4-6-2017

Resilience of Cloud Networking Services for Large Scale Outages

Mahsa Pourvali
University of South Florida, mpourvali@mail.usf.edu

Resilience of Cloud Networking Services for Large Scale Outages

by

Mahsa Pourvali

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Nasir Ghani, Ph.D.
Richard Gitlin, Sc.D.
Ismail Uysal, Ph.D.
Srinivas Katkoori, Ph.D.
Tao Zhang, Ph.D.

Date of Approval:
March 21, 2017

Keywords: Survivability, Network Virtualization,
Disaster Recovery, Progressive Recovery

Dedication

I would like to dedicate this dissertation to:

My mom, a strong woman whose unconditional love and encouragement inspires me to grow. Her support and sacrifice made it possible for me to get to this stage

The loving memory of my dad, who is and will always be the role model of my life and a constant source of pride

My sister, Aysa, who has always believed in me and never left my side

Acknowledgments

First and foremost, I would like to express my strong gratitude and appreciation to my supervisor, Dr. Nasir Ghani for his continuous support, caring, patience, motivation, and advisement. He has been supportive since the very first day I joined his research group, making me feel that I am a part of a family here. He has been a tremendous mentor for me, in my research as well as my career. He provided me with an excellent atmosphere throughout my Ph.D. program and continually encouraged my work and allowed me to grow.

I would like to thank my dissertation committee members, Dr. Richard Gitlin, Dr. Ismail Uysal, Dr. Srinivas Katkoori, and Dr. Tao Zhang, for their valuable guidance and advice on this dissertation work. I would also like to thank my research colleagues, Dr. Feng Gu, Dr. Kaile Liang, Dr. Hao Bai, Mr. Diogo Oliveira, Mr. Mohammed Jasim, and Dr. Khaled Shaban for their suggestions and collaboration in my research. I would like to thank all my amazing friends as well, for their great support and encouragement throughout this journey.

And most especially, my heartfelt gratitude and love goes to my parents and my sister for their endless support. Words cannot express how grateful I am to have them. I would not have made it this far without my family. My dad, who has passed on, taught me that being a good human is the first and most important goal of one's life. Having a father like him is a great source of pride to me. My mom has always encouraged me to continue my education. She is an ever present support for me and has always stood by me through life's ups and downs. My sister, Aysa, is my best friend and has always believed in me. She has always looked out for me and encouraged me to pursue my dreams. I would also like to thank my dog, Oreo, who patiently waited for me all days that I worked late. He constantly reminds me the meaning of faith, trust, and hope.

Table of Contents

List of Figures

Abstract

Cloud infrastructure services are enabling organizations and enterprises to outsource a wide range of computing, storage, and networking needs to external service providers. These offerings make extensive use of underlying network virtualization, i.e., *virtual network* (VN) embedding, techniques to provision and interconnect customized storage/computing resource pools across large network substrates. However, as cloud-based services continue to gain traction, there is a growing need to address a range of resiliency concerns, particularly with regards to large-scale outages. These conditions can be triggered by events such as natural disasters, malicious man-made attacks, and even cascading power failures.

Overall, a wide range of studies have looked at network virtualization survivability, with most efforts focusing on pre-fault protection strategies to set aside backup datacenter and network bandwidth resources. These contributions include single node/link failure schemes as well as recent studies on correlated multi-failure "disaster" recovery schemes. However, pre-fault provisioning is very resource-intensive and imposes high costs for clients. Moreover this approach cannot guarantee recovery under generalized multi-failure conditions. Although post-fault restoration (remapping) schemes have also been studied, the effectiveness of these methods is constrained by the scale of infrastructure damage. As a result there is a pressing need to investigate longer-term post-fault infrastructure repair strategies to minimize VN service disruption. However this is a largely unexplored area and requires specialized consideration as damaged infrastructures will likely be repaired in a time-staged, incremental manner, i.e., *progressive recovery.*

Furthermore, more specialized *multicast VN* (MVN) services are also being used to support a range of content distribution and real-time streaming needs over cloud-based infras-

tructures. In general, these one-to-many services impose more challenging requirements in terms of geographic coverage, delay, delay variation, and reliability. Now some recent studies have looked at MVN embedding and survivability design. In particular, the latter contributions cover both pre-fault protection and post-fault restoration methods, and also include some multi-failure recovery techniques. Nevertheless, there are no known efforts that incorporate risk vulnerabilities into the MVN embedding process. Indeed, there is a strong need to develop such methods in order to reduce the impact of large-scale outages, and this remains an open topic area.

In light of the above, this dissertation develops some novel solutions to further improve the resiliency of the network virtualization services in the presence of large outages. Foremost, new multi-stage (progressive) infrastructure repair strategies are proposed to improve the post-fault recovery of VN services. These contributions include advanced simulated annealing metaheuristics as well as more scalable polynomial-time heuristic algorithms. Furthermore, enhanced "risk-aware" mapping solutions are also developed to achieve more reliable multicast (MVN) embedding, providing a further basis to develop more specialized repair strategies in the future. The performance of these various solutions is also evaluated extensively using custom-developed simulation models.

Chapter 1

Introduction

This dissertation addresses the resilience of cloud-based services, with the focus of post-fault repair strategies and reliable multicast embedding. In order to properly introduce the work, this chapter provides a review of related research and then presents the key motivations for the dissertation effort. The core contributions are then discussed briefly followed by the dissertation outline.

1.1 Background Overview

Cloud computing technologies are allowing users to outsource their infrastructure and service needs in a scalable "pay-as-you-grow" manner. Namely, clients can purchase customized *virtual machine* (VM) setups at datacenter sites to support dedicated development and application hosting needs. As these services gain traction, there is a further need to distribute and interconnect virtualized datacenter resources across multiple sites to improve user responsiveness and reliablility [1],[2]. This interconnection is typically done by establishing bandwidth connections over underlying network substrates. As a result, the expanded *network virtualization* [3] concept has been defined to incorporate a wide range of resource types, i.e., bandwidth, computation, and storage.

Now evolutions in cloud-based services have also led to changes in the operator space, with a more clear separation emerging between *infrastructure providers* (InP) operating physical networks and higher-level *cloud service providers* (CSP) providing virtualized services [4]. Currently many organizations and enterprises have already migrated their applications and services to the cloud, achieving high cost efficiency and scalability. As service providers gain experience with these new paradigms, many are also starting to offer more expansive *virtualized infrastructure* services with full *quality of service* (QoS) support. These offerings limit the need to build and maintain dedicated, costly on-premises datacenter infrastructures, yielding sizable reductions in capital and operational expenditures.

In general, cloud-based services are classified into three types, i.e., *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), and *Software as a Service* (SaaS), see Figure 1.1. IaaS is termed as the lowest-level service and provides a basic VN infrastructure on top of which clients can deploy customized software applications and data repositories. Meanwhile, PaaS and SaaS represent higher-level offerings with more specialized provisions at the VM level, i.e., to run specific operating systems or complete turnkey applications, respectively. However, both PaaS and SaaS generally rely upon underlying IaaS provisions. As a result, this base service type is the focus of this dissertation research.

Now most virtual infrastructure services (IaaS) make extensive use of *network virtualization* techniques to provision customized *virtual network* (VN) overlays for client-specific needs. Figure 1.2 shows an overview of this setup running on top of a cloud-based infrastructure comprising of datacenter sites interconnected by a high-bandwidth networking substrate, i.e., typically *Internet Protocol* (IP)/*multi-protocol label switching* (MPLS) or optical *dense wavelength division multiplexing* (DWDM). Namely, a VN demand consists of multiple *distributed* storage and computing pools, i.e., VN nodes, interconnected by bandwidth connections, i.e., VN links. These VN nodes are "mapped" onto dedicated resources at datacenter sites, whereas VN links are "mapped" onto underlying connections across the substrate network. Hence the main resource provisioning problem in network virtualization involves "mapping" VN demands, commonly modeled as VN graphs, onto physical substrate resources, i.e., *VN embedding* (VNE) [5]. Carefully note that most clients (and non-InP providers) are not concerned with the actual physical location of their mapped resources, granted they meet necessary QoS, policy, and privacy constraints.

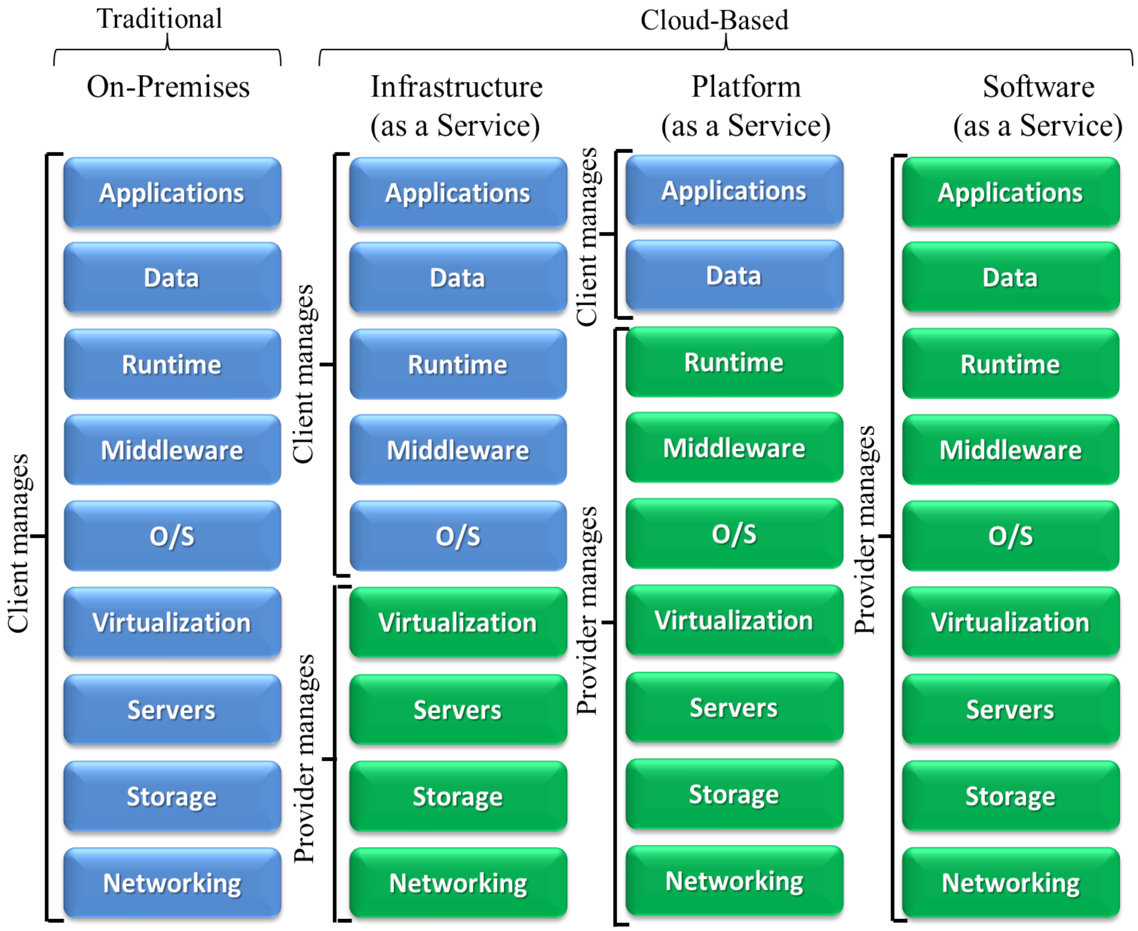


Figure 1.1: Cloud-based service models: on-premises, IaaS, PaaS, SaaS

Overall, VN embedding is a *NP*-hard problem given the high dimensionalities involved, [6]. Hence many different algorithms have been developed to map VN demands over cloud-based infrastructures. For example, some researchers have proposed optimization formulations to minimize resource usage or maximize revenue, see [5]. More computationally-feasible heuristic schemes have also been proposed using two-stage or joint single-stage mapping schemes [4],[5]. The former map all VN nodes to their substrate nodes first and then route VN link connections. Meanwhile, the latter jointly map VN nodes and their associated VN links. Some of the main provisioning objectives here include revenue maximization and cost minimization.

However as more and more applications migrate to the cloud, users are demanding improved service reliability and availability. In particular, large-scale outage events are of particular concern here as they can generate multiple near-simultaneous node/link failures in a given geographical region, i.e., high levels of spatial and temporal fault correlations. In turn, these occurrences can disrupt many VN services [7]. For example, these outages can be caused by a range of natural disasters (hurricanes, floods, earthquakes, storms), as well as malicious man-made attacks, e.g., *weapons of mass destruction* (WMD). Furthermore, these events can also trigger further cascading failures, propagating network and service disruptions across extended geographical and temporal domains, e.g., via damage to critical power grid infrastructures.

1.2 Motivations

Since cloud services reliability is now a major concern, researchers and practitioners have started to develop a range of VN survivability schemes. For example, many providers already use intra-site server and storage redundancy for rapid post-fault recovery [1]. However these schemes cannot recover from larger datacenter site outages or link failures. Hence more capable *survivable* VNE schemes have also been investigated to leverage network substrates and reserve/disperse backup resources over multiple sites, i.e., backup VN nodes or VN links. However most of these solutions only treat single isolated failures, [8],[9], and are ineffective against large disaster-type events affecting multiple (network, datacenter) sites.

In light of the above, some efforts have also incorporated "geo-redundancy" into the VN mapping process to improve resiliency against large regional outages. For example, recent studies have developed pre-provisioned schemes to place backup VN node/VN link resources to recover from such events, [10], [11]. However, most of these studies assume

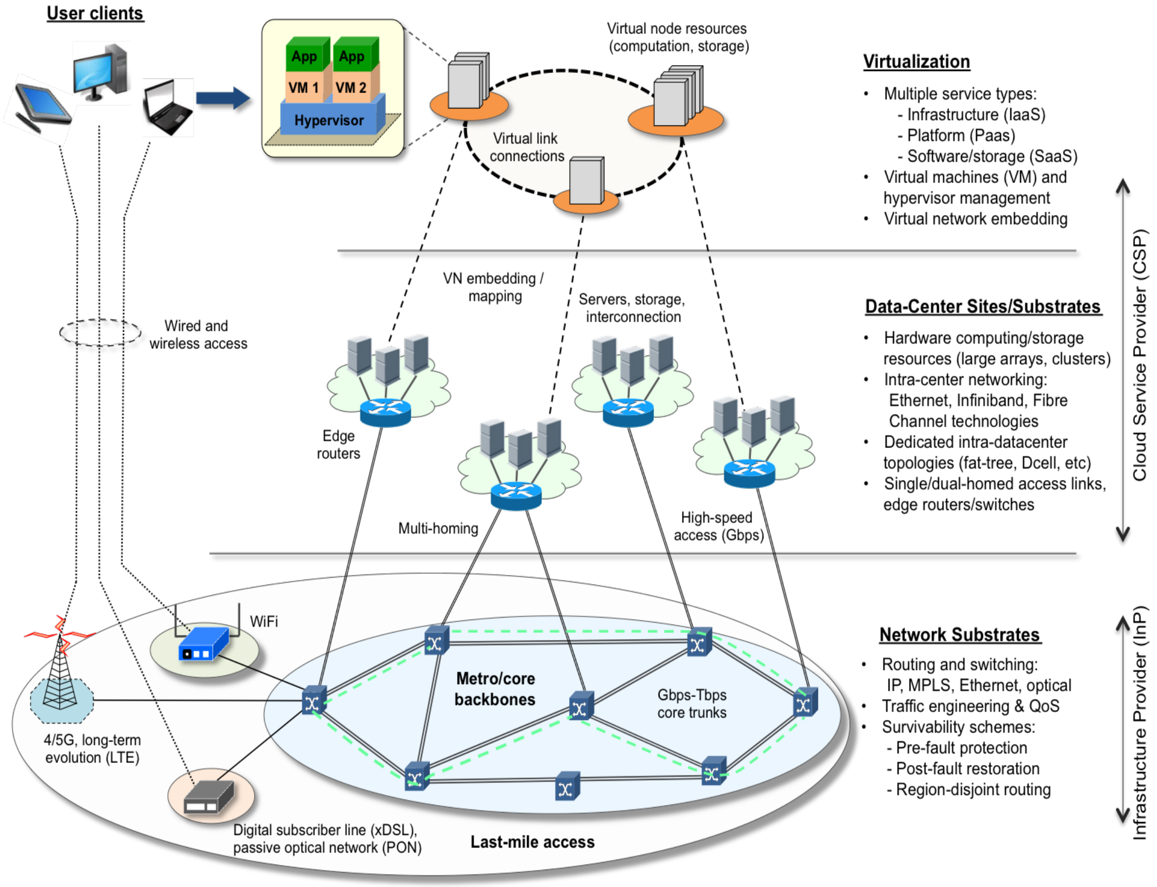


Figure 1.2: Network virtualization and cloud-based user services

static a-priori (probabilistic) failure region models and yield notably higher resource usages and costs. Moreover, most pre-provisioned schemes are susceptible to inaccuracies in the a-priori failure region models and therefore cannot guarantee recovery from all multi-failure conditions [12]. Furthermore, researchers have also studied *multicast VN* (MVN) embedding, [13],[14]. These services represent a critical sub-class of VN demands and are widely used for content distribution and real-time streaming over cloud-based setups. However few efforts have looked at mapping MVN demands in the context of large multi-failure outages [15],[16].

In light of the above, there is a critical need to address *post-fault* infrastructure repair and service recovery in cloud-based environments. Now damaged cloud and networking substrate infrastructures will likely be repaired in multiple *progressive* stages as backup resources arrive. At the same time, most service providers will want to maintain a partial "degraded" level of service for their affected client demands. Hence the intelligent placement, i.e., scheduling, of incoming repair resources will be critical for accelerating VN service recovery and lowering revenue losses and penalties. However only a few studies have investigated progressive recovery design, mostly for point-to-point demands [17]. Although recent work in [18] has studied post-fault infrastructure repair for VN demands, this study makes some overly-simplifying assumptions which will not hold in realistic scenarios. Furthermore, it is also important to address the reliability of more specialized MVN services under large-scale outages. The few efforts in this space, have mostly focused on backup resource (protection) provisioning and tend to treat MVN demands as a series of unicast demands.

1.3 Problem Statement

In light of the above, this dissertation is motivated by the growing need to improve the resiliency of cloud-based infrastructure services, particularly under large-scale outage conditions. Specifically, effective *post-fault* infrastructure repair strategies are required to recover affected services in a timely manner. Specialized mapping solutions are also needed to improve the reliability of MVN services.

To address these concerns, a range of innovative solutions are developed here, including novel heuristic and metaheuristic progressive recovery/repair strategies for VN demands and "risk-aware" MVN embedding schemes. The detailed performance of these proposed strategies is also evaluated and analyzed extensively using discrete-event network simulation.

1.4 Proposed Work and Contributions

This dissertation addresses some critical open challenges in the area of cloud-based services resilience against large-scale outage events. In summary, the main contributions of this work include the following:

- New optimization model to realistically characterize the infrastructure repair process for progressive recovery of VN demands (including finite repair resources and partial demand recovery).

- Scalable polynomial-time heuristic schemes for progressive recovery of affected VN demands (including methods based upon random, physical network connectivity, virtual load, and required resources).

- Metaheuristic algorithms to bound the performance of the above heuristic schemes and provide a feasible solution for the optimization models.

- Novel "risk-aware" heuristic embedding strategies to improve the reliability of MVN services embedding (including algorithms based upon resource usage minimization, failure risk minimization, and hybrid resource usage and failure risk).

Overall, the rest of this dissertation is organized as follows. Chapter 2 presents a brief survey of background work on embedding and survivability of VN and MVN services as well as progressive recovery. The proposed infrastructure repair/progressive recovery framework is then presented in Chapter 3, along with a detailed optimization model and some novel *simulated annealing* (SA) metaheuristic schemes. More scalable polynomial-time heuristic repair methodologies are then presented and analyzed in Chapter 4. Finally, reliable MVN embedding schemes are also presented in Chapter 5, providing a basis for the development of more specialized repair strategies in the future. Detailed conclusions and future research directions are then presented in Chapter 6.

Chapter 2

Background

Network virtualization provides a very effective means for infrastructure providers to amortize their deployed base and also support a wide range of clients, i.e., including CSP operators, content providers, etc. However these trends and evolutions are leading to increased levels of service multiplexing and resource utilization, bringing crucial survivability concerns to the fore. As a result there is a growing body of research on VN survivability, a high-level graphical taxonomy of which is also shown in Figure 2.1. Hence this chapter surveys the relevant work in this space, including a review of multi-failure VN recovery and *multicast VN* (MVN) survivability schemes as well as progressive recovery/infrastructure repair. Further open challenges are then highlighted in order to properly establish the motivations for this research effort.

2.1 Virtual Network Embedding (VNE)

The VNE design problem is known to be *NP*-hard via reduction to the multi-way separator problem, i.e., including the special case of mapping VN links with fixed VN nodes (non-splittable demands) [6]. Therefore researchers have developed a range of VN mapping algorithms. Most notably, several *mixed integer linear programming* (MILP) formulations have been tabled to minimize resource usage or maximize revenue [5]. However due to high variable counts/intractability, more efficient heuristic schemes have also been evolved using two-stage and joint single-stage VN node and VN link mapping strategies. The former methods map all VN nodes to substrate nodes first and then map the VN links. Conversely, the latter methods jointly map VN nodes and their VN links. The key objectives here include revenue maximization and cost reduction [4],[5] and findings show that single-stage strategies yield better efficiency and higher revenues. Consider some further details.

The authors in [19] propose a two-stage VN embedding algorithm which computes ranks for all VN nodes and substrate nodes. These nodes are then sorted in descending order of their ranks, and the higher-ranked VN nodes are embedded onto higher-ranked substrate nodes first. Meanwhile, substrate links that do not meet VN link bandwidth requirements are pruned during the link mapping stage. A shortest-path algorithm is then used to route the VN link connections. The proposed scheme gives lower blocking rates and higher revenues as compared to some existing methods.

Meanwhile [20] proposes another two-stage VN embedding scheme using a window-based batch processing method. Namely, VN requests are accumulated during a batch window and then embedded in descending order of revenue. The VN nodes in each request are sorted in descending order of required resources and mapped onto substrate nodes with the minimum node stress and highest available resources. Link embedding is then done using a k-shortest path algorithm. Results show that the proposed approach reduces substrate node and link stress, but additional results on request blocking are not presented.

Furthermore, the authors in [6] also propose some VNE algorithms to coordinate between the VN node and VN link embedding phases. Namely, an augmented graph is created by adding a "meta-node" for each virtual node and connecting it to a subset of substrate nodes via "meta-edges". The virtual node is then mapped onto the substrate node with the maximum flow passing through its meta-edge. Virtual links are then embedded by solving an optimal flow allocation problem. An *mixed integer programming* (MIP) formulation is also proposed to increase revenue and decrease cost. However due to high intractability, further optimization rounding techniques are used to relax the integer constraints. A lookahead algorithm is also used to map VN demands within a window. Namely, each request is assigned a deadline to specify how long it can wait, and all expired requests are rejected. Results show that these algorithms yield better acceptance ratios and higher revenues with load balancing, albeit with higher costs.

Meanwhile [21] proposes a hybrid VNE algorithm which uses both single and two-stage mappings and also implements batch processing. Namely all batch requests are sorted and processed in ascending order of their lifetimes. Furthermore, each individual VN request is decomposed using a k-core decomposition algorithm to recursively prune VN nodes with unity node degree. This process terminates when there are either no more VN nodes left or there are only two VN nodes. The remaining VN nodes and links are then used to form the "core" VN network, whereas the pruned entities are used to form "edge" networks. The core and edge networks are then mapped in different phases using existing two-stage and single-stage embedding schemes, respectively. Results show that this hybrid algorithm yields higher average revenues and acceptance ratios.

Meanwhile the work in [22] proposes two other VN embedding schemes based upon a node ranking approach, i.e., two-stage and single-stage. Both methods use a Markov random walk model to assign ranks to substrate and VN nodes. Namely, a node's rank is computed based upon its capacity, its adjacent links, and the ranks of its neighboring nodes. Now the two-stage algorithm sorts VN and substrate nodes in descending order of rank and then maps the VN node with the highest rank to the substrate node with the highest rank, and so on. Next, VN links are embedded using either a *multi-commodity flow* (MCF) algorithm (if path splitting is allowed) or a k-shortest path algorithm (if path splitting is not allowed). Meanwhile, the single-stage algorithm constructs a breadth-first tree of virtual nodes in descending order of their ranks. Substrate nodes are also sorted in descending order of their rank, and the one with the shortest path to the parent VN node is selected. Results show that the proposed algorithms yield higher acceptance rates and revenues as compared to some baseline VN embedding schemes.

Finally, a path splitting and migration algorithm is also proposed in [23] for batch embedding of VN demands. Again, all the VN requests in a time window are sorted in descending order of revenue for mapping. Any request which cannot be mapped in a given time window is returned to the request queue, and requests are rejected if they exceed a pre-defined wait delay. The algorithm basically embeds all VN nodes across all requests first and then tries to embed all VN links with successfully-mapped endpoints. Furthermore, VN nodes are also embedded onto substrate nodes with the maximum capacity. Meanwhile, VN links are either embedded using MCF or k-shortest path techniques. Results show that the proposed scheme gives higher revenues when path splitting is allowed.

2.2 Survivable VNE: Single Link Failure

A range of VN survivability schemes have also been developed to allow cloud-based services to recover from substrate network and datacenter site failures. For example, most providers reserve backup resources within a datacenter and then use fast recovery techniques to perform rapid switchovers during server/storage outages [1], i.e., commonly over specialized intra-datacenter topologies such as fat-tree, Dcell, Bcube, etc. Nevertheless, these methods are mostly ineffective against large-scale disruptions affecting complete datacenter sites or underlying network substrate nodes/links. Hence further *network-based* survivable VNE schemes have also been proposed to pre-compute and provision backup (protection) VN node/VN link resources. Some of these efforts are now reviewed.

Earlier studies have looked at the case of single substrate link failures, see Figure 2.1. For example, [8] presents a *survivable virtual network embedding* (SVNE) heuristic that pre-partitions link bandwidth into two components, i.e., working and (post-fault) protection. A set of detour routes are also pre-computed for each physical link using a k-shortest path algorithm. The scheme starts by computing VN node mappings and then maps VN links by solving a MCF problem. Substrate link failures are then handled by re-directing embedded

VN links onto the pre-computed link detours. Although this solution cannot guarantee recovery for all failed flows, it still outperforms a baseline scheme that re-computes new VN mappings for affected demands (in terms of blocking and revenue). The authors also extend their results in [24] by developing a MIP optimization model to provide survivability against single link failures, i.e., by setting up link-disjoint primary and backup flows for each VN link connection.

