a precomputed path through the network. A labeled packet can then be routed by a Label Switching Router (LSR) comparing the label instead of comparing the prefix of an IP address. This technique is especially useful for traffic, which has to be transported to a gateway router. The MPLS-network ingress node, that comes in first contact with the data, can there match the IP address prefix with the BGP routing tables, determine the next AS and add the label of the gateway router that is connected to that AS. After that, Label Switching is used to keep the resources necessary for AS internal routing as low as possible.
In [19] [30], the authors describe two major modes of MPLS; In the *Uniform Mode* and the *Pipe Mode*. The significant difference for this thesis is, that the handling of the TTL field is different between them. The Uniform Mode considers the TTL field, and the egress node, that is the last node of the MPLS network that handles the IP packet, decreases the TTL field by the number of LSRs it has traversed. Pipe Mode only decreases the TTL field of the IP header by one for the whole traversal.
This influences the reliability of trace information, since it violates the assumption that every hop is decreasing the TTL field. As a result, two traces that traverse the same MPLS network in Pipe Mode can be routed by the same LSRs, but because they use different ingress and egress routers, which are the only visible routers in this mode, seem to have no common hops.

3.3.4 Missing Information

One significant influence on the quality of gathered information is given by timeouts. In case of a timeout there is no information about a hop on the path. Incomplete information has a negative influence on the route selection result. We will discuss a method to handle these cases in chapter 4.

3.4 Impact on the Internet

Qiu et al. elaborate in [31] the impact of "selfish traffic", that means traffic that uses the network in a way ideal for the needs of itself, on network topologies. They state, that while selfish traffic can achieve near optimal results in regard to connection metrics like bandwidth and Round Trip Time (RTT), the excessive usage of popular links can lead to negative results for all participants. This is an instance of the Tragedy of the Commons model. It states, that a limited resource, freely available for individuals, is used inefficiently with respect to the common good, to a degree where it harms the individual use. This idea is also applicable to IRS. For the scope of this work, we deliberately choose to neglect it.

3.5 Traffic Engineering

Traffic engineering is the process of optimizing traffic flow concerning a metric. Inter-AS traffic engineering is focused to move traffic as fast as possible to a certain point. This is either a target node inside the AS or it is a gateway node to route traffic to a neighboring AS. The optimization metrics are in first instance of technical nature: less traffic in the system means less load on the system. This is however driven by the economic requirement to invest less in hardware. Techniques like MPLS, which is described in section 3.3.3, are used in this process.
For the matters of IRS it is important, as already broached in section 3.2.1, to understand which optimization efforts lead to BGP routes. Therefore we distinguish between different kind of BGP inter-AS-partnerships. These different kinds emerge from the variable size and importance of ASes.

- Equal ASes can arrange a *peering treaty*. It is important to note, that peering is a fixed expression, where peer, in the meaning of equal-ranked, refers to the rank of the ASes. Peering treatys are usually of mutual benefit for both partners. Sometimes peering partners share specific transit traffic.

- ASes of different size and importance usually connect in a customer-provider-relationship. This means, that the lower ranked customer-AS pays for the traffic with the provider-AS.

A provider-AS is a Transit AS, since the traffic of a customer transits through it. Non-transit ASes can be Stub ASes if they only have one provider or Multi-homed ASes, if they have more than one provider.
A Stub AS must route all traffic through its provider. A gigantic AS like AT&T has only customers and peering partners, hence it will not consider transit costs. But all other AS-constellations in between those two extremes will try to find the cheapest way to get rid of traffic with a destination in another AS.

- When paths cost the same, the shortest will be chosen

- A customer that announces a path is attractive, since traffic with customers brings money

- A peering partner, that announces a path is always peferable to a provider, since peering traffic is cost-neutral
- A Multi-homed AS will send traffic to the cheapest provider
- A Multi-homed AS will announce its prefixes to its providers to be reachable, since it will try to discourage paths through more expensive providers (section 3.5.0.1)
- An AS will not announce a route that costs more than it brings profit

3.5.0.1 Path Prepending

There are circumstances where an AS wants to discourage a certain route. For instance when an AS wants to make sure, the most attractive path to reach its prefixes is through the transit AS of the cheapest provider. An AS will then try to make a path unattractive by making it longer, but still keep it as a backup path to stay reachable in case of a problem in the cheap provider's AS. Length is given by the number of ASes in the path. It can be inflated by adding the own AS number several times. This technique is called path prepending.

3.6 Effect of Source Routing

In contrast to the techniques of ASes to optimize routes according to their own economic advantage, as presented in section 3.5, there are also the desires of Internet users. These two views of an optimal routing do often not overlap. In the following section we will explain source routing, compare the goals presented in section 3.2 and section 3.5 and see how the application of source routing influences the efforts of traffic engineering.

3.6.1 Source Routing

In current IP networks, Hop-by-Hop routing is the standard[4]. This means, that each router analyzes a packets destination address, selects the next router according to that, and forwards the packet to that router. On the contrary, Source Routing (SR) means, that the source of the packet selects a route, or important interims hops and the network forwards the packet according to those instructions.
SR is not possible in the Internet as it is right now: It requires routers, positioned on different positions in the network, which are able and willing to read the source instructions path and forward the packet according to that information. It has, however,

[4]Exceptions to that are for example MPLS routed segments

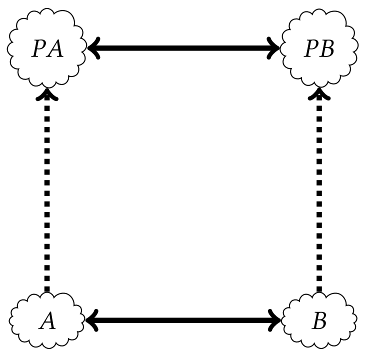

Figure 3.1: Example configuration that could lead to undesired routes

been done in several experimental setups ([7], [32], [12]) using software deployed on server machines in as many different ASes as possible.

3.6.2 Unused paths in BGP routing

As understandable after reading section 3.5, there are links in the Internet, that are not usable because of policy driven routing decisions. In a scenario as seen in figure 3.1, where a cloud represents an AS, a solid arrow shows a peering relationship and a dotted one points from customer to provider.
A will not announce its relationship with *PA*, to *B*. *A* will however announce its own prefixes towards *B*. The path from *B* to *PA* will hence lead be $B \rightarrow PB \rightarrow PA$. The path $B \rightarrow A \rightarrow PA$ exists, but is not wanted by A and therefore not announced. Using IRS, the combination of both paths would be great for resilience by multipath. Also, assuming *PB* is an extremely large network, routing though *A* might be of advantage if the path is shorter.

3.7 Comparing Ideal and Actual Situation

In section 3.2 and section 3.5, we explained the motives for route selection from two different perspectives: AS operators and Internet users. In the following table 3.1, we will confront the real situation with the desired situation.

	Ideally	BGP
number of paths	many diverse	One
path selection	case specific	optimized for AS policies
avoiding ASes	configurable	not possible
link failure recovery	fast or in background	slow convergence after detection

Table 3.1: Comparison of ideal and actual routing situation

Chapter 4

Realization

In the previous chapter, we described the problems of the current Internet routing system from a user's perspective, what Informed Route Selection is and how it can help to overcome those problems. In the following sections, we will explain an approach to realize IRS. First, we will introduce some important metrics which will be important to rate selected routes. Then, we will present an information model and an algorithm to select routes. At the end of the chapter we will explain the design and implementation of an application to experiment with IRS and evaluate the algorithm.

4.1 Conceptualities and Metrics

Before starting to describe the procedure of selecting nodes, one important part is to set out a goal for the algorithm, which we do by defining metrics that will be used to rate the quality of routes. The route selection can then be optimized in regard to these.

Fundamental Terminology

Information flowing through a network follows a certain route. This route can be represented by a path, that traverses all routers, links and obstacles which influence the information. The goal of IRS is a well selected path. A **Path (P)** is a sequence of nodes starting with the Source (S) and ending with the Destination (D). Every item of the sequence is a step.

$$P_{SD} = (S, h_1, h_2 \ldots D)$$

While selecting a path, Source and Destination node are fixed. everything else is, a certain variety assumed, selectable. One Way to rate the character of a path is to look at the set of nodes that make it up, without particular order or counting nodes twice.

Therefore **Node set (N)** a set of nodes of a path without the endpoints S and D. Or more formal:

$$N_{SD} = \{x \mid x \in P_{SD}\} \setminus S, D$$

If S and D are clear from the context, we can simplify by omitting the indexed SD, as in P and N

Path Length

The length is an interesting feature of a path. The longer a path is, the more elements can influence its emerging properties. The whole depends on parts to function properly. Therefore, assuming all nodes have the same reliability, a path twice as long has a twice as high risk to fail. Also especially in networking, a longer path can lead to a higher propagation delay and other unwanted side effects.We use the notation $|P|$ for the length of a path P meaning the length of the sequence that defines P.

Node Diversity

In [15] Rohrer et al. develop a metric to measure path diversity. They argue that it has to consider both nodes and edges of a graph. This is correct, following the explanation in [15, Section III, B]. Based on the graph model explained in section 4.2, this is not necessary, since every Point Of Failure(POF) is modeled as a node and a failing edge is therefore modeled differently than in their case.
When we try to achieve resilience through multipath, it is important to have several paths being as independent as possible with respect to those POFs. If two paths do share a certain POF and it fails, then both paths also stop working and the goal is missed. The metric from [15] takes the number of nodes that are in both paths and puts it in relation with the number of nodes in the shorter path. A pair of ideally diverse nodes does not have a single common node on the path, which means that a single failure in the network can only affect one path. Therefore we define, based on Rohrer et al. the metric Node Diversity (ND) as:

$$ND(N_a, N_b) = 1 - \frac{|N_b \cap N_a|}{|N_a|} \text{ where } |N_a| < |N_b| \text{ and } N_x \neq \emptyset$$

This results in 1 if the N_a and N_b are node disjoint and in 0 if both sets are identical.

Multi Node Diversity

The metric ND from the previous section only works for two paths. However, we wanted a metric for an arbitrary number of paths. If a set of paths is pairwise disjoint for all possible paths, it would be ideal. But much more relevant for our purpose is, if that is not the case. We therefore consider three metrics over the whole set to rate diversity: Lowest Node Diversity (LND) for the worst case in this set, Mean Node Diversity(MND) to judge how the set will generally perform and and Highest Node Diversity(HND) to see the optimal case.

Considering a set of node sets derived from paths as described in section 4.1, so that $M = \{N_1, N_2 \ldots N_m\}$, the Lowest Node Diversity metric is the lowest diversity between all pairs of sets. This will be the main metric for the evaluation.

$$LND(M) = min(\{ND(X,Y) \mid X \neq Y \wedge (X,Y) \in M \times M\})$$

accordingly for the Mean Node Diversity:

$$MND(M) = mean(\{ND(X,Y) \mid X \neq Y \wedge (X,Y) \in M \times M\})$$

and Highest Node Diversity:

$$HND(M) = max(\{ND(X,Y) \mid X \neq Y \wedge (X,Y) \in M \times M\})$$

In an optimal case, these would all return 1 which would mean, that all paths are pairwise fully disjoint. The most relevant metric is the **LND**, because it allows a good assessment of the worst case.

Minimum Node Cut Size

Using multipath can be an ineffective for improving resilience, if all paths share a single POF and it fails. It does not matter wether this is the only overlapping node of all paths or not. Node diversity gives a relative information, which can be misleading. In an extreme example with a set of long paths that all share only one intermediate node, the LND and MND can be close to ideal. If the shared node however is the same for all path pairs, it can shut down all at once. Thus we are interested in the number of POIs that had to fail to totally disconnect S and D.

We define the metric Minimum Node Cut (MNC) as the minimum number of nodes in a graph G, only made from the questioned paths, that had to be removed from G to disconnect S and D

The worst case for this metric has been explained above. The best case scenario would be, if one node from each path had to fail, to totally disconnect S and D. For a Path Set PS_{SD} this would be $|PS_{SD}|$.

Detour Factor

As shown in section 4.1, a shorter path is preferable to a longer one, if we assume that all POIs fail with the same probability. Therefore it is interesting how much longer a path has to achieve diversity. The detour factor therefore rates the length of a path compared to the length of a shorter path.

$$DF(P^a, P^b) = \frac{|P^b|}{|P^a|} - 1 \text{ with } |P^a| \leq |P^b|$$

This metric has a lower bound of 0, if the paths have the same length.

Multi Detour Factor

With the same argumentation as in the section about Multi Node Diversity, we also use the Lowest Detour Factor (LDF), Mean Detour Factor (MDF) and Highest Detour Factor (HDF).

$$LDF(M) = min(\{DF(X, Y) \mid X \neq Y \land (X, Y) \in M \times M\})$$
$$MDF(M) = mean(\{DF(X, Y) \mid X \neq Y \land (X, Y) \in M \times M\})$$
$$HDF(M) = max(\{DF(X, Y) \mid X \neq Y \land (X, Y) \in M \times M\})$$

It is of course ideal, if all of those three are zero but since the metric of DF is not bound by a maximum value, neither are those.

Metric	Range	Good Value	Bad Value
Path Length	$[2, \infty)$	2	∞
Node Diversity	$[0, 1]$	1	0
Minimum Node Diversity	$[0, 1]$	1	0
Minimum Node Cut	$[1, \|PS_{SD}\|]$	$\|PS_{SD}\|$	1
Detour Factor	$[0, \infty)$	0	∞

Table 4.1: Metrics Overview

4.2 Data Model

To evaluate information algorithmically, we have to put the information into a structure. As we already broached in section 3.1, we decided to follow the intuition to interpret the Internet as a graph with network nodes modeled as vertices and links modeled as edges. In this section we will explain, how the additional information can be interlaced into this structure.

