UNIVERSITY OF OSLO
Department of Informatics

IP REDUNDANT TREES FOR PREPLANNED RECOVERY IN CONNECTION-LESS NETWORKS

Master thesis

Ole Kristoffer Apeland

1st November 2006

UNIVERSITAS OSLOENSIS • MDCCCXI •

Abstract

Most networks are inherently prone to failures, and failures do occur on a regular basis. New real-time services, relying on continuous connectivity, are regularly introduced on the Internet - requiring new demands to be met, often extending beyond the Internet's original design goals. Traditionally, recovery has been covered by the IP re-convergence process, which is a lengthy process offering recovery time in the range of seconds. In this thesis IP Redundant Trees (IPRT), a new method for providing IP fast recovery, is introduced. It is based on the redundant tree approach presented by Medard et.al [1], extended to provide a resource efficient way to populate the recovery Forward Information Base (FIB) and furthermore, the mechanisms needed to utilize this information in the forwarding procedure in order to provide local recovery. IPRT is evaluated through the use of graph theory in the initial design phases, and simulations on several real and synthetically generated networks. The evaluation shows that one of the strongest assets of IPRT is the ability to provide 100 % coverage with a minimal and constant amount of extra state information in each router.

Acknowledgments

Working with this thesis has been both interesting and hard work. I would especially like to thank my supervisor, Tarik Cicic, for constant encouragement and valuable input, and my wife for her love and patience through the long hours. I would also like to thank Amund Kvalbein, Tommy André Nykvist, Audun Fosselie Hansen and other people who contributed comments to this work. Furthermore, I would like to thank family and friends for all help and support.

Contents

Chapter 1

Introduction

1.1 Motivation

Over the years, the role of Internet has shifted from scientific use towards acting as a key contributor in both business and recreational services. As services converge and are accommodated by the Internet, the society depends increasingly on the reliability of computer networks. An example may be drawn from the healthcare industry where new services such as Telemedicine and remote diagnostics has stringent requirements on network reliability as any failure may have serious consequences. The versatility of the Internet Protocol (IP) combined with the popularity and vast deployment of the Internet has lead to a migration of services traditionally closely tied to specialized distribution network. Telecommunication services are now being integrated in computer networks through VoIP services and by adding IP capabilities to existing distribution networks such as found in UMTS releases 4 and 5. Furthermore, television and radio companies are starting to use Internet as a distribution channel. It appears that an increasing number of communication systems will be using Internet as a distribution channel.

By moving services from their old technological platforms and introducing them to the Internet, new demands need to be met, often extending beyond original design goals. Old Internet services such as e-mail and web browsing require moderate connectivity, whereas many of the newly introduced services require a more continuous connectivity. This places a higher demand on the network in terms of availability and reliability.

With the increasing popularity and task diversity of the Internet, there is a steady and significant growth in the volume of data transported [2]. In a high-speed network, a failure may render information obsolete or data lost if the network lacks the ability to quickly reroute traffic. Some applications rely on a robust packet delivery service, however, reliability is always desirable and sometimes required. Due to the amount of services accommodated by the Internet it is possible that a single node or link failure may be of negative consequence. Thus, the growing number of services, traffic volume and the ever-increasing dependency on the Internet magnifies the consequences of a network failure and strengthen the need of reliable operation.

Networks are inherently prone to failures, and they do occur on a regular basis in most networks. Sprint's IP backbone was analyzed and results show that 20 percentage of the links had a Mean Time Between Failure (MTBF) of less than 1 day, and 70 percentage of the links had a MTBF of less than 10 days [3]. The failures may originate from a multitude of reasons and have various scale and severity. They may be either physical or logical, and arise from either external or internal causes. An example of an external physical error could be a cable cut, while an internal logical error could be caused by an erroneous configuration. Furthermore, not all errors occur by accident but are rather the result of preplanned service events, e.g. hardware upgrades or large configuration changes. Studies as [4] and [5], show that the single most common cause for failure originate from scheduled maintenance e.g. upgrades, installation or configuration changes. Other significant contributors to failure were power outages, link failures and hardware (router) failures, and combined these unscheduled failures contribute to about 80 % of all failure situations. In addition, it is shown that most failures are short lived.

Because most of the failures are short lived, it seems to be a good idea to implement recovery mechanisms that are able to deal with all failure situations without reconfiguring the whole network. In addition, such mechanisms may act as a buffer providing more time for the network to properly address the failure when it is long-lived.

Assuring a robust network, routing infrastructure has been in focus for quite some time and is a widely investigated area. Nevertheless, former reports have been to a large degree focused on connection-oriented networks. However, solutions for traditional IP networks are not absent. Historically, one of the main goals of the DARPA Internet Architecture was that communication must be able to continue despite loss of network components. This was realized through fate-sharing between the communicating hosts and dynamic distributed interior gateway protocols (IGP). IGP Link State (LS) routing protocols, such as OSPF or IS-IS are the most adopted IGP. LS routing protocols relies on a periodic flooding of a topology information from each router in the network. These messages are then used to form a global view of the topology at each router and further compute the next hop for the traffic.

When a link or node failure occurs in a traditional IP network a wide IP re-convergence is initiated. During this re-convergence all routers in the network are made aware of the new topology and subsequently realized in the routing table of all routers. During this period there may arise inconsistencies where routers disagree on the paths of packets as a consequence of the transition, from the old routing table to the new, that may be asynchronous between the routers. This may lead to microloops, a situation where some traffic is sent back and fourth between two routers in the network. Thus packet drop may occur either through congestion on links suffering from microloops, depleted "time-to-live" in IP headers, or router logic refusing to forward traffic arriving on unexpected incoming interfaces. Traditionally the re-convergence period has been a time-consuming process in the order of seconds. For a traditional IP network with an LS routing protocol the duration needed for the network to return to a coherent state, i.e. recovery time, is equal to the time needed to detect a local topology change, e.g. link-is-down, flood a new Link State Packet and compute the new routes at each node. Several solutions on how to reduce the recovery time have been proposed, and together they address all of the different steps towards recovery.

There are two ways to detect failures in a network; either at the lower level or through exchange of HELLO messages. Link-level detection is the faster method of the two, but it is of limited use as it only detect link failures, i.e. the link may be reported working even if the router logic has failed. The two methods complement each other in terms of fault detection time and thus both method are used to identify failures. HELLO messages may detect both link and node failures but its fault detection time is bound by the HELLO intervals. The detection time can be reduced by reducing the interval time. By using both approaches, the average failure detection time may be kept as small as possible without introducing excessive amounts of HELLO messages to the network. Nevertheless failures may be fluctuating, i.e. last for only a short period of time, and reporting a failure to soon or to frequently can lead to route flaps and stability issues [6].

There has been proposals to reduce the response time by reducing the queuing delays of the LSP traffic and use optimized flooding mechanisms (cite RFC4222). The computation time has also been addressed with optimizations in the algorithms used to compute the new routing tables i.e. incremental routing table updates. However, even with these optimizations, the IGP convergence process might prove too slow for the new Internet services, and it is likely that the requirement for recovery time might be in several orders less of these achievements. The wide IP re-convergence is inherently slow because it only responds to a failure after it is detected and requires every router in the network to respond to the failure. In a recent study[7] it was shown that it was possible to achieve sub-second recovery time, in the range of 0.3 - 1 second. However, this required carefully tuned parameters; The performance were dependent on the number of nodes in the network, the topology, the link propagation delays and, when incremental routing table updates are used, the number of prefixes in the network.

In order to meet the new demands in recovery time new methods need to be utilized. Proactive recovery provides a recovery strategy where all the alternative backup routes are pre-calculated in advance of any possible failure. It allows the failure to be addressed locally without the immediate need to inform other routers of the failure and thus prolonging the period available to perform the time consuming global failure signaling and path calculations. This allows the disruption time to be reduced to the time needed to detect the failure and invoke the recovery rerouting, which is consistent with a timeframe of tens of milliseconds.

IETF has presented such proactive recovery mechanisms in the IP Fast-Reroute Framework, where the recovery is based on single or multi-hop repair paths. The solutions presented in the framework are similar to MPLS Fast Reroute but the mechanisms for providing the backup routes in pure IP networks are necessarily very different. In addition, there has been some work lately on Multi Topology (MT) routing. Where the design has been focused on creating virtual recovery-topologies upon which the routing protocols could act. The main idea being that should a link fail one could choose from one of the virtual topologies and their routing tables not containing the failed link and reroute the traffic according to the selected topology.

However, because most of the research in the field of resilient networks has been focused on providing solutions for connection-oriented networks there are relatively few mechanisms that may address recovery in connectionless networks. The difference of connection-oriented networks, and connectionless networks, i.e. conventional IP networks is in essence how a path is represented. In connection-oriented networks the source router have full control on the path a packet will follow through the network, while connectionless networks share the responsibility of directing a packet to its destination among all routers in the network. Thus, the recovery schemes developed for the two kinds of networks will rely on inherently different mechanisms.

One of the solutions that has been developed to be applicable to connection-oriented networks is the redundant tree (RT) model presented by Medard et.al. [1]. The method has been presented in conjunction with both WDM[8] and MPLS[9].

The main idea of the RT method is to construct two directed trees, named red and blue, in such a way that in case of a single node or link failure a source node is still connected to all operational destinations through either the red or the blue tree. Thus, if a pair of red an blue trees are generated for each source node in the network, every source node may reach all other operational destinations in the network in the event of single-failures.

1.2 Important findings

In this thesis IP Redundant Trees (IPRT), a new method for providing IP fast recovery, is introduced. It is based on the redundant tree approach presented by Medard et.al [1], extended to provide a resource efficient way to populate the backup Forward Information Base (FIB) and furthermore, the mechanisms needed to utilize this information in the forwarding procedure.

The core of IPRT is the r/bTables, the recovery FIBs created by a procedure to extract only essential routes from redundant trees while retaining the functionality and properties of the trees. The procedure allows a number of FIBs equal to two times the number of nodes in the network to be reduced to a constant additional state equal to two additional FIBs at each node. Thus, it reduces the footprint of the RT method in the FIB and may allow the per-packet signaling to be resource effective.

Furthermore, an additional data-structure named Qbit is introduced to support local recovery in a connectionless environment. This is a data-structure that may be merged with the FIBs or be self-contained as an addition to the FIBs. It in addition to support local recovery Qbit but may also allow the forwarding procedure to choose the shortest recovery path if it is available. However, at this point the QoS properties of Qbit is of limited use as a theoretical gain is proven but the effects observed in simulations provide only minimal gain.

1.3 Organization

The rest of the thesis is organized in the following manner; In the background chapter general recovery and routing strategies are introduced along with related work. The main contributions are found in the chapter named IPRT Design. In this chapter all nessesary modifications and design choises needed for RT to be applicable to conventional IP networks are identified and accounted for. Furthermore, the IPRT tree generator and the extensions to the J-sim network simulator needed in order to simulate IPRT is described in the implementation chapter. In the subsequent chapter, the network models key characteristics are identified and alternative methods are accounted for. This is followed by a chapter where the experiments are described, the results obtained from simulating different IPRT enabled networks are shown, and subsequently, the results are analyzed. In the last chapter the conclusions are presented along with future work.

Chapter 2

Goals

The principal goal for the thesis is to research applicability of the RT method to conventional IP networks. We set three goals that need to be reached if the redundant tree recovery mechanism is to be applicable to connectionless IP networks:

1. The IPRT mechanism needs to be able to co-exist with normal routing protocols in times of failure-free operation.
2. The method needs to be able to support local recovery in a connectionless environment. This is because the use of global recovery would generally be too slow to counter any failure before IP re-convergence finishes, and thus void the use of a recovery mechanism.
3. The method needs to provide a mechanism for path representation, i.e. represent the redundant trees. This path representation needs to be unaffected of the re-convergence subsequent to a failure.

Another goal of the thesis is investigate if the IPRT method may be applied in a resource efficient way. The original RT method needs a number of trees equal to twice the number of nodes in a network to be able to provide preplanned recovery. Memory is a very expensive resource in the routers and thus it is a goal to try to reduce the memory footprint of the recovery method.

Furthermore, if it can be proven that the RT method may be applied in conventional IP networks, a sub goal of this thesis is to investigate the performance of the redundant trees and see how the method compares to alternative solutions. I.e. investigate the length of repair paths and how the recovery traffic affects the traffic load distribution in a network during failure.

Chapter 3

Background

In this chapter, general recovery and routing strategies are introduced along with related work. The different methods and approaches to achieve self-healing networks are presented and classified. Furthermore, a short overview of the IP routing algorithms, and how they may be configured or altered to provide a better quality of service, is given. In addition, the original Redundant Tree (RT) algorithm is presented as well as Menger's theorem, which forms the foundation for the qualities found in the RT method. Related work is also presented, introducing Resilient Routing Layers and the IP fast reroute framework.

3.1 Classification of recovery mechanisms

There are many different methods and approaches to achieve self-healing networks, but they are all generally classified by three different criteria.

Criterion	**Schemes**
Recovery	Protection
	Restoration
Computation	Distributed
	Centralized
Scope	Global
	Local

Table 3.1: Classification

The main criterion, classifying the different recovery mechanisms, is characterized by at what stage a recovery is initialized. Protection schemes are based on the idea of having pre-calculated alternative paths in advance of any failure, whereas restoration schemes only calculate new paths after a failure has occurred. As a result, restoration methods generally require more time to respond to a failure situation, consequently adding to the total recovery time. With protection schemes, all the alternative paths need to be registered in the network in advance.Thus, protection schemes add to the state information required in the nodes. Moreover, it is a higher computational cost associated with protection schemes because of the calculation of alternative paths regardless of failures. With both approaches it might be necessary to reserve additional resources in the network, e.g. bandwith. Theoretically, restoration may be able to avoid to reserve the resources in advance of a failure and rather adept to spesific topology after the failure has happened. However the normal approach for both methods is to have the additional resources available in advance of a failure, e.g. by overprovisioning router and link capacity. Consequently, the choice between protection and restoration schemes becomes a tradeoff between router state, computation, and recovery speed.

The computation of routes can be achieved either by a centralized scheme or by a distributed scheme. Both methods should offer the same amount of connections that could be restored. Centralized schemes rely on a master to calculate the route for every router in the network, and allow for a dedicated server, and thus reduce the required computations in nodes. The master must maintain a full global knowledge of the network. This knowledge of the network topology and available resources allows for complex algorithms to be deployed and may provide an advantage in terms of efficient use of resources. However, the centralized scheme has an important weakness; if the master server fails or get sperated from parts of the network - the whole scheme may fail. In addition this solution requires the server to receive periodic and timely information on the present topology to be able to produce accurate and correct results. Distributed schemes are more adaptable to failures in the network, and allow for greater scalability than a centralized scheme. The major drawback is that distributed schemes are more complex. Distributed schemes need to exchange messages in order to have sufficient information to coordinate the recovery, and the converging of this information needs to be proven to ensure stability in the system. As with centralized schemes, the distributed approach may produce inferior or incorrect results if the decisions are based on inconsistent or stale information. In addition, distributed system adds to the workload imposed on the nodes.

Both centralized and distributed schemes may be used in conjunction with either protection or restoration schemes. However, even in all combination are possible, centralized is often used togheter with protection and distributed are used with restoration. In centralized schemes every enquiry to the master introduce a potential delay associated with retrieving the information. This can affect the initial setup-time of a connection when a protection scheme is used, e.g. if specific paths are to be signaled to the nodes from the ingress node. When used with restoration schemes the potential delay adds directly to the total recovery time, and thus the combination is less suited for recovery. Distributed schemes can be used in conjunction with either protection or restoration schemes. As with centralized schemes there is a potential delay to get the information associated with distributed schemes, but because of its local nature the delay is of a lesser degree.

The last criterion divides the schemes by the scope of the recovery. Global recovery requires the ingress node to have the main responsibility for recovery operations, while local recovery relies on a shared responsibility between all nodes in the network. Global recovery covers link and node failure by calculating an alternative end-to-end path. The secondary path may or may not share nodes or links with the primary path. Global recovery utilizes path rerouting, which roughly implies tearing down the original primary path and then reestablish a new end-to-end route. This method has a global nature which lets it spread the alternative paths over the entire network, and allows for a better resource utilization than local recovery. Local recovery lets the fault be handled by the nearest upstream node of the faulty node or link. The connection is reestablished by routing around a neighbor node or link to avoid the node or link presumed faulty. Furthermore, link rerouting does not need to utilize signaling to the ingress node, and thus has the potential to operate on a smaller time frame than global recovery.

3.2 IP Routing

In a traditional IP network packets generally need to traverse multiple network elements to reach the destination. To accommodate this movement, routers use a method called "store-and-forward" to transport the packet to the destination. In addition, the responsibility of determining the path is distributed between all the routers in the network. I.e. when a packet travels from its source towards the destination, each router locally determines the next-hop of a packet and forwards it accordingly. This decision is done for every packet since the best route may have changed since the previous packet was forwarded. To be able to forward packets each router maintains a *forward information base* (FIB), also referred to as a routing table. Each FIB contains the next-hop destination for each reachable IP prefix in the network. Routing is the act of constructing and maintaining the FIB, while forwarding is the act of using the FIB to decide and subsequently transmit the packet towards its next hop.

There are two main ways to provide routing functionality in a network; trough manual configuration or through an algorithmic approach. In small and simple networks, it may be desirable to construct the paths manually. This static and centralized approach works well in environments where the network traffic is predictable and the topology is simple. In small networks the paths may be relatively easy to design and understand. However, the method does not scale well as the FIBs may become cumbersome to maintain by hand, as networks grow in both size and complexity, specially since all changes in the network or traffic patterns need to be addressed manually. Today most routing protocols are automated through routing algorithms which are distributed and operate in a dynamic manner. The protocols configure and adapt the FIB dynamically as destinations are advertised or discovered, and are thus more robust than static routing. In addition, dynamic routing algorithms may have an adaptive property where routing decisions reflect changes in other aspects than topology changes such as changes in traffic patterns. However, some static and centralized aspects are still maintained to accommodate traffic engineering.

Generally, all dynamic routing protocols need to perform some basic operation. One of these is to observe the local network parameters. The needed parameters differ between the routing protocols, but typically a router needs to advertise its presence and discover its neighbor nodes. This operation also enables the router to discover the state of each of it's adjacent node, e.g. if the node is up or down. Other typical network parameters may include link utilization or link delay. The measurement may be continuous, triggered or periodic depending on the desired resolution and response time of the metrics and states. The second operation involves dissemination of this information throughout the network to let each router create a view of the topology and other desired properties of the network. The exact method used for a router to inform all routers of it's local view of the topology is dependent of the routing protocol, but is often achieved through some periodic or triggered reliable flooding. This information is stored at each router, typically if a dedicated data structure other than the FIB is used. Another operation involves the route computation. Based on the information gathered and the routing algorithm used, the routers generate paths with minimum cost towards each possible destination (see section 3.3.1). Finally, if needed, the routers adapt the FIB to reflect any changes discovered from the last dissemination process.

Furthermore the dynamic routing protocols are divided into one of two main families of routing protocols; *distance vector* (DV) and *link state*(LS). Of these two families of routing protocols usually LS is used in conventional IP networks.

3.2.1 Distance Vector

Distance Vector routing protocols rely on the distributed Bellman-Ford algorithm and the exchange of a small amount of information. Routers running a DV protocol are expected to maintain a vector of each reachable IP-prefix in the network and their associated cost and next-hop. Furthermore, each router must measure its local network parameters and update their vectors accordingly. The cost is often a measurement of hop-count, but DV allows it to take on any form and there are implementations using other metrics than hop count [10]. To disseminate the information each router sends their vectors to their adjacent routers whom in turn checks the received vectors against their own and updates it if necessary. This technique gives the DV family of protocols a "local" view of the topology where no node holds the complete view of the topology. In addition, the convergence time becomes directly connected to the network diameter and the timing of the periodic messages.

The "local" property makes DV protocols sensitive to component failures and inclined to populating the routing tables with old and invalid route information. Consider a network with a node A and B, where A is providing the only link towards a subnetwork and B is a direct neighbor of node A as shown in Figure 3.1. Given a failure on the link towards the subnetwork, node A would update it's routing table indicating no link towards the subnetwork. Now suppose node B sends out its vector before A. A would then incorrectly learn that B has a route to the subnetwork and update it's table accordingly. This would result in a loop where the cost of reaching the unreachable sub-network would increase towards infinity.

Several proposals have been proposed to counter the "count-to-infinity" problem i.e. "infinity-defintions" or "split-horizon" found in [11] and [10]. However the "infinity-definition"

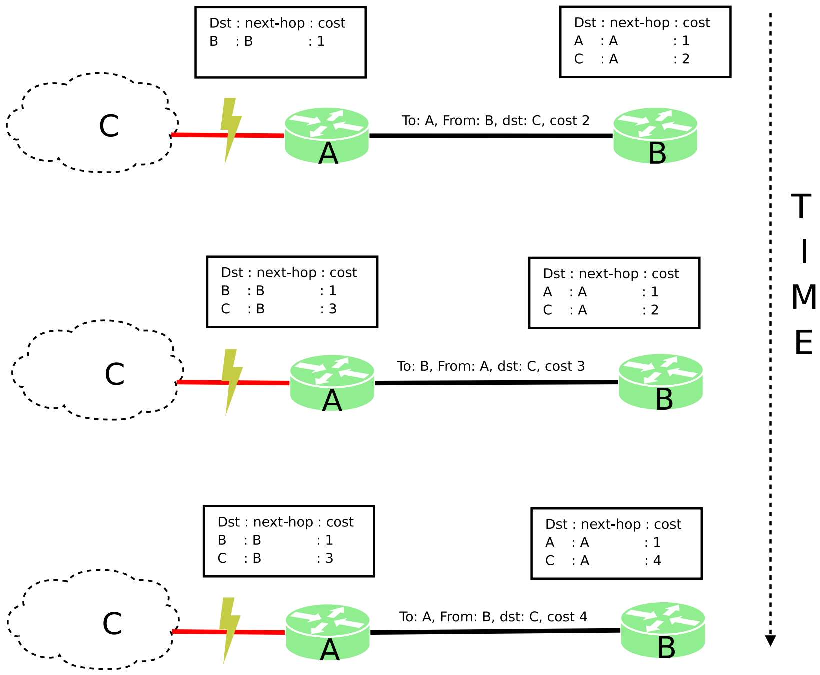

Figure 3.1: Distance Vector cost problem

results in a max-hop count on the paths, and thus imposes a restriction on the max-diameter of a network. Furthermore "split-horizon" provides with rules that minimizes the chances of looping problems, but only partially solves the problem. Overall the short-coming makes DV only usable for small networks.

3.2.2 Link State

In Link State (LS) routing protocols each router in the network is expected to at least identify its adjacent nodes and distribute this information periodically or triggered to all other nodes in the network. The information is usually disseminated through a reliable flooding mechanism providing a guaranteed delivery to all nodes. The nodes store this information in a *Link State Database* (LSDB). The reliable delivery of messages and LSDB provide all nodes in the network with a synchronized global view of the whole network. This enables LS protocols to perform more complex routing calculations, making it easier to calculate paths that are more sophisticated. In addition, all the routers compute the new paths in parallel.

When a failure occurs this may trigger a node to send out a *protocol data unit* (PDU), i.e. an IP packet containing LS routing information, reflecting the new topology. Although micro-loops may occur when the LSDB of two nodes are not synchronized they are short lived compared to the failure situation that may be experienced in DV protocols. In addition LS protocols provides solutions for hierarchic routing where a network are divided into several smaller zones, and thus the convergence time may be reduced.

The major drawback for LS protocols is the amount of CPU time needed to compute the routing tables. Updates are received on a regular basis and thus routers spend much time recomputing their FIB. However, much work has been done in this area, such as incremental route computations and optimized algorithms.

3.3 Default Route Computation

In many of todays routing protocols the shortest-path class of algorithms forms the foundation for route computation. This class of algorithms allows each node to choose the paths of minimum distance based on the routing information collected. There are many implementations on finding the shortest-path where the most known are the Dijkstra and the Bellman-Ford algorithms. Dijkstra is often used with LS protocols, e.g. OSPF [12], while Bellman-Ford is widely used in DV protocols, e.g. RIP [11].

3.3.1 Algorithms and metrics

Some of the shortest-path algorithms may be used in weighted graphs and allow for both dynamic and static adaptation to changes in the network properties, e.g. may be used for traffic engineering. A router may know of several adjacent paths to a destination, and sometimes the shortest path does not always utilize the network resources in the most efficient way. A common approach to achieve a higher quality of service in the routing protocols, is to associate a cost with each link in the network and let the routing procol compute the shortest path to each destination using these cost as length of the link. The cost of a link is often derived from one or more metrics. The metrics used are dependent on the implementation of the routing protocol but usually metrics as number of hops, delay, available bandwidth, traffic and reliability are used. An example may be found in OSPF [13] where a network engineer may manually assign a cost to each link in the network. There are also possibilities to let the routing protocol automaticly adapt the cost as found in early ARPAnet [14]. However, in practice, metrics are mostly assigned manually as automated assignment may induce route flaps or poor overall performance.

Common metrics

The most common routing metric is path-length. This metric is often measured in the number of hops needed to reach the destination. However, it is also possible for network administrators to assign weights to each link in the network, where path-length is measured in the total weight of a path.

Another common route metric is delay. This measures the time required for a packet to traverse the path from source to destination. Delay is affected by many aspects in the network including the physical distance of the path, the bandwidth of the links traversed, congestion, queue length at each router etc. Because delay reflects several important aspects of a path, it is a commonly used and very useful metric.

Bandwidth is also a useful metric as it may tell the available capacity of a link. E.g. when constructing paths a 100-Mbps Ethernet link would be preferable over an ISDN line. However, if the faster link has a high load preferred selection might be reversed as a result of the load and time needed to access the link. Thus, it may be useful to measure available bandwidth instead of the total link capacity. However this requires a more active measurement, e.g. trough probing, and the measured results becomes stale after a relatively short period. Therefore, available bandwidth it is not widely adapted as a metric, and usually over-provisioning are used as a "guarantee" against congestion. Furthermore, there are ongoing research to adept the weights on links to enhance the networks ability to honor increasing demands in traffic and thus provide large parts of the potential gains of traffic enginering through dynamicly adapting weight on links[15].

3.3.2 Dijkstra algorithm

The Dijkstra algorithm was first introduced in 1959 [12]. It allows for computing shortest path from a single source to multiple destinations. In addition it may be used in conjunction with non-negative weighted graphs.