Meanwhile [25] presents another single link failure recovery scheme using path splitting. First, fixed bypass paths are pre-computed for each link to help re-direct affected flows. Next, two heuristic schemes are introduced, i.e., shared on-demand and shared pre-allocation. The former re-uses existing VNE schemes to map VN nodes and then solves an optimization program to route VN links. Resource sharing is also implemented between protection flows on bypass paths (owing to the single link failure assumption). Conversely, the latter pre-allocates backup link resources before processing requests and uses an optimization scheme to pre-partition link capacity between working/backup flows. Results show lower blocking and higher revenues for both schemes (versus some alternate non-sharing strategies).

Finally, a backup node migration scheme is also proposed in [26] for single substrate link failures. Namely, failed VN links are recovered by migrating one of their endpoint nodes to a backup node. Hence this approach induces multiple VN node migrations, but the number of backup nodes may be limited. The algorithm also computes an initial VN mapping using any regular VNE scheme, and then augments it by computing backup mappings for each substrate link failure. Findings indicate lower redundant link costs and higher acceptance rates versus link-disjoint VN protection (albeit VN node migration costs are higher).

2.3 Survivable VNE: Single Node Failure

The above-detailed link-based schemes generally ignore substrate node failures, an essential requirement for any disaster recovery scheme. These faults are much more challenging than substrate link failures, as they can affect whole datacenters and their associated network switching nodes. As such, substrate node failures will likely impact many more VN demands. In response, various studies have also looked at backup VN node provisioning. Note that these strategies can be further coupled with higher-level data replication methods to improve service reliability and recovery times [1]. Some of these contributions are now surveyed here.

The authors in [9] study VN protection for single node failures. Specifically, a critical subset of VN nodes is defined for each request, and two protection strategies are proposed, i.e., 1-redundant and k-redundant. The former computes one backup node for all critical VN nodes, whereas the latter computes k backup nodes (and their backup VN links). Resource sharing is also done between backup paths for different VN nodes. Although the problem is initially specified using a MILP formulation, this method poses high intractability and hence a heuristic solution is proposed instead. Namely, the working VN mapping is first computed using a regular VNE heuristic followed by the backup VN mapping using a simplified MILP model. Results show that the k-redundant scheme is more resource efficient, particularly if substrate link costs are higher than node costs.

Meanwhile, [27] presents another (VN) facility node protection scheme that provisions a single backup VN node. However, unlike [9], unaffected nodes can also be remapped to the backup node in order to improve recovery. Namely, an intermediate grid graph is constructed and VN node/link remappings are computed for all possible VN node failures (using graph transformation/decomposition and bipartite matching techniques). Resource sharing is also done to reduce resource usage across these mappings. Detailed simulation results show that the proposed scheme is very efficient and gives lower blocking than some other algorithms, albeit the number of post-fault node migrations is higher (increased operational complexity). The authors also improve upon their initial work in [28] by proposing a single-node backup scheme with two migration strategies, i.e., only migrating the affected node or remapping the whole VN graph. The latter approach consumes less backup resources but results in more node migrations. Furthermore, [29] determines the "proper" number of backup nodes to provision (along with their adjacent links) by using VN node clustering techniques. Link bandwidth and backup resource sharing is also done by placing backup nodes inbetween paths connecting primary VN nodes. Finally, [30] provisions a backup node that is connected to all other VN nodes in the request. The modified VN demand is then mapped using a Tabu search metaheuristic. The results show near optimal performance for the Tabu search algorithm in terms of resource efficiency.

2.4 Survivable VNE: Multiple Failures

In general, single node/link failure schemes are quite effective for handling random isolated outage conditions. However, larger "disaster-type" events are much more challenging and can yield multiple *regional* (spatial) failures with high levels of temporal correlation [7]. For example, a powerful earthquake can damage multiple datacenter sites and substrate nodes/fibers routes, rendering single fault recovery schemes ineffective. Moreover, these events can also affect critical support infrastructures, e.g., such as electric utility grids, further worsening infrastructure node and link outages. Hence recent studies have also looked at more robust VN disaster recovery schemes for multiple node and link failures. Some of these contributions are now reviewed further.

A "topology-aware" multi-failure VN recovery approach is proposed in [31]. The scheme defines a candidate backup node set for each substrate node from which the recoverability of the substrate node is calculated. Namely, a portion of each substrate node and link is reserved for backup purposes prior to any VN embedding. Critical VN nodes are then mapped onto substrate nodes with higher recoverability. The results show improvements in operating profits as compared to some other existing survivable VNE approaches. Meanwhile, [32] details a survivable VN embedding scheme using an artificial bee colony metaheuristic

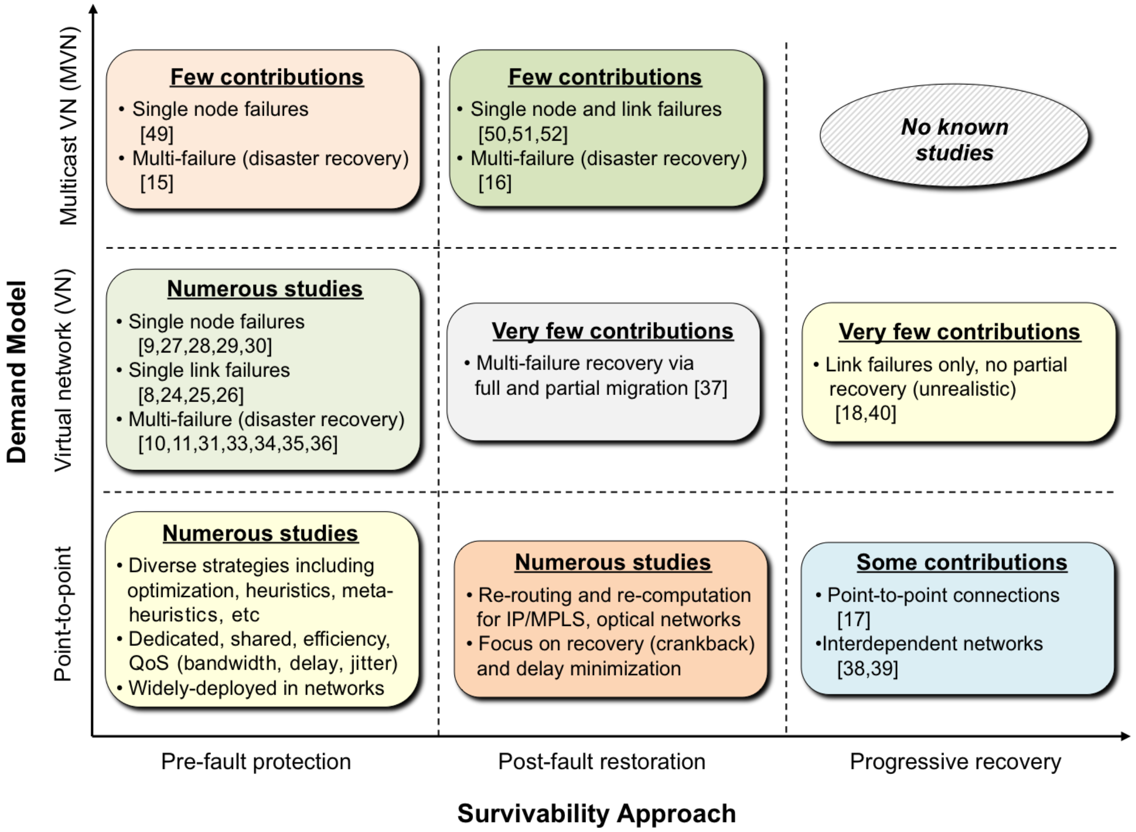

Demand Model
Multicast VN (MVN)
Virtual network (VN)
Point-to-point
Few contributions
• Single node failures [49]
• Multi-failure (disaster recovery) [15]
Few contributions
• Single node and link failures [50,51,52]
• Multi-failure (disaster recovery) [16]
No known studies
Numerous studies
• Single node failures [9,27,28,29,30]
• Single link failures [8,24,25,26]
• Multi-failure (disaster recovery) [10,11,31,33,34,35,36]
Very few contributions
• Multi-failure recovery via full and partial migration [37]
Very few contributions
• Link failures only, no partial recovery (unrealistic) [18,40]
Numerous studies
• Diverse strategies including optimization, heuristics, meta-heuristics, etc
• Dedicated, shared, efficiency, QoS (bandwidth, delay, jitter)
• Widely-deployed in networks
Numerous studies
• Re-routing and re-computation for IP/MPLS, optical networks
• Focus on recovery (crankback) and delay minimization
Some contributions
• Point-to-point connections [17]
•Interdependent networks [38,39]
Pre-fault protection
Post-fault restoration
Progressive recovery
Survivability Approach

Figure 2.1: Existing virtual network survivability schemes: single, multiple failures

approach. Specifically, VN mapping is done based upon the reliability and residual resource levels of physical nodes and links, i.e., where reliability is defined in terms of node and link age (both of which decrease over time). The results show decreased numbers of affected clients after failures versus some other related strategies.

Meanwhile, several other studies on regional failure recovery assume a pre-defined set of disaster risk "regions", or *shared risk groups* (SRG) [10],[11]. Here each region corresponds to a potential outage event and is characterized by a sub-graph of vulnerable nodes/links. Note that these actual regions can be identified using off-line risk analyses based upon geographic, meteorological, or geo-strategic models (out of scope herein). For example, the work in [10] presents a MILP formulation to compute multiple backup VN mappings for a request, i.e., to overcome all potential regional failures. Given the single outage assumption, resource sharing is also done between backup VN mappings to improve resource efficiency. However the MILP model is not solved and two heuristic schemes are presented instead. Namely, the *separate optimization with unconstrained mapping* (SOUM) approach computes backup mappings for each failure region and applies resource sharing between them. However, this strategy is very resource-intensive and hence an *incremental optimization with constrained mapping* (IOCM) algorithm is also developed to iteratively add backup nodes/links to a working mapping. Results indicate that the SOUM scheme gives lower blocking but has higher cost and post-fault VN migration overheads. Meanwhile [11] extends the above work to solve the MILP model using Lagrangian relaxation and decomposition techniques, i.e., by decomposing the original problem into sub-problems equal to the number of risk regions. Tests with a small network indicate that both schemes closely track the full MILP solution in terms of cost. However associated run-times are much lower, and the decomposition scheme gives the best scalability, as expected.

Nevertheless, both the IOCM and SOUM schemes compute backup resources for all disaster regions (SRG). Clearly, this approach is very resource-intensive, especially if the number of regions is large [33]. In general, backup VN nodes should only be mapped outside the risk regions of their working VN nodes to improve recovery. Hence more effective strategies can be designed by using region partitioning methods. Along these lines, [34] presents a MILP formulation to compute risk region-disjoint working/backup VN mappings, i.e., by adding constraints to ensure that VN node mappings fall into two different groups. However due to high complexity, two additional heuristic schemes are also proposed. Namely, the *failure region group-based mapping* (FRGBM) approach partitions risk regions into two groups using a connectivity metric. These groups are then pruned individually from the substrate network, and two separate VN mappings are computed, i.e., working and backup. Again, any regular non-survivable VNE scheme can be used here. Risk region overlaps are also handled by extracting common portions to re-define new regions with higher risks [34]. Overall results show improved recovery performance versus the schemes in [10] and [11].

Nevertheless, static partitioning can give lower resource efficiency under dynamic loads. Hence a further *dynamic failure region disjoint mapping* (DFRDM) heuristic is also proposed in [34]. This approach computes a working VN mapping and then prunes its risk regions to compute a disjoint backup VN mapping. A modified VNE algorithm is then detailed to penalize VN node mappings in new risk regions, i.e., to limit resource usage. Finally, dynamic usage-based node/link weights are used to achieve a proper balance between risk minimization and traffic engineering (load-balancing) objectives. Finally the authors in [35] also present another VN embedding scheme for multiple link failures. Here survivability is gauged by maintaining connectivity between all VN nodes after a failure, i.e., a spanning tree must exist for a VN request in order to survive. A MILP formulation is then proposed for both arbitrary and *shared risk linked group* (SRLG) failures. The results show decreased number of affected demands after multiple physical link failures.

However, the above-detailed multi-failure VN protection schemes assume equi-probable outage events. In general, this is not a realistic assumption and may yield high resource overheads and increased failure rates, i.e., if outage footprints are vastly different from pre-specified risk regions. As a result, *probabilistic* VNE schemes have also been developed by treating disasters as random events with given occurrence probabilities and conditional node/link failure rates. For example, [36] studies multiple independent datacenter (facility node) failures and presents an algorithm to protect a subset of critical VN nodes. Here a VN request is augmented with several shared backup nodes (and VN links) to protect the critical nodes. Opportunistic sharing is also done to lower backup resource usages between VN requests. The resultant problem is then solved using a MCF formulation to map all VN links. Findings confirm that resource sharing greatly reduces VN blocking rates. However this strategy does not consider network substrate link failures, a key concern in large-scale disaster conditions.

Meanwhile [33] defines a more realistic failure model for probabilistic VN recovery in which outages (stressors) are defined as random events with conditional link failure probabilities. The scheme then maps a single backup node for all VN nodes in a request by using an optimization formulation to minimize “risk”, i.e., defined as the sum of lost capacity across all demands due to disconnection. Results show lower post-fault capacity loss versus other resource-based VN mapping schemes. However this study assumes fixed VN node locations, i.e., and not general embedding, and does not treat network or datacenter node outages.

In general, most multi-failure VN recovery schemes use pre-provisioned protection to handle a-priori failure events, i.e., avoidance approach. As such, these strategies can yield high resource usage and cost, even if backup resource sharing is used [33]. Moreover, pre-provisioning cannot guarantee recovery from all possible randomized failures given the highly-stochastic nature of disaster events in the temporal and spatial domains, e.g., if actual footprints are very different from pre-modeled risk regions. Hence [37] presents one of the first studies on post-fault VN restoration after large-scale outages. This approach re-computes failed VN mappings, and two schemes are proposed based upon full and partial VN migration. Namely, the former establishes new mappings for any affected VN, whereas the latter only remaps the failed VN nodes/VN links in affected demands. Results show better recovery rates with the full migration scheme, albeit overheads are notably higher.

2.5 Infrastructure Repair (Progressive Recovery)

In general, multi-failure VN recovery is very resource intensive, particularly if the number of outage regions is high or the regions themselves are large and overlapping [34]. More importantly, it is very difficult (if not impossible) to accurately predict the exact location and scope of large-scale failure events. Hence protection schemes cannot handle all generalized, random disaster events. In light of this, operators have to consider additional *post-fault* repair strategies to rebuild damaged infrastructures and recover affected services, i.e., possibly over a period of days or weeks. Here it is likely that physical repairs will be done in a *multi-stage* progressive manner owing to constraints on available resources, personnel, budgets, and logistics. Therefore the intelligent placement/scheduling of repair resources will be critical for maximizing post-fault VN service performance, i.e., *progressive recovery.*

Along these lines, one of the first strategies on progressive recovery for point-to-point connections is presented in [17]. This effort formulates the repair problem with k post-fault recovery stages and shows it be *NP*-hard. An optimization model is then defined to maximize the weighted sum of flows across *all* recovery stages, and several heuristics solutions are then developed using (scalable) decomposition methods, i.e., multiple single-stage sub-problems.

Meanwhile, the authors in [38] propose a progressive recovery scheme for multi-layered interdependent networks comprising of electric power grids and communication networks. The work focuses on maximizing system utility and presents an interdependency model to identify cascade dependency relations between entities in the two networks. Furthermore, two sets of failures are considered here, i.e., original failures causing inter-layer cascades and failures resulting from cascades. Different cases are then analyzed subject to various dependencies using both optimization and greedy heuristic techniques. Overall findings show that the latter approach gives near optimal results in almost all cases. However this work assumes unlimited repair resources, which is generally not the case after large-scale disaster events. Additionally, the proposed schemes are only evaluated in terms of the utility of network entities, i.e., no further provisions for service demands or connections.

Similarly, [39] considers progressive recovery for interdependent cyberphysical systems and proposes two recovery models, i.e., a reserved model and an opportunistic model. The former assumes that recovery resources for a device are committed up front, but only assigned during multiple stages. Meanwhile, the latter does not guarantee that all required resources for a device will be assigned. Further *integer linear programming* (ILP) formulations are then presented along with relaxation and rounding techniques to solve different problem cases, and repair ordering is calculated using a backward dynamic programming heuristic. However this effort also overlooks actual network demands (loads), instead focusing on system utilization.

Finally, post-fault infrastructure repair for cloud-based VN services is also a critical concern. Now an initial study on this topic is presented in [18], which details an ILP formulation for post-fault resource scheduling for VN services. However this scheme only treats link failures and not larger datacenter outages. Furthermore, the work assumes that failed links are fully recovered in a given stage, e.g., all wavelength transponders replaced on a link. Finally, the ILP model is only tractable for relatively small network sizes and makes a notable simplification by assuming infinite capacity links (unrealistic case). The authors also extend upon their initial work in [40] by developing a further traveling repairman formulation to find an optimum repair schedule for VN demands. Namely, a MILP formulation is outlined here with the objective of minimizing damage. Furthermore, related heuristic and metaheuristic solutions are also proposed, including a greedy algorithm, dynamic programming and simulated annealing schemes. However, akin to [18], this work does not consider datacenter node failures, a very limiting assumption.

2.6 Multicast Virtual Network Embedding (MVNE)

Traditional multicast network services have been used to support a range of applications such as content delivery, real-time streaming, distributed data backups, etc. These offerings transfer data from a source (root) node to a group of terminal (endpoint) nodes, and typically impose tight QoS delay and delay variation constraints. In particular, the latter parameter is usually defined as the difference between the minimum and maximum source-terminal node delay. Now multicasting is commonly done using *tree-based* network connections and assumes the availability of rapid packet replication features in underlying routers/switches, i.e., to copy and transmit incoming packets onto multiple output links. These one-to-many connections can give much better bandwidth efficiency as opposed to setting up multiple unicast connections, i.e., as the latter approach will likely transmit multiple copies of the data over any shared/common physical links.

In general, multicast networking is a very mature area which has seen notable contributions over the past 30 years, i.e., in terms of multicast connection routing algorithms and distributed multicast routing protocols, see [41]. Here the basic multicast routing problem (with fixed nodes) can be treated as a constrained version of the classical Steiner tree problem in graph theory to find a least-cost tree spanning a set of nodes [42]. The decision version of this problem has been shown as *NP*-complete by a transformation from the exact cover by 3-Sets problem [43]. Some restricted versions of this problem are also shown to be *NP*-complete, e.g., such as unweighted graphs and bipartite graphs [44]. The optimization problem of finding the minimum cost Steiner tree is also shown to be *NP*-hard [44]. As a result, many different graph-based Steiner tree heuristics have also been proposed, and some of the more notable solutions are briefly reviewed here.

Earlier work in [45] proposes a heuristic scheme to compute a Steiner tree in an undirected graph with an upper bound on its cost. Namely, a complete undirected distance graph is first constructed from the original graph. The *minimal spanning tree* (MST) is then computed for this complete graph, and the edges in the tree are replaced by their shortest paths in the original graph. The final Steiner tree is then constructed by removing edges in the minimal spanning tree. The authors also prove that the worst case upper bound on cost for their algorithm is with a factor of 2 from the optimum (i.e., based upon the number of leaves in the minimal Steiner tree).

Meanwhile, the authors in [46] study multicast communication routing and propose a heuristic algorithm to find a delay-constrained Steiner tree. Namely, minimum cost paths are first computed between all pairs of MVN nodes subject to the delay constraint, based upon which a complete graph is constructed. Next, edges are added to a sub-tree of the spanning tree in this complete graph using a greedy approach until all terminals are reached. The findings show near-optimal routes with very low network resource cost. Furthermore, [47] also presents another heuristic to construct a multicast tree with delay and delay variation constraints. Namely, a tree of shortest paths from the source to all terminal nodes is first constructed. If this tree does not meet the delay variation constraint, a further search is executed to build a new tree that satisfies this constraint. This algorithm starts with the terminal node with the longest path from the source and re-constructs the tree by computing k shortest paths from other terminal nodes to the rest of the tree. The path that meets the delay variation constraint is then selected. Overall results also indicate that the scheme can determine the least cost tree among the feasible set of multicast trees.

However, as cloud-based services continue to see broader adoption, many cloud-based organizations are also interested in offering *multicast VN* (MVN) services to support content distribution and live streaming applications for their clients. In particular, MVN demands can be considered as a special case of VN demands which only specify source and terminal nodes (and not any VN links). Furthermore, most MVN demands also impose further delay and delay variation constraints. Hence various MVN embedding studies have also emerged in recent years. For example, [13] proposes a restricted MVN embedding scheme which incorporates delay and delay variation constraints, as well as mapping cost and load-balancing concerns. The work assumes fixed node locations (i.e., no node mapping) and first computes the k-shortest feasible paths from the source to each terminal, subject to the delay constraint. Next, all feasible combinations of paths that meet the delay variation constraint are generated by using a sliding window technique. However, this scheme still treats a multicast service request as series of unicast requests, and hence is not very resource efficient. Meanwhile, another study in [14] defines a waiting time for each MVN request based upon its lifetime. Any request which cannot be embedded right away is re-processed in the idle time between two other MVN requests, as long as its wait time does not expire. Meanwhile, the actual MVN node and link mappings are done based upon the predicted loads at physical nodes. Finally, MVN paths are adjusted to meet delay variation constraints.

The work in [48] also presents an ILP model and heuristic scheme for MVN embedding. This algorithm first identifies a list of eligible substrate nodes to host the MVN source, i.e., subject to available resources. Multicast trees are then constructed from each eligible node using *depth first search* (DFS) and *breath first search* (BFS) strategies subject to delay constraints. An ILP node mapping model is then solved to find the optimal mapping. A dynamic programming approach is also presented to map terminal nodes onto substrate nodes that belong to a given tree with a given source, i.e., such that the tree has minimum cost. Namely, the tree is first divided into multiple sub-trees, each subject to the delay variation constraint. The minimum cost sub-tree that can map all the terminals is then selected. A Tabu search is also proposed to replace the ILP node mapping model. Results show improved blocking rates, revenues, and utilization as compared to some other algorithms.

2.7 Survivable Multicast VN (MVN) Embedding

More recent efforts have also looked at survivable MVN embedding design, albeit only for isolated single failures. For example, the authors in [49] propose a MILP formulation and a survivable MVN embedding heuristic for single node failures. In particular, a constrained node mapping (demand) model is used to restrict MVN node mappings to a subset of physical locations. The minimum number of physical nodes required to cover all MVN nodes is then obtained by applying the min set cover algorithm. Now if there are multiple solutions with equal node costs, then the one with the minimum average path length to the source is selected. Meanwhile, shared backup nodes are also assigned for each MVN node on a backup server in the same substrate node, i.e., intra-datacenter protection. Furthermore, multiple working (and backup) paths are computed from the MVN source node to each terminal node, subject to delay and delay variation constraints. Nevertheless, this approach also computes multiple unicast embeddings and hence is not very resource efficient.

Meanwhile, [50] also studies MVN embedding design for single node failures. Namely, reliability values are defined for each substrate node and used to compute the k most reliable paths for each node pair in the substrate network. Furthermore MVN source and terminal node failures are also treated differently here. For the former, a path convergence algorithm is triggered by the terminal nodes to select a feasible substrate node with the lowest cost as the new source location. For the latter, a hop-by-hop search is initiated from an active MVN node to find the first reachable substrate node with sufficient resources to replace the failed MVN terminal (with preference given to re-using existing MVN links).

An optimization formulation for post-fault MVN recovery from single substrate node/link failures is also presented in [51]. This work also presents a heuristic algorithm for single node failure. Now akin to [50], both MVN source and terminal node failures are considered here. For source failures, a receiver driven path-convergence strategy is used to identify a new source that meets the delay constraint. Meanwhile for terminal failures, a hop-to-hop search is triggered from the MVN source by leveraging assisting MVN terminal nodes with sufficient delay margins. The results show fast restoration after a single node failure. However, this heuristic scheme does not consider link failures. Another study in [52] also looks at single link failures. Namely, a MVN link failure is treated separately based upon whether it connects to an intermediate or a leaf node. In the former case an alternative path is calculated to re-connect the affected terminal(s) to the rest of the tree. Meanwhile in the latter case the leaf node connected through the affected link is migrated. However if the leaf node is also the source node then the tree has to be reconstructed.

Finally, MVN survivability for multiple failures has also received some attention recently. For example, the authors in [15] present a survivable MVN embedding scheme which considers pre-defined failure regions. Here, the initial embedding is computed in a regular manner and then additional failure region-disjoint backup nodes and links are introduced to protect MVN nodes/links mapped onto specific failure regions. Another study in [16] also proposes a MILP model and heuristic method for MVN restoration after multiple node and link failures. Specifically, node ranking and filtering techniques are used to reduce the backup node search process by selecting from intermediate physical nodes hosting the non-failed parts of the affected MVN. Different attributes are also used to rank failed source or terminal nodes here, i.e., such as node connectivity, closeness to the source node, available resources, and node location in the MVN demand. Findings show improved restoration rates and recovery times. Nevertheless this approach resolves MVN demands as multiple point-to-point (unicast) demands, yielding low resource efficiency.

2.8 Open Challenges

Overall, VN service survivability for large-scale outage conditions remains a major concern for network providers. Although this topic has received some notable attention in recent years, many open challenges still remain. Foremost there is a need to develop improved post-fault repair (progressive recovery) frameworks that incorporate a range of realistic conditions, e.g., such as limited repair resources, partial recovery needs, etc. Scalable heuristic and meta-heuristic strategies are also required as many operators prefer such methods in real-world operational environments. Finally, there is also a further need to develop more reliable MVN embedding schemes for multicast cloud services. These methods should incorporate a-priori knowledge of high-risk (outage) regions and achieve a good tradeoff between survivability and resource efficiency. These various challenges form the main motivating factors for this dissertation research.

Chapter 3

Progressive Recovery In Cloud Settings: Optimization Strategies

Progressive recovery design for cloud infrastructure services after large-scale outages is a relatively new and unexplored area. Hence it is important to properly define the problem space first and also develop formalized optimization-based models. Now recently [40] presents an initial ILP model for progressive VN recovery, as reviewed in Section 2.5. However this formulation has some very notable shortcomings. Foremost, the model only considers link failures, and not datacenter node failures. This assumption will clearly not hold in real-world scenarios where disaster-type events can easily affect whole datacenter sites. The model in [40] also assumes that a failed link is fully recovered in its scheduled recovery stage. Again, this is a rather simplistic assumption as generally, only a finite number of incoming link and datacenter repair resources will be available in each stage, e.g., transponder units, switching modules, server racks, storage arrays, etc. Namely, infrastructure recovery will occur in a partial multi-stage manner. Finally, the formulation does not incorporate partial demand recovery, a key shortcoming since most customers will be willing to accept degraded service performance in the aftermath of an outage.