4.2.1 Two-Layered Graph

The fundamental concept of this approach is a Two Layered Graph (TLG). One of the layers, called the Overlay Graph (Overlay, OG), contains vertices representing network participants like endpoint nodes and routers and edges representing routable links. The second layer is called Underlay Graph (Underlay, UG). Nodes in this graph represent information of different kind and the edges connect it. Every edge in the Overlay is associated with a path in the Underlay. In this work, sometimes we call edges in the Overlay "virtual edges" to symbolize the fact that they represent a whole path and not only an edge. While the Overlay can generally be considered undirected, the Underlay is preferably directed.

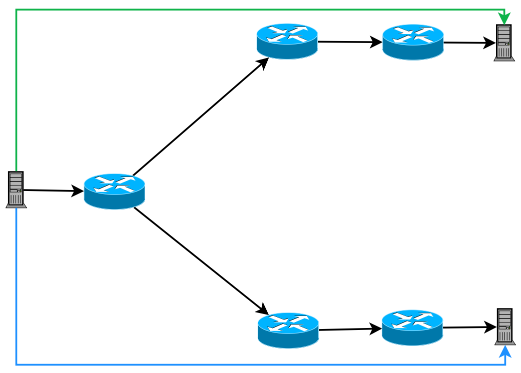

Figure 4.1: Virtual edges share underlay nodes

4.2.1.1 Overlay

The overlay graph represents the routable network. It contains every router and endpoint, which means all participants that try to select a route will find themself in that graph. Every edge is annotated with a sequence of Underlay-nodes, representing a path. If that is not possible in the Underlay, it can not be associated with a virtual edge. The graph is also weighted. The specific weight of each virtual edge depends on the path that the edge represents in the Underlay.

4.2.1.2 Underlay

The Underlay is the information graph in this structure. It means, that every node contains some information about a part of a link. Every node of the Overlay graph but not the edges are present in the Underlay, too. But not all nodes in the Underlay are selectable for routing. Even if there is a more attractive way in the Underlay: Only the complete paths, associated with an Overlay edge can be chosen as units. Underlay edges are not weighted. They are to be understood as relationships in an information network and not as really selectable paths.

4.2.2 Information Mapping

The following examples expound the way different types of information can be mapped to the Underlay without any claim to comprehensiveness.

4.2.2.1 Direct Connection

The most trivial case is of a connection cause a direct connection between two overlay nodes A and B. In that case, the edge (A, B) will be directly adopted in the Underlay. In a real network this would represent a direct cable from one host to the other.

4.2.2.2 Point of Failure

Assuming a connection from A to B and another connection from A to C. Both connections lead over the gateway router R of As home subnet. If R fails, both connections break down. This is a common point of failure for both connections. In the Underlay there would be a path $A \rightarrow R \rightarrow B$ mapped to the virtual edge (A, B) and $A \rightarrow R \rightarrow C$ to the virtual edge (A, C). Such a point of failure can be anything that is sharable and prone to failure. Explicitly, we want to point out that a fiber optic cable, shared by different colors, should also be modeled as a point of failure and therefore as a node, not only an edge. This will be the focus of a resilience model, since it can predict the impact of a component's failure on every edge of the Overlay.

4.2.2.3 Points of Interest

A Point of Interest (POI) could be anything, that is relevant for a routing algorithm. An example is a node that signals a satellite connection.

4.2.2.4 Areas

In section 3.2.2, we mentioned the idea to stay inside a certain legislative system. For this, there can be areas defined in the Underlay. Considering an edge $(n^{(1)}, n^{(2)})$ that has one node $n^{(1)}$ in one area and the other node $n^{(2)}$ in another one; a border node b can be inserted and annotated, that signals the transit to another area and the edge $(n^{(1)}, n^{(2)})$ can then be replaced by the edges $(n^{(1)}, b)$ and $(b, n^{(2)})$. This has to be done with every edge in the underlay, that has its ends in different areas. Border nodes are a special case of a node of interest.

4.2.2.5 Edge Annotations

An important convention is: there is no information in the Underlay edges, but all information in the nodes of the Underlay. If there is the need to annotate an edge, it is always possible to create a node, annotate it and put it between the edges endpoints.

4.3 Inserting Information

In this section, we will explain how information is gathered and added using the traceroute information which is already known from section 3.3.1. For that purpose we anticipate the setup of an overlay network from section 4.5. In this setup, the nodes in the OG are hosts, that run our prototype routing software. Edges are links, established between the nodes. In such a scenario, a packet, sent along the virtual edge from one Overlay node to another follows a certain path through the Internet. Every hop along this IP path is relevant for the edge. And it is possible, that one hop is used by several edges in the Overlay. Each one on this path is therefore a Point of Failure and a sensible information for the Underlay.
The first step is to gather trace information, as described in section 3.3.1. This raw data has to be slightly altered to fit our needs as described in the following sections.

4.3.0.1 Adjusting Granularity

In section 3.3.2 we explained, that ASes tend to keep information confidential. We also illustrated how a site with several routers that share the load, are likely to be affected by the same disruptive events. We also explained in section 3.3.3, how traffic engineering techniques, like MPLS can distort the information gathered using this method. But it is possible to map the public IP of a hop-router to an AS, using BGP data. MPLS is a AS internal traffic engineering method and load balancing between routers is only sensible, if the routers belong to the same AS. It seems hence reasonable to use AS-level granularity. Also, from another point of view, it is reasonable: Most of the problems

we are addressing are a result of the BGP protocol, which uses AS level routing. AS level therefore seems to be coarse enough to handle the information inaccuracy and fine enough to solve the BGP problems.
It is most likely that with this method, some consecutive hops have the same AS assigned. This can be countered by just merging all of those series into one hop. As a last step, each AS is mapped to a node in the Underlay. If there is no node, for an AS, it is created.

4.3.0.2 Handle Missing Information

As already described in section 3.3.1, there are some probe packets, that are not answered. This can be a connection issue, but more likely, it is a node, that is not willing to respond. In such a case, there is no use in trying more than a few times[1] to get the information about the host. Again as in section 4.3.0.1, a sequence of several missing hops get merged into one. Then we distinguish two cases:

- If there is the same AS before and after the missing hop: Since the structure of BGP makes only one outgoing path per destination possible, otherwise a packet had to traverse an AS twice. If it would it would leave towards the same next hop as before and circle around until the TTL exceeds. Thus, in this case the preceding, the missing and the succeeding hop can be merged into one.
- If the AS before and after this hop differ, there is no way to tell what kind of hop it is. As a consequence a node, which is identified as ”the one between the preceding and the succeeding AS“ is created and handled as a point of Failure. This is furthermore a Point of Interest, that signals uncertainty.

4.3.0.3 Association

After all nodes have been setup in the correct way, the path through the Underlay can be associated with the virtual edge in the overlay. This is done by annotating the Overlay edge with the sequence of nodes.

4.4 Algorithmic Informed Route Selection

In this section, we will explain the selection algorithm. The method modifies the graphs. An implementation should therefore work on a copy of the graph or make sure, that the initial state can be recovered after the process.
The general requirements for the path selection algorithm are: Given the TLG, find x

[1]most implementations repeat three times

paths from a S to D while avoiding nodes of interest with a certain predicate and do so while minimizing the impact of a failure.

4.4.1 Filter Nodes

The first step for IRS is to remove all Underlay nodes, matching certain criteria. The algorithm iterates trough all nodes; if a node is a POI and it matches one criterion, it is deleted. After this step, the Underlay only contains a subset of nodes, that fulfill the boolean requirements of IRS.
After this, the data model is in an inconsistent state, because the Overlay contains virtual edges with paths, which are not longer possible. This state has to be corrected by deleting all of those inconsistent Overlay edges. Afterwards, the Overlay represents a graph, that matches the boolean requirements.

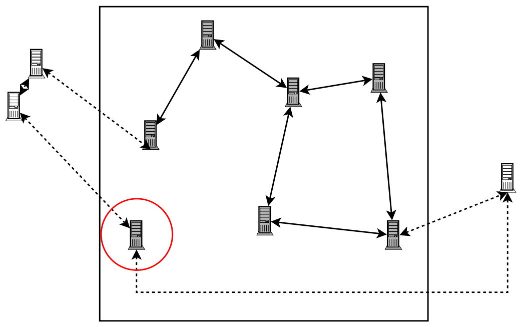
Figure 4.2: Node filtering can isolate nodes even it they do fit the filter

4.4.2 Maximum Disjoint Paths Problem

In the attempt to reduce the impact of a node failure on multipath networking, often graph theory is consulted. From that, there are several possible problem statements, that could fit different setups. From the overview in [18], the Maximally Disjoint Paths Problem seems promising, since it considers the case in which for parts of a network, no node disjoint paths are available. While this is a necessary condition, it is not a sufficient one for out needs. Modeling our problem into a Maximally Disjoint Paths Problem would minimize the impact of a failure, but:

- It scales bad with an increasing number of nodes, since this is an NP-Hard problem
- An optimal solution for the Maximally Disjoint Paths Problem might not be an optimal solution for IRS, because it may achieve a better result in node diversity at the cost of long paths.
- Also, it would seem that most algorithms that deliver an optimal solution are not built to find an additional path in case one breaks.

For these reasons, we decided to develop an heuristic, that results in a non-optimal but good enough path diversity, avoids big detours and works incremental, meaning it finds a path $P^{(x)}$ after paths in $\{P^{(y)} \mid 0 < y < x\}$ have already been found.

4.4.3 Find Shortest Paths

The heuristic works in an iterative way to incrementally create a set of paths. In this section, will refer to the usage of a shortest path algorithm. This could be any shortest path algorithm, that works on weighted directed graphs like the A* or Dijkstra's algorithm.
The first step in each iteration is to calculate the weights of the virtual edges. Afterwards the shortest path through the Overlay is selected by a shortest path algorithm. Then, each Point of Failure in the Underlay, that is associated with one of the virtual edges in the returned path, gets *burdened* by this path. The path is added to the result set and the procedure starts again in the next iteration. This is repeated until the break condition holds.

4.4.3.1 Weight Calculation

The burden on a node is an integer annotation, that states how many Overlay paths already rely on this particular node to work. By implication this is also how many paths break down in case of a failure. A simple interpretation is, that a node with a high burden is unattractive for new path. The burden is taken for the calculation of the virtual edge weight afterwards each round of the algorithm. This results in a higher edge weight for edges with a high burden and thus makes edges unattractive for an algorithm if they share POFs with already selected paths. This results in the following two steps to calculate an edge weight:
The Burden(B) function returns the burden of a node or the sum of the node-burdens of a path.

$$B(X) = \begin{cases} \text{annotated burden} & \text{if } X \text{ is an Underlay node} \\ \sum_{n \in X} B(n) & \text{if } X \text{ is a path} \end{cases}$$

The Weight (W) function takes the path of a virtual edge to calculate its weight:

$$W(P) = |P| + f(B(P))$$

where f is a mapping from burden to additional weight. In chapter evaluation two approaches will be compared:

$$f_\theta(x) = \theta * x \qquad \text{for a linear discouragement}$$

$$f(x) = 2^x \qquad \text{for an exponential discouragement}$$

The weight function W of a path has two goals: Discourage non-diverse paths and if there are two paths with the same burden, take the shorter one. This method does not distinguish between one highly burdened POF and several lower burdened POFs since the consequences of choosing one over the other were not irrefutably and would have brought another dimension into the evaluation. The parameter θ will determine how hard the reuse of a POF will be punished and be subject to the evaluation.

4.4.3.2 Break Condition

The obvious break condition, that ends the iterative process is reached if the requested number of diverse paths is found. But, since this heuristic does not analyze the how many paths are possible between two nodes, it will not be able to detect the end of the possibilites. In such a case, the obvious break condition is never met. We decided to abort the process after ten consecutive unsuccessful tries to find a new path.

4.5 Prototype of a Router Software

In the following section we describe the design and important implementation decisions of a prototype. The software is a participant of a network of nodes that experiment with IRS. The software is written in python, which allows a fast development due to a very high level API.

4.5.1 Design Overview

The software consists of three components as sketched in figure **??**. It is deployed on hosts over the internet. The software then connects to other hosts to build a network on basis of manually configured pairings. The design is plugin based, without the plugins, the system will read out a configuration file, setup the plugin environment, manage connection states and apart from that run idle. The plugins will fill the gap and bring functionality. With this approach, we achieve a high modularity to experiment with different setups. The task sharing is outlined in the following way: The central object, called "VEdges-object", is a singleton, that initiates all action and orchestrates the interaction between the other parts. The communication system takes care of network interactions of all kind. The PluginManager is responsible for the execution of plugin calls and managing the plugin library. The software is a prototype for experiments with multipath and IRS.

4.5.2 Asynchronous Approach

The prototype is event oriented, since it reacts either to a local (user) interaction or to a packet that comes over the network. This makes it especially suitable for an eventloop based architecture with asynchronous nonblocking I/O. This means, there is a loop running in one execution thread, that processes tasks. Those can be created externally, for instance in the code before the loop is started, or in other tasks during the execution. There is no scheduling that disrupts one task to schedule another during its execution, there is however the possibility to voluntarily hand the control over back to the event loop. This happens by registering to wait for another event. Those can be I/O operations, interrupts or timers. There is however no forceful control stealing. Handling critical sections in code can be easier in such an architecture, since one can make sure, data is not accessed by another method between two operations. It does however raise the importance of manual control management. Long term calculations in the background can restrict the systems ability to react to events.
Python, after version 3.5, provides the package *asyncio*, which contains an eventloop implementation.

4.5.3 Communication

The communication subsystem has two main segments. One constantly monitors the consistency of it. Because the communication system is configured by setting properties, not by using an active interface, and since other properties like the state of an connection, can be changed through external influence, it is important to iteratively examine the sanity of the system. The other subsystem reacts to external events and handles the communication tasks. These external events are either local communication requests or data from the network. This data must then be verified, classified and further processed. The different types of procedures in the system are highly entangled but separated by purpose in the following sections.
One important design decision for the communication was the selection of UDP as a link transmission protocol. This was, because we wanted to keep the option to build IP transport and TCP over IP over this network on the table. But stacking TCP connections is considered bad practice. It introduces uneccessary overhead and can lead to a TCP meltdown [33].

