MTRP: Multi-Topology Recovery Protocol

Paulo V. A. Pinheiro
Universidade Estadual do Ceará (UECE)
Av. Paranjana 1700
Fortaleza - CE - Brazil
paulovap@larces.uece.br

Marcial P. Fernandez
Universidade Estadual do Ceará (UECE)
Av. Paranjana 1700
Fortaleza - CE - Brazil
marcial@larces.uece.br

***Abstract*—Link and node failure recovery is critical in any production network and recover the failures in times below 50 milliseconds is desired to maintain the quality of real time application. In this paper, we propose the Multi-Topology Recovery Protocol (MTRP) that provides network protection using pre-calculate routes and a multi-topology approach. The protocol was based on the state-of-the-art recovery and protection techniques. MTRP is based on Resilient Routing Layers (RRL) algorithm, used to generate the sub-topology. The MTRP prototype was implemented and tested in a virtualized environment, providing real IP stack in an actual operating system. The tests show that the MTRP provides a quick convergence, below few milliseconds, similar to ring protection protocol for partial mesh topology. MTRP evaluation shows that it also produces recovery paths with costs (distance) almost as low as the primary ones.**

Keywords-network recovery; protocol; resilience.

I. INTRODUCTION

Since the inception of the Internet, the problem of protection and failure's recovery on networks has aroused interest of researchers. This area has received enough attention because it keeps the quality of services provided by communication networks as regards the stability and availability. This is done by using mechanisms that aim to ensure protection of the network. Not limited to this, these mechanisms also restore the network to its normal operating condition, since there is a failure situation. This ability that the network has to keep itself alive, thus in an operational state, is called, in literature, survivability or resilience [1]. The most networks are designed to take advantage of that assumption, exploring the use of two topologies types: mesh and ring topology.

Ring recovery protocols are simple and provide recovery in a shorter time, restoring the network to its fully functional state in a time close to 50 milliseconds. However, this topology has limited recoverability: only a single failure can be recovered (only one spare path). These are some examples of protocols for this type of architecture: Ethernet Automatic Protection Switching (EAPS) [2], Ethernet Ring Protection Switching (ERPS) [3] and SONET/SDH. A mesh recovery protocol permits broader recovery, limited only by the amount of multiple paths available, but it has a greater recovery time. Since the protocol does not know the network topology to perform its protection, it always takes longer time to calculate a new viable path, typically in the order of seconds. Despite this recoverability, it provides a slow return to the natural state of the network because, when there is an error, a signaling process to notify the topology change to all nodes starts. These are examples of this type of protocol: Spanning Tree Protocol (STP) [4], Open Shortest Path First (OSPF) [5] and Intermediate System to Intermediate System Routing Exchange Protocol (IS-IS) [6].

This paper proposes a new protocol to ensure recovery of a partially mesh network quickly and efficiently. This proposal takes advantage of better recoverability from a mesh network, but it offers a recovery time close to the ring topology networks, such as SONET/SDH. The protocol will be based on the pre-computation of paths and the use of multi-topologies for network recovery, based on a technique known as Resilient Routing Layers (RRL) [7]. Our contribution is to propose and validate a new recovery protocol based on a small variation of this algorithm. The proposal validation was made by implementing a prototype in a virtualized network, created with Mininet tool [8].

This article is organized as follows. In Section II, we present related works. In Section III, the proposal of the Multi-Topology Recovery Protocol (MTRP) protocol is described. Then, in Section IV, the proposal evaluation is described. In the Section V, results are shown. Finally, in Section VI, we present the conclusion and future works.

II. RELATED WORKS

These are some related works that propose failure recovery protocol in mesh topologies in a short convergence time.

Barreto [9] presents a proactive approach that is added to the OSPF protocol. Emergency paths are computed in advance in each node, adding a secondary route for each available neighbor. This proposal is based on IP Fast reroute scheme.

