Evaluating Local Proactive Recovery Schemes for IP Networks

Master thesis

Tommy Andre Skancke Nyquist

15th May 2007
Abstract

The Internet is nowadays used for a diverse set of services. Some of them have stringent demands regarding latency, data rate, jitter, etc. Some examples of applications which make such demands are Voice over IP, IP TV, telemedicine, stock exchange information, gaming, etc. In general the diversity of tasks being solved by using the Internet as the communication medium continuously increases the demand availability of connections which is the focus of this thesis.

This thesis focuses on discussing and evaluating recovery schemes that fit the IP Fast Reroute Framework (IPFRR) developed by the IETF Routing Area Working Group. IPFRR is a good framework for recovery schemes that provide fast reroute in connectionless IP networks. Two recovery schemes have been evaluated, namely IP Fast Reroute Using Not-via Addresses (Not-via) and Failure Insensitive Routing (FIR).

The concept of the Not-via recovery scheme is that whenever a link fails, it will consider the neighboring node as down. It will then find a path towards the next-next-hop towards the destination from the path in the failure free case.

The FIR scheme on the other hand is able to infer network failures by looking at the flight of a packet. The flight of a packet refers to the path it takes through the network. Based on this the scheme is able to proactively create forwarding tables that makes sure that traffic never traverses failed network elements.

To be able to compare performance of the recovery schemes in both real life and synthetic networks, a routing simulator was developed. It was used to show that for most networks, FIR will provide shorter recovery paths than Not-via. Since longer recovery paths leads to more links being used for recovery traffic, Not-via will introduce a larger amount of load than FIR. However, an important feature of the Not-via scheme is that it is able to recover from both node and link failures, whereas the FIR scheme is only able to recover from link failures.

It has found that IP Fast Reroute using Not-via addresses would probably be a better choice if a scheme should be implemented in hardware since it has less requirements for doing so. This scheme also has better coverage than the Failure Insensitive Routing scheme and as such would be a better choice for any network operator.
Acknowledgements

I would like to thank some people for making this thesis possible.

First of all, I would like to thank my supervisors Amund Kvalbein and Audun Fosselie Hansen who have been both very inspiring for me when I wrote my thesis. They have also been very fast in giving me good feedback. I must also say that they have been extremely patient and I am thankful for that.

I would also like to thank my head supervisor Tarik Ćićić for his role in making my thesis become a reality. He has always been positive and have given me good feedback.

I would also like to thank good friend and fellow students Ole Kristoffer Apeland for the good discussions we have had regarding the topics of my thesis. The same goes for Johannes Oudenstad who have been my partner in crime when writing a master thesis was not the most exciting thing to do.

From my employer Birdstep Technology ASA I would like to thank Leif Hansson for his good ideas on the evaluation. I would also like to thank Kevin Flynn for helping me correct mistakes in my English.

I also want to thank my family. They have always been positive and inspired me to finish my master thesis even though I have started working.

My greatest thank goes to my beloved wife Anne Marie Skancke Nyquist who has inspired me and kept me going whenever I felt times were tough.
Contents

1 Introduction .............................................. 1
   1.1 History ............................................. 1
   1.2 Motivation and focus ............................... 2

2 Problem statement ....................................... 5
   2.1 Organisation ........................................ 5

3 Background ............................................... 7
   3.1 IP network operation ............................... 7
      3.1.1 Collecting information .......................... 8
      3.1.2 Topology dissemination ......................... 8
      3.1.3 Route calculation ................................ 10
      3.1.4 Forwarding ...................................... 12
      3.1.5 Connection oriented ............................. 13
      3.1.6 Connectionless .................................. 14
      3.1.7 Connection oriented vs. connectionless .......... 15
   3.2 Recovery ............................................ 17
      3.2.1 Motivation ...................................... 17
      3.2.2 Recovery strategies ............................. 18
      3.2.3 Overview ....................................... 22
      3.2.4 Recovery and reversion cycles .................. 22
      3.2.5 Problems with normal IP recovery ............... 25
      3.2.6 Problems recovery schemes tries to solve ....... 26

4 Recovery schemes focused on in this thesis ............. 29
   4.1 IP Fast Reroute Framework ........................... 30
      4.1.1 ECMP ........................................... 30
      4.1.2 Loop Free Alternates ............................ 32
   4.2 IP Fast Reroute Using Not-via Addresses ............. 32
      4.2.1 Concept ......................................... 32
      4.2.2 Overview ....................................... 33
   4.3 Failure Insensitive Routing ........................... 37
      4.3.1 Concept ......................................... 37
      4.3.2 Overview ....................................... 39
   4.4 Comparison of Not-via and FIR ....................... 45
      4.4.1 Coverage ........................................ 45
      4.4.2 Link/node errors ................................ 46
      4.4.3 Last-hop ........................................ 46
CONTENTS

4.4.4 Recovery path endpoint ........................................ 47
4.4.5 Topology layout requirements ................................ 48
4.4.6 Hardware/software requirements ................................ 48
4.4.7 Complexity .................................................... 49
4.4.8 Recovery path identification ................................... 49

4.5 Other schemes .................................................... 50
4.5.1 Reactive IP recovery schemes ................................ 50
4.5.2 Proactive IP recovery schemes ................................ 50

5 Method ............................................................. 53
5.1 General ........................................................... 53
5.2 Routing simulator .................................................. 54
5.2.1 Capabilities and technical information ......................... 55
5.3 Choosing topologies .............................................. 57
5.3.1 Topology characteristics ....................................... 57
5.3.2 Real topologies used ........................................... 58
5.3.3 Generated topologies .......................................... 61
5.3.4 Overview of chosen topologies ................................ 62
5.4 Choosing Traffic Matrices ....................................... 63
5.4.1 Generating Traffic Matrices ................................... 64
5.4.2 Mapping traffic load levels to source-destination pairs ...... 65
5.4.3 Initialization of capacity ...................................... 67
5.5 Scheme implementations ......................................... 67
5.5.1 IP Fast Reroute Using Not-via Addresses ..................... 67
5.5.2 Failure Insensitive Routing ................................... 68

6 Evaluation ......................................................... 69
6.1 General ........................................................... 69
6.1.1 Schemes put to the test ....................................... 69
6.1.2 Test types ..................................................... 69
6.1.3 Regarding test output .......................................... 70
6.2 Path length of affected paths .................................... 70
6.2.1 Hop count of real topologies (flast cost) ...................... 72
6.2.2 Path length stretch of real topologies ......................... 74
6.2.3 Hop count across multiple synthetic topologies ............. 76
6.3 Link load .......................................................... 79
6.4 Total load ........................................................ 81
6.5 Path choices ...................................................... 83
6.5.1 Path choices of local optimal scheme and FIR ............... 83
6.5.2 Path choices of local optimal scheme and Not-via .......... 85
6.5.3 Comparison of Not-via and FIR path choices ................. 86

7 Summary and further work ......................................... 87
7.1 Summary .......................................................... 87
7.2 Further work ...................................................... 89

Bibliography .......................................................... 90

List of Figures ......................................................... 96
CONTENTS

List of Tables 99
A Glossary 101
Chapter 1

Introduction

1.1 History

The Internet a gigantic collection of interconnected computer networks, and it is publicly accessible worldwide. It is based on packet switched transmission of data and it uses the standard Internet Protocol (IP). All types of networks form the Internet, such as academic, corporate, military and governmental. One of the reasons why the Internet is as big as it is today is that it is designed to exist in a heterogeneous environment, and heterogeneity is inevitable since multiple types of hardware and software exist. In addition, the Internet Protocol can be run over almost any link-layer protocol and almost any service can run on top of IP.

The architecture of the Internet has evolved in an evolutionary way, rather than from a master plan. Since the early days of the ARPANET, it has been constantly developing and changing. Principles that some years ago seemed inviolable are today deprecated. Sacred principles of today may be deprecated tomorrow. However, some design principles of new protocols have generally been followed, such as the ones specified in “The design philosophy of the DARPA internet protocols” [12] and “Architectural Principles of the Internet” [11]. Even so, the fact that the use of Internet is ever changing is maybe the only principle that will survive indefinitely. A good analogy for the development of the Internet is a city with its individual streets and buildings. Instead of razing the whole city whenever change is needed, we tend to incrementally replace houses, increase size of residential areas, build bigger roads, etc.

In the early days, the Internet was mostly used for scientific purposes, but during the last decade it has been evolving into a large enabler for both business and recreational services. The number of services accommodated by the Internet is ever increasing and some of the services have stringent demands for network reliability. In the beginning there were services such as e-mail and the World Wide Web which were typical client-server applications with request-response functionality. Sending e-mail was much faster than sending a letter by mail as it was only a matter of seconds or minutes for an e-mail to reach its destination. However, these services have evolved and some have been forked into new
services. E-mail is today considered by many young people to be slow, as they are used to chat using instant messaging software.

However, the Internet is nowadays used for many more things than World Wide Web services and e-mail. Some of them also have stringent demands regarding latency, data rate, jitter, etc. Some examples of applications which make such demands are Voice over IP, IP TV, telemedicine, stock exchange information, gaming, etc. One broker losing his connection to a stock exchange can be disastrous for his company and its customers, even if the outage only lasts for a few minutes or even seconds. And in the case of telemedicine, it would be terrible if during an operation, the surgeon lost contact with the external expert he had a video conference with. And if surgery is being directly controlled by an external person then a failure may have fatal consequences. Another example might be a broadcasting company streaming live video from a sports event over the Internet to massive numbers of users. A loss of IP connectivity to all its customers could be a financial disaster.

In general the diversity of tasks being solved by using the Internet as the communication medium continuously increases the demand for data rates and availability of connections. This thesis will focus on the latter.

1.2 Motivation and focus

Computer networks consist of hosts, routers, links and protocols. Hosts communicate with each other by sending packets through the network. Routers forward the packets around in the network based on different characteristics. The routers are connected with each other through links. The protocols to handle routing and forwarding of data in IP-networks are specially designed to work with the Internet Protocol. The theory of routing can easily be related to graph theory: routers are vertices and links are edges in a graph which resembles the network.

A router may be a hardware component specifically designed to do IP routing, or it may even be a computer specially configured to act as an Internet router. When sending traffic from a source towards a destination, a path has to be found throughout the network and the act of finding such a path is called routing. A routing protocol will typically find the shortest path based on some metric.

The information about the path towards any given destination is typically stored in a data structure that is fit for the forwarding mechanism in use for the specific network. A typical way to do this in IP networks is that for each destination, the routing protocol stores information about the next hop, that is, which link to use to send traffic towards a given destination. Whenever a path has been found, traffic may traverse the network. Each node receiving the traffic looks up the next hop in the forwarding information base (FIB) and passes the traffic on to this link. The act of looking up next hops and passing traffic on to the next router is called forwarding.

There is basically two types of forwarding mechanisms, namely the connectionless and the connection oriented. In short, connectionless forwarding is modelled similar to the postal system, where the destination address of a packet
1.2. MOTIVATION AND FOCUS

is looked up at each stop and forwarded towards the final destination. The connection oriented mechanism are more similar to circuit switched system like the telephony system, where a complete path from source to destination are first set up, before traffic can be sent. By design, the IP protocol is connectionless.

Large networks will at some point inevitably experience some kind of error. The reasons are many and there are several degrees of severity. Both physical and logical errors exist. Among the physical errors there are links failing because of cable cuts, satellites may become unreachable because of solar winds, power outages may cause routers to fail, etc. For logical failures you have typically configuration errors, both man-made and routing protocol instabilities. Some error types may cover both physical and logical failures, as for example sabotage. However, not all errors are caused by accidents. Some are also caused by scheduled downtime due to hardware or configuration changes.

Since there are a vast number of failure types it is in practice impossible for network operators to provide measures against failed entities from all types. As some examples, hardware may be worn out, natural disasters leave cities without power, humans make errors, etc. Therefore, network operators typically focus to identify frequently occurring errors, use these to find a set of failure scenarios and then provide measures against those. A lot of research has been conducted on network failures and from these studies it is shown that the two main classes of failures are single-link and single-node failures [22] [29] [28] [34].

If an error occurs in the Internet, the paths must in some way be changed for the traffic to be able to reach its destination. The process of finding new paths for traffic that would otherwise be lost due to failures in the network is called recovery. The theory of recovery can be divided into several sub categories. First of all, recovery schemes can be divided into proactive and reactive. The proactive schemes have calculated a recovery method and path for the entity they are protecting before any error occurs, whereas the reactive method does this after the error has happened and are by nature slower.

Recovery methods may also work in a local or global fashion. In general, local recovery schemes are able to provide means of recovery without involving the whole network, whereas the global recovery methods typically involve several or even all nodes in the network. The different types will be able to provide different quality of recovery and the resulting paths may differ quite a bit depending on the type of recovery scheme. In general, local recovery schemes are faster, but may provide less optimal paths than the global recovery schemes.

To be able to provide high availability for IP networks, recovery methods have to act fast upon changes and failures of components. Proactive methods are typically faster to activate than the reactive ones, and as such they are better to use when fast recovery is important. Many of the networks connected to the Internet are pure connectionless IP networks and as such it is not possible to use connection oriented recovery mechanisms for those networks. Also, since local recovery schemes in general are faster to activate, such recovery schemes may be more fit for fast recovery in IP networks.

The IETF has the Routing Area Working Group (rtgwg) looking into the topic of fast local recovery schemes for connectionless IP networks and have created a draft for this, namely the IP Fast Reroute Framework [55]. The frame-
work provides guidelines for development of IP fast-reroute recovery schemes. Today, there are three central implementations that fit this specification, namely Multiple Routing Configurations (MRC) [27], IP Fast Reroute Using Not-via Addresses (Not-via) [9] and Failure Insensitive Routing (FIR) [31] [40].

The MRC scheme is based on generating multiple configurations of the topology it is used to do recovery for. It uses these configurations to provide recovery for both node and link failures without knowing the cause of a failure. The scheme marks recovered traffic with the correct routing configuration protecting the failed component.

The Not-via scheme is based on forwarding traffic around a failure by setting up tunnels towards the other side of the failure, which is the endpoint of the tunnel.

The FIR scheme does local recovery by looking at a packets flight (path through the network) and is thereby able to identify possibly failed links.

Among these three recovery schemes, the MRC scheme has had the most thorough evaluation. The Not-via and FIR schemes have not had such thorough evaluations and they are therefore evaluated and compared in this thesis.
Chapter 2

Problem statement

The basic problem statement for this thesis is:

Discuss and evaluate different recovery schemes that provide fast reroute in connectionless IP networks

This thesis will focus on discussing and evaluating recovery schemes that fit the IP Fast Reroute Framework developed by the IETF Routing Area Working Group, namely IP Fast Reroute Using Not-via Addresses (Not-via) and Failure Insensitive Routing (FIR).

It will do this by looking into proposed solutions based on different concepts and discuss how the concepts in general fit for being used for fast rerouting in IP networks.

Two important characteristics of recovery schemes are which paths a scheme uses for recovery and how much additional load it introduces to the network, as this has influence on how networks should be designed. This thesis therefore aims to look into how dependent each scheme is on the topology and see for each topology how long the recovery paths are and how large the increase in load is.

It is also important for a recovery scheme to be able to scale to large networks so it can be applied in all types of IP network topologies. An important aspect of scaling for recovery schemes is the complexity of the calculations needed to provide recovery paths. Recovery schemes may also differ in what type of failures they are able to recover from, namely their coverage and they may have different requirements to the hardware or software they are to be implemented on. This thesis therefore provides a functional analysis of the two recovery schemes where these characteristics are covered.

2.1 Organisation

To be able to provide the knowledge listed above, the thesis is divided into four chapters in addition to a summary.
Chapter 2 consists of information that is needed to understand the rest of the thesis. It covers normal IP network operation such as routing and forwarding, recovery mechanisms.

Chapter 3 will take a deeper look into the two recovery schemes focused on in this thesis. It describes the concept of the recovery schemes in addition to provide an overview of how it works. In the end of the chapter is the functional comparison of the recovery schemes.

Chapter 4 will introduce the research methods used. A routing simulator has been developed in order to evaluate the recovery schemes in terms of recovery paths and load increase. The simulator is described in this chapter. The chapter also describes the different topologies and load matrices used for the evaluation and why they are good choices for this type of evaluation. The last part of the chapter contains details on implementation choices to fit the recovery schemes into the framework.

Chapter 5 contains the results of the evaluations of the recovery schemes. The recovery schemes are in this chapter evaluated and discussed based on their performance in the test cases.

Chapter 6 includes a short summary and describes possible future work.
Chapter 3

Background

In this chapter the thesis will describe several basic concepts that are in use in the Internet. In the Internet, an autonomous system is a collection of IP networks under the control of a single entity that present a common routing policy to the Internet, as defined in [19]. The set of routing protocols that are used within such and autonomous systems are referred to as interior gateway protocols (IGP). The focus of this chapter is to describe how paths for sending traffic through the autonomous system is found (routing), how to transmit the traffic (forwarding), and what happens whenever a component fails (recovery). It will describe classifications of recovery mechanisms and which phases a recovery goes through. The last part of this chapter lists related work and examples of recovery schemes, including the ones that this thesis will evaluate.

3.1 IP network operation

Both desktop users and large server setups are considered as hosts or end points in the Internet. The Internet is a gigantic collection of networks, all being able to communicate through one common protocol, namely the IP protocol which is defined in [45]. However, for an end user to be able to browse the web pages of CNN, some traffic has to be sent from the user, through the Internet, and to the webserver hosting the site. Since the nodes are not directly connected, a path has to be found, and the traffic has to be transmitted on this path.

The main function of routers is to find the path towards the destination for every packet. When a router receives a packet not destined for it self, it must forward it towards the destination through one or more of its neighboring routers. The neighboring router will also forward the packet towards the destination, and so will all routers on the path until the destination is reached.

The process of determining routes for where to send packets to ensure that packets reach their final destination is called routing, while the act of looking up routes and transmitting packets is called forwarding. The concept of routing and forwarding is divided into several distinct parts which are all described in this section.
To be able to route packets, each node in the network needs to collect information about the topology local to the node itself. Among this information it will find a list of neighbors and some metric to define the distance to them.

After getting hold of this information the nodes begin to disseminate their knowledge of the topology with other nodes connected to the same network. In IP networks of today, the most widely used algorithms for this are link state and distance vector which both will be described in this section.

After all nodes in the network have all the information they need and have stored this in their Routing Information Base (RIB), they can start the calculation process where the routes for all traffic are determined. The process will result in tables containing information about where to transmit traffic destined for all destinations called the Forwarding Information Base (FIB). The FIB contains destination addresses and which node who should receive the packet next, called the next-hop.

When all destination addresses are stored together with next-hop information the routers in the network are ready to forward packets. However, in some networks it is needed to create logical connections between nodes in the network, and this is called a connection oriented approach. An approach where setting up connections is not needed is called a connectionless approach. Both these forwarding schemes will be described later in this section.

### 3.1.1 Collecting information

To be able to calculate routing tables, each node first of all needs to collect information about the topology local to the node. This process is called neighbor discovery. Every node needs some mechanism to determine which nodes it currently has a link to, and a typical way of doing this is to run a reachability protocol with each of its directly connected neighbors. There exists multiple ways of doing this and one of these is to periodically send packets to a neighboring node. The neighbor then needs to answer before some predefined timeout value. The absence of one or more such answers can be seen as the node or link being down. Other systems again rely on notifications from lower layers about link and/or node status. In addition, some protocols are able to gather information about the link quality, which later on can be used as a cost metric when generation routing tables.

### 3.1.2 Topology dissemination

For all nodes to be able to calculate routing tables, they must have knowledge about the surrounding network. It is therefore important that each node spreads the information it has collected to other nodes. When all nodes aggregate the information they receive, they should be able to create routing tables.

#### Link state

A common way to do topology dissemination is to use the principle of link state. Two protocols are commonly used in IP networks today, namely OSPF [36] and
IS-IS [10]. In link-state protocols every node broadcasts which nodes it has a direct link to, called a link state advertisement (LSA). Together with the list of adjacent nodes, it also sends the cost for each link. This cost is later used for calculating shortest path between nodes. A link state advertisement typically contains information about the node sending it, a list of all direct links the node has, the cost for each link and also a sequence number. The sequence number is increased each time the node has a change in its link state to make it possible for other nodes to identify which link state advertisement is the latest for the node that produced the advertisement. The information from the advertisement is stored in the nodes receiving the information in a link state database. Since all nodes in the network broadcast this information, every node has the same link state database. In Figure 3.1 a network is shown where each edge in the graph represents a bidirectional link with cost 1. The related link state database is shown in Table 3.1 where '-' indicate that there is no link and '1' indicate that there exists a link.

### Distance vector

Not all networks use link state as their method of disseminating topology information, some use distance vector routing protocols, which include RIP [33] and IGRP [20]. Routers running such routing protocols send all or a portion of their routing tables during a routing update to their neighbors instead of their link state.