There are several implementations of this algorithm. A common implementation is to make the algorithm find the shortest path from a source S to any other node in a graph with nonnegative arcs. This is done by iteratively growing the set of nodes of which it already know the shortest path. At each step of the algorithm, the unknown vertex with the smallest distance is added to the set of known vertices. This vertex's neighbors then get their distance updated to equal the distance of the node being treated plus the link to each neighbor weight, but only if this weight is lower than the weight already known at the neighbor vertices. The algorithm then loops and process the next vertex not in the known set until all are known. The running time of this algorithm is $O(n^2)$.

3.4 Connectivity and Menger's theorem

Informally, a graph is a finite set of vertices connected by links called edges. A graph is connected if there is a path connecting every pair of vertices, and furthermore two vertices are adjacent if they are connected by a single edge.

At minimum, the network needs to be link-redundant (two-edge connected) if one wants to recover from a link failure, and node-redundant (two-vertex connected) if one wants to recover from a node failure; E.g. for a graph to be deemed link- or edge-redundant, the graph needs to be connected after removing a single given edge or vertex, respectively. Articulation point, or single point of failure, is a point in the network where a failure would disconnect two parts of the network. Failure in articulation points cannot be remedied by any of the protection or recovery schemes.

Menger's theorem

Let G=(V,E) be a graph and A,B $\in$ V. From Menger's theorem it follows that the minimum number of vertices needed to be removed to separate A from B in G is equal to the maximum number of node-disjoint A $\rightarrow$ B paths in G. Likewise the number of edges needed to be removed to separate A from B in G is equal the the maximum number of edge-disjoint A $\rightarrow$ B paths in G.

This gives us that for any two vertices in a vertex or edge redundant graph there exists at least a pair of vertex or edge disjoint paths respectively.

3.5 Redundant Trees

The redundant tree (RT) model presented by Medard et.al. [1] is a pre-planned protection scheme for global recovery. The method is applicable to connection-oriented networks, and has been presented in conjunction with both WDM[8] and MPLS[9]. Furthermore, RT utilizes a centralized approach for computation of the recovery paths, and may be used to protect against both node and link failures.

The main idea of the RT method is to construct two directed trees, named red and blue, in such a way that in case of a single node or link failure a source node is still connected to all operational destinations through either the red or the blue tree. Thus, if a pair of red an blue trees are generated for each source node in the network, every source node may reach all other operational destinations in the network in the event of single-failures.

The concept is shown in Figure 3.2. This figure shows a fictive two-vertice connected network, and a pair of red and blue trees where A is the root node. The redundant trees in this example are constructed to be able to withstand a single node failure. If in this example, node F were to fail, the blue tree would only provide connectivity to nodes C and D. However, with use of the red tree node A would be able to reach the remaining nodes, B, E and G, and also provide a second option to reach node D. Similarly, if node D were to fail, both the red and blue trees would provide connectivity for node A to the remaining operational nodes.

The trees are constructed by gradually growing the connected nodes from the source node S. Informally this is done by constructing a cyclic path from the source node S. By following the path in opposite directions, one direction for each of the red and blue tree, the vertices are added to the respective tree. The last link in the cycle, i.e. the link that leads back to S when following the cyclic path, is not included in either tree. If not all vertices are included in the cycle the RT method constructs an arch; a path that start and end on the cycle. In the same fashion as the cyclic path the arch is traversed in opposite directions, and appended to the red and blue tree in such a way that the last link on the path is not included in the tree. If there is still nodes that are not included, new arches are constructed in the same way, starting on a node included in the trees, following one or more nodes not included and ending on another node already included. The algorithm used to construct the redundant trees is presented in detail in section

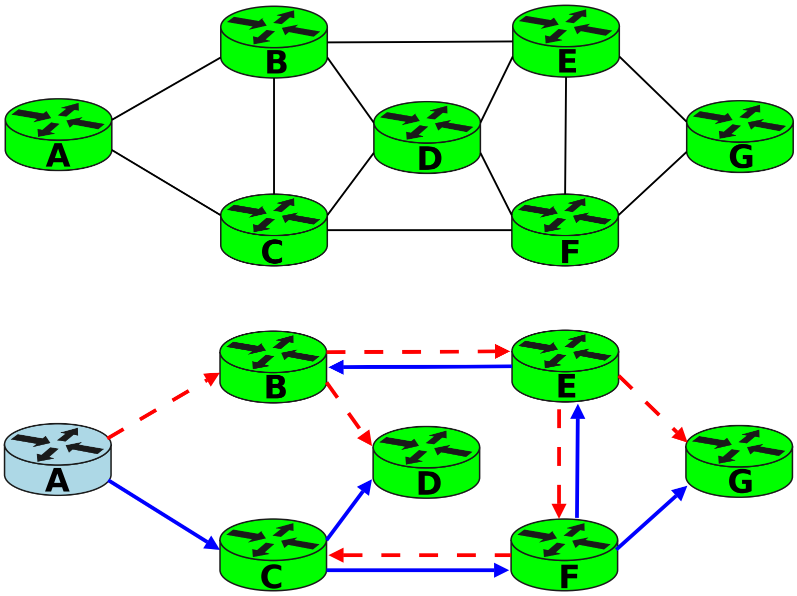

Figure 3.2: A red and blue tree with root node in A

The RT approach provides some methods for optimizing the performance. By choosing the cycle and the arches in different ways the paths used for recovery may be altered to better achieve a desired behavior. In the algorithms presented in the original RT article[1], the trees are grown in an arbitrary way. However, there is an ongoing research by Xue et.al [16] [17] to enable the method to make informed decisions when selecting the cycle and arches, e.g. to minimize delay or cost of recovery paths. In addition, the research also focus on optimize the run-time of the RT algorithm.

If the redundant tree model is to be used to protect against node or link failures the network topology needs to be node- or link-redundant, respectively.

The redundant tree approach is mainly focused on global recovery in networks operating in a connection-oriented manner. In [16] it is proposed to let one of the trees represent the “working path“, e.g. let traffic follow one of the trees when the network is not experiencing failures. In case of a failure, the affected flows or paths may be moved to the other three by the source node.

3.6 Related work

3.6.1 Resilient Routing Layer (RRL)

The Resilient Routing Layer model presented in [18] is based on a protection scheme with pre-planned paths for both global and local recovery. The method utilizes a centralized computation of the alternative paths in the implementing phase of the network, and is used for protection against both node and link failure.

The core of RRL is the utilization of a simple global abstraction referred to as routing layers. Each layer is a subset of the network topology, which contains all nodes but only some of the links in the network. A safe node is a node that only has one link in a given layer, and that layer is defined as the safe layer of the node. Every node in the network, if not originally deemed an articulation point should be safe in at least one layer.

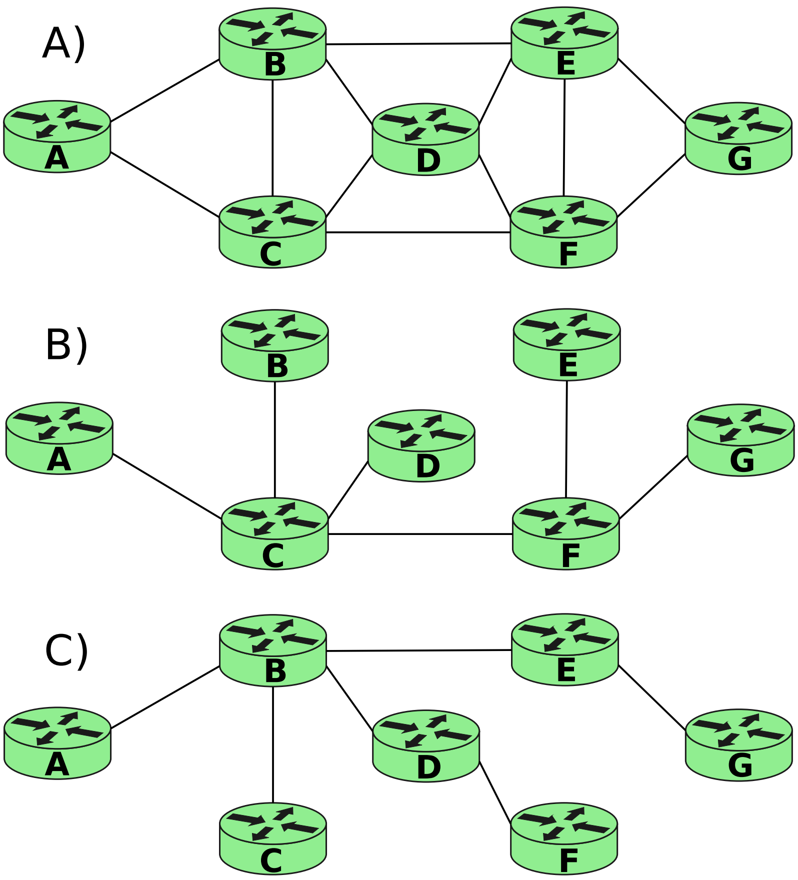

Figure 3.3: An example network. B) and C) represent the layers based on A)

If RRL is to protect against node or link failures the mapped graph needs to be two-vertex connected or two-link connected, respectively. In contrast to RT, the RRL method can still be utilized without modifications even if the network does not meet these requirements, but this leads to a situation where RRL cannot guarantee the fault tolerance for every node in the network. In addition, there are some requirements to the operation of the network. In connection-oriented networks, each new layer requires a new set of paths to be signaled. For a connectionless network, the packet header should identify the current valid layer. In a failure situation, only packets on route through the failed node or link should have their headers updated to a valid layer. When global recovery is used, the ingress node should know of the entire route of the traffic or packet, and what paths are affected by the failure. In local recovery, these requirements are not necessary as the node will know if any of the adjacent links or neighbor nodes are down. In addition, the rest of the network will be routed according to the full topology.

The RRL scheme lays no boundaries on how to produce the layers. However a general approach is to reduce the number of layers by making as many nodes as possible safe in the first layer, and furthermore keep track of articulation points and safe nodes - and process the nodes by removing all adjacent links of a node but one. The nodes are processed until all are safe, i.e. to the point when the graph will become disconnected by removing another link. Thus, all remaining nodes have become articulation points. Subsequently, the next layer is computed, where the focus is to make nodes that were not safe or originally deemed articulation points safe. To attain more equal routing performance in the different layers, a post-processing of the layers is performed, where the goal is to equalize the number of safe nodes in each layer.

RRL offers two possibilities for optimizing the recovery topology. First, choosing which nodes to be safe in a given layer can be used to reach a desired behavior. When the number of safe nodes is reduced in a safe layer, more links will be freed to be utilized in the same layer, and thus the average recovery path-length is reduced. This may be used in order to reduce the number of safe nodes in a safe layer of a node that is expected to fail often. Second, RRL offers freedom in the number of layers that is to be used. With more layers, less safe nodes must residue in each layer and thus more links are freed leading to a reduction in the average recovery path-length.

RRL uses three important observations when protecting against node failures. First, when a node is in a safe layer, it will not experience any transit traffic. This leads to the second observation; whenever a safe layer of a failed node is used, all nodes except the failed node are unaffected by the failure. The third observation is that whenever a node has failed all traffic to that node is lost in any circumstance. For link failures, a somewhat different approach is needed. Generally, a safe layer for a link is the safe layer of a downstream node n. This rule does not comply if the final destination is n itself and the failed link is the leaf link of node n. To mend this, one can try to use the safe layer of the detecting node, but only if the leaf link is unaffected. If this fails, the final solution is to use the detecting nodes own safe layer, but deflect the traffic to another link than the leaf link. This would guarantee the traffic to avoid the failed link.

3.6.2 IP fast reroute framework

Internet Engineering Task Force (IETF) is a standardization body that contributes to the engineering and evolution of Internet technologies. Fast recovery is an important issue in the Internet, and the Routing Area Working Group (rtgwg), a working group within IETF, is currently mapping technologies and working on a framework for IP fast reroute (IPFRR) [19].

The main goal in IPFRR is to provide a framework for mechanisms that protect against link or node failure in connectionless networks. The repair paths should be locally determined, and furthermore, the mechanisms should focus on solving failures that would require multi-hop repair paths. There are also some strong requirements on the functionality of the mechanisms developed with this framework. The mechanisms should not impose any constraints on the network topology or assigned link cost. In addition it should never perform worse than existing router convergence techniques and provide co-existence with non-IP fast reroute capable routers in the network.

Repair paths

There are several solutions for providing repair paths in IP networks. In IP fast reroute framework there are mentioned three general ways; Equal Cost Multi Path (ECMP), loop free alternate paths and multi-hop repair paths.

ECMP is generally considered one of the simpler approaches to provide repair paths. It is a routing technique originally developed for load balancing of the traffic among multiple equal cost paths. In this scheme, the router keeps track of paths towards a destination where the cost for each alternative path are equal. The gathered information may subsequently be used to disperse the traffic bound for a specific destination over more links. This mechanism may also be used in a recovery situation where one may route all traffic over the paths unaffected by the failure. This solution is very simple and easy to use as it is common for networks to have ECMP schemes deployed. However, the coverage for this solution is not very good because equal cost paths are not always available.

To expand the coverage of ECMP, loop-free alternate paths may be used. This is a technique for rerouting packets where the adjacent nodes have a path towards the destination that is unaffected by the failure. Generally, a loop free alternate path requires all the nodes in the network to compute the shortest path trees of their neighbors. Then, for each possible adjacent node or link failure, each node uses the shortest path threes to find which of the neighbor nodes that may have an unaffected path for all possible destinations. The next-hop recovery neighbors are selected in such a way that loops in the network are avoided. In addition, the scheme proposes several optimizations on the computation of the neighbor shortest path trees and recovery paths. This technique extends the coverage over the ECMP scheme but as ECMP, it does not provide full coverage. It is anticipated that ECMP and the loop-free alternate paths combined can provide about 80% coverage[20] [19], but the exact percentage will depend on the network topology.

In order to reach higher coverage, multi-hop repair paths may be used. This method provides the most complex protection against failure, but in return, it may provide full coverage. In a failure situation it may be necessary to reroute the traffic several hops away from the failure before a node who's path is unaffected by the failure is found. Multi-hop repair paths may further be classified into two categories depending on what mechanisms are needed to represent the recovery paths. Furthermore, the approaches are also divided by the signaling procedure needed to guide the packet along the selected paths.

Pre-computed FIB

With pre-computed FIB, one or more alternative FIBs are pre-computed in all routers. In a failure situation the recovery FIB are used to forward IP packet. There are two ways of signaling when to change to the alternative FIB; 1)It is possible for the routers to switch to the alternative FIB by performing logical checks. If a packet arrives at an invalid interface, the router may forward this packet by the alternative FIB. 2) It is possible to mark each individual packet and let the marked packet indicate what FIB to use.

This solution is used in the RRL method.

Tunneling

A series of tunneling based approaches have been proposed in [19]. What kind of tunnels are to be used, are not defined, but IP-in-IP described in [21] is a common technique where IP packets are encapsulated in the payload of another IP packet where the destination address is set at the end of the tunnel. At the end of the tunnel, the payload is extracted and forwarded as usual towards the destination. One of the advantages of tunnels is that they may be utilized without any changes to the FIB, but as with source routing the workload imposed on the nodes in the network may be considerable.

In both IPv4 and IPv6, it is possible to support source routing. This is a method where the source of an Internet datagram supply the routing information to be used by the intermediate nodes in the forwarding process. IPv4 has a field for strict source routing that the ingress node can use to specify a path based on pre-computed recovery paths [22]. However, this scheme imposes a considerable workload on the nodes, as every router along the recovery path needs to rewrite or update the IP header. Furthermore, the solution may interfere with the achieved throughput as the header size grows and less of the maximum transportation unit (MTU), i.e. the largest allowed IP packet size, is available for actual data. In addition, the IP header allows this field to appear only once in a datagram, and is meant to be used by the real source of the traffic. Thus, the method may force the ingress node to check if a path has been specified and check if the specified path is violated by adding the recovery path, if this cannot be guaranteed the datagram cannot be delivered, or the packet needs to be tunneled.

The TUNNELS method [23] tunnels the traffic around the failure terminating the tunnel at an intermediate node. When the packet arrives at the "other side" of the failure it may be forwarded as though it had traversed the failed link or node. No signaling is required for this method to work, and the tunnels are pre-computed at each node and kept in a separate data structure. This enables the scheme to not affect the size or structure of the FIB. The method may be used to compute recovery paths for every possible failure, as long as the cost defined on the links are not asymmetric. When asymmetric cost is used, 100% coverage may not be possible.

"Not-via" [24] is somewhat similar to the TUNNELS approach. Each component in the network, i.e. node, interface of nodes and links, is assigned a special address called the not-via address of a component. The nodes in the network broadcast the not-via addresses, i.e. each node broadcast the addresses of its components. In a failure situation the traffic is tunneled, as in [23], to the appropriate not-via address in such a fashion that it avoids the not-via component. Apparently, this method is able to repair all possible failures.

3.6.3 Last hop

Most schemes addressing both link and node failures need to deal with the "last-hop problem". The "last-hop problem" arises when the node immediate upstream of the failure is the last-hop node before reaching the destination address. In these situations, it may be impossible for the node initiating the recovery procedure to determine if the failure originate from a node-failure or a link-failure. However, the traffic should be tried recovered once as there is a chance that the node is operational. In these situations, it may be necessary for the recovery mechanism to distinguish between node and link failure. This is because if the failure is actually a node-failure and it is treated as a link-failure it may create loops in the network. E.g. if a recovery scheme were to solve the "last-hop problem" by sending the involved packets to another of the destinations neighbors, without informing that it was trying to solve the "last-hop problem" and the failure was a node-failure, the recipient node would repeat the procedure - possibly leading to looping of the recovered traffic.

Chapter 4

Method

In this chapter the medhod and environment used to realize a model of the IPRT method is presented.

4.1 Choosing the environment to model IPRT

The use of models gives a great freedom in the networks available for experimentation as any real or imagined network may be freely modeled. It grants the opportunity to freely experiment with configurations on existing networks. In this way models may accommodate for experimenting with both scenarios and configurations that might not have been possible in other circumstances. For example by providing an environment where network technologies may be tested without inducing disruption in an operational commercial network.

There are three general approaches to consider when modeling a new network protocol. The utilization of one of these approaches should not exclude the use of the other ones, as they may complement each other. The approaches differ in how easily available the models are, the results that may be obtained and the flexibility they offer. The three approaches are listed below.

1. Mathematical analysis
2. Testbeds and prototypes
3. Simulations

The advantage of using a mathematical model is that the environment this approach provides, may be used to quickly produce results. This is especially true in situations where the problem area consists of a common problem where a formula or calculation method is known in advance of the model. Furthermore, it might provide a clear overview of the environment, as well as parameters that governs the results. The drawbacks of using a mathematical analysis is that the models may, to a large extent, need to simplify the environment they are supposed to model. Furthermore, when modeling large and complex systems, the states needed to properly represent the model, may grow too large and render this approach unpractical.

One of the strongest assets of testbeds and prototypes is that they provide results with a high degree of credibility. The use of testbeds or prototypes may be superior to mathematical analysis when the problem area is complex and simplification of the real system is impossible or undesirable. However, this approach may require a complex development process, where it might prove difficult to verify or build the protocol in incremental stages. Thus, this is an approach that is often used to implement and verify the abilities of a protocol where requirements and behavior has matured. Furthermore, because the tests are run in a real environment, it might be difficult to access all the necessary measurements and it might also involve expensive instruments.

Simulations can be a very flexible and efficient method when analyzing complex systems. This approach can be used to model protocols where the problem area is too complex to be tested in a mathematical environment and where a flexible development cycle is needed. Furthermore, this approach provides the ability to choose the level of abstraction. This allows the simulated environment to better suit the desired level of complexity, and by varying granularity, it is possible to accommodate both detailed and high-level simulations. However, the results that are obtained from a simulator is usually less credible than those obtained from testbeds or prototypes. If a simulator with an existing framework can be used, this might save development time and, in addition, remove some of the need for simplified or static assumptions on higher and lower layers in the network.

In this thesis, a new method for providing IP fast recovery will be tried developed, and thus, it is anticipated that there will be a need for a model environment that is flexible and allows easy adaptation of new ideas in the development cycle. Some graph theory will be used in the initial design phases to be able to convert the original RT method to applicable to conventional IP networks. Furthermore, the problem area is considered to be complex to be completely understood in an easy or proper fashion when using a mathematical model. Thus, a simulated environment will be used to model, verify the findings and measure the performance of the IPRT method. This enables the evaluation of the performance and abilities of a IPRT as it interacts with other protocols in varying conditions. The level of control makes it easy to measure and inspect all state information of the model, at any stage of the execution. In addition, it may allow for a rapid change in model properties because of the control that is provided over the different aspects of the system. For example may topologies easily be replaced and method easily added or deducted from the model.

Because simulation is performed on models and assumptions and abstractions are used, less of the actual events are found in the simulation. Thus, simulation may give a good indication on the performance and correctness of a model, but guarantees may be hard to give.

4.2 Simulators

Generally, networks may be described through stochastic models where one or more distinct events affect the state or components in the network. A variety of simulators specializes on specific aspects of network simulation. However, simulators are generally set apart by how they model time and how they model events.

Time advance in a simulator may be modeled with either a "next-event" or a "time-slicing" approach, both using a discrete time. The time-slicing approach advances the simulation time by moving forward at fixed intervals, e.g. every second, regardless of the activities being simulated. With the "next-event" approach the time is advanced to the time scheduled by each event. In most cases the "next-event" mechanism is more efficient and allows models to be evaluated more quickly. I.e. the simulator may jump directly from one event to the next scheduled event without affecting the overall results of the simulation.

Furthermore, the way changes in system state occur is also defining for a simulator. It may be modeled through the use of events, activities or processes. The event approach describes a change as an immediate change in one or more related system variables. The activities approach is somewhat similar to the event approach but use duration to describe changes in states. The process approach joins collections of events or activities together to describe the life cycle of an entity. Because the events are discrete and ordered it is difficult to model two events that overlap in time and at the same time may interact or interfere with each other.

The most commonly used approach is discrete-time event-driven simulators. A commonly known simulator that uses this approach is the "network simulator" also known as "ns" or "ns2"[25]. Another approach, found in J-sim [26], is a real-time process-driven approach to simulation. In such systems, the evolution of a system is defined by processes taking place at real-time along a virtual time-axis. Each process is executed in an independent execution context and process interaction is modeled, as it would happen in a real implementation. Furthermore, the real-time process-driven approach is an extension to the discrete-time event-driven approach. E.g., the process-driven approach is also event-driven. However, the interactions between the entities, i.e. processes, are explicitly defined through dependencies and synchronizations between the processes.

In this thesis J-sim will be used to provide the simulator environment in which IPRT will be implemented. The J-sim simulator is described in more detail in the following section:

4.2.1 J-sim

J-sim is a component-based discrete event simulator written in Java that may be used for network simulation. It provides many network related packages including a network package (INET Framework), wireless package, sensor network package, and a differentiated service (Diffserv) framework.

The simulator is founded on the "Autonomous Component Architecture" (ACA). This environment tries to mimic a "black-box" approach to modeling often found in development of integrated circuits. The defining properties of a "black-box" is that both its purpose and the in/out signal pattern through pins or ports are fully specified. This allows the components to mimic the behavior of real-world systems through message passing and an independent execution model. This is achieved by allowing data arriving to a port of a component to be immediately processed by that component in an independent computation context. In addition, the ACA allows for a function call execution model where a component may send data to another component to be computed in the same computation context as the sender.

There are several reasons for why J-sim was chosen in this thesis. It is written purely in Java, and thus, provide with a well-known programming environment. Furthermore, it provides with all the basic components needed to simulate networks and perform measurements. In addition, SIMULA has implemented a version of the RRL recovery procedure in J-sim that could provide with both a good foundation for initial development and an opportunity to test both RRL and IPRT under similar conditions.

Chapter 5

IPRT Design

This chapter contains an overview of the important design choices for implementing IPRT for conventional IP networks. In each section, the problems will be identified and possible solutions will be presented.

5.1 Construction of the trees

The original RT work [1] presents two different algorithms; one for link failures, and one for node failures. Both of the algorithms follow the basic idea of growing the red and blue trees gradually by adding new redundant paths. Furthermore, the algorithms introduce a "voltage rule" to ensure that the redundant property of the tree pair is achieved. This is done by letting the rule impose a complete order on the nodes as the trees are grown to form a pair of red and blue trees. There are several attributes that govern the performance and behavior of the IPRT tree construction, where the main attributes are path and cycle creation, and run-time of the algorithm.

5.1.1 Introducing the redundant tree algorithm

The "Node-algorithm", shown in Algorithm 1, is designed to work with two-vertex-connected graphs. It starts by adding a randomly chosen cycle, containing three or more nodes, from the graph with a root S. The root is then assigned two voltages; one for the red tree and one for the blue tree, such that $V_{blue} = v_{max}$ and $V_{red} = 0$. Subsequently, the remaining vertices in the cycle are given voltages in a decreasing fashion, following the cycle in an arbitrary chosen direction. The voltages assigned are within the boundaries of v_{max} and 0. Furthermore, the selected nodes, starting from S, are added to the blue and red tree, in such a way that the assigned voltages are decreasing and increasing, respectively. Subsequently, the trees are grown by connecting two nodes already assigned to the trees, with one or more nodes not assigned; i.e. forming an arch starting and ending in the tree. The arch is then oriented so that the starting node is the one with the higher voltage of the two nodes in the trees. Following a directed path from the starting

Algorithm 1 Algorithm for vertex-redundant graphs [1]

1: j =1
2: Choose any cycle $(S, C_1, ..., C_k, S)$ in the graph with k $\geq$ 2. Let N_1 be the set of vertices $\{S, C_1, ..., C_k\}$ and order these vertices by $v_{max} > v(C_1) > v(C_k) > 0$
3: $A_1^B = \{(S, C_1), (C_1, C2), ..., (C_{k-1}, C_k)\}$
$A_1^R = \{(S, C_k), (C_k, C_{k-1}), ..., (C_2, C_1)\}$
4: **while** $N_j \neq N$ **do**
5: $j = j + 1$
6: Choose a path $P_j = (X_{j,0}, X_{j,1}, ..., X_{j,L_j}), L_j \geq 2$ in the graph such that $X_{j,0} \in N_{j-1}$ and $X_{j,L_j} \in N_{j-1}$, with $v(X_{j,0}) > v(X_{j,L_j})$.
If $X_{j,L_j} = S$ **then** $v(X_{j,L_j}) = 0$
If $X_{j,0} = S$ **then** $v(X_{j,0}) = V$
The other vertices, $X_{j,i}, i \leq i \leq L_j$, are chosen outside of N_{j-1}.
7: $N_j = N_{j-1} \cup (X_{j,1}, ..., X_{j,L_j})$
8: Order the vertices in P_j by $v(X_{j,0}) > v(X_{j,1}) > ... > v(X_{j,L_{j-1}}) > (v_{max})$, where $v_{max} = \frac{max}{y \in N_{j-1}}(v(Y) : v(Y) < v(X_{j,0}))$
9: $A_j^B = A_{j-1}^B \cup \{(X_{j,0}, X_{j,1}), (X_{j,1}, X_{j,2}), ..., (X_{j,L_{j-2}}, X_{j,L_{j-1}})\}$
$A_j^R = A_{j-1}^R \cup \{(X_{j,L_j}, X_{j,L_{j-1}}), (X_{j,L_{j-1}}, X_{j,L_{j-2}}), ..., (X_{j,2}, X_{j,1})\}$
10: **end while**

N_j The set of vertices included in the red and blue trees at stage j

A_1^B The set of links present in the Blue tree at stage j

A_1^R The set of links present in the Red tree at stage j

S The root node of the red and blue trees

P_j The arch (path) found at stage j

L_j Length of the arch found at stage j

$X_{j,i}$ Node X added at stage j at place i in P_j

$v(X)$ The voltage assigned to node X

node, the arch is traversed. All nodes, except the first and last, are assigned voltages in a decreasing manner. The first and the last node on the arch is not assigned voltages as they are already included in the trees, and has therefore already been assigned voltages. The new nodes are given voltages within the boundaries of $v(X_{j,0})$ and the highest voltage assigned to any node in the tree that does not exceed $v(X_{j,0})$ - not necessarily the voltage of X_{j,L_j}. The new nodes are then connected to the blue tree, through $X_{j,0}$, and to the red tree, through X_{j,L_j}, following the same voltage rules as when the cycle was created.