Psenak et al. [10] propose an RFC that describes an extension of the OSPF protocol, called OSPF-MT. This extension suggests the use of multi topologies for using of the routing protocol. Among the possible uses, there are: new route's creation, isolation of classes of service and the management. Przygienda et al. [11], proposed an extension

 ISBN: 978-1-61208-245-5

to the IS-IS Protocol, in which they suggest the use of multiple topologies for general purpose.

The literature presents many proposals of IP Fast Reroute (IPFRR) technologies [12]. IPFRR refers to the set of mechanisms aiming to provide fast rerouting using pure IP protocol forwarding and routing. Several proposals have been made to IETF IPFRR, such as, Release Point, Downstream Routes, Loop-Free Alternates, U-Turns and Not-Via Adresses [13]. The goal of the IPFRR mechanism is to set alternate routing paths, which avoid micro loops under node or link failures. However, IPFRR recalculates new routes after failure detection (reactive), and it requires to work only over IP protocol.

III. Multi-Topology Recovery Protocol (MTRP)

In this section, a new protocol for network recovery will be presented; whose primary goal is to ensure the recovery of failure in links or nodes in times near to the telecommunications networks based on rings (typically below 50 ms), using the least of redundant resources.

The protocol presented in this work was called Multi-Topology Recovery Protocol (MTRP), due to its most striking characteristic: using a set of sub topologies created from the actual topology of the network to keep the packets routing. MTRP has features to optimize the recovery process and provide new features to the routing protocol, making it more flexible, efficient and looking forward to becoming a good option for network's deployment.

A. Characteristics

This section aims to present several of these features which were incorporated to MTRP.

1) Local Recovery: When a failure occurs, the affected traffic is redirected by passing through the recovery mechanism, following a new path. This path, or more specifically the track used in the path, can be built through the principles of local and global recovery. The local recovery is done as close as possible to the point of failure, so in general, detection and recovery are performed much faster than the global recovery, and in most cases, requires fewer states. In order to achieve times shorter than $50ms$, MTRP protocol uses this local recovery scheme, where the traffic is rerouted to an alternative route to join a node next to the failure.

2) Hardware Failures Detection: The main mechanism to detect a failure is the hardware mechanisms in equipment physical layers. A mechanism on the equipment operating systems detects the link down by the loss of optical signal at an interface, starting the recovery procedure. So it can achieve a shorter failure detection time, and it can change to a new topology. In these there is not a hardware detection engine, it is necessary to send `HELLO` messages to supply this lack, consuming a little more network bandwidth and delaying the failure detection time.

3) Precalculated Recovery Paths: Using reactive schema to perform network nodes and link's recovery has been shown to be inefficient when the recovery time is important. Much time is spent in signaling; the dissemination of the new topology is slow and, finally, it is necessary to run an algorithm to generate the shortest path tree. A way to optimize the recovery time is the pre calculation of some steps that can be in advance for quick recovery. Using pre-computed paths to ensure recovery of all possible failures is impractical due to the large number of states that would be necessary. However, it is likely to cover 100% of single failures and ($\geq$ 80%) of multiple failures with few layers [7].

4) Multiple Topologies: The use of Multiple Topologies (MT) is a consequence of pre-calculating paths. Each possible path takes part of one (or more) viable topology. It is a relatively new approach, which many authors have been proposing to incorporate this feature to the traditional protocols such [10] [11]. Its basic principle is to generate virtual topologies based on its real network topology, where resources belonging to the network may not be present in all topologies.

5) Centralized Processing: The best set of routes to the network recoverability can be obtained when you have all the information about the topology and define the configuration in the same place. The Central Processor unit is the equipment responsible for the generation and distribution of routes to the routers, giving a huge management power to network operator.