4.5.3.1 Link Management

To simulate a routing environment, the software maintains links to a subset of the network routers. In theory, the most fine grained routing result can be achieved by connecting to routers, that are close to each other, which in this context means a short path in the Underlay. Moreover, the less Underlay nodes are in one path, the smaller is

the chance of failure for a particular link as argued in section 4.1. In practice however, we use pre-configured links. The underlay path length depends on the BGP and is not fixed. A more dynamic system could lead to a better selection of neighboring nodes and therefore a better performance of the algorithms.
A link is identified by the address of the partner. This information is inserted into a list by the VEdges system and then iteratively revised. A link can have one of several states, which leads to certain required actions:

Disconnected means, the link is not active. Either because it has recently been added, the connection attempts failed or because the other side sent a disconnection notice. The suggested action is to try to connect to this partner.

Connecting is the state a link has, during the two way handshake. It means, a connection attempt has been started but not been answered. The suggested action is to check, wether a timeout has occurred or not. If not there is nothing to do, but wait, else a disconnect notification should be sent.

Connected the normal state, in which a link can be used so transfer data. It has to be checked, whether or not the last interaction on this link is older than a certain timeout. If so, a Keep Alive (KA) request is sent, to examine the sanity of the link. An active partner will respond to the request with a KA response. Both packets have the sole purpose to probe link sanity.

Incomming only is the one-way version of a **connected** link. The link partner is allowed to send traffic through the link, but it will not be used to forward traffic, which is received over the other links. The same sanity probe mechanism as for **connected** links should be applied.

A connection is added by simply inserting a disconnected link into the link list. A disconnect however happens by deleting a link and immediately sending a disconnect notification. After every change in a link's state, an event is triggered via callback, that notifies all plugins of a certain type, as further explained in section 4.5.4.6.

4.5.3.2 Connection Establishment

The connection procedure has two steps.The first one is to send an association request to the other host. The second one is to await the answer which is either an association confirmation or a disconnect notification. When receiving a association confirmation, the link is considered connected. The connection procedure can be triggered by two different circumstances; either the link management detects a disconnected link, as described in section 4.5.3.1, or the impulse is given by an association request from the outside. In such a case, it is a policy decision to either connect, accept or reject. Accepting means to only send an association confirmation. This corresponds to a one-way connection. Rejecting is, to send a disconnect notification. Connecting implies to

send an association confirmation and follow up with an association request. A pseudo random number is used to match a request to an answer. Although this is not sufficient for security implications, it can help to raise the trust in the genuineness of a partners IP address.
After the connection has been mutually accepted, both sides start a trace to the partner. This is done, in order to understand the underlaying hops of the link. The technique used is described in 3.3.1. We use UDP traces to the same ports that are also used in the link connections. This gives the highest probability to be routed the same way a link packet is routed.

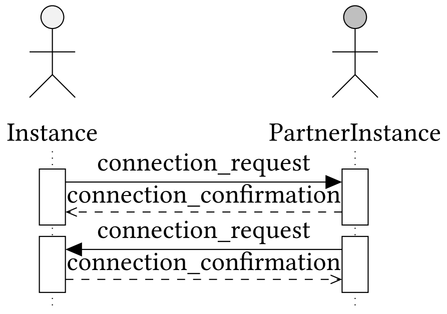

Figure 4.3: Bilateral Connection Process

4.5.3.3 Packet Structure and Types

All communication between the nodes, shares the same packet structure. We therefore choose a layered approach. Each layer has a *Type* field, a *Length* field and *data*. Type implies the interpretation of the data. Length holds the information how long the data is. Some layer types do not use the data field. There are message packets, consisting only of one layer. This is for example the KA packet. In general however, information is transported in several stacked layers. The disconnect notification for example consists of a Management layer with data length zero, followed by a management layer of type ”Disconnect“.
In python, this is implemented as a linked list, where every PacketLayer object has a reference to the next layer. During the transmission over the network, the packets are transformed into a bytearray where one layer begins in the byte succeeding the previous layer’s last byte. The length field is used to determine the boundary between two layers. This layered approach is not efficient in regard to saving data volume, but it helps to stay flexible during development. Also, one might notice, there is no source field. We decided to leave everything except the routing to the overlaying protocols, for instance IP.
As there are several different layer types, we present the most important ones without particular order:

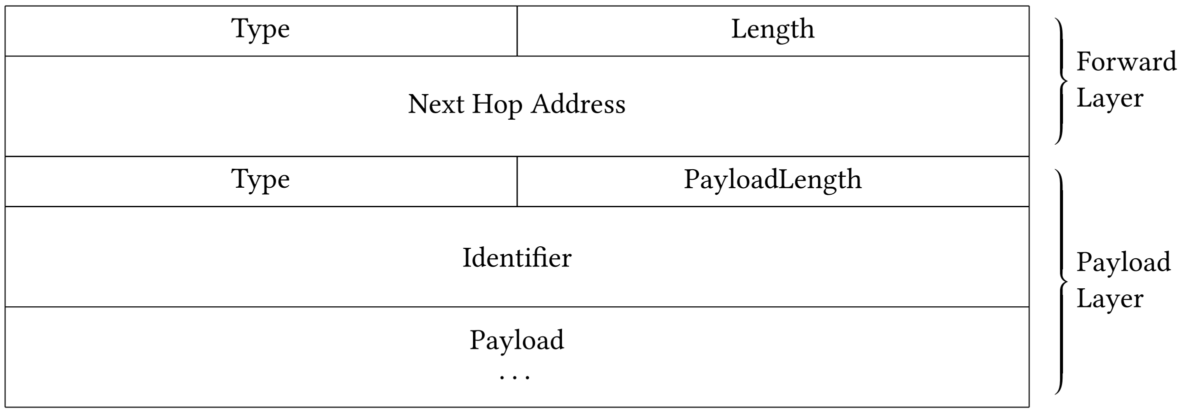

Figure 4.4: Payload Wrapped in Forward Layer

Management layers have direct connection to the link management. They are used for association requests and confirmations, disconnect notifications as well as KA requests and responses. Indications for these are explained in section 4.5.3.1 and 4.5.3.2.

Payload layers carrie application data, is generated by the overlay applications as explained in section 4.5.4.5, transported through the network and delivered to the target's overlay application.

Operation layers transport messages that are in a sense like Payload messages between plugins but for all plugin types. They are however treated differently by both, the routing system and the VEdges object, as described in section 4.5.4.3.

Forward layers hold the hop-wise routing information for a packet. The forwarding procedure is explained in section 4.5.3.7.

Message Identification—The identity field of a payload is there for error handling. Unfortunately, we did not have the time to implement a full error handling in the prototype. Yet we think, the idea should be mentioned here.

Since packets of the types Payload and Operation carry information for outside of the communication system, they are treated slightly different. This is also visible in the structure if those layers. The Payload layer has an eight byte identification field. The Operation layer has an additional *channel* field. These channels will be explained in section 4.5.4.3.

4.5.3.4 Address Translation

Because the Overlay has its own address space, which can be totally independent from the IP network this software is built upon, there must be a translation mechanism. Since routes are a sequence of Overlay addresses and the links in the communication system are IPv4 addresses, there is no way for the communication system to understand the next-hop information without a proper translation mechanism. This translation is done via callback. The VEdges system therefore sets a callback, for contacting the Overlay plugin. Further information about the Overlay plugin will be in section 4.5.4.6.

4.5.3.5 Sending

Packets that ought to be sent over the network can have different origins. We call the those originating from inside the communication system internal packets. Data from outside of the communication system, for example in a plugin, is used in external packets.
Internal ones are generated, wrapped [2] in a layer that carries the next hop's IP address and then put into the send queue. This is a priority FIFO queue and thereby makes it possible to prioritize management and operations messages, but keep the order of packets in general. A task then gets elements from the send queue and processes them further: It transforms the packet into the binary format and applies encryption and hashing as described in section 4.5.3.8. Afterwards, it sends them to the next hop using nonblocking socket operations.
External packets are given to the communication system as a combination of payload and path. The payload is then wrapped into a payload layer and the id is computed by using a randomly initialized hash function. Afterwards, it is wrapped into a forward layer for every hop on the path - except the first - beginning with the last.

```
for index in reversed(range(1, len(path))):
            packet = ConnectionManager._wrap_forward_hop(
               ↪ packet, path[index])
```

This will result in an onion-like packet structure, to be interpreted by the forwarding hops as described in section 4.5.3.7. After this procedure, the first hop and the packet are further processed like an internal packet and its destination.

[2] *wrapped* is a figurative term that assumes, one layer encases the lower ones like a piece of cloth forming a pouch or like the layers of an onion. This is used even though there is no tailing part after the lower layers, as one could assume by the term.

4.5.3.6 Receiving

On the receiving end, the first action is to read bytes from the socket and then reverse the encryption and check the hash, as described in section 4.5.3.8. After a successful verification, the packet is put into a queue for further processing. Another task then sorts the packets by their outer layer: Management and other internal packets are processed immediately or send to an internal processing queue. Payload and Forward layers are only put in queues for further processing, if they have been sent over an established link. The same accounts for Operation packets which are sent to the respective plugin using an event callback.

4.5.3.7 Forwarding

The forwarding procedure is the least complex one. The outer most Forward layer, is interpreted as an instruction where to next send the rest of the packet. If the next hop is not a link partner or the link state is not *connected*, an error is generated. Otherwise, the packet is stripped of the most outer layer and then treated like an internal packet, described in the sending procedure in section 4.5.3.5.

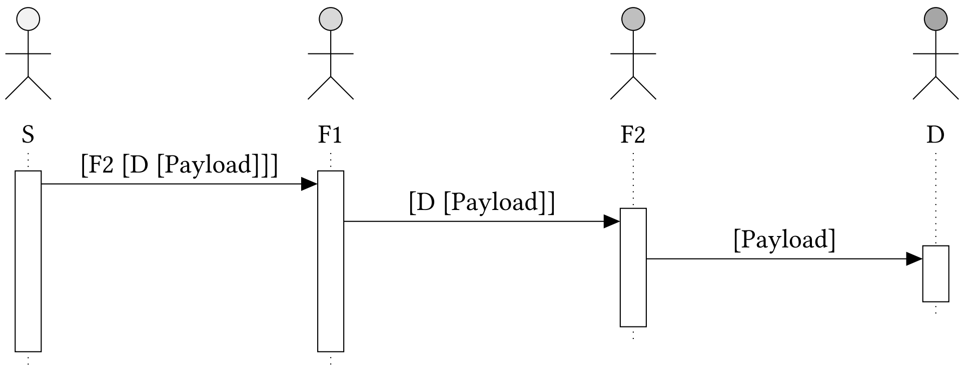

Figure 4.5: Forwarding Packets

4.5.3.8 Encryption and Error Checking

The software is made to use it over the Internet. Therefore, we decided to add the possibility to encrypt data. This is done using a plugin as described in section 4.5.4.6. The cryptographic operation is the last one applied before a packet is sent over the Internet and thus the first to be performed after receiving a packet.
Right before the encryption a hash over the bytearray is calculated and appended to it. This hash can then be reproduced on the receiving end, right after decrypting the packet, to check if a packet has been transmitted without an error. Also, this is a

good check to see if the right encryption key was used: If the decryption key differs from the encryption key, it is unlikely, that the decryption process results in a matching packet-hash combination. In the python implementation we used the non-cryptographic Murmur hash. This seemed to be a good option, since it is fast and reproducible on different machines as compared to the normal python hash, that seems to use a random initialization at the beginning of a session, which is sufficient for reproduction inside one program run, but not over different instances.

4.5.4 Plugins

A lot of the main program logic is outsourced into plugins. This section will explain the purpose of the different plugin types and how they have been implemented in this thesis's test implementation. It is important to bring to mind, that one plugin can have several types.

4.5.4.1 Plugin Manager and Base Functions

The Plugin Manager (PM) is the object of the plugin subsystem. It provides an interface to add and remove plugins from the library. Each plugin will add itself to the library after its class definition, to be available for use. The PM however during the setup of the system, is prompted to scan certain directories for plugin files[3] and import those.

```
from multiroutema.vedgesplugin import PM
PM.register_plugin(MyPlugin())
```

Every call to a plugin's function is also processed through the PM. Therefore, it is addressed by its name and type. The call to a filter representing the first step of IRS, can be seen in the example:

```
PM.execute_plugin_call("avoid_asn",          # plugin name
                       PluginGraphFilter,    # plugin type
                       "filter_nodes",       # plugin method
                       underlay)             # parameter
```

To make this possible, each class has to provide four essential methods:

get_plugin_name returns the name, used in configuration files and plugin calls

get_plugin_id returns a unique id number, used in messaging

get_plugin_dependencies returns names of plugins,necessary for its execution, in a list.

[3]It is assumed, that all .py files, that are put into the plugin directories, are in fact plugin files

set_state is the function to set the current state, which includes the current plugin configuration

The name works as the unique ID. It is used in method calls and for the identification of the plugin's section in the configuration file.

4.5.4.2 Dependencies

If a plugin depends on another to be present, it is useful to verify this at initialization time. Any plugin can call any other plugin even without having it in the dependency list, but this can lead to malfunction. In general however plugins also call other plugins using the PM.

4.5.4.3 Inter Plugin Communication

This type is used for plugins, wishing to communicate with other instances over the network. For this purpose, they get a callback set by the control system to send messages. The callback, puts the message in an operation layer with the channel identification number, given by the return value of the plugins get_plugin_id function. The idea is, to have separate channels for different plugins. Only messages of their counterparts on other instances in the network will reach plugins, which can therefore interpret every packet without a need to filter. This is for convenience and not a security mechanism, because every plugin can define the channel it communicates on by responding with the corresponding ID when queried.