An example is shown on the topology in Figure 5.1, where node A is the root node.

From this node, the cycle $[A - B - E - F - C - A]$ is found, and the nodes are assigned their voltages according to line two. The red tree thus consists of increasing voltages, and the blue decreasing voltages, following the three-growth rule found in line three and nine in Algorithm 1. Next, the arch $[E - G - F]$ is found. Since F has the higher voltage of 6, this becomes the upper boundary. The lower voltage 4, which is owned by E, becomes the lower boundary. Thus, G is assigned a voltage of 5. E is then added to the red and blue tree in accordance to the voltage rule found in line 8. The next arch to be found is $[C - D - B]$, and since C has the highest voltage, it becomes the upper boundary. However, the lower boundary is voltage 6 owned by node F. Consequently D is assigned a voltage of 7 and added to the red and blue trees accordingly.

The "Link-algorithm" is very similar to the former presented "Node-algorithm". The main difference between the two, is that the link-failure algorithm allows the arches to start and end at the same node. Thus, it becomes necessary to assign two voltages to each node and in this manner expand the voltage rule. The "Link-algorithm" may potentially yield shorter recovery paths at the expense of introducing articulation points in the redundant trees. This is because more links may be freely used when creating trees with the "Link-algorithm". However, the "Link-algorithm" is not required to produce articulation points where none are needed. It may therefore be made to protect against node failures to the extent allowed by the topology. This may be done by refraining from letting an arch start and end at the same node as far as possible. However, this may introduce a bigger computational cost in the algorithm.

The layout of topologies where the "Node-algorithm" and the "Link-algorithm" respectively may successfully be applied, is inherently different. This is because the "Node-algorithm" has a stricter requirement on the connectivity of the networks it may be applied to. To use the "Node-algorithm", the network needs to be at least two-vertex-connected, whereas the "Link-algorithm" only needs the network to be two-edge-connected.

5.1.2 Important properties

There are several attributes that govern the performance and behavior of the IPRT tree construction, and they may influence resource usage in both routing and forwarding:

- The selection-algorithm used for path and cycle creation
- Run-time of the algorithm
- The ability to provide QoS properties

The manner of how the cycle and the arches are selected, is of great importance to the attributes of the redundant trees. In the original RT algorithms, e.g. Algorithm 1, this was left as an open question. However, there has been some research in this area by Xue et. al. [27] [16] [17]. Their results show that the creation of cycles and arches is vital for IPRT to meet the desired performance criteria. A good example may be how the trees would perform with a shortest-path versus longest-path selection of the cycle and arches; Consider a node that has two neighbors, where a link has failed between the root node and

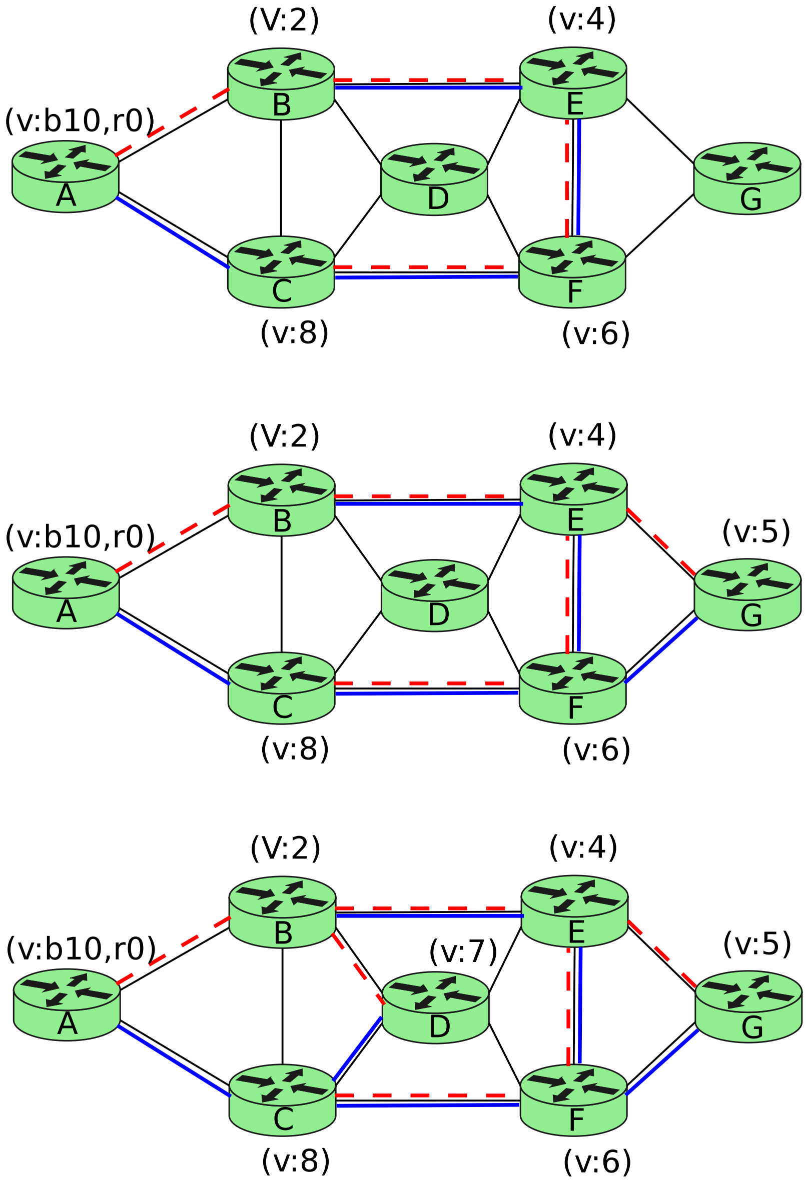

Figure 5.1: A redundant tree set is grown from rootnode A

one of its neighbors, whom the root node is trying to communicate with. The recovery path between the two nodes through either the red or the blue tree would then be of equal size to the length of the cycle - i.e. the length of A_1^B or A_1^R in line three in Algorithm 1. Thus, if a shortest path was used, there would be a guarantee that this recovery path would be of minimal length. With a depth-first search that terminates at the first node in-tree, the IPRT method would not be able to give such a guarantee. If the initial cycle was created using a longest-path algorithm, the path-length might be significant. However, the influence the chosen selection-algorithm exerts on the performance and behavior of the IPRT method is dependant upon the topology. For example, in a pure ring-topology, the various selection-algorithms would have no impact at all on the performance of IPRT. It has been shown that an approach, wherein the length of the cycle and the arches are kept to a minimum, does show a significant improvement with respect to the average recovery path length[17]. In addition, it has been shown that this approach generates trees with a higher degree of coverage in case of multiple concurrent failures.

It is important for the IPRT algorithm to support some level of QoS, as a randomly driven approach may have a negative impact on the performance and resource usage. Often, costs on links are used for QoS, i.e. to maximize available bandwidth or to minimize the number of hops. Through link cost, the RT method may also support some Traffic Engineering aspects, such as the ability to set a high cost on links known to fail regularly. However, advanced requirements may introduce a bigger computational cost.

Determining the best redundant trees - i.e. QoS oriented IPRT - for a topology, is an NP-complete problem somewhat similar to the Travelling Salesman-problem. An informal definition of NP problems is a class of problems that can be verified by a deterministic Turing machine in polynomial time. Furthermore, NP-complete problems are a subclass of NP problems that has the property that any problem in NP can be polynomially reduced to it. To find the best set of red and blue trees, the algorithm needs to try every possible combination and composition of circles and arches in a given topology. To search for the perfect solution would be impossible in practice, as the time needed to calculate all the possible solutions would be far too great for the search to be practical. However, there are many optimizations and approximations available, providing seemingly good or probably good solutions. For example, a greedy-algorithm implementation of Algorithm 1, using a depth-first-search selection-algorithm with a runtime of $O(n+v)$, would have a runtime of $O(n^2(|n|+|v|))$. This is because line two will have a maximum execution time of $O((|n|+|v|))$, and will be executed exactly once. Subsequently, line six will be executed at most $O(n)$ times, as each iteration will add at least one node to the trees and thus terminate at line four after $O(n)$ iterations. The execution time of line six will be of magnitude $0(n(n+v))$, given the depth-first search initiated from each node with a runtime of $O(n+v)$. Furthermore, if the search-method could be exchanged with an algorithm with runtime of $O(n)$, the algorithm would have a runtime of $O(n^3)$ [1]. Other algorithms provide more specialized solutions for QoS IPRT, of which the best have a run-time of only $O(n+v)$ [27].

5.1.3 Observations

For a real implementation, a fast algorithm is of great importance, as limited computational resources must be assumed. In addition, IPRT should complete the tree generation process as fast as possible to have the necessary information available to any subsequent failures. In this thesis, the run-time is a subject of less importance as the failure-scenarios may be pre-planned, and the trees may be computed off-line. Thus, the greedy-algorithm approach would be sufficient. This provides a more versatile solution, as the algorithm is not optimized towards a single QoS goal. To be able to give an accurate picture of the performance and abilities of the IPRT algorithm, the QoS properties must be considered. With the greedy-algorithm approach, it would be trivial to exchange the different selection-algorithms.

As the run-time needed to generate each tree may be brought down to a minimum of $O(v+n)$, it has been shown that the computational cost of the algorithm may be optimized enough to accommodate for a IP solution. Furthermore, the algorithm may fulfill several QoS needs and demands, and may be versatile enough to be used in existing IP networks. The algorithm may also be used on a variety of networks, but might perhaps yield optimal results in a two-vertex connected network, as this would yield the best failure coverage.

5.2 Routing

5.2.1 Enabling IPRT to co-exist in a failure-free environment

The IPRT mechanism needs to be able to co-exist with normal routing protocols in times of failure-free operation. To achieve this the usage of the original RT recovery procedure needs to be altered.

One of the original ideas for RT routing was to let one of the two trees be the foundation for failure-free operation, i.e. to be used as a "working tree". In a connection-oriented environment this approach give some advantages in the recovery process, as a failure reported on a connection would enable a router to immediately switch to the backup path. One of the main disadvantages of applying this approach to conventional IP networks is the length of the default paths. The construction of the redundant trees needs to follow strict rules to be able to provide node-disjoint paths. Thus, it is probable that the paths generated by IPRT does not provide the best path choices available. The construction of the redundant trees needs to follow strict rules to be able to provide node-disjoint paths, and thus it is probable that the paths generated by IPRT do not provide the best path choices available. Furthermore, this approach does not provide a valid recovery procedure. If the IPRT method is used as a basis for the forwarding mechanism during failure-free operation, it is also a subject for the IP re-convergence process. Thus, even though the method would able to recover traffic at times of failure, it does not solve micro-loops.

To be able to recover traffic at all times, additional virtual recovery-topologies could be used in addition to the normal topology during failure-free operation. Subsequently, the FIBs obtained from the additional recovery-topologies may be used to forward the traffic affected by a failure. This approach is used in RRL, and is also required by the IP fast reroute framework. The additional topology enables the routing algorithm to behave and be configurable like expected in a normal network. This, without being hampered by the recovery mechanism, and only depending on the restrictions of the preferred routing protocol. The FIBs generated from the recovery-topologies are therefore only used for forwarding in recovery operations.

When the IPRT algorithm is run, the resulting trees are laid as an overlay on the original topology, and subsequently each of the links outside the tree are assigned a cost of very high value, i.e. the maximum available. Setting the link cost with a sufficiently high value is the same as excluding the link from the topology when the shortest path algorithm is used. As a result all the links are a part of all the recovery topologies, but not used for packet forwarding.

When IPRT is used in conjunction with an LS routing protocol, the LS protocol may provide the RT method with the needed topology information and routing mechanism. As an example, in multi-topology IS-IS [28] it is possible to let each topology either have their own dedicated routing protocol where LSPs are marked according to the tree ID, or share the routing scheme where LSPs are shared between the topologies. In addition, the IPRT recover mechanism may use the LSDB to get the needed topology information. Similar operation is also available in OSPF.

In a simulated environment, the routing protocol may be represented off-line and implemented in a static manner. By doing this, more time may be used on implementing and testing of IPRT, and in addition, the static property provides a simpler scenario to analyze. This approach also helps to ensure that the routing is executed in a deterministic manner, and in this manner reduce the needed work and potential problems that may occur when testing the IPRT method. This approach does not lock the implementation to a specific routing protocol, but leave this work for future implementation and design decisions.

5.2.2 Computation

The computation of recovery routes can be achieved in IPRT either by a centralized scheme, a distributed scheme or a combination of both. The choice of which approach to use depends on what kind of resources that are available in the network; bandwith or CPU cycles.

The computation in the original RT scheme was meant to be done by a centralized server. At connection-setup, the node were to query the server, and obtain the working path along with the recovery path. In a connection oriented solution, connection-setups may be rare. Therefore, the delay associated with a centralized scheme may be acceptable, as long as it is within the order of delay required for setting up the connection. In a connectionless network, no connections are set up prior to initiating a communication. This does not void the use of a centralized scheme in a connectionless environment, as the recovery FIBs or topologies may be computed at a centralized server and distributed along any periodic or triggered route update.

Another valid approach is to use a distributed scheme in order to produce the redundant trees. This is possible if the IPRT method is used in conjunction with an LS routing protocol. In such a scheme, all routers are required to compute the pair of trees for every node in the network. This may be done assuming all nodes have a synchronized view of the LSDB, the IPRT algorithm is deterministic, and use the same snapshot of the LSDB as input to the algorithm.

It is also possible to utilize a combination of distributed and centralized computation if IPRT is used in conjunction with a LS routing protocol. In this approach each node in the network is responsible for computing the pair of redundant trees of which they are the root node. As with the distributed scheme this approach require a synchronized view of the LSDB at the time the trees are computed. The information may then be broadcasted as a part of the LS route update messages or as a separate package with the same delivery and send properties as an LS route update message. If this mechanism is to be effective in response to a failure the redundant trees should be computed and broadcasted as a part of the re-convergence process. This is because if the computation and subsequently the broadcast are delayed to after the re-convergence has finished, the mechanism would be more vulnerable to failures coming in rapid succession. This approach is guaranteed to work if one assumes all LS route update messages are guaranteed delivered and that all routers broadcast LSPs. Furthermore, if a router does not deliver a pair of redundant trees the router must have failed or been disconnected from the network. Since there is no way or reaching a failed or disconnected router there is no need for recovery paths to this router anyway.

The centralized scheme may result in an increased amount of traffic when compared to a pure distributed scheme. This is because all the trees need to be broadcasted to every router in the network whereas a distributed scheme could use the LSDB without adding to the amount routing protocol related traffic. The actual amount of network traffic needed is implementation dependant but it would need to represent $2 * n$ topologies. In addition, the scheme also suffers from the general drawbacks of having a centralized responsibility, e.g. it introduces a single point of failure in the IPRT scheme. However, centralized approach does not require the individual routers to calculate the trees, and thus has a lower computational cost at each node. With the distributed scheme these drawbacks are not present. This is because the only information needed to be broadcasted is the LS route update messages. However, this approach imposes a higher computational cost at each node as the tree pair of every node needs to be computed. If the combination of centralized and distributed computation is used, the amount of generated network traffic is still high. However, the computational cost is reduced at each node, and at the same time this approach does not suffer from the general drawbacks of a centralized approach. Thus the different approaches become a question of available computational and network resources.

This shows that the IPRT method may be flexible and resource usage may be shifted between either network or computational usage. However, since the IPRT method is to be realized in a simulator this resource usage is not a governing criteria. The centralized scheme seems the best choice since it allows to utilize a centralized approach that computes the trees offline and at the same time is non-dependant on any routing algorithm.

5.2.3 Signaling and path representation

When a failure occurs, it is important that all the nodes in the network are able to detect the packets affected by the failure and ensure that all packets are forwarded along the correct recovery path. In the framework for IP fast recovery, three methods are presented as possible solutions to represent multi-hop recovery paths, and IPRT may draw inspiration and conceptual design from these approaches. However, if these schemes are to be used in conjunction with IPRT they may need some alteration.

Two main mechanisms need to be accounted for.

- To be able to forward a packet along a multi-hop path that differs from the default routes there is need for a mechanism to represent the alternative recovery paths in each router. This is achieved by introducing an additional recovery FIB at each router. In a broad definition, the forward information base in a router is a data structure that helps the router decide the next hop of a packet, thus the actual data-structure of the recovery FIBs may take on many forms.
- The trees, and thus the different recovery paths, may be distinguished by both the root-node and the color of the tree. The signaling provides the routers with a mechanism to identify a recovered packet and thus forward the packet according to the signaled redundant tree topology. However, the path representation and the signaling mechanisms may be closely related to each other and a single solution may extend to cover all the needed mechanism.

Path representation	**Signaling**
Source routing	Network-local addresses
	Mark IP header
Separate FIBs	Network-local addresses
	Mark IP header

Table 5.1: Possible paths and signaling mechanisms

Path representation

In IP fast reroute framework there are proposed several solutions to properly represent the paths. One approach described is to utilize source routing. In this scheme, the recovery FIB would contain a series of pre-computed chains of intermediate routers the recovered traffic would need to traverse. Thus, the responsibility to forward the traffic along the correct path is assigned to the node initiating the recovery. By using source routing, the affected traffic may follow the recovery path by the means of the normal forwarding procedure. I.e., no intermediate nodes may forward all traffic according to the normal routing table as long as no local failures are present. Even though the forward-procedure does not need to know the recovery packet, the packet needs to be signaled as a recovered packet as this information is needed in the event where the recovered traffic encounters a second failure.

Another approach to represent the recovery paths is to create additional recovery FIBs where the structure and representation is equal to the FIB used in a conventional IP network. This solution lets all the routers share the responsibility of forwarding a packet according to the selected recovery route. Thus the signaling will need to both identify a recovered packet and what recovery FIB the intermediate routers needs to use when forwarding said packet. Thus, this method extends the forwarding procedure at each router, as the forwarding procedure must decide what FIB to use on a per packet basis.

Signaling

As with the path representation, there are several approaches to signal the existence and selected path of a recovered packet. One possibility is for the IPRT solution to utilize an identification approach somewhat similar to the one found in "not-via" addresses [24]. By not assigning special addresses, but rather assign special-subnets this scheme could fully represent the topologies of the redundant trees. The Internet Assigned Numbers Authority (IANA) has reserved a class A IP address space for private internets[29]. By utilizing a addressing scheme where the different classes of address spaces corresponds to the different unique identifiers of a redundant tree, e.g. the root-node, the color of the tree and the nodes represented in the tree, every relation of a tree is maintained in the signal. This would allow the IPRT scheme to identify all nodes in the network and in addition provide information on both the color and the root of a redundant tree. Furthermore, the addressing scheme would enable the routers to be able to identify a recovered packet based on the address of a packet. An example could be to let the different root nodes be mapped to different class B sub-networks, and use the class C sub-network to identify the red and blue tree within each class B network. Furthermore, the higher eight bits could correspond to the higher eight bits of all nodes in the network. Thus a node with address on form XXX.XXX.XXX.8 would be assigned two addresses of form 10.8.[1,2].8 in the red and blue tree where it was root node. This scheme may also accommodate for a lower granularity if desired and thus allow a restructuring of the address assignment to fit a specific implementation. The schematic of each recovery tree address is that a recovered packet must be delivered to the node according to the topology represented in the recovery address.

Another approach is found in [18] the solution is to mark the packet. This may be done through the type of service (ToS) field, often used by diffserv, of an IPv4 packet. Since redundant trees needs to be calculated for each possible sender the maximum needed configurations is two times the number of nodes in the network. However, this solution require that the number of bits are small in order to be practical. It may prove that the total number of FIBs that is needed to support IPRT solution may allow for IPRT to coexist with a typical QoS service within a network, but such an emerging of code points is outside the scope of this thesis.

Effect of path and signaling mechanisms

Choosing different paths and signaling mechanisms may interfere with several aspects of the network:

- The size of the state information needed to implement IPRT
- The forwarding procedure
- The traffic overhead associated with IPRT
- The topology size IPRT may address

The choice of recovery path representation mechanism does affect the size of the recovery FIBs. As an example consider the recovery path length; the length of the recovery paths depend on the network diameter, where the best average path length would at minimum equal half the diameter of any given network. Thus, if source routing were used as a path representation this would result in recovery FIBs where the average number of intermediate nodes needed in each FIB entry would be at least equal to the network diameter. However, it is possible that the number of addresses listed to represent the path could be eased by doing some extra computation in order to omit the addresses that already follow the path selected for the traffic in a normal operation. This reduction requires that the recovery path and the normal traverse the same nodes for at least three hops before one address could be removed from the entry. Thus, the possible gain in lowering the size of the FIB would come at the cost of additional computation. However, when the multiple FIB solution is used the size of the recovery FIBs are not depending on the average recovery path length. Thus, this approach requires less memory to represent the redundant trees at each router.

Furthermore, the scheme for representing the recovery paths may also affect the normal routing procedure. Depending on the method chosen to implement the source routing the routers may be forced to update the IP header of recovered packets at each intermediate node. During normal operation, no such tasks need to be performed and thus the additional cost in the forwarding procedure does only affect recovered packets. However, source routing is an optional forwarding procedure, and there may be introduced additional cost in the form of requirements for the routers to enable support for this mechanism.

If this cost is present in the multiple routing table mechanism is depending on the signal mechanisms. If the packets are marked, the forwarding mechanism must inspect for the signals on a per packet basis. Thus, a constant increase in the time usage for the forwarding-mechanism is introduced. However, this increase is not very high as the only operation required in a failure-free situation is to read an additional field in the header. Furthermore, if the special-addresses are used, no additional cost is introduced in the forwarding procedure, as it only needs to treat the recovered packets as any normal packet.

The choice of signaling scheme may also affect the throughput of a network. When IP encapsulation is used, e.g. IP-in-IP tunnels, the total maximum transportation unit

(MTU) of the network is reduced, as additional space is needed to represent the new IP headers. As an example consider the special-addressing scheme were the nodes in the network are given additional addresses. To use these addresses the original package needs to be encapsulated within a new IP packet where the new recovery-address is set as destination. Furthermore, by performing this encapsulation, the package may get segmented or the total MTU of the network must be lowered equal to the number of bytes used by the additional header. This effect may also be observed in conjunction with path representation such as source routing. However, when this method is used, the lowering in MTU may not be an option as the new size of the header may grow to account for a significant amount of bytes. When the original IP packets are marked the need for a tunnel may be avoided as special fields in the IP header may be used, e.g. the ToS field, without affecting the original settings in the header.

The scheme of using the special-subnet addresses has a scalability problem. When the address scheme is used as described in the example the number of addresses that the scheme are able to represent is less than 255. Thus, this solution may prove to provide to few addresses to be a practical solution when used as described in the example. The source routing is also affected by the number of nodes and carries an increase in resource usage as the diameter of the network expands. However, it could prove that a combination of multiple routing tables and marking the packet may be the most resource efficient and scalable scheme for IPRT. The multiple routing table optimizations needed for packet marking to be a practical solution, are discussed in detail in the following section.

5.2.4 r/bTables

The multiple routing tables proposal carries a high cost in memory usage, as the number of routing tables are directly dependent on the number of nodes in the network. Therefore, this solution is not a valid option in a conventional IP network. r/bTables is a solution for reducing the footprint of the IPRT method in the FIB structure. The main idea is to remove all non-essential routes from the recovery FIBs while retaining the functionality and properties of the redundant trees. The basis for the multiple routing table solution is to view the upstream node of the failure as the root, and with this as a starting point try to recover the traffic using either its red or blue recovery tree. One of the major disadvantages in this approach is that the root is required to reach all other nodes. Thus the recovery FIBs of the root node needs to contain the next hop for each possible destination.

If one is able to reverse this situation and use the destination as the root, there would only be need for one entry in each of the recovery FIBs, i.e. since there is only one possible destination instead of all other nodes except the root, the FIB would be reduced accordingly.

The reduction is based on the following observations.

- Every node in the network has its own unique pair of red and blue tree in which it is the root node.

- Consider a red and blue tree pair rooted in a node (S). In both trees, there exist paths from S to every node in the tree.
- If the trees are constructed from a topology with only bidirectional links, there exist a reverse path for every path.
- In a single-failure situation S should be able to reach any operational node either by its red or blue tree. Given the reverse path the opposite should also hold true; in a failure situation, any operational node should be able to reach Seither by S's red or blue tree, provided S is operational.
- Every node in the network is represented exactly once in each tree.
- Every node, except S, has exactly one parent in each tree.

The voltage rule ensures that a node is represented at most once in each tree, i.e. if a node is encountered more than once it must have obtained more than one voltage in the tree and thus break the ordering the voltage rule imposes on the nodes. If a node is not in the tree this would mean that the network does not comply with the connectivity requirements, e.g. the network is segmented or one-edge-connected, and that the IPRT method could not have been successfully applied to the topology.

The resulting topologies, after the RT algorithm has completed the computation, are two trees. If, for example, a node Y has more than one parent, the voltage rule could not have been enforced as the voltage of a node may only take upon a single red and a single blue value, and by following the loop nodes are encountered more than once and thus break the descending or ascending voltage rule for the red or blue tree respectively. Thus, the reverse path from any of the children of the root node is unambiguous and loop-free.