In light of the above, this chapter presents a more realistic and comprehensive modeling of progressive recovery for cloud-based (VN) infrastructure services. First, the overall repair framework is presented. Next the requisite notation is introduced for the optimization formulation. Subsequently, a detailed *mixed integer non-linear programming* (MINLP) formulation is presented with the objective of maximizing the number of recovered VN demands in each stage. However the associated complexity of this formulation makes it very difficult to solve for realistic networks and demand scenarios. As a result, further *simulated annealing* (SA) metaheuristic schemes are also developed and analyzed. These latter strategies can be used to provide improved comparison benchmarks for the subsequent heuristic schemes presented in Chapter 4. Complete details are now presented.

3.1 Progressive Recovery Framework

A high-level overview of the proposed multi-stage progressive recovery framework is shown in Figure 3.1. First, consider a cloud infrastructure in working (non-failure) condition supporting a set of embedded client VN demands. When an initial outage/disaster event occurs, this infrastructure will experience multiple datacenter node and link failures, i.e., Stage 0, Figure 3.1. Here it is reasonable to assume that all links attached to failed/affected datacenter sites and their network routers/switches will also fail. In turn these physical faults will trigger higher-layer client VN demands to fail, either fully or partially, i.e., VN node and VN link outages. In response service providers will usually initiate a multi-stage repair process, as driven by the arrival of finite (limited) amounts of repair resources after arbitrary time intervals. In particular, these resources can include datacenter node components (such as computing racks and storage units) as well as link-level switching and transmission components (such as transponders, switching fabrics, etc). Overall, this repair process will be repeated over multiple stages until the physical infrastructure returns to its original pre-fault operating condition. Now consider the details of each recovery stage.

In general, two key steps must be performed in during recovery, i.e., placement of physical repair resources at damaged nodes/links and restoration of affected (logical) VN demands. Hence two different strategies are considered. Foremost, an idealized single-stage optimization approach is outlined to *jointly* place resources and maximize service recovery (as per desired metrics). Subsequently, a more practical dual-stage approach is also proposed to first schedule (place) incoming repair resources and then restore failed VN demands, see Figure

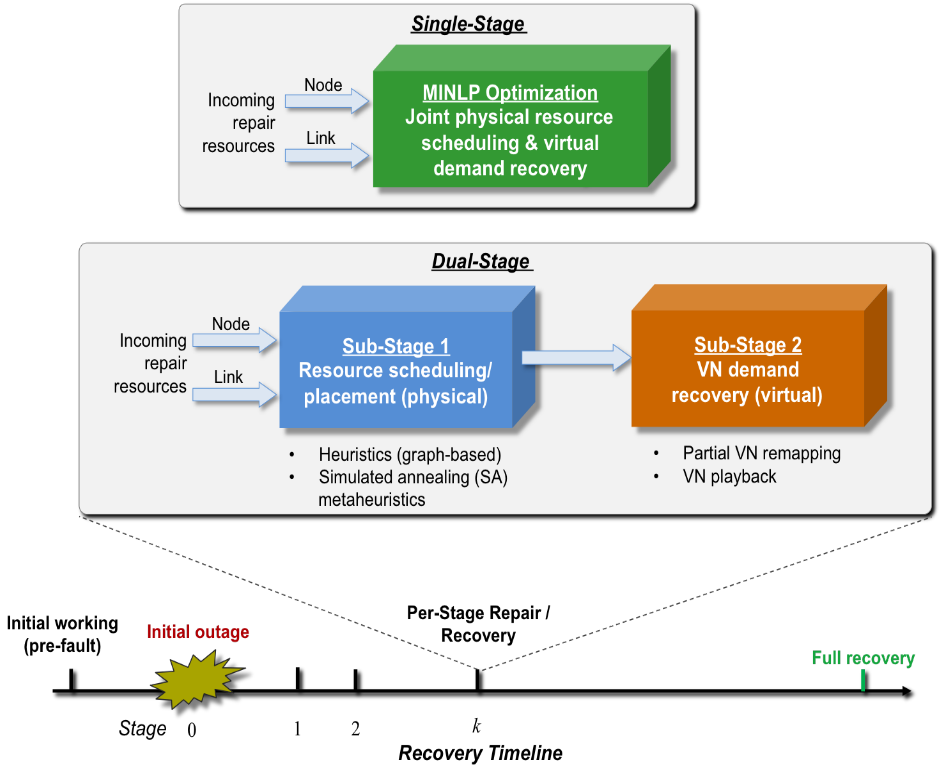

Single-Stage
Incoming repair resources
Node
Link
MINLP Optimization
Joint physical resource scheduling & virtual demand recovery
Dual-Stage
Incoming repair resources
Node
Link
Sub-Stage 1
Resource scheduling/ placement (physical)
Sub-Stage 2
VN demand recovery (virtual)
Heuristics (graph-based)
Simulated annealing (SA) metaheuristics
Partial VN remapping
VN playback
Initial working (pre-fault)
Initial outage
Per-Stage Repair / Recovery
Full recovery
Stage
0
1
2
k
Recovery Timeline

Figure 3.1: Overview of proposed multi-stage progressive recovery framework

3.1. Now carefully note that many existing VNE schemes are already available to handle the latter requirement, i.e., sub-stage 2, Figure 3.1 (see Section 2.1). Hence the focus of this research is on repair resource placement, i.e., sub-stage 1 in Figure 3.1, and various strategies are investigated, including metaheuristic (Section 3.4) and polynomial-time heuristic schemes (Section 4.1).

3.2 Notation Overview

Before presenting the various recovery schemes (single, dual-stage), the requisite notation is first introduced. Namely, a physical cloud substrate is represented by a topology graph, $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$, where $\mathbf{V}_s$ is the set of datacenter nodes and $\mathbf{E}_s$ is the set of network links, as shown in Figure 3.2. Here each node $n \in \mathbf{V}_s$ has a maximum resource capacity of R_n^{max} units, and each physical substrate link $e = (m, n) \in \mathbf{E}_s$ has a maximum bandwidth of $B_{(m,n)}^{max}$ units.

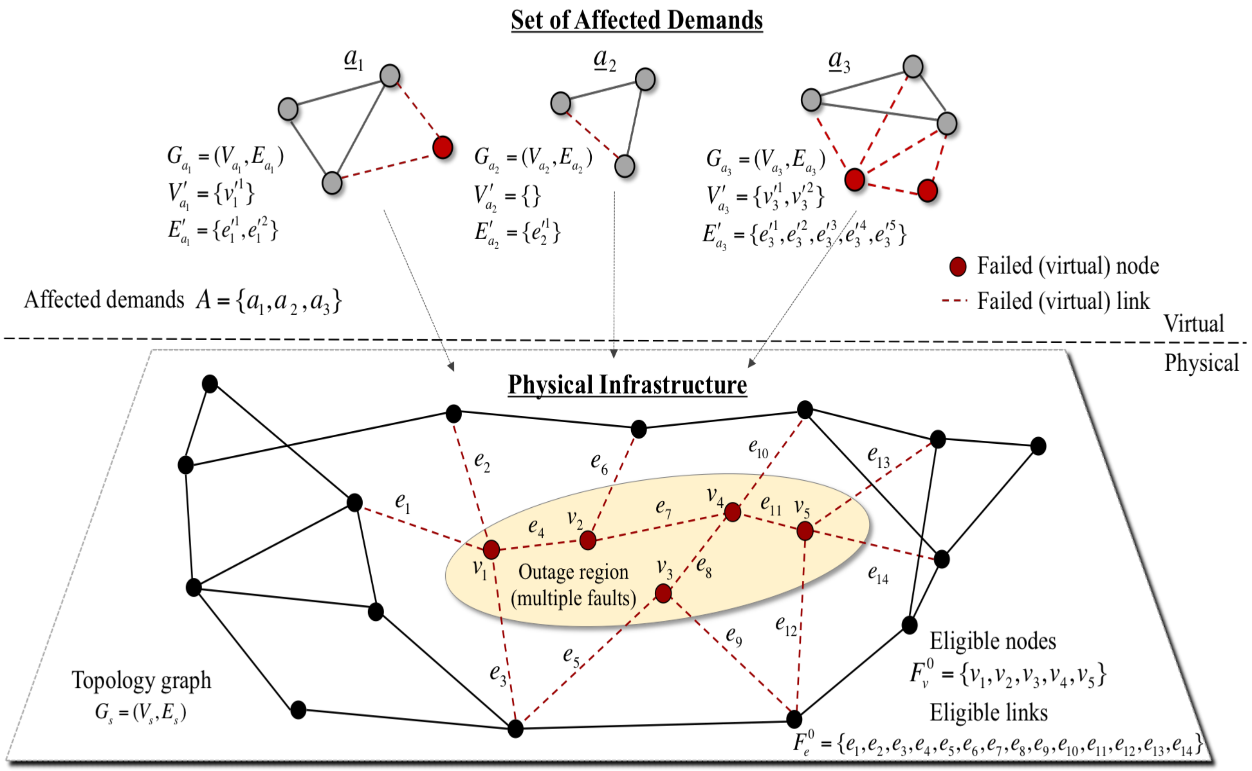

Set of Affected Demands
a_1
a_2
a_3
$G_{a_1} = (V_{a_1}, E_{a_1})$
$V'_{a_1} = \{v_1'^1\}$
$E'_{a_1} = \{e_1'^1, e_1'^2\}$
$G_{a_2} = (V_{a_2}, E_{a_2})$
$V'_{a_2} = \{\}$
$E'_{a_2} = \{e_2'^1\}$
$G_{a_3} = (V_{a_3}, E_{a_3})$
$V'_{a_3} = \{v_3'^1, v_3'^2\}$
$E'_{a_3} = \{e_3'^1, e_3'^2, e_3'^3, e_3'^4, e_3'^5\}$
Failed (virtual) node
Failed (virtual) link
Affected demands $A = \{a_1, a_2, a_3\}$
Virtual
Physical
Physical Infrastructure
e_1
e_2
e_3
e_4
e_5
e_6
e_7
e_8
e_9
e_{10}
e_{11}
e_{12}
e_{13}
e_{14}
v_1
v_2
v_3
v_4
v_5
Outage region
(multiple faults)
Topology graph
$G_s = (V_s, E_s)$
Eligible nodes
$F_v^0 = \{v_1, v_2, v_3, v_4, v_5\}$
Eligible links
$F_e^0 = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}$

Figure 3.2: Overview of notation for VN embedding

Meanwhile a VN request (IaaS demand) is denoted by an abstract graph, $\mathbf{G}_a = (\mathbf{V}_a, \mathbf{E}_a)$, where $\mathbf{V}_a$ is the set of VN nodes and $\mathbf{E}_a$ is the set of VN links. Without loss of generality, it is also assumed that each VN node $p \in \mathbf{V}_a$ requires $r(a)$ in node resources, and each VN link $(p, q) \in \mathbf{E}_a$ requires $b(a)$ in bandwidth capacity.

Meanwhile, assume that a large-scale outage event occurs at time T_0 and affects a subgraph of $\mathbf{G}_s$, as shown in Figure 3.2. This initial damage is followed by multiple sequential recovery stages at arbitrary times, T_k, k=$1, 2, ...$ $(T_{k+1} > T_k)$, as shown in Figure 3.3. Now it is assumed that all affected physical nodes/links fail completely, and their available/working resource levels drop to zero. These levels then start to rise back up as the repair stages progress, achieving full recovery by the final stage, i.e., R_n^{max} or $B_{(m,n)}^{max}$ units, Figure 3.3. Therefore affected nodes and links transit through several states, i.e., fully-failed to partially-recovered to fully-recovered. Based upon this transition process, only failed or partially-recovered entities in stage k are *eligible* to receive incoming resources, and these are denoted by the sets $\mathbf{F}_v^k$ (eligible nodes) and $\mathbf{F}_e^k$ (eligible links). Specifically, $\mathbf{F}_v^0$ ($\mathbf{F}_e^0$) represents the initial set of failed nodes (links) and $\mathbf{F}_v^k \subseteq \mathbf{F}_v^{k-1}$ ($\mathbf{F}_e^k \subseteq \mathbf{F}_e^{k-1}$), see example in Figure 3.2 with 5 failed nodes and 14 failed links. Hence the aggregate datacenter node resource loss across the whole network is given by:

$$X_0 = \sum_{n \in \mathbf{F}_v^0} R_n^{max} \tag{3.1}$$

as shown in Figure 3.3. Similarly, the aggregate network link capacity loss is also given by:

$$Y_0 = \sum_{(m,n) \in \mathbf{F}_e^0} B_{(m,n)}^{max} \tag{3.2}$$

Now as mentioned before, failures in the substrate network will cause failures in embedded VN demands. Hence the set of all affected VN demands is further denoted by $\mathbf{A}$, e.g., 3 failed demands in Figure 3.2, $\mathbf{A} = \{a_1, a_2, a_3\}$. Furthermore, let $\mathbf{V}_a^{'}$ and $\mathbf{E}_a^{'}$ represent the set of affected virtual nodes and virtual links in VN demand $a \in \mathbf{A}$, respectively. Namely, all VN nodes in $\mathbf{V}_a^{'}$ and all VN links in $\mathbf{E}_a^{'}$ fail at time T_0 (Stage 0). Finally, it is assumed that a total of X_k datacenter repair resources (computing racks, storage disks, etc) and Y_k link repair resources (optical link transponders, switching units) arrive in stage k. Here all resources are assigned at the integral granularity level and the *current* and *available* resource levels at node n in stage k are also denoted by R_n^k and $R_n^{'k}$, respectively. Similarly, the current and available resource levels of link (m,n) in stage k are also given by $B_{(m,n)}^k$ and $B_{(m,n)}^{'k}$, respectively. In particular, the former quantity represents the amount of *physical* resources in place at a given link or node. Meanwhile, the latter quantity represents the portion of the current resources which have not been assigned to any user demands. Hence the available resource level is always less than or equal to the current resource level, i.e., $R_n^{'k} \leq R_n^k$ and $B_{(m,n)}^{'k} \leq B_{(m,n)}^k$.

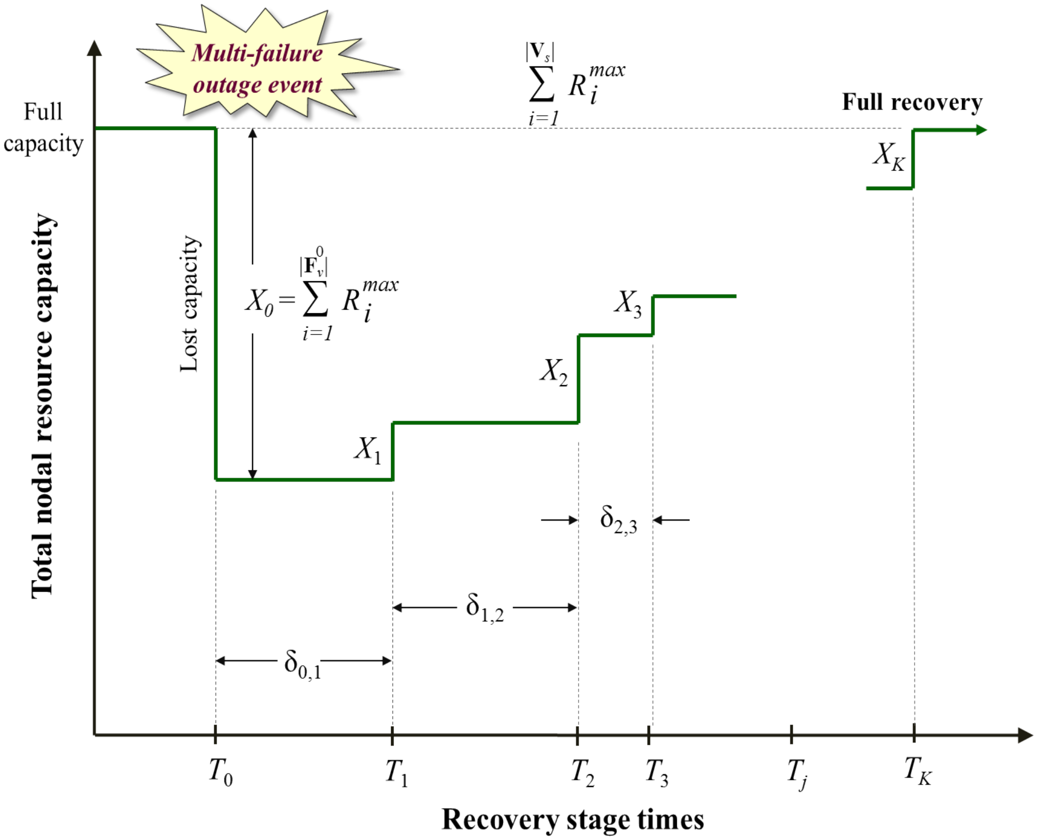


Figure 3.3: Progressive recovery stages (aggregate datacenter node resource)

3.3 Optimization Formulation

An idealized optimization formulation is now presented to jointly optimize the placement of the physical repair resources and recover logical VN demands, as per single-stage approach in Figure 3.1. Now with regards to the latter, two VN restoration approaches are considered here, i.e., remapping of failed VN nodes/VN links, i.e., *VN remapping* and restoration of failed VN nodes/VN links to their pre-fault mapping, i.e., *VN playback*. Hence at the end of each stage, all of the above-detailed optimization parameters are updated for use in the subsequent stage. Furthermore without loss of generality, it is assumed that the repair process is terminated when the original substrate topology is restored. In practice, however, infrastructure providers may choose to re-design and modify their physical substrate plants in response to widespread damage, i.e., to improve future resiliency. However this broader topic is not considered here in order to simplify the work and allow proper evaluation of the repair scheduling schemes, i.e., recovering original demands over the same topology. Carefully note that additional damage to other critical infrastructures may also prevent arbitrary placement of repair resources, i.e., road transportation networks, electric utility grids, etc. Again these contingencies are not considered here in order to bound the complexity of the problem.

A novel optimization formulation is now presented for progressive VN recovery using a MINLP approach. First consider some requisite variable definitions as follows:

- $\beta^{k}_{(m,n)}$: Amount of link bandwidth repair resources allocated to physical link (m, n) in stage k
- α^{k}_{n}: Amount of node repair resources allocated to physical node n in stage k
- $\rho^{p,a,k}_{n}$: Binary variable to indicate if VN node p from virtual request $a \in \mathbf{A}$ is restored (remapped) onto physical node n in stage k
- $f^{(p,q),a,k}_{(m,n)}$: Binary variable to indicate if a part of VN link (p, q) is restored (remapped) onto physical link (m, n) in stage k
- $\tau^{a,k}$: Binary variable to indicate if the VN request $a \in \mathbf{A}$ is restored in stage k

Based upon the above, overall objective function is defined as:

$$max \sum_{\forall a \in \mathbf{A}} \tau^{a,k} \tag{3.3}$$

subject to the following constraints:

$$\rho^{p,a,k}_{n} \in \{0, 1\}, \quad \forall n \in \mathbf{V}_s, \forall p \in \mathbf{V}^{'}_{a}, \forall a \in \mathbf{A} \tag{3.4}$$

$$\tau^{a,k} \in \{0, 1\}, \quad \forall a \in \mathbf{A} \tag{3.5}$$

$$f_{(m,n)}^{(p,q),a,k} \in \{0,1\}, \quad \forall (m,n) \in \mathbf{E}_s, \forall (p,q) \in \mathbf{E}_a^{'}, \forall a \in \mathbf{A} \tag{3.6}$$

$$\sum_{n \in \mathbf{F}_v^k} \alpha_n^k \leq X_k \tag{3.7}$$

$$\sum_{(m,n) \in F_e^k} \beta_{(m,n)}^k \leq Y_k \tag{3.8}$$

$$R_n^k + \alpha_n^k \leq R_n^{max}, \quad \forall n \in \mathbf{V}_s \tag{3.9}$$

$$B_{(m,n)}^k + \beta_{(m,n)}^k \leq B_{(m,n)}^{max}, \quad \forall (m,n) \in \mathbf{E}_s \tag{3.10}$$

$$\sum_{a \in \mathbf{A}} \sum_{p \in \mathbf{V}_a'} r(a) * \rho_n^{p,a,k} \leq R_n'^k + \alpha_n^k, \quad \forall n \in \mathbf{V}_s \tag{3.11}$$

$$\sum_{a \in \mathbf{A}} \sum_{(p,q) \in \mathbf{E}_a'} b(a) * (f_{(m,n)}^{(p,q),a,k} + f_{(n,m)}^{(q,p),a,k}) \leq B_{(m,n)}'^k + \beta_{(m,n)}^k, \quad \forall (m,n) \in \mathbf{E}_s \tag{3.12}$$

$$\beta_{(m,n)}^k = \beta_{(n,m)}^k, \quad \forall (m,n) \in \mathbf{E}_s \tag{3.13}$$

$$f_{(m,n)}^{(p,q),a,k} = f_{(n,m)}^{(q,p),a,k}, \quad \forall (m,n) \in \mathbf{E}_s, \forall (p,q) \in \mathbf{E}_a^{'}, \forall a \in \mathbf{A} \tag{3.14}$$

$$\prod_{p \in \mathbf{V}_a'} \sum_{n \in \mathbf{V}_s} \rho_n^{p,a,k} = \tau^{a,k}, \quad \forall p \in \mathbf{V}_a^{'}, \forall a \in \mathbf{A} \tag{3.15}$$

$$\sum_{n \in V_s} f_{(m,n)}^{(p,q),a,k} - \sum_{n \in \mathbf{V}_s} f_{(n,m)}^{(p,q),a,k} - \rho_m^{p,a,k} + \rho_m^{q,a,k} = 0, \quad \forall m \in \mathbf{V}_s, \forall (p,q) \in \mathbf{V}_a^{'}, \forall a \in \mathbf{A} \tag{3.16}$$

Furthermore, some more specialized variables definitions and constraints are also required for the earlier-defined VN remapping and VN playback recovery strategies. In particular, the following constraints are specified for the VN remapping case:

$$\sum_{n \in \mathbf{V}_s} \rho_n^{p,a,k} = \tau^{a,k}, \quad \forall p \in \mathbf{V}_a^{'}, \forall a \in \mathbf{A} \tag{3.17}$$

$$\sum_{p \in \mathbf{V}_a'} \rho_n^{p,a,k} \leq 1, \quad \forall n \in \mathbf{V}_s, \forall a \in \mathbf{A} \tag{3.18}$$

Meanwhile, for the case of playback recovery two additional binary input parameters are introduced in order to track *pre-fault* VN mapping allocations, i.e., $\eta^{(p,q)}_{(m,n),a}$ and $\gamma^{p,a}_{n}$. Namely, value of unity for $\eta^{(p,q)}_{(m,n),a}$ implies that the pre-fault connection path assigned to VN link (p,q) in VN request a uses physical link (m,n). Similarly, a value of unity for $\gamma^{p,a}_{n}$ implies that VN node p from VN request a is mapped onto physical node n prior to the outage event. Based upon the above, the following constraints are specified:

$$\eta^{(p,q)}_{(m,n),a} \leq f^{(p,q),a,k}_{(m,n)}, \quad \forall (p,q) \in \mathbf{E}^{'}_{a}, \forall a \in \mathbf{A} \tag{3.19}$$

$$\gamma^{p,a}_{n} \leq \rho^{p,a,k}_{n}, \quad \forall p \in \mathbf{V}^{'}_{a}, \forall a \in \mathbf{A} \tag{3.20}$$

Overall, the objective function in Eq. 3.3 tries to maximize the number of restored VN demands in each stage and as noted earlier, a VN request is considered restored if all its VN nodes and VN links are in a working state. Also note that the above formulation is classified as a MINLP formulation owing to the product term in Eq. 3.15.

Overall, Eqs. 3.4 - 3.16 are common to both VN restoration approaches. Namely, Eqs. 3.4 - 3.6 denote the required binary constraints on $\rho^{k}_{p,a}$, $\tau^{a,k}$ and $f^{(p,q),a,k}_{(m,n)}$, respectively. Also, Eqs. 3.7 and 3.8 ensure proper bounds on node and link repair resources at each stage. Next, Eqs. 3.9 and 3.10 limit the amount of repair resources assigned to damaged physical nodes and links to the amounts needed to reach their maximum pre-fault capacity levels. Meanwhile, Eqs. 3.11 and 3.12 bound physical node and link capacities, respectively. Furthermore, Eq. 3.13 ensures that bandwidth resources assigned to a bi-directional physical link are the same in both directions. Similarly, Eq. 3.14 guarantees that flows are equal in both directions of a physical link. Also, Eqs. 3.15 and 3.16 implement flow conservation. Meanwhile, Eqs. 3.17 and 3.18 are only applicable to the VN remapping approach. Namely, Eq. 3.17 ensures that only one physical node is selected for a VN node if its VN request is restored. Also, Eq. 3.18 ensures that a physical node can only be allocated to one VN node in a given VN request.

Clearly, these constraints are not required for the VN playback approach since pre-fault VN embeddings are used (for which these constraints have already been met). Similarly, Eqs. 3.19 and 3.20 are only applicable to the VN playback restoration approach, i.e., to ensure that failed virtual nodes and links are only restored to their previously-assigned physical nodes and links, respectively.

Overall, the non-linear nature and scale of the above optimization formulation imposes some very serious computational limitations. For example, consider a small physical network with 10 physical nodes and 15 physical links. Furthermore assume that this substrate hosts a moderate number of VN demands, approximately 100 in total, with each averaging 4 VN nodes and 6 VN links. In the worst case, all 100 demands can be affected after a failure, possibly even failing completely. Hence the total number of variables in the MINLP optimization formulation in this case is approximately 13,000 variables, i.e., $10{\cdot}4{\cdot}100$ $\rho_n^{p,a,k}$ variables, $6{\cdot}15{\cdot}10$ $f_{(m,n)}^{(p,q),a,k}$ variables, in addition to α_n^k, $\beta_{(m,n)}^k$ and $\tau^{a,k}$ variables (which are generally fewer in this particular case). Similarly, the total number of constraints is almost 30,000. Clearly, these numbers impose very high computational burdens given the non-linear nature of the formulation. In addition, the fact that the optimization model performs *joint* physical and virtual recovery adds to its complexity. As a result, further optimization *approximation* strategies will have to be developed in order to find practical working solutions to this MINLP problem, e.g., via techniques such as relaxation (to linear programming), decomposition, iteration, or combinations thereof. However, even these modified optimizations themselves will be sub-optimal. In light of these realities, more realistic and computationally-feasible metaheuristic strategies are proposed here, as detailed next.

3.4 Simulated Annealing (SA) Metaheuristics

A more practical solution to the progressive recovery problem is now presented based upon the dual-stage strategy in Figure 3.1. Namely, *simulated annealing* (SA) metaheuristics are developed for resource placement, i.e., sub-stage 1 in Figure 3.1. Meanwhile, as noted in Section 3.1 any existing VNE algorithm can be used for VN demand recovery, i.e., sub-stage 2, Figure 3.1.