First of all, all routers are assumed to know the distance to its neighbors.
Once in a while each router sends their routing table to their neighbors. It is not a broadcast message, so this information is not propagated further into the network. When a router has received such an update, it starts its own updating process. As an example, let us say that the distance between the neighboring nodes 1 and 2 is defined as the metric 4. Then node 2 sends an update to node 1 with its complete routing table including the cost for each destination. Node 1 will then for each of the destinations add the value 4 to the cost and compare this with its own routing table. If the cost is lower, it will update its own routing table with node 2 as next hop for that specific destination.

In the nature of this lies the count-to-infinity problem which exists because there is no way for a node to know whether it is on a path published from its neighbors. In the previous example, consider node the node 1 connected to node 2 which is again connected to node 3. If node 1 goes down, node 2 will have no path to node 1. However node 3, not knowing that node 1 is unavailable, will publish that it has a two-hop path to node 1 (which really goes through node 2), which is false. The information slowly propagates throughout the network, in the end reaching a preset value for infinity. Solutions have been proposed for this in [53] where the author proposes to identify cycles and using this information to find self-passages for a node.

3.1.3 Route calculation

After receiving information about the network topology, one has to calculate paths between the nodes. This may either be done in a central server, or the calculation can be distributed to each node. The latter method is typically used in real life IP networks today. For link state algorithms all nodes will have the same view of the whole network, so there is no need for a central server other than to spare the routers for some CPU cycles or to guarantee a consistent topology. Having a single entity to calculate routing tables will introduce overhead since calculated routing tables have to be distributed to every node from the server. It also introduces a single point of failure.

When calculating routes the it is important to find a shortest path tree for each of the nodes towards all destinations. The tree is typically based on some metric for the cost of each link. There exists different heuristics to set the link cost, and some of them are the inverse of the capacity, time delay, total number of packets queued along the path. A typical way of finding shortest paths in a network is to create a shortest path tree with the source router as the root. There may exist more than one such tree (i.e. more than one path with the same cost), and routing protocols try for each node to find one such tree. The information in the shortest path tree is used to find the next hop towards each destination. This next hop is saved in the FIB with a vector with destination and next hop.

In distance vector networks one does not have a view of the whole network and therefore the distributed routing tables are used directly and the calculation is only to find minimum cost for all destination nodes amongst all the received routing tables. The calculation is done while the node is receiving the table. It chooses the next-hop that gives the minimum distance to the node it wants to find a shortest path to. But when using link state protocols, all nodes have
a global view of the network and it is then possible to calculate the shortest paths at each node. A common way to do this is to use Dijkstra’s algorithm or a modified version of it.

**Dijkstra’s algorithm**

Using Dijkstra’s algorithm [13] will find one shortest path tree. The way this algorithm works is that in its initial state it marks the source node with a permanent label with distance 0. All other nodes are marked with a tentative label and distance infinity. Then the algorithm follows the links from the source to its neighboring nodes and labels them with a tentative label with the distance from the source node and it is also marked with the previous node so that we later can reconstruct the final path. Then it goes into a loop with finding the node with the shortest distance and marks this node with a permanent label. Then the algorithm follows all links from this node and marks its neighboring nodes with tentative labels with the new distance and itself as the previous node. When reaching a node which already has a tentative label with a smaller distance, the label is not changed. When in the end all nodes have been marked as permanent, we have all the information to construct a shortest path tree following the path from each node to the root of the tree.

**Example**

A popular link state routing protocol today is Open Shortest Path First [36]. This protocol is used in both pure IP networks and networks based on MPLS [51].

When using a link state routing algorithm it is crucial for all routers to synchronize the link state database. If they are not synchronized it may lead to packet drops and loops. This will be explained in detail later. OSPF simplifies this by requiring only adjacent routers to remain synchronized. To maintain the database of neighbors OSPF uses Hello packets. These packets are sent out on all links of the node. Since adjacent nodes also send out such packets the OSPF protocol finds bidirectional links by finding itself in its neighbors Hello packet. OSPF was derived from an early version of the IS-IS protocol [10], which is another link state protocol.

The network in Figure 3.1 is running as a connectionless network (see Section 3.1.6). Node A could use the link state database shown in Table 3.1 to calculate the shortest path tree shown in Figure 3.2. Based on this tree, node A calculates which node it should forwards packets to reach their destinations. Only nodes directly connected to node A, its children, may be chosen as next hops. These next hops are saved in a routing table. The corresponding table for this example with node A is shown in Table 3.2.

As an example, if node A wants to send a packet to node I, it looks up the destination I in the routing table, and finds that it should transmit the packet to node B. The process of looking up destinations and forwarding to the next hop is called forwarding, and is described below.
CHAPTER 3. BACKGROUND

Figure 3.2: Shortest path tree from the network in Figure 3.1 with node A as source

Table 3.2: Sample network 1 - routing table for node A

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>B</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>D</td>
</tr>
<tr>
<td>I</td>
<td>B</td>
</tr>
</tbody>
</table>

This thesis will focus on link-state routing protocols, as it will evaluate schemes that only work well in link-state networks.

3.1.4 Forwarding

After a route is found, we want to transmit the data packet from the source node through the network to the destination node based on the routing tables. The process of looking up the destination in the routing table and send the packet out on the correct link is called forwarding. There exist two basic models of doing packet forwarding or packet switching.

One way is to use virtual circuits from the source node to the destination node. All packets destined for that specific destination follows that predefined path. The other way is to use destination based routing where all packets are routed independently. All routers on the path do a lookup of the destination address in the packet to find the next hop and transmit the packet on the correct outgoing link.

The former method is called connection oriented and the latter is called connectionless. They are both described below, and a short comparison of the two is given.
3.1.5 Connection oriented

The connection oriented service is modelled somewhat like a telephone system. To talk to someone you pick up the phone, dial the number, get a connection, talk with the person in the other end, and then you hang up. Similarly when using a connection oriented service, one establishes a connection, uses the connection before in the end tearing down the connection. In the telephone example, when a phone conversation is set up, a physical line is dedicated for this purpose and this is therefore called a circuit switched technology. The same principle may also be used in optical networks, for example in the form of Wavelength Division Multiplexing (WDM) together with General Multi-Protocol Label Switching (GMPLS). Ordinary MPLS, which is not circuit switched, but packet switched, is described later in this section.

To be able to use a connection oriented service in a packet network, one must first set up a path through the network before any data packets can be sent. The resulting path is often called a virtual circuit \[58\] in analogy to physical circuits set up by the early phone system.

To establish virtual circuits a signalling protocol must be used to propagate information about the path to all affected nodes. A typical signalling protocol is RSVP \[7\] which is often used for connection oriented networks such as MPLS. There are basically two ways of choosing the path to signal. One way is to use so called source-based routing where the source router defines the path for the virtual circuit based on its view of the network. Information about the complete path is then sent to all the affected nodes. But source based routing is only possible when the source router has a complete view of the network, and thus not applicable for distance vector networks. The other way is for the source router to use its next-hop table and transmit the signalling packet to the next router. This router again looks up the destination address in its routing table and transmits the packet accordingly and this continues to the final destination of the path.

When the source router uses the path to transmit data, it first finds the correct path for the packet and marks the packet with a path identifier, referred to as a label. This is so all routers on the path can know where to send the packet. When the packet reaches its destination it may be that the label has to be removed from the packet, depending on how marking of packets are implemented.

When the connection is released, the virtual circuit is torn down, typically by using the signalling protocol to inform all routers on the path to remove their knowledge of the path.

Example

Multiprotocol Label Switching (MPLS) \[50, 51\] is an example of data-carrying mechanism based on label switching. In the OSI-model it is operating between the Link-layer and the Network-layer. It was designed to provide a data-carrying service for both circuit-based and packet-switched clients by providing a datagram service model.
As an example, one can look at the network in Figure 3.1. If node C wants to establish contact with node G then the path has to be set up. Node C sends a path message through node F and H to node G, when G replies to node H with a free label. Node H then replies node F with a free label and stores a record in the label switch table, for example: "IN: F, LABEL: 1 \rightarrow OUT: G, LABEL: 1". Then node F replies node C with a free label and stores a record, for example: "IN: C, LABEL: 1 \rightarrow OUT: H, LABEL: 1". But consider now that node I also wants to contact node G. Just as node C, node I transmits its path message through the network to its destination G. Since F already has a label for the destination G, it can reuse the label and merge this traffic. The same goes for node H. The corresponding tables are shown in Table 3.3 and 3.4.

### 3.1.6 Connectionless

In contrast to connection oriented forwarding, the connectionless service is modeled somewhat like the postal system. Each packet contains the complete destination address and is routed through the system independent of other packets. There is no connection setup phase, in contrast to the connection oriented methods. Such connectionless data carrying mechanisms are often called destination based forwarding, since the packets are forwarded based on their destination address.

Ordinary IP-routing is connectionless which means that the sender does not know which route the packet will take during its transmission through the network. This is decided at each router on the path, based on the routing algorithm used.

#### Example

We can again look at a scenario based on Figure 3.1 where C wants to send a message to G. The interesting part of the forwarding tables for the nodes involved in the forwarding is shown in Table 3.5.

First C looks up in its routing table how to get to G. Based on the routing protocol used it may have B, F or both as next-hops. Lets assume it has F as its next hop and it forwards the packet to F. F then looks up G in its routing
Table 3.5: Sample of forwarding tables for nodes C, F, H and I

<table>
<thead>
<tr>
<th>Node</th>
<th>Destination</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>G</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>I</td>
<td>G</td>
<td>F</td>
</tr>
</tbody>
</table>

table, finds H as the next hop, and forwards to H. H then looks up node G in its routing table and finds that is has a direct link and transmits the packet to its final stop.

If now node I wants to contact node G this packet will be routed independently of any other packet. So node I looks up G in its routing table, and finds F as its next hop so it forwards the packet to F. Node F does a lookup to find the exact same record for the destination G, which is node H and forwards the packet. Node H again finds that is has a direct link to G and forwards the packet.

### 3.1.7 Connection oriented vs. connectionless

Since the nature of the connection oriented and connectionless data-carrying mechanisms are very different, their features are of course not equal. This section will introduce some of the differences.

#### Quality of Service

From a Quality of Service standpoint, Internet Service Providers (ISPs) will more easily be able to manage different data streams based on priority and service plan by using a connection oriented method. However, research has been done regarding QoS in connectionless networks as well, and some proposals are listed in [63].

QoS is typically important for ISPs regarding high paying customers, or customers that have high demands regarding bandwidth, jitter, packet loss etc. Such customers can often benefit from a connection oriented network specifically designed to meet such requirements since it is possible to guarantee QoS throughout the whole virtual circuit whereas connectionless mechanisms can only provide QoS on a hop by hop basis.

However, both mechanisms are able to do QoS in some way based on metrics such as destination address, bandwidth demands, latency demands and differentiated services classification. However, only one lookup is needed for connection oriented (ingress node), whereas connectionless domains must lookup this information on every hop.
CHAPTER 3. BACKGROUND

Setup speed

For connection oriented network it is needed to setup a circuit, which at least takes as long as the round-trip time (RTT), since one has to create a path from the source router to the destination router. So for a short stream of data, it may be faster to use a connectionless data-carrying mechanism. However, the use of connection oriented services such as MPLS is often used in the backbone of the network, where the LSPs are typically already set up by network administrators, and thus there is no need for fast and dynamic generation of paths throughout the network lifetime. Regarding connectionless networks, naturally no time is used for setting up paths.

Forwarding speed

Algorithms for label switching are generally faster in comparison to do a longest prefix match on the destination address in the routing table. So as soon as a path has been setup it is quick to do lookups and use label switching. But today the routing speed of IP-routers (which are connectionless) is the same as the link speed, so in practice it is no difference in speed for lookups.

Another feature regarding routing speed for connection oriented networks is that a virtual circuit can be created with very much information, such as which protocol should be used, what type of service it should receive and so on. In addition this classification is only done in the edge of the network. Connectionless routers however, would have to lookup this information in each packet that arrives before forwarding it and all routers on the path would have to do this.

Traffic engineering

For ISPs it is important to control the traffic flowing through its core network. The subject of controlling network traffic is called traffic engineering. With connection oriented networks it is often more easy to do traffic engineering since it is possible to setup different paths for different types of traffic and for different destinations. As an example MPLS is typically used by ISPs to set up VPN connections to connect large corporations with several offices. Virtual circuits could also be used to reserve data rate capacity on all the hops of a path.

Traffic engineering features have also been researched for connectionless networks. In [15] they propose to calculate appropriate costs for each link based on already known demand matrices. Often, in backbone networks of ISPs, the demand matrices for different times of the day are generally known, and the authors propose to utilize this to make clever choices of costs for each link.

An advantage of connectionless mode over connection oriented is that it allows for multicast and broadcast of messages. This may save network resources if the same message is to be sent to multiple recipients. In contrast, the connection oriented mode is generally unicast. However, MPLS provides functionality for point-to-multipoint communication.
3.2. RECOVERY

State information

In [12] it is stated that in the Internet architecture it was chosen to store state information in the endpoints of the networks, and if an endpoint was disconnected, all its state information would be lost, called fate-sharing. However, in connection oriented networks, state information is also stored in the core of the networks, since routes are preinstalled, in contrast to the design philosophy of the Internet.

As pure IP networks are connectionless, this thesis will focus on recovery schemes which are designed for connectionless networks.

3.2 Recovery

Even though network stability is something that many end users take for granted, networks are inherently prone to failures, and in most networks they occur on a regular basis. This section will give you an idea of how often errors happen, and what kinds of errors that typically occur. The introduction to network failures will motivate the use of recovery mechanisms in networks, and this will be thoroughly described later in the section.

3.2.1 Motivation

Failures in networks have been subject to a lot of research. There has been several analyses made of existing networks and in [22] failure patterns in the IP backbone of Sprint was analysed. Their results show that 20 percent of the links had a Mean Time Between Failure (MTBF) of less than 1 day and 70 percent had a MTBF of less than 10 days. The reasons for failures are different and they have various scales and severities. Both physical and logical failures can occur in networks and they can be divided into external and internal causes. One of the most understandable examples of an external physical error is a cable cut. We have all heard of unlucky excavators distorting both water pipelines, phone lines etc. On the logical part there may be configuration errors, both man-made and the result of protocol instability. But in fact, not all errors are accidents, but quite a large portion of the errors are the result of preplanned service events such as hardware upgrades or large configuration changes. This has been studied in [29] (technical report in [28]) and [34], and it is stated that the single most common cause of failure is scheduled maintenance. This may be configuration installation or changes, upgrades etc. Among other significant contributors to network failures they found power outage, link failures and router failures. If aggregated, the unscheduled failures contribute to about 4/5 of all failures. In addition it is stated that 85 percent of unplanned failures affect only a single link and 46 percent can be considered transient since they only last for less than one minute. As much as 86 percent last less than ten minutes. Hence, effective handling of transient link and node failures is a key for ensuring high service availability.
In practice it is impossible to provide measures for all types of failures in communication networks. The hardware can wear out, be overloaded, humans may make mistakes, and there may be software bugs and even intentional failures such as sabotage. In addition, the impact of dramatic failures due to for example natural disasters such as earthquakes are simply too great, and other errors may be too costly to protect against. Thus most ISPs will instead focus to identify frequently occurring errors, classify these in a limited set of failure scenarios, and then create cost-effective recovery methods for such errors.

Two main classes of failure scenarios types are the link and node failure. The link failure scenario is a situation where the connection between two neighboring nodes fails. This means that no traffic can traverse this specific link. The other scenario with node failure occurs when a node or router fails and thus no communication can be transmitted where this node is included in the path. Such an error can occur by for example power outage, hardware failure, software crash, etc. and it will also put the attached links out of service. However, it is not always that the neighbors will be able to see that the link in fact should be considered dead, since lower layer mechanisms may report that the link is up and running. In addition, it may be difficult for a node adjacent to a failure to identify whether it is a link failure or a node failure.

Among link and node failure scenarios, a single failure at any time is the most likely scenario, based on the assumptions of [59]:

- In most cases, the failure of one link is statistically independent of the failure of another link or node in the network, at least when assuming that dramatic outages from such as natural disasters are considered unlikely.
- If the network scale is not too large, the Mean Time To Repair (MTTR) for a single link or node failure is often much smaller than the Mean Time Between Failures, i.e.: \( MTTR << MTBF \) Hence, the likelihood of two failures occurring at the same time can be neglected in comparison to the probability of a single-link or single-node failure.

However, a failure in an IP network could be caused by a failure in lower levels such as a cable cut which will cause connection disruption on the physical layer. Since one physical cable may be used for several logical network layer links, such a failure may lead to the failure of more than one IP link since the network layer does not know the underlying physical architecture. To model such scenarios, the IETF has defined a concept of Shared Risk Link Groups (SRLG) [47] and a more general Shared Risk Group (SRG). An SRG is a way of pointing to resources that are all affected by the same cause. In contrast to single-link or single-node failure scenarios where all errors are statistically independent of each other, the SRG concept expresses a statistical dependency between failures of individual links or nodes.

### 3.2.2 Recovery strategies

To be able to recover from errors in the networks, there are several strategies to be applied. They can be divided into two main strategies: Either there exists a
3.2. RECOVERY

backup solution before any error happens, or a backup solution must be created whenever an error occurs. The former proactive way is called protection and the latter reactive way is called restoration. In addition both strategies are divided into local and global recovery, which are described later in this section. After the network operators have decided which strategy to use for recovery, there are several steps to occur as soon as an error in fact happens. Basically, first of all one needs to detect the error. After the detection it is sometimes necessary to inform other nodes about the error. When all necessary nodes have been informed, some change will have to be implemented in the network to cope with recovery. This is often by changing part of the routing tables.

 Protection

Protection means that the routers in the network proactively have calculated a backup solution to use when a failure occurs in the network.

As an example, in a virtual circuit network this is possible by establishing two disjoint paths between two nodes, one of which is the default path and the other the backup path. Whenever an error occurs on the default path, all traffic is switched over to the backup path. This method is for end to end protection, but it is also possible to implement this strategy as a local protection.

 Restoration

Restoration means that the routers are using a reactive method and begins the process of finding a new path excluding the point of failure after an error has occurred.

As an example, the link state IP protocols such as OSPF use restoration. When an error occurs, information about the error is disseminated to the other nodes, and each router finds a new route around the problem. This is called IP reconvergence and typically takes several seconds.

 Protection vs. restoration

Protection is typically implemented on connection oriented networks and often has faster error correction than restoration. But this means that extra information has to be in the memory of the routers. Regarding restoration it often takes some time to solve problems using this method, but it is often easier to implement, so it is heavily used in today’s networks. As recovery information is calculated on the fly, restoration based schemes will typically have lower demands for router memory.

Today, users of the Internet have constantly increasing demands for stability of the network. Since using a protection strategy inside a network will result in faster error handling, users with real-time demands such as video conferencing and voice over IP will experience a better Quality of Service using this strategy.
CHAPTER 3. BACKGROUND

Since protection schemes are generally faster to activate than restoration schemes, this thesis will focus on connectionless recovery schemes which are designed to be proactive.

**Local**

In local recovery schemes, recovery is initialized by the node discovering the error and no information needs to be published to other nodes about the failure.

This will usually ensure a solution which is applicable pretty fast, but it may lead to loops. Let us say we have the network in Figure 3.1 and node C wants to send a packet to node A. But suddenly the link between node B and node C goes down as shown in Figure 3.3. This means that it has to route the packet some other way. If it were to decide this for itself, it could send the packet through the path F-E-D-A or F-H-G-A. Either way, this includes node F. But let us now say that node F not yet knows that the link B-C has gone down and it has F-C-B-A as a its shortest path. So when node F gets the packet from node C, it will send it through node C, which then again sends the packet back to node F and the loop is complete as shown with numbered forwarding in Figure 3.3. This is since not all actions in a network are happening simultaneously and this may lead to differences in routing tables between routers which again may lead to such transient loops. However, after a while node F and the rest of the network will get information about the link that is down, and the network can again enter a consistent state. In [16] they propose a solution to this problem with microloops by ordering the FIB updates on the routers.

An example of local connection oriented is shown in Figure 3.1 and on the shortest path tree for node A shown in Figure 3.2. If node A should have protection for the link A-B going down it may have to have backup paths for all paths that would usually use that link, depending on the recovery scheme used. This does not scale very well, any may create a substantial amount of backup paths, as is shown in Figure 3.4. However, it is usually possible to stack labels, and thereby fixing this problem with a single backup path from node A to node B.

**Global**

Global recovery is different for connection oriented networks and connectionless networks. In connection oriented networks such as MPLS a node detecting an
3.2. RECOVERY

Figure 3.4: Connection oriented local protection with A-B link down

Figure 3.5: Connection oriented global protection between A-F

error sends an error message back to the ingress node and this node finds a new route. The rerouting process of connectionless link state IP networks includes broadcasting link state updates in the network, an IP reconvergence. After a while, the paths in the network will change and the point of failure is omitted. However, in both cases, packets will be dropped until a rerouting occurs.

As an example, connection oriented networks can achieve global protection by creating additional paths from the source to the destination. Such an example for default and backup path is shown in Figure 3.5 which is from node A to node F in the network shown in Figure 3.1. It is preferable to let the backup paths be disjoint from the default path. The dotted line is the default path, while the dashed path is the disjoint backup path.