The redundant tree algorithm require the voltage found in the red and blue tree to be monotonic increasing and decreasing, respectively. Traversing the trees from a leaf node will change the red tree to increase and the blue tree to decrease. Let $X \neq S$ be an arbitrary vertex that is removed from the graph and let another node $Y \neq S$. In this example Y may still reach S in either the red or the blue tree. Since the vertices are ordered one of the following properties must be true; either $v(Y) > v(X)$ or $v(Y) < v(X)$. For the first case S may be reached through the red tree as it provides parents who have voltages that are monotonic increasing. For the opposite direction the blue tree could be used as it provides parents who have voltages that are monotonic decreasing.

The r/bTable method solves the last-hop problem. Consider the root node S, in the red tree it has a voltage equal to zero, and in the blue tree it has a voltage equal v_{max}. Furthermore, imagine that a link between a neighbor node, Y, and the root node, S, has failed. If Y want to send traffic to S over the failed link, one of the following properties must be true; either $v(Y) > v(S_{red})$ or $v(Y) < v(S_{blue})$. Thus, traffic from Y to S may be recovered using either the red or blue path as shown in the former paragraph.

These properties ensure that there exists a loop-free and unique path from each leaf node to S, and that all the paths from the parents are proper sub-paths of its children's paths. I.e. the path from a root of a subtree towards S is independent of the starting point in its subtree.

By only considering the paths from the children towards the root of each tree the size of each corresponding FIB may be reduced to only contain one entry indicating the next-hop on the path towards the root and its directly attached hosts and subnets. In an IP packet the destination address is always present, and from this information it is possible to determine which pair of red and blue recovery trees the destination is root node of. Furthermore, since all nodes are present in both the red and blue trees, and that a path is guaranteed in case of a single-failure, the upstream node of the failure may use at least one of these trees to recover the traffic affected by the failure.

An example from a synthetic topology is shown in Figure 5.2. Only the red and blue tree where node A is root is shown for simplicity, and the red and blue directional arrows show the next hop from a node following the red or blue tree towards the root. Furthermore it is shown a situation where node B has failed and a situation where node C has failed. In both situations the root node is reachable trough the red or the blue tree, for failure on B or C respectively.

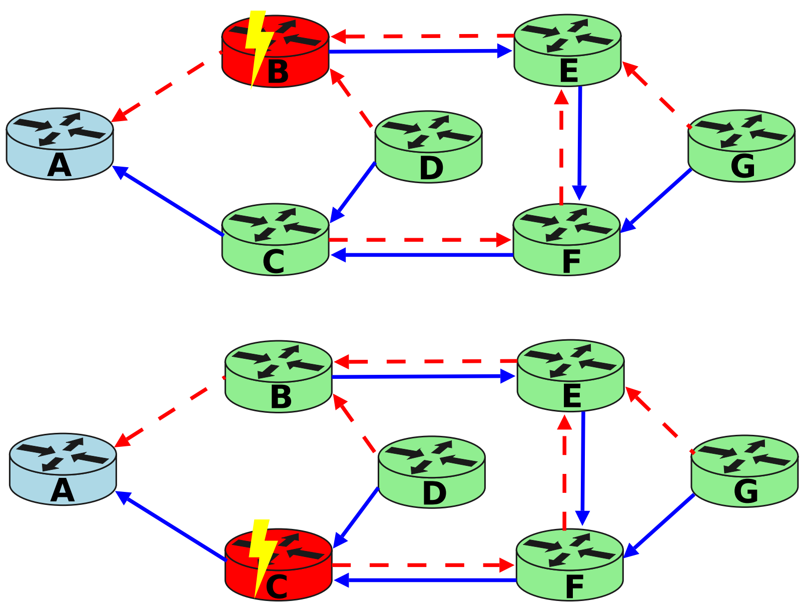

Figure 5.2: Two different failures in a r/bTable enabled network

The union of the recovery FIBs created from the red trees, and a union of all the recovery FIBs created from the blue trees will provide with a complete red FIB (rTable) and a complete blue FIB (bTable). This may be done since all the possible destinations are represented exactly once in each tree if the network is valid for recovery. I.e. if the destination is not in the tree, it is not possible to recover traffic bound for that destination. This union of the recovery FIBs is not necessary for the procedure to function properly, but allows the signaling to use less resources. This is because the marked packet only needs to contain a “recovered” bit and a bit indicating which of the recovery tables is to be used. With the multiple reduced routing tables the marking needs more bits to indicate what table to use.

In the r/bTable method the destination will dictate what pair of redundant trees are used as basis for the path a recovered IP datagram use. Normally the destination of an arbitrary IP packet would not be a router in the network but rather a host or sub-network attached to an egress (last-hop) router. Thus, if the r/bTable approach is to be used, the view also needs to incorporate a binding between all possible destinations and their respective egress (last-hop) routers. I.e. when a pair of red and blue trees are computed for an arbitrary router in the network the trees would need to be associated with all the possible neighbor destinations except destinations who themselves are routers in the same AS. There is almost no additional computational cost associated with this binding requirement as the number of iterations over the tree generation algorithm does not increase.

The observations seen from the multiple routing tables solution are still valid for this method as the red and blue tables yield a valid routing table that may be used in conjunction with a “normal operation FIB” or by itself. By concatenating the FIBs the cost and original topology view is lost and therefore the recovery method still needs to rely on a routing protocol to obtain the full topology view needed when constructing the redundant trees. However this solution would require packets to follow a different set of recovery paths when a two way communication is used. The reason for the paths A→B and B→A to be disjunct is because they stem from two different redundant trees. With the multiple routing table solution this could be avoided since this solution provides with a proper FIB for each of the generated topologies, and thus leaves the necessary information to be able to route packets both ways following a FIB created from the same topology.

The big advantage for this solution is that it removes the dependency between the number of FIBs and the number of nodes in the network, and replaces the memory usage associated with this method with a constant factor. The resource reduction is only found in memory usage as the computational resources needed does not differ from the multiple routing table solution. I.e. all trees need to be computed and subsequently a shortest path on the corresponding topology needs to be performed. However, this is still an excellent reduction in the memory-footprint needed for IPRT.

The reduction in state information allow IPRT to be applied in a memory efficient way. Furthermore, since r/bTable allow IPRT to guarantee that only two FIBs are needed to provide IP fast recovery marking packets may be implemented. This allow IPRT to provide a per packet signaling that is efficient in therms of traffic overhead. In this thesis the r/bTable method will be used to represent the recovery FIBs.

5.3 Recovery

The redundant tree method was originally intended for global recovery in a connection-oriented network. The main goal for IP fast-reroute framework is to provide local recovery.

With redundant trees, it is possible to perform global recovery since it is based on a scenario where all the nodes in the network share the same view of the recovery paths. As with all recovery operations global recovery have a higher possibility to provide a shorter recovery path than local recovery.

However, to be able to support a true global recovery several support functions are needed. The node detecting the failure needs at least to inform the source of either at what component the failure was discovered or provide the source node with the correct recovery three, path or topology. At the source node, a synopsis of the destinations affected by the failure must be maintained in a soft-state data structure. These two procedures needs to be repeated for every IP-prefix affected of the failure, since the redundant tree method only guarantees recovery through either the red or the blue tree. Furthermore, the recovered traffic needs to be maintained in a soft-state to avoid the need for a source node to rely on signaling from the immediate upstream node of the failure to revert the traffic back to normal operation.

Even though there might be some gain in using a global recovery in terms of path length the total cost of maintaining a soft state data-structure of the traffic needed to be reverted to recovery operation and the cost of checking every IP packet for the need of recovery may be far greater than the benefits in reduced total load in the network. In addition, IPRT recovery method is intended to work as a buffer between failure and re-convergence. Thus, the time span the global recovery would be operational and effective would be even further diminished. This makes it unacceptable to utilize global recovery for IPRT.

5.3.1 Enabling local recovery

It is possible to utilize the IPRT information to provide local recovery. This is because each node in the network has its own pair of redundant trees and IPRT is therefore capable to recover traffic bound for any destination through either the red or the blue tree. Furthermore, since traffic bound for any arbitrary destination may be recovered, the recovery may be performed regardless of the source. This also holds true for the r/b Table solution since all possible destinations are present in both the rTable and the bTable. However, the use of a separate routing table for normal operation introduce a problem;

- In a failure situation, the immediate upstream node may experience a situation where it has the option to recover traffic using either the red or the blue path. However, because a separate routing table is used for forwarding in a failure-free environment IPRT has no information available on the possibility that one of the recovery paths may encounter the same failure further downstream. This situation requires the node initiating the recovery to have a node degree of at least three and furthermore, neighbors that are present in the real topology that are not adjacent nodes in the recovery trees.

Consider the situation shown in Figure 5.3. In this example the traffic from source node $R1$ traverse the failed node F during failure-free operation. Furthermore, during a failure $R1$ may freely choose between either the red or the blue recovery path. However, if the red path is chosen the recovered traffic would encounter the same failure a second time when being forwarded from router $R2$.

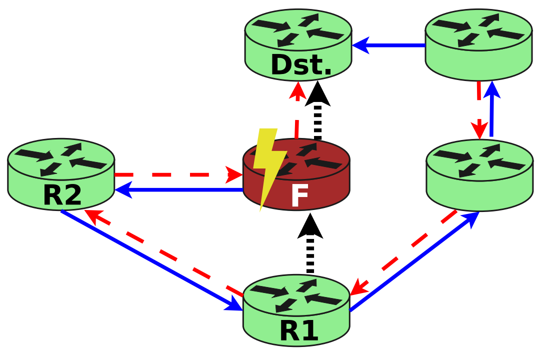

Figure 5.3: A node with different IPRT neighbors and real neighbors

To counter this problem some additional computation is needed in the IPRT routing procedure to be able to tell which routes are valid options in a recovery procedure. This information may be pre-calculated and made available to the recovery procedure in advance of any failure.

Two possible solutions are described:

- An exact procedure, where the healthy recovery path is identified by computation
- A probabilistic procedure, where potentially affected recovery paths are identified

The exact procedure is implementation dependant but logically each node needs to be verified to check if the failure-free, i.e. default, next-hop towards the root is contained among the red or blue next-hop of a redundant tree pair. If this is not true, the algorithm may need to traverse the red and blue path in order to verify which, if any, of the paths are affected by such a failure.

Another, and probabilistic, approach to the problem, that requires less computation, is to use a difference in the set of neighbor nodes found in the real topology and the neighbors found in a pair of redundant trees to result in successful identification of a preferred recovery path, i.e. no effort is made to examine what recovery paths are healthy. Furthermore, a positive identification results in a default route selection regardless of what path or paths provides a valid recovery path. However, the forwarding procedure must support to move recovered traffic between the red and blue path; assume that the red FIB is always selected in such recovery situations, and the only valid change of color is from red to blue. Furthermore, assume that a node F has failed and that the failure is affecting traffic forwarded from node $R1$, using a link different from the available red and blue next-hops. At this stage there is no easy way of evaluating voltages, i.e, if $v(R1) > v(F)$, thus it is by chance if the red path provides a working recovery path. If the recovered traffic encounters the same failure a second time being forwarded from $R2$ the voltage of the different nodes must be $v(F) < v(R2) < v(R1)$. Because $v(F) < v(R2)$ the blue path must always provide a failure-free path to the destination. This is because if the same error was encountered a third time $v(F)$ must have been higher than $v(R2)$, which is impossible in accordance to the voltage rule. This shows it is possible to deflect recovered traffic from between the two colors. However the deflection should only be done once to counter the possible loops that may arise from multiple concurrent failures.

The probabilistic procedure may result in undesirable behavior; increasing the network load and deflect healthy recovered traffic in the event of multiple concurrent failures. The use of deflection may result in longer recovery-paths. E.g. packets bound for a failed destination will be recovered at the last hop, furthermore, if they use the red recovery FIB they will be deflected when they are tried delivered to the failed destination a second time. When a recovered packet traverse an increased number of links it generate load at more links and thus increase the total load in the network. Furthermore, multiple concurrent failures may provide a problem. This is because the method may not be able to distinguish between two different failures. Consider some recovered traffic following a red path. If one assumes that the red path was the correct choice for the recovered traffic, i.e. the red path provides a path unaffected by the first encountered failure. If this traffic were affected by a second failure, the forward procedure would deflect the traffic to the blue path. This may lead to a situation where the traffic may loop back and encounter the first failure again. Thus, adding to the total amount of traffic without being able to accomplish a successful recovery. In addition, the deflection routing adds to the complexity of the forward procedure, as additional decisions become available. However, the deflection may enable packets to be successfully recovered from a second failure, i.e. the new blue recovery path does not necessarily equal the reverse red path, but this is outside the scope of this thesis. In addition, the deflection procedure does enable the IPRT method to operate in a transparent manner in the presence of ECMP routing if desired, but this is outside the scope of the thesis.

The exact procedure does enable the recovery procedures to pick the correct, i.e. failure-free, recovery path at first try. Thus, this approach does not have a negative impact on the length of a recovery path. In addition, this procedure trades computation that is more complex during the routing procedure to enable the use of a simpler forwarding procedure. I.e., Deflection routing may be used even if the correct recovery paths are known in advance to get the benefits of a possible better coverage during multiple concurrent failures. In addition, this approach may be utilized in a distributed fashion where each router only is required to verify their own neighbors. I.e., each router only needs to verify their own next-hops in each redundant tree set. Thus, the time needed to compute the affected recovery paths may not be of significance if compared to the deflection approach. However, the gain of using this mechanism in a distributed fashion varies between the various link degrees of the nodes in a given topology.

5.3.2 Representing the local recovery path correction

When the paths that may be affected by a single failure a second time are known, the information must be made available to the forward procedure. Since the situation where a recovery path includes the failed node as an intermediate node requires the next-hop on both recovery paths to be operational, the forwarding procedure needs to be able to make an informed decision. The idea presented here is to let a pair of bits indicate which recovery path is healthy.

Two possible solutions are described here:

- Utilize an additional recovery helper FIB. In such a solution, each vulnerable triplet of failed interface, destination and safe recovery path is maintained in a separate table.
- Store the information directly in the FIB. In this solution one or two, depending on the implementation, additional bits are used at each entry in one of the FIBs indicating the preferred next-hop.

When the recovery helper FIB is used, an additional lookup is required when performing the initial recovery procedure. However, if bits are stored directly in the FIBs, the needed information may be retrieved at the time of a normal table-lookup procedure. Thus, this approach may require a lower total number of lookups in the forward procedure during a failure situation.

The selected data-structure may have an influence on the total amount of state-information needed. When a separate helper table is used, only the destinations that have a recovery path that may be encounter the same failure a second time will be represented. The drawback of this solution is that each entry requires additional information, i.e. destination and failed interface, to be properly represented. However, when the information is stored directly in the FIB every entry must contain the additional bits. Thus, depending on the recovery paths and the topology, the two approaches have scenarios where they less state information than the other. Generally will the recovery helper FIB provide less additional state information when few paths must be identified.

Both the methods described provide the forwarding procedure with the amount of information needed to guarantee a successful recovery in the event of a single failure and are thus valid approaches. However, as shown in the next section (see 5.3.3), the bits may be used for other purposes increasing the amount of paths that must be identified by the bits. Thus this approach, from now named "Qbit", is used to store the needed information.

5.3.3 Quality bit (Qbit)

When a node upstream of the failure detects a failure and needs to recover traffic there is a possibility that the next hop for both the red and the blue recovery path is unaffected by the failure. For this to happen the upstream node needs to have a link degree of at least three, i.e. one failed and two remaining healthy links - one for each of the recovery paths. Furthermore, in such a situation it is also a possibility that the length of the two available recovery paths are of different size. The idea of the Quality bit is to let the node make an informed decision on which of the two paths it should choose, rather than leaving it to chance, and thus trading increased computational costs with a potentially better network utilization. When using this QoS optimization the paths that may be affected by failure have precedence.

The possibility that a single-failure should affect the next-hop of a packet and at the same time leave both the red and the blue recovery path usable is highly dependant on the topology, node-degree and algorithm used to create the cycle and arches in the IPRT recovery routing protocol. As an example picture a root node in a network where the initial cycle is created from the boarder nodes of the network. In such a situation there is a fair chance that node directly opposite, e.g. at the "other end" of the topology, of the root node use a shortest path traversing directly through the network as a working path during failure free operation. Thus leaving both of the red and blue recovery path available if a failure should arise.

An example is shown in figure 5.4 where the sender normally would send the traffic along the black dotted line. Thus when the middle node fails this leaves the node with the option of using either the red or blue tree. As shown if the blue tree is the longest, and could be avoided with if the router was given the needed information.

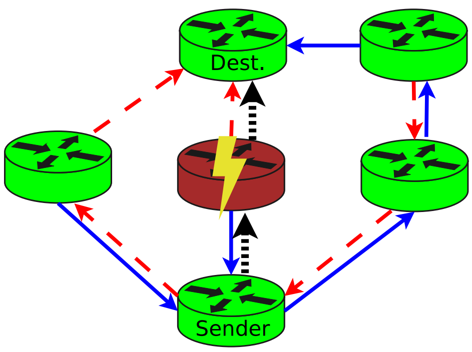

Figure 5.4: Two available recoverypaths in a r/bTable enabled network

To let the node choose the best recovery path this information needs to be pre-calculated along with the FIB at each node. A simple approach would be to register the length to each destination in the two trees, and then store this information in one of the FIBs, as described in the former section.

5.4 Forwarding

To be able to correctly forward packets in an IPRT enabled network where r/bTables is used in conjunction with marked packets for signaling, the forward procedure needs to be expanded. In this environment the forward procedure is expanded with two separate and distinct responsibilities; 1)the recovery procedure where the recovery FIB needs to be identified and selected, and 2)the subsequent forwarding along the next-hops based on the signaled FIB.

1. When a failure is present in one of the adjacent links or routers, the forward procedure at the nodes upstream of the failure must identify which packets are affected. Furthermore, the router forward procedure must consult both Qbit information and the recovery FIBs be able to determine the recovery paths that may be used.

2. Each individual packet needs to be inspected at arrival to identify recovered packets. If recovered packets are encountered the lookup procedure should do a table lookup in accordance to the signaled FIB ID. If no signals are present in the packet the forward procedure may use the default normal FIB.

The basic forward procedure may be done through a four-step procedure(see Figure 5.5). In the described procedure, the recovery helper FIB is a logical entity, and may be represented as a data structure of its own or information contained in the normal or red FIB as described in Qbit. The only difference being how much time is spent retrieving the information. After a successful lookup procedure, the packet must be marked according to the selected recovery FIB, if recovery was necessary, and subsequently forwarded to the selected outgoing interface.

First, the normal routing table is consulted. If this lookup returns a non-failed interface, the packet is not affected and may be forwarded according to the result. However, if the selected interface has failed the forward procedure must identify what recovery path is to be used. This may be done by first consulting the red recovery FIB to see if it has a valid outgoing interface. If the interface returned is a valid interface, the forward procedure must check the recovery helper FIB to see if the result is valid. If the destination is not contained in the helper FIB, or the helper FIB returns that the red FIB should be used, the packet may be forwarded according to the red FIB. However, if the recovery helper FIB returns that the blue FIB should be used the packet must be forwarded accordingly. Furthermore, if the red FIB returns the failed link the recovery procedure may proceed to try the blue FIB. If both the red and the blue FIB returns a failed outgoing interface the node is experiencing more than one failure, and the packet must be dropped.

There are several possible optimizations to this lookup procedure. For example, if Qbit is used the forward procedure is in effect a three-step procedure, and the placement of the Qbit information may help skew the possibilities of completing procedure within the shortest possible timeframe.

Another possibility is for the lookup procedure to rearrange the order of which the recovery tables are tried. To do this one could rearrange the tables such that the table

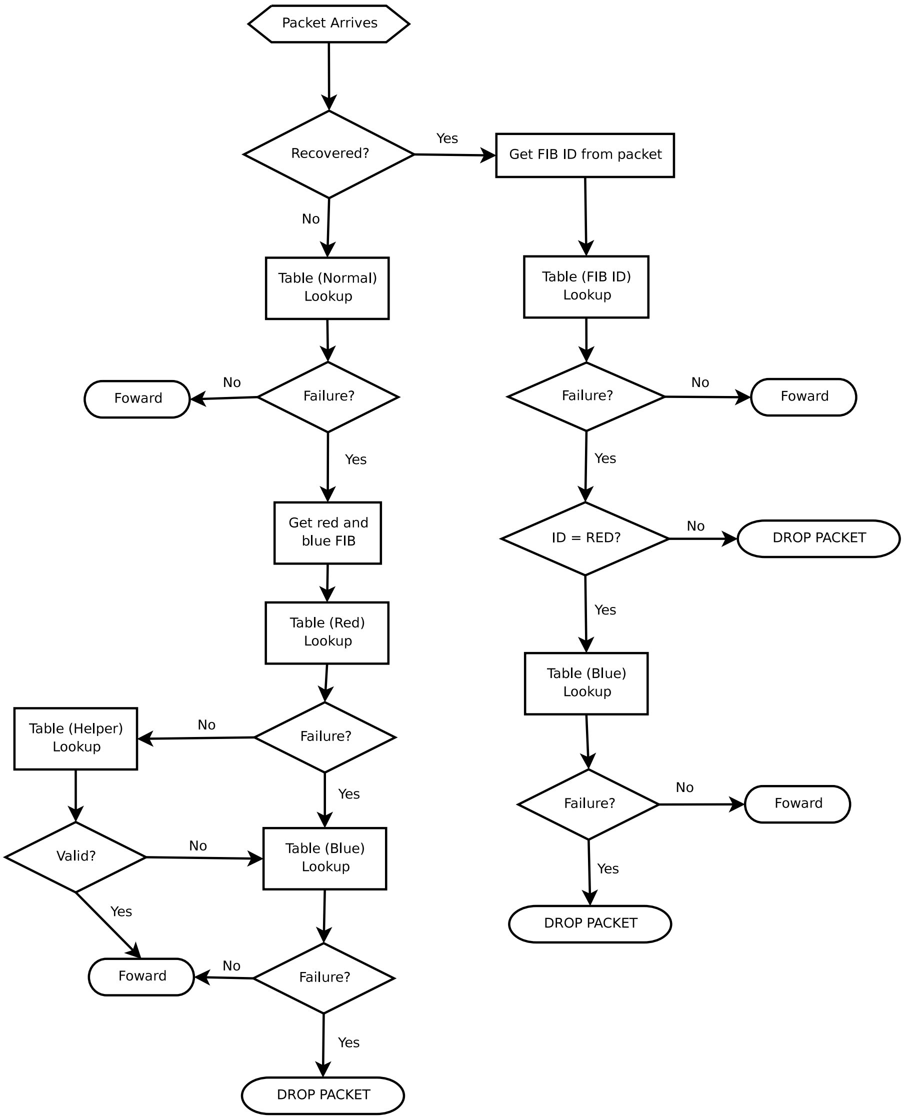

Figure 5.5: The forward procedure

with the least outgoing interfaces in common with the normal operation FIB is checked First. This may be done internally at each router if it does not affect the other routers in the network - i.e. the packets gets marked with the correct FIB ID regardless of the order they are tried.

Furthermore, if Qbit is not used but rather an external data structure, the nodes unaffected by the "hidden neighbors" problem described in the recovery section, may omit the check for a valid recovery path.

However, given a router with enough computational resources, these are optimizations done in the forward procedure and does not affect the actual performance of the IPRT method. In addition, the simulator is not likely to be able to simulate the exact time needed inside a router and even in the event of this being possible; the simulator model would grow immensely in complexity.

5.4.1 Selecting best FIB for storing Qbit

The exact location to place Qbit information may influence the number of needed table lookups in a failure situation. There are two possibilities when storing this information - the first recovery FIB to be accessed when a failure arise or the normal operation FIB. None of them adds to the worst case scenario in number of lookups, but they might change the distribution in number of lookups within the original range.

If the information is stored in the first accessed recovery FIB, i.e. the red recovery FIB then the average number of lookups might grow when compared to the original forwarding procedure. This is because the quality information introduces a situation where the first tried lookup might yield a valid result that at the same time is undesired. Thus, a second recovery lookup would have to be performed, and there is no guarantee that this lookup would yield an entry not affected by the failure.

If the information is placed in the FIB used for normal information this table would grow in size. The impact the increase in size would have on real performance is depending on the implementation of the FIB and the lookup procedure. However, when the information is placed in the normal operation FIB the number of lookups may be reduced. When a failure is detected the best table is then already known and the first recovery lookup may try this table. If the first choice fails the next table would be tried. With this approach the average number of lookups should be about the same as for the original forward procedure. However, since the lookup order might have been changed the average number of lookups might be skewed when compared to the original procedure, but this would be depending on both IPRT routing algorithms and topologies. I.e. this enables the lookup procedure to try the preferred FIB first and by doing this the chance of getting a valid result from the first tried recovery table that does not yield the preferred recovery path is non-existent.

Both solutions may be used with the r/bTable solution and only needs to use one additional bit indicating either the red or blue tree. For an implementation in a simulator the combination of choices and the placement of the bits becomes of less importance as the same amount of memory is needed for all combinations and thus only the number of lookups will have a real impact on simulation time.

Chapter 6

IPRT Implementation

The IPRT implementation is divided into two distinct parts; the IPRT tree generator, and the extensions to the J-sim network simulator. The tree generator is used to mimic parts of the IPRT routing protocol, and is responsible for creating the recovery topology for each node in the network. The J-sim extensions use these pre-calculated topologies to populate the FIBs at each node, and in addition provide the modules needed to simulate IPRT enabled networks.

The reason for separating the simulation environment and the IPRT tree generation, was initially to provide an easier start on the programming assignment for this thesis. This approach allowed the graph environment to be tailored for IPRT tree generation. Furthermore, several methods for mapping the resulting topologies were readily available in J-sim. Thus, this approach provided tried methods for representing topologies and populating the FIBs, and was therefore less prone to failure. After the generator was implemented, no big drawbacks were present, removing the need to merge the J-sim implementation and the generator. Another advantage is that the recovery trees of a topology may be computed once, and subsequently be used in an arbitrary number of experiments, thus cutting the time needed to prepare a simulated scenario. The separated environment allows the tree generator to be freely used independently of the simulator.

The implementation has four general requirements to the topology and the addressing scheme.

- The networks used must be at least two-vertex connected.
- The node id in the graphs, and the IP addressing scheme used during simulation, must correspond to each other.
- The nodes must be given addresses ranging from zero to $n - 1$.
- The link cost must be non-negative, and assigned in such a manner that the lowest weight indicates the link that is most likely to be used.

The reasons for these requirements will be clarified in the following sections.

6.1 IPRT Tree generator

The IPRT tree generator is implemented as a standalone application using the Java programming language. Its primary task is to calculate the redundant trees of a topology. In addition, the implementation provides an optional capability to consider QoS in the tree generation process. It may use link-weight when constructing the cycle and arches of a tree, and furthermore, it may be configured to compute QoS Qbit information.