An example of the dual-stage progressive recovery scheme is also shown in Figure 3.4, for a small physical infrastructure carrying a single 3-node/3-link embedded VN request experiencing an outage event. Here, the numbers next to the physical links represent the available bandwidth levels, whereas the numbers next to the VN links represent the requested bandwidth levels (virtual node demands are not shown in order to simplify the presentation). Specifically, assume that the VN is already mapped onto the physical network prior to the outage event (Figure 3.4a). Now the multi-failure event takes down multiple infrastructure nodes/links in the physical network, resulting in virtual link failure in the mapped virtual request (Figure 3.4b). Subsequently, both the physical and virtual networks are left in partially-working condition (Figure 3.4c). Now the first recovery stage is triggered by the arrival of repair resources. As illustrated, progressive recovery is done in two sub-stages, i.e., physical recovery and virtual recovery. Namely, in the former stage, the repair resources are placed in the network (see Figure 3.4d). In particular, this example assumes that 200 units of link repair resources are available and equally distributed among the two failed physical links (shown in green). Also sufficient node repair resources are also assumed to fully recover the failed physical node. Next, the VN recovery sub-stage is executed to remap the affected VN links/VN nodes onto the repaired physical infrastructure (Figure 3.4e).

Now consider the design of SA resource placement strategies. Overall, this metaheuristic approach is widely-used to generate approximate solutions for large combinatorial problems. SA basically tries to compute a “global” optimum for a given system objective function, usually (but not necessarily) defined over a discrete space. The method works by modeling the initial system state at a high “temperature” and then running an iterative “cooling process” to find improved solutions. Namely, at each iteration an alternate solution (neigh-

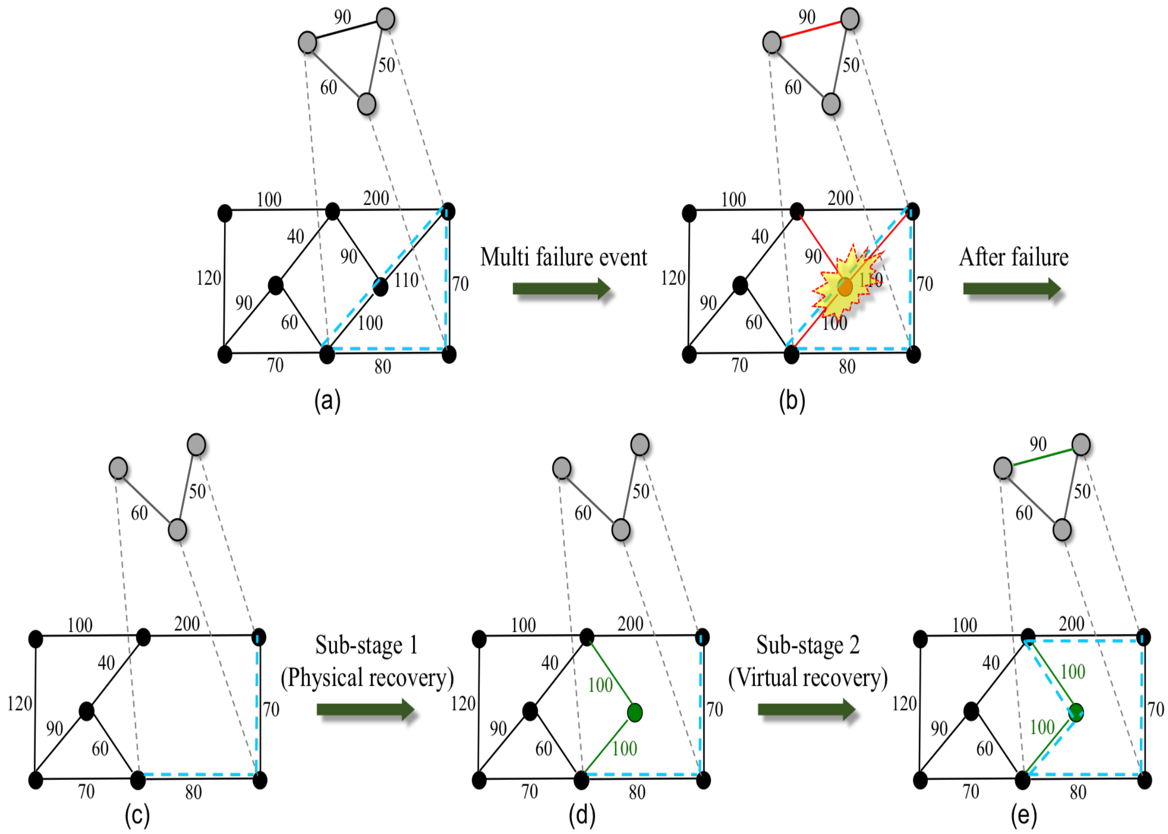


Figure 3.4: An example of dual-stage progressive recovery scheme

boring state) is generated by slightly modifying or permuting the current state in a given manner, i.e., termed random move generation. If this new state yields an improved value for the system objective function, then the current system state is appropriately updated. Conversely, if the neighboring state does not yield any gains, it may still replace the current state in a probabilistic manner. Also, the probability of accepting a new solution is higher at increased temperatures and gradually drops as the temperature decreases. These features allow SA to achieve more variability in its search process, thereby reducing the chances of finding a local optimum.

Now consider the SA search process in more detail. Although SA performance is strongly dependent upon the initial temperature, there is no general rule for setting this value. However, in general, the initial temperature should be high enough to accept most feasible solutions in the initial steps. Overall, the acceptance probability is defined as:

$$prob = \begin{cases} exp(\frac{-\Delta r}{temp}) & \text{if } \Delta r \geq 0 \\ 1 & \text{if } \Delta r < 0 \end{cases} \tag{3.21}$$

where *temp* is the temperature and Δr is the difference in the objective value between current and new state. The search process is finally terminated when the system objective function reaches an acceptable value or the system has cooled sufficiently, i.e., the temperature reaches a fixed threshold. Now two different SA adaptations are proposed for the progressive recovery problem here. The first *selective* approach explicitly selects the nodes and links which will receive incoming repair resources, whereas the second *distributed* approach computes proportional weights for nodes and links for repair resource distribution. However both strategies still use the same system objective function defined in the optimization model (Eq. 3.3). Furthermore the initial values for the system temperature and SA cooling rate, α, are also obtained via experimentation. These two SA techniques are now presented for the repair resource placement problem, i.e., selective and distributed.

3.4.1 Selective SA (S-SA) Scheme

This approach considers a binary search space consisting of all possible combinations of the affected nodes and links, and its psudeocode is shown in Figure 3.5. Namely, the current system state is represented by two binary vectors, one for the node solution, $\mathbf{V}^s_{cur}$, and one for the link solution, $\mathbf{E}^s_{cur}$. Here, $\mathbf{V}^s_{cur}[i]$ is associated with the i-th affected physical node in $\mathbf{F}^k_v$. Similarly, $\mathbf{E}^s_{cur}[i]$ represents the i-th affected physical link in $\mathbf{F}^k_e$. Now $\mathbf{V}^s_{cur}[i] = 1$ ($\mathbf{E}^s_{cur}[i] = 1$) implies that the corresponding node (link) is included in the current solution and will receive resources, whereas $\mathbf{V}^s_{cur}[i] = 0$ ($\mathbf{E}^s_{cur}[i] = 0$) implies otherwise.

The scheme starts by generating a random binary node vector for initial node resource assignments, i.e., randomly assigns values of 0 or 1 to the $\mathbf{V}^s_{cur}$ entries. Carefully note that the total amount of resources required by all nodes with value of 1 in $\mathbf{V}^s_{cur}$ cannot exceed the available node repair resources, i.e., the following condition must hold:

$$\sum_{i=0}^{|\mathbf{F}^k_v|-1} \mathbf{V}^s_{cur}[i](R^{max}_i - R^k_i) \leq X_k \tag{3.22}$$

Hence the above condition is checked for each entry in $\mathbf{V}^s_{cur}$, and if the chosen node i violates it, then $\mathbf{V}^s_{cur}[i]$ is set to 0.

Next the scheme proceeds to generate a random binary link vector as well. Namely, leveraging the above node distribution vector and the current network state, the candidate link set is defined as the subset of links with non-failed endpoint nodes or endpoint nodes included in the current node solution, i.e., assigned a value of 1 in $\mathbf{V}^s_{cur}$. $\mathbf{E}^s_{cur}[i]$ is also set to 0 for each non-candidate link i, i.e., those links which are not eligible to receive resources. An initial binary link solution is then generated by randomly assigning a value of 1 to the remaining entries in $\mathbf{E}^s_{cur}$, i.e., entries associated with the candidate links. Similarly, the following condition must hold:

$$\sum_{i=0}^{|\mathbf{F}^k_e|-1} \mathbf{E}^s_{cur}[i](B^{max}_i - B^k_i) \leq Y_k \tag{3.23}$$

Again, $\mathbf{E}^s_{cur}[i]$ is set to 0 if selecting link i violates the above condition. The value of the objective function is then calculated for the current state solution.

Subsequently, the vectors $\mathbf{V}^s_{cur}$ or $\mathbf{E}^s_{cur}$ are randomly selected to generate a new state in each iteration. Namely, the neighboring state (random move) is derived by swapping two random entities in $\mathbf{V}^s_{cur}$ or $\mathbf{E}^s_{cur}$, either resulting in a new node solution vector, $\mathbf{V}^s_{new}$, or a new link solution vector, $\mathbf{E}^s_{new}$. For example, if $\mathbf{V}^s_{cur}$ is selected to generate the new state, then $\mathbf{V}^s_{new}$ is given by:

$$\mathbf{V}^s_{new} = \begin{cases} \mathbf{V}^s_{new}[i] = \mathbf{V}^s_{cur}[j], \mathbf{V}^s_{new}[j] = \mathbf{V}^s_{cur}[i] & \text{for } i, j \\ \mathbf{V}^s_{new}[k] = \mathbf{V}^s_{cur}[k] & \text{for } k \neq i \quad \& \quad k \neq j \end{cases} \tag{3.24}$$

where i, j are the two randomly-chosen indices to be swapped, $0 \leq i, j < |\mathbf{F}^k_v|$. If this process yields the same (current) state, it is repeated until a new solution is generated. Also if the new state is generated by updating the node solution vector, the candidate link set and link solution state are also updated accordingly, i.e., in order to avoid selecting isolated links and limiting the search space. The value of the objective function is then re-calculated for the newly-generated system state, and it is accepted with the probability defined in Eq. 3.21.

3.4.2 Distributed SA (D-SA) Approach

The distributed approach, meanwhile, considers a floating-point state space consisting of all possible weighted resource assignments of affected nodes and links. Namely, the current system state is now represented by two floating-point vectors, i.e., a node solution vector, $\mathbf{V}^d_{cur}$, and a link solution vector, $\mathbf{E}^d_{cur}$. Hence $\mathbf{V}^d_{cur}[i]$ defines the share of node repair resources that are allocated to the i-th affected physical node in $\mathbf{F}^k_v$, given as:

$$\Delta^v_i = \mathbf{V}^d_{cur}[i] X_k \tag{3.25}$$

where $\sum_{i=0}^{|\mathbf{F}^k_v|-1} \mathbf{V}^d_{cur}[i] = 1$. Similarly, $\mathbf{E}^d_{cur}[i]$ defines the share of the link repair resources that are allocated to the i-th affected physical link in $\mathbf{F}^k_e$, given as:

$$\Delta^e_i = \mathbf{E}^d_{cur}[i] Y_k \tag{3.26}$$

where $\sum_{i=0}^{|\mathbf{F}_e^k|-1} \mathbf{E}_{cur}^d[i] = 1$. Expectedly, this approach presents a much larger (infinite) search space as compared to the binary one in the S-SA approach. However, in order to accelerate the D-SA search process, the *distributed virtual load* (D-VL) heuristic (to be introduced in Chapter 4) is used to calculate an initial solution rather than simply choosing random node and link weight vectors. This approach prevents the initial state from being too far away from the optimum state, i.e., SA is more likely to find a near-optimum solution when an efficient initial state is used.

Now akin to the S-SA approach, each iteration randomly selects $\mathbf{V}_{cur}^d$ or $\mathbf{E}_{cur}^d$ to generate a neighboring solution and then swaps randomly-selected weights in the current solution state to generate a candidate neighboring state. Hence a new node solution vector, $\mathbf{V}_{new}^d$, or a new link solution vector, $\mathbf{E}_{new}^d$, is generated. For example if the link solution vector is selected to generate the new state, $\mathbf{E}_{new}^d$ is derived as follows:

$$\mathbf{E}_{new}^d = \begin{cases} \mathbf{E}_{new}^d[i] = \mathbf{E}_{cur}^d[j], \mathbf{E}_{new}^d[j] = \mathbf{E}_{cur}^d[i] & \text{for } i, j \\ \mathbf{E}_{new}^d[k] = \mathbf{E}_{cur}^d[k] & \text{for } k \neq i \quad \& \quad k \neq j \end{cases} \tag{3.27}$$

where i, j are the two randomly-chosen indices to be swapped, $0 \leq i, j < |\mathbf{F}_e^k|$. Furthermore, the updated node or link weights (after swapping) must not result in excessive resources being assigned to a node or link, i.e., the generated $\mathbf{V}_{new}^d$ must meet the following constraint:

$$X_k \mathbf{V}_{new}^d[i] \leq (R_i^{max} - R_i^k), \quad \forall i \in \mathbf{F}_v^k \tag{3.28}$$

Similarly, the generated $\mathbf{E}_{new}^d$ must also meet the following constraint:

$$Y_k \mathbf{E}_{new}^d[i] \leq (B_i^{max} - B_i^k), \quad \forall i \in \mathbf{F}_e^k \tag{3.29}$$

Now if the constraints in Eq. 3.28 or Eq. 3.29 cannot be met, then the respective weight is appropriately modified (reduced) to only provision up to the maximum node or link capacity, i.e., R_n^{max} and $B_{(m,n)}^{max}$, respectively. The leftover proportion of the weight is then assigned to the other node or link involved in the swapping process. Akin to the S-SA approach, the objective function is also re-calculated for the new state, and the current state is updated in a probabilistic manner as per Eq. 3.21. The D-SA scheme is also detailed in Figure 3.5.

3.5 Performance Analysis

The two progressive recovery SA variants are now evaluated using custom-developed network simulation models in $OPNETModeler^{TM}$. These tests are done using two different network topologies including a 24-node/86-link and a 46-node/152-link topology, with large failure outages affecting 20% of physical nodes and links, as shown in Figure 3.6. All substrate nodes and links have 100 units of resource capacity and 10,000 units of link bandwidth, respectively. Meanwhile, VN requests are generated randomly with 4-7 nodes each and an average node degree of 2.6. Here each VN node requires 1-10 units of capacity, and each VN link requires 50-1,000 units of bandwidth. Also, VN requests arrive in a random manner with exponential inter-arrival times (mean 60 seconds) and with infinite durations, reflecting realistic long-standing demands. Meanwhile, the average amount of incoming node (link) repair resources in each stage is set to 200 (30,000) units and 200 (50,000) units each for the 24-node and 46-node topologies, respectively. Furthermore, all recovery stages are triggered after random intervals.

Now both the VN remapping and VN playback restoration approaches are evaluated here for SA recovery, i.e., after repair resource placement during the recovery stages (sub-stage 2, Figure 3.1). These methods are appropriately labeled by the postfixes “RM” and “PB”, respectively. In particular, the former approach uses the *non-survivable virtual infrastructure mapping* (NSVIM) VNE scheme in [10], albeit other VN mapping schemes can also be re-used here. Furthermore, only *partial* VN recovery is done in order to recover the failed portions (sub-graphs) of the affected VN demands, i.e., partially-affected VN demands keep running during the recovery stages. Additionally, both SA variants are evaluated for two different values of the SA cooling rate, i.e., $\alpha = 0.1$ and $\alpha = 0.2$. Finally to stress the schemes, recovery performance is only measured for medium-heavy loads, and all results are averaged over 10 independent runs. Several performance metrics are now evaluated to gauge the various metaheuristic schemes.

3.5.1 Fully-Restored VN Ratio

The fully-restored VN ratio is defined as the proportion of fully-restored VN demands, i.e., a cumulative measure. The results for this metric for both the 24-node and 46-node topologies are shown in Figure 3.7 and Figure 3.8, respectively. Foremost, these findings indicate that VN remapping performs significantly better than VN playback in all stages (except for the final stage). Moreover, this difference is more noticeable in the selective schemes as compared to the distributed schemes. For example, consider a comparison between the S-SA and D-SA schemes for SA cooling rate of $\alpha = 0.1$ in the second recovery stage in the 24-node network topology (Figure 3.7). Here, the VN remapping instance S-SA_0.1-RM gives 30% higher recovery than its counterpart VN playback instance S-SA_0.1-PB, whereas the distributed VN remapping instance D-SA_0.1-RM only gives 15% higher recovery than its counterpart VN playback instance, D-SA_0.1-PB. In general, the benefit of distributing repair resources across more physical locations has more of an impact on the performance of the VN playback scheme, i.e., since the option of relocating VN nodes/links is not available.

Next, carefully note that the VN playback approach gives full VN recovery by the final stage, whereas the VN remapping schemes mostly level off at approximately 80-90% VN recovery. This differential is expected since remapping demands over partially-working substrate networks is not as efficient as embedding them over working infrastructures. By contrast, the VN playback scheme simply copies back prior restorations, ensuring full recovery by at least the final stage when the network is restored to its pre-fault operating condition. Hence operators can appropriately choose whether to use the VN remapping or VN playback strategy based upon their preferences, i.e., faster VN recovery in earlier stages or full recovery by final stage. Alternatively, a combination approach can also be used in which VN remapping can be used in the early-middle repair stages followed by VN playback in the final stage. However, this approach will likely cause VN service disruptions as recovered demands will have to be re-assigned to their pre-fault mappings by the final stage.

The results in Figure 3.7 and Figure 3.8 also show the effect of the SA cooling rate on post-fault VN restoration. Namely, smaller values of α give better performance, e.g., compare the D-SA_0.1-RM and D-SA_0.2-RM instances in Figure 3.7. Moreover, this improvement is more noticeable in the distributed SA variants since the search space is larger. Overall, these results are expected since larger values of the SA cooling rate yield less examination of the search space. These findings also show that the D-SA variant outperforms its S-SA counterpart in all stages for both network topologies, i.e., since it implements more even resource distribution. For example, the D-SA_0.1-RM instance gives almost 20% higher recovery than the S-SA_0.1-RM instance in the second stage for 24-node network (see Figure 3.7). Similarly, the D-SA_0.2-PB instance also gives 35% higher recovery as compared to the S-SA_0.2-PB instance (Figure 3.7).

3.5.2 Long Term Penalty

VN downtimes will affect the penalties incurred by a service provider. Hence a service disruption penalty is also defined for an affected VN demand, a, as follows:

$$P(G_a) = \mu \sum_{e \in \mathbf{E}'_a} b(a)\mathbf{P}(e) + (1-\mu) \sum_{n \in \mathbf{V}'_a} r(a)\mathbf{P}(n) \tag{3.30}$$

where $\mathbf{P}(e)$ and $\mathbf{P}(n)$ are the unit link and node penalty costs, respectively, μ is a relative scaling factor to weight the node and link penalty costs ($0 \leq \mu \leq 1$), and $b(a)$, $r(a)$, $\mathbf{E}_a^{'}$ and $\mathbf{V}_a^{'}$ are defined in Section 3.2. Both the $\mathbf{P}(e)$ and $\mathbf{P}(n)$ values are set to unity, and μ is set to 0.5 in order to treat node and link failures as equivalent. Overall, the service disruption penalty defines the degradation level experienced by a VN demand, i.e., in terms of the physical resources (nodes and links) required by a VN demand to maintain its original pre-fault allocation. Therefore the long term penalty across all affected demands is given by:

$$P_{total} = \sum_{a \in A} P(G_a) \tag{3.31}$$

Based upon the above, the results for Eq. 3.31 are plotted in Figure 3.9 and Figure 3.10 for the 24-node and 46-node networks, respectively. Foremost, these findings indicate that the VN remapping scheme gives notably lower penalty costs than the VN playback scheme in all stages except for the final stage. However, only the VN playback scheme achieves zero penalty costs in the end as it fully recovers all failed demands. Additionally, the distributed SA variants give lower penalties than their selective counterparts, i.e., for the same recovery approach (VN remapping or VN playback) and SA cooling rate parameter, α. In particular, the D-SA scheme with VN remapping, i.e., D-SA_0.1-RM instance, gives the overall lowest penalty cost.

3.5.3 Restoration Overhead

In general, VN remapping entails added overheads in case VN nodes have to be remapped to different physical nodes and/or VN links need to be re-routed onto different paths in the physical network. Therefore an overhead metric is also defined as:

$$Overhead = \zeta|\mathbf{V}_r| + (1-\zeta)|\mathbf{E}_r| \tag{3.32}$$

where $\mathbf{V}_r$ is the set of all failed VN nodes that are successfully migrated (remapped) to new locations, $\mathbf{E}_r$ is the set of all failed VN links that are successfully remapped, and ζ is a relative scaling factor to appropriately weight the overhead costs, i.e., $0 \leq \zeta \leq 1$. In general, this metric gauges the amount of control protocol overheads needed to migrate the recovered nodes and links. Furthermore, restoration overheads are only measured for the VN remapping approach, i.e., since there are no VN node or VN link relocations with the VN playback restoration approach.

Restoration overheads for the 24-node and 46-node topologies are plotted in Figure 3.11 and Figure 3.12, respectively. As expected, increased restoration rates lead to increased restoration overheads since more VN demands are recovered. As an example, consider a comparison between the D-SA_0.1-RM and D-SA_0.2-RM instances for the 24-node network (Figure 3.11). Here the D-SA_0.1-RM instance restores more VN demands in the first and second recovery stages (Figure 3.7), and hence also gives higher overheads. In the third and final stage, the D-SA_0.2-RM instance gives higher per-stage VN recovery, resulting in increased overheads. Finally, in the forth stage the two schemes give approximately the same increase in recovery rates and as a result show similar overheads. Also note that the S-SA_0.1-RM and S-SA_0.2-RM instances give approximately the same increase in the number of VN demands restored in all stages (Figure 3.7), resulting in roughly the same level of overhead, Figure 3.11. Similar tradeoffs between restoration rates and overheads are also observed in the 46-node network, e.g., compare the D-SA_0.1-RM and D-SA_0.2-RM instances in Figure 3.8 and Figure 3.12.

The findings for the 24-node network also show that the distributed SA variants generate less overheads versus their selective counterparts, i.e., since they restore VN nodes and VN links closer to their pre-failure locations owing to improved distribution of repair resources. As an example, compare the performance of the D-SA_0.2-RM and S-SA_0.2-RM instances in the third stage, Figure 3.7. While these runs give approximately similar increases in the number of restored VN demands, the D-SA_0.2-RM instance gives lower overhead (Figure 3.11). Similar results also hold for D-SA_0.1-RM and S-SA_0.1-RM instances in the first stage, see Figure 3.7 and Figure 3.11.

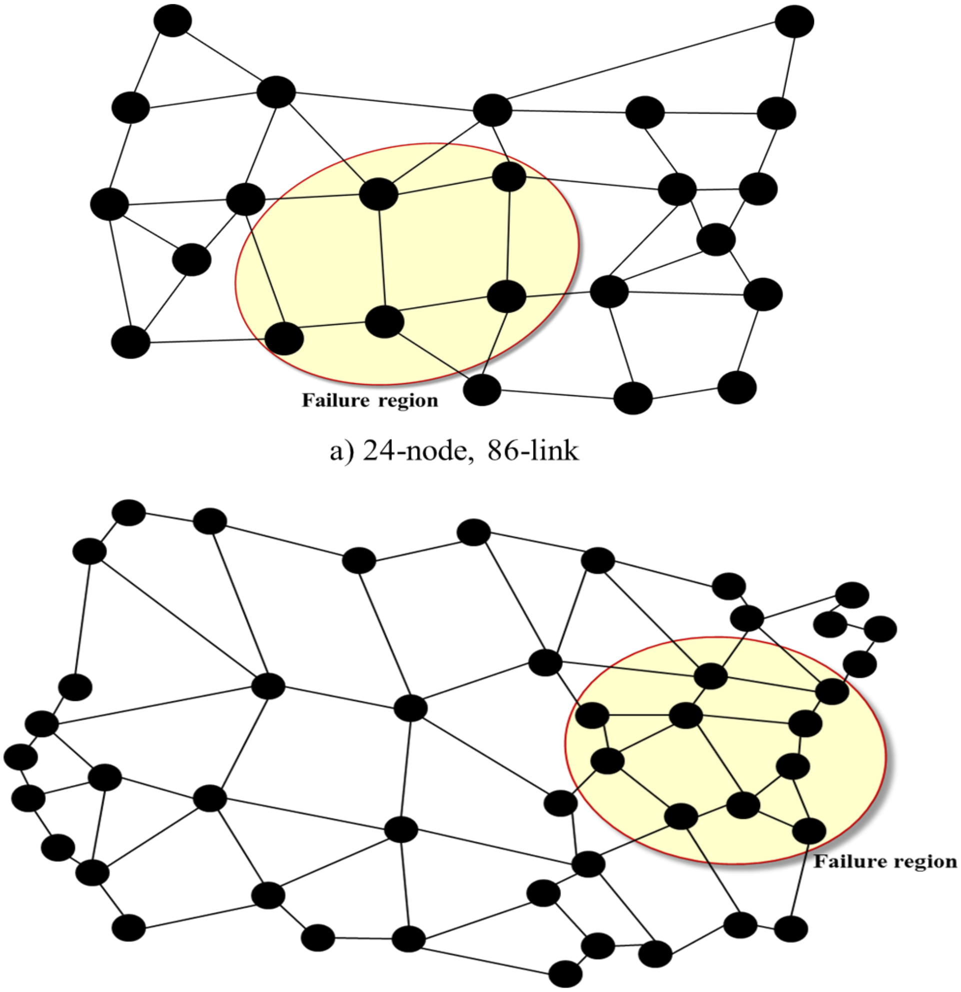
3.5.4 Average VN Path Length (Utilization)

Average VN link path lengths are also measured in order to gauge post-fault network link (bandwidth) utilization. Namely, the hop counts for all VN link connections are averaged across all working (non-failed) demands. Furthermore, these averages are only computed for the case of VN remapping recovery, i.e., since VN playback gives the same hop count utilization as before the outage.