Local vs. global

Global recovery will often take some time to initiate, both for connection oriented and connectionless networks. In connection oriented networks, the node upstream of the error has to inform the ingress node that the error has occurred. This is opposed to local recovery where there is no need with a message exchange with the ingress node. In connectionless networks, global restoration will typically be used as a complete reconvergence has to take place link state IP networks where.

A global recovery scheme may construct more optimal paths for traffic, depending on how it is constructed, since it may route packets on a path that does not include the failed component all from the source. Whereas in local
schemes, packets are first sent towards the point of failure, before it is rerouted to a backup path. However, many global recovery schemes aim to find two or more disjoint paths and as such they do not guarantee the shortest path for recovery paths.

### 3.2.3 Overview

In Table 3.6 a short overview over different available recovery mechanisms is given. In general, if a connection oriented approach is chosen to do recovery, MPLS is the typical choice for IP networks. However, if for optical networks WDM may be used. And if an IP network is run on top of ATM, the recovery mechanisms of ATM may be used. The connectionless methods are described under Section 4.5.1 for reactive and Section 4.5.2 for proactive schemes.

This thesis evaluates recovery schemes which are defined as local, proactive schemes, which fit into the design of the IP Fast Reroute framework. More about this in Section 4.1.

### 3.2.4 Recovery and reversion cycles

During the lifetime of a network, several faults can occur, and when it does, the recovery cycle has to start. When a node knows about an error it can take many actions. First of all it can try to fix the error itself by sending packets some way around the problem. The node can also inform others about the error that has occurred and wait until data traffic starts using other paths. It can also first fix the error locally, and see for a little while whether the error persists. If the error persists, it can inform other nodes about the error. The node can also send the packets back to the source and make that node decide what to do next.

Which actions to be taken to perform the recovery operation depend on the specific recovery scheme used, but they all have phases they cycle through, and this section describes which phases that happen from the failure occurs until the traffic has been rerouted, called the recovery cycle [56]. However, after the traffic has been rerouted, the error may be repaired. For example, a cable cut may be mended, a configuration error may be fixed, etc. The traffic should then be routed back on the standard path, following a reversion cycle.

This thesis follows the steps described in [56] which is a framework for MPLS recovery. Even though MPLS is connection oriented, the same steps will typi-
3.2. RECOVERY

The recovery cycle shown in Figure 3.6 has five time intervals or phases and they are all explained below in the order they occur in a recovery operation.

Fault Detection Time  As previously mentioned in this section, errors are prone to occur in networks, and this phase covers the time between the network impairment arises and until the error has been detected by a node in the network. The time between an error occurring and its detection may depend on, for instance, the frequency of signals sent. As an example some routing protocols frequently exchange messages such as state information, explicit hello, ping or echo messages etc. In addition there exists routing protocol independent protocols, such as Bidirectional Forwarding Detection (BFD) [25] which establishes a session between two endpoints and in the asynchronous mode of BFD, endpoints periodically send Hello packets to each other. If a number of such packets fail to arrive, the intended receiver can then consider the session dead. The main advantage of BFD over faster hello messages is that it is possible to implement BFD on the linecards themselves, thus making it possible to use shorter time limits and without consuming a lot of CPU resources. Fault Detection Time may also be dependent on fault detection capabilities in lower network layers, notification times towards upper layers, the time it takes to gather information about abnormal signals and derive exact fault state etc. As an example, when using SDH and SONET it is possible to achieve failure detection times of less than 10 milliseconds by using signalling from the physical layer [59].

Fault Hold-off Time  The Hold-off Time is defined as the time between the detection of a fault and before the node takes an action. The reason for such a waiting time is for instance that during this time, lower layer protection scheme may go into effect. As an example, in an IP network supported by an optical
transport network, a cable cut could quickly be repaired using optical recovery mechanisms and the IP link would then become operational again after a short time. If the link is not operational after the defined threshold, proper measures could be taken, such as entering the next phase, the Fault Notification Time. However, in some recovery schemes, the Hold-off time is set to zero, and in such schemes the algorithm goes directly to the next phase. The value of the Hold-off Time may be either dynamically calculated or a static value. In the former case, the value may be an increasing timer based on the number of failures that has occurred recently. The more errors, the longer the Hold-off Time will be. This technique is called dampening [59] and it helps with stabilizing the network in cases where the status of the network flaps between operational and non-operational. It is also possible to have Hold-off Times on both the physical network and in the routing protocol.

**Fault Notification Time** The time after the Hold-off Time is the Fault Notification Time, which is the time it takes to notify the nodes needed to start recovery. In connection oriented networks this could be the node responsible for the virtual circuits passing through the failed network element. In local protection schemes for connectionless networks this is typically the node that identified the error, so the Fault Notification Time will in such cases be zero and this is the case for the schemes investigated in this thesis.

**Recovery Operation Time** The Recovery Operation Time is defined as the time between the first and the last recovery operation. This includes all messages sent between nodes needed to coordinate the operation. But this is not the same as the overall recovery time, which covers all phases. As an example, the Recovery Operation Time when fixing errors in link state routing algorithms such as OSPF includes the time where the link state table is updated and a new shortest path tree is calculated based on the new table. After this the routing table is created from the first hops of the new shortest path tree and this is then typically pushed down to the linecards on the router. In schemes this thesis focus on no explicit message is sent between nodes, so this phase may for example be the time it takes for the node discovering the error to change its forwarding tables.

**Traffic Recovery Time** After the last recovery action, the traffic is routed on the new recovery path. However, it could still take some time before all traffic is completely recovered. The Traffic Recovery Time takes into account the time it takes for traffic to once again arrive at the point in the network that experience disrupted service due to the occurrence of the fault. The length of this phase typically depends on the delay along the path, the location of the fault, and the recovery scheme used.

**Reversion cycle**

When an error has been repaired through the means of recovery schemes the traffic may be flowing on a suboptimal path compared to the path before the
3.2. RECOVERY

Therefore, as soon as an error is repaired, it is desired to switch back to the default routing paths to reduce excessive load in the network because of detours around failed network elements. Typically, a dynamic rerouting protocol may be initiated to optimize the usage of network resources in the new situation. It is also the possibility to wait for the repair, and then redirect traffic from the recovery path to the working path once the failure is completely repaired which is called a revertive technique. The switch-back operation consists of five phases, which resemble the phases in the recovery cycle. Together, they are called the reversion cycle. However, this is not a focus of this thesis, but it is thoroughly described in [56].

In contrast to the recovery operation, the reversion operation is not acting upon an unforeseen event. Instead, it is possible to plan this operation well in advance. There is often no need for a hasty operation, so a well-controlled, synchronized switch-back mechanism is typically preferred, since it will yield minimal disruption. In other cases, where the recovery path is unable to provide a quality of service that is comparable to the preferred path it may be desirable to do a fast reversion.

3.2.5 Problems with normal IP recovery

This thesis focuses on doing recovery in link-state, connectionless IP networks, and as such, there are some problems with normal recovery in such networks. There has been research of this topic, and the main problem is the speed of recovery. New services are emerging which have stringent demands to Quality of Service such as Voice over IP, video streaming etc. The proposed solutions for this can typically be put into two main categories. Either a recovery scheme can focus on optimizing link-state convergence, or it can change either the whole or parts of the recover procedure.

Existing routing protocols today such as OSPF and IS-IS use a global restoration method for handling link failures. One way to cope with faster recovery is to increase the speed of the convergence time for normal IP convergence. This is done by optimizing the phases listed in [3.2.4], such as increasing the number of hello messages sent between nodes etc. However, fast failure handling can make the network unstable and during reconfiguration of the network micro-loops (see Section 3.2.6) may occur. But then again, slow failure handling causes forwarding discontinuity and as such, a trade off between routing stability and fast failure handling must be made.

In link-state networks, it may take quite a while from an error occurs until the network recovers from the failure and again is in a new stable condition. One way of shortening the total time is to shorten different parts of the recovery cycle phases. A group of researches discussed how one could use short time-to-live values for adjacent nodes, called hold times, without compromising network stability [24]. They would then be able to optimize the Fault Detection Time and thereby leading to faster convergence. This Internet Draft [24] has however expired. In [37] the authors propose a way to shorten the Fault Notification Time by only informing a minimal number of nodes who need to know when a link has failed to ensure loop-free routing. This will lead to less flooding of
information whenever a change has occurred, thereby reducing total load of the network and reducing CPU usage on routers.

The whole process of recovery has been studied in [17], and they have thoroughly analysed the factors that influence the convergence time. They do detailed measurements of the different operations of recovery on routers using implementations of link-state algorithms. They then build a simulation model based on the measurements, and use this model to study convergence in large networks. The conclusion of their work is that it is feasible with sub-second recovery times without any compromise on stability.

The other category involves changing whole or parts of the recovery procedure. Such recovery schemes are the focus of this thesis, and they are detailed in Chapter 4.

3.2.6 Problems recovery schemes tries to solve

This section describes two well known problems that authors of recovery schemes may try to solve, namely the formation of recovery loops and the last hop problem.

Recovery loops

There exist basically three different types of loops regarding recovery, and they are all described below.

Micro-loops A problem with IP recovery is the formation of micro-loops. In networks using protocols such as OSPF, when a router detects that a link has gone dead, other routers will be notified during its next link state update. This will trigger a rerouting process that excludes the link that has gone down. These link state updates propagate through the network and trigger the rerouting process. But since not all routers calculate their new shortest path trees simultaneously, instability in the network and transient loops may occur. A looping state may last for several seconds and this state prevent packets from reaching their goal and create excessive load in the network. This problem is not a focus of this thesis, but an Internet Draft introducing a framework to solve this, has been published in [8].

Loop when finding recovery paths A badly designed recovery scheme may lead to loops when forwarding traffic on a recovery path. This may happen for example if a recovery scheme does give enough information to routers which are part of the recovery path. However, as this is a problem the designers know about, it is usually solved before publishing information about a recovery scheme.

Loops when too many failures exist Some recovery schemes try to recover traffic which is already recovered, which may lead to forwarding loops. If for
3.2. RECOVERY

Figure 3.7: Topology to illustrate last hop problem

example a recovery scheme is designed to handle single-link or single-node failures, two failed links at the same time may cause the recovery scheme to send traffic back and forth in the network.

Last hop problem

If a recovery scheme is trying to recover from a link failure inside a network, some schemes consider the node downstream of the failed link as down, and do a node recovery as this may in some schemes be easier to achieve. However, it can only do this for intermediate nodes, and if it always use the same method, it may lead to unreachable destination nodes. E.g. the last link towards the destination, the last hop, fails, which leads to the destination not being reachable.

In Figure 3.7 a part of a larger network topology is shown. The scenario is that the link between node 1 and 3 is dead and node recovery is in use. If node 1 then receives traffic with node 3 as the destination it will not be able to send the traffic. The reason for this is that it will consider node 3 as dead even though it is only the link between node 1 and 3 which is really dead. However, as can be seen in the figure, a path does exist and could be used for recovery, namely the path $1 \rightarrow 2 \rightarrow 3$.

Some recovery schemes solve this problems, and some do not. This topic is covered for the schemes evaluated in this thesis in Section 4.4.3.
Chapter 4

Recovery schemes focused on in this thesis

During the last decade, high availability for IP networks has been a large concern for IP network operators and as such a lot of research has been conducted to find good methods. To be able to provide high availability for IP networks, recovery methods have to act fast upon changes and failures of components. In general, since a proactive recovery scheme have precalculated recovery paths it is faster to activate than a reactive recovery scheme. This makes proactive schemes a better fit than reactive ones for fast recovery schemes.

A recovery scheme can either work in a global or local manner. A local recovery scheme do not have to inform other routers in the network to be able to start recovery, and it is then possible to reduce reaction times by using such schemes.

Also, many of the networks connected to the Internet are pure connectionless IP networks and as such it is not possible to use connection oriented recovery mechanisms for those networks. This means that a recovery scheme should be able to work in a connectionless environment.

This thesis focuses on two recovery schemes. They are both proactive, local, connectionless recovery mechanisms that offer protection of links, nodes or both. As such they do not depend on being able to exchange information with other nodes in the network when discovering an error, since everything is precalculated and preinstalled.

Both recovery schemes fit well into the IP Fast Reroute Framework (IPFRR) and are mentioned in [55]. IPFRR is thoroughly described in Section 4.1.

The first scheme described in Section 4.2 is developed in the IETF Routing Area Working Group and is referred to as IP Fast Reroute Using Not-via Addresses (Not-via). The concept of this scheme is to encapsulate traffic into tunnels that do not traverse failed network elements. They identify such paths by creating specific addresses for each component they want to protect. The semantics of such an address is that routers must route packets to the router
advertising the address, but without passing the protected component which the address is associated.

The second is referred to as Failure Insensitive Routing (FIR) and is described in Section 4.3. The concept of this scheme is to infer network failures by looking at the flight (path) of the packet, and thereby proactively creating paths around failures. They have developed an algorithm that can identify possibly failed links whenever a packet arrives on an interface that is not normal for the packets destination. By having interface specific forwarding tables they are able to exploit this concept to provide link protection.

In Section 4.4 a qualitative comparison of the two recovery schemes is given. The schemes are also thoroughly compared in Chapter 6 where results from the tests are shown.

Section 4.5 lists other proposed recovery schemes.

4.1 IP Fast Reroute Framework

The IP Fast Reroute Framework is defined in [55] and introduces a framework for creating locally determined repair paths for link and node failures. It is heavily inspired by the MPLS Fast Reroute Framework [43], but is suitable for connectionless IP networks. An important aspect of recovery mechanisms implementing the framework is to prevent the loss of packets during reconvergence of the network after an error has occurred. The framework divides solutions into three categories. First it suggests using Equal-Cost Multi-Path (ECMP) paths as backup paths for failed links. These paths ensure that no loops occur. In addition it is stated that in some networks and for some failed links, there may exist nodes adjacent to the node detecting the error which may be considered loop free alternates (LFAs). This implies that if the detecting node sends the packet to this LFA it will never return the packet because it will always have a shorter path to the destination. The authors state that these two solutions together will generate backup solutions for 80% of all failures. However, for the 20% of failure scenarios they do not generate a backup solutions for, the authors introduce the third method, the multi-hop repair path. This section will list one solution to find LFA and four solutions which implement the multi-hop repair path scheme. The following list sums up the three solutions in the IP Fast Reroute Framework:

1. ECMP paths
2. Loop Free Alternates
3. Multi-hop repair paths

4.1.1 ECMP

The first alternative, the Equal-Cost Multi-Path, is thoroughly analyzed in [21] and is a routing technique for routing packets along multiple paths of equal cost, and using these paths as backup paths.
In Figure 4.1 a topology is shown where all link costs are equal. In the network, two shortest path exist from node S to node D, namely path A = $S \rightarrow 1 \rightarrow 2 \rightarrow D$ and path B = $S \rightarrow 3 \rightarrow 4 \rightarrow D$. If ECMP is not in use, an arbitrary choice of path A and B will be used to send traffic. However, if ECMP is enabled source node S sends traffic towards destination node D using both paths. These paths could be used for either load balancing, recovery or both.

If ECMP paths are used for load balancing traffic will typically flow all ECMP paths simultaneously and thereby distributing the load in the network. Typically the load for each path can be expressed as:

$$P_i = \frac{L}{N} \quad (4.1)$$

where $P_i$ represent load traversing each ECMP path, $L$ is the total load of the traffic and $N$ is the number of ECMP paths. When ECMP is used for recovery, two options are available for the node discovering the error:

1. Subtract 1 from $N$ in Equation 4.1 and send load through rest of ECMP paths which are available

2. Recover traffic on failed ECMP path by other means of recovery available

As an example from Figure 4.1 if the link between node S and 1 should fail, node S could send all traffic through path B. However, when using ECMP paths as recovery paths, the load balancing may be hampered. For example, if 50% of traffic was going through path A and 50% through path B, then after the failure 100% of the traffic would go through path B. This may have implications for the load on the links and routers that path B follows. Another way of doing recovery is to send 50% of the traffic as normal through path B and then recover path A using some other form of recovery. In the network shown in Figure 4.1 this could be done by using the path $S \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow D$. Which solution to choose is up to network administrators and how they consider the impact on their traffic engineering setup.
4.1.2 Loop Free Alternates

The second solution is described in [14] where the authors describe simple solutions to identify loop free alternates, which guarantee loop free paths when transmitting traffic to such loop free alternate neighboring nodes. It will work in pure IP networks in addition to label distribution networks and can handle single failures for links, nodes and shared risk link groups. The basic idea of identifying a loop free alternates neighbor $N$ of source $S$ for destination $D$ can be described by the following criteria:

$$\text{distance}(N, D) < \text{distance}(N, S) + \text{distance}(S, D)$$  \hspace{1cm} (4.2)

Where $N$ is the neighbor, $D$ is the destination and $S$ is the source that looks for loop free alternates.

This is showed with an example in Figure 4.2 where all links in the topology have equal cost. Node $S$ tries to send a packet to destination $D$ and the normal path for this type of traffic would be $S \rightarrow 1 \rightarrow 2 \rightarrow D$. If then the link between $S$ and 1 is unavailable node $S$ could use the neighboring node $N$ as a loop free alternate because the path $N \rightarrow 3 \rightarrow 2 \rightarrow D$ is strictly shorter than $N \rightarrow S \rightarrow 1 \rightarrow 2 \rightarrow D$.

The next two sections in this chapter thoroughly describes the two recovery schemes evaluated in this thesis which both fit the last solution of the IP Fast Reroute Framework, namely the multi-hop repair paths.

4.2 IP Fast Reroute Using Not-via Addresses

The IETF Routing Area Working Group has developed a recovery scheme referred to as IP Fast Reroute Using Not-via Addresses and is defined in [9].

This chapter will first explain the concept of the recovery mechanism in Section 4.2.1 and then give a more thorough overview in Section 4.2.2.

4.2.1 Concept

The purpose of repair is to ensure that packets do not traverse a failed network element on their flight towards the destination, and instead route the packets
4.2. IP FAST REROUTE USING NOT-VIA ADDRESSES

around the failure. In Not-via, this is solved by assigning a set of special addresses to each protected component. The semantics of such an address is that routers must route the packets to the router advertising the address, without passing the protected component with which that address is associated.

To illustrate this with an example, we look at the network in Figure 4.3. In this example, source S wants to send a packet to the destination D and normally it would send this packet through node P. However, the node P has failed, so node S can not send the packet through node P. In the Not-via approach it will then try to send the packet to node B Not-via node P to reach the other side of the failed network element. Node B is advertising its address Bp which is to be used for traffic towards node B where traffic does not traverse node P. Node S can then encapsulate this packet and send it to Bp. The path from S to Bp is the shortest path from S to B not going via P. As long as there is a path from S to B that does not pass through P, then the packet will arrive at the destination. However, if the failure of node P partitions the network, it would not be possible to send a packet from S to D regardless of which network layer recovery mechanisms is chosen. When node B receives the encapsulated packet destined to node D, it will decapsulate it, and then forward the packet towards its final destination.

It is worth to mention that the repair path may include nodes on the shortest path between node B and the final destination D. However, since node B where the packet is decapsulated is closer to node D than node S who encapsulated the packet, this will not lead to a loop.

For complete protection of node P, all the adjacent nodes of P will require a Not-via address that allows traffic to be directed to that node without traversing node P. This is shown in Figure 4.4.

4.2.2 Overview

The Not-via scheme requires that all routers are able to find a path to a recovery interface protecting a specific component. From the previously mentioned example in Figure 4.3, this means that all the routers on the repair path from
B to S needs to have a path to Bp. To calculate this path, protecting the node P, they use their ordinary link state database. However, they consider node P and all its links down before running a shortest path first algorithm. After this calculation, each router is able to find its next hop to node B. This next hop will be saved in the routing table as Bp, since it is the protected path from that router to node B that is not passing by node P.

**Repair Path Computation**

Every router in the network must calculate a next hop for the possible failure of any other router in the network. As an example, a specific router R can consider a router in the network as router P from Figure 4.4. It then fails P, and calculates its own path to Ap, Bp, Cp and Sp which then is guaranteed to not traverse node P.

To be able to generate a complete routing table, by failing each router and thereby being able to calculate the shortest path to its protection addresses, every router has to calculate \( n - 1 \) SPF's where \( n \) is the number of routers in the network. This will typically not scale very well and the authors have proposed an optimization of the calculation. Each router X in the network would calculate its SPF tree and use this to find the default path to all nodes, in addition to all Not-via addresses. It is possible to do this as a part of the normal SPF calculation. When this is done, router X will for each router P in the network perform the following operations.

1. Remove router P from the topology.
2. Perform an incremental SPF calculation which aborts as soon as all Not-via P addresses are attached to the SPT.
3. Revert to the old topology.

By using incremental SPF instead of ordinary SPF the calculation time will be severely reduced, and thus the computational effort is much less than \( n - 1 \).
SPFs. In fact, the authors refer to experiments from real world networks with 40 to 400 nodes which suggest that the worst case computational complexity using the mentioned optimizations is equivalent of performing 5 to 13 full SPFs.