4.5.4.4 Graph Filter and Path Finder

Those plugins bring the essential functionality for IRS. The Filter type is there for boolean requirements. There can be several Filter plugins at the same time. These plugins provide a graph, safe in regard to an unwanted POI.
The Path Finder plugins, of which only one can be selected at a time, then search for the requested number of paths. We did,as mentioned in the previous chapters, focus on IRS with respect to diversity, and optimize our paths for the metrics from section 4.1. The plugin structure however allows to implement Path Finders, that are optimized for any other kind of metric.

4.5.4.5 Overlay Applicatons

Overlay applications are plugins that use the whole network for a purpose. It can create data to send or wait for input from other instances.

As an example overlay application, we implemented a chat system, which can be used to send any text that is typed into the command line prompt and sends it to a specified target. This application generates data by user inputs and at the same time waits for messages from the network to display them to the terminal. A long-term goal would be an overlay application that provides a TUN/TAP interface to send arbitrary IP traffic over the network.

4.5.4.6 Other Plugin Types

Node Labeler—necessary to annotate underlay nodes. It creates POIs for IRS. In our setup, we used this plugin type to tag underlay nodes with their AS number. These labels can later be used in Filter-type plugins. There can be several plugins of this type at the same time.

Cryptography—provides an encrypt and a decrypt method. Our very basic version of this uses python library for AES-CBC encryption. AES is an encryption algorithm, that is widely seen as secure. CBS stands for Cypher Block Chaining and can be used to encrypt messages of variable length.

Identity Provider—important for the network construction, as it provides the node identity in the layered graph. In our implementation we used an address string, set in the configuration file. This is feasible enough with relatively small[4] networks, as we used them.

Graph Provider—provides an interface to get Overlay and Underlay including the association between them. More details in section 4.5.4.7

Link Changed Responder—if selected, gets called whenever a link state changes. This is useful for plugins that want to keep track of the overlay topology. We also use this in the example below.

4.5.4.7 Example: Gossip Graph Provider

As an example of how the plugins work, we will now describe a plugin, used as a graph provider in the test setup, that is evaluated in chapter 5.
The plugin is of type Link Changed Responder, Communication Partner and of type Graph Provider. It works by exchanging topology information with the direct neighbors

[4] 50 to 100 nodes per setup

and propagating changes from one neighbor to the others.
To do so, the plugin reacts to link changes: A freshly established link with a partner gets explored by asking the neighbor for his overlay identity and topology. This new information gets then merged with the already existing topology the plugin has gathered and updates about nodes get propagated to all already existing neighbors. If a link state changes to disconnected, the overlay link to the corresponding neighbor gets removed from the Overlay and again the other neighbors get informed. This plugin uses the communication type API to exchange data with the neighboring instances. For data synchronization, a vector clock is used, which keeps track of every nodes version. The version gets raised by one, every time the node or its outgoing connections change. That way, merging two graphs can be done using the most current information from both sides.
Every node manages itself and the outgoing virtual links, including the exploration of the underlay paths. A new information, gathered by the node or learned from a neighbor, is propagated. So eventually all nodes have the same view on Overlay and Underlay.

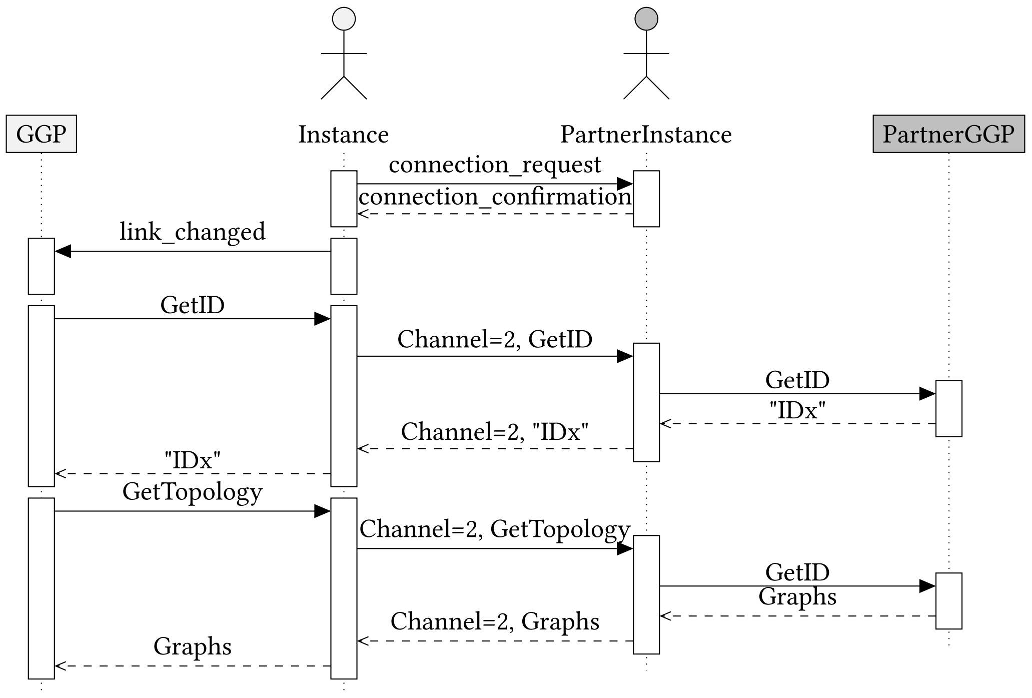

Figure 4.6: GossipGraphProvider Example Sequence after Connection

4.5.5 Orchestration

The core element of the system is the VEdges object, which has been named after the key idea of the system: Virtual Edges, that are associated with information paths. The subsystem's tasks are interpreting the configuration and setting up the system, the management of interactions between plugins and networking, the management of data streams in different directions and logging basic information about the process.

4.5.5.1 Initialization and Plugin Selection

The initialization is done in three stages. First, the configuration, which is given in a file, is further parsed and divided into the VEdges configuration, used to configure the control element itself, subsections, used for other subsystems, like the plugin system and one additional configuration section per plugin.
Another task is to *select* plugin instances for certain jobs as configured by the user. This is important because it enables the software to have several instances of the same type loaded, but only one selected for a certain task. The initialization process is also responsible for dissolving dependencies and it must be able to handle cyclic dependencies. This stage is therefore also responsible for checking the formal saneness of the plugin system.
The network setup is done by setting the parameters of the communication system, opening the socket and binding it to an address. Moreover the callbacks, necessary for the communication system to interact with plugins have to be set. We think, a callback based approach is the best way to stay fast responding, while leaving the VEdges object in full control over the interactions between the plugins and the network.
As the last step for the initialization, the eventloop is started. This starts the normal operation mode.

4.5.5.2 Input and Output Loop

The input and output loop do mostly the same thing, but react to different events. Their purpose is to connect the overlay application, given by plugin, to the network. The input loop waits for the local overlay application to initiate communication. The output loop waits for the network and hence for a communication partner to initiate communication. The output loop will then deliver the packet from the network to the overlay application. In case the plugin callbacks yields data, it is forwarded to the send queue. One difference between input loop and output loops is however: The send loop, if configured to do so, will visualize the current topology. This is relevant for the evaluation of IRS results in context with potential graph originalities.

4.5.5.3 Send Loop

The send loop waits for packets, to be sent over the network. Once such a packet is available, it performs IRS and sends it through the network. This is done involving two plugin types. First, all selected filter plugins are called, to make sure, the boolean requirements are met. Afterwards, a plugin call to a path finder, will select a requested number of routes. Depending on the actual number of found paths a third plugin, of the redundancy type, will prepare the data. The prepared data is then handed over to the Connection manager and sent over the calculated paths. If anything fails in this process, the overlay application is notified and can react accordingly.

4.5.6 Visualization

To evaluate path selection algorithms or other plugin types in general, it can be useful for the user to visualize the network. For this purpose, there is an option to visualize overlay and underlay after sending a packet. This has to be enabled with care, because it slows the program massively down. The visualization merges both graphs into one picture. Typically for graph visualizations, there are circles for nodes and lines between them for the connecting edges. This fits well for the nodes, since the nodes in the Overlay are also represented in the Underlay. Overlay nodes are therefore just drawn slightly bolder. The edges however are also dashed to make a clear distinction possible. When using the Path visualization, which is also configurable, a path and the corresponding Underlay path are highlighted in the same color. If two or more paths share an edge, it is colored in only one color. An example visualization can be seen in figure 4.7 and another, without underlay expansion can be seen in figure 4.8.

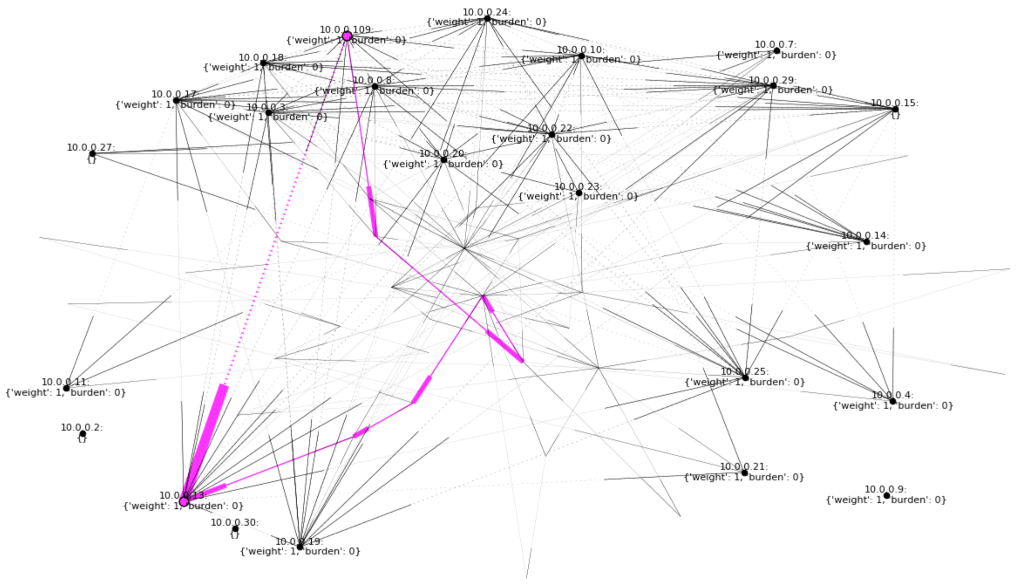
10.0.0.24:
{'weight': 1, 'burden': 0}
10.0.0.109:
{'weight': 1, 'burden': 0}
10.0.0.18:
{'weight': 1, 'burden': 0}
10.0.0.10:
{'weight': 1, 'burden': 0}
10.0.0.7:
{'weight': 1, 'burden': 0}
10.0.0.8:
{'weight': 1, 'burden': 0}
10.0.0.29:
{'weight': 1, 'burden': 0}
10.0.0.17:
{'weight': 1, 'burden': 0}
10.0.0.3:
{'weight': 1, 'burden': 0}
10.0.0.15:
{}
10.0.0.22:
{'weight': 1, 'burden': 0}
10.0.0.27:
{}
10.0.0.20:
{'weight': 1, 'burden': 0}
10.0.0.23:
{'weight': 1, 'burden': 0}
10.0.0.14:
{'weight': 1, 'burden': 0}
10.0.0.25:
{'weight': 1, 'burden': 0}
10.0.0.11:
{'weight': 1, 'burden': 0}
10.0.0.4:
{'weight': 1, 'burden': 0}
10.0.0.2:
{}
10.0.0.21:
{'weight': 1, 'burden': 0}
10.0.0.13:
{'weight': 1, 'burden': 0}
10.0.0.9:
{'weight': 1, 'burden': 0}
10.0.0.30:
{}
10.0.0.19:
{'weight': 1, 'burden': 0}

Figure 4.7: Virtual Edge Visualization Example

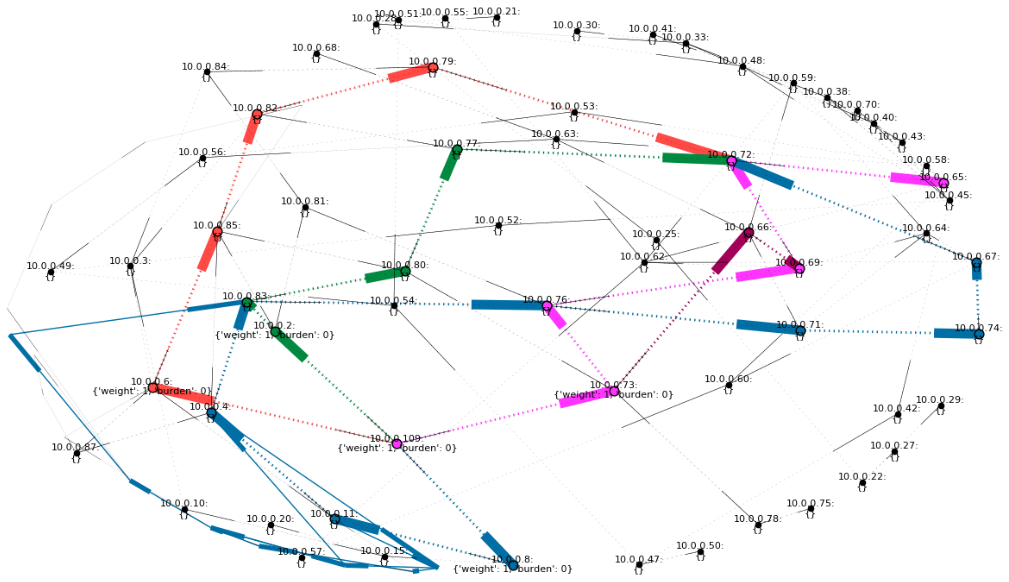

Figure 4.8: Multiple Paths Visualized Without Underlay

Chapter 5

Evaluation

In the previous chapters, we elaborated why path diversity useful for routing, the problems that arise in the Internet, how we plan to overcome those and which design and implementation decisions were made in order to build a prototype. The following chapter will cover the evaluation of IRS concerning its performance to provide diverse paths in a network. We will explain the experiments' intuitions, the experiment setup and execution and in the end, analyze the results.