B. MTRP Architecture

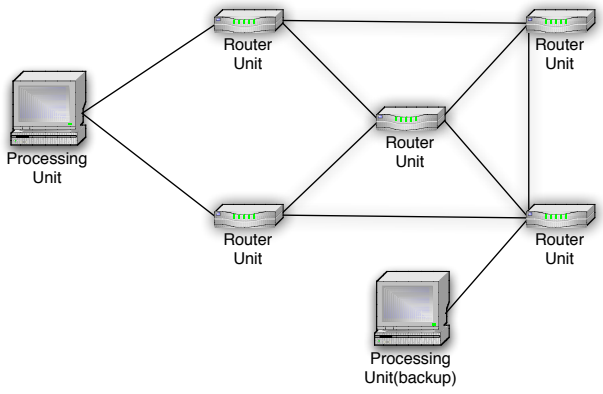

Figure 1. MTRP Architecture

The MTRP routing process involves two entities: the central node, called **Processing Unit (PU)**, which will be responsible for the generation of multiple network topologies and the routes' calculation, as well as some minor managerial activities, and the router, called **Router Unit**

(RU), which will manage the traffic that passes through the network using route tables generated by PU.

During the network initializing, each RU recognizes its neighbors and discovers the link metrics through their *HELLO* message. At the same time, the PU sends a message identifying itself as the control node. This message, called **Processing Unit Advertisement (PUA)**, is sent via broadcast to all the network RUs, which in their turn, learn by which interface it should communicate to the PU. If the PU receives this message from more than one interface, it will consider only the interface from which the PUA message came first.

Once the information about the neighbors and the path to the PU are established, each RU sends its routing table to the PU, using the message **Neighbor Advertisement (NA))** through the newly discovered PUA message path. PU receives all NAs and creates the actual topology, called **Real Topology (RTOP)**. With the RTOP defined the PU starts the process to create n virtual topologies, called **Virtual Topology (VTOP)**. The VTOP will be used to define the predefined routing paths (**Routing Table (RT)**).

C. Protocol Messages

The MTRP protocol uses the following messages: (1)Forwarding Packet, When a data packet enters on the network, it is encapsulated within a MTRP and it can be forwarded through the network; (2)Hello packets, used to maintain the neighborhood relationship between nodes, and detect failures that were not detected by the hardware; (3) Processing Unit Advertisement, Message sent just by PU to all RUs in the network to inform the active PU; (4) Neighbor Advertisement, message sent only by the RU to inform PU the list of its neighbor's nodes and the metrics of each link; (5)Route Database Update, after the routes are created, the PU will send this packet to update each RU routing tables, according to the number of sub topologies; (6)Acknowledgement, message used to confirm the messages received.

D. Processing Unit (PU)

The Processing Unit (PU) is the entity responsible for characterization, topologies generation, route's computation, topologies distribution and, finally, network monitoring.

1) Topologies Generation: The main feature of PU is the route generation, including pre-calculated primary and backup routes. In order to have it, it is necessary that PU has a set of VTOPs which will ensure these alternative routes, no matter how they are created. There are several ways to create VTOPs according to different network routing characteristics. VTOPs can be created prioritizing the size of backup paths, increasing coverage to multiple failures, minimizing the amount of VTOPs to simple failure's protection and many other possible customizations.

As a standard tool for VTOPs generation, the MTRP uses a variation of the Resilient Routing Layers (RRL) algorithm proposed in [14], showed at Algorithm 1. Our algorithm aims to ensure greater network redundancy, expanding coverage to multiple failures (*k-fault*) and avoid the back hauling effect [1]. The Algorithm 1 shows the layers generation.

Algorithm 1: MTRP: VTOP Generation for k-failures

Input: $G(V, E)$: Bidirectional Graph of Real Network Topology.
Input: $nTopo$: Minimum Number of Layers to be Generate
Input: k: Number of Simultaneous Failures Supported
Input: $artPoints(G)$: List of articulated nodes, i.e., nodes that if removed would split the network
Result: $L[i]$: List of Graphs of Virtual Topologies VTOP

```
S = artPoints(G)
foreach n ∈ V do
    c(n) = 0
    cl(E) = 0
end
i = 0
while (i < nTopo) or (|S| < |V|) do
    L_i(V_i, E_i) = G
    P = {}
    while |P| < |V| do
        n = min(c(n)) such that n ∉ P
        if n ∉ artPoints(L_i) then
            {l_1, ..., l_k} = links(n, E_i)
            E_i = E_i − {l_1, ..., l_k}
            e = min(cl({l_1, ..., l_k}))
            E_i = E_i + e
            cl(e) = cl(e) + 1
            S = S ∪ {n}
            c(n) = c(n) + 1
        end
        P = P ∪ {n}
    end
    i = i + 1
end
```