By using loop free alternates it is possible to optimize this even further. If a router detects that it is possible to protect a link with an LFA, it can mark its link state updates with this information. When routers computing Not-via routes see this, they can drop to calculate iSPF for these links or nodes.

Repairing Errors

The Not-via scheme covers node and link failures and how this section describes how it does this for both types.

Node Failure The normal operation of Not-via is to assume that there is a node failure whenever a router encounters a failure. If router S in the network showed in Figure 4.3 encounters an error on the link S-P it assumes that the node P has failed and thereby sends the packet to a Not-via address which ensures that no traffic will traverse the node P. However, since there are several Not-via addresses protecting P, S has to choose the correct one which is on the shortest path downstream to the destination. This is possible because in Not-via, a router has an extra field in its routing table, called the next-next-hop. This is the node that the assumed failed neighbor of S would normally send the packet to had there been no error. If the destination is D, then this next-next-hop would be node B, since it is the next hop from P on the shortest path towards D. This next-next-hop is denoted the repair target. When router S has figured out which Not-via address to use, it encapsulates the packet and sends it on the shortest path to Bp. When the repair target B receives the packet, it decapsulates it and sends it downstream towards the destination.

With this technique it is only necessary with one level of encapsulation, regardless of the link weights, asymmetry etc. in the network. However, if the failure was a single point of failure which partitioned the network, Not-via will of course not be able to fix this error, since there exists no way of repairing such errors.

Link Failure Even though Not-via assumes that there is a node which has failed when detecting an error there may exist destinations which are only reachable through the failed node. In this case, it is desirable to attempt to repair the error by assuming that only the link has failed by forwarding the packet to a neighboring node and and ensuring that this node will never send the packet through a path traversing the assumed failed link.

To perform a link repair in Not-via, again look at the network in Figure 4.3. If S is to do a link repair for the link S-P it will encapsulate the packet and send it to the Not-via address Ps, which is the address of P where the path does not pass node S. Since all neighbors of S have calculated a path to Ps in case S itself failed, this will not lead to any extra computations. S could then send the packet to any of its neighbors except node P, since S-P is down. However,
it is desirable to send it to the shortest path towards P not passing S-P. It is possible to find this path by running an SPF with the link S-P failed and this is possible by doing an incremental SPF, and aborting as soon as the address Ps has been reattached.

This proposal can be used to solve the last-hop problem as described in Section 4.4.1. Whenever a link repair is performed, the packet is encapsulated and sent to the correct repair target. However, if in fact there was no link error, but instead the whole node P had failed then Not-via ensures that no loop occurs since it does not recover traffic already encapsulated. This is important since it should never repair a repair path. It does automatically because Not-via addresses never have a backup path. As an example, S sends a packet to Ps and the packet reaches A and node A discovers that the link A-P is down. Since the packet is already sent to a Not-via address, node A will not do an extra layer of encapsulation, and then instead drop the packet thus preventing the formation of a loop.

Deployment

The Not-via scheme has some strict demands for deployment in a network. One of them is that it needs to encapsulate packets. Any IETF specified IP in IP encapsulation mechanism could be used, for example by using IP in IP [44], GRE [18] or L2TPv3 [30]. All routers in the network that are supposed to protect a network component need to have this functionality. In addition, it is not possible to do incremental deployment of this scheme. However, it is of course possible to create islands of nodes without IPFRR support, but this will not be an optimal solution as recovered traffic needs to be sent around this island of non-supported nodes.

Tunneling

Whenever Not-via is in use for a recovery path, it uses tunneling [57] [44]. Tunneling introduces overhead for all nodes and links included in the recovery path.

First of all, the node discovering the error must find the next hop for the tunnel towards the next-next-hop of the original path. Afterwards it must encapsulate the packet inside another IP packet and then sending it. All nodes and links are then encumbered with additional link load since an extra IP header exists. This is not considered in the results chapter of this thesis when looking at load. In addition, when the packet reaches the end of the tunnel, this node must decapsulate the packet before sending it towards its final destination. This is the nature of most tunneling protocols, but still it is worth to mention.

Another problem with tunneling is that by adding an extra IP header it may lead to fragmentation of packets on the MAC layer as shown in Figure 4.5. Typically normal, unrecovered traffic will use the MTU available, and when the IP-in-IP tunnel introduce an extra 40-byte IP header this will then exceed the
4.3. Failure Insensitive Routing

The Failure Insensitive Routing (FIR) scheme is described in [31] [40].

This chapter will first explain the concept of FIR in Section 4.3.1 and then give a more thorough overview in Section 4.3.2.

4.3.1 Concept

The basic concept of the FIR scheme is that of inferring failures in the network by looking at each packets path through the network, referred to as the packets flight. Based on knowledge of the network topology and the destination address of packets they are able to identify links that possibly may be dead and create routing tables that utilizes this information. Under operation of the network the recovery scheme can then carefully reroute packets around such possible points of failure. Their solution is totally proactive, so everything is calculated before an error in fact happens.

To be able to do recovery in this way, they need to use interface specific forwarding tables instead of one common forwarding table for all interfaces on the node as is usual for normal IP routers. Having the routing tables stored per interface, makes it possible to base the next hop both on destination and incoming interface. For optimization reasons each line card usually has one routing table anyway, so the authors consider this a small change to normal router design.

The interface specific forwarding tables are calculated by using modified versions of SPF algorithms. First of all they identify for all incoming links which interfaces are unusual for every destination. By unusual they mean an interface that would not be used for the packets destination had there been no error in the network. For each of such incoming interface - destination pairs they identify which links that are potentially failed, known as key links. When key

![Diagram of Fragmentation of a Packet Fit to MTU When Using a Tunnel](image)

Figure 4.5: Fragmentation of a packet fit to MTU when using a tunnel

MTU, thereby causing fragmentation. In the figure, the same payload is shown with both the untunneled and the tunneled approach.
links are identified they remove these links from the topology before calculating the shortest path towards the given destination. This SPF calculation is used to find only the next hop towards the given destination.

When traffic is flowing through the network, FIR routers always use the precalculated interface specific forwarding tables regardless of whether the traffic is recovered or not. However, the node identifying the failure needs to have additional routing tables, referred to as interface specific backwarding tables which are also precalculated. These tables are used for the initialization of a recovery. Whenever a node is about to send traffic over a failed link, it looks up the destination in the backwarding table for that interface, and will find a new next hop for the given destination. They are calculated basically the same way as the interface specific forwarding tables, except they also remove the link the backwarding table is for.

Example

This section describes how packets would be routed using conventional routing, and then describes how the same scenario would turn out using FIR enabled routers. The network is shown in Figure 4.6 where each link is labeled with its weight.

If conventional routing is used and a packet is sent from node 1 to node 6 the shortest path would be $1 \rightarrow 2 \rightarrow 5 \rightarrow 6$. But consider now, that the link 2-5 is down. Node 1 will first forward the packets to node 2 as usual. When reaching node 2 the packets will be dropped because there exists no next hops from node 2 with destination 6 when 2-5 is down. As soon as node 2 becomes aware of the failed link, it will recompute its routing tables without the link 2-5 and the next hop for destination 6 from node 2 will be node 1. If the other routers in the network have not recomputed their routing tables, node 1 will send packets to destination 6 back to node 2, which will then send it back to node 1 etc. A micro-loop exist until node 1 also has recomputed its routing table without link 2-5, creating excessive load. After the TTL has been decreased to 0, the packets will be dropped. When all nodes have recomputed their routing tables, traffic will again traverse the network in a correct manner. This may take quite a lot of time, typically measured in seconds or even tens of seconds.

When routers are using the FIR scheme the example with link 2-5 down, and node 2 receives the packet it does not drop the packet. It will instead sense the failure and locally reroute the packets to node 1 using its backwarding table.
4.3. FAILURE INSENSITIVE ROUTING

for that interface. When packets destined for node 6 come from node 2 to node 1, node 1 infers that some link on the shortest path to 6 has failed, even though node 1 is not explicitly notified of any failures. Otherwise node 2 would not forward packets to node 1 with destination 6. Node 1 knows that the key links for the interface 2→1 for destination 6 are the links \{2-5,5-6\} and has calculated a routing table without these links for that destination. This means that since the shortest path node 1 considers is the path 1 → 4 → 6, it will forward the packet to node 4 and thus avoiding both the potentially failed links 2-5 and 5-6. The complete route for the packet will be 1→2→1→4→6. Note that even though node 1 appears twice in the path, it does not constitute a loop. With interface specific forwarding, there is no loop until a packet traverses the same link in the same direction twice.

4.3.2 Overview

As is required by the IP Fast Reroute framework, the FIR scheme does not change behavior of routing whenever a failure does not exist. This means that there will be no effects on traffic engineering schemes during normal operation. As Not-via, FIR is also possible to use together with ECMP.

The FIR scheme provides continuous forwarding, even during existence of failures. The forwarding continuity does not depend on the time used for table computation since the scheme precalculates the forwarding tables. It provides this feature by suppressing the notification of failures and instead enabling local rerouting of the packets. This means that there is no need to fine tune link failure propagation times and other parameters, as is proposed by studies on optimizations of IP reconvergence [37] [17]. During the existence of link failures packets are routed through alternate paths, which may be suboptimal. But as soon as the failed link comes up again, the node adjacent to the link starts forwarding through the previously failed link on the shortest path, and no advertisement needs to be sent.

For persistent failures, the node may send out a global advertisement and initiate a full IP reconvergence. The suppression-time before sending out link failure notifications can be set by network operators. Since a large portion of failures last less than a minute [34], the suppression-time will will typically be set to at least one minute. It is possible with FIR to have such a large suppression time, because the recovery scheme guarantees that traffic still will flow even during the existence of failures.

The FIR scheme introduces changes to both the routing plane and to the forwarding plane. In the routing plane, the shortest path algorithms has to be replaced with algorithms implementing one of the algorithms the authors propose for use in FIR routers. The interface specific forwarding tables do not require large changes to the forwarding table, since they typically already have a forwarding table stored on the line card. However, the backwarding table must be stored somewhere. Either an extra forwarding table must be stored on the line card, or it must be stored in some central component of the router. If the centralized approach is to be used, the content of the backwarding table must be used to overwrite the forwarding table whenever an error occurs. The author’s proposal for this is described later in this section.
It is important to mention that the FIR approach does not support node failures. However, the concept has been further developed in [64] to also support node failures.

The rest of this section is devoted to describe the algorithm for finding key links and how these are used to create interface specific forwarding and backwarding tables.

Algorithm

When routers identify key links, they have to iterate through all links $u \rightarrow v$ in the network. They have to do this for every incoming interface $j \rightarrow i$ where $j$ is the neighboring node and $i$ is the node itself. A key link $K_{j \rightarrow i}^d$ for a specific incoming interface $j \rightarrow i$ and a specific destination $d$ can then be defined as follows:

1. with the link $u \rightarrow v$, the node $j$ is a next hop from node $i$ to destination $d$.
2. without $u \rightarrow v$, the directed edge $j \rightarrow i$ is along a shortest path from node $u$ to destination $d$.

The authors propose a non optimized algorithm that is able to find all such key links and this is listed in Algorithm 2 with the legend specified in Table 4.1. The algorithm is described in detail in [40]. They also propose two optimized algorithms which are not as easy to understand as the one mentioned here, namely the Available Shortest Path First (ASPF) and an incremental version of this algorithm called Incremental ASPF (IASPF). The last one uses more space during calculation, but takes advantage of previously stored information instead of calculating everything for each iteration. To show the result of the key links algorithm, all key links for the FIR test network in Figure 4.6 are listed in Table 4.2.

Interface specific forwarding tables

After finding the key links for a specific interface, they are used to create the interface specific forwarding tables. This is a rather straightforward process and basically the process is to run a SPF for each destination after removing all key links for a specific incoming interface $j \rightarrow i$ and destination $d$. Since FIR supports ECMP, a forwarding table entry does not contain a single entry, but is defined as a set of next hops.

Consider the incoming interface $j \rightarrow i$ and let $\mathcal{E}$ represent all links in the network. Also, let $\mathcal{R}_i^d(\mathcal{X})$ represent all next hops from node $i$ to destination $d$ given the set of links $\mathcal{X}$. In addition the set of next hops towards the destination $d$ for packets arriving at $i$ through the interface $j \rightarrow i$ is denoted $\mathcal{F}_{j \rightarrow i}^d$. This entry can be calculated using ordinary SPF after excluding all the key links $K_{j \rightarrow i}^d$, as shown in Equation 4.3.

$$\mathcal{F}_{j \rightarrow i}^d = \mathcal{R}_i^d(\mathcal{E} \setminus K_{j \rightarrow i}^d)$$ (4.3)
Algorithm 2 Algorithm for finding key links

1: for all \( d \in \mathcal{V} \) do
2: \( \mathcal{K}_{d \rightarrow i} \leftarrow \emptyset \)
3: end for
4: \( \mathcal{T}_i \leftarrow \text{SPF}(i, \mathcal{V}, \mathcal{E}) \)
5: if \( j \notin \mathcal{N}(j, \mathcal{T}_i) \) then
6: return \( \mathcal{K}_{j \rightarrow i} \)
7: end if
8: for all \( u \rightarrow v \in \mathcal{E} \setminus \{ j \rightarrow i \} \) do
9: \( \mathcal{T}_{u \rightarrow v} \leftarrow \text{SPF}(u, \mathcal{V}, \mathcal{E} \setminus \{ u \rightarrow v \}) \)
10: if \( j \rightarrow i \in \mathcal{E}(\mathcal{T}_{u \rightarrow v}) \) then
11: for all \( d \in \mathcal{V}' \land \mathcal{V}(S(i, \mathcal{T}_{u \rightarrow v})) \) do
12: \( \mathcal{K}_{j \rightarrow i} \leftarrow \mathcal{K}_{j \rightarrow i} \cup \{ u \rightarrow v \} \)
13: end for
14: end if
15: end for
16: return \( \mathcal{K}_{j \rightarrow i} \)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d )</td>
<td>destination node</td>
</tr>
<tr>
<td>( \mathcal{V} )</td>
<td>set of all vertices</td>
</tr>
<tr>
<td>( \mathcal{E} )</td>
<td>set of all edges</td>
</tr>
<tr>
<td>( \mathcal{K}_{d \rightarrow i} )</td>
<td>set of key links for destination ( d ) and incoming interface ( j \rightarrow i )</td>
</tr>
<tr>
<td>( \mathcal{K}_{j \rightarrow i} )</td>
<td>collection of key links ( { \mathcal{K}_{d \rightarrow i} \forall d } )</td>
</tr>
<tr>
<td>( \mathcal{T}_i )</td>
<td>shortest paths tree rooted at ( i )</td>
</tr>
<tr>
<td>( \mathcal{T}_{u \rightarrow v} )</td>
<td>SPT of ( i ) without edge ( u \rightarrow v )</td>
</tr>
<tr>
<td>( P(k, \mathcal{T}) )</td>
<td>parents of node ( k ) in tree ( \mathcal{T} )</td>
</tr>
<tr>
<td>( N(k, \mathcal{T}) )</td>
<td>next hops to ( k ) from root of tree ( \mathcal{T} )</td>
</tr>
<tr>
<td>( S(k, \mathcal{T}) )</td>
<td>subtree below ( k ) in tree ( \mathcal{T} )</td>
</tr>
<tr>
<td>( V(\mathcal{T}) )</td>
<td>set of all vertices in tree ( \mathcal{T} )</td>
</tr>
<tr>
<td>( E(\mathcal{T}) )</td>
<td>set of all edges in tree ( \mathcal{T} )</td>
</tr>
<tr>
<td>( \text{SPF}(i, \mathcal{V}', \mathcal{E}') )</td>
<td>computes SPT rooted at ( i ) given the graph ( (\mathcal{V}', \mathcal{E}') )</td>
</tr>
<tr>
<td>( \mathcal{K}_{i \rightarrow j}(\mathcal{X}) )</td>
<td>all next hops from node ( i ) to destination ( d ) given the set of links ( \mathcal{X} )</td>
</tr>
<tr>
<td>( \mathcal{F}_{j \rightarrow i} )</td>
<td>set of next hops towards the destination ( d ) arriving at ( i ) through the interface ( j \rightarrow i )</td>
</tr>
</tbody>
</table>

Table 4.1: Legend/notation used for the FIR scheme
CHAPTER 4. RECOVERY SCHEMES FOCUSED ON IN THIS THESIS

$K_{1\rightarrow 2} = \{K_{1\rightarrow 2}^1 = \{1\rightarrow 3\}\} \cup \{K_{1\rightarrow 2}^2 = \{1\rightarrow 4\}\}$
$K_{1\rightarrow 3} = \{K_{1\rightarrow 3}^2 = \{1\rightarrow 2\}\}$
$K_{1\rightarrow 4} = \emptyset$
$K_{2\rightarrow 1} = \{K_{2\rightarrow 1}^5 = \{5\rightarrow 6, 2\rightarrow 6\}\} \cup \{K_{2\rightarrow 1}^6 = \{2\rightarrow 5\}\}$
$K_{2\rightarrow 5} = \{K_{2\rightarrow 5}^1 = \{2\rightarrow 1\}\}$
$K_{3\rightarrow 1} = \emptyset$
$K_{3\rightarrow 5} = \emptyset$
$K_{4\rightarrow 1} = \emptyset$
$K_{4\rightarrow 6} = \emptyset$
$K_{5\rightarrow 2} = \{K_{5\rightarrow 2}^6 = \{5\rightarrow 6\}\}$
$K_{5\rightarrow 3} = \{K_{5\rightarrow 3}^2 = \{5\rightarrow 2\}\}$
$K_{5\rightarrow 6} = \emptyset$
$K_{5\rightarrow 2} = \{K_{5\rightarrow 2}^3 = \{5\rightarrow 3\}\}$
$K_{6\rightarrow 4} = \{K_{6\rightarrow 4}^3 = \{6\rightarrow 5\}\}$
$K_{6\rightarrow 5} = \{K_{6\rightarrow 5}^3 = \{6\rightarrow 4\}\}$

Table 4.2: Complete list of key links for the FIR test network shown in Figure 4.6

<table>
<thead>
<tr>
<th>Interface/Destination</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 → 1</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3 → 1</td>
<td>2</td>
<td>-</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4 → 1</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.3: Interface specific forwarding table for node 1

This equation will yield the forwarding table for node 1 shown in Table 4.3. Each row defines the forwarding table for the interface specified in the first column. Each entry in the table specifies the next hop for the destination listed in the top row.

As an example, whenever node 1 is about to send a packet to destination 6 and receives this packet from node 2 it will use the row denoted as 2 → 1 in Table 4.3. The next hop for destination 6 is 4. This is because the key links are $K_{2\rightarrow 1}^6 = \{5\rightarrow 6, 2\rightarrow 6\}$ and the next hop with the shortest path towards 6 when both 2 → 5 and 5 → 6 is considered down is 4.

Interface specific backwarding tables

The key links are also used to create interface specific backwarding tables. These tables are used by the nodes that in fact detect an error. Whenever a node discovers that one of its links has failed, it looks up the destination in its backwarding table specified for the interface that has no link. Here it will find the substitute interface, and thereby initiating the local rerouting process.

The entries in the backwarding table denoted by $B_{i\rightarrow j}^d$, give the set of alternate next hops towards the destination $d$, when the link $i \rightarrow j$ is down. They are precomputed the same way as the forwarding table, however, the failed link
4.3. FAILURE INSENSITIVE ROUTING

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
Interface/Destination & 2 & 3 & 4 & 5 & 6 \\
\hline
1 $\rightarrow$ 2 & 3 & 4 & 3 & 3 & 3 \\
1 $\rightarrow$ 3 & 4 & 2 & - & 4 & 4 \\
1 $\rightarrow$ 4 & - & - & 2 & - & - \\
\hline
\end{tabular}
\caption{Interface specific backwarding table for node 1}
\end{table}

itself is naturally removed from the set of available links before running the SPF as shown in Equation 4.4.

\[ B^d_{i \rightarrow j} = R^d_i(E \setminus K^d_{i \rightarrow j} \setminus i \rightarrow j) \] (4.4)

The resulting backwarding table for node 1 is shown in Table 4.4. The first column denotes the failed interface, and the entries in the table denote for each destination the next hop to be used.

As an example, consider a packet sent from node 1 to node 6. The normal path towards destination 1 would be 1 $\rightarrow$ 2 $\rightarrow$ 5 $\rightarrow$ 6. However, consider the link 1 $\rightarrow$ 2 down. When node 1 discovers the failed link, it will consult its backwarding table and find 3 as the next hop for destination 6. The flight of the packet will be 1 $\rightarrow$ 3 $\rightarrow$ 5 $\rightarrow$ 2.

It may seem strange that for the interface 1 $\rightarrow$ 2 in Table 4.4 the next hop for destination 3 is 4 and that the next hop for destination 4 is 3. To understand this, remember that packets from 1 to 3 are only forwarded on 1 $\rightarrow$ 2 if in fact 1 $\rightarrow$ 3 itself is dead. Then node 1 will have no other outgoing interface than through node 4. The same argument is used for reaching node 4, since 1 $\rightarrow$ 2 will only be used to forward traffic when 1 $\rightarrow$ 4 itself is dead.