5.1 Intuitions

For the experiments, some fundamental assumptions are made. We will now explain those and elucidate the experiment design decisions with respect to the algorithms, used for comparison, the number of paths that are computed and compared and the burden to weight mapping in IRS. The comparison will first be made on different sized graphs, generated with the Hyperbolic Graph generator from [16]. Afterwards, the same measures will be applied to a topology, extracted from the Internet using the Planetlab infrastructure.

5.1.1 Comparison Algorithms

To classify the performance of IRS, we decided to compare it to two other algorithms.

5.1.1.1 Random Selection Overlay Routing

This algorithm looks for routes in the Overlay only using the information given in that graph. To achieve Path Diversity, it selects random nodes as relays. It then calculates a shortest path to the relay node and a shortest path from there to the destination. If the

path between source and relay leads over the destination node, it is shortened to this direct path from source to destination. A Path can only occur once in the result path set. Therefore, as in IRS, we use a mechanism to avoid endless iterations.
This is derived from the approach in [12]. They use a random set of detour nodes and claim that this yielded the best results for reliability to have a fast recovery after a single node failure.

5.1.1.2 Random Selection Full Mesh Routing

This algorithm will also select a random set of Overlay nodes as relays. The routing between them is however not done by finding a shortest path in the Overlay, but in the Underlay. This violates the assumption, edges in the Underlay are only information associations, from section 4.2.1.2. But since in our experiment, the underlay only consists of (AS-wise) routing hops, this is nevertheless a legitimate routing algorithm. It implements the algorithm from above (section 5.1.1.1) considering we had an ideal path between each pair of nodes. This represents an idealized full mesh.

5.1.2 Variables

The experiments are executed with respect to several variables. The effect of each of the variable on the performance of IRS, as compared to the other algorithms will be subject of the evaluation.

5.1.2.1 Number of Paths

The number of paths requested and returned is an interesting variable. It helps to see how well an algorithm fits to a specific use case. We selected four different values for this variable:

Two Paths as a lower bound for multipath. An example use case was regular data routing with an additional backup path if necessary.

Three Paths as an example multipath networking, with parallel utilization of several lines and FEC for resilience without application layer visibility.

Five Paths as an example for multipath with very high resilience requirements or for applications that use resource pooling.

Ten Paths as an prospect on how well the algorithm scales in the given graph.

5.1.2.2 Burden Punishment Factor

As mentioned in section 4.4.3.1, the linear mapping function from burden to Overlay edge weight will be parameterized by θ. By tweaking this factor θ, this parameter will also be subject to the evaluation. In fact, we pick one particular θ value after a range of experiments and use this experimentally determined parameter in the comparison of algorithms.
Through sampling pre tests, we determined a value-range for θ to be good between zero and 14. We noticed, that there are is very small rate of change after that. Also, negative Values would encourage the use of already used virtual edges, which is not what we want.

5.1.2.3 Different Graphs

The size of the Overlay and Underlay Graphs are also taken into account when comparing algorithms, to observe the influence of the number of nodes and the number of hops per virtual edge influence the performance of IRS.
The last step is to review the insight we have gained in the previous steps towards the real world. For this, we use our software on the Planetlab infrastructure, create a layered graph model and run the calculations again, with the optimized value for θ on this data.
For graph sizes we used a very small graph, with 300 Underlay nodes and 40 Overlay nodes to simulate a network as it could be in a company. A medium sized graph with 1.000 Underlay nodes and 100 Overlay nodes for an intermediate step. And to get an Idea how it could perform in the Internet with about 75.000 ASes, we used another Graph with 100 Overlay nodes and 10.000 Underlay nodes. The Planetlab graphs turned out to have 80 Overlay nodes and 200 Underlay nodes.

5.2 Experiment Setup and Execution

The experiments are set up to determine how well IRS performs. The approach will be to test whether or not IRS out performs random detour routing. If it does, we want to find out how much better it can get. Because we want to use the algorithms on a set of different setups, we will test the applicability to the real world in a last step.

5.2.1 Planetlab

Planetlab is a research network of 1353 nodes at 717 sites around the globe [34]. A list of the 81 nodes, used in our experiment, can be found in the appendix.

On the Planetlab our software did not run natively. Because the nodes run different operating systems, we decided to compile Python3.5 on each node and created an environment to run our software. Also, on some nodes the raw socket api was disabled, even with root access. We therefore had to use the linux traceroute implementation for information gathering.
We used several scripts to set up the nodes in two different topologies: One was a full mesh, to gather all traceroute data we could. the other one was a ring structure. We chose the ring structure to make sure, there is always more than one completely Overlay disjoint paths to a certain node. Because Planetlab nodes are in most cases not multi homed, the underlay paths will have their home ASes twice: on the path to the node and on the way back into the Internet.
The scripts uploaded a customized configuration file to the host. Afterwards, each node had the same setup except from the configuration file and we were able to control all nodes using the same commands.
On top of this structure, we also ran an experiment where we sent an echo request to nodes. This was to verify, bidirectional communication throughout the network was possible.

5.2.2 Overlay Application for Test

To test the algorithms, we created an OverlayApplication plugin. This application first takes some input:

1. select a PathFinder plugin
2. select a configuration file parameter to increase after each round
3. select start and end value for the parameter
4. set a number of paths to request
5. add one or more graphs to test the algorithms with
   those can be either pickled or generated hyperbolic graphs
6. specify the name of a results folder

It then, for each pair of Overlay nodes in each of the given graphs and for each value of the configuration parameter, tries to find the number of paths with the given plugin. After that it calculate the metrics from section 4.1 and save the result of those to a CSV file. It will also dump a binary form of the calculated paths into subfolder.

5.3 Results

In the following results we compare the three algorithms. The axis named Theta marks the parameter θ. The two comparison algorithms that are not depending on θ, are displayed as straight lines. Each Image displays all three algorithms and the whole range of θ values but only in one graph and for one number of requested paths. Metrics like LND, that are taken over a set, are susceptible for distortion if the set size is not constant. Therefore, in order to have meaningful statistics, only used the runs, that returned as many paths as requested. An overview over the used and unused data is given in table 5.1. There, it is also striking, that the number of discarded runs is especially high for IRS. Since Random Selection Full Mesh Routing (RSFMR) and Random Selection Overlay Routing (RSOR) do successfully find enough paths, this has to be a problem of the IRS algorithm. We assume, this occurs when the burdening leads to a weight distribution that has been there before, in that particular run. A solution, we were not able to test due to time constraints, could be non deterministic burdening, in case a path is found, that is already in the path set.

In the graphs below, dashed lines will represent the results from the RSOR, dotted lines show the results of the RSFMR and solid lines or box plots will show the IRS data. Because only IRS depends on θ, we decided to draw a horizontal line over all θ values for the other algorithm results. When box plots are used, the corresponding quantiles are drawn as such lines. This makes it possible to exactly compare every θ value with the other algorithms. Other metrics, like the Path Length and the Detour Factor are represented through three lines, where the lowest is the averaged minimum, the middle one is the averaged mean and the other is the averaged maximum.

5.3.0.1 Small Graph

The Node Diversity results for two and for three requested paths look similar. The quantiles are generally higher than the comparison algorithms'. There is in both cases a minimum θ from which on there are almost no changes in the Node Diversity. For two paths we have $\theta_2 = 9$ and for three paths, we have $\theta_3 = 3$.
This does however not mean, that θ has no further influence on the routing. In contrary, it can have a negative effect on metrics like the path length as seen in figure 5.3. In this plot, it is also visible, that RSFMR is better in finding similar sized paths, compared to IRS, that has an ideal first path and potentially increasingly longer additional paths.

Why do the figures render in different sizes?

In figure 5.4 we see, that for five requested paths, the result for diversity looks a bit different, because the RSFMR 25% quantile and the median are bot higher than the median for IRS. Apart from that, there seems to be no major influence on the LND of a higher θ after $\theta_5 = 2$

Algorithm	Requested Paths	Graph	Total Runs	Used	Discarded
IRS	2	big	138600	118157	931
IRS	2	medium	138600	136054	2546
IRS	2	small	31200	29237	1963
IRS	3	big	138600	117273	727
IRS	3	medium	138600	130793	7807
IRS	3	small	21840	19995	1845
IRS	5	big	138600	110661	5307
IRS	5	medium	138600	128746	9854
IRS	5	small	31200	20041	11159
IRS	10	big	138600	101558	37042
IRS	10	medium	198000	198000	0
IRS	10	small	31200	31200	0
RSOR	2	big	9900	9900	0
RSOR	2	medium	9900	9900	0
RSOR	2	small	1560	1560	0
RSOR	3	big	9900	9900	0
RSOR	3	medium	9900	9900	0
RSOR	3	small	1560	1560	0
RSOR	5	big	9900	9900	0
RSOR	5	medium	9900	9899	1
RSOR	5	small	1560	1560	0
RSOR	10	big	9900	9898	2
RSOR	10	medium	9900	9899	1
RSOR	10	small	1560	1558	2
RSFMR	2	big	9900	9898	2
RSFMR	2	medium	9900	9894	6
RSFMR	2	small	1560	1558	2
RSFMR	3	big	9900	9890	10
RSFMR	3	medium	9900	9893	7
RSFMR	3	small	1560	1559	1
RSFMR	5	big	9900	9893	7
RSFMR	5	medium	9900	9869	31
RSFMR	5	small	1560	1557	3
RSFMR	10	big	9900	9894	6
RSFMR	10	medium	9900	9871	29
RSFMR	10	small	1560	1555	5

Table 5.1: Used and Unused Data

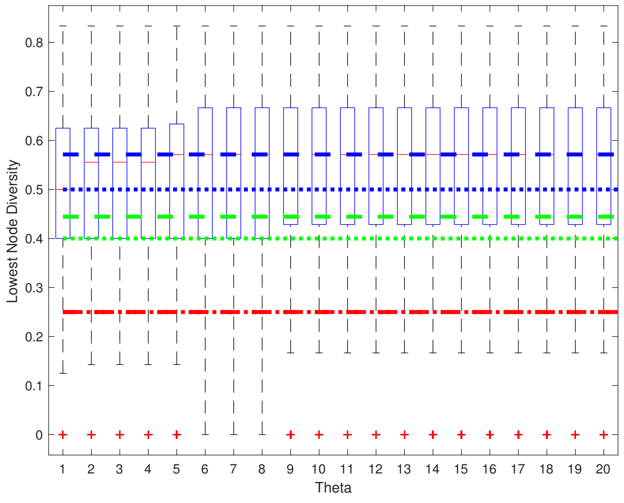

Figure 5.1: LND, Varied θ, Two Paths,Small Graph

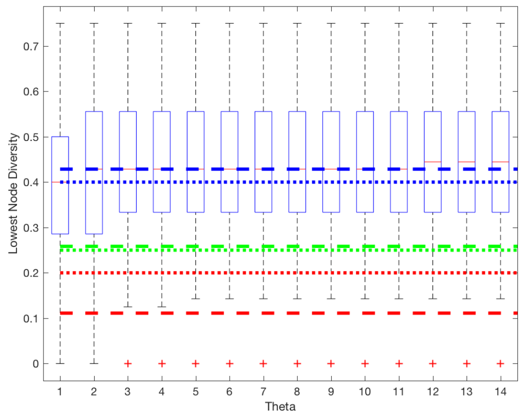

Figure 5.2: LND, Varied θ, Three Paths,Small Graph

5.3.0.2 Medium Sized Graph

In the medium sized graph, the most noticeable detail is, that, when requesting only two paths, the RSOR's 75% quantile is at a higher LND than IRS with a small θ, but with a rising θ, they are equal. The performance gap gets bigger with a rising number of paths. Apart from that, IRS performs well with respect to the LDN quantiles. Also, it seems, IRS performs especially well for three to five paths. This might be, because in the medium sized graph many detour nodes are only accessible using the same first hops.
The plot for ten paths in figure 5.8 shows very low quantiles for the Lowest Node Diversity. To add some context, we put the Mean Node Diversity in figure 5.9 right below it. From that we can see either the limits of the metric LND or the limits of useful

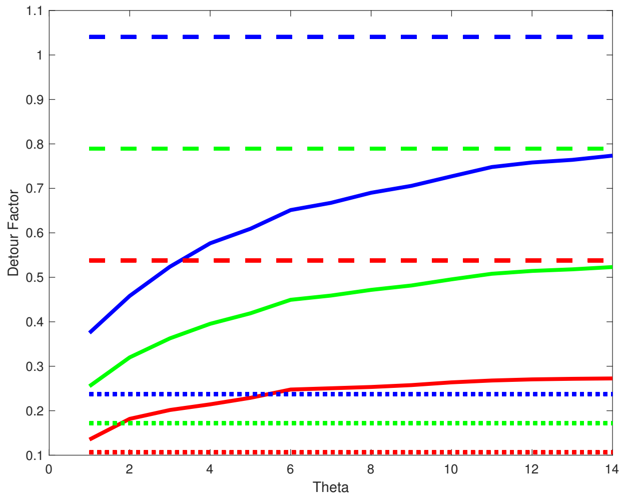

Figure 5.3: Detour, Varied θ, Three Paths,Small Graph

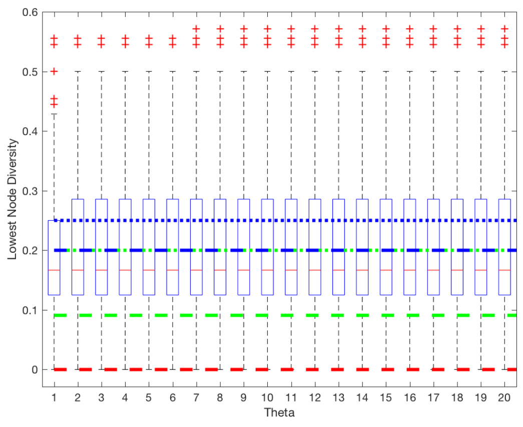

Figure 5.4: LND, Varied θ, Five Paths,Small Graph

multipath: As more paths are added to the set, they get pairwise less diverse. Therefore a POF failure would more likely affect more than one path. Depending on the use case however, this could be acceptable, since there are also more paths available. The MND plot shows, that there is a good pairwise Node Diversity.