The overall results for the 24-node and 46-node topologies are plotted in Figure 3.13 and Figure 3.14, respectively. Note that Stage 0 here represents the average VN path length prior to failure and is also equal to the final average path length for the VN playback scheme. These findings show a clear increase in average path length after the first recovery stage. This increase is expected since VN link routing is being done over partially-recovered network, leading to longer/less efficient routes to bypass failed nodes. These findings also indicate that the distributed SA approach gives lower average VN path lengths as compared to the selective SA approach despite achieving higher VN restoration ratios (for both values of the SA cooling rates, $\alpha = 0.1$ and $\alpha = 0.2$). For example, compare the D-SA_0.1-RM and S-SA_0.1-RM instances in the 46-node topology, where the former yields lower average hop count utilization (Figure 3.14) but generally higher recovery performances across all stages (Figure 3.8).

```
Given eligible nodes/links, $\mathbf{F}_v^k/\mathbf{F}_e^k$, incoming node/link repair resources ($X_k/Y_k$ units)
/* Initialize SA parameters */
Initialize temperature $temp$ and cooling rate $\alpha$
if (selective SA)
    Initialize state binary node vector, $\mathbf{V}_{cur}^s$, in random uniform manner
    Generate candidate link set (links w. working endpoints or endpoints in current node solution)
    Initialize state binary link vector, $\mathbf{E}_{cur}^s$ (randomly select candidates)
else if (distributed SA)
    Initialize state node weight vector, $\mathbf{V}_{cur}^d$ (calculate weights using Eq. 4.5)
    Initialize state link weight vector, $\mathbf{E}_{cur}^d$ (calculate weights using Eq. 4.7)
Calculate objective function value for current state
/* Main iterative loop to assign resources to selected nodes and links */
while (!done)
    Randomly select node or link solution
    if (node solution)
        if (selective SA)
            Randomly select two binary entries in $\mathbf{V}_{cur}^s$
            Swap entries to get new vector, $\mathbf{V}_{new}^s$ (if not unique repeat)
            Update candidate link set
            Build new link state vector, $\mathbf{E}_{new}^s$ (randomly select candidates)
        else if (distributed SA)
            Randomly select two weights in $\mathbf{V}_{cur}^d$
            Swap entries to get new vector, $\mathbf{V}_{new}^d$ (if not unique repeat)
            if (excessive resources at any node) /* Check resource bounds */
                Modify weights
    else if (link solution)
        if (selective SA)
            Randomly select two binary entries in $\mathbf{E}_{cur}^s$
            Swap entries to get new vector, $\mathbf{E}_{new}^s$ (if not unique repeat)
        else if (distributed SA)
            Randomly select two weights in $\mathbf{E}_{cur}^d$
            Swap entries to get new vector, $\mathbf{E}_{new}^d$ (if not unique repeat)
            if (excessive resources at any link) /* Check resource bounds */
                Modify weights
    Calculate objective function value for the new state
    /* Compute acceptance probability */
    if (improved objective value)
        $prob = 1$
    else
        $prob = exp(\frac{-\Delta r}{T})$
    Update current state to new state with probability $p$
    $temp = temp(1 - \alpha)$
    if ($temp < 1$) /* Check stopping condition */
        $done = 1$
```

Figure 3.5: Simulated annealing (SA) algorithm for repair scheduling



a) 24-node, 86-link

b) 46-node, 152-link

Figure 3.6: Test network topologies

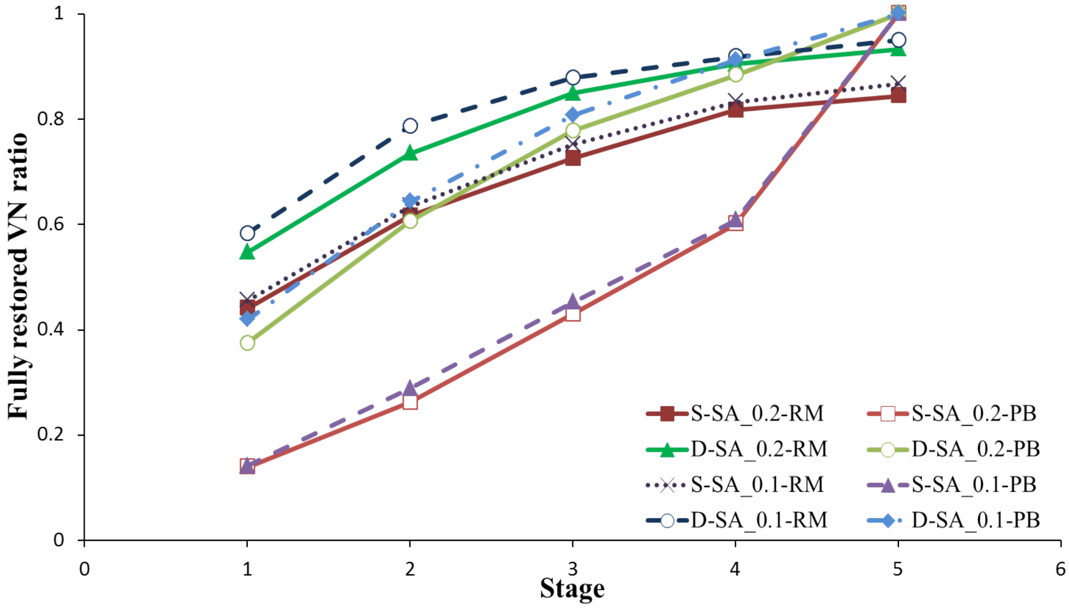


Figure 3.7: Fully-restored VN demands for 24-node network

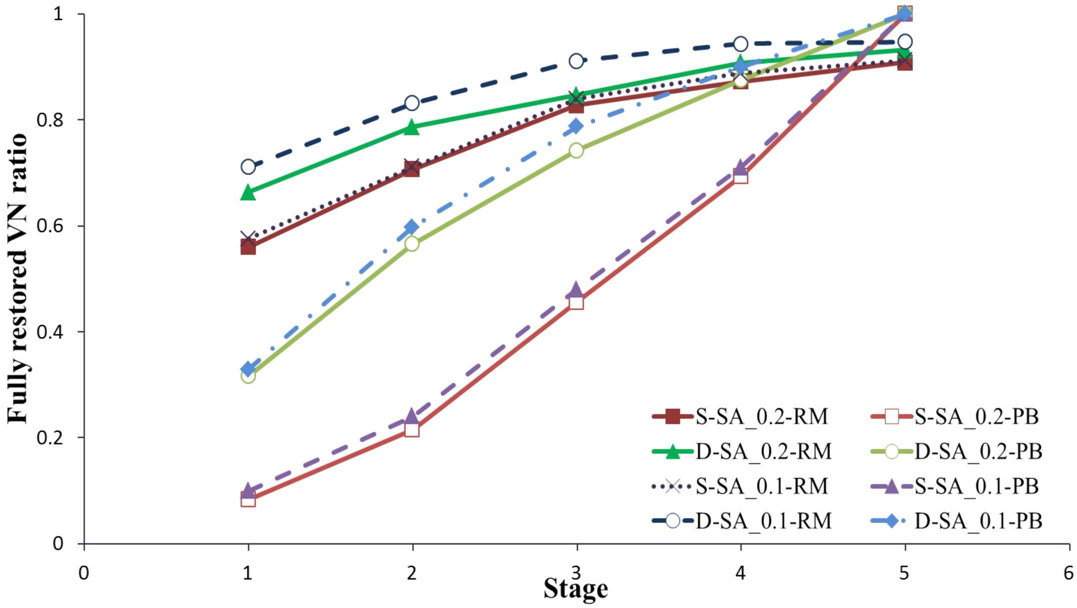


Figure 3.8: Fully-restored VN demands for 46-node network

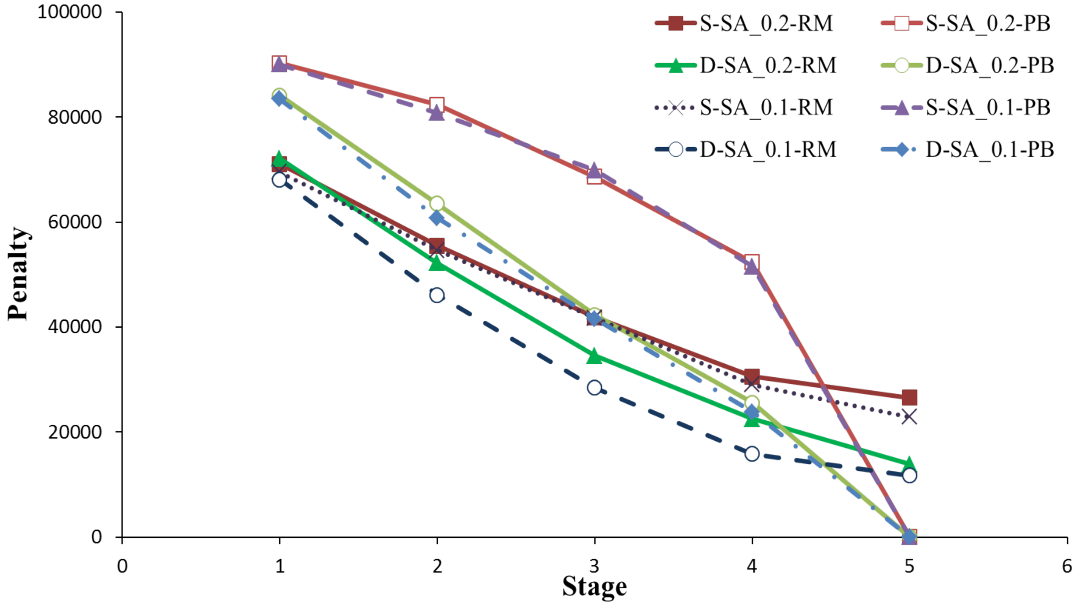


Figure 3.9: Long term penalty for 24-node network

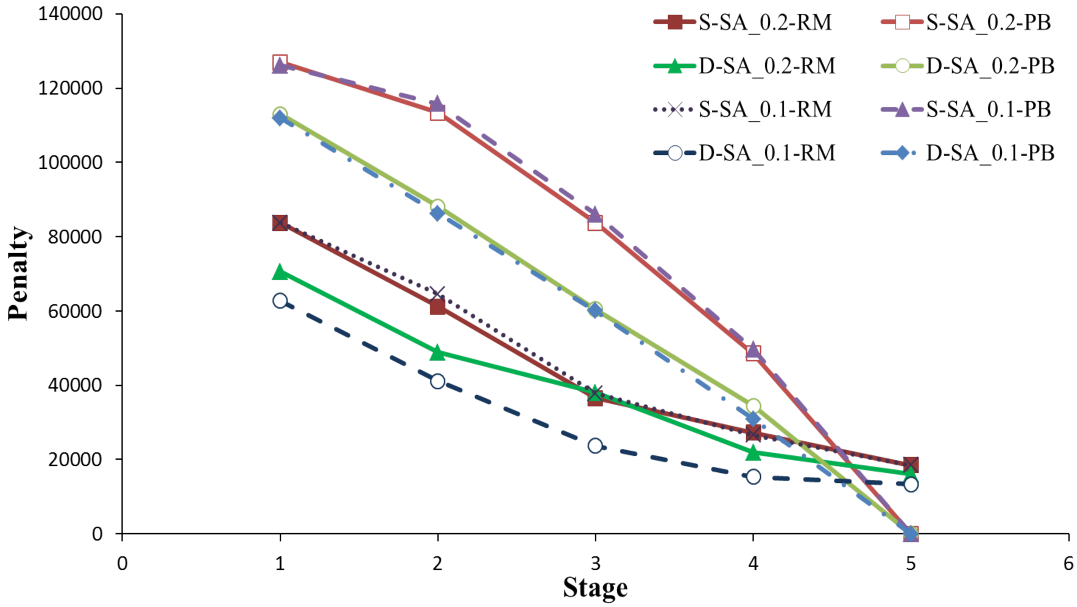


Figure 3.10: Long term penalty for 46-node network

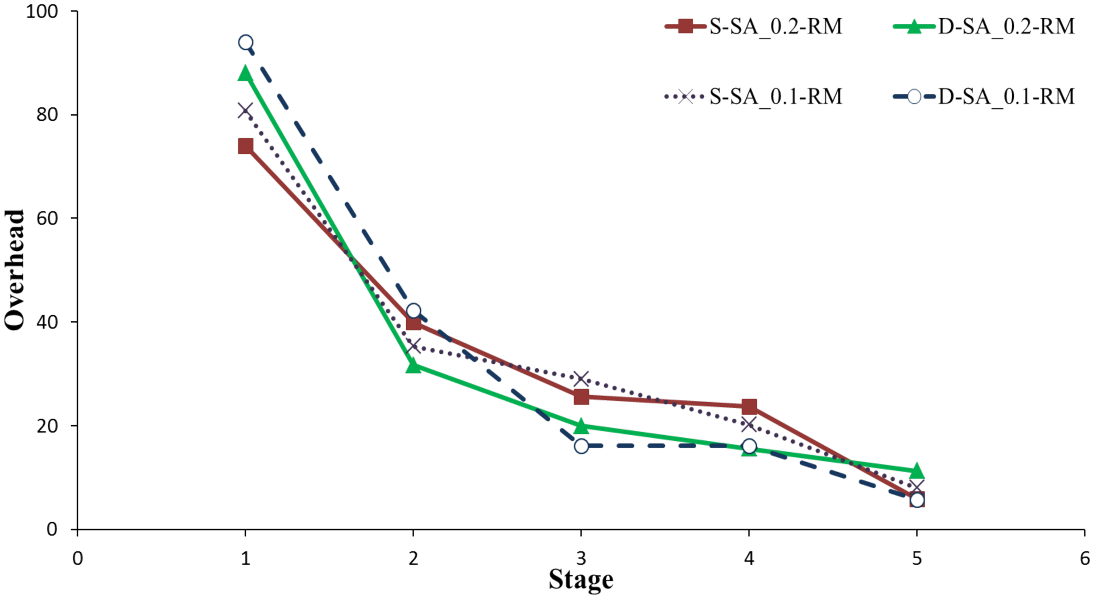


Figure 3.11: Overhead for 24-node network

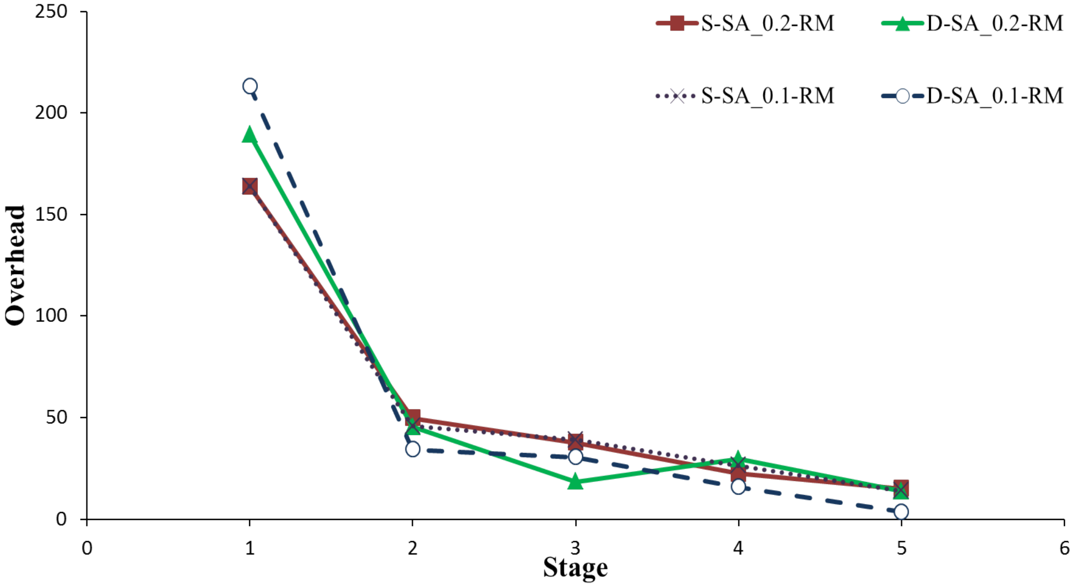


Figure 3.12: Overhead for 46-node network

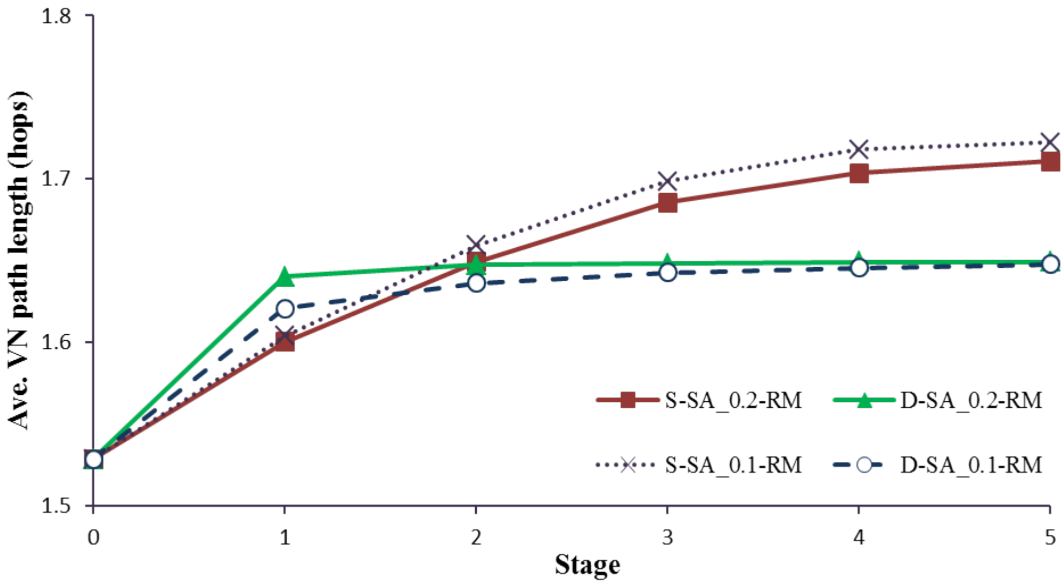


Figure 3.13: Average VN path length for 24-node network

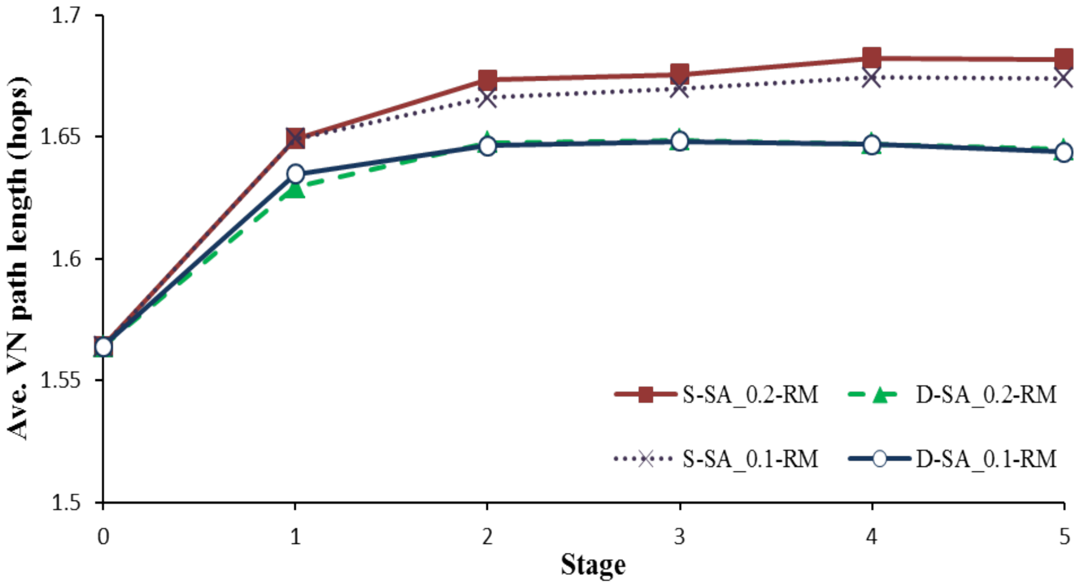


Figure 3.14: Average VN path length for 46-node network

Chapter 4

Progressive Recovery In Cloud Settings: Heuristic Strategies

In general, non-linear optimization problems with large numbers of variables and constraints are very difficult to solve, e.g., such as the single-stage progressive recovery model formulated in Chapter 3. As a result, complicated optimization approximation methods are typically required to generate feasible solutions here. Nevertheless, even these methods are sub-optimal. Furthermore, most optimization formulations are somewhat idealistic in nature, since they try to provision or recover a complete set of a-priori input (VN) demands in a given stage. Now in practice, network operators will generally require more computationally-efficient algorithms to process and place incoming repair resources within reasonable time frames. Ideally, these schemes should also provide relatively close performance to any known optimization or metaheuristic repair solutions.

To address these concerns, this chapter presents a series of intelligent scalable heuristic algorithms for progressive infrastructure repair, i.e., implemented in sub-stage 1 of the dual-stage solution approach shown in Figure. 3.1. Specifically, these schemes have bounded polynomial-time complexity and use a variety of resource scheduling/placement strategies based upon random, physical network connectivity, virtual load, and required resources. These strategies are analyzed in detail and also compared with the SA metaheuristics developed in Chapter 3.

4.1 Polynomial-Time Heuristic Repair Strategies

Overall, the proposed heuristic schemes re-use much of the notation introduced in Section 3.2. In order to improve the utilization (efficiency) of incoming repair resources, all of these schemes further partition the sets of eligible nodes and links in each stage into *candidate* sets. Namely, candidate nodes are defined as eligible nodes with at least one *non-failed* neighbor, i.e., $\mathbf{F}_v^{'k} \subseteq \mathbf{F}_v^k$ in stage k. In other words, any eligible nodes with partially-recovered or fully-working neighbors are valid. Similarly, candidate links are defined as eligible links with non-failed endpoints, i.e., $\mathbf{F}_e^{'k} \subseteq \mathbf{F}_e^k$ in stage k. Hence the heuristic algorithms basically select nodes and links from these candidate sets for resource allocation. Note that physical nodes and links can only receive repair resources upto their pre-fault maximum levels, i.e., R_n^{max} and $B_{(m,n)}^{max}$ units, respectively. The pseudocode listing for the various heuristics are shown in Figures 4.1-4.3 and complete details are now presented.

4.1.1 Baseline Random (RD) Placement

This basic scheme places incoming resources at random nodes and links. Namely, node repair resources in stage k (X_k) are placed at a random failed or partially-recovered node in $\mathbf{F}_v^{'k}$, Figure 4.1. Similarly, incoming link repair resources (Y_k) are assigned to a random failed or partially-recovered link in $\mathbf{F}_e^{'k}$, Figure 4.2. If the assigned repair resources exceed the level needed to fully recover the selected entity, then the leftover amounts are iteratively assigned to other randomly-selected nodes or link entities, see Figure 4.1 and Figure 4.2. All fully-recovered nodes and links are then removed from their respective candidate sets.

4.1.2 Physical Degree (PD) Placement

This strategy assigns repair resources based upon the physical node degree in the working *pre-fault* substrate topology. The objective here is to recover nodes with increased link connectivity first since they can generally embed and recover more affected VN nodes/links. Namely two PD variants are considered here, i.e., distributed and selective.

4.1.2.1 Distributed PD (D-PD)

This scheme assigns node repair resources to all nodes in a weighted manner. Namely, a static weight is computed for each candidate node $n \in \mathbf{F}_v'^k$ in proportion to its physical node degree (connectivity) in $\mathbf{G}_s$=$(\mathbf{V}_s, \mathbf{E}_s)$ as:

$$w_n = \frac{nd_n}{\sum_{i \in \mathbf{F}_v'^k} nd_i} \tag{4.1}$$

where nd_i is node i degree before failure and $\sum_{\forall i \in \mathbf{F}_v'^k} w_i = 1$. Based on the above, the portion of repair resources assigned to node n in stage k is given by (Figure 4.1):

$$\Delta_n = w_n X_k \tag{4.2}$$

Similarly, link repair resources are distributed among all candidate links in $\mathbf{F}_e'^k$ in a weighted manner. Again, static link weights are computed proportional to the sum of their endpoint nodes degree in $\mathbf{G}_s$=$(\mathbf{V}_s, \mathbf{E}_s)$, i.e., the weight for link $e = (m, n)$ is given by (Figure 4.2):

$$w_e = \frac{nd_m + nd_n}{\sum_{(i,j) \in \mathbf{F}_e'^k} nd_i + nd_j} \tag{4.3}$$

where nd_i and nd_j are the link (i, j) endpoint node degrees before failure. Based upon this, the portion of repair resources assigned to link e in stage k is given by:

$$\Delta_e = w_e Y_k \tag{4.4}$$

4.1.2.2 Selective PD (D-PD)

This scheme assigns node repair resources to the candidate node with the highest node-degree/connectivity, i.e., $max_{n \in \mathbf{F}'^{k}_{v}} nd_n$, Figure 4.1. Similarly, link repair resources (Y_k) are assigned to the "most-connected" candidate link, i.e., endpoints with highest node degrees, $max_{(m,n) \in \mathbf{F}'^{k}_{e}}(nd_m + nd_n)$, Figure 4.2. Any leftover resources are then assigned in an iterative manner.

4.1.3 Virtual Load (VL) Placement

This strategy assigns resources to candidate nodes and links based upon their dynamic VN load levels prior to failure. Again distributed and selective variants are considered here.

4.1.3.1 Distributed VL (D-VL)

This scheme places resources at all nodes in a weighted manner. Namely, the weight for each candidate node $n \in \mathbf{F}'^{k}_{v}$ is proportional to the amount of embedded VN nodes (load) it carries in pre-fault working mode:

$$w_n = \frac{VL_n}{\sum_{i \in \mathbf{F}'^{k}_{v}} VL_i} \tag{4.5}$$

where VL_i is virtual load at node i prior to the failure and $\sum_{\forall i \in \mathbf{F}'^{k}_{v}} w_i = 1$. Hence the portion of resources allocated to node n in stage k is (Figure 4.1):

$$\Delta_n = w_n X_k \tag{4.6}$$

Similarly, all candidate links in $\mathbf{F}_e^{\prime k}$ are weighted proportional to their pre-fault embedded VN link loads, i.e., the weight for link e is computed as:

$$w_e = \frac{VL_e}{\sum_{i \in \mathbf{F}_e^{\prime k}} VL_i} \tag{4.7}$$

where VL_i is the amount of embedded connections on link i prior to failure. Based on this, the portion of repair resources assigned to link e in stage k is given by (Figure 4.2):

$$\Delta_e = w_e Y_k \tag{4.8}$$

4.1.3.2 Selective VL (D-VL)

This scheme places incoming node repair resources at the candidate node in $\mathbf{F}_v^{\prime k}$ carrying the most amount of embedded VN nodes (load) prior to the outage event, i.e., $max_{n \in \mathbf{F}_v^{\prime k}}\{VL_n\}$, Figure 4.1. Meanwhile, incoming link repair resources are directed to the candidate link in $\mathbf{F}_e^{\prime k}$ carrying the most amount of pre-fault connections, $max_{e \in \mathbf{F}_e^{\prime k}}\{VL_e\}$, Figure 4.2.