The algorithm starts by creating two lists ($c()$ and $cl()$) containing integers indicating how many times a node ($c(n)$) and a link ($cl(E)$) were used for layer's generation. Then, a loop starts in each sub topology created, which will be added to the list L. P stores the list of nodes already used for creation of the L_i sub topology. In the innermost algorithm loop, there is a link removal to ensure they will be saved by other layers. For each layer, the node less used is selected $min(c(n))$, so that, their edges are removed ($E_i = E_i - l_1, l_k, \dots$) excepting the not as much used edge ($E_i = E_i + e$)). After all nodes have been saved for at least one layer ($S = V$), the algorithm ends and returns the list $L(i)$ of sub topologies.

If the network is a connected graph, the path is guaranteed

by the routing algorithm, e.g., in our experiment Dijsktra's was used. The only constraints we assume is that the graph should have, at least, one path from any node to any other node in the graph.

E. Router Unit (RU)

The Router Unit (RU) is the entity responsible for the packets' forwarding on the network. Unlike to what happens with traditional routing protocols, such as OSPF and IS-IS, where the network control is done in a distributed way, the RU takes care only for the packet forwarding, report changes and errors. This ensures that the RU may become a simpler equipment with less processing power and thus, lower cost. Another advantage is the reduction of signaling overhead that occurs in distributed routing protocols, e.g., the tables' synchronization among routers and designated router election in OSPF. The following sections describe the RU state machine, the messages between RU-PU and the algorithms to recovery mechanism.

1) Signaling: There are two possible signals that start from RU: topology change and packet drop alert. The topology change may occur due to a failure, resulting in the loss or addition of links or nodes. The change notification is sent only to the PU and occurs in parallel to other RU activities, there is no dependency in that signals nor other critical router functionalities.

2) Recovery Mechanism: Since its route tables are completed, the RU is able to perform the packet forwarding and repair paths down. As described previously, the protocol used the table referring to RTOP during its regular functioning. Once there is the failure, the neighbor, that would use it as the next hop to deliver the packet, runs the Algorithm 2 to select a new routing table that does not present the failure element.

Algorithm 2: MTRP: RU Packet Forward Algorithm

```
Input: T_n: Route table
Input: pkt: Packet to be forward

sent = False;
if sendNextHop(pkt) not True then
    foreach t ∈ T_n do
        if canSendVia(t,pkt) then
            mark(pkt,t);
            sendNextHop(pkt);
            sent = True;
        end
    end
    if sent not True then
        drop(pkt);
    end
end
```

At the beginning of the Algorithm 2, the RU tries to send the packet using the RTOP routing table, or the VTOP table in use marked on the packet. The packet is sent using the `sendNextHop()` function that receives the packet to be sent as a parameter. This function checks which table should be used reading $TOPOIDX$ field on packet and checks if the next hop in this table is possible (if the link has no failures). If it is not possible to send, this function returns the value `False`, which causes the algorithm to find a new route table to be used for this packet. Once the destination is found on the table, the function `canSendVia` returns the value `True` indicating that the table t can be used. The packet is marked by the `mark` function to indicate the next RU which table should be used. Then it is sent by `sendNextHop` function, but now using another table.

IV. Prototype Validation

The computational virtualization enables various operating systems to run concurrently on shared hardware equipment, so it is possible to have multiple operating systems logically isolated between them running on a single hardware. The use of virtual machines connected through network interfaces has been shown to be quite efficient and presents a higher degree of realism and interactivity than the use of network simulators like ns-2 [15].

MTRP evaluation shall be carried out in a scenario created for the rapid prototyping network tools Mininet [8]. This tool creates a network topology through the use of *Lightweight Virtualization*, i.e., a virtualization scheme where an isolated environment is created to group processes in containers, where logical devices of each container are not shared among themselves.