5.3.0.3 Big Graph

The big graph is especially interesting, because its Underlay size is closer to the Internet's than the others' and the Overly size is in a frame that would be affordable for a big company to build up. We will therefore analyze the results for this graph in more detail. For that purpose, we have the LND in figures 5.10 to 5.12, the path length in figures 5.13 to 5.15 and the Minimum Node Cut in figures 5.16 to 5.18.

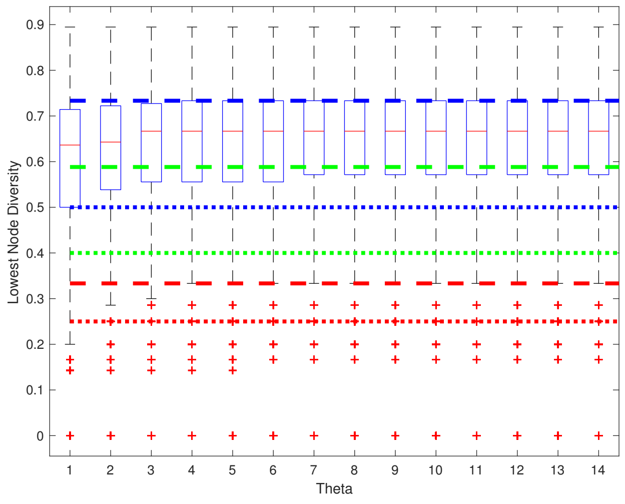

Figure 5.5: LND, Varied θ, Two Paths, Medium Sized Graph

For two paths, we look at $\theta_2 = 4$. IRS performs well here. 75% of the routing pairs have a Node Diversity of better than 0.6 and the average highest path length is about 25. When considering, a trace from Germany to a popular social network in the USA has up to 18 hops, this is a good trade off for more resilience with a backup path.
The three paths plot shows similar path lengths but a smaller LND for a $\theta_3 = 2$.
For the Five paths plot, the LND gets considerably worse. Which is, as explained above, not implying that the pairwise Node Diversity can not be good. But also, the path average worst case path length rises into unattractive heights. For $\theta_5 = 5$, it is almost 40 hops long. This is two and a half times the trace from above.
These values are all very good, when seen with respect to the results of RSOR, which produces very long paths without achieving the LND quantiles of IRS and also with respect to RSFMR, which despite producing desirably short paths, does not perform well for our metrics, because the LND is also far worse than IRS.
Also, the Minimum Node Cut, meaning the number of nodes necessary to fail to disconnect source and destination completely is for every $\theta_x >= 2$ constantly better with IRS.

5.3.0.4 Transfer to Planetlab Graph

The last step of the evaluation is an experiment using the Planetlab infrastructure and real traces, to see whether or not the assumptions made in this thesis hold. Therefore, we deployed the prototype on Planetlab nodes and first executed a test to ensure, bidirectional communication was possible; We wrote and OverlayApplication plugin to perform an echo request based on the ICMP echo messages. Even though there was loss because of some unstable links, the results, timing per host visualized in the boxplot in figure 5.19, showed a working bidirectional transport system. The timing includes all

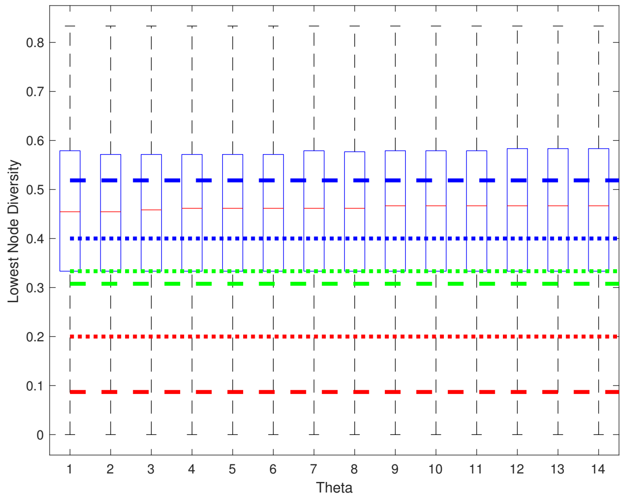

Figure 5.6: LND, Varied θ, Three Paths, Medium Sized Graph

IRS steps and for the transport, up to three paths were used.

For the last experiments, we set up two different topologies: One that almost assembles a full mesh network, seen in figure 5.20 and one ring-like topology, but with every node having direct connections to three other nodes, as seen in figure 5.21.
It turns out, IRS on the the Planetlab graphs shows similar behavior as on the small graph. This does make sense, because the size of the Underlay is the closest of all graphs. In the figures 5.22 to 5.24 we see the Node Diversity for the full mesh network plotted. Those plots also show that after a certain θ, there is no major influence on the Node Diversity. Similar behavior is also visible in the plots of the Planetlab ring topology, as seen in figure 5.25 to 5.27.

In figure 5.28 however we see, that a variable θ has virtually no influence in the MNC even in the full mesh. We found out, that almost every other virtual edge has the AS11537 in it, which belongs to Internet2, an AS that has been built up by academic institutions in the USA. Close to all Overlay paths use this AS and therefore share the same POF. This was similar in all Planetlab runs, not only in the plotted example.
This setup can also explain, why the LND of the full mesh topology has not that much higher quantiles. The transfer to shows, IRS, which has been simulated using the generated hyperbolic graphs and performed very good there, is also applicable to the Planetlab network, but does not perform equally good. This might be, because the selection of nodes was not very representative, because of the shared networks of institutions that partake in the project or because of an inaccurately chosen proportion of Overlay to Underlay nodes.
The results were however still very good. A LND of 0.5, which is the median value in figure 5.22 with a $\theta >= 2$, after all means, that it is guaranteed, that up to 50% of the shorter path's node failures will not break the connection.

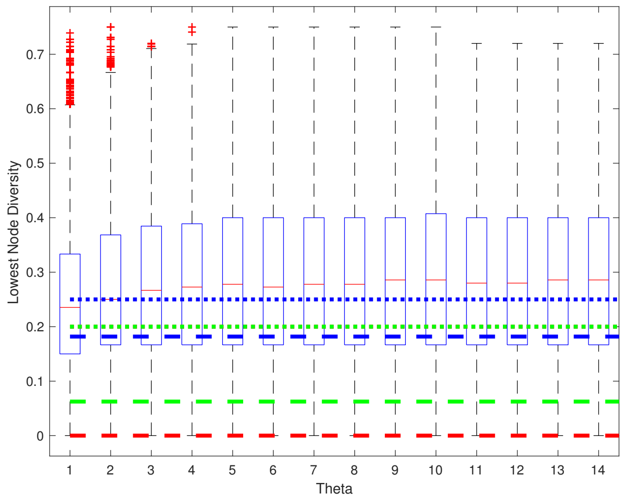

Figure 5.7: LND, Varied θ, Five Paths, Medium Sized Graph

This evaluation shows, that even though the exact parameters should be further researched, an Underlay aware Informed Route Selection is a working and good way to improve routes from the perspective of the user.