The cells marked with '-' is whenever the FIR algorithm does not yield a feasible next hop. This means that when that the destination will be unreachable.

A bad thing about backwarding tables though, is that they in fact introduce a change in the forwarding plane of the router. Routers are designed to have only one forwarding table, at least per interface, so when another table is introduced, a change needs to be made. The authors realize this problem, and state that they do not know the total cost of changing the forwarding plane in routers, so they also give us a solution to the problem. However, this solution would transform FIR into being a reactive protocol, since they propose to do changes to the routing tables after an error is discovered. To keep FIR as a proactive scheme, the change to the forwarding plane has to be implemented.

The authors propose to keep the backwarding table in the control plane, and create a merge between the forwarding table and backwarding table which will then be used to overwrite the original forwarding table. The way this can be done is shown in Equation 4.5. To understand the equation, suppose that the link $i \rightarrow k$ has failed and the new forwarding table is denoted with $\tilde{F}$. Then, for destination $d$ for interface $j \rightarrow i$, where $j \neq k$ the new forwarding table can be computed as:
Figure 4.7: Topology to show loops in FIR-networks

\[ \tilde{F}^d_{j \rightarrow i} = \begin{cases} B^d_{i \rightarrow k} & \text{if } F^d_{j \rightarrow i} = k \\ F^d_{j \rightarrow i} & \text{otherwise} \end{cases} \quad (4.5) \]

The basics of this equation is that all destinations that have \( k \) as their next hop are replaced with an entry from the backwarding table.

Equation 4.5 however does not take into account ECMP. If ECMP is used in the network, Equation 4.6 has to be used.

\[ \tilde{F}^d_{j \rightarrow i} = \begin{cases} F^d_{j \rightarrow i} \setminus k \cup B^d_{i \rightarrow k} & \text{if } k \in F^d_{j \rightarrow i} \\ F^d_{j \rightarrow i} & \text{otherwise} \end{cases} \quad (4.6) \]

The basic of Equation 4.6 is that if \( k \) is in the set of next hops in the forwarding table entry \( F^d_{j \rightarrow i} \) it has to be replaced by an entry from the backwarding table \( B^d_{i \rightarrow k} \). However, the rest of the next hops are preserved.

Two Link Failures

The algorithm for finding key links and generating forwarding and backwarding tables defined in [40] may in fact lead to loops if two links fail in the same network simultaneously. I will justify this with an example where I show a variation of the example network from the article as shown in Figure 4.7. The network is the same as before, except that the cost of the link 4-6 has been changed from 3 to 1.

Let us now consider that both links 2-5 and 4-6 fail simultaneously and that node 1 is sending a packet to node 6. The shortest path is the same as before, so node 1 will again transmit the packet to node 2. However, node 2 will detect that 2-5 has failed and thereby do a lookup in its backwarding table for the link 2-5. It will find 1 as its next hop and transmit the packet to 1. When node 1 receives a packet on the interface 2 → 1 it knows that \( K^d_{2 \rightarrow 1} = \{5 \rightarrow 6, 2 \rightarrow 6\} \) and will find node 4 as its next hop in the interface specific forwarding table. However, node 4 will also find a dead link, namely 4-6. In the backwarding table of 4-6 node 4 will of course find node 1, since it is the only other neighbor. After receiving the packet on the interface 4 → 1, node 1 will know that since it has no key links for that interface, it will use its interface specific forwarding table, which will yield the same result as an ordinary OSPF routing table in this case. So node 1 will transmit the packet to node 2. In the FIR scheme a
4.4. Comparison of Not-via and FIR

This section compares the two schemes evaluated in this thesis, namely Not-via and FIR. It will compare the schemes in the areas of coverage, link/node errors, last-hop, recovery path endpoint, requirement to topology layout and hardware/software of underlying components, complexity and recovery path identification.

4.4.1 Coverage

Coverage is an important feature of a recovery scheme. In [34] they state that around 20% of the failures are planned down time due to maintenance. Whereas for the unplanned failures more than 85% affect only a single link or a single router which means that a recovery scheme tackling both node and link failures will have large coverage for network failures.

It is therefore important to point out that the Not-via scheme provide complete repair coverage for single link or node failures, whereas the FIR approach only covers single link failures. It does however provide coverage for all single link failure scenarios. This is definitely an area where Not-via outperforms FIR substantially. Depending on the nature of the network, an operator may want to rather have full coverage than using a scheme with better recovery paths but only partial coverage.

It could therefore be interesting to investigate the features of the FIR-based recovery scheme proposed in [64] which also is able to protect node errors, and compare this to the features of Not-via. Based on statistics from failures in [34], the authors state that the proposed scheme will provide repair paths all planned maintenance and unplanned single link/node failures, which is 88.6% of all types of failures.

Since single link failures are so common, it is natural to ask whether it is important to get node protection for a recovery scheme if doing so introduces a significant overhead for recovering link failures. That is; if a solution can support either link failures or both node and link failures, and the link failure solution is far more efficient than the other, would the operator then want the efficient recovery, or the recovery scheme which had the best coverage? This is for instance the case for FIR which is in this thesis described and evaluated as a scheme which can cope with link failures, but a node-protecting solution has been proposed, but with less optimal paths.
CHAPTER 4. RECOVERY SCHEMES FOCUSED ON IN THIS THESIS

4.4.2 Link/node errors

The Not-via recovery scheme basically look at all failures it discovers as node failures. This means it will always consider the neighboring node downstream of the link failure as failed. Whenever a node is considered dead, so will all links directly connected to it. Whereas in the link failure case, only the link known to be unavailable is considered failed. Whenever more links are removed from a topology, a recovery scheme is likely to create less optimal paths.

This can easily be seen by looking at Figure 4.8 where all link costs are equal and node S sends a packet to node D. The link between A and B is down. However, a node-protecting scheme will also consider all the dashed network entities as down, namely node B, the link $B \rightarrow C$, and the link $E \rightarrow B$. This means that the only available path in this scenario is $S \rightarrow A \rightarrow E \rightarrow F \rightarrow G \rightarrow C \rightarrow D$, a total of 6 hops.

The only time Not-via does not consider a link failure to be a node failure is when there is no other means of recovery, i.e. the neighboring node is the destination node. For more information about last hop, see Section 4.4.3 below.

The FIR scheme however, does not consider all failures as node failures since it only protects links. Whenever it receives a packet it has considered which links that could be down when it receives a packet on that specific interface. This means it will have a more options available for recovery paths than the Not-via scheme as can be seen in Figure 4.9 which is the same scenario before. Since only the link $A \rightarrow B$ is considered down, the shortest path will then be $S \rightarrow A \rightarrow E \rightarrow B \rightarrow C \rightarrow D$, which is a total of 5 hops.

4.4.3 Last-hop

Both Not-via and FIR solve the problem with the last link before the egress node failing. Not-via does this by using the not-via address of the destination not-via the node discovering the error as described in Section 4.2. Since FIR is only protecting links, this is implicit.
However, if the FIR-based recovery scheme which is able to support node-failures is used, this would not solve the last-hop problem [64]. In fact, the proposed algorithm would return no key nodes (similar concept as key links), and they would therefore have no recovery path for the destination.

### 4.4.4 Recovery path endpoint

Another important difference between Not-via and FIR (including FIR with support for node failures) is that of how they find paths around a failure. In Not-via, they aim to find a path to the next nexthop towards the destination. As an example, look at the topology shown in Figure 4.10. Consider the case where traffic is to be sent from source S towards destination D. This would mean that traffic normally would use the path $S \rightarrow A \rightarrow B \rightarrow C \rightarrow D$.

If in this topology the link between A and B fails, Not-via will try to find a path from A to C not-via node B. This yields the recovery path $A \rightarrow E \rightarrow F \rightarrow G \rightarrow C$. The complete path from source to destination will be $S \rightarrow A \rightarrow E \rightarrow F \rightarrow G \rightarrow C \rightarrow D$, namely 6 hops.

This is totally different from FIR which would in the same scenario find a recovery path from node A directly towards the destination node D. In most scenarios, the path chosen by Not-via is included in the set of possible solutions that the FIR scheme consider, the path for traffic with FIR will mostly have either equal cost or lower cost, compared to Not-via.

The recovery path FIR would choose in the scenario mentioned above is $A \rightarrow E \rightarrow F \rightarrow G \rightarrow D$ and is shown in Figure 4.11. The complete path from source to destination would be $S \rightarrow A \rightarrow E \rightarrow F \rightarrow G \rightarrow D$, and this path is only 5 hops. The same path would be found if the FIR-based recovery scheme which supports node failures was used.

In Figure 4.12 a scenario where Not-via provides the best recovery path is shown. Source S tries to send a packet to destination D, but the link $A \rightarrow B$ is down. The path choices for FIR is shown in Figure 4.13. The reason why Not-via is best in this scenario is because it considers only node B and the links directly connected to B as down since it is node-protecting. The next-next hop
will be node $C$ and this is also yields the same as the shortest path possible. FIR on the other hand finds three key links: $A \rightarrow B$, $B \rightarrow C$ and $C \rightarrow D$ where $C \rightarrow D$ is in fact part of the Not-via recovery path.

4.4.5 Topology layout requirements

Neither the Not-via or FIR scheme has stringent requirements for the network topology other than it must have 2-link connectivity to be able to support link-failures. That is, the network is not partitioned if a single link is removed, regardless of which. And since Not-via is node-protecting, the network must also be 2-node connected which means any one node could be removed without partitioning the network. Connectivity of topologies is detailed in Section 5.3.1.

4.4.6 Hardware/software requirements

The FIR scheme has two important requirements for hardware and/or software that may not be possible for some types of equipment. The first requirement is that FIR needs all Network Interface Cards to have its own forwarding table. This is as an example not true for a default Linux installation (as per kernel 2.6). However, most hardware routers for efficiency reasons do in fact have one forwarding table per NIC and this should make FIR a feasible choice for such equipment. The only change is that today’s equipment keeps the same forwarding table for all line cards, whereas FIR has a different one for each line card.

The second requirement is regarding the backwarding tables. FIR can either make rapid changes to the forwarding tables to comply with its demand of a backwarding table. The authors of FIR propose a fast solution of overwriting affected entries in the forwarding table with data from the backwarding table. Because of this, it may be feasible to say that FIR is a reactive scheme and not a proactive scheme. However, if a new hardware based router was to be produced, it would be possible to create it with a backwarding table, and thereby making FIR a proactive scheme. This thesis looks at FIR based on the latter assumption.
The Not-via scheme does not have stringent requirements for hardware or software. However, Not-via introduces a set of additional addresses to be used for repairs which will increase the CPU load for all SPF calculations. FIR does not introduce a new address space, but it do increase the CPU load. The complexity of the recovery schemes is discussed in Section 4.4.7 below. In addition, Not-via uses tunneling to forward traffic on the recovery paths and it requires that the routers are capable of setting up and using IP tunnels.

4.4.7 Complexity

Since both Not-via and FIR are more complex schemes than basic shortest path first routing they naturally increase the overhead of creating routing tables, both in time and space domain.

An unoptimized implementation of FIR would consume a large amount of CPU cycles during calculation of key links. The reason for this is that every node must find all key links towards all destination nodes for all its incoming interfaces, so it must consider $|VE|$ number of topologies. Here, $V$ is number of nodes, and $E$ is number of links. In addition, it has the same complexity for the backwarding tables, so the calculation complexity will be multiplied by 2. The authors of FIR propose two ways of optimizing this with one being considerably faster than the other, but then again using more memory and they have both done analysis and run real tests of the calculation times. The slowest method ASPF has a complexity of 8 to 16 SPF computations from network sizes from 30 to 200. In real tests run, it was using typically 22 times as long time as a normal SPF computation for a network with 200 nodes. The other method, IASPF, uses typically 6 times an SPF calculation for the same kind of topology. Its complexity ranges from 1 to 5 times a normal SPF calculation. However, this method uses an additional space of $D^2|V|$ where $D$ is the diameter of a network, and $V$ is number of nodes.

When looking at Not-via they also have to do a lot of calculations, namely $V - 1$ times an SPF calculation, where $V$ number of nodes. However the authors have also proposed an optimized solution and they have done tests of this based on real world network topologies. For topologies with a node count from 40 to 400 the computational complexity is equivalent of doing between 5 and 13 SPF calculations.

4.4.8 Recovery path identification

Another difference between Not-via and FIR is that Not-via needs to use tunneling to be able to transfer packets correctly through the recovery paths as described in Section 4.2.2. A nice feature of this is that Not-via capable routers can identify whenever a recovery path is in use, and therefore be able to not repair a repair path. The downside of this is that it introduces extra CPU load, link load and may cause fragmentation of packets.

FIR does not use tunneling and thereby does not introduce the same problems. Instead, it identifies a recovery path implicitly by looking at the packets flight. However, this may in some cases lead to problems when more than one
link fails, as described in Section 4.3.2 since FIR then will try to repair a repair path.

4.5 Other schemes

This section lists other proposals for doing recovery in IP networks, both reactive and proactive. A wanted property of recovery mechanisms is that they must not change the normal behavior of IP network operation and never introduce loops, but still being able to optimize recovery times. In [46] they list several recovery schemes and it is a nice introduction to this field of research.

4.5.1 Reactive IP recovery schemes

This section list schemes which focus on new ways to route traffic whenever an error occur, calculating the new routes at the time the error is detected.

Liu et. al [32] have created a recovery scheme which finds the next feasible hop if the default path has failed, and thereby enforcing a reroute of the traffic. This is a bit similar to finding a loop free alternate described later in this section, but this scheme does not demand that the alternate next hop is adjacent to the node detecting the error. Instead, when it has found a feasible next hop they tunnel all traffic to this node. They have created means of identifying affected traffic by assigning sequence number to all nodes in the shortest path subtree of the source node. However, if the rerouting path upstream nodes, then the node detecting the error will first inform these upstream nodes about the failure. Therefore this scheme is not strictly proactive even though this may have been calculated proactively.

A similar concept has also been developed in [37] where they identify the minimum number of nodes that needs to be informed of a failure. In fact they are able to do this with only minimal changes to any existing link state protocol. However, their focus is that tunneling may create problems for lower layer protocols in addition to overhead since each packet needs to be encapsulated at the node adjacent to the failure, and then decapsulated when it reaches the other end of the tunnel. More about properties of tunnels in Section 4.4. They propose instead to change the routing tables of the same routers that would be affected by a tunneling. But this will in fact in some cases lead to local loops, and in their article they provide solutions to this by using a vector-metric algorithm.

4.5.2 Proactive IP recovery schemes

If reactive IP recovery schemes are not fast enough, there exist even faster methods, called proactive schemes. These methods have precalculated backup next-hops and therefore no communication is needed between nodes during the recovery phase. A precalculated recovery scheme is defined in [23]. But they are basing their research on using MPLS which is a connection oriented mechanism.
4.5. OTHER SCHEMES

It is possible to do this in connectionless networks as described earlier in this chapter.

**Deflection routing**

A way of solving recovery for link failures is by using deflection routing schemes. As an example, in [60] they present such a solution where each node computes a map for each of its links to an alternate link. If the original link fails, the alternate link will be used. Since the number of links a node has is typically much smaller than the number of destinations in the routing table, the alternate link table is a lot smaller than the ordinary routing table. Whenever a link is down the deflection map is used. It is also used if the incoming interface is the same as the outgoing interface in the FIB, since it can then infer that something has gone wrong in other parts of the network.

**O2 routing**

[54] introduces a recovery scheme called O2. It is short for outdegree 2, which is graph theory terminology for each node having at two edges or links. O2 is an algorithm for doing recovery in such networks that supports both link and node failures. Basically they divide all traffic on at least two outgoing links, and whenever one link fails, they are able to recover traffic by using the remaining links. Two other routing algorithms based on the idea of O2 are described in [49]. One of them is a metric based algorithm which is link protecting. The metric-based algorithm finds the shortest detours and forwards packet on this path, whereas the other algorithm is pattern based, and extends the current routing graph by certain patterns. They have defined four different patterns, and by using these, they are able to create a correct recovery mechanism.

**Multiple Path Algorithm (MPA)**

It is also possible not always to use shortest path first algorithms, and in [39] a multiple path algorithm is proposed. This algorithm stores more than one next-hop for each destination, assuring that none of them will form a routing loop. A nice feature about this algorithm is that not all routers in the network will have to be upgraded to support MPA. But only routers supporting MPA can recover from link failures by using the algorithm. Other routers must wait for a normal IP convergence to occur and thereby creating a new viable path. However, with MPA, the order of packets is not guaranteed, and additional schemes may have to be implemented to ensure FIFO ordering of packets on the destination.

**Multiple Routing Configurations (MRC)**

A proactive recovery scheme that is designed to fit the IPFRR framework is Multiple Routing Configurations (MRC). It has a totally different way of calculating multi-hop repair paths is by having multiple logical routing configurations. This strategy is heavily utilized in Multiple Routing Configurations as defined in [27].
It is based on storing additional information in routers and using this information to ensure recovery from any single failure scenario, without knowing the cause for using recovery paths. They have one single mechanism, that solves both node and link failures. MRC is completely connectionless and can be implemented with only minor changes to existing routing solutions. They solve this by precalculating more than one routing configuration where each node and link is isolated in one configuration, i.e. it is not used for forwarding in this configuration. Whenever a router encounters a failed link or neighboring node, it looks up which configuration that this link or node is isolated in, marks the packet with this configuration and transmits on the next hop specified. Since all nodes use the same configurations, the isolated link or node will never be used for traffic. In the article they present MRC and analyze it with respect to scalability, backup path lengths and load distribution after failures.

An important thing to notice about MRC is that it covers both node and link failures. It also has a solution to the last hop problem. MRC has been heavily evaluated at Simula Research Laboratory and it is therefore not a part of the detailed analysis of this thesis.

IP Redundant Trees (IPRT)

IP Redundant Trees (IPRT) [6] is a new recovery scheme based on the connection oriented recovery scheme named Redundant Trees (RT) [35]. The redundant trees ensure that the root node of a pair of trees, named red and blue, may reach all other nodes through either the red or the blue tree in case of a single node or link failure.

The basic idea of IPRT is, for each network destination, create a pair of redundant trees of which the destination is the root node. Thus, the trees are reversed in comparison to RT and instead of guaranteeing that a root node may reach all other nodes through either the red or the blue tree, IPRT guarantees that all children may reach a root node through either the red or blue tree. Furthermore, for each pair of trees, the children use the next-hop dictated by the trees of which the destination is the root node to populate a single entry in two recovery FIBs. In addition IPRT introduce some logic for determining which of the recovery FIBs should be used in the event of a failed node or link.

The results in this design in terms of path length is comparable to the ones in MRC, even though this solution only will increase the number of routing tables to two, whereas MRC introduces several. As MRC, IPRT provide full coverage. It was released even later than MRC and is therefore a very new recovery scheme. IPRT was developed at Simula Research Laboratory as a part of a master thesis and as such it is not a focus in this thesis.
Chapter 5

Method

5.1 General

When evaluating recovery schemes three general approaches are available. One is to do a mathematical analysis by developing formulas describing the recovery scheme and use this to calculate how well the scheme would perform. Another approach would be to use a deployed network and do statistical analysis of the performance in either a deployed network or a testbed. The third approach is to implement a routing simulator designed to analyse recovery schemes. These approaches do not exclude the use of the other ones, but may for sure complement each other.

An advantage of evaluating recovery schemes with mathematical representations is that it can produce results quickly when it is developed. In addition, parameters could be altered to be able to test different configurations in a fast and consistent way. It may also help the evaluator in understanding how different parameters have an impact on the results. However, a common drawback of mathematical analysis is the need to simplify the model to fit into a mathematical representation. When a model is simplified, it may be easier to understand the results, but they may not be entirely correct. In addition, when modeling large and complex systems with this approach, the states needed to represent the model may become very large and in practice unpractical or even impossible unless a large computer cluster is at hand.

The use of test beds for evaluation of recovery schemes provide results with a high degree of credibility. The results may be superior to the mathematical analysis because they can provide results on how the recovery scheme works in a real life environment. Characteristics such as calculation times, resource congestion, etc. may be easier to find with the use of test beds. However, to create a testbed is typically a complex task and it may be difficult to implement all details of the recovery scheme as it may require changes to underlying entities such as linecards and kernels. In addition, it could be difficult to access all the information needed to do measurements, since some of the state information may be stored inside black boxes in the hardware. Another problem is that the financial cost of deploying a testbed may be considerable. Also, a testbed is by
nature a static environment, so there would be few possibilities to test several
different topologies. None of the recovery schemes evaluated in this thesis have
been deployed in testbed and as such this approach is unfit to use as a base for
the evaluation.