4.1.4 Smallest Request First (SRF) Placement

This heuristic is only applicable to the VN playback recovery approach and tries to directly maximize the VN recovery rate. The pseudocode for this scheme is shown in Figure 4.3 and starts by sorting all affected VN demands in increasing order of their required node and link resources (for full recovery). Specifically, $\mathbf{A}$ denotes the set of affected VN demands (as defined in Chapter 3). A weighted resource requirement, r^a, is then computed for each demand $a \in \mathbf{A}$ as follows:

$$r^a = \Gamma |\mathbf{V}_a^{'}| r(a) + (1 - \Gamma)|\mathbf{E}_a^{'}| b(a) \tag{4.9}$$

```
Given eligible nodes, $\mathbf{F}_v^k$, and incoming node repair resources ($X_k$ units)
Given current resource level of node $n$, $R_n^k$ units
Define candidate node subset, $\mathbf{F}_v^{'k}$, i.e., nodes in $\mathbf{F}_v^k$ with at least one
partially-recovered/working neighbor
/* Select appropriate node(s) for placement */
while ($X_k \neq \emptyset$ and $|\mathbf{F}_v^{'k}| \neq \emptyset$)
    if (random placement)
        Select random node $n \in \mathbf{F}_v^{'k}$
    else if (physical degree)
        if (distributed)
            Assign $w_n$ to each $n \in \mathbf{F}_v^{'k}$ using Eq. 4.1
        else if (selective)
            Select node $n \in \mathbf{F}_v^{'k}$ with max. node degree in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$)
    else if (virtual load)
        if (distributed)
            Assign $w_n$ to each $n \in \mathbf{F}_v^{'k}$ using Eq. 4.5
        else if (selective)
            Select node $n \in \mathbf{F}_v^{'k}$ with max. pre-fault embedded VN loads
    /* Assign resources to selected node(s) */
    if (distributed)
        Assign resources to each $n \in \mathbf{F}_v^{'k}$ up to $min\{X_k w_n, R_n^{max}\text{-}R_n^k\}$
        $X_k = X_k - min\{X_k w_n, R_n^{max}\text{-}R_n^k\}$
    else if (selective)
        Assign resources to the selected node up to $min\{X_k, R_n^{max}\text{-}R_n^k\}$
        $X_k = X_k - min\{X_k, R_n^{max}\text{-}R_n^k\}$
    if ($R_n^k = R_n^{max}$)
        $\mathbf{F}_v^{'k} = \mathbf{F}_v^{'k}-$ {node $n$}
    Update candidate node subset, $\mathbf{F}_v^{'k}$
```

Figure 4.1: Node repair resource distribution for RD, VL, PD schemes (stage k)

where Γ is a relative scaling factor between node and link resources, $0 \leq \Gamma \leq 1$, and $r(a)$, $b(a)$, $\mathbf{V}_a^{'}$, $\mathbf{E}_a^{'}$ are defined earlier in Chapter 3. The algorithm then cycles through each affected demand to check if it can be recovered by the amount of the available repair resources. If so, the requisite resources are assigned to the affected physical nodes and links, and the respective demand is recovered. Overall, this approach can also be termed as a distributed strategy since repair resources are spread across multiple affected physical nodes and links carrying affected demands. The algorithm terminates when there are not enough repair resources or all affected VN demands are recovered.

```
Given eligible links, $\mathbf{F}_e^k$, and incoming link repair resources ($Y_k$ units)
Given current resource level of link $e$, $B_e^k$ units
Define candidate link subset, $\mathbf{F}_e^{'k}$, i.e., links in $\mathbf{F}_e^k$ with
partially-recovered/working endpoints
/* Select appropriate link(s) for placement */
while ($Y_k \neq \emptyset$ and $|\mathbf{F}_e^{'k}| \neq \emptyset$)
  if (random placement)
    Select random link $e \in \mathbf{F}_e^{'k}$
  else if (physical degree)
    if (distributed)
      Assign $w_e$ to each $e \in \mathbf{F}_e^{'k}$ using Eq. 4.3
    else if (selective)
      Select link $e \in \mathbf{F}_e^{'k}$ with max. pre-fault endpoint node degree
  else if (virtual load)
    if (distributed)
      Assign $w_e$ to each $e \in \mathbf{F}_e^{'k}$ using Eq. 4.7
    else if (selective)
      Select $e \in \mathbf{F}_e^{'k}$ with max. pre-fault VN link connections
  /* Assign resources to selected link(s) */
  if (distributed)
    Assign resources to each $e \in \mathbf{F}_e^{'k}$, up to $min\{Y_k w_e, B_e^{max}\text{-}B_e^k\}$
    $Y_k = Y_k - min\{Y_k w_e, B_e^{max}\text{-}B_e^k\}$
  else if (selective)
    Assign resources to the selected $e$ up to $min\{Y_k, B_e^{max}\text{-}B_e^k\}$
    $Y_k = Y_k - min\{Y_k, B_e^{max}\text{-}B_e^k\}$
  if ($B_e^k = B_e^{max}$)
    $\mathbf{F}_e^{'k} = \mathbf{F}_e^{'k} -$ {link $e$}
```

Figure 4.2: Link repair resource distribution for RD, VL, PD schemes (stage k)

4.2 Complexity Analysis

Now consider the run-time complexity of the proposed heuristic algorithms. Clearly, the RD scheme is of $O(1)$ complexity as it simply selects one random node or link at a time. Meanwhile, both the selective PD and VL schemes (i.e., S-PD, S-VL) require sorting of candidate physical nodes and links and therefore have $O(|\mathbf{V}_s|\log|\mathbf{V}_s| + |\mathbf{E}_s|\log|\mathbf{E}_s|)$ complexity. Conversely, the distributed instances of the PD and VL schemes (i.e., D-PD, D-VL) only compute the corresponding weights for affected nodes and links and hence are of

```
Given incoming node/link repair resources ($X_k/Y_k$ units)
Given affected VN set $\mathbf{A}$ and node/link resources $r(a)/r(b)$
required by VN demand $a \in \mathbf{A}$
Given affected VN node set $\mathbf{V}^{'}_{a}$, affected VN link set $\mathbf{E}^{'}_{a}$ for VN demand $a \in \mathbf{A}$
for (each $a \in \mathbf{A}$)
    Calculate $r^a$ using Eq. 4.9
Sort $\mathbf{A}$ in increasing order of $r^a$
for (each $a \in \mathbf{A}$)
    if ($|\mathbf{V}^{'}_{a}|r(a) \leq X_k$ and $|\mathbf{E}^{'}_{a}|b(a) \leq Y_k$ )
        Assign resources to pre-fault locations of affected VN nodes/VN links
        $X_k = X_k - |\mathbf{V}^{'}_{a}|r(a)$
        $Y_k = Y_k - |\mathbf{E}^{'}_{a}|b(a)$
        Restore VN demand $a$
        $\mathbf{A} = \mathbf{A} -$ {VN demand $a$}
    else
        return
```

Figure 4.3: Node/link repair resource distribution for SRF scheme (stage k)

$O(|\mathbf{V}_s| + |\mathbf{E}_s|)$ complexity. Finally, the SRF algorithm requires sorting of all affected VNs, and hence is of $O(|\mathbf{A}| \log |\mathbf{A}|)$ complexity.

4.3 Performance Analysis

The progressive recovery heuristics are now evaluated using custom-developed network simulation models in $OPNETModeler^{TM}$. For evaluation and comparison purposes, many of the test parameters are kept the same as those used in Chapter 3. Foremost, the same topologies and associated failure regions are used here, i.e., 24-node/86-link and 46-node/152-link networks with 20% node and link failures, as shown in Figure 3.6. Also, all substrate nodes and links have 100 units of resource capacity and 10,000 units of link bandwidth, respectively. Again, VN requests are generated randomly with 4-7 nodes each with an average node degree of 2.6. Meanwhile each VN node requires 1-10 units of capacity, and each VN link requires 50-1,000 units of bandwidth. All VN requests arrive in a random manner with exponential inter-arrival times (mean 60 seconds) and have infinite durations. The average amount of incoming node (link) repair resources in each stage is set to 200 (30,000) units and 200 (50,000) units each for the 24-node and 46-node topologies, respectively. Finally, the NSVIM scheme [10] is re-used for remapping failed VN demands during the recovery stages, and *partial* VN remapping is done to recover only the failed portions (sub-graphs) of affected VN demands. As per Chapter 3, recovery performance is only measured for medium-heavy loads and all results are averaged over 10 independent runs.

The results are now presented. In terms of labeling notation, the distributed and selective schemes are denoted by the prefixes "D" and "S", respectively. Also, the VN playback and VN remapping options are denoted by the postfixes "PB" and "RM", respectively.

4.3.1 Fully-Restored VN Ratio

VN recovery performance is gauged by measuring VN restoration ratios and the results for the VN playback recovery approach are presented first in Figure 4.4 (24-node topology) and Figure 4.5 (46-node topology). These findings indicate that the baseline RD scheme gives the lowest overall performance in both networks. Next, the selective schemes give moderate recovery gains over the baseline, i.e., see results for S-PD-PB and S-VL-PB instances. By contrast, the distributed schemes give much better recovery, yielding well over 30% higher VN restoration ratios than their selective counterparts at intermediate recovery stages (for both physical degree and virtual load placement). For example, compare the results for the D-PD-PB and S-PD-PB instances in Figure 4.4. These findings indicate that spreading recovery resources across more eligible nodes/links gives more effective recovery. Finally, the SRF scheme gives the best overall performance, averaging over 10% (5%) higher recovery than the best distributed scheme in most stages in the 24-node topology (46-node topology). Meanwhile, the more compute-intensive distributed SA metaheuristic is also plotted here for comparison sake and outperforms all degree-based schemes but not the SRF approach. Note that all playback schemes give full recovery by the final stage.

The VN restoration ratios for the VN remapping approach are also plotted next in Figure 4.6 (24-node network) and Figure 4.7 (46-node network). For reference comparison, results for the SRF scheme (a playback approach) are also shown here, i.e., since this scheme gives the best recovery performance amongst all VN playback schemes. Again, the baseline RM scheme gives the lowest restoration performance here, followed by the selective schemes for physical degree (S-PD-RM) and virtual load (S-VL-RM). Also the distributed methods give notable improvements across all stages, particularly in the 24-node topology. Nevertheless, the difference between the distributed and selective variants is much lower here as compared to the VN playback case, i.e., particularly in the larger 46-node topology.

Now as noted in Chapter 3, VN remapping cannot achieve full VN recovery (as compared to VN playback). However, results with the SRF scheme are still included in Figure 4.6 and Figure 4.7 for comparison purposes. Here it is seen that the best overall VN remapping strategy (i.e., D-SA-RM) generally outperforms the SRF scheme in both topologies at most intermediate stages. In general, most of the distributed heuristics also give notably better recovery performance than the SRF scheme in the larger 46-node topology, see Figure 4.7. In fact, it is only (close to) the final stage where this method achieves full recovery and outperforms VN remapping. Although the SRF scheme does not perform as well with large numbers of affected VN demands (e.g., early stages in 46-node network), this approach does show improved performance as this number declines.

4.3.2 Long Term Penalty

Long term penalty costs are also evaluated for the various heuristic schemes based upon the metrics defined in Eqs. 3.30 and 3.31. In particular, Figure 4.8 and Figure 4.9 plot the results for the 24-node and 46-node topologies, respectively, using the VN playback recovery method. Again, the findings concur with the earlier results on VN restoration ratios with VN playback (Figure 4.4 and Figure 4.5), with the distributed schemes giving notably lower penalties in most stages. Note that the penalty costs for the VN playback schemes also fall to zero by the final stage. This drop is expected since playback schemes give full recovery by the final stage. Furthermore, the distributed heuristics closely track results with the SA metaheuristic, i.e., D-PD-PB and D-VL-PB instances in Figure 4.8 and Figure 4.9. Finally, the SRF scheme outperforms all other schemes in terms of penalty costs, giving the fastest penalty reduction in both network topologies.

Next, the long term penalty costs for VN remapping are also plotted in Figure 4.10 (24-node network) and Figure 4.11 (46-node network). These findings show slightly lower penalties with the selective schemes (i.e., S-PD-RM, S-VL-RM instances), albeit the relative improvement over the distributed schemes is much lower. For reference sake, the SRF scheme is also presented here as it generally gives the best performance of all VN playback strategies. Here, the VN remapping schemes generally give lower penalties in the earlier stages (particularly distributed D-PD-RM, D-VL-RM instances). However, since penalty costs are directly related to the number of failed VN demands, the SRF scheme eventually recovers more (all) demands and gives the lowest costs. Hence these findings indicate that dynamic remapping is a more effective strategy in the intermediate recovery stages.

4.3.3 Restoration Overhead

Restoration overheads are also measured for the various heuristic schemes using the metric defined in Eq. 3.32. The overall results for VN remapping recovery are shown in Figure 4.12 and Figure 4.13 for the 24-node and 46-node topologies, respectively. As expected, increased numbers of restored VN demands lead to increased restoration overheads since more VN nodes and VN links are remapped. As an example, compare the D-PD-RM and D-SA-RM instances in the 24-node network. Here, in the first recovery stage, the D-SA-RM instance recovers 10% more VN demands (Figure 4.6) and therefore gives higher overheads (Figure 4.12). In the second stage, however, both schemes show a 20% increase in the number of recovered VN demands, resulting in roughly the same level of overhead. Meanwhile in the third stage, the D-PD-RM instance restores more VN demands as compared to the D-SA-RM instance, which leads to higher overheads. Similar tradeoffs between restoration and overhead are also seen in the 46-node network, e.g., compare the S-SA-RM and S-PD-RM instances in Figure 4.13.

The findings for the 24-node network also indicate that the virtual load schemes give lower overheads as compared to the physical degree schemes, whereas both yield the same performance in terms of VN recovery. For example, the S-VL-RM and S-PD-RM instances show approximately the same increase in restoration rate in the first, second and fourth stages in the 24-node network (Figure 4.6). However, the S-VL-RM instance gives slightly lower restoration overheads in those stages as compared to the S-PD-RM instance. Similar results also hold between the distributed VL and PD variants. In general, virtual load schemes place resources at nodes/links carrying more pre-failure VN demands and are therefore more effective in restoring VN mappings to their original pre-fault mappings.

4.3.4 VN Path Length (Utilization)

Finally the average connection (VN link) path lengths are also measured for VN remapping recovery (note that the VN playback scheme restores VN links to their original routes, yielding the same pre- and post-fault average path lengths). In particular, Figure 4.14 and Figure 4.15 plot results for the 24-node and 46-node topologies, respectively (with Stage 0 representing the pre-fault average path lengths, i.e., VN playback recovery). Again, as observed with the SA metaheuristic schemes, VN path lengths tend to increase over successive recovery stages as VN remapping generates longer and less efficient routes. However, the distributed heuristics generally give slightly lower overheads here versus their selective counterparts, for both topologies, i.e., about 3-5 % lower for both the physical degree and virtual load variants. This improvement is achieved in addition to the fact that distributed strategies also give better VN restoration ratios, i.e., Figure 4.6 and Figure 4.7. Finally, these variants closely track the path length results for the distributed SA metaheuristic as well. Overall, these findings confirm that distributing recovery resources across multiple damaged sites gives notable gains, i.e., as opposed to concentrating them at the most heavily-connected or overloaded sites.

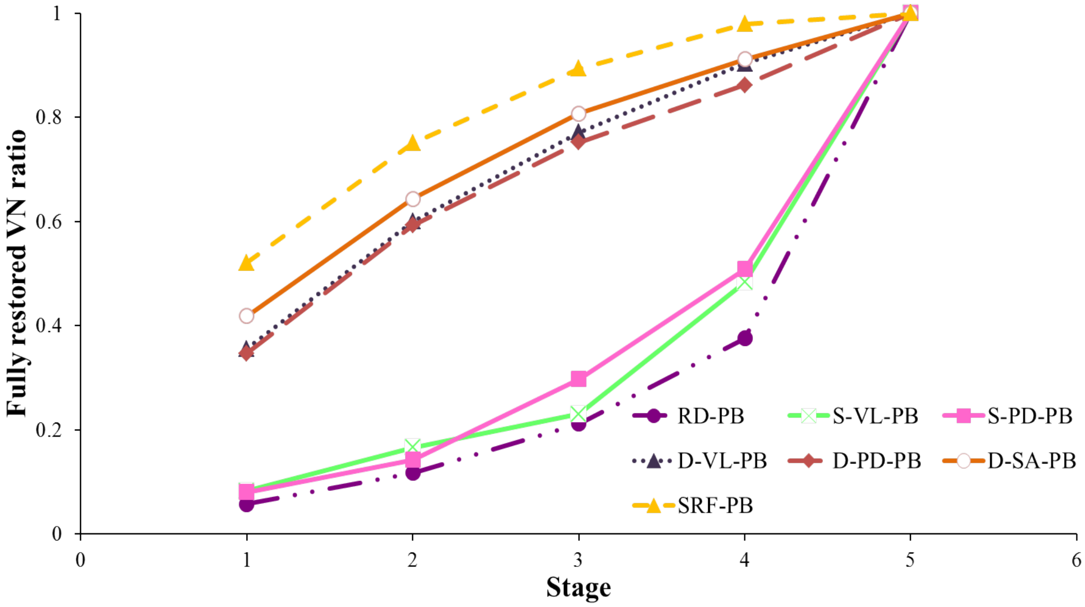


Figure 4.4: Fully-restored VN demands for 24-node network (VN playback)

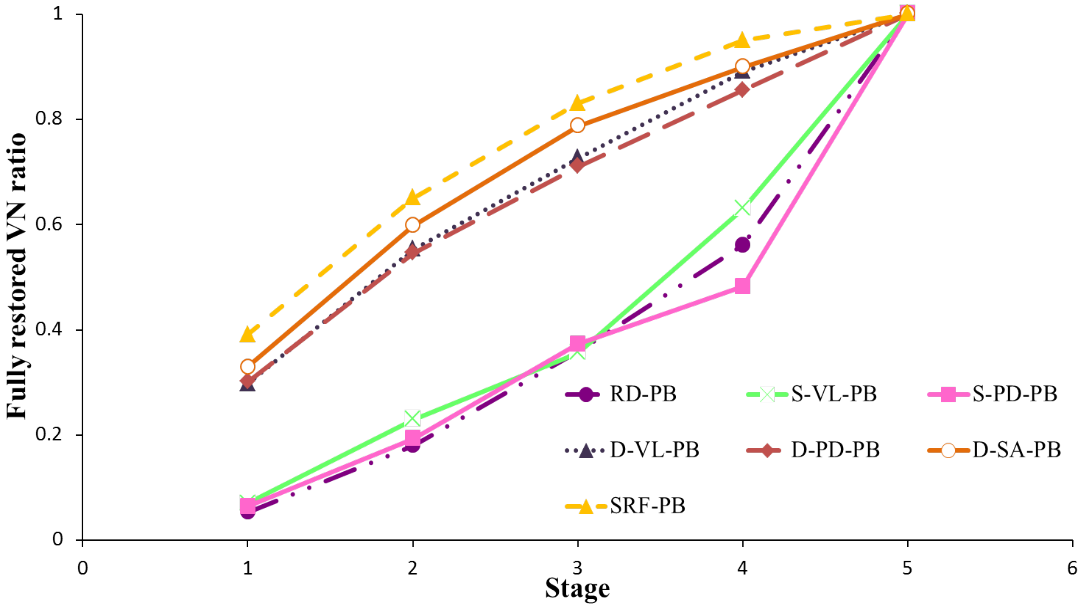


Figure 4.5: Fully-restored VN demands for 46-node network (VN playback)

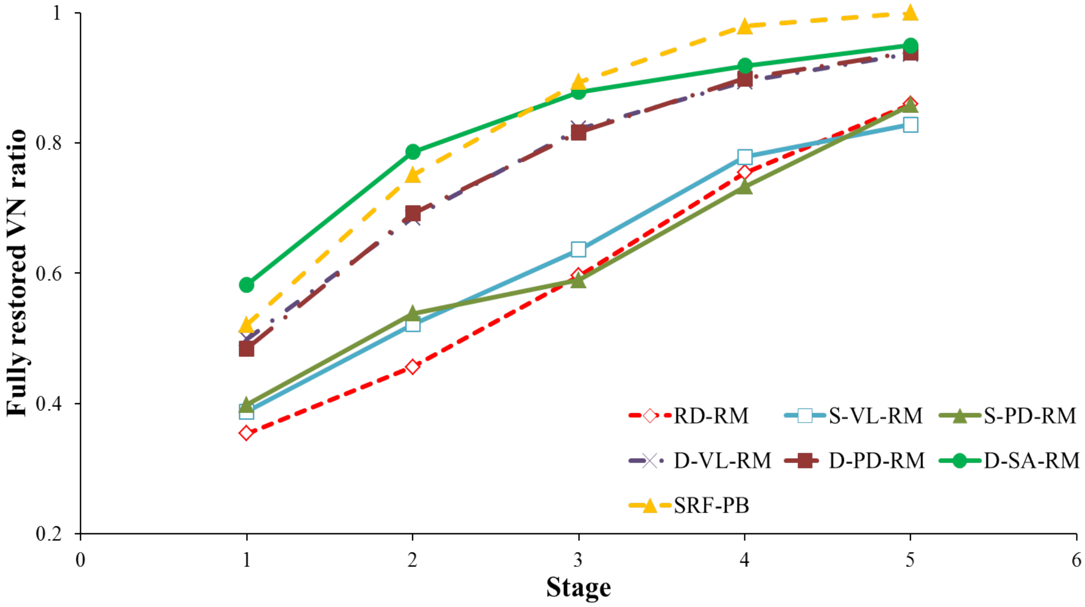


Figure 4.6: Fully-restored VN demands for 24-node network (VN remapping)

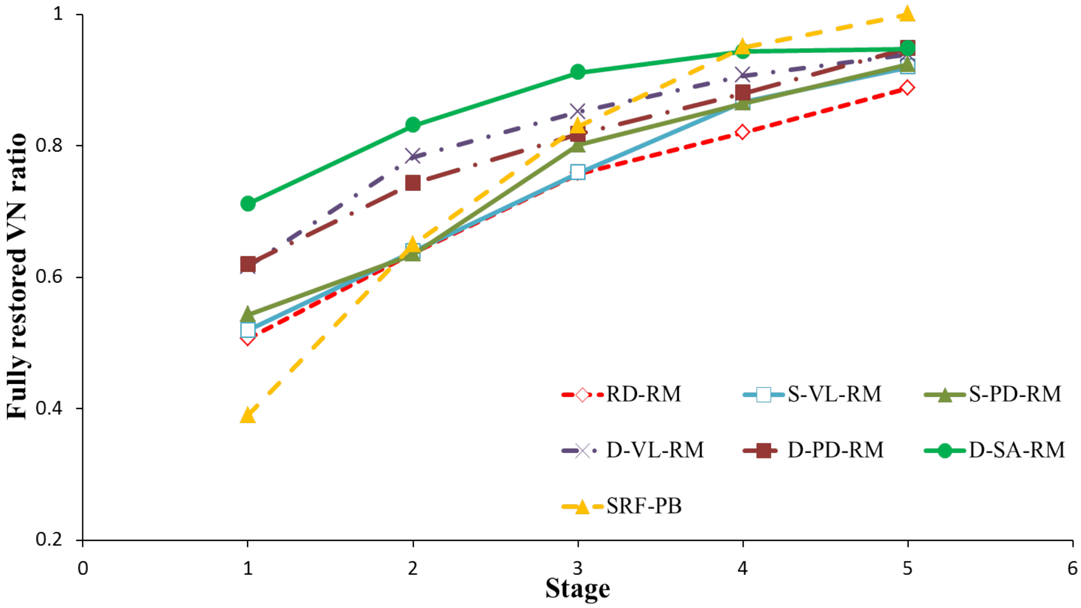


Figure 4.7: Fully-restored VN demands for 46-node network (VN remapping)

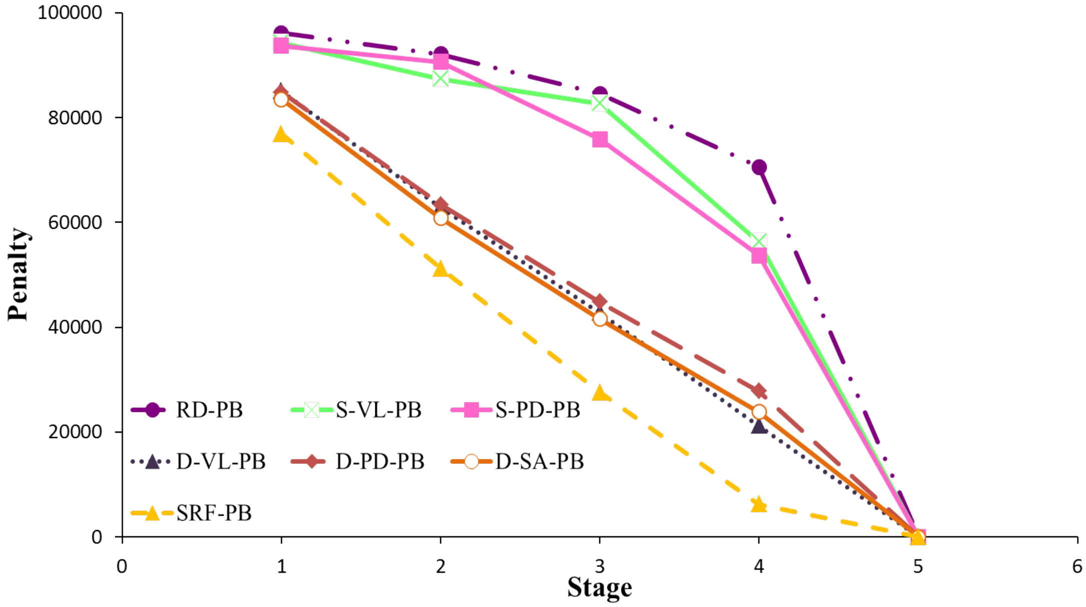


Figure 4.8: Network penalty for 24-node network (VN playback)

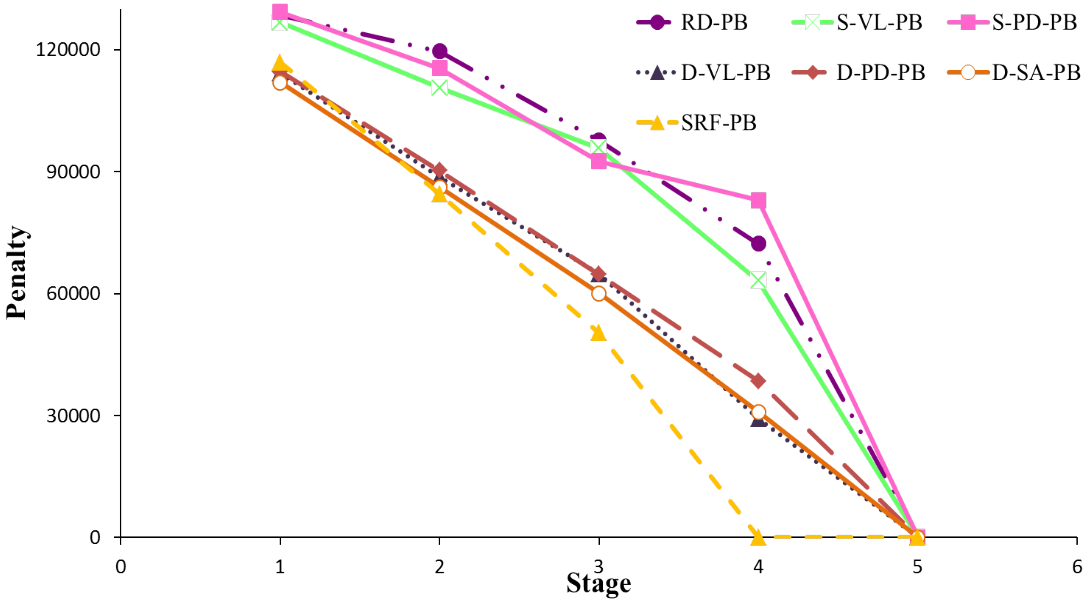


Figure 4.9: Network penalty for 46-node network (VN playback)