Scenarios were created based on real-world topologies: GEANT, IINET and SPRINT. Database containing information about these and other actual topologies can be found on [16]. For worst case evaluation; it was used the bigG topology, an artificial network generated from the algorithm proposed in [17], containing 123 nodes and 243 links.

For each topology, the metrics, network initialization time, recovery path distance, memory used to store routing table and time to recover a simple failure will be reviewed. With such data, we can have an overview of the performance of the protocol on these networks.

V. Results

In this section, the results of the tests will be presented and analyzed, demonstrating the effectiveness of the protocol.

A. Network Initialization

Tests to establish the network startup time were made to define how the use of a centralized point for processing and distribution routes would affect the network startup. The startup time is the time interval between the first `HELLO` message sent by PU until the last `ACK` message received by

 ISBN: 978-1-61208-245-5

it, indicating that all routes were sent successfully. Table I shows the startup times for each topology [18].

Table I
NETWORK INITIALIZATION TIME

Topology	Network Initialization Time (sec)
Sprint$(11, 18)$	0.89882
Geant$(27, 38)$	0.999489
Iinet$(31, 35)$	1.03966
medG$(42, 81)$	1.51030
bigG$(123, 243)$	5.1478

According to the Table I, the network boot time is around one second, gradually increasing with the rise of network complexity. The MTRP protocol presented low boot times compared to others such as Routing Information Protocol (RIP) and OSPF. According to [18], in a network with seven routers, the OSPF protocol takes, in average, ten seconds to carry out the convergence of its nodes, which is about the time of a network initialization. RIP takes, in average, more than 100 seconds.

B. Average Route Length

Table II shows the average of all possible paths between two different nodes in the whole topology in all topologies generated by MTRP algorithm. To perform the calculation the Equation 1 was used:

$$\bar{x} = \sum_{s,t \in V} \frac{d(s,t)}{n(n-1))} \quad (1)$$

where s and t are nodes belonging to the network, n represents the total number of nodes and $d(s,t)$ is the smallest path which goes from s to t. Table II shows the results of average of route's length.

In a failure situation, the protocol will use one of the sub-topologies created to perform the packets' forwarding, choosing the shortest path between source and destination. The failure was simulating by removing a link or a node. However, we only considered the situations where the graph still connected, i.e., there is at least one backup path. Non-connected graphs cannot be fully restored, so it is impossible to infer an average route length.

On the Sprint network, it was noted that the paths generated by sub-topologies are on average $1.19x$ to $1.49x$ larger than the original path without failures. Geant presented a variation from $1.10x$ to $1.50x$ and, finally, the algorithm obtained from $1.10x$ to $1.24x$ in Iinet. BigG, in its turn, presented sub-topologies with the smallest paths (Topo3 - $1,002x$) as well as the largest ones (Topo1 - $3.6x$) in the tests.

The VTOP generation algorithm used does not consider the topology metrics to split the network into layers. So, it was created the "bad" Topo 1 topology on bigG. However, the table shows the worst path in this topology, so there

Table II
MTRP - AVERAGE PATH LENGTH (NR. OF HOPS)

Topology	Real	Topo 1	Topo 2	Topo 3
Sprint$(11, 18)$	1.89091	2.2545	2.8363	2.3818
Geant$(27, 38)$	2.9373	4.5641	4.2222	3.2421
Iinet$(31, 35)$	2.8215	3.1053	3.5053	3.3677
bigG$(123, 243)$	2.9648	10.6743	3.3145	2.9722

are other topologies that give the optimal paths, thus more eligible to be used as recovery topology. This is not really an issue once you use a fair number of virtual topologies (5, 6..), and it would not be difficult to generate the VTOP that the "good" paths would be fairly distributed by the VTOPs.

It is noticed that the algorithm generates good recovery paths, with sizes close to the original one. Even with good results, there are some ways to improve the algorithm so that it generates shorter paths. The use of more topologies is necessary, and each topology may have a larger set of bindings. It is important to note that as a consequence of the number of topologies rises, there is also the increase use of memory in the router.