The third approach with doing simulations may be better fit whenever the
problem area is too complex to fit into a mathematical representation. It en-
ables the evaluator to run tests with different levels of abstractions. Different
abstractions level can typically be set during runtime or as configuration pa-
rameters. This will enable the evaluator to get results with different levels of
granularity. The simulations can either be very detailed by for example showing
the path for every packet, or the simulation can be high-level and for example
show the average recovery path length on different topologies. To get an initial
understanding of a recovery scheme, the scheme can easily be tested on different
topologies by just changing the input topology, as opposed to using a testbed,
where the setup is rather static. However, the results will not be as credible as
when the tests are run on a testbed.

To create the results for this thesis, the third approach was chosen, as the
recovery schemes were documented enough to enable the implementation of
them in a simulation environment. In addition, problems would arise if the
mathematical analysis was chosen, since it is not possible to fit all the details
into a mathematical model. The use of a testbed was ruled out as the resources
were not available, and the requirements for underlying hardware/software could
not be fulfilled (multiple forwarding/backwarding tables for FIR). For the tests
chosen to run in this thesis the simulation approach would also give good enough
results for both the recovery schemes. In addition, when using a simulator, it
is also possible to easily implement several extra schemes for recovery such as
full IP reconvergence and the theoretical local optimal shortest path towards a
destination.

5.2 Routing simulator

A routing simulator has been developed for this thesis. The routing simulator
is initialized with information about the topology and what type of traffic that
is to flow through the network in addition to which recovery scheme to use for
traffic recovery. It uses this information to decide which routers and links are
to be used for which traffic.

The topology layout is decided by information about what routers are part
of the network, and how are they connected through links. A link is defined by
the end points of the link, the cost to traverse it and its capacity.

Input information about the traffic to flow through the network is defined in
a traffic matrix. The traffic matrix consists of pairs of source and destination in
addition to the amount of traffic to flow from the source towards the destination.

The traffic is routed with normal shortest path routing whenever the sim-
ulator operates with a failure free network. If a failure is introduced to the
network, the recovery scheme is used to recover traffic that flows through the
network.
The simulator starts in its initial state, and maintains a list of predefined events. When all calculations for the initial state are finished, it starts to run through all predefined events from the list, immediately recalculating all traffic paths. For each step, including the initial, all state information for all nodes and links is recorded.

The traffic matrix is constant throughout the whole simulation, whereas the topology changes with each event. For more information about the traffic matrices used for this thesis, see Section 5.4. One event can consist of one of the three following changes:

- set state of one link to down
- set state of one link to up

In its initial state, all links in the topology are considered up, which enables the simulator to calculate all states in the failure free case. A typical run of the simulator is then to iterate through all links in the network, and set their state to down, one at a time. When the iteration is done, the simulation is over and all states have been stored.

The simulator developed uses link failures as the basis of all tests. The reason for this is that link failures is one of the largest types of failure in a typical IP network and it is therefore interesting to see how different schemes complied with such failures. It could also be possible to look at either only node failures or both node and link failures. This thesis focuses only one of the failure types. Another reason is that the FIR scheme does not support node failures. As previously mentioned, there is another proposal from the authors of the scheme that supports node failures as well, but that is not a focus of this thesis.

It is also important to mention that the framework always uses ECMP (see Section 4.1) as the first recovery mechanism for all the recovery schemes. It does however, not implement the loop free alternates as specified in [14]. This means that since only single link failures are considered in this thesis, the recovery schemes will only be in use if there are no more available next hops towards the destination in the normal forwarding table. That is, only one next-hop is specified in the normal forwarding table and the link towards that next hop has now failed.

ECMP is also constantly in use for load balancing of traffic. This means that all traffic is divided evenly among all available ECMP paths whenever such a path exists.

### 5.2.1 Capabilities and technical information

The routing simulator can be used to simulate all types of recovery schemes, such as proactive and reactive. It also supports both local and global recovery. In addition to the recovery schemes evaluated, some extra schemes have been implemented to be able to do a better evaluation.

The schemes implemented in the framework are the following:
CHAPTER 5. METHOD

- Full IP re-convergence
- theoretical local recovery scheme (explained below)
- IP Fast Reroute Using Not-via Addresses
- Failure Insensitive Routing

The theoretical local recovery scheme uses the normal routing tables for all traffic unless a failed link is defined to be used as the shortest path. Whenever such a link is encountered, the recovery scheme forwards the traffic through a tunnel which goes through the shortest path towards the destination without the failed link.

In addition, since the initial state of the simulator is always recorded, the simulator also enables the evaluator to compare results with the failure free case. The collection of statistical data is handled by the routing simulator and not the recovery scheme modules themselves, which means that all schemes are equally treated in the simulator. This ensures that the output data is consistent and easily comparable to other schemes.

The simulator reads as input topology files on the BRITE format [1]. For more information about the topologies used in this thesis, see Section 5.3. All bidirectional links should have two entries, one for each direction. The simulator also has traffic matrices as input which must define load for all source-destination pairs in the topology (see Section 5.4).

The output of the framework includes the following:

- per link information
  - total load traversing each link
  - exact load from each source-destination pair
  - capacity
- for each source-destination pair
  - hop count
  - path length
  - record if the pair is affected of an error or not
- total load in the network
- load percentiles
  - this is the link that has the amount of load that N\% of all links have a lower link load
  - calculated for all values in a 5\% interval (0, 5\%, 10\%, \ldots, 90\%, 95\%, etc.)
  - for each percentile, which link it is and the load of the specific link is recorded
Since all this information is recorded for each state (one link is dead at a
time), the simulator also provides minimum, maximum and mean values calcu-
lated based on all simulator states except the failure free.

The developed framework is implemented in Java and is easily extensible to
other recovery schemes.

5.3 Choosing topologies

To give credibility to the results shown in Chapter 5, the recovery schemes have
been tested on multiple topologies with different characteristics. Some of the
topologies are well connected and some of them are not. In addition, some of
the topologies are created with individual link costs, whereas others have a flat
cost for all their links. In addition, to be able to test how the recovery schemes
adjust to different topologies several synthetic topologies have been used. A list
of all topologies tested can be seen in Table 5.1.

5.3.1 Topology characteristics

Three main characteristics of topologies are widely used to describe a topology,
namely node degree, connectivity, the layout. The three characteristics are
described below.

The degree of a node is determined as the number of links the node has to
other nodes. An increase in the average node degree will result in more links
being present in the topology and as such give recovery schemes more options
available for finding a recovery path. In addition, a higher node degree will also
typically result in shorter recovery paths as there are more to choose from and
the network is better connected.

Another important characteristic of a topology or graph can be found by
using Menger’s theorem which characterizes the node-connectivity and edge
connectivity of a graph in terms of number of disjoint paths between vertices.
If you have a graph G which includes the vertices u and v, then an independent
collection of paths between the two vertices is defined as the number of separate
paths that do not share a vertex other than u and v themselves. The paths are
edge-independent if they share no edges. For a topology to support single link
failures, for all pairs of source and destination in the network there must exist
at least two edge-independent paths. For a topology to support single node
failures there must exist at least two node-independent paths. Since Not-via is
based on treating all failures as node failures, all topologies examined in this
thesis have at least two node-independent paths for all source-destination pairs.

The third main characteristic of a topology is its layout and this charac-
teristic may be completely different even for topologies where the two previous
characteristics are equal. The layout of a network describes how the topology
is connected, that is, which nodes are connected to each other. As an example,
if a low connected node is located at the edge of a network, its failure will not
influence very much traffic. However, if it is located in the core of the network,
it will possibly influence more traffic flows.
However, there are also other important characteristics of a network, namely the characteristics of the links connecting the nodes. The two main characteristics of links are the capacity and a metric of the cost for traversing the link.

The capacity of a link defines how much traffic the link is capable of transporting during a specified amount of time. It is often referred to as the maximum data rate for traffic traversing the link. In a digital environment the data rate is typically defined as number of bits per second. Whenever the traffic volume exceeds the capacity in a packet based network, thus causing the link to be congested, packets are typically dropped. Even though a link is congested, they are typically designed so that control messages still may pass the link to ensure that routing may continue.

In real-life networks the links often have a large diversity in their capacity. Some links are used as main paths for large volumes of traffic whereas others may either be used for recovery or only connect nodes which typically are not the source or destination for large traffic volumes.

Whenever a link fails in a network, the traffic previously traversing the link must be introduced to other paths which may or may not have room for this extra traffic. Such a situation may lead to an offered traffic volume that exceeds the available capacity of a link and thus causing the link to be congested.

The second important metric for links are the cost for traversing the link. This metric is used by routing algorithms to find the shortest path in a network. By setting the cost of a link, network operators can ensure that the network resources are used as they plan and thereby providing a better quality of service for the users of the network.

The Dijkstra algorithm used by many routing protocols to find the shortest path between two vertices in a graph has some constraints on how the cost of a link can be defined. First of all it must be a non-negative value as this will keep the algorithm from terminating. In addition, the edge to most likely be part of a path should have the smallest value.

There is not necessarily a connection between the capacity of a link and its cost since it is possible to use link costs to make sure that traffic follows specific paths instead of always using the ones with the largest capacity. However, routers from Cisco by default set the cost of a link \( i \) to

\[
m_i = \frac{1}{c_i}
\]

where \( m_i \) is the cost metric and \( c_i \) is the capacity of that link.

### 5.3.2 Real topologies used

Four real-life topologies have been chosen to be used for the evaluation process in this thesis. The four topologies are Abilene, COST239, GÉANT and UNINETT. Since the topologies (except COST239) are deployed they have typically been designed based on knowledge on what the specific topology could expect regarding traffic later to flow through the network. In addition, they
5.3. CHOOSING TOPOLOGIES

have usually been the subject of heavy evaluation by the network operators themselves and external experts. Since many studies use real-life topologies the use of them also make it possible to compare results across different studies.

The Abilene network is part of the Internet2 project and The Internet2 Backbone Network is run by Qwest, Indiana University, Juniper Networks and Nortel Networks. The network tested in this thesis has 11 nodes and 14 bidirectional links. All the links have weighted costs, but the costs are symmetric. The average node-degree of the topology is 2.55. The network is depicted in Figure 5.1. More information can be found at [4].

COST239 is in fact not a real deployed network, but it is a hypothetical pan-European topology heavily used in routing protocol and recovery scheme evaluations and as such it will be easy to compare results for this topology with other evaluations. With its 11 nodes, 26 bidirectional links and average node-degree of 4.73 it can be considered as a highly connected topology and is often used to see how routing protocols and recovery schemes behave in highly connected networks, which is also true for this thesis. The topology is depicted in Figure 5.2 and more information can be found at [2].

GÉANT topology is a result of an international collaboration of National Research and Education Networks representing 30 European countries, the European Commission and DANTE. The network is reserved for educational and research purposes. The version of the topology used in this thesis consists of 23 nodes and 37 bidirectional links. The average node-degree for the topology is 3.22. In Figure 5.3 the topology is depicted. The numbers for each link specify the cost. More information about the GÉANT project can be found at [3].

UNINETT is a Norwegian network implemented by UNINETT which is a collaboration of universities and colleges in Norway. It is constantly under development and updated information can always be found at [5]. The version of the topology tested in this thesis has 21 nodes and 32 bidirectional links and an average node-degree of 3.05. The topology is depicted with link costs in Figure 5.4.
CHAPTER 5. METHOD

Figure 5.2: COST239 topology

Figure 5.3: GÉANT topology
5.3. CHOOSING TOPOLOGIES

5.3.3 Generated topologies

To be able to test the recovery schemes on a diverse set of topologies, some have also been synthetically generated using publicly available tools developed for this.

The topologies generated for this thesis have been generated using Boston university Representative Internet Topology gEnerator (BRITE) [1]. BRITE supports several generation models such as flat AS, flat router and hierarchical topologies. The topologies used in this thesis have been generated using the Waxman model [62].

The Waxman-model proposed in [62] introduces two ways of generating a topology. The two methods are divided by how they define the distance between nodes. After defining the distance, both models use the same way of introducing edges which is by using a mathematical formula for probability.

The first method for creating a graph with the Waxman-model is to use a two-dimensional grid and for all \( n \) nodes generate two coordinates for every node \( i \) as \((x_i, y_i)\). Both \( x_i \) and \( y_i \) are generated using a uniform random distribution. The distance between the nodes is then calculated using the Euclidian metric, i.e. the distance between node \( u \) and node \( v \) is defined as

\[
d(u, v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}
\] (5.2)

The second method of finding the distance between the nodes is to choose
a value from a uniform random distribution in the range $(0, L]$ where $L$ is the maximum distance between two arbitrary nodes $u$ and $v$.

The Waxman-model then defines the probability of an edge connecting the nodes $u$ and $v$ as

$$P(\{u, v\}) = \beta e^{-d(u,v)/L^\alpha}$$  \hfill (5.3)

where $d(u,v)$ is the distance from the node $u$ to $v$, $L$ is the maximum distance between two nodes to be applicable for an edge between the nodes. $\alpha$ and $\beta$ are parameters in the range $(0, 1]$. Small values of $\alpha$ increase the density of short edges relative to longer ones. Large values of $\beta$ results in graphs with high densities on a general basis. The cost of the edge is set to the distance between the two nodes it connects.

Two types of such topologies based on the Waxman model have been generated for this thesis, both types with 32 nodes. The first, T2-32, has an average node degree of 4 and the second, T3-32, has an average node degree of 6. 10 topologies have been generated for each type. Both the topology types are at least 2-node connected which assures that any single link failure scenario can be recovered. To create these topologies, the default values in in BRITE was chosen which at the time at generation was $\alpha = 0.15$ and $\beta = 0.2$ and to get the correct average node degree $m$ had to be set to $m = 2$ or $m = 3$ for node degrees of 4 and 6 respectively.

5.3.4 Overview of chosen topologies

An overview of the topologies used in this thesis can be found in Table 5.1. In the table, the following legend is used:

- N: number of nodes
- L: number of links
- ND: node-degree
- W: weighted costs (links have different capacity)
- S: symmetric
- A: asymmetric
- F: flat capacity (equal for all links)
- I: individual (links have different capacity)
- R: real, deployed topology
- D: hypothetical designed topology
- SY: synthetically generated topology
5.4 Choosing Traffic Matrices

In addition to choose a topology it is also important to find traffic matrices that fit well to the topology to test the recovery scheme on. A traffic matrix (TM) describes the total traffic volume network-wide carried within a domain. Every element in the matrix corresponds to a pair of a source and a destination for the traffic and describes the demand for such a pair. A complete matrix will include demands for all possible source-destination pairs. The matrix does however, not say anything about how the traffic should be routed in the network as this is up to routing protocols and traffic engineering configurations of a network.

The values of an element in a traffic matrix will typically represent an average traffic volume over some time. The information needed to lookup a specific element in the matrix is just the tuple (source, destination). However, this will only work in a static environment where the values do not change over time. A dynamic approach defines a traffic volume for a source-destination pair for a specific moment in time. Basically the TM will then have three dimensions, namely (source, destination, time). When evaluating a topology for a specific moment in time, it will be the same as evaluating the same topology with a static traffic matrix with the same values as the dynamic matrix at that time. This thesis uses only static TMs.

Several network design tasks require a TM as input to be able to produce good results. Some examples are routing protocols, link weight setting protocols for IGP, capacity planning, reliability and failure analysis, bottleneck avoidance mechanisms, etc. The TMs can be used as input to both computations and simulations. It is therefore important to find TMs that fits the topology and design task that is to be evaluated [42].

As this thesis focus on how well recovery schemes cope with failures in the underlying topology, it is important to have a traffic matrix designed to fit the test case. Without a TM it is possible to find hop counts and path length for all source-destination pairs and their recovery paths for each failure scenario. However, it is not possible to find out whether the extra load imposed on a network following a component failure will cause the load traversing a link in the network to exceed its capacity.

Obtaining real life traffic matrices however, is not an easy task as they are generally unavailable. The reason is twofold and first of all network carriers consider TMs to be proprietary and confidential so they do not publish this

<table>
<thead>
<tr>
<th>Topology</th>
<th>N</th>
<th>L</th>
<th>ND</th>
<th>Cost</th>
<th>Capacity</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABILENE</td>
<td>11</td>
<td>14</td>
<td>2.55</td>
<td>W, S.</td>
<td>F, S</td>
<td>R</td>
</tr>
<tr>
<td>COST239</td>
<td>23</td>
<td>37</td>
<td>3.22</td>
<td>W, S</td>
<td>I, A</td>
<td>R</td>
</tr>
<tr>
<td>GEANT</td>
<td>21</td>
<td>32</td>
<td>3.05</td>
<td>W, S</td>
<td>I, S</td>
<td>R</td>
</tr>
<tr>
<td>UNINETT</td>
<td>32</td>
<td>64</td>
<td>4</td>
<td>F, S</td>
<td>F, S</td>
<td>SY</td>
</tr>
<tr>
<td>T2-32</td>
<td>32</td>
<td>96</td>
<td>6</td>
<td>F, S</td>
<td>F, S</td>
<td>SY</td>
</tr>
</tbody>
</table>

Table 5.1: Overview of topologies used in this thesis
information for general access. In addition, if access is granted to traffic information it may be easy to extract link load for all links in the topology through means of SNMP. However, to be able to extract information for traffic matrices, flow-level granularity of the data is in fact needed and this may definitely be more difficult. The Abilene Internet2 community [4] has collected flow-level statistics from their routers and this data is commonly used in the research community. The data is not in form of traffic matrices, but it is computable from the Juniper flow sampling data.

5.4.1 Generating Traffic Matrices

Since it is hard to obtain data from deployed networks it has become apparent to the research community that there is a need to generate traffic matrices. In addition, there is also a need to generate probable traffic matrices for synthetically generated topologies. However, how these traffic matrices have been generated has changed during the last few years.

In [42] they propose a way to be able to generate traffic matrices that fit to the underlying topology without the need to have statistical data for the specific topology. The solution has two basic steps, namely:

1. generate traffic load levels to use for source-destination pairs
2. map the traffic load levels to specific pairs of source and destination

Generation of traffic load levels

To be able to generate traffic load levels some probability distribution must be used. There are several distributions that can be used for this purpose, but in [42] they discover that three distributions fit recorded statistical data from the Abilene network and a backbone network of Sprint. The three are lognormal, loglogistic and inverse gaussian. However, they found that only lognormal was a good fit for all their tests. They also mentioned that a gravity distribution also could be used, but they did not investigate it further. However, the use of the gravity distribution for this purpose has been investigated in [52] and been found to fit even better than lognormal, even though it is easier to calculate and only needs one parameter, namely the mean. The lognormal distribution needs both the mean and the variance as input.

Before [42] was published, the use of a uniform distribution was common in the research community. However, the authors of [42] found that the uniform distribution does not fit the data extracted from real life networks and they strongly encourage other researchers not to use that approximation of traffic load levels.

In this thesis the traffic load levels are generated using the gravity model. The model is based on Newton’s formula for gravity between two physical objects and it is commonly used to model movement of people, goods or information between geographic areas. The formula is defined by Newton to be:
5.4. CHOOSING TRAFFIC MATRICES

\[ F = \frac{m_1 m_2}{d^2} \]  

(5.4)

where \( F \) is the force, \( m_1 \) is the mass of object 1, \( m_2 \) is the mass of object 2 and \( d \) is the distance between the objects. In a gravity model, the formula has been generalized to be:

\[ X_{ij} = \frac{R_i A_j}{f_{ij}} \]  

(5.5)

where \( X_{ij} \) is the matrix element representing the force to move from \( i \) to \( j \). \( R_i \) represents the repulsive factor of leaving \( i \) and \( A_j \) represents the attractive factors of entering \( j \). The friction of moving from \( i \) to \( j \) is represented by \( f_{ij} \). In [52] the author has found that it is possible to use this approach to create traffic load levels. In his proposal, traffic entering at node \( i \) and exiting at node \( j \) is defined as \( X_{ij} \). Here \( R_i \) is defined as the traffic entering the network through node \( i \) and \( A_j \) represents the traffic leaving the network through node \( j \). The author found that it was feasible to set the friction of moving traffic from node \( i \) to node \( j \) to a constant value for all pairs of \( i, j \). The author used this theory to propose a way of generating a list of traffic load levels.

5.4.2 Mapping traffic load levels to source-destination pairs

When the traffic loads levels are generated, the output is just an array of random numbers which follow a specific distribution and the length of the array is equal to \( N(N - 1) \) where \( N \) is the number of nodes in a topology. The number of source-destination pairs is not equal to \( N^2 \) because there will traverse no traffic from node \( i \) to node \( j \) when \( i = j \). This set of data could also consist \( N(N - 1) \) streams of traffic load level data if the data is dynamically generated using a time domain in addition to the space domain. However in this thesis only static distributions are discussed.

Every element in the set of traffic volume then needs to be mapped to a specific (source, destination) tuple. This can not be done at random since this will lead to either an infeasible distribution of load or at least the traffic matrix will be ill-matched to the underlying topology. By infeasible it is meant that in the failure free case, after examining the load of every link, at least one of them will have more traffic traversing it than it capacity allows. By ill-matched it is meant that a topology typically matches the traffic that flows through it, since it has over time evolved based on earlier measurements of network-wide load analysis. An arbitrary mapping of traffic load levels to source-destination pairs would not capture this effect.