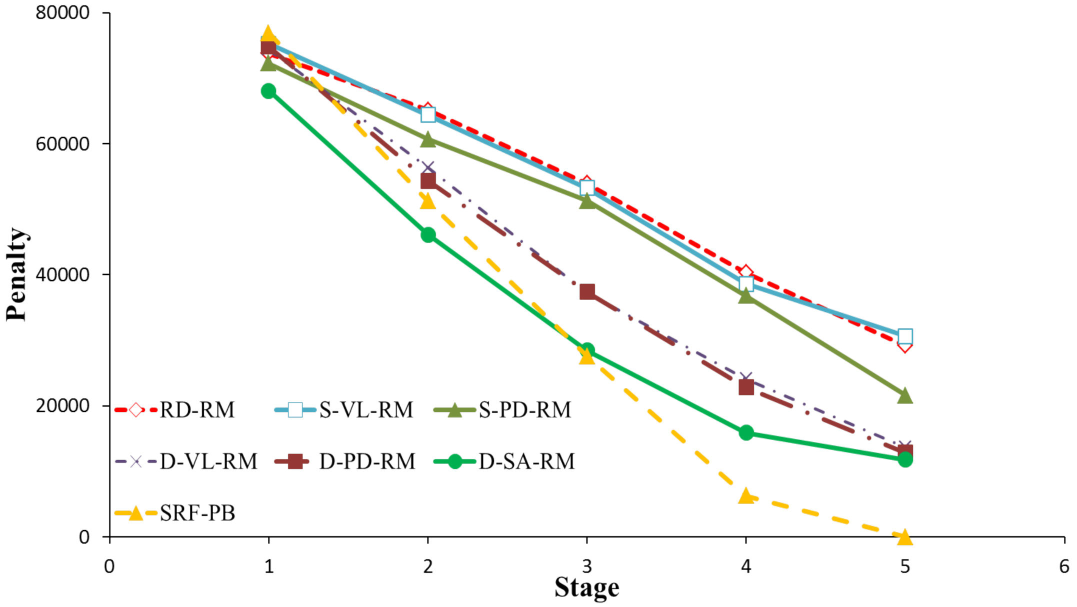


Figure 4.10: Network penalty for 24-node network (VN remapping)

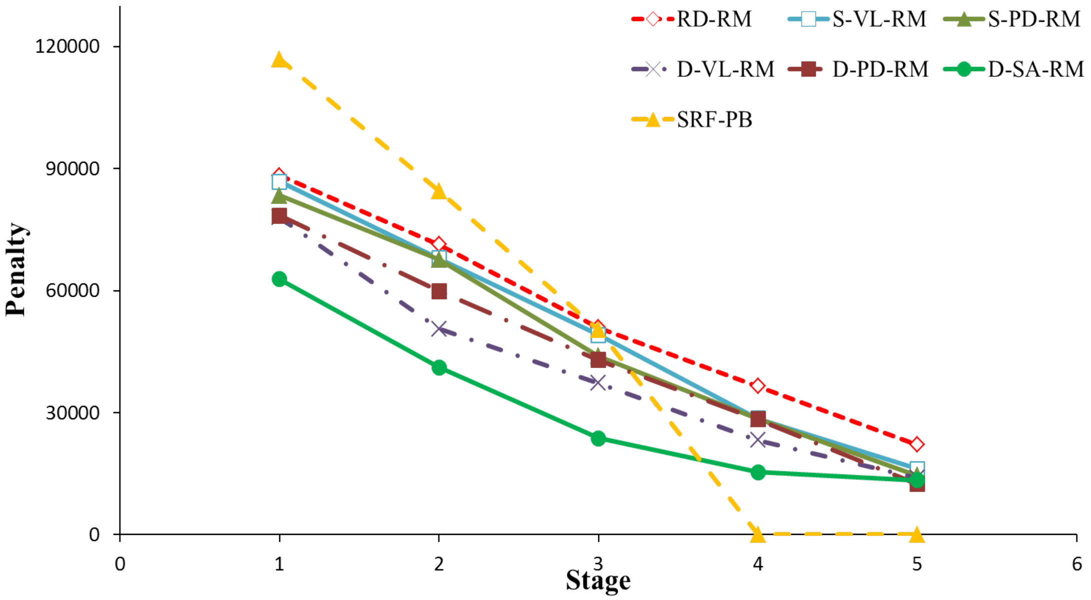


Figure 4.11: Network penalty for 46-node network (VN remapping)

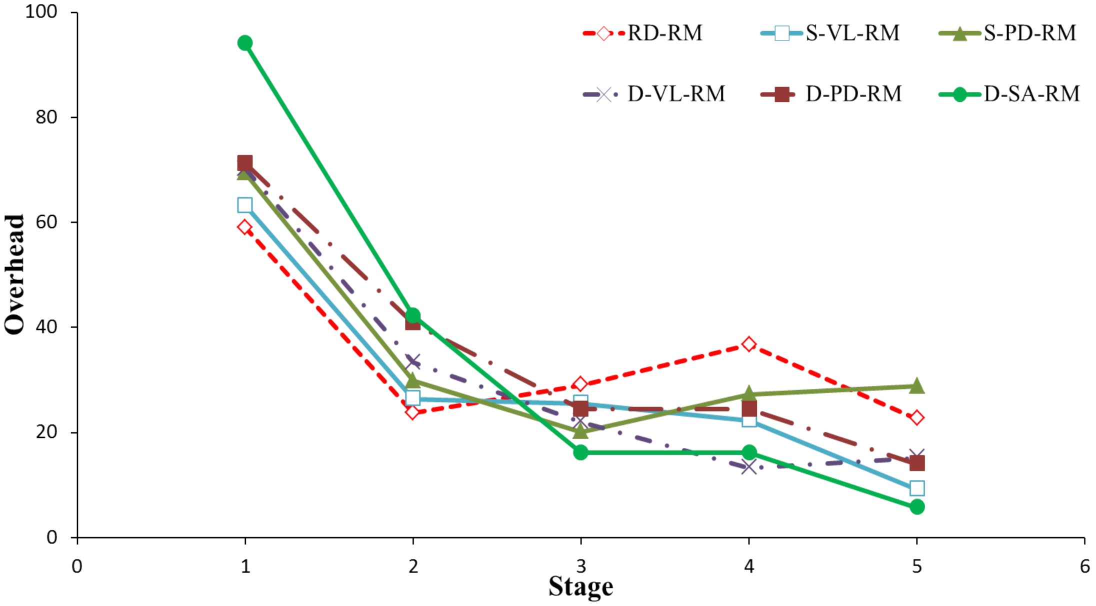


Figure 4.12: VN restoration overhead for 24-node network (VN remapping)

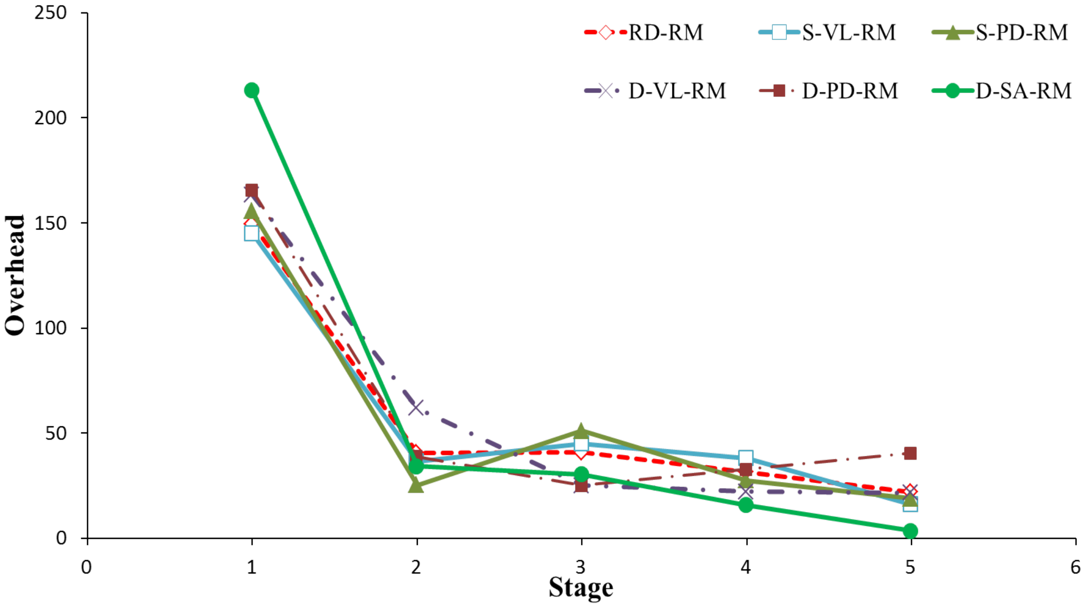


Figure 4.13: VN restoration overhead for 46-node network (VN remapping)

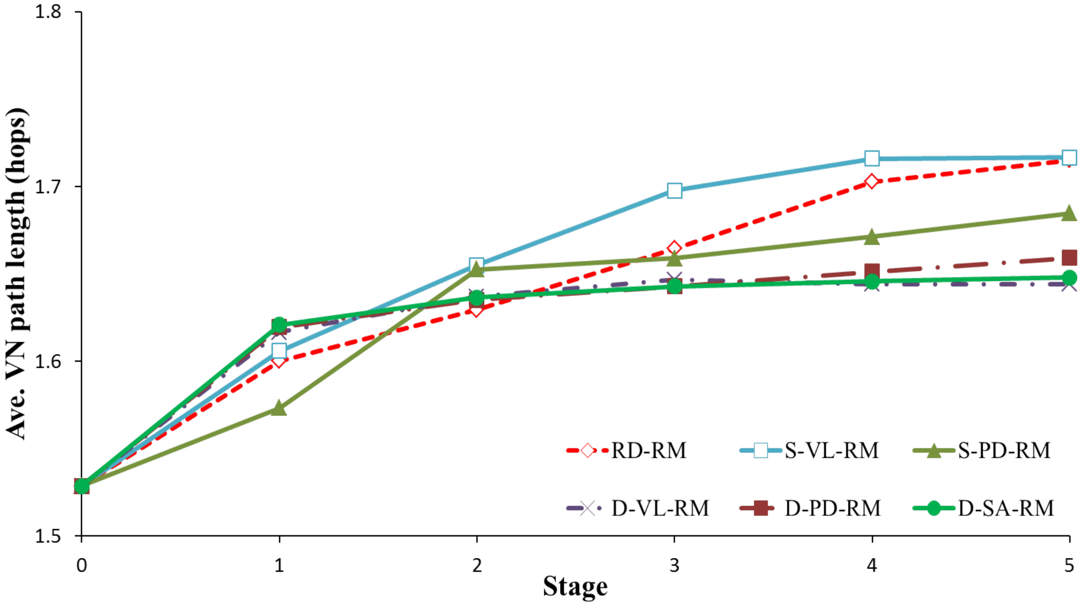


Figure 4.14: Average VN path length for 24-node network (VN remapping)

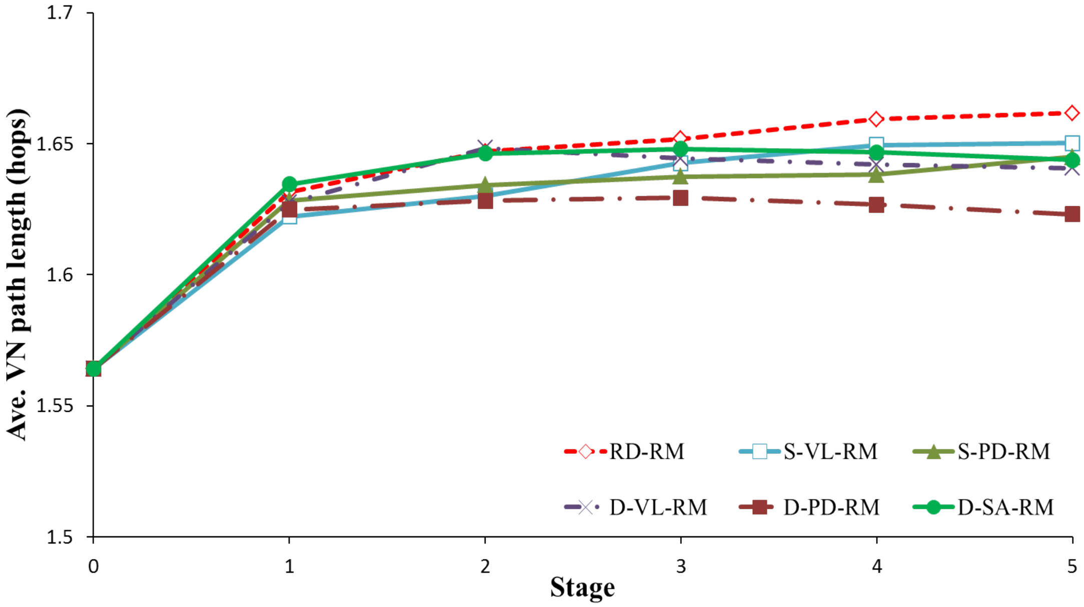


Figure 4.15: Average VN path length for 46-node network (VN remapping)

Chapter 5

Cloud-Based Multicast Service Mapping

Many service providers are also interested in offering *multicast VN* (MVN) services to support real-time data streaming and content distribution needs. Since these offerings use more specialized demand models, there have been some further studies on MVN embedding and survivability design, see surveys in Sections 2.6 and 2.7. However most of these efforts focus on backup pre-provisioning schemes to recover isolated single node/link failures, and not more challenging multi-failure outages. More importantly, many of these solutions do not leverage more efficient tree-based multicast routing techniques. Instead, client MVN demands are resolved as series of unicast VN requests, and existing VNE solutions are simply adapted to handle delay and delay variation constraints. As a result these solutions yield high resource inefficiency and reduced revenues for operators.

In light of the above, there is a pressing need to develop new "risk-aware" embedding algorithms to improve MVN services resiliency against large outages yielding multiple node/link failures. Hence this chapter presents the associated MVN demand model as well as a more realistic probabilistic failure model for multi-failure outages. Subsequently, novel graph-based heuristic schemes are presented to improve MVN embedding resiliency and also achieve a proper tradeoff with resource usage, i.e., including methods based upon resource usage minimization, failure risk minimization, and hybrid resource usage and failure risk . Detailed analyses are also conducted to gauge the performance of these methods.

5.1 Network Model and Notation Overview

The overall network model for MVN embedding is presented first followed by the requisite notational details for the demand and multi-failure outage models, as shown in Figure 5.1.

5.1.1 Physical Network

As per Chapters 3 and 4, the physical network is represented by the graph $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$, where each node $v \in \mathbf{V}_s$ has a limited amount of node capacity and each link $e \in \mathbf{E}_s$ has a limited amount of bandwidth capacity. Namely, the available resource capacity of a physical node v is given by $R^{'}_{v}$, and the available bandwidth capacity for a physical link e is given by $B^{'}_{e}$. Furthermore, each link $e \in \mathbf{E}_s$ has an associated cost $c(e)$, and delay $d(e)$. Based upon the above, the end-to-end delay for a path route, $\mathbf{P}$, comprised of a series of edges (links) is given by:

$$d(\mathbf{P}) = \sum_{e \in \mathbf{P}} d(e) \tag{5.1}$$

Similarly, the total cost for this path is also given by:

$$c(\mathbf{P}) = \sum_{e \in \mathbf{P}} c(e) \tag{5.2}$$

5.1.2 Multicast Virtual Network (MVN) Demand

As noted in Section 2.6, a MVN demand is a special type of VN request which only specifies the source and terminal nodes (with additional constraints for delay and delay variation). Ideally, this demand should be provisioned as a multicast tree and not as a VN request with a star-based "hub-and-spoke" topology, i.e., individual VN link connections for all terminal nodes. Now the traditional multicast routing problem assumes fixed source and terminal endpoint nodes, i.e., no node mapping required. By contrast, most recent studies on MVN embedding assume full flexibility in terms of node mapping/placement. In reality, however, most clients will prefer a median approach with some level of geographic localization/variability for their MVN terminal nodes endpoints, i.e., in order to achieve sufficient distribution of content and improved end-user responsiveness. Therefore a *constrained* MVN demand model is chosen here, similar to that in [49]. Specifically, this formulation allows clients to pre-specify a generalized mapping region for each MVN node, but not necessarily the exact substrate node. Consider the requisite notational details, as shown in Figure 5.1.

The MVN request is denoted by the multi-entry tuple $(s, \mathbf{D}, \delta, \gamma, r, b)$ where s is the source node, $\mathbf{D} = \{d_1, d_2, ...\}$ is the set of terminal nodes, δ is the maximum allowable delay bound, γ is the maximum delay variation bound, r is the node resource requirement (source, terminals), and b is bandwidth capacity requirement to all terminals. Furthermore, as per the constrained MVN demand model, the source can only be mapped to a subset of physical nodes given by $\mathbf{loc}_s \subset \mathbf{V}_s$. Similarly, each terminal node $d_i \in \mathbf{D}$ can only be mapped to a subset of physical nodes give by $\mathbf{loc}_{d_i} \subset \mathbf{V}_s$. This overall notation is also shown in Figure 5.1 for a sample substrate topology hosting a single MVN request.

5.1.3 Multi-Failure Outage Model

A more realistic *probabilistic* model is used to define large-scale outage events, yielding multiple highly-correlated spatial and temporal node and link failures. As noted earlier, these events can include natural disasters, malicious attacks, and cascading power failures/blackouts, etc. Along these lines, the set $\mathbf{U}$ defines an a-priori set of outage events, $\mathbf{U} = \{u_1, u_2, ...\}$, where each event $u_i \in \mathbf{U}$ has an associated occurrence probability, $p(u_i)$. As per [12],[34], it is also assumed that all events are sufficiently rare and therefore can be treated as independent and mutually-exclusive. In other words, only one large scale outage event can occur at any given time:

$$\sum_{\forall u_i \in \mathbf{U}} p(u_i) = 1 \tag{5.3}$$

Also, without loss of generality, it is assumed that all outage events are non-overlapping in the geographic/spatial domain, i.e., a physical node v can only be associated with a single event $u_i \in \mathbf{U}$. Hence a failure probability, $\omega(v)$, is defined for each physical node $v \in \mathbf{V}_s$ that falls in the region of outage event u_i. Namely, $\omega(v)$ represents the failure probability for node v in case outage event u_i occurs. Similarly, a failure probability, $\omega(e)$, is also defined for each physical link $e \in \mathbf{E}_s$ that falls in the region of outage event u_i (with respect to the occurrence of the outage event u_i). These node and link failure probabilities are also assumed to be independent from one another.

5.2 Heuristic Methodologies

As noted in Section 2.6, the generalized multicast tree routing problem with multiple constraints is *NP*-complete [43]. Therefore several heuristic MVN embedding schemes are developed here to provision incoming demands in a rapid "on-line" manner. In particular, these solutions use graph-based algorithms to route multicast trees between the MVN source and terminal nodes. Now the main objective here is to incorporate critical a-priori risk information to improve the reliability of MVN mappings under multi-failure outages, i.e., as per the probabilistic model defined in Section 5.1.3. Specifically, three different heuristic algorithms are proposed with varying objectives, i.e., resource usage minimization, failure risk minimization, and hybrid resource usage and failure risk. In general, all of these schemes follow a common high-level MVN mapping approach. Namely, an iterative search process is used to embed MVN terminal nodes and then interconnect them to the multicast tree subject to demand constraints, i.e., on node resources, bandwidth, delay, and delay variation $(s, \mathbf{D}, \delta, \gamma, r, b)$. Consider some further details here.

Detailed pseudocode listings for the various MVN embedding heuristics are presented in Figure 5.2 and Figure 5.3. The first step basically generates a working copy of the substrate graph by pruning all non-feasible physical nodes and links. Specifically, all substrate nodes in the MVN source and terminal node location sets ($\mathbf{loc}_s$, $\mathbf{loc}_{d_i}$) with insufficient resource levels are removed first, i.e., $R_v^{'} < r$. Next, substrate links with insufficient bandwidth capacity to carry a MVN link connection are also removed, i.e., $B_e^{'} < b$. As a result, only feasible substrate nodes and links are considered for MVN mapping purposes.

Next, the algorithms proceed to compute the MVN tree, which is denoted by the graph $\mathbf{T} = (\mathbf{T}_v, \mathbf{T}_e)$, where $\mathbf{T}_v$ is the set of tree nodes (source, terminal), and $\mathbf{T}_e$ is the set of tree edges. Namely, $\mathbf{T}$ is initialized first ($\mathbf{T}_v, \mathbf{T}_e = \emptyset$), and the MVN root node, s, is mapped to a feasible location v in $\mathbf{loc}_s$. Note that the exact location chosen here will depend upon the particular heuristic scheme (objective). An iterative phase is then initiated, where each step selects a new MVN terminal node to embed and connect (add) to the tree. Specifically, this selection is done by scanning all unmapped terminal nodes $d_i \in (\mathbf{D} - \mathbf{T}_v)$ and computing an appropriate *mapping cost* for all potential feasible embedding locations, i.e., $v \in \mathbf{loc}_{d_i}$ (see nested for loops in Figure 5.2 and Figure 5.3). Specifically, this value is computed as a weighted sum of two terms:

$$cost_m = (\varepsilon)cost_v^n + (1 - \varepsilon)cost_v^t \tag{5.4}$$

where $cost_v^n$ is the node mapping cost, $cost_v^t$ is the tree update cost, and ε is a relative scaling factor between these two costs, $0 \leq \varepsilon \leq 1$. In particular, $cost_v^t$ reflects the cost of the path route to connect the candidate node to the current tree, $\mathbf{T}$. Again, these specific node mapping and tree update costs depend upon the particular heuristic scheme (objective). Overall, node locations $v \in \mathbf{loc}_{d_i}$ with lower mapping costs are selected and added to the multicast tree (main search loops in Figure 5.2 and Figure 5.3). This addition involves inserting the selected node v in to $\mathbf{T}_v$ and adding all physical links for its associated path. In order to ensure MVN demand requirements, all candidate paths must also meet the required delay and delay variation constraints. Finally, cost values are also set to zero for all substrate links that have been added to $\mathbf{T}_e$, i.e., since re-using them does not impose any further resource usage cost. Similarly, risk values are also set to zero for all substrate nodes added to $\mathbf{T}_v$ as well as all substrate links added to $\mathbf{T}_e$. The heuristic schemes are now detailed further.

5.2.1 Minimum Resource Usage (MRU) Embedding

The *minimum resource usage* (MRU) scheme is detailed in Figure 5.2 and attempts to lower the overall resource usage of the MVN embedding (subject to delay and delay variation constraints). Note that this approach does not incorporate any a-priori risk information and hence is used as a baseline for comparison purposes. Now as per the high-level algorithm flow detailed above, the MVN source is mapped first. Namely, the feasible substrate location with the maximum amount of free (available) resources is chosen in order to achieve a level of load balancing across the network, i.e., source node selection done inversely proportional to available resources, $R_v^{'}$ (line 13, Figure 5.2).

Meanwhile, the main iterative loop in Figure 5.2 selects and adds MVN terminal nodes to the multicast tree based upon resource usage. As noted in Eq. 5.4, the mapping cost, $cost_m$, is comprised of two components, i.e., the node mapping cost, $cost_v^n$, and the tree update cost, $cost_v^t$. Now given the focus of the MRU scheme on resource minimization, the former ($cost_v^n$) is computed as inversely proportional to the available resources at the node (akin to MVN source node selection, i.e., line 21, Figure 5.2). Meanwhile, the latter $cost_v^n$ is computed as the hop count length of the (delay-constrained) shortest physical path, $\mathbf{P}_v$, that connects the given node v to the rest of the tree, i.e., $c(\mathbf{P}_v)$ computed by summing link costs along all physical path links using Eq. 5.2. In order to pursue hop count (resource) minimization, all link costs are also set to unity, i.e., $c(e) = 1$. However, in general link costs can be set in an arbitrary manner to either static or variable quantities. For example, some operators may prefer dynamic load-based values to improve traffic engineering efficiency.