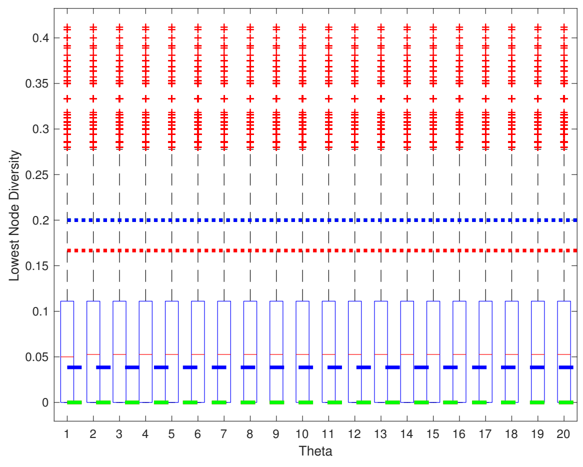

Figure 5.8: LND, Varied θ, Ten Paths, Medium Sized Graph

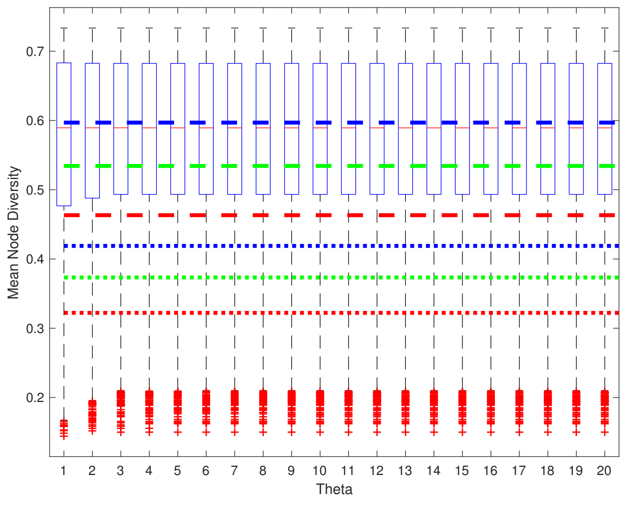

Figure 5.9: MND, Varied θ, Ten Paths, Medium Sized Graph

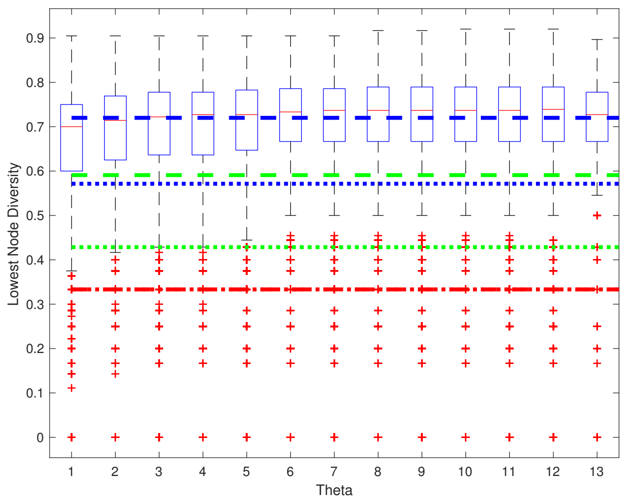

Figure 5.10: LND, Varied θ, Two Paths, Big Graph

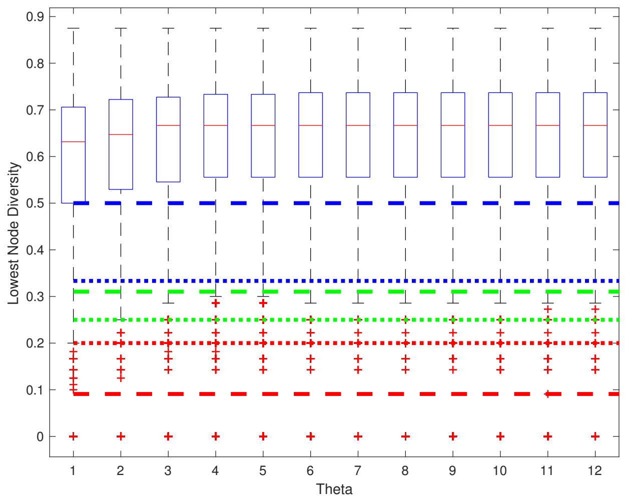

Figure 5.11: LND, Varied θ, Three Paths, Big Graph

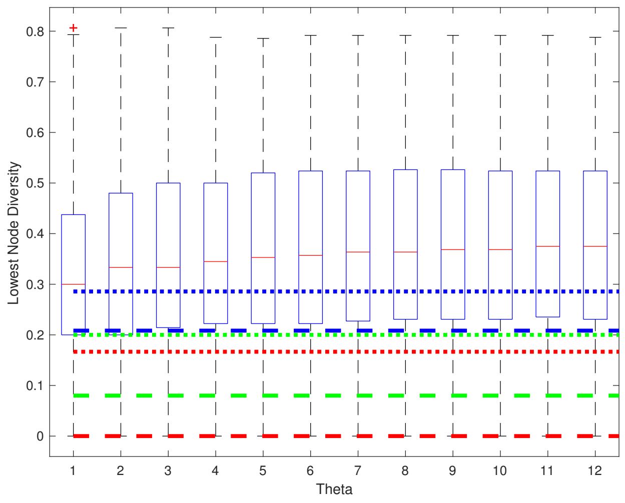

Figure 5.12: LND, Varied θ, Five Paths, Big Graph

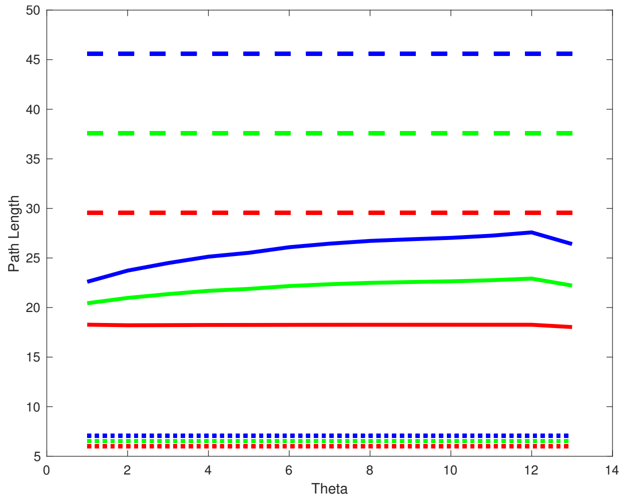

Figure 5.13: Path Length, Varied θ, Two Paths, Big Graph

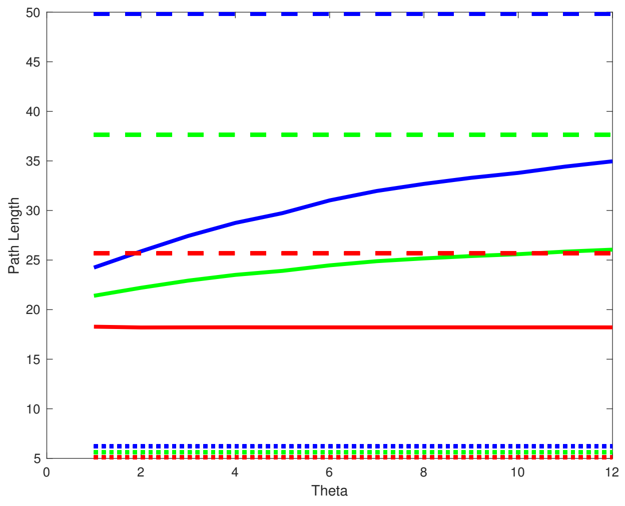

Figure 5.14: Path Length, Varied θ, Three Paths, Big Graph

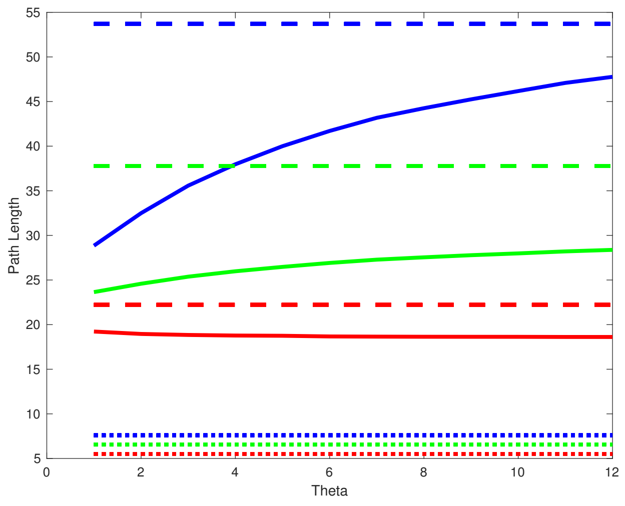

Figure 5.15: Path Length, Varied θ, Five Paths, Big Graph

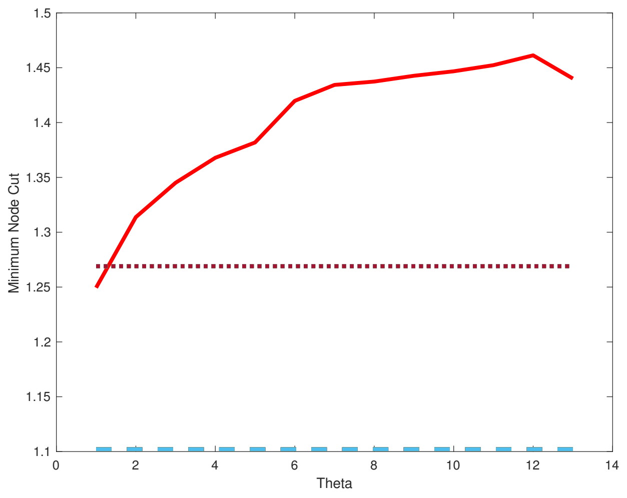

Figure 5.16: MNC, Varied θ, Two Paths, Big Graph

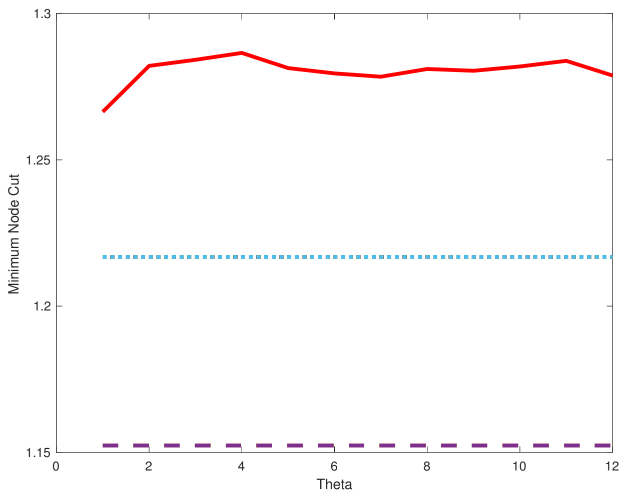

Figure 5.17: MNC, Varied θ, Three Paths, Big Graph

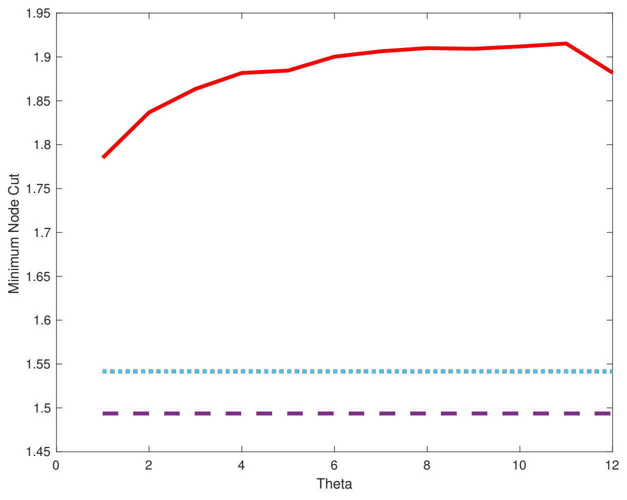

Figure 5.18: MNC, Varied θ, Five Paths, Big Graph

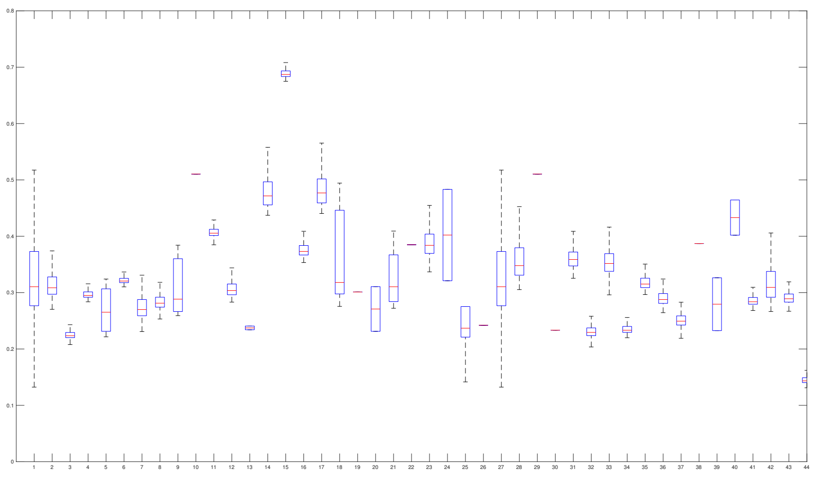

Figure 5.19: Echo Timing

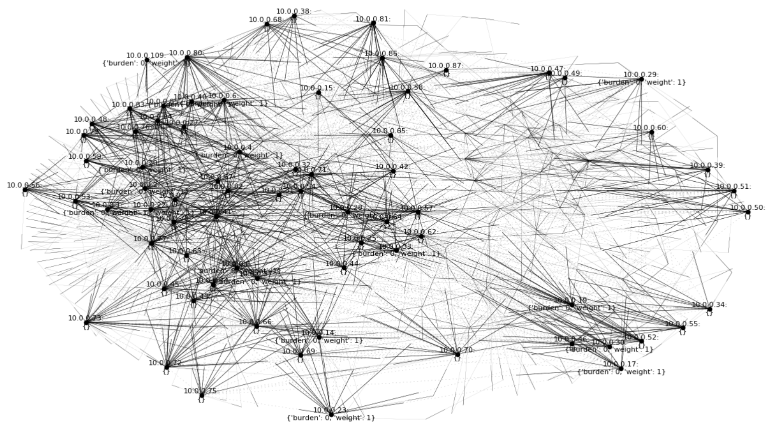

Figure 5.20: Planetlab Topology - Full Mesh

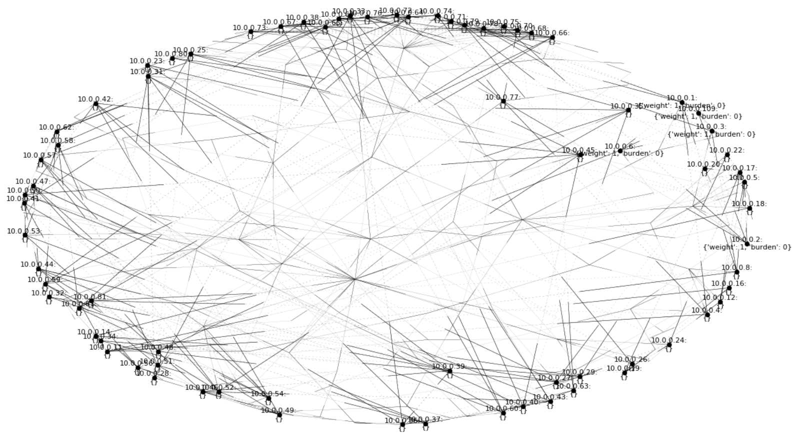

Figure 5.21: Planetlab Topology - Ring

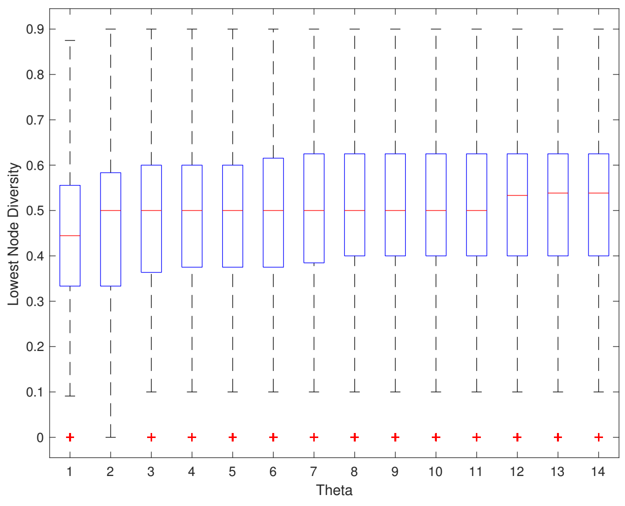

Figure 5.22: LND, Varied θ, Two Paths, Planetlab Full Mesh

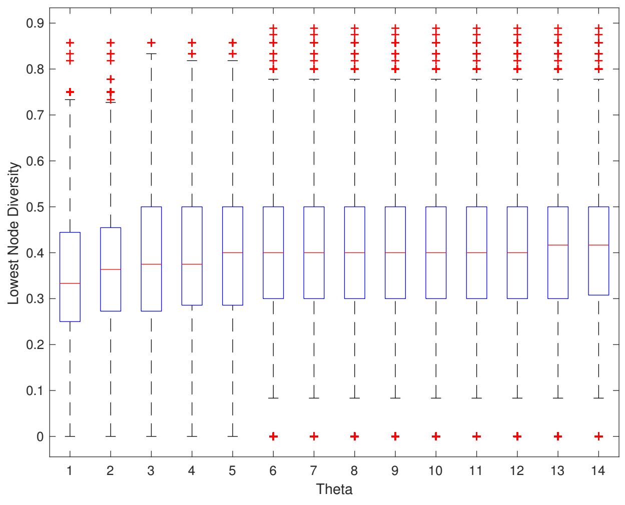

Figure 5.23: LND, Varied θ, Three Paths, Planetlab Full Mesh

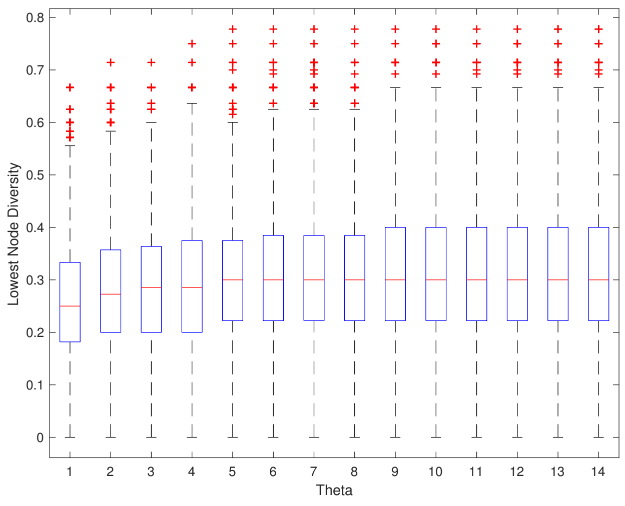

Figure 5.24: LND, Varied θ, Five Paths, Planetlab Full Mesh

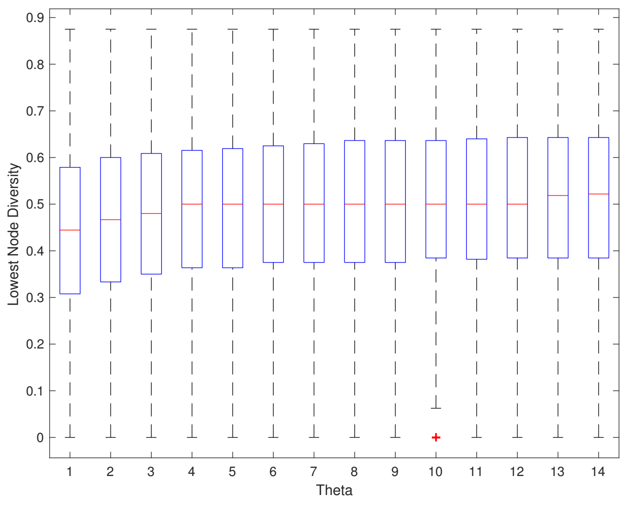

Figure 5.25: LND, Varied θ, Two Paths, Planetlab Ring

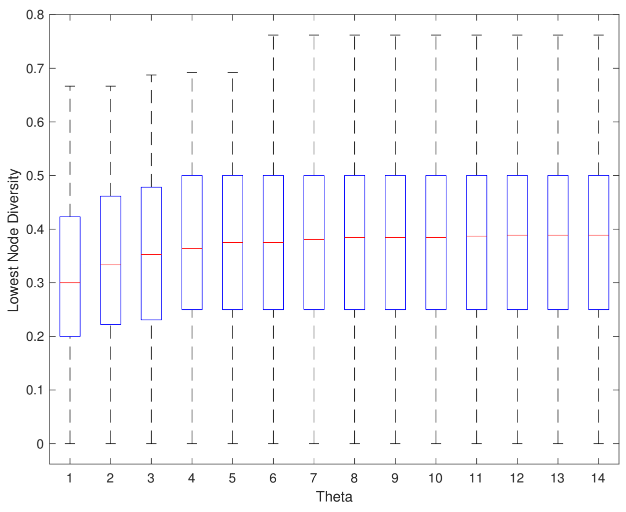

Figure 5.26: LND, Varied θ, Three Paths, Planetlab Ring

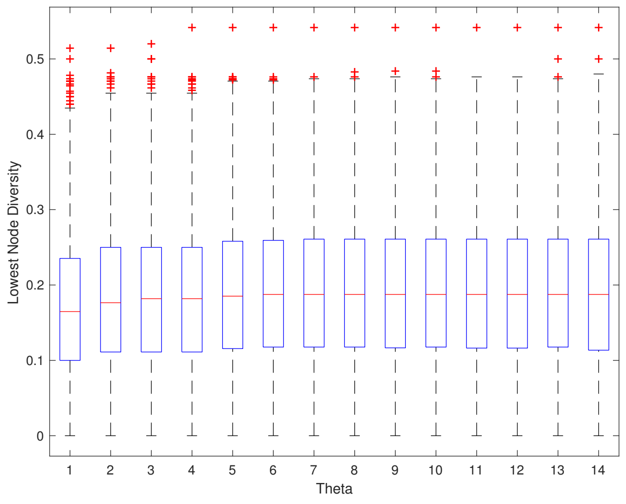

Figure 5.27: LND, Varied θ, Five Paths, Planetlab Ring

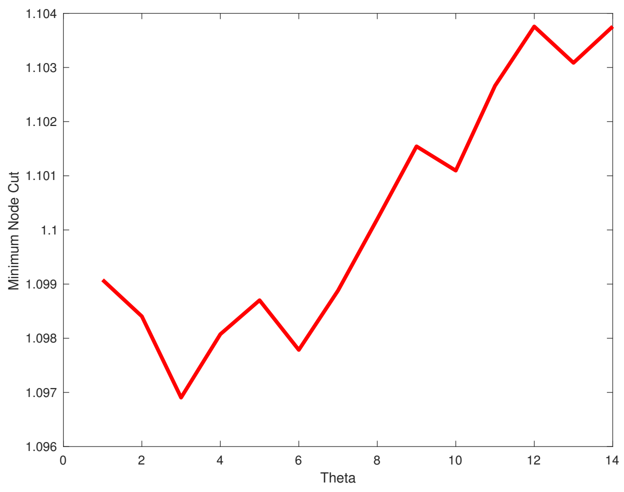

Figure 5.28: MNC, Varied θ, Three Paths, Planetlab Full Mesh

Chapter 6

Conclusion and Future Work

In this thesis we introduced IRS to improve end to end resilience in the Internet. We did this by first explaining the current situation in the Internet in chapter 2, where also resilience and different approaches to improve it were elaborated. In the course of the thesis, we looked further into routing techniques and requirements in chapter 3, where we also introduced the concept of IRS. Afterwards, in chapter 4, we developed a prototype, first in theory, then in praxis to experiment with IRS in different scenarios. This prototype and the concept of IRS were then evaluated in chapter 5, where we found out, that IRS performs good on the artificial environment and also archieved a good LND on the Planetlab infrastructure.
Yet, there are still questions open for further research:

Information Gathering IRS allows to model and evaluate all different kinds of requirements into a layered routing graph. In the course of this work, we were able to successfully demonstrate this using AS information, extracted from traceroute traces and optimized IRS for resilience. But different information could be used to improve different goals. Examples like the requirement to stay in one jurisdiction were named in section 3.2.

Dynamic Pairing As it is now, the prototype relies on preconfigured links. A better system would try to pair with the closest nodes, with respect to underlay paths. This would lead to a better performance, because a finer path selection would be possible. An example implementation could probe the paths to different targets and select the most attractive ones while still taking care no subnetwork segregation happens.

Router Distribution In the evaluation we used all of the Planetlab nodes we could and since there was a certain shortage, were not able to use carefully selected nodes for IRS. Similar to dynamic pairing, a carefully selected set of routing points could improve the performance of IRS.

IP Overlay Networking A very interesting OverlayApplication plugin would create a TUN/TAP network interface and enable routing real IP traffic over the network.

Intelligent FEC The current FEC plugin will only clone packets and therefore not use the multipath capabilities efficiently. A better plugin could improve the troughput and take a step into the direction of resource pooling.

The plugin based design of the prototype will allow all of this and more.

Appendix A

Planetlab Test

Allocated Nodes

mars.planetlab.haw-hamburg.de
planetlab3.cesnet.cz
ple1.cesnet.cz
planetlab2.inf.ethz.ch
ple2.cesnet.cz
planet-lab-node2.netgroup.uniroma2.it
onelab2.pl.sophia.inria.fr
planetlab3.inf.ethz.ch
planetlab2.tlm.unavarra.es
planetlab-2.research.netlab.hut.fi
planetlab3.mini.pw.edu.pl
planetlab4.inf.ethz.ch
stella.planetlab.ntua.gr
planetlab4.mini.pw.edu.pl
dplanet2.uoc.edu
planetlab-05.cs.princeton.edu
planetlab2.cesnet.cz
planetlab1.cesnet.cz
plab1.cs.msu.ru
salt.planetlab.cs.umd.edu
planetlabone.ccs.neu.edu
planetlab3.rutgers.edu
pl1.rcc.uottawa.ca
planetlab1.dtc.umn.edu
plonk.cs.uwaterloo.ca

planetlab2.dtc.umn.edu
planetlab5.eecs.umich.edu
planetlab-5.eecs.cwru.edu
node1.planetlab.mathcs.emory.edu
node2.planetlab.mathcs.emory.edu
saturn.planetlab.carleton.ca
planetlab1.cs.uml.edu
aguila2.lsi.upc.edu
planetlab2.cs.uml.edu
planetlab1.temple.edu
pl1.ucs.indiana.edu
pl2.ucs.indiana.edu
plink.cs.uwaterloo.ca
pl1.cs.montana.edu
planetlab-2.cse.ohio-state.edu
pluto.cs.brown.edu
planetlab2.cs.ubc.ca
planetlab3.eecs.umich.edu
planetlab3.wail.wisc.edu
planetlab1.cs.du.edu
ricepl-2.cs.rice.edu
planetlab02.cs.washington.edu
planetlab3.cs.uoregon.edu
ricepl-5.cs.rice.edu
planetlab04.cs.washington.edu
planetlab2.cs.uoregon.edu
planetlab1.cs.uoregon.edu
planetlab01.cs.washington.edu
planetlab4.cs.uoregon.edu
planetlab1.unr.edu
ricepl-1.cs.rice.edu
planetlab2.pop-mg.rnp.br
planetlab1.pop-mg.rnp.br
planetlab1.cs.ucla.edu
planet-lab1.uba.ar
planetlab2.cs.ucla.edu
pl1.sos.info.hiroshima-cu.ac.jp
planet-lab4.uba.ar
pl2.eng.monash.edu.au
pl1.eng.monash.edu.au
planetlab2.cs.otago.ac.nz

planetlab2.ecs.vuw.ac.nz
planetlab1.cs.otago.ac.nz
planetlab1.ecs.vuw.ac.nz
planetlab-1.sjtu.edu.cn
planetlab-2.scie.uestc.edu.cn
planetlab-1.scie.uestc.edu.cn
pl2.pku.edu.cn