In [42] the authors propose some basic desirable network properties to measure performance of mapping, which any TM should satisfy:

1. the TM should be feasible, i.e. not exceed any link capacities
2. the TM should not be “skewed”, i.e. not load any particular link excessively
They further propose two solutions based on different metrics to map the traffic load levels to specific source-destination pairs. The first one is called load minimization solution, and the second is called ranking metrics heuristics.

The load minimization solution is using a metric to minimize the congestion of a network, which is a common metric used for traffic engineering. It guarantees that the solution will not for any link exceed the capacity of that link, if such a solution exist. However, there are several problems with this approach. One, the problem that this solution tries to solve is a Generalized Assignment Problem and it is NP-complete [42]. Two, the solution maps to a specific routing. And the third and most important problem is that the solutions only focus on a single metric which may not be sufficient to provide a good fit for a TM.

The second solution, the ranking metrics heuristics is capable of mapping the traffic load levels to three different metrics. The basic of this heuristics is to generate two lists where the first list is created by taking $K$ samples of the source-destination traffic loads and order them descending. The authors refer to this as list1. Secondly, they create a list of source-destination pairs that are ranked in descending order according to how likely that specific source-destination pair is to carry a large traffic load, referred to as list2. Given list1 and list2, these are mapped on a one to one basis. That is, the $i$th entry in list1 is mapped to the $i$th entry in list2.

The authors then propose three different metrics to be used to generate list2 based on how carriers typically evolve their backbones. Based on their research of Abilene, half of the source-destination pairs contributed to 95% of the network-wide load, and for Sprint one third of the pairs contributed to 95% of the load. Thus they consider the top half of the ordered source-destination pairs to be the most important.

Their three metrics are all ordered in a descending manner, and are based on the following properties:

1. the total incoming/outgoing capacity of a node
2. number of links for each node
3. number of flows during a failure passing a node (inverse)

For the first two metrics, the minimum of both the nodes are considered as the value for that specific pair since it is the smaller node that determines the likelihood of the pair carrying a large source-destination load.

Regarding the last metric, it is based on routing information and can only be used if that is specified for the underlying topology. Since a topology typically has some nodes which are basically used for recovery traffic, these nodes will themselves not have large amounts of traffic entering or exiting the network through them. However, recovered traffic will typically pass through them and as such they will have a high number for this metric. It is therefore used as an inverse ($1/\text{flows}$) so that recovery nodes will be ordered last based on this metric.

The traffic matrices used in this thesis is based on the ranking metrics heuristics.
5.4.3 Initialization of capacity

In some of the synthetically generated topologies, the traffic matrices generated did not match the capacity of the links. The capacities of the links in the networks were therefore scaled so that the results would fit nicely into the charts. The capacity was set to have the same value for all links, and the link cost was set to a constant value. A test was then run to find all link loads in the failure free case. This result was used to find the maximum link load for that topology. The capacity of all links was then set to the value such that the maximum load found was two thirds of the capacity. That is:

\[ C = \frac{2}{3} \max(\text{load}) \]  

(5.6)

where \( C \) is the capacity and \( \max(\text{load}) \) was the maximum load found in the failure free case.

5.5 Scheme implementations

This section describes the implementation choices that have been taken during development because of both undocumented features and the need to fit the recovery schemes to the simulator.

5.5.1 IP Fast Reroute Using Not-via Addresses

The IP Fast Reroute using Not-via addresses recovery scheme has been implemented based on the Internet Draft in [9]. However, not everything is defined in the draft, therefore some implementation choices had to be made and this section describes those choices.

The Not-via scheme by default support both node and link failures. However, the authors specify that all implementations of Not-via should choose to consider all failures as node-failures. This is done in the simulator as well. However, whenever it is impossible to reach a specific destination by using the node-failure recovery mechanisms, the authors propose to use link-failure recovery. This is typically whenever there is a last-hop failure. Both strategies are implemented in the framework. The method chosen is based on checking the next-hop specified in the forwarding table against the destination of the traffic; if they match the link-protecting mechanism is used. The same effect can be achieved in real life IP networks since during generation of forwarding tables in link-state routing protocols, all information for all prefixes are known.

Something that is not specified in the Internet Draft for Not-via is whenever the next-hop has an ECMP path towards the destination. This means that there will be more than one next-next hop. There were several ways to cope with this.

One way would to choose an arbitrary node at random as the next-next hop or base it on the lowest ID.
Another way could be to choose the path that had the lowest load/capacity on its links. However, this would not be possible to precalculate since such loads are typically dynamic. It could however be based on knowledge of previous traffic matrices and use this data to guess the load in the network.

Yet another way would be to choose the one with the shortest path from the current node (the node which discovered the error). This is possible because the current node already has a shortest path entry towards the next-next hop defined by the Not-via address for that node Not-via the next-hop for the current node.

The method that was implemented was to divide the traffic evenly among all the next-next hops, by creating several IP-in-IP tunnels. This was possible to do without doing extra SPF calculations at all based on the same argument as the previous method.

5.5.2 Failure Insensitive Routing

The FIR scheme has been implemented more or less exactly as specified by the authors of [31] [40]. However, some changes had to be made to make it fit the simulator.

One thing is that none of the optimized algorithms proposed by the authors, namely ASPF and IASPF, have been implemented. Instead, the algorithm for finding key links as shown in Algorithm has been used to create the results. The reason for this is that calculation time does not have any impact on the results of this thesis. In addition, both interface specific forwarding and backwarding tables were calculated and stored for every node since there were no space constraints from the simulator. Whenever a node encounters an error, the backwarding table is used.

The authors propose that only the key link closest to the destination should be considered down since this would create the same result as considering all of them as down and would lead to lower calculation times for both ASPF and IASPF algorithms proposed in their article. However, since this was also just a proposed optimization and had no impact on the results the implemented version FIR consider all key links as down.

A FIR-enabled network is typically located in the core part of an operator’s network. This means that no traffic will typically emerge from the nodes in the network itself, but instead be sent to the nodes from hosts connected to each node. However, since the input data used for the simulator is only specified as source-destination pairs, there is no incoming interface for the source. Therefore, an additional table was used on the source node, namely the ordinary forwarding table. If the failed link was the only one listed in the ordinary forwarding table, the backwarding table was used instead of the ordinary forwarding table. All other nodes on the path use their interface specific forwarding tables.
Chapter 6

Evaluation

6.1 General

The discrete event simulator detailed in Section 5.1 has been used to conduct several tests to see how well the recovery schemes focused on in this thesis perform in different types of topologies.

6.1.1 Schemes put to the test

For each test and topology there will typically be a comparison of five different schemes, both theoretical and proposed recovery schemes. First of all the failure free case will be used as a comparison for the other schemes. In addition, the schemes will be compared to the topology and routing that exist after a full IP reconvergence has taken place.

The third scheme is a theoretical scheme not possible to implement in a real environment, but used only for theoretical comparison. It is a mix of full IP reconvergence and a local recovery scheme. When the network is in a failure free state traffic traverses the network as normal. If however, a link goes down, the recovery scheme is activated. The router that detects the failure tunnels the traffic using the shortest path towards the destination without using the failed link.

The two last schemes are the IP Fast Reroute using Not-via addresses and Failure Insensitive Routing which are implemented as per the description in Section 5.5.

6.1.2 Test types

Three different types of test have been conducted for these five schemes. The first test investigates the path lengths achieved by the recovery schemes. The second test looks at the link loads for all the links in the network. And the third test focuses on the total load in the network before and after a failure.
CHAPTER 6. EVALUATION

| FF | Failure free scheme |
| RE | Routing after full IP reconvergence |
| LO | Local optimal shortest path towards destination |
| NV | IP Fast Reroute using Not-via addresses |
| FIR | Failure Insensitive Routing |

Table 6.1: Abbreviations used

The first test investigating the path length will provide results for both hop counts and the sum of all link costs of traversed links referred to as path length. It focuses on both synthetically generated networks and real life IP networks. The results are presented in Section 6.2.

The second test focuses on how the failure of a single link affects the load on each link in the network when some of the traffic is following recovery paths. The results are given in Section 6.3.

The last type of test investigates the total load in the network before and after the failure of single links. The loads are compared to the failure free case or the fully reconverged network to find the increase for all schemes. The results are given in Section 6.4.

Section 6.5 gives a thorough comparison of the path choices of Not-via and FIR which explains why there are differences in the results shown the rest of this chapter.

6.1.3 Regarding test output

The output obtained from the simulator provide the same type of information for all schemes described above. They have all been created by failing a single link at a time and recording all state information in the network after every event. This output has then been used to get the results presented in this chapter.

It is again important to note that the simulation framework developed always uses ECMP both for load balancing and as the first measure against failures. This is thoroughly described in Section 5.2. For load balancing, all traffic is evenly divided amongst all available ECMP paths. Regarding recovery, the only times the recovery schemes are put to the test is when only a single link is listed as the next hop for a destination, and that link has failed. If several ECMP paths are available, all traffic are routed through these paths.

Because of the space constraint of some tables and plots, abbreviations have been used throughout this chapter and they are presented in Table 6.1.

6.2 Path length of affected paths

The first test conducted investigates how long the paths are after doing recovery of traffic. The length of recovery paths provides information about how well the
recovery schemes are able to find short paths to use for recovery. Short paths will typical result in fewer links being in use and as such the overhead when doing recovery will be low. Low overhead when doing recovery will make it less possible that the recovery will create congested routers and links.

Real topologies are investigated in addition to a comparison across several synthetic topologies. The real topologies are compared both with regards to hop count and path length. Since the synthetically generated topologies have a constant value for link cost they are compared based on hop count.

Only affected source-destination pairs are part of the results. Affected pairs are the source-destination pairs whose traffic in the failure free case traversed the now failed link. The reason for this is that most of the traffic is not at all affected, and this means that it would be very hard to see any differences between the different recovery schemes if the path length for all source-destination pairs were to be plotted. As an example, for the generated 32-node networks with a node-degree of 4, only about 2.5% of the paths are affected.

The path is measured all the way from the source towards the destination. The reason for measuring the whole path is described in Section 4.4.4 and is basically because the Not-via and FIR scheme consider different end points of the recovery path. Not-via considers the next-next-hop as the endpoint of the recovery path whereas FIR finds a recovery path towards the destination node.

The use of hop count as the basis for this test is only interesting whenever all links in the network have the same constant value for the cost of traversing the link. For topologies with individual cost for all links, the lowest hop count of recovered traffic will not necessarily lead to the shortest path in terms of summing up all costs for the traversed links.

The layout of the real topologies which have individual cost have been used to generate topologies with the same layout by a flat cost for all links. The number of nodes and links are the same, and the links still connect the same nodes as before. This made it possible to compare the results for hop count for more networks. However, the results should only be considered informational, since by not using the original link costs, the design of the network regarding traffic engineering are not taken into consideration.

Regarding the numbers that make up the base for the plots, they refer to the maximum path length for each source-destination pair. Traffic may very well be splitted into several substreams and as such only parts of the traffic may be affected of an error. This happens whenever there exist two or more paths with equal cost towards the destination. However it is the maximum value for path length that is measured, which will in such cases be the path that was in fact affected. The reason for this is that it is interesting to know the worst case of the recovery path lengths as network should be designed with worst case recovery paths in mind in addition to how the network performs in the failure free case.

One note about this test is that every source-destination pair may very well contribute more than once to the final result. The reason for this is that one source-destination pair may be affected by several link-failures if the original path consists of more than one hop.
CHAPTER 6. EVALUATION

Figure 6.1: Hop count distribution for affected paths for the network 'COST239', LO=Not-via=FIR

6.2.1 Hop count of real topologies (flast cost)

The three plots depicted in this section shows the unmodified COST239 topology in addition to the modified Abilene and UNINETT topologies. They are modified so that all link costs are set to the same constant value as described above. The reason for this is that the lowest hop count does not necessarily mean the shortest path for networks where all links have individual values for cost.

In Figure 6.1, the plot for the COST239 topology is depicted and it shows that both Not-via and FIR chose the exact same paths as the local optimal scheme. The topology is highly connected and this together with the fact that the link costs are constant throughout the topology most errors will in fact be fixed by using ECMP paths. The few times it was not possible to use ECMP paths, all schemes were able to constantly find the shortest path to use for recovery because it was always possible to find a path with the same length as the local optimal even by removing either a node (Not-via) from the topology or all key links (FIR). This can be seen by removing any link or node from the topology in Figure 5.2.

Regarding the three schemes local optimal, Not-via and FIR and the paths that in the failure free case had a hop count of 1, 96% of them got a new hop count of 2 and 4% had a new hop count of 4. For paths with original hop count of 2, 93% got a new path with hop count 3, and the rest got hop count 4. For those paths which originally had a hop count of 3, all got a new hop count of 4.

For the modified versions of Abilene and UNINETT the results are shown in Figure 6.2 and Figure 6.3. In both these topologies, there is a distinct difference between Not-via and FIR where FIR typically has shorter paths. That is typically because first of all, Not-via considers most failures as node-failures, and secondly, their endpoint of their recovery paths are typically different, as described in 4.4.

For example, in the Abilene network from Figure 6.1, consider sending traffic from Seattle to Kansas City and then the link between Seattle and Denver goes down. FIR will then send the traffic through Sunnyvale and Denver to
6.2. PATH LENGTH OF AFFECTED PATHS

Figure 6.2: Hop count distribution for affected paths for the network 'Abilene' (modified), LO=FIR

Figure 6.3: Hop count distribution for affected paths for the network 'UNINETT' (modified), LO=FIR


<table>
<thead>
<tr>
<th>Topology</th>
<th>FF</th>
<th>LO &amp; FIR</th>
<th>Not-via</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abilene</td>
<td>3.05</td>
<td>4.97 (62.95%)</td>
<td>5.29 (73.44% - 6.44%)</td>
</tr>
<tr>
<td>COST239</td>
<td>1.95</td>
<td>2.91 (49.23%)</td>
<td>2.91 (49.23% - 0%)</td>
</tr>
<tr>
<td>GÉANT</td>
<td>3.07</td>
<td>4.94 (66.91%)</td>
<td>5.21 (69.71% - 5.47%)</td>
</tr>
<tr>
<td>UNINETT</td>
<td>3.19</td>
<td>4.85 (52.03%)</td>
<td>5.52 (73.04% - 13.81%)</td>
</tr>
</tbody>
</table>

Table 6.2: Mean of hop count of affected paths compared to failure free case

the destination Kansas City. However, since Not-via considers the whole node Denver as down, it will send the traffic via Sunnyvale, Los Angeles and Houston before reaching Kansas City.

A similar example can easily be found in the UNINETT topology from Figure 5.4. Consider sending traffic from Stavanger to Ålesund and then the link between Stavanger and Bergen is down. The FIR-scheme will simply send traffic via Haugesund, Bergen and then reach Ålesund. However, Not-via will consider the node in Bergen as down, and as such it will send traffic via Kristiansand, St. Olav (Oslo), Oslo and Trondheim before reaching Ålesund which is quite a trip through Norway compared to sending traffic via Haugesund.

A table comparing the average hop count for the affected pairs is shown in Table 6.2. The values are the average hop count, and the increase compared to the failure free case is noted in the paranthesises. Since the local optimal scheme and FIR have the exact same result in this test, they are listed in the same column. For Not-via the increase compared to the local optimal scheme is also listed. As can be seen, Not-via yields a path that in average is up to 13.81% longer than FIR for the UNINETT topology which is a significant increase in path length.

6.2.2 Path length stretch of real topologies

The results in the previous section was based on modified versions of the deployed topologies. In this section, the topologies are used as they are defined which is with individual link weights and as such the results can not be presented as hop counts since this does not give a correct picture.

The length of the recovery path length has been compared to the path length in the failure free case. That is, the path for all affected source-destination pairs have been compared to the path length of the same source-destination pair in the failure free case.

To make the results easier to view, the results have been spread out based on the hop count in the failure free case. This means that for all the plots, all the path length increases with the same hop count in the failure free case have been plotted on the same horizontal line. Since there were some extreme values the x-axis is shown with a logarithmic scale.

The networks Abilene, GÉANT and UNINETT networks are shown in Figure 6.4, Figure 6.5 and Figure 6.6 respectively.
6.2. PATH LENGTH OF AFFECTED PATHS

Figure 6.4: Path length stretch for affected paths for the network 'Abilene', LO=FIR

Figure 6.5: Path length stretch for affected paths for the network 'GÉANT'
Figure 6.6: Path length stretch for affected paths for the network 'UNINETT'

In the Abilene network the FIR scheme found exactly the same paths as the local optimal scheme. In the two other networks it found mostly the same paths as the local optimal, but the results were not exactly the same. Regarding Not-via it found mostly the same paths, but for some affected source-destination pairs, it had longer paths in all three topologies.

What can be seen from the plots is that the Not-via scheme tends to have more values in the right of the plot. This means that for some source-destination pairs, Not-via has found a recovery path which is considerably longer than in the failure free case. The reason for this is again the same as previously mentioned and it has been detailed in Section 6.5.

6.2.3 Hop count across multiple synthetic topologies

The motivation for this test was to measure how well the different schemes performed with different layouts of a topology. Some schemes may perform well with some layouts, but not very well with others. This is an interesting characteristic of recovery schemes and this test has been created to find exact values for this.

To be able to measure the schemes based on this motivation, ten different layouts were used, with the same amount of nodes and links and as such they have the same average node-degree. The hop count distribution was recorded for each layout and the mean and standard deviation for each discrete value for hop count was calculated. In addition, values for mean +/- standard deviation were calculated. This made it possible to create plots with three lines for all schemes: the mean value alone in addition to the mean value plus/minus the standard deviation. The lower the standard deviation, the less distance there will be between MEAN-STD and MEAN+STD, which will visually create a thinner line. For the schemes, it is better to have a low value for the standard
6.2. PATH LENGTH OF AFFECTED PATHS

![Figure 6.7: Hop count distribution for affected paths for the T2-32 networks](image)

![Figure 6.8: Hop count distribution for the failure free case](image)

The tests were run for both the topologies with an average node-degree of 4 and 6 as described in Section 5.3. However, the results for the topologies with an average node-degree of 6 were very similar for all the schemes. The reason for this is that whenever more links are available, it is easier to find and optimal path to use for recovery.

For the topologies with an average node-degree of 4 there were some differences, and the mean value for the hop count distribution across the layouts are showed in Figure 6.7. Here one can see that the Not-via scheme typically had a higher hop count than the FIR scheme. The FIR schemed performed equal to the local optimal scheme in all tested layouts and as such the local optimal scheme is not shown here.

The plots as described above can be seen in Figure 6.8, 6.9, 6.10 and 6.11.

From the results one can see that the failure free case does not vary very much at all. This is not the case for the results for the fully reconverged network. It varies quite a bit because the paths will be very different since the fully reconverged network can choose a different path all from the source, whereas...
CHAPTER 6. EVALUATION

Figure 6.9: Hop count distribution for a fully reconverged network

Figure 6.10: Hop count distribution for the FIR scheme

Figure 6.11: Hop count distribution for the Not-via scheme
the local recovery schemes always traverse the network towards failed link first.

The FIR scheme seems to be less dependent on the layout of the underlying topology, compared to Not-via. This is made even clearer in Figure 6.12 where only the standard deviation for Not-via and FIR is plotted. This plot shows that the standard deviation for Not-via is typically higher than the one for FIR and as such it seems that FIR is the recovery scheme that is less dependent on the layout of the underlying topology.

The reason why FIR will vary less is that it typically only considers one or a few links as dead whereas Not-via considers a whole node. When removing a whole node from the network, a lot of possible recovery paths are removed and as such, the recovery scheme will more often find longer paths around the failure. This means that Not-via will sometimes find short paths, just as FIR, and sometimes it will find very long paths around a failure and as such it varies quite a lot compared to FIR.

### 6.3 Link load

For network operators to be able to provide a good quality of service for their customers it is important that their network can constantly be able to transport all the offered traffic, even during failures. Whenever the offered load becomes larger than the capacity of a link, packets are dropped and this reduces both the quality of service and the throughput of the network.

This test investigates the link load during single link failures using different recovery schemes. For each state change in the network, the link loads of each link is recorded, and the maximum link load for that state is recorded. The reason for using the maximum link load is that network operators need to plan their networks for a worst case scenario.

The traffic matrices used have been generated based on the specification in Section 5.4.

The values used to create the plots are calculated with the following formula:
\[ f = \frac{l}{c} \]  \hspace{1cm} (6.1)

where \( l \) is the offered load, \( c \) is the capacity of the link and \( f \) is the fraction of a link that is in use based on the capacity. If the value of \( f \) exceeds 1, it means that the link is congested.

In Figure 6.13, the results from the Abilene topology is shown. The plot shows that the Not-via scheme typically has a higher link load for many of the failed links. The reason for this is that since the recovery paths of Not-via are longer than the ones with FIR as shown in the previous section. This means that the load will traverse more links and thus increasing the total load of those links.

However, it is important to notice that both the Not-via and the FIR scheme have nice characteristics for distributing load. Whenever a link that in the failure free case was used for large amounts of traffic, this was distributed to other, not so heavily loaded links.