The IPRT tree generator is implemented in a simple manner, following the original RT algorithm as closely as possible. As it is meant to be used offline, it does not focus on achieving optimized run-time. Thus, optimal run-time is traded for simplicity and a tidy code-base. To provide an easy way to change between different selection-algorithms should the need arise, the methods are kept as modularized as possible. The IPRT tree generator contains several classes, shown in Table 6.1.

The core responsibilities for the IPRT tree generator are:

- Create a graph representation of the original network.
- Generate a series of redundant tree pairs, so that every node becomes the root node of a red and blue tree.
- For each node, generate Qbit information needed to ensure a successful local recovery procedure.
- Present the created recovery trees and Qbit tables so that they may be easily utilized in the J-sim environment.

The responsibilities are described in more detail in the following sections.

Class	**Description**
Node	Used to represent the nodes in the network.
Link	Used to represent the links in the network.
Graph	Administration of links and nodes.
RTGen	Main class.
Qbit	Used for creating Qbit information.
CreateTreesQoS	Used for creating redundant trees.
VerifyTrees	Used to verify properties of the redundant trees.
NodeOrder	Used for sorting and comparing nodes.
FileHandler	Used for reading and writing files.
Djikstra	Djikstra implementation.

Table 6.1: RTGen classes

6.1.1 Creating the graphs

When the IPRT tree generator starts, its first responsibility is to create a graph representation of the network from an input topology. This graph is used provide the foundation from which the redundant trees may be generated. Furthermore, at a later stage, the graph environment is used to represent the redundant trees. In this section, the initial mapping of the original graph and the options offered will be presented.

The IPRT tree generator models the network by representing both links and nodes as objects. This approach gives a good representation of the network, and allows arbitrary amounts of information to be associated with both links and nodes. Furthermore, an adjacency list representation is used to connect the network entities. The list representation forces any search for a given node or property to iterate the list from the top. However, it is not affected by any decline in connectivity, and is the preferred data structure for any graph that has sparse connectivity, e.g. $|e| < |v^2|$. However, the amount of nodes to be used in the networks that the IPRT tree generator is to compute, is not big enough to have any great influence on the selection of the data structure. Nevertheless, a linked list representation was used in the implementation, as this is the standard for low connectivity graphs.

The implementation only supports one link between any given pair of nodes. This is done to provide an easier graph environment for the IPRT tree generation algorithm. Furthermore, it was not needed more than one link between any nodes in the topology used to conduct the experiments.

Specifying link-cost

The most important option, that may be specified when creating the initial graph representation of the network, is how the link-cost is represented. Depending on the choice, this will, at a later stage, decide if the redundant trees will be created according to the link-cost specified by the input topology, and thus show the ability of IPRT to respond to the use of link-cost.

When mapping the original network, the IPRT tree generator provides the option of using the link-cost specified by the input topology, or to ignore this information, and set a flat cost on all links. The default behavior of the generator is to use a flat link-cost. Because of limitations in the IPRT selection-algorithm, the weight used to represent the link-cost must be non-negative. Furthermore, it is assumed that the link-cost is symmetric, i.e. the cost of a link does not depend on which direction the link is traversed. The reason for choosing this approach, was also to provide an easier environment for the IPRT tree generation algorithm.

6.1.2 Creating the redundant trees

The redundant trees created in this implementation support recovery of traffic caused by a single node failure. Furthermore, the tree generation process honours the link weight specified in the topology when constructing the cycle, and subsequently arches. In this section the method used to create the trees is reviewed. The implementation has three main tasks it attends to:

- It must ensure that all nodes is the root of a pair of red and blue trees and inform the user if this fails.
- It is responsible for identifying the cycles and arches that form the redundant trees.
- It must guarantee that the trees are generated in a correct fashion according to the voltage rule.

To create all the redundant trees, the process iterates through all the nodes in the network and uses each node as the root of a red and blue recovery tree. If it at any stage fails to build the tree pairs, the computation will be halted, and an error message produced.

When creating each individual pair of redundant trees, the IPRT algorithm tries to follow the original RT algorithm as closely as possible, i.e. the redundant trees are generated by growing them in a series of stages. Furthermore, the algorithm used in IPRT is implemented in a greedy manner, where the algorithm, at each stage, choose the shortest cycle and subsequently arches available.

The selection-algorithm used, i.e. the algorithm that chooses the cycle and the arches, is a simple implementation of the Dijkstra algorithm that considers the weight of links when searching for the shortest path tree. This algorithm was chosen in order to keep the length of the cycle and arches to a minimum at each stage. Furthermore, it is implemented in such a manner that it is able to avoid any nodes that are already contained in the redundant trees generated at this stage.

A candidate node indicates a node that is not in the tree itself, but which has a neighbor contained in the tree, i.e. it is a node at the border of a node already added to the trees and thus a strong candidate to be incorporated in the redundant trees at this stage.

To generate the initial cycle, the candidate nodes are identified. Furthermore, a shortest-path tree is generated for each candidate node. In this shortest path calculation, the root is avoided and is thus not included in the search. Subsequently, each tree is checked to find the path that provides the smallest cost between any two of the candidates, including the cost needed to reach the root node from each tested pair of neighbors. The shortest of these paths is subsequently used as a basis for the initial cycle. If no path is found between the candidate nodes it is considered a failure, and the execution is halted and an error-message produced. This approach guarantees that if the initial cycle is found, it contains at least two nodes in addition to the root node. A possible optimization could be decreasing the number of shortest path tree calculations. However, such an optimization was not implemented.

The same procedure is used to identify the shortest arch at each subsequent stage. In addition to calculating the shortest path trees from each candidate node, each candidate is individually examined to determine if it has two or more distinct neighbors that are contained in the redundant trees. By adding this last check, the arches may contain one or more nodes in addition to the start and end nodes, that are already part of the redundant trees.

When assigning the voltages, the maximum voltage is derived from the number of nodes in the network. Furthermore, the voltages assigned are distributed evenly within a voltage range. The voltage range is constrained by the highest of the two voltages obtained from the head and tail of the cycle or arch, and the voltage assigned to the tree that is immediately beneath the first. With this approach, the worst possible assignment scenario entails that arches of minimal size is added in such a fashion that the available voltage range is cut in half with each new added arch. To counter this effect, the implementation represents the voltages using a double representation. Additionally, a relatively large maximum voltage is used. If the voltage range should prove to be too small to be represented, the implementation will halt the tree generation, and produce an error message. Other valid approaches might have been to provide a max voltage that could be proven large enough given the worst-case scenario, or to rearrange the voltages, should the need arise. If the voltages were to be rearranged, it would be important that the ordering of the nodes was kept intact. However, the worst-case scenario is unlikely to happen in practice, and the implemented approach has proven to be adequate for the topologies used in this thesis.

By utilizing this greedy approach to construct the trees, the implementation provides an approximation to the optimal solution for each pair of recovery trees. The reason for choosing this approach, is that it might resemble an approach that could be practical to implement in a real-life routing protocol. In this manner, the greedy approach will provide with results that are more realistic.

6.1.3 Calculating Qbit information

The Qbit information calculated in the implementation may be used for two purposes. Firstly, it is to be used to ensure the provisioning of local recovery. Secondly, it may be used for selecting the shorter of two valid recovery paths when available. In the implementation, the bits set to ensure correct forwarding has precedence over any QoS optimizations. Both methods for populating the Qbit are optional, allowing for more experimentation with different configurations.

The Qbit has three states. The two first indicate if either the red or the blue tree provides with the shortest recovery path, and whether the cost towards the destination is equal for both recovery paths. The last state is not needed to populate the Qbit table correctly, but is used in the implementation to enable the measurement of the scenarios where this mechanism is useful.

In the design chapter, two options were presented for creating the Qbit information used to ensure a successful local recovery procedure: The Qbit information could be populated with exact results, identified by the recovery paths that will traverse a failure, or a probabilistic approach that forces the use of the red path, if the neighbor topology could result in a recovery path that traversed the failure. In this thesis, the latter approach was used.

The reasons for choosing this latter approach were twofold. The implementation of the static routing protocol in the J-sim simulator uses a specialized and optimized shortest-path route computation, making it difficult to identify the shortest path used during the simulation in the IPRT tree generator. Furthermore, the graph representation used in the J-sim implementation differs from the representation used in the IPRT tree generator, making it hard to map the outgoing interfaces to correspond between the IPRT tree generator and the J-sim LSDB. Thus, the implementation of the tree generator populates the Qbit based upon the presence of a topology environment that potentially could result in an invalid recovery path. It is anticipated that this approach may provide less than ideal QoS Qbit results, as the implemented approach may identify false positives, reducing the ability to freely configure the QoS Qbit for additional source/destination pairs.

In order to calculate the Qbit information used to ensure a successful local recovery, the implementation iterates through all pairs of red and blue recovery trees. For each set of trees, all the nodes are examined to see if the union of neighbor nodes contained in the red and blue tree, differ from the neighbor nodes present in the original topology. If there are more nodes present in the original topology, the red and blue recovery paths to the root node are traversed in order to check if any of the missing neighbors are present in these paths. If missing neighbors are found in both paths, the Qbit is set to follow the red path towards the root of the tree. If they are present in only one of the paths, the opposite color is set in the Qbit table.

The method used for calculating the QoS Qbit uses a shortest-path algorithm that calculates the cost from the root node to all other nodes in each tree. For each pair of redundant trees, the nodes are examined to see if the cost differs between the two trees. If the Qbit is not already populated by the forwarding correction method, it is set to identify the shortest possible path towards the root of the tree.

6.1.4 Results

The results obtained by the IPRT generator are written to three different files, separating the topology representations, Qbits and link-cost.

6.2 J-sim implementation

The extensions to the J-sim implementation cover a diversity of tasks. However, the most important are the methods that support populating multiple FIBs at each node, i.e. the routing protocol, and the ability to perform recovery and properly forward traffic both affected and unaffected by a simulated failure, i.e. the forwarding procedure. In the simulation environment, two stages logically separate the implementation. The "Setup-stage" provides the methods needed to configure the simulated environment before the simulation is started - such as building the network, populating the FIBs, preparing the traffic generators, and scheduling failures. The next stage, the "Operational-stage", provides methods for correct IPRT simulation according to the configurations performed in the "Setup-stage", such as forwarding with or without IPRT recovery, simulating failures and performing measurements.

Many of the J-sim extensions implemented in this thesis are inspired by the J-sim implementation of RRL. Most of the class structure is gathered from this implementation, and some of the classes are used with only minor changes to variable names. The changes in variable names was intended both as an exercise to enable understanding of the structure, and to separate the two implementations. These classes are used to support or provide functionality outside the core of IPRT. Classes that only have been subject to minor changes are CoreLayer, Shiva, RTConstants, RTPacket, Builder, LinkCost and NamTrace. The classes that provide the core IPRT functionality, i.e. the routing protocol and the forwarding procedure, and other utilities used to configure the IPRT environment, are completely rewritten, or altered to a great extent. The different classes are shown in Table 6.2.

Class	**Description**
Builder	Used to connect the components used in the simulator.
CoreLayer	Container Class for building a network node.
Dispatcher	Used to implement the IPRT forwarding procedure.
GetHops	A J-sim application for probing path lengths.
GetHops_pkt	The packet used by the GetHop application.
LinkCost	Used to represent the weight of a link.
NamTrace	Used to produce a tracefile of all network traffic.
Routing	Used to implement static routing and IPRT recovery routing.
RTConstants	Used to hold constants used in the implementation.
RTFileTopology	Used to read the files produced by the IPRT tree generator.
RTPacket	Extension of the IP packet format, adds a field for tree color.
Shiva	Used to trigger faults in the network.
Util	Used to configure the simulation scenarios.

Table 6.2: IPRT J-sim extensions

6.2.1 Routing protocol

The implementation of the routing protocol is an extension to the original static routing protocol found in J-sim. This is done to add the cabability of populating more than one FIB, and to provide the functionality needed to create the entries in the r/bTables. The routing protocol may only be utilized in the "Setup-stage" of the implementation. To be able to address more than one FIB, the implementation stores the different FIBs in a static manner, using an array, where the first entry is the routing table used for normal operation. The rTable and bTable are stored in the second and third entry in the FIB array, respectively.

When populating the FIB used for failure-free operation, the static routing protocol follows the standard behavior of the J-sim implementation. This implies that an implementation of the Dijkstra algorithm is used to calculate the minimum shortest path trees rooted at each node in the network. Next, this information is used to populate the FIB used for normal operation in each node. The only change in this procedure is that it is possible to specify which FIB to populate.

The static routing protocol supports non-negative weighted links. In the implementation of the utility functions, preparing the network expects the link-cost to be specified even if it only contains a unified flat cost on all links. Thus, the link cost is always considered when using the Dijkstra implementation found in J-sim.

Populating the r/bTables

To populate the r/bTable, the J-sim implementation reads the redundant trees created by the IPRT tree generator in such a way that the root and color of each tree is identified correctly. Furthermore, the trees are used in turn to populate a single FIB entry, according to the tree color, each node identifying the next-hop towards the root node. By iterating through all the trees obtained from the IPRT tree generator, the r/bTables are fully populated.

In the implementation, the trees are not modelled as virtual topologies, but rather used to assign new link-costs to the original topology. In this process, all the weights of the links are initiated with a high cost and subsequently the links found in a specific tree is assigned a low cost.

Subsequently, the original topology and the newly obtained link cost matrix are used in a modified version of the J-sim static routing protocol. The link cost forces the J-sim implementation of the Dijkstra algorithm to follow the path according to the redundant tree, when calculating the shortest path tree rooted in the root node of the redundant tree. When the shortest path algorithm has finished, only the entries identifying the next-hop towards the root are installed at the FIBs. Which FIB to update is determined by the color of the tree being processed.

By using this approach, the interface ID, i.e. the ID identifying the different links attached to a node, does not need to be synchronized or translated between the IPRT tree generator and the J-sim implementation. The drawback of using this approach is that unnecessary calculations are done in the Dijkstra algorithm, as shortest-path trees rooted in each node are calculated - even though only the root node of the redundant tree being processed is used.

Preparing the Qbit information

The Qbit information is read from the Qbit file generated by the IPRT tree generator, and made available to the forwarding procedure at each node.

The Qbit information is not stored as a part of the routing table, but rather contained in a separate data-structure. The reason for choosing a separate data-structure, was that this approach provides the simplest solution, as it does not require any changes to the FIB entries or to the table-lookup procedure as found in the J-sim implementation.

Furthermore, the implementation requires the nodes to be given addresses, ranging from zero to $n-1$, to minimize the time needed for obtaining the Qbit information. This reduction may be achieved because the addressing scheme allows the Qbit information to be stored in an array, where the entries correspond to the address of the different destinations, enabling the Qbit information to be retrieved in $O(1)$ time given a known destination.

6.2.2 Forwarding

The forwarding procedure has been implemented using the SIMULA RRL J-sim implementation as a template, where key parts have been rewritten to fit the IPRT implementation. In essence, the implementation of the forwarding procedure of RRL and IPRT follows the same general procedures. For example, recovered packets are marked in the IP header, and receiving a packet so marked requires a table lookup in a recovery FIB. Furthermore, available safe layers or tree colors must be identified during the initial recovery procedure.

Generally, the dispatcher, i.e. the entity containing the forwarding procedures, may work in two different environments; either in a failure-free environment or in an environment where local failures are present.

The implementation of the dispatcher will detect if a packet has been attempted forwarded over a failed link. Furthermore, the packet will go through a series of checks to identify if the packet already has been recovered, or if this is the first recovery attempt.

If the packet has never been recovered, the IP header of the packet is converted to an IPRT header. The new header includes an extra field containing the FIB ID, keeping the value of all the original fields in the IP header. To populate this header, the dispatcher provides a method for determining which of the recovery FIB are to be used. The implementation of this procedure follows a two-lookup procedure. This implies that as the normal routing table is verified to have failed, one of the recovery tables must provide a valid outgoing interface. When entering this procedure, the Qbit is obtained before any table lookup, emulating a situation in which the Qbit information was made available through the normal operation FIB, or obtained from a separate data-structure in advance of identifying the recovery FIB. When deciding which tree to choose, the procedure first attempts the red tree, and if it returns a valid result, and the Qbit corresponds to the

FIB color, the FIB ID is returned. If it returns an invalid interface, or the Qbit indicates that the blue FIB gives the best result, a second table-lookup is performed, returning the blue FIB ID if the table lookup was successful. However, if the latter returns a failed link, the red FIB ID is returned anyway.

This approach differs from the ideal implementation given the time the Qbit was known. However, it is a valid approach, and it gives the opportunity to more closely monitor the gain from using QoS Qbit. Furthermore, the packet is re-queued to be forwarded by the router. This approach has no effect on the headers, as they are updated at arrival, and not in the forwarding procedure.

In a failure-free environment, the IP header of each packet is checked to see if it matches the IPRT header. If this is the case, the forwarding procedure routes the packet using the FIB identified by the header, otherwise the normal FIB is used.

Deflection

The Qbit information responsible for providing a guaranteed successful local recovery procedure, is computed with the probabilistic approach (see section 5.3.1). This makes it necessary to support deflection, i.e. requiring the forwarding procedure to move recovered traffic from the red to the blue path if a failure is encountered twice. This is done by expanding the forward procedure.

When the forwarding procedure receives a packet already encapsulated in an IPRT header, it may either be a second failure or the red recovery path has been followed and the same failure encountered for a second time. Thus, the forward procedure checks to see if the FIB ID registered in the IPRT header is equal to the red FIB ID. If this is the case, the IPRT header is updated to equal the blue FIB ID and sent to be forwarded again. Furthermore, if the packet contains a blue FIB ID, this is registered as a scenario where a second failure has been encountered and the packet is dropped.

6.2.3 Utilities

The implementation contains several utilities. Generally, these are designed for automating tasks that would otherwise have to be done by hand during configuration of a simulation scenario. In addition, some utilities are designed to help obtaining results in the experiments, and have a direct influence on how the experiments are conducted. The two most important utilities used for obtaining results in experiments are described here.

Traffic generator

The traffic generator may be used to install traffic generators on each of the nodes in the network. In the implementation, a Poisson traffic generator, included in J-sim, is used as a basis for producing traffic. In each node, several generators are installed, each representing the different destinations the node may send traffic to.

The implementation allows refraining from installing sources bound for a specific node, in addition to avoiding installing any sources on the specified node. This is done without changing the rate at which the sources generate traffic, had the node not been removed.

GetHops

The GetHops application is written to obtain the path length between a source and possible destinations. It uses the "raw" IP protocol, and provides no guarantee for delivery.

The application may be used to obtain path length from a source to a single destination, to all destinations, or to all destinations minus one specified exception. It also has the ability to perform two separate measurements during a scenario, and comparing them with each other.

If a destination is unreachable, GetHops does not provide any information as to why. This due to it not being able to guarantee the delivery of the probe packets. However, it has functionality to detect whether the number of replies differs between two separate measurements, or if the number of replies differs from the number of requests.

Chapter 7

Design of simulation scenarios

The authenticity and trustworthiness of the results obtained from simulation scenarios will depend on how the network is modelled. In this chapter, the key characteristics of these models are reviewed, as well as alternative methods for their implementation.

7.1 Network

The network topology, and the manner in which it is modelled, forms the foundation for how the simulation scenarios may be implemented. In this section the key properties are discussed.

7.1.1 Topology

The network topologies play a crucial role in how well IPRT, and any other multi-hop recovery schemes, performs. The most important property of the networks being modelled, is that they are two-vertex connected. This is needed for IPRT to provide full coverage for both node and link failure, and is a requirement for all topologies used in this thesis. However, within the boundaries of this criterion, it is expected that differences in network properties may influence the performance of IPRT. In this section, the main attributes, and different means to obtain the topologies, will be discussed.

Some of the main attributes of a network topology may be described through average node degree, connectivity and layout.

Among the various topology properties, it is expected that the average node degree is the most influential. Generally, an increase will result in more links to be present, and thus more options are available during the tree generation phase of IPRT. This enables the trees to contain smaller cycles and arches and thus provide shorter recovery paths.

The vertex connectivity of a topology is defined to be equal to the number of vertices that may be removed, before connectivity is lost between any of the nodes in the network.

Following Menger's theorem, it is known that IPRT requires two-vertex connectivity to be able to construct the red and blue trees. However, if the connectivity rises, more paths become available for the IPRT tree generator to use.

Even when two networks has the same average node degree and connectivity, there may be differences in the performance, caused by topology layout. For example, if nodes with low connectivity degrees are located at the edge of a network, the result may be a minor reduction of the average recovery path length of a recovery scheme, compared to a topology where they are located at the edge of the network. This is because a failure at the edge of the network is likely to affect traffic less.

The topologies may be whether modelled from real or synthetic networks. Many network operators, especially from big networks, make available data from the networks. For example, Sprint makes available large amounts of data, that may be used to produce high-fidelity simulations, based on real network topologies, and traffic data that may simulate a real event. Another possibility is to utilize topology generators. Several free and commercial applications can be used to generate synthetic topologies.

The difference between using a synthetic topology and real ones does not have a great impact on RT. However, real topologies are generally both known and used in measurements, and thus make it easier to obtain and compare results across studies. A drawback of using real topologies is that their number is limited as opposed to those from topology generators. Conversely, a drawback to using a synthetic topology is that design issues are not as well-validated as real topologies. Consequently, when real topologies are used, there is generally a better chance of the networks having been subject of both discussion and informed design decisions.

For this thesis, a mix of well-known real and synthetic topologies will be used. Furthermore, the topologies should provide variety in the network properties.

7.1.2 Link capacity

Link capacity is a property of networks that sets the limits on how much traffic it may successfully transport. As offered traffic in the network converges towards the limits found in the links, packets will be dropped. It is usual to utilize mechanisms that ensure that this congestion does not interfere with the routing protocol, allowing routing traffic to continue.

In a network, it is not unusual to have some diversity in the link capacity, and to use traffic engineering to distribute the load in an efficient manner. However, when recovery procedures are introduced in networks, a failure will result in traffic being moved from its usual routes, and introduced into other paths in the network. This could lead to situations where the offered load exceeds the available capacity and result in packet drops.

If it should be desirable to determine whether this scenario happens during simulation, the load of the network should be modelled after the load usually found in the simulated networks. Also, the link capacity should be set in accordance to the topology information available. If congestion occurs, the total offered load on a link may be measured as a combination of link load and dropped packages. However, if it is not desireable to observe packet drop as a result of congestion, the link capacity may be set relatively high as compared to offered traffic. This does only affect packet drop; it is still possible to observe if the traffic on a specific link is higher than expected.

When the link capacity and offered load is modelled to mimic that of an actual network, the simulation results give the most accurate description of the performance of the recovery procedure. The disadvantage to this approach, is that it is much more complex both to model, and to measure. Furthermore, when the link capacity is set so that the link does not risk experiencing congestion, the results obtained may also be used to measure the same effects that can be found in the more accurate simulation scenarios.

In this thesis, the link capacity will be set to such a value that no packet loss may result from lack of bandwidth.

7.1.3 Link metrics

Link metrics may be used to influence the path selection of a routing protocol when several paths to a destination exists. By associating a cost to each link, the network resources may be utilized in a more efficient manner, allowing the routing protocols to provide a higher quality of service to the network. In this thesis, both the J-sim static routing protocol, and the IPRT tree generator, may use link metrics when calculating the paths. Furthermore, they both use the Dijkstra shortest-path algorithm to calculate the paths.

The routing algorithm governs the metric values that may be used. When a Dijkstra shortest-path algorithms is used, the link cost may not be negative, as this would keep the algorithm from terminating. Furthermore, the link cost must be of such a nature that the link most likely to be used, is represented through the smallest values in the different link metrics of a network.

One of the big problems when using metrics, is that the different recovery procedures might need different metrics to perform as optimally as possible. The optimal metrics are often defined through heuristics. In RT and thus in IPRT, there is still ongoing research in trying to find good heuristics, but this is not part of the goals for this thesis. Nevertheless, the use of metrics may give an indication of whether IPRT is able to respond to them. Thus, the thesis will include the metrics obtained as a part of the network topologies, even though there is no guarantee that they used will produce good results.

There is not necessarily a connection between the link capacity and the link metrics. Even if the link capacity is not set according to the real topologies used in this thesis, the metrics may help achieve correct results. However, the metrics should be derived from the real topologies, as a mismatch between the link bandwidth and the metrics used when creating the FIB, might result in the traffic following paths not intended for heavy use.

7.2 Traffic

When traffic is generated, it forms a network-wide load pattern, i.e. the load found on all links at a given time or interval. When a failure occurs in a network, the traffic affected by the failure will be rerouted - following new paths in the network and forming a new network-wide load pattern. How well IPRT is able to distribute this generated load is of importance when evaluating its performance.

Characteristics in the traffic imposed on networks may play a crucial role in evaluating the performance of recovery schemes. There are two major influences to the performance and behavior of networks during a failure; firstly, the amount and destinations of traffic introduced in the network, and secondly, in what manner the traffic is transported. In this section, these properties will be examined to provide solutions used in the simulation scenarios in this thesis.

7.2.1 Transport layer protocol

The IP packet structure is designed to provide a minimal set of mechanisms to transport traffic in a connectionless environment, and has proven itself a versatile packet representation, leaving the preferred choice of traffic delivery methods to the upper layers. TCP and UPD are two commonly known transport protocols used to extend the capabilities of IP, and both transport protocols are implemented in the J-sim simulator. Generally, TCP is used to transport the majority of internet traffic (72-94 %) while UDP is the other significant contributor (5-27%) [30]. The protocols differ in the manner they deliver packets, and this difference may influence the obtained results, or the manner in which the simulations are conducted.

The main feature of the TCP transport protocol is to provide mechanisms for the reliable delivery of packets. One of these is TCP slow start, which adjusts the transmission rate in accordance with the perceived capabilities of the network and the receiver. When a packet is initially affected by a failure, it will need to be re-routed using an alternative path to reach the destination. If the packet needs to traverse more links according to the new path, this may result in a temporary change in the rate the sender receives acknowledgements. This may lead TCP to interpet the change in the reply rate as congestion, or loss of packets, triggering the TCP slow start mechanism. Initial tests with the J-sim TCP implementation revealed that the TCP streams did go into TCP slow start during a simulated failure - even though recovery mechanisms were present.

The approach used by UDP does not differ much from a raw IP delivery protocol, as it provides a "best effort" delivery scheme of packets, that is, no guarantees are made as to delivery. The UPD protocol provides an option to use a slow-start mechanism, but this mechanism will not be used in this thesis. Accordingly, when UDP is used to transport packets in a network experiencing failure, the sender is not informed whether delivery was successful. This enables any source using UDP as transport protocol to send at a constant rate, regardless of the condition of the network itself.