pl1.pku.edu.cn
pl1.6test.edu.cn
pl2.6test.edu.cn
planetlab-2.calpoly-netlab.net
planetlab-n2.wand.net.nz
planetlab-n1.wand.net.nz
planetlab2.cs.cornell.edu
planetlab2.utdallas.edu

Bibliography

[1] [Online]. Available: https://wiki.ittc.ku.edu/resilinets/Main_Page

[2] (2016, 06). [Online]. Available: http://www.internetworldstats.com/stats.htm

[3] R. Durairajan, P. Barford, J. Sommers, and W. Willinger, "Intertubes: A study of the us long-haul fiber-optic infrastructure," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 565–578, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787499

[4] N. I. A. Council. (2009, 09) 08 fall national infrastructure advisory council critical infrastructure resilience final report and recommendations. [Online]. Available: https://www.dhs.gov/xlibrary/assets/niac/niac_critical_infrastructure_resilience.pdf

[5] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, Aug. 2012. [Online]. Available: http://doi.acm.org/10.1145/2377677.2377680

[6] B. Carpenter, "Architectural principles of the internet," Tech. Rep., 1996.

[7] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 131–145, Oct. 2001. [Online]. Available: http://doi.acm.org/10.1145/502059.502048

[8] A. F. Hansen, A. Kvalbein, T. Cicic, S. Gjessing, and O. Lysne, "Resilient routing layers for recovery in packet networks," in *2005 International Conference on Dependable Systems and Networks (DSN'05)*, June 2005, pp. 238–247.

[9] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast recovery from link failures using resilient routing layers," in *10th IEEE Symposium on Computers and Communications (ISCC'05)*, June 2005, pp. 554–560.

[10] M. Menth and R. Martin, "Network resilience through multi-topology routing," in *The 5th International Workshop on Design of Reliable Communication Networks*, 2005, pp. 271–277.

[11] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path splicing," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 27–38, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402963

[12] P. K. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, D. Wetherall *et al.*, "Improving the reliability of internet paths with one-hop source routing." in *OSDI*, vol. 4, 2004, pp. 13–13.

[13] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 159–170, Aug. 2006. [Online]. Available: http://doi.acm.org/10.1145/1151659.1159933

[14] J. P. Rohrer, A. Jabbar, and J. P. Sterbenz, "Path diversification: A multipath resilience mechanism," in *Proceedings of the IEEE 7th international workshop on the Design of Reliable Communication Networks (DRCN)*, 2009, pp. 343–351.

[15] J. P. Rohrer, R. Naidu, and J. P. G. Sterbenz, "Multipath at the transport layer: An end-to-end resilience mechanism," in *2009 International Conference on Ultra Modern Telecommunications Workshops*, Oct 2009, pp. 1–7.

[16] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá, "Hyperbolic geometry of complex networks," *Phys. Rev. E*, vol. 82, p. 036106, Sep 2010. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.82.036106

[17] D. Habibi and Q. V. Phung, "Graph theory for survivability design in communication networks," 2012.

[18] F. Iqbal and F. A. Kuipers, "Disjoint paths in networks," *Wiley Encyclopedia of Electrical and Electronics Engineering.*

[19] J. Sommers, P. Barford, and B. Eriksson, "On the prevalence and characteristics of mpls deployments in the open internet," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 445–462. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068858

[20] K. Chen, D. R. Choffnes, R. Potharaju, Y. Chen, F. E. Bustamante, D. Pei, and Y. Zhao, "Where the sidewalk ends: Extending the internet as graph using traceroutesfrom p2p users," *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 1021–1036, April 2014.

[21] R. Dube, "A comparison of scaling techniques for bgp," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 3, pp. 44–46, Jul. 1999. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/505724.505729

[22] Y. Rekhter, T. Li, and S. Hares, "Rfc 4271," *Internet Engineering Task Force, http://www. rfc-editor. org/rfc/rfc4271. txt, access on*, vol. 6, 2014.

[23] T. G. Griffin and B. J. Premore, “An experimental analysis of bgp convergence time,” in *Network Protocols, 2001. Ninth International Conference on*, Nov 2001, pp. 53–61.

[24] T. G. Griffin and G. Wilfong, “An analysis of bgp convergence properties,” *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 277–288, Aug. 1999. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/316194.316231

[25] J. Mayer, P. Mutchler, and J. C. Mitchell, “Evaluating the privacy properties of telephone metadata,” *Proceedings of the National Academy of Sciences*, vol. 113, no. 20, pp. 5536–5541, 2016.

[26] M. Allman, V. Paxson, and W. Stevens, “Rfc 2581: Tcp congestion control,” 1999.

[27] J. Postel *et al.*, “Rfc 791: Internet protocol,” 1981.

[28] N. Spring, R. Mahajan, and D. Wetherall, “Measuring isp topologies with rocketfuel,” in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM ’02. New York, NY, USA: ACM, 2002, pp. 133–145. [Online]. Available: http://doi.acm.org/10.1145/633025.633039

[29] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, “Traffic engineering with mpls in the internet,” *IEEE Network*, vol. 14, no. 2, pp. 28–33, Mar 2000.

[30] P. Agarwal and B. Akyol, “Time to live (ttl) processing in multi-protocol label switching (mpls) networks,” Tech. Rep., 2002.

[31] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, “On selfish routing in internet-like environments,” in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM ’03. New York, NY, USA: ACM, 2003, pp. 151–162. [Online]. Available: http://doi.acm.org/10.1145/863955.863974

[32] Y. Li, Y. Zhang, L. Qiu, and L. Simon, *Smart Tunnel: A Multipath Approach to Achieving Reliability in the Internet.* Computer Science Department, University of Texas at Austin, 2006.

[33] I. Coonjah, P. C. Catherine, and K. M. S. Soyjaudah, “Experimental performance comparison between tcp vs udp tunnel using openvpn,” in *Computing, Communication and Security (ICCCS), 2015 International Conference on*, Dec 2015, pp. 1–5.

[34] (2016). [Online]. Available: https://www.planet-lab.org/