C. Memory Use

The amount of memory used for tables storing is an important parameter for the protocol, because the creation of new sub topologies implies the increase of memory usage for storing the routing tables. For the scenarios described earlier, Table III shows the amount of memory use for each topology, when the protocol is written in Python [19]; and the amount of the memory use in an equivalent protocol written in C.

Table III
MEMORY USED TO STORE ROUTING TABLE

Topology	Topo number	Mem(Py)	Mem(C)
Sprint$(11, 18)$	3+1	7.5923 kb	352 b
Geant$(27, 38)$	3+1	21.1996 kb	864 b
Iinet$(31, 35)$	3+1	22.2615 kb	992 b
bigG$(123, 243)$	3+1	84.1110 kb	3.84375 kb

The Python implementation of the protocol implies in high memory usage. It is caused by the excessive use of the object orientation. To store the value of the link cost or router ID in Python, we need to use an *integer* type, which is 24 bytes long. In a more efficient implementation in C, these elements would be stored in an *unsigned int* of 4 bytes. In Table III we can see that the memory usage in an implementation in C is small, that permit to store more layers in the router. For the memory estimation in C, the we use the formula $mem = |V| * (t + 1) * obj$, where $|V|$ is the number of vertex of the graph, t is the number of sub topologies created and obj is the memory size required to store the RouterID and the link cost (8 bytes). In summary,

 ISBN: 978-1-61208-245-5

the use of memory space for storage of routing tables grows linearly ($O(t)$) with the number of created sub topologies.

D. Network Recovery Time

Recovery time of MTRP protocol basically consists of three steps: failure detection, next-hop table lookup and set the packet tag. In our proposal, any failure detection protocol can be used, e.g., the Bidirecional Forwarding Detection (BFD) protocol [20]. BFD protocol works with IP protocol and guarantee link failure detection in milliseconds (usually below $50ms$).

The Table IV shows the times found in tests to perform the table lookup of the next hop, returning the network address of the next hop of the new path, without failures. The values showed in Table IV did not consider the time of failure detection, in order of tens of milliseconds. The packet should follow the next hop and the time of packet tag, indicating which recovery table will be used. Then, to perform the recovery time, the tasks were performed one thousand times, to define the table lookup execution time with greater accuracy. To get the recovery time, only the worst case was considered. The tests were conducted by generating different amounts of sub topologies.

Table IV
RECOVERY TIME OF SINGLE FAILURE

Topology	t(topo=4)	t(topo=8)	t(topo=12)
Sprint$(11, 18)$	2.740 μs	5.279 μs	7.463 μs
Geant$(27, 38)$	2.995 μs	5.524 μs	7.204 μs
Iinet$(31, 35)$	2.790 μs	5.031 μs	7.422 μs
bigG$(123, 243)$	3.119 μs	5.139 μs	7.049 μs

The table lookup time grows linearly considering the quantity of topologies used by network, as it was expected due to its linear complexity algorithm. Packet tagging time is constant and was included in the test's execution time. Considering one thousand executions, we can realize that the tasks of lookup and packet tagging are performed on the microseconds' scale (10^{-6}), becoming insignificant compared to the hardware detection time. Therefore, the recovery time may be considered, only the failure detection time; it shows a big performance gain compared to the reactive protocols that take more than a second to perform the recovery, such as OSPF.

VI. CONCLUSION AND FUTURE WORKS

Through the results presented, it is possible to verify the viability of the MTRP protocol for network recovery in mesh topology. The MTRP shows recovery times in the milliseconds range, equivalent to SDH/SONET networks. In addition, you can see that the use of a central authority reduces the amount of processing required for signaling, compared to distributed protocols. Reducing the amount of messages relieves the individual processing of each router, ensuring the possibility of the use of equipment with less processing power to perform the transfer of packets. The subtopologies generation is done only on network initialization, 5 sec in BigG network (bigger than tradicional operator backbones), we may guarantee the proposal scalability for real networks. As the recovery is done using pre computed paths, there is no impact of network length in recovery time.

The MTRP promotes a quick startup, because link state distribution throughout the network is not necessary, only to the control node, PU. Our work also confirms the validation of RRL algorithm done in [7] and [14].