When using TCP, several of its specific mechanisms influence the network-wide load during failure. Traffic transported using TCP will experience a TCP slow start, where the sending rate is reduced to a minimum for a period. For a period following immediately after the failure, the load will be unaffected, as recovered packets traverse the new path to their destination. Furthermore, after the sources intiate the TCP slow start mechanism, the length of the period during which traffic is restored to its full sending rate, depends on both the time needed for the J-sim TCP implementation to itself restore the transmission rate after the TCP slow start, and the propagation time of a packet between each source and destination. The impact of the TCP slow start relation would depend on the duration of the failure. Furthermore, when the failure is prolonged, and no recovery procedure is offered, traffic bound for a failed node will experience a series of TCP slow starts, starving the network of this traffic. By using UDP for transport, the simulation scenario becomes easier to administrate, as the transmission rate does not depend on the network condition or the actual delivery of the packets.

Considering the distribution of the transport protocols used in the Internet, a realistic approach would be to utilize a combination of TCP and UDP. However, by using TCP, the construction of a simulated scenario becomes more complex both to configure and to analyze. By only using UDP, the simulated scenario may represent a worst-case scenario. This also removes the dependency between the duration of the node-failure and the obtained measurements, and shows the results that would be obtained if the failure was long-lived and all traffic bound for a failed destination was using UDP.

For this thesis, the transport protocols used will be either UDP, or a pure IP solution. The reason for choosing this approach is to enable measuring the generated load in a worst-case scenario, and to provide an easy environment to analyze.

7.2.2 Generating traffic

There exists a number of ways to model the traffic to be introduced into a network. Generally, it is desirable that the generated traffic resembles that of a real-life network, as this gives a good foundation for evaluation of actual performance. However, properly generating traffic that resembles the traffic found in the Internet is a topic that is heavily debated, without a clear consensus. Thus, a model has been chosen without too much discussion on all the available options.

The traffic introduced into the network should be extensive enough to ensure to accurate verification of network performance. E.g. if a traffic generator only produces one single packet every second it would be hard to determine any effects in the load patterns.

To generate the network-wide load two techniques are used:

- A traffic generator based on the Poisson distribution, to generate the load between all sources and destinations.
- A traffic matrix is used to individually scale the load of the traffic generators, in order to provide load-distribution.

In the article by Thomas Karagiannis et.al. [31], the authors anticipate that Poisson models may be used to accurately model Internet backbone traffic on sub-second time scales. Furthermore, the articles states that simulations may get sufficiently accurate results by varying the arrival rate of a Poisson process. In J-sim, several traffic generators are available, and for this thesis, a Poisson traffic source from the J-sim package will be used.

The traffic will be weighted according to a traffic matrix made to mimic the load-distribution found in each individual network [32]. The traffic matrix describes the demand between all sources and destinations, but it does not describe how the traffic is routed, nor the amount of traffic to be generated. The use of a traffic matrix provides many features not covered in this thesis. Commonly, the traffic matrix would be scaled to match the generated load to the link capacity. However, in this thesis, it is only used for generating a credible load pattern between sources and destinations, as the link capacity of the network modelled does not reflect actual capacity. This is because the goal for this thesis does not include determining the best load distribution IPRT is able to provide, but rather to determine if IPRT is able to distribute load at all.

In the thesis, a total load generated by all sources combined will be specified. This enables the J-sim traffic generator utility, described in the IPRT implementation chapter, to calculate the average load generated by all Poisson sources. Furthermore, the different traffic sources are scaled according to a traffic matrix, which specifies a relative ratio of the amount of traffic bound between every source/destination pair in a given network. This enables the traffic to resemble the network-wide load pattern that cold potentially be present in the topologies.

7.3 Failure models

The failure model used in the simulator governs the results that are possible to obtain from the simulated scenarios and the time needed to simulate all possible failures. In a real network, failures may originate for a multitude of reasons, differing in scale and severity. They may be either physical or logical, and arise from either external or internal causes. An example of an external physical error could be a cable cut, while an internal logical error could be caused by an erroneous configuration. Furthermore, not all errors occur by accident, but are rather the result of planned service events - hardware upgrades or large configuration changes, for example. However, in this thesis, a simpler failure-scenario resembling severe physical error will be used - simulating a complete failure, where the failed entity ceases operation.

The failure models that may be used include:

- Node failure, the model which provides the most comprehensive failures, as a node failure also entails a failure on all adjacent links of the node.
- Link failure, which produce a sub-set of the failures generated in the node failure model. It allows the nodes to remain operational, and retain functionality on all operational links.

The node failure model is the model that provides the most comprehensive failures, as it entails a failure on all adjacent links of the failed node. Because IPRT has been designed to withstand node failure, this property should be verified in the simulated scenarios. Node failure should also be the failure model that provides the fastest access to a full verification of the IPRT model. However, by using node failure as the only failure model, the "last-hop problem" may not be confirmed in the simulator. This is because the "last-hop problem" deals with how to recover traffic when the node immediately upstream of the failure is the last-hop node before reaching the destination address, and the cause of failure is unknown.

The link failure model will produce a subset of failures generated in the node failure model. It may not be used to verify the IPRT method, as it is not capable of providing an environment where traffic meant to avoid a specific node is easily detected. To detect such traffic would require additional information in the recovered packet, and specific detection logic would be required in the dispatcher. However, by using this failure model, it may be shown in the simulated environment that the IPRT method solves the "last-hop problem".

It would be possible to utilize a combination of the two failure models, to show that IPRT is capable to recover traffic when a node failure is present, and to solve the "last-hop problem" when a link failure is present at the destination node. However, this would require a large increase in the number of simulation scenarios.

Because of a desire to keep the number of simulations down, and that it has been shown in theory that IPRT is capable of solving the "last-hop problem", the node failure model will be used in this thesis. Furthermore, if IPRT is capable of providing fast recovery during all possible node failures, there are no good reason for it to not be capable of solving the "last-hop problem".

7.4 Modeling re-converged scenarios

The re-converged results are not modelled accurately in accordance to a real scenario. However, the model ensures that the failed node in the network is not used to generate, receive, or forward any traffic. The differences may be found in the way the routing tables are created.

- The routing tables used in the re-converged scenario contain an entry to the failed node.

To ensure that the failed node is not used as an intermediate node for any traffic, the adjacent links are given a very high cost when initializing the scenario. The cost is set so high that the shortest-path static routing algorithm does not use the node in any path, except where the failed node is the destination.

When simulating the re-converged scenarios, the failed node is failed with the same mechanisms used to model failure when simulating an IPRT or RRL enabled network.

This ensures that no traffic bound for or traversing the failed node may be successfully delivered.

To verify the behavior of the re-converged model, the coverage of the re-converged scenario may be tested. It should provide full coverage when excluding the failed destination.

7.5 Performing measurements

When performing measurements on a simulated scenario, some changes in the network state may need a transient period. More precisely, a period during which the state change fundamentally alters the network behavior, introducing a duration where some time is needed for the network to reach a new equilibrium. During this period, any measurements could potentially provide data that may skew the results in an unrealistic direction. For example, if the J-sim TCP implementation was to be used for transport, a node failure would result in a TCP slow start of the sources affected by the failure, lowering the throughput of a network immediately following the failure. Furthermore, during this post-failure period, there will be a time span where the traffic affected by the failure is being rerouted, and because of the propagation time, there will be a delay before the effect of the failure may be seen in the link-load of the links downstream of the failure. If the measurements are based on this post-failure period, they may not provide an accurate picture of the actual influence of the state change on the network.

In the simulated experiments with IPRT, the main focus is the duration between failure and subsequent re-convergence. It has been shown that the IP re-convergence may be reduced to sub-second response time [7]. However, an important motivation for implementing IP fast recover mechanisms, is to provide the re-convergence process with more time to properly respond to the failure, and thus, the duration of the recovery procedure should be considered a larger interval.

To ensure that the results obtained from the simulation are correct, the duration between failure and re-convergence should be run more than once. There are several ways of achieving this, but a commonly used approach is to use a simulation technique named sectioning [33]. This method allows a single scenario to count as a series of independent runs, by running the same scenario for a magnitude longer than the original duration the simulation was intended to run for. This is done by splitting the simulation into a series of discrete periods where measurements are performed, and furthermore desist from measuring for a short interval between the periods, in order to make the discrete measured periods independent. By using this technique, less total time is required for the transient period, i.e., the period from system startup until a stable state is reached, while it is still possible to provide a range of independent measurements.

Chapter 8

Experiments

In this chapter, the experiments will be described and the results obtained from the simulated scenarios will be shown, followed by a discussion of the results.

The main goal of the experiments is to show that the IPRT mechanism may be used for recovery in conventional IP networks. Thus, an important part of the experiments is to show that no packet affected by a single node failure, excluding the traffic sent to the failed node, is lost during the duration of the failure. Furthermore, a sub-goal of the experiments is to show to show how the IPRT method performs in terms of recovery and how the method affects the network-wide load.

8.1 Experiment description

In this section, the experiments and their aims will be presented. Furthermore, in these experiments, the IPRT specific parameters that will be varied include what cost has been used, when the recovery trees have been generated, the use of QoS information in the Qbit tables, and the use of deflection in the forwarding procedure:

- Either the trees may be constructed using a flat cost on all links or use the link-cost specified obtained from the topology. By using a the cost obtained from the topologies it may be possible to see if the algorithm used in the IPRT tree generator is able to respond to changes in link-cost.

- Qbit may be used for optimization, i.e. selecting the best outgoing interface, and guaranteeing a correct forwarding procedure using local recovery, i.e. selecting a valid recovery path. When referring to Qbit in this chapter it is the QoS responsibilities that are referred. Thus, the Qbit tables are always populated with the information needed to ensure correct forwarding, but does not include the QoS information unless explicit stated otherwise.

- The Qbit information responsible for provide a guaranteed successful local recovery procedure is computed with the probabilistic approach (see section 5.3.1), making it necessary to support deflection, i.e. require the forwarding procedure to move recovered traffic from the red to the blue path is a failure is encountered twice. By turning off deflection, it may prove that the recovery FIBs represent a valid configuration for forwarding without deflection, i.e. a set of trees that could have been computed from the exact procedure to populate the Qbit information. Thus, enable the experiments to include an assertion on the effects of using deflection in the forwarding procedure.

8.1.1 Coverage and model verification

The degree of coverage the IPRT may provide is the most important experiment in this thesis. The goal of the test is to show the amount of traffic affected by a single failure that the recovery method may successfully deliver to the destinations. Furthermore, because it is proven in theory that the coverage should be 100%, this test is able to verify the simulation model. A secondary goal of this test is to investigate the degree of coverage achieved when deflection is turned off in the forwarding procedure.

To verify the coverage of IPRT, the test needs to be able to determine the percentage of destinations that can be recovered during a single failure, and furthermore, ensure that all potential failure situations are covered. Furthermore, the test needs to be able to verify a successful recovery procedure, or discover if a recovery procedure has failed.

In order to identify the nodes that may be reached during a single failure, the test does only need to verify the amount of nodes that the neighbors of the failed node may successfully reach. This is because IPRT provide local recovery, and thus, only the neighbors of a failed node may initiate a recovery procedure. A possible optimization could be to verify the destinations where the next-hop would have been the failed node, and thus, reduce the number of destinations that needs to be checked.

In order to verify every failure situation the method for verifying the coverage of a single failure needs to be applied such that each node in the network is failed once. Because the recovery path to a destination is determined by the pair of redundant trees of which the destination is the root node of, the path of a recovered packet depends on the destination, the geographical position of the node initiating the recovery, and which of the redundant trees are failure-free.

This approach ensures that all possible local-recovery scenarios for a specific failure situation are tested. Furthermore, to ensure that the time between to failure situations is of such a degree that there is no chance for a packet to be affected by two different failures before it reaches the destination, the coverage provided during a single node-failure will be tested in separate simulation scenarios.

The test must be able to verify that the lack of coverage is successfully determined. In order to provide a successful recovery IPRT needs to fulfill two criterions, which forms the foundation for the coverage verification:

- The initial recovery needs to be successfully performed i.e., the forwarding procedure needs to be able to identify the packets affected by the failure and choose which recovery tree to use.

- All nodes in the network need to be able to forward the recovered packets along the specified tree.

One possible way of verifying the amount of successful recovery procedures is to verify that all sent packets are actually delivered, and furthermore identify when packets are dropped. The coverage could be determined by only verifying the successful delivery of packets. However, by identifying the dropped packets it is possible determine the reason for a failed delivery in more detail. In order to ensure that a packet is delivered it is possible to use the GetHop utility, which will report if a destination has not been reached. Furthermore, it is possible to instruct the dispatchers at each node to report when packets are dropped and a detailed description on the reason for the drop. Thus, if a packet drop occurs this will be recorded in the log file of the simulation.

Thus, the coverage will be verified based on a multiple simulation runs for each listed scenario where results are drawn from two separate measurement methods. In each run, a single node is failed, and furthermore, one of the neighbors of the failed node tries to communicate with all other nodes except itself and the failed node. The test iterates such that for each simulated node-failure, all neighbor nodes initiate the communication once. The node initiating the communication keeps track of which nodes it was able to reach before and after the failure. These results are compared against each other. Furthermore, the dispatcher at each node is configured to log any packet drop.

8.1.2 Path-length

The scenario needed to determine path-length is very similar to the coverage and model verification test. However, it is necessary to include measurements from more nodes in each failure-situation. Furthermore, this test will include measurements from failure-free, re-converged and RRL-recovery operation to compare to IPRT performance.

The test needs to verify how IPRT influence the path length between a source and destination. It should also be able to ensure that all situations where failure may influence a path are covered by the test. The test does not need to verify the number of paths that may not be recovered, as this will be apparent from the coverage test.

- When measuring path-length it is the entire path that needs to be measured. This is necessary to be able to determine the differences between results obtained from the normal, recovery and re-converged scenarios, as the intermediate nodes found in a path between a specific source and destination may have changed between the scenarios.

- The recovery path to a destination is partly determined by the geographical position of the node initiating the recovery, and which of the redundant trees are failure-free. Thus, even if the same destination node is used, a failure on different intermediate nodes in the path between a source and destination may yield different increases in path-length. This makes it important to, for each single-node failure, measure the influence the failure have on the paths where the failed node would be a part of the path during failure-free operation.

As in the coverage test, the path-length test utilizes the HopCount application. This application is run at each node. At the source node, the application is used to probe the network, sending a single HopCount packet to a specified set of nodes. At the each receiving nodes, the application records the number of hops and if the packet has been recovered. Subsequently it include this information in a reply sent to the source node.

To obtain the results the simulation is repeated such that each operational node initiate the communication once in each node-failure scenario. The network is probed once during failure free operation to be able to give a reference point to the path lengths in a failure free environment. Subsequently a single node is failed, and the application is rerun. The time between the first and the second run of the packets is set large enough to guarantee that the failure does not influence the results obtained at each stage. Furthermore, a list is printed to the log, specifying the hop-count of paths affected by a failure. This enables the collection of data on the path-length before and after the failure in addition to specifying the node initiating the communication and the node receiving the packets. Subsequently, the pairs of sender and receivers are recorded and used as input to the HopCount application in a re-converged simulation scenario. However, this step is only conducted if the first run reveals paths influenced by the failure.

Care must be taken to avoid producing erroneous results. A potential source to errors originates from the time between the measurements and the failed node. Thus, care must be exercises to ensure that the failure and the measurements do not interfere with each other. By using a high link capacity and a substantial amount of time between the checkpoints in the simulation this is guaranteed. I.e. the time and the link capacity and propagation time must exceed the time to live, such that no traffic is present in the network at the time the failure is simulated.

8.1.3 Throughput and load distribution

The throughput test aims to reveal the capabilities of IPRT to spread load in the network. Furthermore, it will be used to determine how deflection in the forward procedure influences the load in the network.

Because every destination node has a different set of redundant trees, and the redundant trees are generated through means of short cycle and arches, it is anticipated that IPRT may be able to spread the load of a recovered traffic in a way that may resemble the re-converged paths. To verify this assumption the results of the throughput test will be shown on link-level, giving detailed description on the load observed at each link.

When using the deflection mechanisms to assure that all traffic is capable of being recovered it is anticipated that the method may generate a higher load than an approach where accurate Qbit was pre-computed for this purpose. Furthermore, it is anticipated that about half of the traffic bound for a failed destination will be deflected. This is because the forward correction Qbit is not set for the destination and when traffic from all other nodes is present the overall traffic recovered to the lower ID is expected to account for an average of half the traffic bound for the failed node.

To gather the results each link is fitted with a traffic monitor as implemented in the J-sim simulator. It is able to print the total throughput of a link every second, resetting the counters at each log point. However, when no load is present on a link it is not able to log this as a zero load. This could lead to a problem; if a link is operational but not used to transport traffic, its load of zero would not be registered as part of the measurements. This is solved by generating a logfile resembling the result that would have been produced by the traffic monitor had it been able to log correctly. The files are only generated for links that were operational but not used to transport packets by using the scenario logfile to separate failed and operational links.

To obtain the results each experiment is repeated once for every possible node-failure. In each run, the traffic generators are installed and a single node is failed at simulation startup. The traffic monitor will start to measure after the scenario has been running for 5 seconds, and the entire scenario ends at 1000 seconds. Because UDP is used to transport the traffic, this duration should be sufficient for the scenario to stabilize the load at each link. Furthermore, when analyzing the results, it is assumed that the time between failure-detection and re-convergence is 10 seconds and 1 second is used to separate the sections.

When simulating the re-converged scenario, the traffic generators generating traffic bound for the failed node are not removed. This enables a direct comparison between the re-converged scenario and the IPRT scenario in terms of the amount of traffic introduced in the network. However, this makes the re-converged scenario less realistic as it normally would not have any route to the failed node, and thus, in a real scenario, no traffic bound for this node would be introduced in the network.

To generate the load a series of Poisson traffic generators, where each load is scaled according to the traffic matrix of the topology, will be used. All sources combined will be configured to generate a total load of 10 Mbps of traffic regardless of the topology. By introducing a total of 10 Mbps, each source generates an average of 111 Kbps if 10 nodes are present in the network or 26 Kbps when 20 nodes are used. Furthermore, the source will generate packets with a constant size of 480 bit in the payload, thus producing packets of 500 bits when the IP header is included.

To guarantee that no traffic is lost due to small link-capacity the links are configured to handle a load of 100 Mbps.

8.1.4 Topologies

The topologies used to evaluate IPRT behavior is Abilene (8.3), Geant (8.1), Uninett (8.4) and Cost239 (8.2). They provide a variety in both connectivity and layout and average node degree, and are well-known topologies often used in research and evaluation. Because of the variety in the network, properties it is expected that the results obtained from the different topologies will vary. E.g. Abilene provide a topology that is expected to produce longer recovery paths than Cost239.

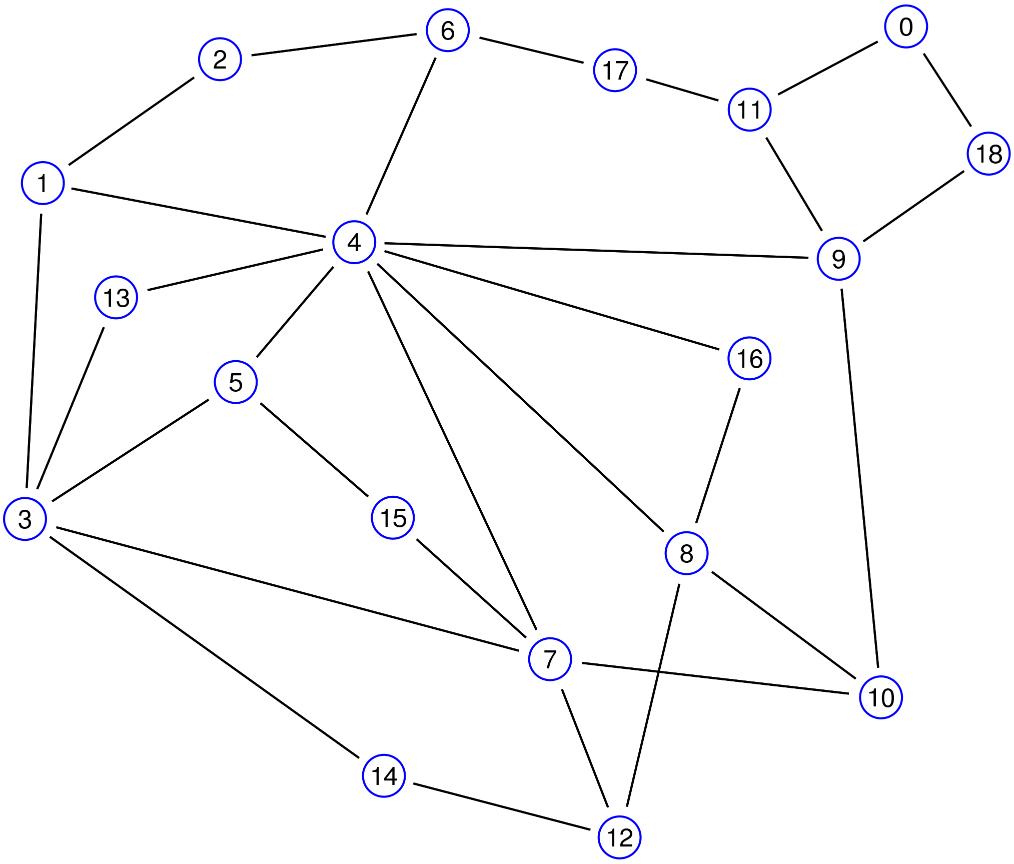

Figure 8.1: The Geant topology

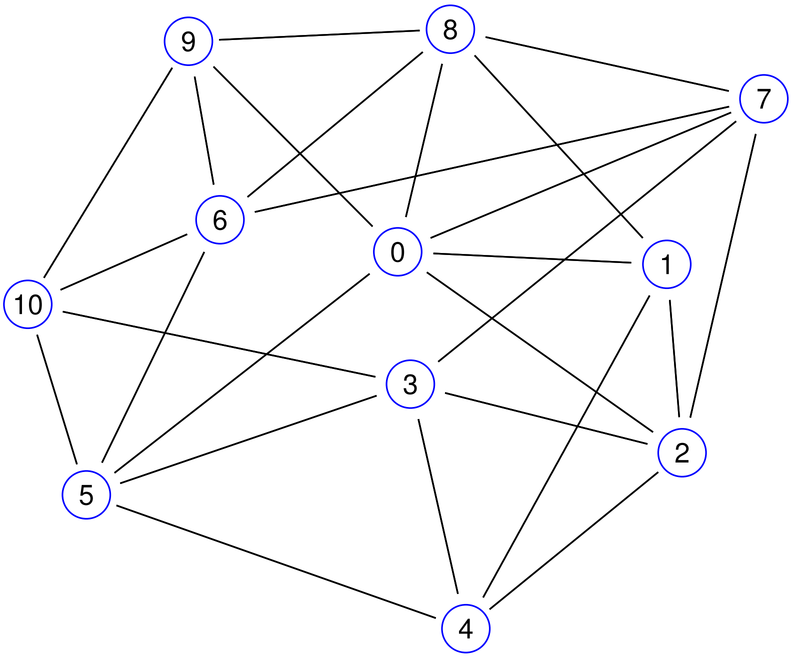

Figure 8.2: The Cost239 topology

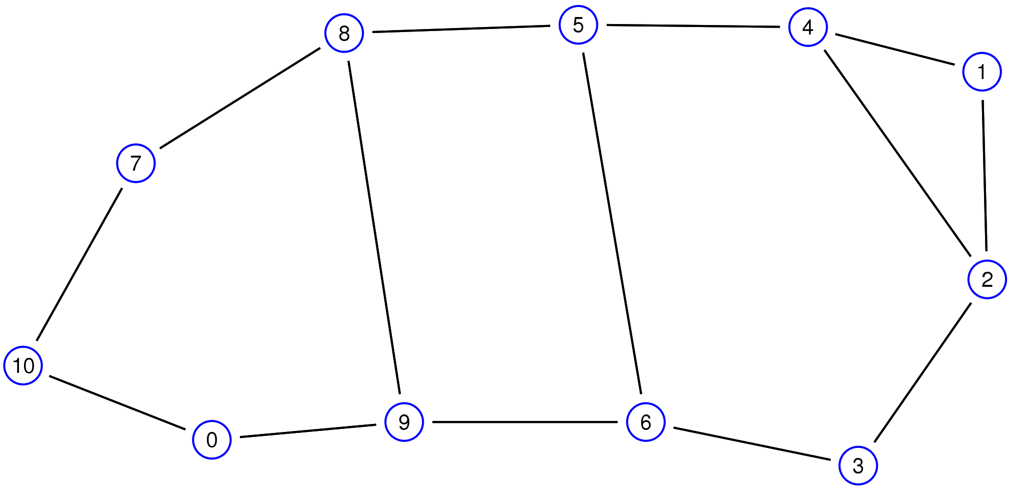

Figure 8.3: The Abilene topology

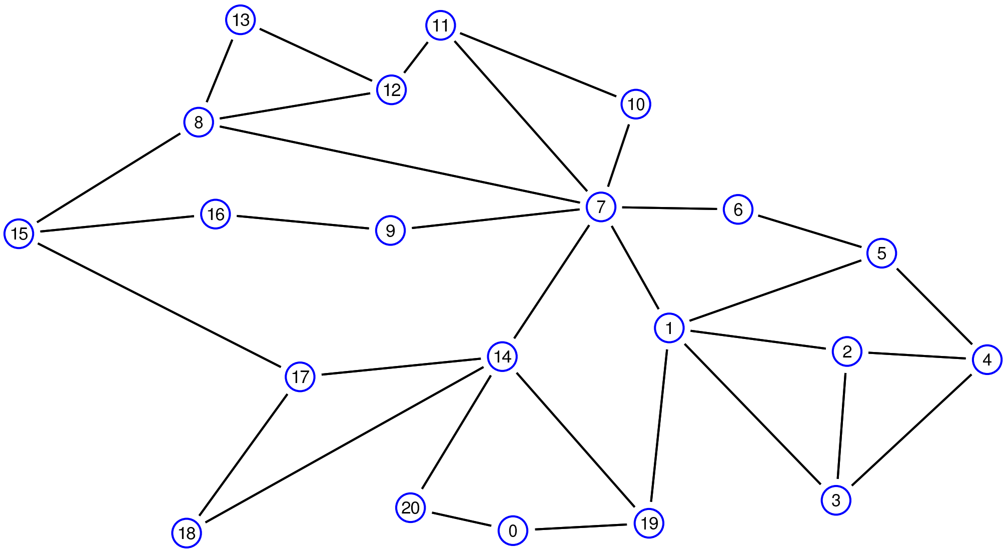

Figure 8.4: The Uninett topology

8.2 Results

This section contains the results from the experiments described. All discussion will be conducted in section 8.3.