This can be seen in the left part of the plot by comparing the gap between the failure free case and the recovery schemes. The gap is very small which means that the links that carried large loads in the failure free case has not been chosen to carry large loads when doing recovery. It can also be seen that no links were congested in this test as the fraction \( f \) (shown on the y-axis) never exceeds 1.

To the right in the plot one can see that there is a large gap between the link load carried of a link in the failure free case and the maximum link load of that link considering all states tested. The reason why this gap is large is that the less loaded links are located close to heavily loaded links and were used as recovery paths when recovering traffic from the heavily loaded links.

Since the traffic is routed through different paths when comparing Not-via and FIR, the link loads are also different. In some cases the maximum link load in the network is less when using Not-via than with FIR even though FIR finds shorter paths. The reason for this is that the two recovery schemes choose different paths, and as such it is not always that the same links carry load for recovered traffic.
6.4. TOTAL LOAD

The last test in this thesis investigates how large the total load in the network becomes when using different recovery schemes. Networks are typically designed to fit the load of the network in the normal case, but they should also be designed to cope with the extra loads that is introduced when links or routers go down. The previous test investigated this on a per link basis, and this test shows the aggregated load in the whole network.

The total load in the network has been found by adding the load traversing every link in the network. The link that is considered down does naturally not contribute to this sum, as that link load is always zero.

The traffic matrices used are the same as the ones used for the link load tests have and are described in Section 5.4.

When investigating the total load of a network it is important to know that whenever traffic traverses more links, it will also increase the total load. This means that since there is no guaranteed connection between the hop count and the shortest path this test can not use the original real topologies. Therefore the modified topologies with a flat link cost for all links have again been used to produce the results.

In the following plots, the total load using each recovery scheme has been compared to the total load in the network after a full IP reconvergence. Since in the typical case the use of a recovery scheme is only temporary, it is interesting to see how this temporary load increase affects the whole network.

From Figure 6.14, 6.15 and 6.16 one can again see that the FIR scheme performs equal to the local optimal scheme. The Not-via scheme has in some cases a higher total load then FIR. As previously mentioned, this is because of the path chosen by the two recovery schemes are different and as such Not-via will let the same load contribute to the total load more times than FIR.

The total load of the synthetically generated topologies has also been investigated. For each of the ten generated topologies the minimum, mean and maximum total load has been found. This has been done for both the topologies
Figure 6.15: Total loads network 'UNINETT' (modified), LO=FIR

Figure 6.16: Total loads network 'GEANT' (modified), LO=FIR
6.5. PATH CHOICES

<table>
<thead>
<tr>
<th>Node degree</th>
<th>Minimum</th>
<th>Mean</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Mean</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE</td>
<td>0.000%</td>
<td>0.005%</td>
<td>0.031%</td>
<td>0.000%</td>
<td>0.012%</td>
<td>0.017%</td>
</tr>
<tr>
<td>LO &amp; FIR</td>
<td>0.005%</td>
<td>0.842%</td>
<td>4.018%</td>
<td>0.012%</td>
<td>0.414%</td>
<td>2.502%</td>
</tr>
<tr>
<td>Not-via</td>
<td>0.005%</td>
<td>0.893%</td>
<td>4.350%</td>
<td>0.012%</td>
<td>0.416%</td>
<td>2.502%</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td>0.000%</td>
<td>0.012%</td>
<td>0.017%</td>
<td>0.002%</td>
<td>0.414%</td>
<td>2.502%</td>
</tr>
<tr>
<td>Mean</td>
<td>0.002%</td>
<td>0.414%</td>
<td>2.502%</td>
<td>0.012%</td>
<td>0.416%</td>
<td>2.502%</td>
</tr>
<tr>
<td>Maximum</td>
<td>0.017%</td>
<td>2.502%</td>
<td>2.502%</td>
<td>0.017%</td>
<td>2.502%</td>
<td>2.502%</td>
</tr>
</tbody>
</table>

Table 6.3: Increase in total load compared to failure free case

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>1820.21</td>
<td>1567.20</td>
</tr>
<tr>
<td>Mean</td>
<td>1914.16</td>
<td>1603.64</td>
</tr>
<tr>
<td>Maximum</td>
<td>2039.35</td>
<td>1637.82</td>
</tr>
</tbody>
</table>

Table 6.4: Total load in the failure free case

that have an average node degree of 4 and 6. The result can be found in Table 6.3.

For all these tests, the FIR scheme performed equally to the local optimal scheme, so their result is shown in the same column. In general, the increase in total load of all the test scenarios is not significant for any of the schemes evaluated and they both perform either close to or equal to the local optimal scheme. This is expected for networks with such good connectivity.

Another thing that can be seen in Table 6.3 is that both the mean and maximum increase in load is less for the networks with an average node degree of 6 compared to those with an average node degree of 4. The reason for this is that in a network with better connectivity it is easier to find good recovery paths and as such be closer to the failure free case.

For the same reason, the total load in the failure free case is less for a network with an average node degree of 6 compared to the ones with 4. This can be seen in Table 6.4. Whenever a path consists of less hops, the total load decreases.

6.5 Path choices

6.5.1 Path choices of local optimal scheme and FIR

Since FIR performed either equal to or extremely close to the local optimal case for all test topologies, it would be interesting to see why FIR performs so good, and this section identifies which cases where FIR performs equal to and different from the local optimal case.

When looking at Figure 6.17 where the link 4 → 6 is down and a packet is sent from node 1 to node 6, one can see the path that the local optimal case will use, which is also the same path as FIR would choose. In the case of FIR,
when the packet reaches node 4, it will find that $4 \rightarrow 6$ is down and therefore look up destination 6 in its backwarding table for the link $4 \rightarrow 6$ and find node 2. Then, when the packet reaches node 2, it will have already calculated the correct path towards the destination, where it has taken into consideration that the link $4 \rightarrow 6$ is a key link.

One can then look at the network depicted in Figure 6.18. Again, a packet is sent from node 1 to node 6. The path choice of the local optimal scheme when link $2 \rightarrow 5$ is down, is $1 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 6$. This means that the local optimal case has a hop count of 5 and a path length of 7. However, in that network and for that specific error, FIR will not perform as good as the local optimal scheme.

This can be seen in Figure 6.19 where the path is shown that the path it takes has the same hopcount, but the path length is 8. The reason for this is that FIR considers the link $5 \rightarrow 6$ as a key link and therefore unusable. When traffic reaches node 2, it will bounce back to node 1 based on node 2’s backwarding table. This is still the same path as the local optimal. However, when node 1 receives traffic from node 2 with 6 as the destination, it knows that either the link $2 \rightarrow 5$ or $5 \rightarrow 6$ is down, and it considers both as key links and unusable. It has therefore no other choice than to send traffic around both these links, using the path $1 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 7 \rightarrow 6$.

The following requirements must be met for FIR not performing as good as the local optimal scheme:

1. The KeyLinks-algorithm must return at least two key links for the given destination.
2. One of the key links, must in fact not be down, but be part of the local optimal path.

Whenever both those conditions are met, FIR will perform sub optimal. The
6.5. **PATH CHOICES**

By looking at Figure 6.20 one can see a scenario where the Not-via performs equal to the local optimal scheme. All link costs have the same constant value and a packet is sent from node 1 to node 9. Even though Not-via considers node 6 as down and all the links attached to it, it does not have any influence on the path length towards the destination and the hop count for both schemes is 8.

However, by adding one link between node 5 and 6 as seen in Figure 6.22, the Not-via scheme will not perform equal to the local optimal anymore in the same scenario. Here, Not-via still have a hop count of 8, but the local optimal path shown in Figure 6.21 will decrease to 7 hops.

The following requirements must be met for Not-via not performing as good as the local optimal scheme:

1. Not-via must consider a node as down, and not only a link

**6.5.2 Path choices of local optimal scheme and Not-via**

By looking at Figure 6.20 one can see a scenario where the Not-via performs equal to the local optimal scheme. All link costs have the same constant value and a packet is sent from node 1 to node 9. Even though Not-via considers node 6 as down and all the links attached to it, it does not have any influence on the path length towards the destination and the hop count for both schemes is 8.

However, by adding one link between node 5 and 6 as seen in Figure 6.22, the Not-via scheme will not perform equal to the local optimal anymore in the same scenario. Here, Not-via still have a hop count of 8, but the local optimal path shown in Figure 6.21 will decrease to 7 hops.

The following requirements must be met for Not-via not performing as good as the local optimal scheme:

1. Not-via must consider a node as down, and not only a link
2. The node Not-via considers down must be part of the local optimal shortest path

Whenever both these requirements are met, Not-via will not perform as well as the local optimal recovery scheme.

### 6.5.3 Comparison of Not-via and FIR path choices

There are two main differences in how Not-via and FIR chooses the paths.

First of all, FIR considers only the failed link as down, where Not-via considers the neighboring node down. In the general case, this leaves fewer options for Not-via to use for recovery. However, based on the calculation of key links, FIR can also consider more links than the one that is in fact down, and as such it can not consider all the same paths as the local optimal scheme.

The second big difference in path choices is that Not-via finds a path to the next-next-hop whereas FIR finds a path towards the destination. This may in some cases lead to completely different path choices.
Chapter 7

Summary and further work

This section starts by relating the thesis to the problem statement introduced in Chapter 2.

Next it will cover the main findings of this thesis. This will give a short explanation of the different recovery schemes and their functional characteristics. It will also describe the routing simulator developed for this thesis.

Lastly it will describe the findings when testing the recovery schemes on both deployed and generated topologies with different characteristics.

The last section will describe topics discussed in this thesis that would need further work and topics that could increase the value of this research.

7.1 Summary

In Chapter 2 the following problem statement was defined:

Discuss and evaluate different recovery schemes that provide fast reroute in connectionless IP networks

As described and motivated for in both the introduction and in Chapter 4, the IP Fast Reroute Framework is a good framework for recovery schemes that fit the problem statement. As such, two recovery schemes was chosen, namely IP Fast Reroute Using Not-via Addresses (Not-via) and Failure Insensitive Routing (FIR). The reason for why these two was chosen is that they are both implementations of the IP Fast Reroute Framework which means that they are local, proactive schemes working in a connectionless environment.

The concept of the Not-via recovery scheme is that whenever a link fails, it will consider the neighboring node as down. It will then find a path towards the next-next-hop towards the destination from the path in the failure free case. It does this by using IP-in-IP tunneling. When the traffic reaches the next-next-hop it is decapsulated and forwarded towards the destination.
CHAPTER 7. SUMMARY AND FURTHER WORK

The FIR scheme on the other hand is able to infer network failures by looking at the flight of a packet. The flight of a packet refers to the path it takes through the network. Based on this the scheme is able to proactively create forwarding tables that makes sure that traffic never traverses failed network elements. It does this by identifying possibly failed links (key links) whenever a packet arrives on an interface that is not normal for the packets destination.

Two of the characteristics that this thesis aimed to evaluate for the recovery schemes was their choice of recovery paths and how much extra load they introduced in the network when doing recovery.

Regarding path choices there are two main differences between the recovery schemes. The Not-via scheme considers a whole node as down, whereas FIR considers the key links identified as down. In addition, Not-via aims to find a path to the next-next-hop of the original path, whereas FIR finds a path directly towards the destination.

It has been shown that for most networks, FIR will provide shorter recovery paths than Not-via. Since longer recovery paths leads to more links being used for recovery traffic, Not-via will introduce a larger amount of load than FIR.

The functional characteristics of the recovery schemes have also been evaluated. Both schemes are more complex than ordinary shortest path first routing. The Not-via scheme has a computational complexity of 5 to 13 SPF calculations for networks ranging from 40 to 400 nodes. The authors of the FIR scheme propose two possible implementations. One has a complexity of 8 to 16 SPF calculations for networks from 30 to 200 nodes. The other implementation has heavier demands for memory during calculation, but it has only the complexity from 1 to 5 times an SPF calculation for the same type of networks.

An important feature of the Not-via scheme is that it is able to recover from both node and link failures, whereas the FIR scheme is only able to recover from link failures.

Regarding requirements to the network topologies they both need the network to be either two-link connected for link failures and for Not-via it needs the topology to be two-node connected. Regarding requirements to hardware/software of routers the FIR scheme requires more changes than Not-via.

To evaluate the recovery schemes they were also tested on different network topologies, both synthetically generated and real life networks. The traffic matrices for all networks was synthetically generated. The choice of the networks and generated traffic matrices was explained in Chapter 5.3.

To be able to compare performance of the recovery schemes in both real life and synthetic networks, a routing simulator was developed. It was able to compare the two recovery schemes with each other in addition to compare them to the failure free case, a full IP reconvergence and a theoretical recovery scheme which finds the shortest path from a failed link to the destination.

It has found that IP Fast Reroute using Not-via addresses would probably be a better choice if a scheme should be implemented in hardware since it has less requirements for doing so. This scheme also has better coverage than the Failure Insensitive Routing scheme and as such would be a better choice for any network operator.
7.2 Further work

The simulator that was developed does not support the use of multiple routing layers and as such it is not fit to evaluate recovery schemes such as MRC [27]. However, it is nothing that the design of the simulator prohibits, so support for this is definitely possible to implement in the future.

Since there were large differences between a scheme that considered single link failures as node failures and one that considered it as link failures it would be interesting to investigate the FIR-based recovery scheme proposed in [64]. This scheme is able to protect from node errors as well and a comparison between this scheme and Not-via would be interesting to perform.
Bibliography


# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Sample network</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Shortest path tree from the network in Figure 3.1 with node A as source</td>
<td>12</td>
</tr>
<tr>
<td>3.3</td>
<td>Sample network with B-C link down</td>
<td>20</td>
</tr>
<tr>
<td>3.4</td>
<td>Connection oriented local protection with A-B link down</td>
<td>21</td>
</tr>
<tr>
<td>3.5</td>
<td>Connection oriented global protection between A-F</td>
<td>21</td>
</tr>
<tr>
<td>3.6</td>
<td>Recovery Cycle introduced in [56]</td>
<td>23</td>
</tr>
<tr>
<td>3.7</td>
<td>Topology to illustrate last hop problem</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Topology to illustrate ECMP</td>
<td>31</td>
</tr>
<tr>
<td>4.2</td>
<td>Topology to illustrate Loop Free Alternates</td>
<td>32</td>
</tr>
<tr>
<td>4.3</td>
<td>Not-via repair of router failure</td>
<td>33</td>
</tr>
<tr>
<td>4.4</td>
<td>The set of Not-via P Addresses</td>
<td>34</td>
</tr>
<tr>
<td>4.5</td>
<td>Fragmentation of a packet fit to MTU when using a tunnel</td>
<td>37</td>
</tr>
<tr>
<td>4.6</td>
<td>Topology used for the illustration of FIR</td>
<td>38</td>
</tr>
<tr>
<td>4.7</td>
<td>Topology to show loops in FIR-networks</td>
<td>44</td>
</tr>
<tr>
<td>4.8</td>
<td>Recovery path for node-protecting schemes</td>
<td>46</td>
</tr>
<tr>
<td>4.9</td>
<td>Recovery path for link-protecting schemes</td>
<td>46</td>
</tr>
<tr>
<td>4.10</td>
<td>Recovery path for Not-via</td>
<td>47</td>
</tr>
<tr>
<td>4.11</td>
<td>Recovery path for FIR</td>
<td>47</td>
</tr>
<tr>
<td>4.12</td>
<td>Recovery path for Not-via when it has the best path</td>
<td>48</td>
</tr>
<tr>
<td>4.13</td>
<td>Recovery path for FIR when Not-via has the best path</td>
<td>48</td>
</tr>
<tr>
<td>5.1</td>
<td>ABILENE topology</td>
<td>59</td>
</tr>
<tr>
<td>5.2</td>
<td>COST239 topology</td>
<td>60</td>
</tr>
<tr>
<td>5.3</td>
<td>GÉANT topology</td>
<td>60</td>
</tr>
<tr>
<td>5.4</td>
<td>UNINETT topology</td>
<td>61</td>
</tr>
</tbody>
</table>
6.1 Hop count distribution for affected paths for the network 'COST239', LO=Not-via=FIR ............................. 72
6.2 Hop count distribution for affected paths for the network 'Abilene' (modified), LO=FIR ............................. 73
6.3 Hop count distribution for affected paths for the network 'UNINETT' (modified), LO=FIR ............................. 73
6.4 Path length stretch for affected paths for the network 'Abilene', LO=FIR ............................................. 75
6.5 Path length stretch for affected paths for the network 'GÉANT' ............................................................. 75
6.6 Path length stretch for affected paths for the network 'UNINETT' .......................................................... 76
6.7 Hop count distribution for affected paths for the T2-32 networks ......................................................... 77
6.8 Hop count distribution for the failure free case ......................................................................................... 77
6.9 Hop count distribution for a fully reconverged network ............................................................... 78
6.10 Hop count distribution for the FIR scheme ............................................................................................ 78
6.11 Hop count distribution for the Not-via scheme ...................................................................................... 78
6.12 Hop count distribution and STD - FIR vs. Not-via ............................................................................... 79
6.13 Link loads network 'Abilene', LO=FIR .................................................................................................... 80
6.14 Total loads network 'Abilene' (modified), LO=FIR ................................................................................. 81
6.15 Total loads network 'UNINETT' (modified), LO=FIR ........................................................................... 82
6.16 Total loads network 'GÉANT' (modified), LO=FIR ................................................................................. 82
6.17 FIR performs equal to local optimal ...................................................................................................... 84
6.18 FIR does not perform equal to local optimal - Local optimal path ......................................................... 84
6.19 FIR does not perform equal to local optimal - FIR path ........................................................................ 85
6.20 Not-via performs equal to local optimal ................................................................................................. 85
6.21 Not-via does not perform equal to local optimal - Local optimal path .................................................... 85
6.22 Not-via does not perform equal to local optimal - Not-via path .............................................................. 86
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Link state database for network shown in Figure 3.1</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Sample network 1 - routing table for node A</td>
<td>12</td>
</tr>
<tr>
<td>3.3</td>
<td>Node F label switching table (IN - OUT)</td>
<td>14</td>
</tr>
<tr>
<td>3.4</td>
<td>Node H label switching table (IN - OUT)</td>
<td>14</td>
</tr>
<tr>
<td>3.5</td>
<td>Sample of forwarding tables for nodes C, F, H and I</td>
<td>15</td>
</tr>
<tr>
<td>3.6</td>
<td>Overview of recovery mechanisms</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>Legend/notation used for the FIR scheme</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>Complete list of key links for the FIR test network shown in Figure 4.6</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Interface specific forwarding table for node 1</td>
<td>42</td>
</tr>
<tr>
<td>4.4</td>
<td>Interface specific backwaring table for node 1</td>
<td>43</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview of topologies used in this thesis</td>
<td>63</td>
</tr>
<tr>
<td>6.1</td>
<td>Abbreviations used</td>
<td>70</td>
</tr>
<tr>
<td>6.2</td>
<td>Mean of hop count of affected paths compared to failure free case</td>
<td>74</td>
</tr>
<tr>
<td>6.3</td>
<td>Increase in total load compared to failure free case</td>
<td>83</td>
</tr>
<tr>
<td>6.4</td>
<td>Total load in the failure free case</td>
<td>83</td>
</tr>
</tbody>
</table>
LIST OF TABLES
Appendix A

Glossary

downstream Consider a stream of data from source A to destination B and an intermediate node N where the data packet currently is located. Then all nodes and links on the path from N to B inclusive are said to be downstream.

downstream

downstream Consider a stream of data from source A to destination B and an intermediate node N where the data packet currently is located. Then all nodes and links on the path from N to B inclusive are said to be downstream.

edge See link.

edge

egress A border node of a domain where traffic exits the domain.

egress

flight A packets path through a network.

flight

forwarding The act of looking up routes and transmitting packets.

forwarding

forwarding table See routing table

forwarding table

FIB Forwarding Information Base.

FIB

FIR Failure Insensitive Routing.

FIR

IGP Interior Gateway Protocol.

IGP

ingress node A border node of a domain where traffic enters the domain.

ingress node


IS-IS

link A point-to-point connection between two adjacent nodes

link

LDP Label Distribution Protocol.

LDP

LSP Label Switched Path.

LSP

LSR Label Switch Router.

LSR

MPLS Multiprotocol Label Switching.

MPLS

MTTR Mean Time To Repair.

MTTR

MTBF Mean Time Between Failure.

MTBF

MRC Multiple Routing Configurations.
NIC  Network Interface Card.

node  An endpoint of a graph or a junction common to two or more edges in a graph. In network terminology the word node is often used for the network element router.

Not-Via  IP Fast Reroute Using Not-via Addresses.

router  A network layer device that uses one or more metrics to determine the optimal path along which network traffic should be forwarded. Routers are interconnected by links and serve as control points in the network.

routing table  a table which holds information about next hops for each destination in a network

routing  The process of determining routes for where to send packets to able to move packets towards their final destination.

upstream  Consider a stream of data from source A to destination B and an intermediate node N where the data packet currently is located. Then all nodes and links on the path from N to A inclusive are said to be upstream.