The MTRP protocol presents a range of possibilities for future works. To continue the development of the MTRP protocol, a better definition of message authentication system and use of header fields are important. Several improvements can be made to increase the protocol performance, such as implementation in C and integration of routing packets with the kernel of the operating system. Another possibility is the addition of Quality of Service (QoS) extension in protocol. A network can provide different classes of service to clients and divide its network into sub topologies with distinct QoS classes, where their packets will be routed exclusively for each sub topology. When a packet from a client joins the network edge, this packet would be already marked for the topology to which the class belongs.

Finally, a study on the integration of the protocol with the *OpenFlow* technology can be made. The centralizing feature of the MTRP protocol is similar to the OpenFlow control architecture, so the adaptation should not be difficult.

REFERENCES

[1] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[2] S. Shah and M. Yip, "Extreme Networks' Ethernet Automatic Protection Switching (EAPS) Version 1," RFC 3619 (Informational), Internet Engineering Task Force, Oct. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3619.txt

[3] J. Ryoo, H. Long, Y. Yang, M. Holness, Z. Ahmad, and J. Rhee, "Ethernet ring protection for carrier ethernet networks," *Communications Magazine, IEEE*, vol. 46, no. 9, pp. 136–143, 2008.

[4] R. Perlman, *Interconnections: bridges and routers*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1992.

[5] J. Moy, "OSPF Version 2," RFC 2328 (Standard), Internet Engineering Task Force, Apr. 1998, updated by RFCs 5709, 6549. [Online]. Available: http://www.ietf.org/rfc/rfc2328.txt

[6] D. Oran, "OSI IS-IS Intra-domain Routing Protocol," RFC 1142 (Informational), Internet Engineering Task Force, Feb. 1990. [Online]. Available: http://www.ietf.org/rfc/rfc1142.txt

 ISBN: 978-1-61208-245-5

[7] A. Kvalbein and A. F. Hansen, "Fast recovery from link failures using resilient routing layers," in *ISCC '05: Proceedings of the 10th IEEE Symposium on Computers and Communications*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 554–560.

[8] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *HotNets*, G. G. Xie, R. Beverly, R. Morris, and B. Davie, Eds. ACM, 2010, p. 19.

[9] F. Barreto, "Esquema de caminhos emergenciais rápidos para amenizar perdas de pacotes," Ph.D. dissertation, Universidade Técnologica Federal do Paraná, 2010.

[10] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF," RFC 4915 (Proposed Standard), Internet Engineering Task Force, Jun. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4915.txt

[11] T. Przygienda, N. Shen, and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)," RFC 5120 (Proposed Standard), Internet Engineering Task Force, Feb. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5120.txt

[12] M. Shand and S. Bryant, "IP Fast Reroute Framework," RFC 5714 (Informational), Internet Engineering Task Force, Jan. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5714.txt

[13] M. Gjoka, V. Ram, and X. Yang, "Evaluation of ip fast reroute proposals," in *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on*. IEEE, 2007, pp. 1–8.

[14] T. Čičić, A. Hansen, S. Gjessing, and O. Lysne, "Applicability of resilient routing layers for k-fault network recovery," *Networking-ICN 2005*, pp. 173–183, 2005.

[15] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu *et al.*, "Advances in network simulation," *Computer*, vol. 33, no. 5, pp. 59–67, 2000.

[16] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765–1775, october 2011.

[17] S. Dorogovtsev, A. Goltsev, and J. Mendes, "Pseudofractal scale-free web," *Physical Review E*, vol. 65, p. 066122, Jun 2002.

[18] H. Pun, "Convergence behavior of rip and ospf network protocols," Ph.D. dissertation, University of British Columbia, 1998.

[19] M. Lutz, *Programming python*. O'Reilly Media, Inc., 2011.

[20] D. Katz and D. Ward, "Bidirectional Forwarding Detection (BFD)," RFC 5880 (Proposed Standard), Internet Engineering Task Force, Jun. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5880.txt

 ISBN: 978-1-61208-245-5