The RRL layers was generated by an layer generator obtained at SIMULA. The layer generator was an old implementation and may not represent the actual qualities of RRL or Multiple Routing Configurations (MRC) [34], which is a further development of RRL. The number of layers used in the simulations is shown in table 8.1.

Network	Layers used
Abilene	5
Cost239	2
Geant	5
Uninett	7

Table 8.1: The number of recovery layers used by RRL in the simulations

8.2.1 Coverage

Table 8.2 and 8.3 shows the coverage of IPRT and RRL with the different parameters.

Network	Link-cost	Qbit	Coverage
Abilene	Yes	Yes	100%
		No	100%
	No	Yes	100%
		No	100%
Cost 239	No	Yes	100%
		No	100%
Geant	Yes	Yes	100%
		No	100%
	No	Yes	100%
		No	100%
Uninett	Yes	Yes	100%
		No	100%
	No	Yes	100%
		No	100%

Table 8.2: IPRT recovery coverage

The tests was repeated once with forward deflection, i.e. the ability to move traffic from the red to the blue path, turned off. The coverage obtained from this test was the same as for the one with deflection turned on, i.e. full coverage. These results validates

Network	Link-cost	Coverage
Abilene	no	100%
Cost239	no	100%
Geant	no	100%
Uninett	no	100%

Table 8.3: RRL recovery coverage

further testing with deflection turned off in the forwarding decisions even though the trees were not initially generated to support this usage. This may be done because it has been shown that the recovery FIBs used represent valid configurations for forwarding without deflection. I.e. it has been shown in the coverage section that they provide with 100 % coverage without deflection turned on.

8.2.2 Path-length

The graphs found in Figure 8.5, 8.6, 8.7, 8.8, 8.9, 8.10 and 8.11 show the distribution of path-lengths affected by a failure. The path-length is a measurement on the amount of links a packet must traverse to get from the source to the destination. In IPRT the recovery path is determined by the redundant trees of which the destination is the root node of, the geographical position of the node initiating the recovery and which of the redundant trees are available for this node to use. Thus, A→B and B→A are disjunct paths and treated as such - even though they might follow the same links. Paths unaffected by failure are not included in the graphs.

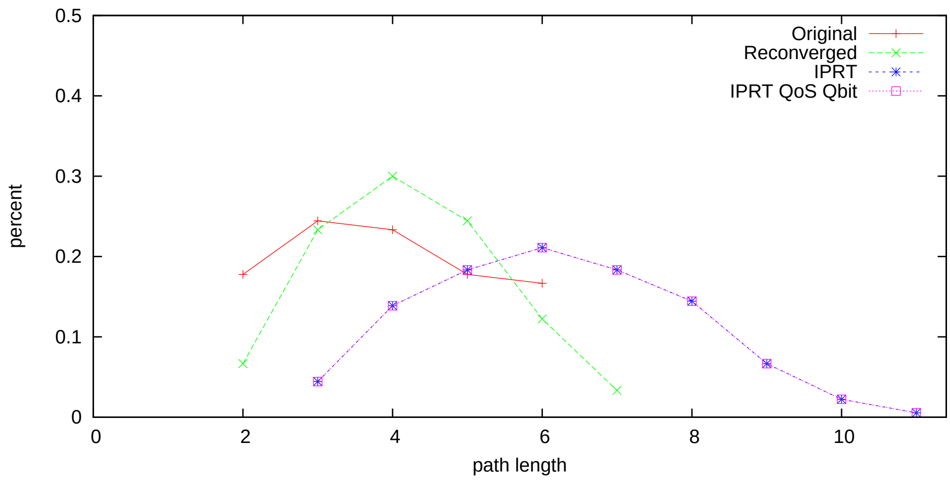

Figure 8.5: Abilene, with link-cost enabled

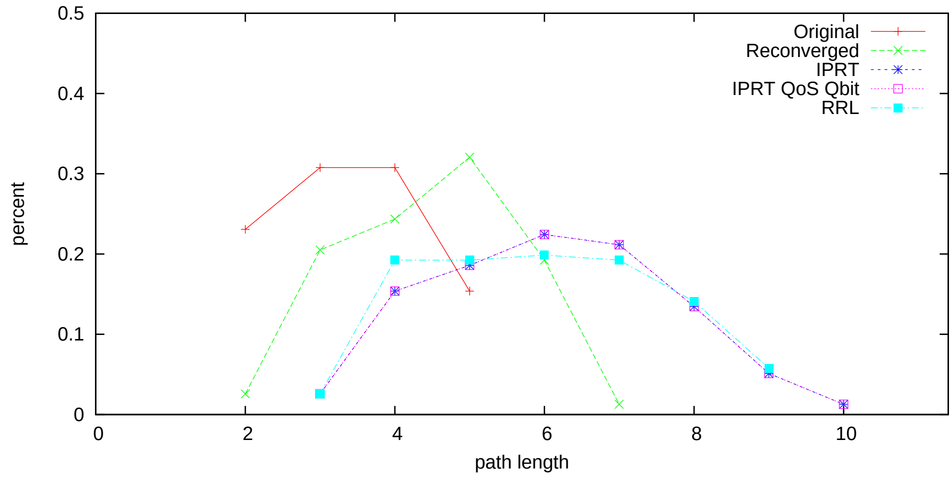

Figure 8.6: Abilene, with flat link-cost

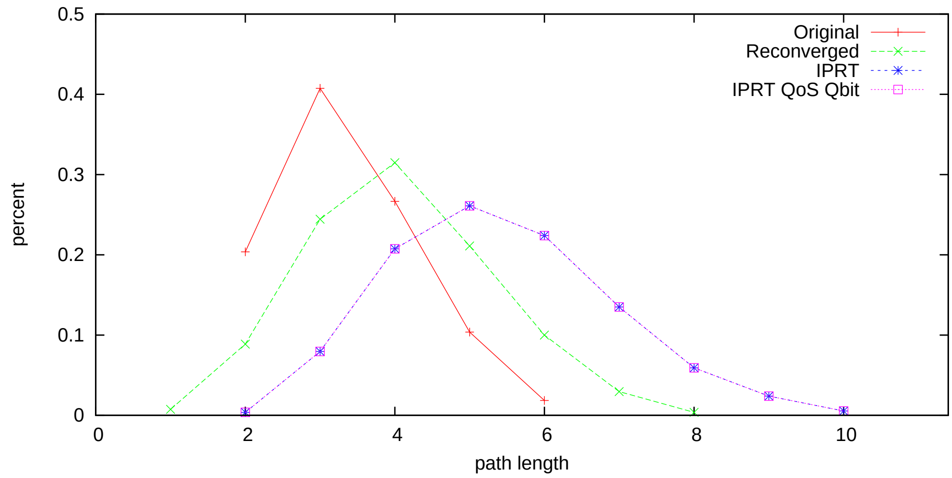

Figure 8.7: Geant, with link-cost enabled

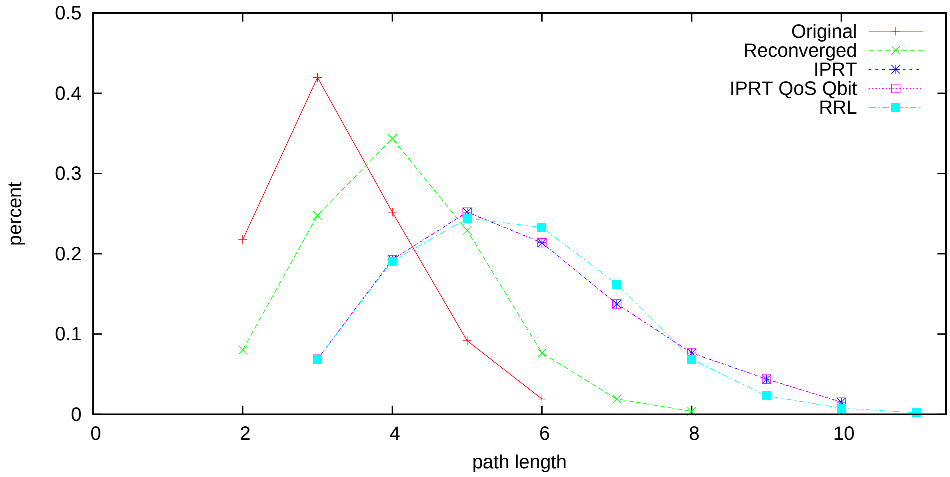

Figure 8.8: Geant, with flat link-cost

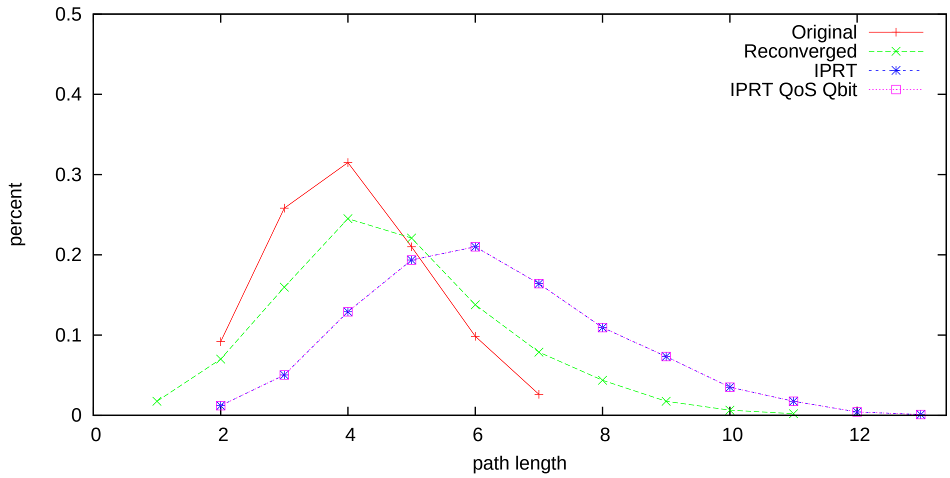

Figure 8.9: Uninett, with link-cost enabled

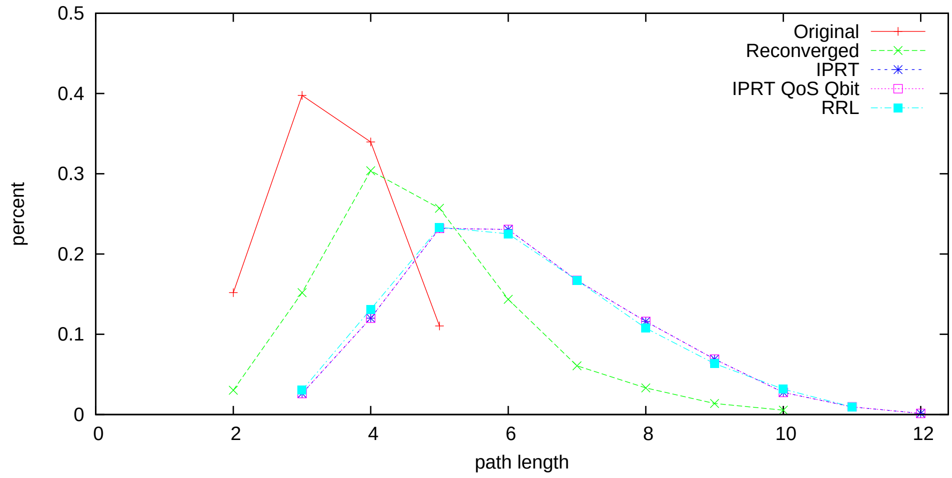

Figure 8.10: Uninett, with flat link-cost

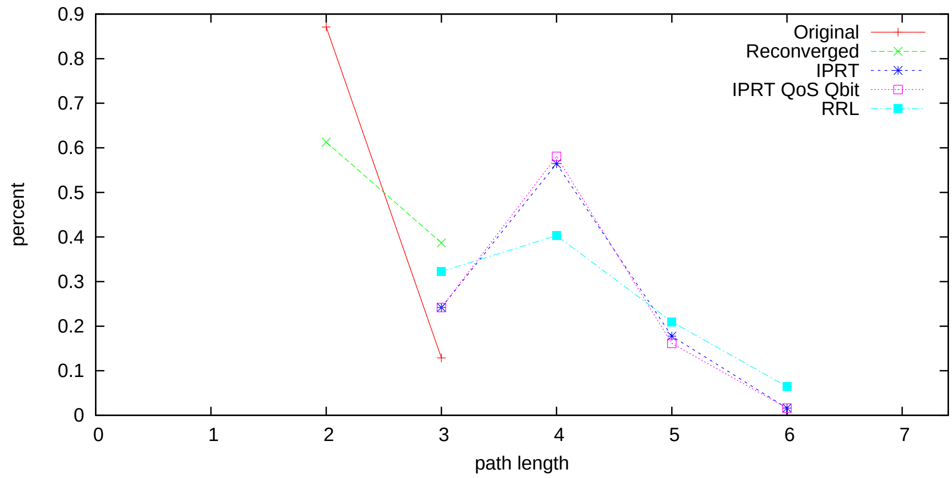

Figure 8.11: Cost239, with flat link-cost

8.2.3 Load distribution

The graphs found in Figure 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18 and 8.19 show the median throughput including the 97.5 and 2.5 percentile measured at each link. In the tests UDP was used as a transport protocol making the measured values mimic a worst-case scenario. In addition the traffic between each source and destination node was weighted according to topology specific traffic matrixes.

The results obtained from running the simulations with QoS Qbit are omitted as they did not differ from the results obtained without QoS Qbit turned on.

Note that the re-converged results include traffic from sources sending traffic to failed nodes. This traffic is dropped at the last-hop before it reach the failed nodes.

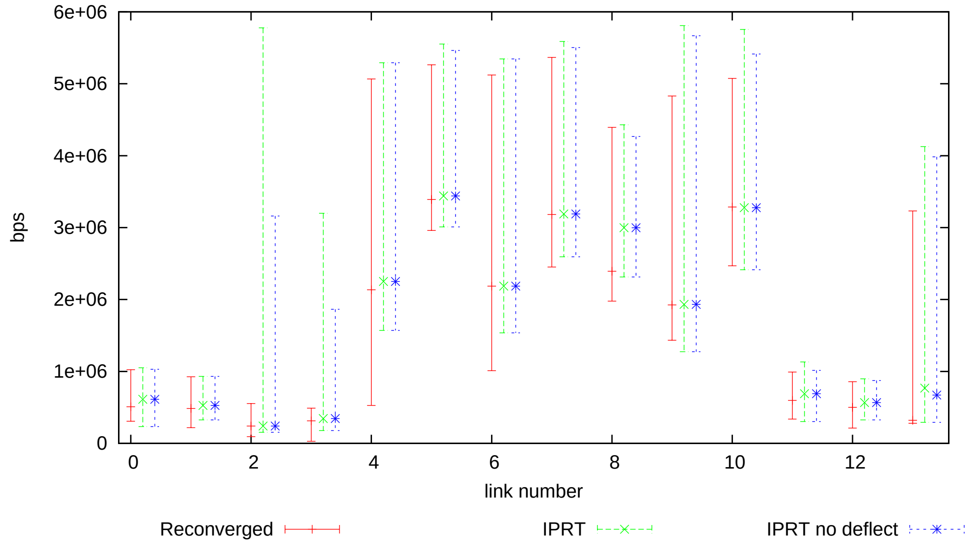

Figure 8.12: Abilene, with flat link-cost

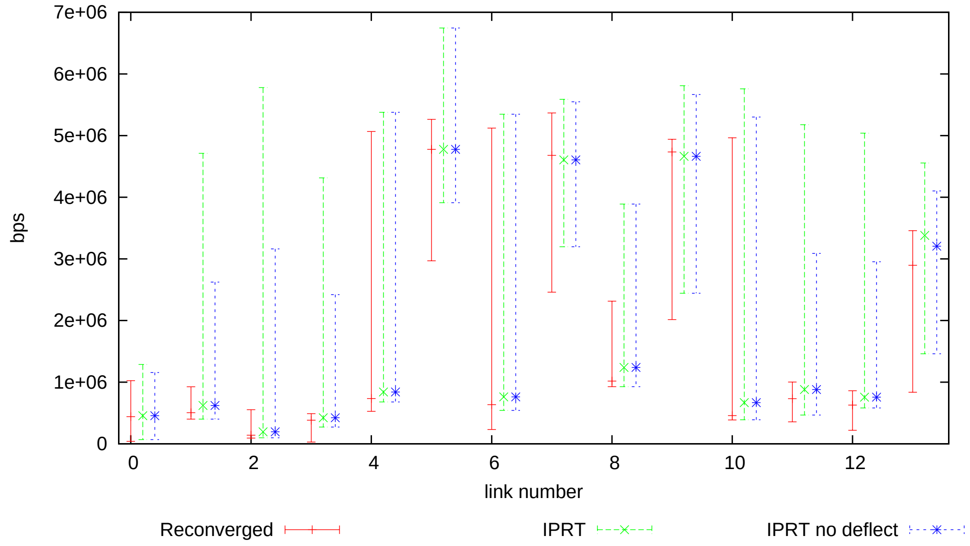

Figure 8.13: Abilene, with link-cost enabled

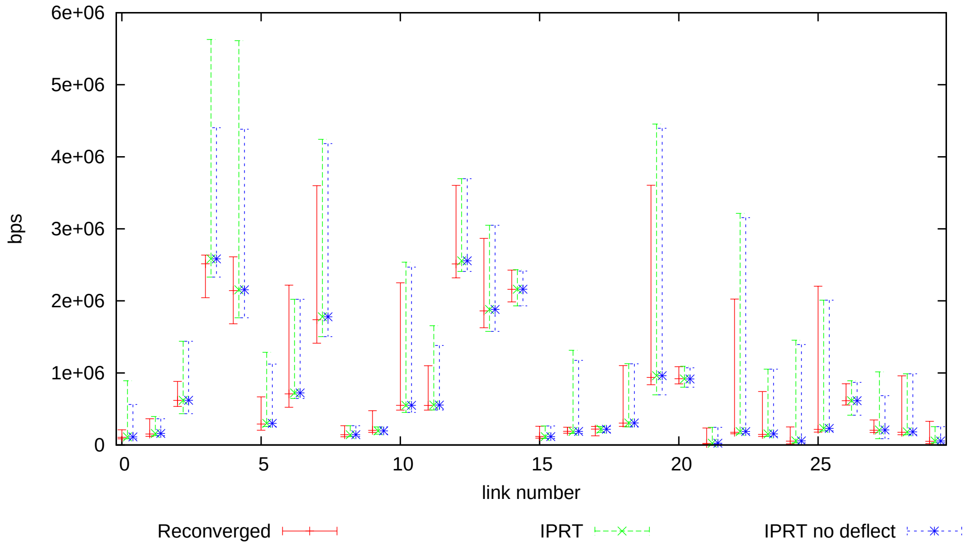

Figure 8.14: Geant, with flat link-cost

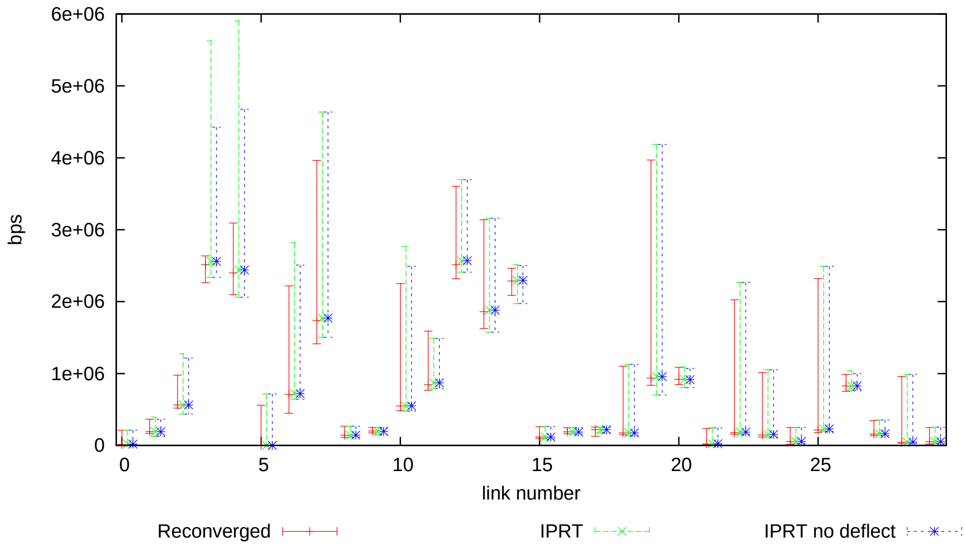

Figure 8.15: Geant, with link-cost enabled

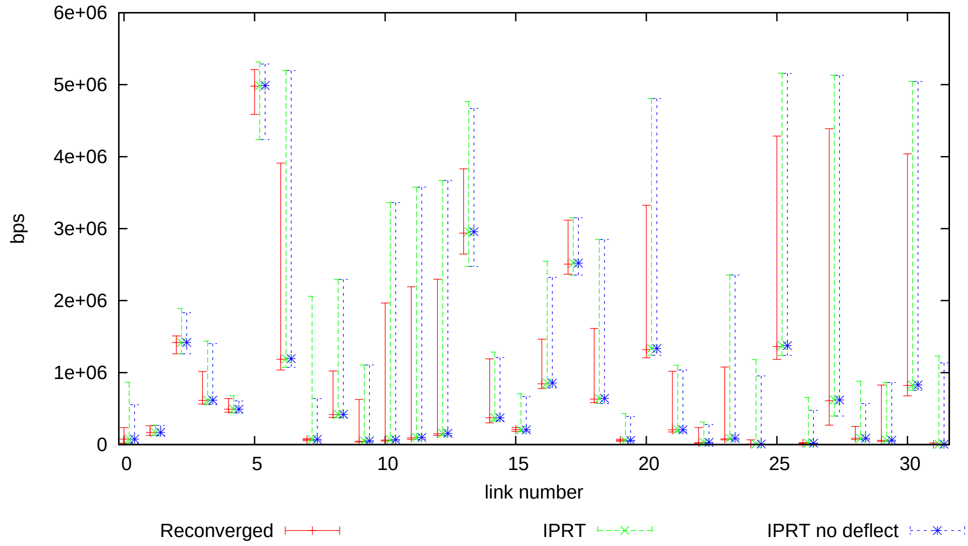

Figure 8.16: Uninett, with flat link-cost

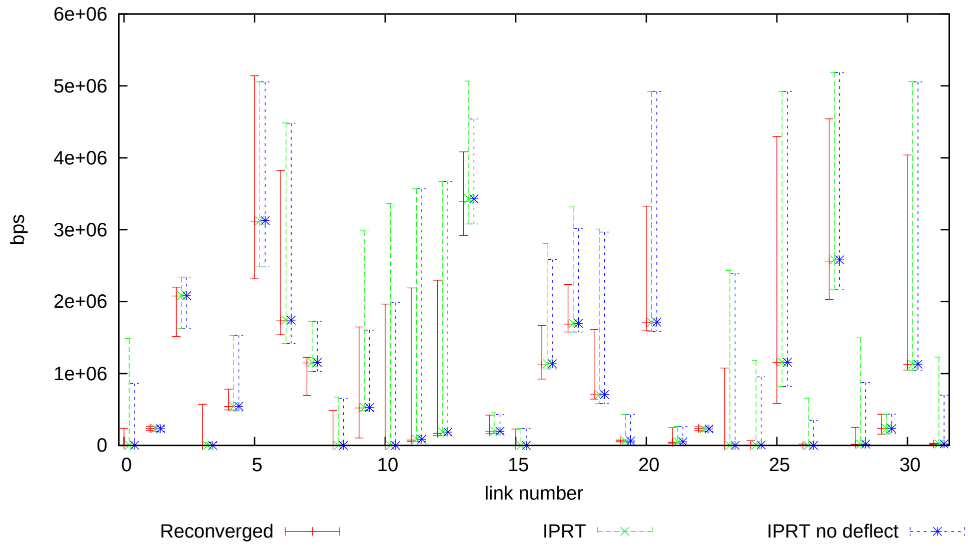

Figure 8.17: Uninett, with link-cost enabled

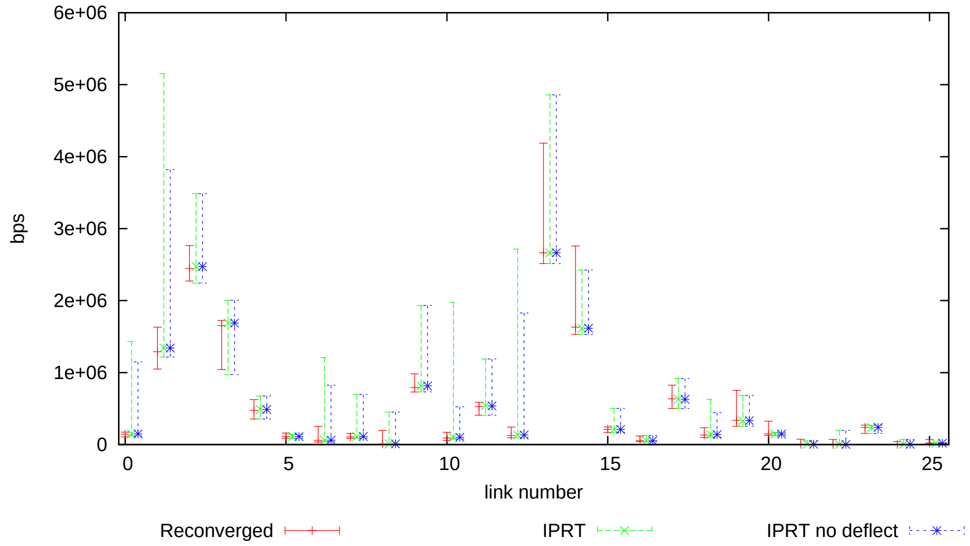

Figure 8.18: Cost239, with flat link-cost

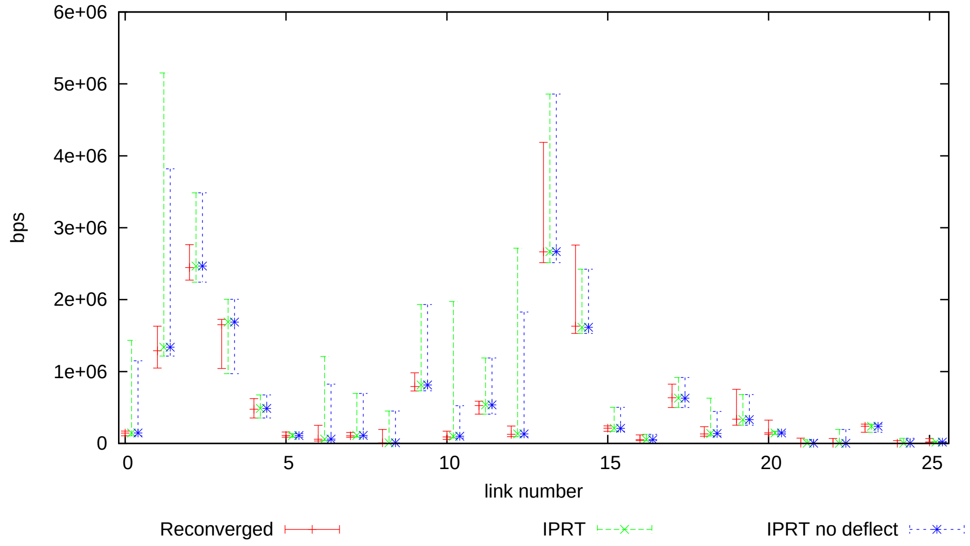

Figure 8.19: Cost239, with flat link-cost and QoS Qbit

8.3 Discussion

8.3.1 Coverage

Of the results obtained from the simulation runs, the coverage results show the most important feature of IPRT. Both IPRT and RRL are proven capable of delivering 100 % coverage, which is an excellent result considering this may be the single most important property to legitimate the use of extra resources in a network in order to provide IP fast reroute recovery. However, IPRT is able to provide this coverage using a small and constant amount of additional state information. In these tests RRL needed between two and seven recovery FIBs (see fig. 8.1) while IPRT used a constant amount of two additional FIBs.

Because it has been proven in theory that the IPRT method should be able to provide 100 % coverage, the results seems to verify that the IPRT simulator model has been implemented correctly. Since node failure has been used as the only failure-model in this thesis, it has not been shown in the simulator that the IPRT solve the "last-hop problem". However, following directly from the tree-generation process it is proven in theory that IPRT does solve this problem without the need of any additional functionality in the IPRT model.

8.3.2 Path-length

The results obtained from the path-length tests shows that IPRT is capable of delivering short recovery paths. This is an important property because the number of links used to forward the packets influence directly on the total amount of load generated in a network. When the number of links used to forward a packet increase the total load increase as the packet generate load on all links it traverse.

The graphs show that all normal traffic affected by a failure has a path-length of two or more. This is because the measured paths show the path-length of traffic affected by failure that is bound for an operational node. Because the destination needs to be downstream of the failure to be affected by the failure, the path-length needs to be at least two hops. In some circumstances, when link cost has been enabled, the path-length after a re-convergence becomes only one hop. This is because the re-converged traffic then follows a path with higher cost.

Furthermore, it may be observed that the routes obtained after a re-convergence are shorter than the recovery paths obtained from both RRL and IPRT. This may be explained from the fact that a re-convergence may be compared to the best result that a global recovery could produce. Furthermore, both RRL and IPRT needs to implement a local recovery procedure resulting in paths that may need to backtrack upstream of the failure. In addition, both RRL and IPRT are using restricted sub-topologies to route packets affected by the failure as a result of the layer creation or tree creation needed to guarantee full coverage.

In these tests, IPRT and RRL provide recovery paths that are comparable in terms of recovery path-length distribution. However, the test result should only be considered to give indications on the relative performance of IPRT. RRL was not tweaked to obtain good results, and the number of layers was set to the minimum number of layers that the layer generator could provide for each topology. Furthermore, RRL could use more layers to provide with shorter recovery paths.

8.3.3 Load distribution

In this section, IPRTs general ability to distribute load is analyzed. In addition, it is shown that the use of deflection, i.e. the ability to move a IPRT packet with a RED FIB ID to a BLUE FIB ID, does increase the load in some worst-case scenarios and should be avoided. The QoS specific properties, i.e. the ability to respond to link-cost and QoS Qbit, are discussed in the next section.

The measurements obtained from the post-failure load-distribution experiment give information on several aspects of the IPRT recovery procedure. The most important observations are listed and discussed below.

- IPRT is generally capable to follow the load distribution obtained at re-convergence.
- The median load is generally higher when IPRT is used (see fig. 8.4).
- In some failure situations IPRT increase the maximum load on specific links significantly.

In the measurements obtained from simulating the re-converged state in the network, the failed nodes have traffic bound for them. Thus, the amount of traffic introduced in the network does not differ between the IPRT recovery scenarios and the re-converged scenarios.

In the scenarios where IPRT recovery is used, only the traffic affected by the failure is rerouted by the IPRT recovery procedure. In most failure situations the traffic will be unaffected by the failure, allowing the traffic to be forwarded according to the FIBs obtained from the normal routing procedure. When re-convergence is used, the routes will be very similar to the normal routing table, as only the shortest paths affected by the simulated failure are altered. Thus it may be expected that most of the traffic introduced in the network will follow the same links and thus to a great extent allow traffic to follow the same paths in both failure-free and failed situations. This is also valid when cost is used as the link-cost is not altered, except for when the re-convergence is simulated, and then only links affected by the failure are altered.

The redundant trees are generated to provide with recovery paths that are close to the shortest path available. Thus, it is likely, that the recovery paths follow the re-converged shortest path in a close manner. However, because the IPRT redundant tree generator needs to follow strict rules when constructing the trees, it is expected that there will, to a small extent, be some deviation from the shortest path at most times.

The paths found in the r/bTables are based on each individual nodes pair of redundant trees. This should enable IPRT to provide good spread of traffic downstream of a failure. However, in this implementation it is partly governed by chance. I.e. if there is a link with low cost downstream of the failure and link-cost is used, it is a higher probability that the link will be included in all trees. However, because of the redundant properties of the red and blue paths the link will probably not be included in all the available recovery paths downstream of the failure.

Network	Link-cost	Deflect	avg total inc	min link inc	max link inc
Abilene	Yes	Yes	6.65%	-1.53%	45.89%
		No	5.89%	-1.53%	45.89%
	No	Yes	7.27%	-0.29%	139.15%
		No	6.82%	-0.29%	109.48%
Cost 239	No	Yes	1.15%	-4.00%	11.50%
		No	1.15%	-4.00%	11.50%
Geant	Yes	Yes	1.42%	-2.39%	15.00% *
		No	1.42%	-2.39%	15.00% *
	No	Yes	1.39%	-3.55%	11.11%
		No	1.39%	-3.55%	11.11%
Uninett	Yes	Yes	0.70%	-3.70%	100.00% *
		No	0.68%	-3.70%	100.00% *
	No	Yes	0.70%	-1.04%	12.28%
		No	0.70%	-1.04%	12.28%

Table 8.4: IPRT median throughput compared to re-converged values on total load and link-level load (* does not include measurements where re-converged link-load is zero)

The general increase in the median load when IPRT is used, is caused by the fact that IPRT is using a local recovery strategy. When using a local recovery strategy the length of the recovery paths usually contains more hops than if a global recovery strategy is used. It is apparent from the path-length experiments that this is valid for IPRT. Because more links are used to forward the traffic a single packet that has been subject of recovery, is bound to generate more load as more links are used to transport the packet to its destination.

Furthermore, some load-spikes may be observed in the results from the simulations. It may be observed that it is during only a few of the failure situations that a load-spike is present because of the deviation between the median value and the high percentile. An example on such spikes may be seen in Figure 8.12 and 8.13 on link 2 (connecting node 1 and 2) and link 3 (connecting node 3 and 4), similar examples are also present in all the other topologies. There are several reasons to why the spikes come to presence but generally they are a result of one or more of the two circumstances listed below.

1. When IPRT is enabled, all traffic affected by failure is treated uniformly and thus traffic bound for failed nodes are subject for the recovery procedures of IPRT. Furthermore, the initial cycle used when generating the redundant trees follows a shortest path starting and ending in the root node. This approach makes it likely that the recovered traffic destined to the failed node to be contained in a near geographical proximity of the failed node and thus creating a "hot-spot" of traffic.

2. When recovered traffic encounters a failure for the second time, i.e. recovered traffic with the failed node as destination, the traffic following a red path is tried recovered a second time. In the simulations used here the only traffic that is subject for this deflection is the traffic bound for the failed node, thus increasing the "hot-spot" effect. Generally, the amount of traffic that is deflected equal to half the traffic bound for the failed destination. This may be observed from the measurements performed with deflection turned off.

The observed effect from these two circumstances are further increased by the re-converged scenarios traffic bound for the failed node is not tried recovered. In addition, the traffic matrix used to vary the load generated by the traffic generators contain variations in the amount of traffic bound for a node and a few of the traffic generators generates an above average heavy load.

To verify the findings the simulation of the Abilene topology, where the example load-spikes were shown, was repeated and the results may be seen in figure 8.20 and 8.21. However, in these experiments all traffic bound for a failed node was pruned from the network. I.e, the traffic generators sending traffic to the failed destination were not installed. Thus, the results obtained from these simulations contains less traffic than the previous conducted experiments, and may not be compared directly to the previous shown graphs. Furthermore, the results does not show an accurate picture of how IPRT would perform as the traffic should be tried recovered. I.e., the upstream node should presume that all failures are link-failures, as this approach is the only approach that may guarantee 100 % coverage of all failures, i.e. the "last-hop problem". However, the results obtained here seem to confirm the claim that the recovery of packets bound for the failed node is responsible for the load-spikes.

The observed increase in load makes it clear that the probabilistic approach to populate the Qbit tables - in order to ensure local recovery, is a poor choice. The deflection result in a increase in link-load when the destination node has failed. Thus, the better choice between the probabilistic and exact approach, is the latter approach - even though this choice requires a small increase in computational requirements.

8.3.4 QoS

This section aims to discuss IPRTs ability to support QoS routing, i.e. the observed effect of QoS Qbit and IPRTs ability to respond to the use of link-cost and how this influence the load-distribution.

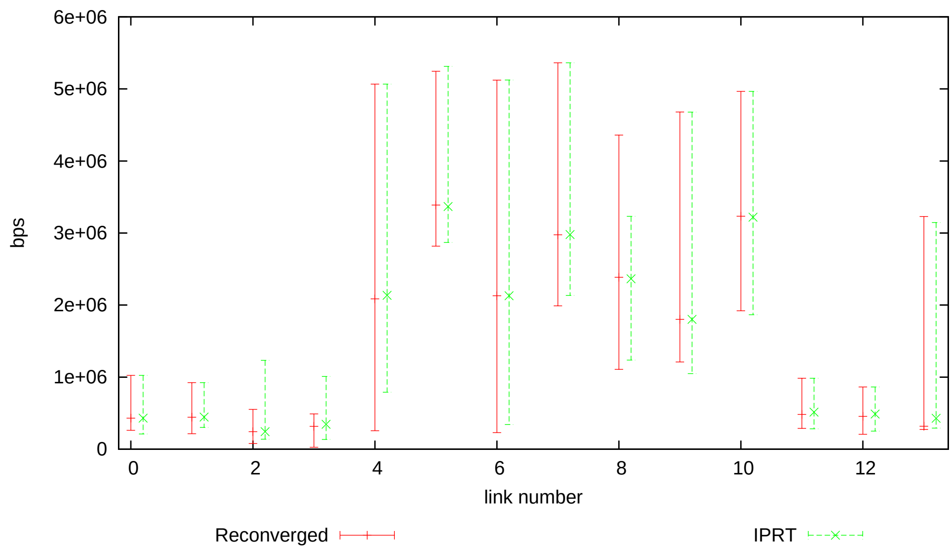

Figure 8.20: Abilene, no traffic to or from failed node with flat cost on links

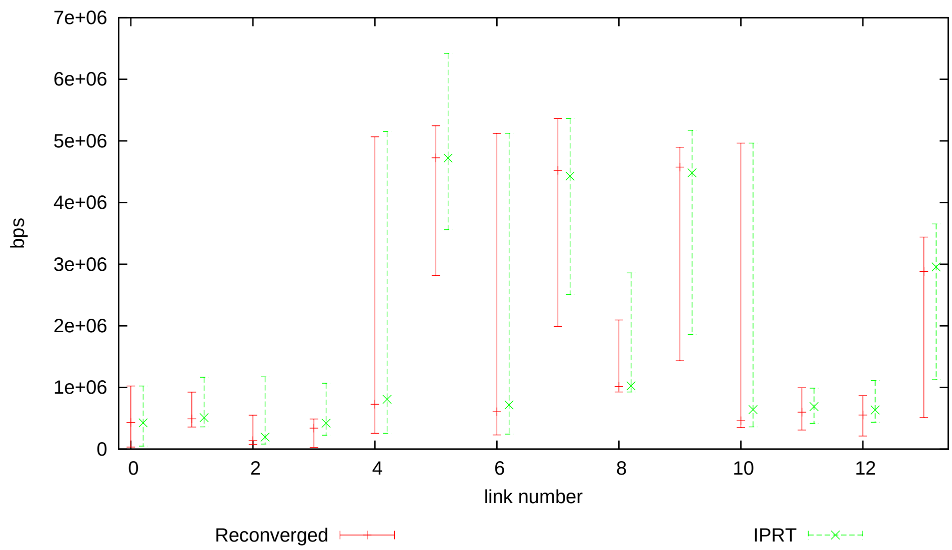

Figure 8.21: Abilene, no traffic to or from failed node with cost on links

Qbit

In the IPRT design chapter it was shown that Qbit could be populated with QoS information, in order to enable the IPRT forwarding procedure to select the shorter of two recovery paths when available. The results show that the QoS Qbits did not have any influence on the results gathered from the simulations, with the exception of the cost239 network. In cost239, it did reduce the average length of the recovery path, and furthermore, it was shown that the reduction in recovery path-length did not influence the worst-case link-load in the network (see Figure 8.18 and 8.19).

The reason for the low benefit obtained from using QoS Qbit in terms of better performance may be a result of several circumstances.

- Because of the requirement of a degree of three or more on the number of links only a subset of the nodes in the network may potentially benefit from QoS Qbit.
- The redundant trees are generated in such a way that the cycle and arches are tried kept to a minimum length. Thus a greater amount of links are used and makes it more likely that one of the paths follows the path used for routing in normal operation. A good example on this effect is found in the simulation with the Geant topology where the log files showed that there were no circumstances where both recovery paths were available during a failure.
- The topology requirements that needs to be fulfilled to provide QoS Qbit is very similar to the scenarios where Qbit forward correction needs to be utilized. Furthermore, because deflection was used, the basis for setting and locking a Qbit entry were based on whether or not it was potentially needed and thus this method may include false positives.
- When the Qbit information was calculated equal cost paths was tagged with a "third-color" and always interpreted as a free selection resulting in the red path being chosen.

The author does still believe that the use of QoS Qbit may potentially benefit the performance of IPRT. However, given the circumstances used in the simulations in this thesis, it is clear that the cost of calculating the QoS Qbit is not justified in terms of better performance.

Cost

From the results it may be observed that the IPRT method is generally capable of responding to changes in link-cost.

One way to observe the changes in link-cost is to consider the median obained from the measurements done on the topologies. An example can be found in the abilene (Figure 8.13 and 8.12) in links 4, 6, 8, 9 and 10 or uninett (Figure 8.17 and 8.16) in links 3, 5, 6, 17 and 27. In both these examples the median in the re-converged simulations has a clear change in the link-load and in both examples IPRT is able to follow the changes.

In Table 8.4 one may observe that the median total load show a stable average when cost is introduced in geant and uninett. As more links are used to transfer packets between the destinations, it is expected that the total load generated will increase when link-cost is used. However, in both geant and uninett it may also be observed an elevated percentage value in max link increase and a decrease in percentage values in minimum link increase. This gives an indication that IPRT and the re-converged recovery paths differs more, compared to that of a flat link-cost, when using the cost specified by the topology. In the abilene topology this situation is reversed - providing more closely related paths when link-cost is introduced.

The reason for observing the differences between geant or uninett and abilene may stem from the IPRT tree generation. In the abilene topology the link cost enabled the paths obtained in the re-converged scenario to more closely relate to the IPRT recovery paths as the topology provide only small variations in the trees. In a similar manner, the geant or uninett re-converged scenarios using link-cost provided paths that IPRT was not flexible enough to fully respond to the changes. However, because IPRT is a local recovery procedure, it is expected that the recovery method may not be able to provide an optimal solution when compared to a re-converged scenario. Another reason for some of the links to be utilized, even if it has a low cost, may stem from the way the redundant trees are built. In the implementation used in this thesis the tree generation needs to follow strict rules, and may not always be able to freely choose between all available links.

Even though IPRT is generally capable of responding to the use of link-cost, more work needs to be done in order to obtain knowledge on how to configure and utilize link-cost in a IPRT enabled network. This is apparent from the link-load, where some failure scenarios do result in a utilization of links that should have been avoided.

Chapter 9

Conclusions and future work

IPRT is based on the redundant tree approach presented by Medard et.al [1], extended to provide a resource efficient way to populate the backup Forward Information Base (FIB) and furthermore, the mechanisms needed to utilize this information in the forwarding procedure. By reversing the redundant trees obtained for the tree generator and use the root as a destination, IPRT is able to provide IP fast recovery using two recovery FIBs and guarantee full coverage of all failures. Furthermore, IPRT may be the only IP fast recovery procedure to provide a fixed requirement in additional state information needed at the routers.

The main goal of the thesis has been solved:

- By supplementing the FIB used for normal operation with the two additional recovery FIBs, named r/bTable, and use tunneling or IP header marking for identifying recovered packets, IPRT is able to co-exist with normal routing protocols in times of failure-free operation.
- In order to provide local recovery, IPRT uses Qbits, a bit indicating what recovery FIB is preferred in a recovery situation, to be able to identify a recovery path unaffected of failure. The data-structure holding the Qbit information may be merged with the FIBs or be self-contained as an addition to the recovery FIBs. The Qbits may be populated through means of a probabilistic algorithm that is able to identify and mark the needed paths in a fast way, or through use of an exact procedure that requires more CPU usage but avoid the use of deflection in the forwarding procedure (see section 5.3.1). However, the exact procedure should be used in order to decrease the additional load associated with the use of IP fast recovery.
- Because the recovery FIBs are self-contained they are unaffected by any potential re-convergence.

One of the strongest assets of IPRT is the ability to provide 100 % coverage with a minimal constant amount of extra state information in each router. This may be achieved through use of the r/bTable solution for populating the recovery FIBs. In comparison to

RRL, this is a very good result; even though RRL is shown to need a small number of layers to provide IP fast recovery no guarantees may be given on the upper bounds.

Because IPRT may identify and correctly forward packets based on only the color of the recovery path and the destination, it gains an advantage in state information needed to provide the signaling mechanism. By guaranteeing that only two bits needs to be set in the IP header to correctly forward a recovered packet IPRT it may be easier to accommodate for IP headers to be used as signal-carriers in a real implementation. This could enable IPRT to be used without tunneling and thus avoid the increased overhead in network load and possible packet fragmentation associated with the use of IP encapsulating.

The ability to provide low additional state information is achieved at the cost of increased amounts of calculations when compared to RRL. To be able to populate the r/bTable each node in the network needs to perform a number of shortest path tree (SPT) calculations equal to two times the number of nodes in the network. RRL also requires one additional SPT computation per layer, but the number of layers needed by RRL is considerable lower than the number of redundant trees needed to populate the r/bTables in IPRT. At the present time the best known algorithm for generating a pair of redundant trees has a run-time of $O(n + v)$ [27].

The Qbits may provide QoS properties, however the mechanisms and situations where this may be fully utilized is not yet understood to a full extent; other approaches to generate the redundant trees may benefit from QoS Qbit to a larger degree than observed in this thesis.

IPRT is able to provide with a good load-distribution in failure situation, and provides good potential in the ability to respond to QoS link-cost. In the present implementation, there are situations that create hot-spots on certain links. However, the effect may be minimized by populate the Qbit in such a way that deflection may be disabled.

9.1 Future work

The main disadvantage of IPRT is the number of computations needed to create all the redundant trees. Future work should include an investigation if it is possible to provide incremental calculations of the trees. If this is possible, it could strengthen IPRT as a contestant among the IP fast reroute recovery procedures.

In this thesis, IPRT was implemented with a static routing approach. Future work could include research on how IPRT may interact with a real routing algorithm implementation as OSPF or IS-IS. In addition, it could prove interesting to learn how IPRT may be implemented in a hierarchical, segmented or area routing environment. Another topic that is to some extent related to this kind of routing environment is to provide recovery of multihomed destinations. IPRT may provide with an environment where this could be solved, and future work could include a research in this area.

IPRT may benefit from improved load-balancing and good heuristics for configuring the link-cost. Work has been done in this area by Xue et.al [16] [17], however the effect from using these algorithms are not known. Furthermore, IPRT may operate free of the routing protocol used for normal routing and could potentially gain from using separate link-costs or link-metrics. A possible path to follow is to dynamically update the metrics as the trees are built, gradually increasing the cost of the links included in the tree. Another approach may be to investigate if the Qbits may be used to spread the traffic more evenly during specific failures. I.e. try to balance used outgoing interfaces through use of Qbits given a specific neighbor node failure. This could be done locally on each node as the use of Qbit only governs the color of the recovery path.

Another topic for future work may be to investigate the ability to withstand n-failure situations. The procedure used to provide single failure IP fast recovery with IPRT may be used to protect against failures that are coherent. However, it is anticipated that other approaches apart from RT must be used to generate the appropriate trees, and run-time of such an algorithm may be of importance.

Bibliography

[1] Muriel Medard, Steven G. Finn, and Richard A. Barry. Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs. *IEEE/ACM Transactions on Networking*, 7(5):641–652, 1999.

[2] Andrew M. Odlyzko. Internet traffic growth: sources and implications. volume 5247, pages 1–15. SPIE, 2003.

[3] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an ip backbone. In *Proc. of ACM SIGCOMM Internet Measurement Workshop 2002, Marseille, France*, Nov 2002.

[4] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental study of internet stability and wide-area backbone failures. Technical Report CSE-TR-382-98, University of Michigan, 1998.

[5] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of failures in an ip backbone, 2004.

[6] Anindya Basu and Jon Riecke. Stability issues in ospf routing. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 225–236, New York, NY, USA, 2001. ACM Press.

[7] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure. Achieving igp convergence in large ip networks, 2005.

[8] Muriel Médard, Steven G. Finn, and Richard A. Barry. Wdm loop-back recovery in mesh networks. In *INFOCOM*, pages 752–759, 1999.

[9] R. Bartos and M. Raman. A heuristic approach to service restoration in MPLS networks, 2001.

[10] Cisco Systems Inc. Enhanced interior gateway routing protocol – white paper: http://www.cisco.com/warp/customer/103/eigrp-toc.html, 2003.

[11] G. Malkin. RIP version 2. RFC 2453, Internet Engineering Task Force, November 1998.

[12] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math*, 1:269–271, October 1959.

[13] J. Moy. OSPF version 2. RFC 2328, Internet Engineering Task Force, April 1998.

[14] A. Khanna and J. Zinky. The revised arpanet routing metric. *SIGCOMM Comput. Commun. Rev.*, 19(4):45–56, 1989.

[15] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing OSPF weights. In *INFOCOM (2)*, pages 519–528, 2000.

[16] Guoliang Xue, Li Chen, and K. Thulasiraman. Qos issues in redundant trees for protection in vertex-redundant or edge-redundant graphs. *IEEE International Conference on Communications*, 5:2766– 2770, 2002.

[17] Guoliang Xue, Li Chen, and K. Thulasiraman. Quality-of-service and quality-of-protection issues in preplanned recovery schemes using redundant trees. *IEEE Journal on Selected Areas in Communications*, 21(8):1332–1345, October 2003.

[18] A. F. Hansen, A. Kvalbein, T. Cicic, S. Gjessing, and O. Lysne. Resilient routing layers for recovery in packet networks. In Bob Werner, editor, *International Conference on Dependable Systems and Networks (DSN 2005)*. IEEE Computer Society, 2005.

[19] Stewart Bryant and Mike Shand. Ip fast reroute framework. *IETF Internet Draft, draft-ietf-rtgwg-ipfrr-framework-05.txt*, March 2006. (Work in Progress).

[20] A. F. Hansen, T. Cicic, and S. Gjessing. Alternative schemes for proactive ip recovery. In xxx, editor, *2nd Conference on Next Generation Internet Design and Engineering, Valencia, Spain April 3-5*, pages 1–8. IEEE, 2006.

[21] W. Simpson. Ip in ip tunneling. RFC 1853, Internet Engineering Task Force, 1995.

[22] J. Postel. RFC 791: Internet Protocol. RFC 791, Internet Engineering Task Force, September 1981.

[23] Stewart Bryant. Ip fast reroute using tunnels. *IETF Internet Draft, draft-bryant-ipfrr-tunnels-02*, April 2005. (Work in Progress).

[24] Stewart Bryant, Mike Shand, and Stefano Previdi. Ip fast reroute using not-via addresses. *IETF Internet Draft, draft-bryant-shand-ipfrr-notvia-addresses-02.txt*, March 2006. (Work in Progress).

[25] Steven McCanne and Sally Floyd. ns Network Simulator. http://www.isi.edu/nsnam/ns/http://www.isi.edu/nsnam/ns/.

[26] Hung-Ying Tyan. *Design, Realization and Evaluation of a Component-Based Compositional Software Architecture for Network Simulation.* PhD thesis, 2002.

[27] W. Zhang, G. Xue, J. Tang, and K. Thulasiraman. Linear time construction of redundant trees for recovery schemes enhancing qop and qos. *INFOCOM 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 4:2702– 2710, March 2005.

[28] Cisco Systems Inc. Multi-topology is-is – white paper, 2003.

[29] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. De Groot, and E. Lear. Address Allocation for Private Internets. Internet Engineering Task Force: RFC 1918, February 1996.

[30] Marina Fomenkov Ken. Longitudinal study of internet traffic in 1998-2003.

[31] Thomas Karagiannis, Mart Molle, Michalis Faloutsos, and Andre Broido. A nonstationary poisson view of internet traffic. In *INFOCOM*, 2004.

[32] A. Kvalbein, T. Cicic, and S. Gjessing. Routing efficiency with link failures using multiple routing configurations. Technical Report 02-2006, Simula Research Laboratory, 2006.

[33] *Pålitelighet og ytelse i informasjons- og kommunikasjonssystem.* tapir akademisk forlag, 2003.

[34] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast ip network recovery using multiple routing configurations. In Zhili Zhang Arturo Azcorra, Joe Touch, editor, *INFOCOM 2006*, pages xx–yy, Barcelona, Spain April 23 – 29, 2006. IEEE.

Chapter 10

APPENDIX A

Appended to the thesis is a CD-ROM containing the following hightlights:

- Readme file
- Source code for IPRT Tree Generator
- Source code for IPRT J-sim extentions
- Source code for J-sim 1.3 v3
- Simple example explaining how to simulate an IPRT enabled network in J-sim
- Topologies and traffic matrixes used in this thesis

The Readme file may be found at the root of the CD - specifying further usage and content.