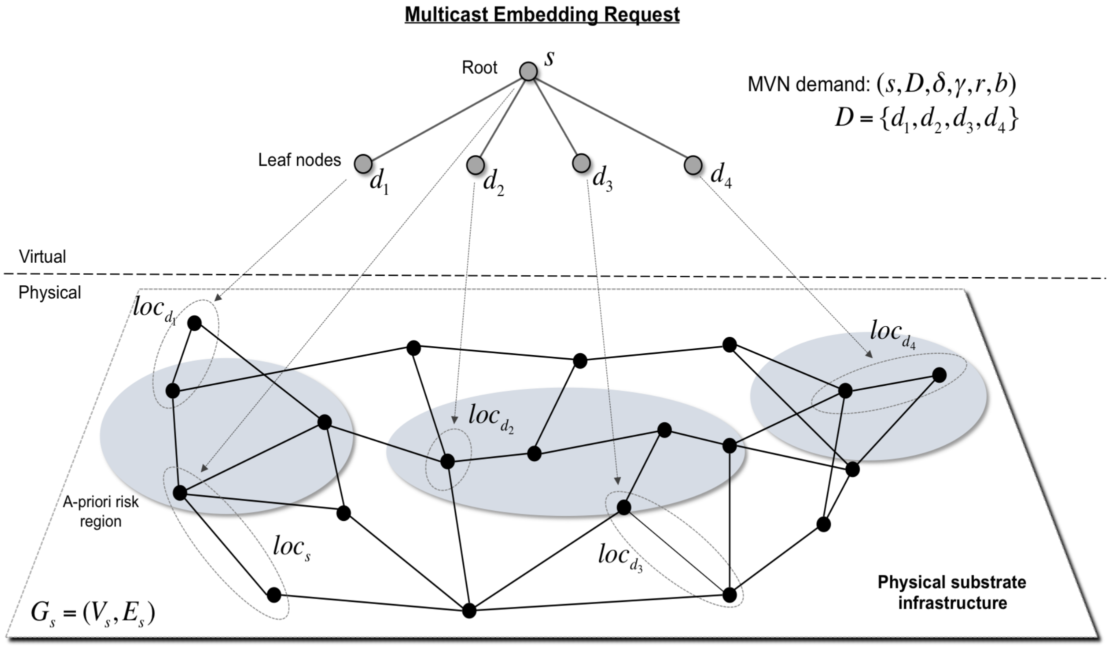


Figure 5.1: Overview of notation for MVN embedding

5.2.2 Minimum Risk Failure (MRF) Embedding

The *minimum risk failure* (MRF) scheme is also detailed in Figure 5.2 and attempts to lower the overall failure risk/vulnerability of embedding a MVN demand (subject to delay and delay variation constraints). Hence this algorithm directly incorporates probabilistic a-priori information on multi-failure outage regions (as defined in Section 5.1.3). Now in order to translate the failure probabilities into additive metrics for use in shortest path computing algorithms, modified logarithmic values are also computed for all node and link failure probabilities, as per [53]:

$$\xi(v) = log\frac{1}{1-\omega(v)} \tag{5.5}$$

$$\xi(e) = log\frac{1}{1-\omega(e)} \tag{5.6}$$

Based upon the above, the overall failure risk of an MVN demand is dependent upon the risks associated with its MVN nodes and MVN link connections. In particular, the risk for an individual MVN node (source, terminal) is set equal to the modified logarithmic failure risk of its mapped substrate node, i.e., $\xi(v)$. Meanwhile the risk for a MVN link is dependent upon the modified logarithmic failure risks of its traversed substrate nodes and links, i.e., path risk:

$$\xi(\mathbf{P}_v) = \sum_{e \in \mathbf{P}} \xi(e) + \sum_{v \in \mathbf{P}} \xi(v) \tag{5.7}$$

Now given the focus on risk mitigation, the MRF scheme first maps the MVN source node to the feasible substrate location with the lowest failure probability (line 15, Figure 5.2). Subsequently, the main iterative loop in Figure 5.2 selects and adds MVN terminal nodes to the existing multicast tree, $\mathbf{T}$, using risk-based node costs. As noted above, the mapping cost in Eq. 5.4 is now equal to the (modified logarithmic) node failure probability, i.e., $cost_v^n = \xi(v)$, line 25, Figure 5.2. Meanwhile, the tree update cost for a given node v in Eq. 5.4 is now equal to the path risk of the (delay-constrained) lowest-risk path, $\mathbf{P}_v$, that connects this node to the rest of the tree, i.e., $\xi(\mathbf{P}_v)$ computed by summing the modified logarithmic failure probabilities along all physical path nodes and links using Eq. 5.7.

5.2.3 Hybrid Resource and Risk (HRR) Embedding

In general, strictly focusing on risk minimization (MRF approach) can give longer, circuitous MVN link routes, resulting in low bandwidth efficiency, i.e., as per earlier studies on risk-aware connection routing [54]. Meanwhile, strictly focusing on resource minimization (MRU approach) can yield less reliable MVN node placements and connection routes, resulting in higher post-fault MVN failures. Therefore a *hybrid resource and risk* (HRR) scheme is also proposed to achieve a balance between these two divergent strategies. The pseudocode for this scheme is presented in Figure 5.3 and further details are now presented.

```
Input: Physical network, $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$, MVN request, $(s, \mathbf{D}, \delta, \gamma, r, b)$
Output: Multicast tree, $\mathbf{T} = (\mathbf{T}_v, \mathbf{T}_e)$
/* Prune location sets for $\{s, d_i \in \mathbf{D}\}$ */
for (each $v^{'} \in \{s, \mathbf{D}\}$)
    for (each $v \in loc_{v^{'}}$)
        if ($R_v^{'} < r$ or $min_{e \in adj_v}\{B_e^{'}\} < b$)
            $\mathbf{loc}_{v^{'}} = \mathbf{loc}_{v^{'}} - v$
    if $\mathbf{loc}_{v^{'}} = \emptyset$ return FAIL
/* Prune substrate links */
for (each $e \in \mathbf{E}_s$)
    if ($B_e^{'} < b$ )
        $\mathbf{E}_s = \mathbf{E}_s -$ {link $e$}
/* Initialize search */
$\mathbf{T} = \emptyset, cost_v^t = \infty$
/* First map MVN source $s$ to node with min. cost or risk */
if (MRU)
    Select $v$ from $loc_s$ with min. $\frac{1}{R_v^{'} + \epsilon}$
else if (MRF)
    Select $v$ from $loc_s$ with min. $\xi(v)$
Map $s$ onto $v$, add to tree, i.e., $\mathbf{T}_v = \mathbf{T}_v \cup$ {node $v$}
/* Iterate and map MVN terminal nodes, $d_i \in \mathbf{D}$ */
while ($|\mathbf{T}_v| != |\mathbf{D}| + 1$)
    for (each $d_i \in (\mathbf{D} - \mathbf{T}_v)$) /* All unmapped MVN nodes */
        for (each $v \in loc_{d_i}$) /* All valid locations */
            if (MRU)
                $cost_v^n = \frac{1}{R_v^{'} + \epsilon}$ /* Node mapping cost */
                Compute $\mathbf{P}_v$ from $T$ to $v$ with min. hop count, $c(\mathbf{P}_v)$ s.t.
                delay and delay variation constraints
                $cost_v^t = c(\mathbf{P}_v)$ /* Tree update cost */
            else if (MRF)
                $cost_v^n = \xi(v)$ /* Node mapping cost */
                Compute $\mathbf{P}_v$ from $\mathbf{T}$ to $v$ with min. failure risk, $\xi(\mathbf{P}_v)$ s.t.
                delay and delay variation constraints
                $cost_v^t = \xi(\mathbf{P}_v)$ /* Tree update cost */
    if $cost_v^t = \infty, \forall v \in loc_{d_i}, \forall d_i \in (\mathbf{D} - \mathbf{T}_v)$ return FAIL
    $cost_m = (\varepsilon)cost_v^n + (1 - \varepsilon)cost_v^t$
    Select $d_i$ with $v$ of min $cost_m$, map $d_i$ onto $v$ and connect it to $\mathbf{T}$ through $\mathbf{P_v}$
    $\mathbf{T}_v = \mathbf{T}_v \cup$ {node $v$}, $\mathbf{T}_e = \mathbf{T}_e \cup$ {link $e$}, $\forall e \in \mathbf{P_v}$
    Update $c(e) = 0$, $\xi(e) = 0$, $\xi(v) = 0$, $\forall e, v \in \mathbf{P_v}$
return $\mathbf{T} = (\mathbf{T}_v, \mathbf{T}_e)$
```

Figure 5.2: Min. resource usage (MRU) and min. risk failure (MRF) MVN schemes

```
Input: Physical network, $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$, MVN request, $(s, \mathbf{D}, \delta, \gamma, r, b)$
Output: Multicast tree, $\mathbf{T} = (\mathbf{T}_v, \mathbf{T}_e)$
/* Prune location sets for $\{s, d_i \in \mathbf{D}\}$ */
for (each $v^{'} \in \{s, \mathbf{D}\}$)
    for (each $v \in loc_{v^{'}}$)
        if ($R_v^{'} < r$ or $min_{e \in adj_v}\{B_e^{'}\} < b$)
            $\mathbf{loc}_{v^{'}} = \mathbf{loc}_{v^{'}} - v$
    if $\mathbf{loc}_{v^{'}} = \emptyset$ return FAIL
/* Prune substrate links */
for (each $e \in \mathbf{E}_s$)
    if ($B_e^{'} < b$ )
        $\mathbf{E}_s = \mathbf{E}_s -$ {link $e$}
/* Initialize search */
$\mathbf{T} = \emptyset, cost_v^{t,1} = \infty$
/* First map MVN source $s$ */
Select $v$ from $loc_s$ with min. $\varepsilon(\frac{1}{R_v^{'}+\epsilon}) + (1-\varepsilon)\xi(v)$
Map $s$ onto $v$, add to tree, i.e., $\mathbf{T}_v = \mathbf{T}_v \cup$ {node $v$}
/* Iterate and map MVN terminal nodes, $d_i \in \mathbf{D}$ */
while ($|\mathbf{T}_v| != |\mathbf{D}| + 1$)
    for (each $d_i \in \mathbf{D} - \mathbf{T}_v$) /* All unmapped MVN nodes */
        for (each $v \in loc_{d_i}$) /* All valid locations */
            $cost_v^{n,1} = \frac{1}{R_v^{'}+\epsilon}$ /* Node mapping cost (resource usage) */
            $cost_v^{n,2} = \xi(v)$ /* Node mapping cost (failure risk) */
            Compute $\mathbf{P}_v$ from $\mathbf{T}$ to $v$ with min. $c(\mathbf{P}_v)$ s.t.
            delay and delay variation constraints
            $cost_v^{t,1} = c(\mathbf{P}_v)$ /* Tree update cost (resource usage) */
            $cost_v^{t,2} = \xi(\mathbf{P}_v)$ /* Tree update cost (failure risk) */
            $cost_m^1 = \varepsilon cost_v^{n,1} + (1-\varepsilon)cost_v^{t,1}$ /* Mapping cost (resource usage) */
            $cost_m^2 = \varepsilon cost_v^{n,2} + (1-\varepsilon)cost_v^{t,2}$ /* Mapping cost (failure risk) */
            Sort all $v$ in increasing order of $cost_m^1$ and select first $l$ nodes
            Select $d_i$ with $v$ of min. $cost_m^2$ from the first $l$ nodes, map $d_i$ onto $v$
            $\mathbf{T}_v = \mathbf{T}_v \cup$ {node $v$}
    if $cost_v^{(t,1)} = \infty, \forall v \in loc_{d_i}, \forall d_i \in (\mathbf{D} - \mathbf{T}_v)$ return FAIL
    Compute $k$ shortest paths from $\mathbf{T}$ to the selected node $v$
    Select the path $\mathbf{P}_v$ with min. $\xi(\mathbf{P}_v)$, connect $v$ to $\mathbf{T}$ through $\mathbf{P}_v$
    Update $c(e) = 0$, $\xi(e) = 0$, $\xi(v) = 0$ for all $e, v \in \mathbf{P_v}$
return $\mathbf{T} = (\mathbf{T}_v, \mathbf{T}_e)$
```

Figure 5.3: Hybrid resource and risk (HRR) MVN scheme

Overall, the HRR scheme uses a *weighted* approach to combine failure risks and resource usage costs. For example, the MVN source is first mapped to a feasible substrate node that minimizes a *joint* cost that is comprised of the node failure probability and resource usage (load), i.e., line 12, Figure 5.3. Meanwhile, the main iterative loop also embeds MVN terminal nodes by defining a *joint* mapping cost. In particular, this value is now comprised of two different mapping costs, i.e., $cost_m^1$ and $cost_m^2$. In particular, $cost_m^1$ is defined as follows:

$$cost_m^1 = (\varepsilon)cost_v^{n,1} + (1-\varepsilon)cost_v^{t,1} \tag{5.8}$$

where $cost_v^{n,1}$ is inversely proportional to the resource usage (akin to MRU scheme, see line 17, Figure 5.2), $cost_v^{t,1}$ is equal to the hop count length of the (delay-constrained) shortest physical path, $c(\mathbf{P}_v)$, that connects node v to the rest of the tree (akin to MRU scheme, line 20, Figure 5.2), and ε is a fractional scaling factor, $0 \leq \varepsilon \leq 1$. Similarly, $cost_m^2$ is computed as follows:

$$cost_m^2 = (\varepsilon)cost_v^{n,2} + (1-\varepsilon)cost_v^{t,2} \tag{5.9}$$

where $cost_v^{n,2}$ is equal to the node modified logarithmic failure probability (akin to MRF scheme, see line 18, Figure 5.2), $cost_v^{t,2}$ is equal to the path risk of the (delay-constrained) lowest-risk path, $\xi(\mathbf{P}_v)$, that connects node v to the rest of the tree (akin to the MRF scheme, see line 21, Figure 5.2), and ε is defined above. However, the HRR scheme differs slightly in its selection of the final mapping location of a MVN terminal node. Namely, it ranks all nodes in increasing order of their $cost_m^1$ values, i.e., resource-based node mapping and tree update costs. The top l nodes are then searched to find the one with the minimum $cost_m^2$ value for final mapping, i.e., risk-based node mapping and tree update costs. The k-shortest paths are then computed from the selected node to the tree, and the path with minimum risk is selected.

5.2.4 Complexity Analysis

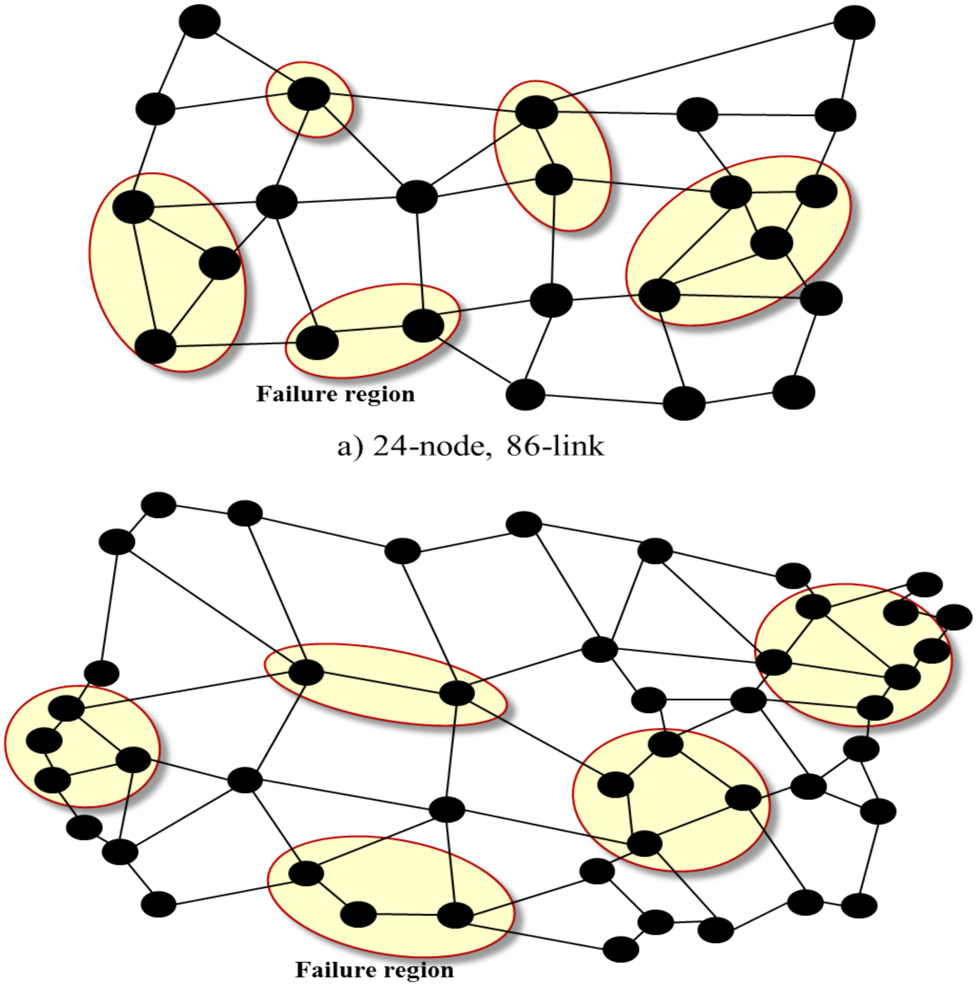
Consider the run-time complexity of the proposed heuristic schemes. Foremost, the complexity of the MRU and MRF algorithms is dominated by the loop procedures to select the MVN terminal node and build the multicast tree, Figure 5.2. Namely, both schemes loop through all valid locations for all terminal nodes, and for each, find a feasible minimum cost path to nodes in the existing tree. This main loop is iterated until all MVN terminal nodes are embedded and hence yields a complexity of $O(|\mathbf{D}|^3|\mathbf{loc}_d|)$. Given a shortest path computation complexity of $O(|\mathbf{E_s}|log|\mathbf{V_s}|)$, the total complexity of the MRU and MRF algorithms is bounded by $O(|\mathbf{D}|^3|\mathbf{loc}_d||\mathbf{E_s}|log|\mathbf{V_s}|)$.

Meanwhile, the complexity of the HRR scheme can also be analyzed in a similar manner. Namely, the overall process follows the same iterative process of checking all MVN terminal nodes and locations, see Figure 5.3. However, the k-shortest path algorithm is now executed for each selected location, and this has a complexity of $O(k|\mathbf{E}_s|log|\mathbf{V}s||\mathbf{D}|)$. Hence this yields a total HRR algorithm complexity bound of $O((|\mathbf{D}||\mathbf{loc}_d| + k)|\mathbf{D}|^2|\mathbf{E}_s|log|\mathbf{V}s|)$.

5.3 Performance Evaluation

The proposed MVN embedding heuristics are evaluated using custom-developed models in $OPNETModeler^{TM}$. Again tests are done using the same 24-node/86-link and 46-node/152-link topologies from Chapter 3 and 4, and five different failure regions are defined for each, see Figure 5.4. All substrate nodes and links have 100 units of resource capacity and 10,000 units of link bandwidth, respectively. Meanwhile, MVN requests are generated randomly with 4-7 and 7-12 nodes each (source and terminals) for the 24-node and 46-node topologies, respectively. The respective location subsets for these MVN nodes are also selected randomly. Namely, each MVN source and terminal node specifies a location set with two substrate nodes, and without loss of generality, these sets are considered to be disjoint. Furthermore, all MVN nodes require 1-10 units of capacity, and each MVN link requires

50-1,000 units of bandwidth. Meanwhile, the delay and delay variation constraints are set to 6 and 3, respectively, and all MVN requests have infinite durations with exponential inter-arrival times (mean 60 seconds). The multi-failure outage events are triggered randomly as per their pre-defined occurrence probabilities. Accordingly, substrate nodes and links that belong to the selected failure region are failed independently as per their pre-defined failure probabilities. All tests are done for medium-to-heavy load services and results are also averaged over 10 independent runs.



a) 24-node, 86-link

b) 46-node, 152-link

Figure 5.4: Test network topologies

5.3.1 Blocking Rate

The MVN blocking rate is a critical metric for operators as it is indicative of the load-carrying capacity of a particular scheme. Hence the blocking rates for the 24-node and 46-node networks are plotted in Figure 5.5 and Figure 5.6, respectively. The overall results indicate that the MRF scheme gives the highest blocking since it strictly focuses on risk minimization and ends up routing longer detour link connections. As such, this approach cannot provide good operator revenues. By contrast, the MRU scheme gives the lowest blocking rates as it focuses on resource efficiency concerns. Meanwhile, the HRR scheme, which considers both risk and resource efficiency concerns, yields a very good tradeoff between the two algorithms. In particular, this scheme closely tracks the blocking rates for the MRU scheme in the 24-node topology, i.e., within 5% deviation at heavy loads (Figure 5.5).

5.3.2 Revenue

The net revenue performance for the various schemes is also measured here. As per [34], this value is computed based upon the revenue, cost, and disruption penalty across all demands. Namely the revenue generated by provisioning a MVN demand tree, **T**, is [22]:

$$REV(\mathbf{T}) = \psi \sum_{e \in \mathbf{T}_e} b\mathbf{I}(e) + (1-\psi) \sum_{v \in \mathbf{T_v}} r\mathbf{I}(v) \tag{5.10}$$

where $\mathbf{I}(e)$ and $\mathbf{I}(v)$ are the revenue per unit of bandwidth and node resource, respectively, and ψ is a relative scaling factor for bandwidth and node resource revenues, $0 \leq \psi \leq 1$. Meanwhile, the cost of mapping the tree **T** is also given by [22]:

$$COST(\mathbf{T}) = \pi \sum_{e \in \mathbf{T}_e} \chi_e^{\mathbf{T}} * C(e) + (1-\pi) \sum_{v \in \mathbf{T}_e} \Upsilon_v^{\mathbf{T}} * C(v) \tag{5.11}$$

where $\chi_e^{\mathbf{T}}$ is the amount of bandwidth allocated to tree link e, $\Upsilon_v^{\mathbf{T}}$ is the amount of node resource allocated to tree node v, $C(e)$ and $C(v)$ are the unit bandwidth and unit node resource costs, and π is a relative scaling factor, $0 \leq \pi \leq 1$. Finally, the service disruption penalty associated with an affected MVN request is also computed as [34]:

$$P(\mathbf{T}) = \varrho \sum_{e \in \mathbf{T}_e} b(e) * \mathbf{P}(e) + (1 - \varrho) \sum_{n \in \mathbf{T}_v} r(n) * \mathbf{P}(n) \tag{5.12}$$

where $\mathbf{P}(e)$ and $\mathbf{P}(n)$ are the unit node and link penalty costs, respectively, and ϱ is a relative scaling factor for the link and node penalties, $0 \leq \varrho \leq 1$. Based upon the above terms, the net revenue is computed by summing across all required demands [34]:

$$REV_N(\mathbf{T}) = \frac{\sum(REV(\mathbf{T}^i) - COST(\mathbf{T}^i)) - \sum_j P(\mathbf{T}^j)}{Time}, \forall \mathbf{T}^i \in \mathbf{A}_{mvn}, \forall \mathbf{T}^j \in \mathbf{F} \tag{5.13}$$

where $\mathbf{A}_{mvn}$ is the set of successfully-mapped (accepted) MVN demands, $\mathbf{T}^i$ is the i-th MVN tree in $\mathbf{A}_{mvn}$, $\mathbf{F}$ is the set of affected MVN trees, $\mathbf{T}^j$ is the j-th MVN tree in $\mathbf{F}$, and $Time$ is the total simulation run-time.

The net revenues are shown in Figure 5.7 and Figure 5.8 for the 24-node and 46-node topologies, respectively. These findings confirm that the MRF scheme gives the lowest revenues since it is not very resource efficient. By contrast, the MRU scheme achieves the highest revenues. Again the HRR scheme achieves a very good tradeoff between the other two strategies. For example, this method gives roughly identical revenues to the MRU strategy at higher loads in the smaller 24-node topology (Figure 5.7).

5.3.3 MVN Failure Ratio

Finally, MVN failure ratios are also plotted for the heuristic strategies in Figure 5.9 and Figure 5.10 for 24-node and 46-node networks, respectively. In particular, these values represent the fraction of demands affected by the outages. Again, the MRU heuristic gives the lowest reliability compared to MRF and HRR algorithms since it does not take into account any a-priori failure risk information. Meanwhile the MRF algorithm gives the highest reliability, and the HRR scheme achieves a good median between these two schemes.

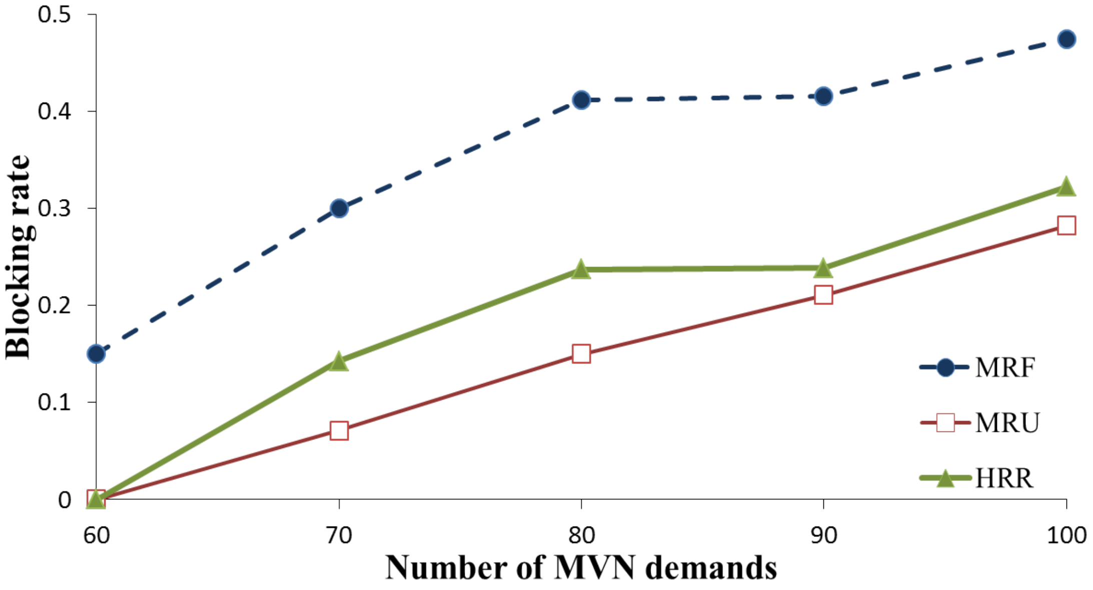


Figure 5.5: Blocking rate for 24-node network

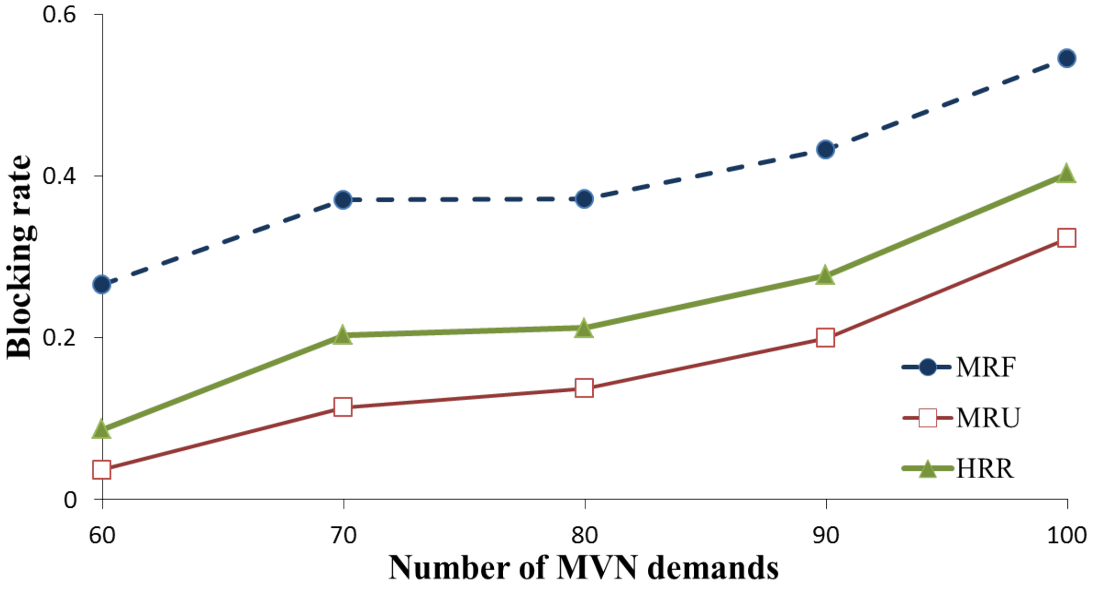


Figure 5.6: Blocking rate for 46-node network

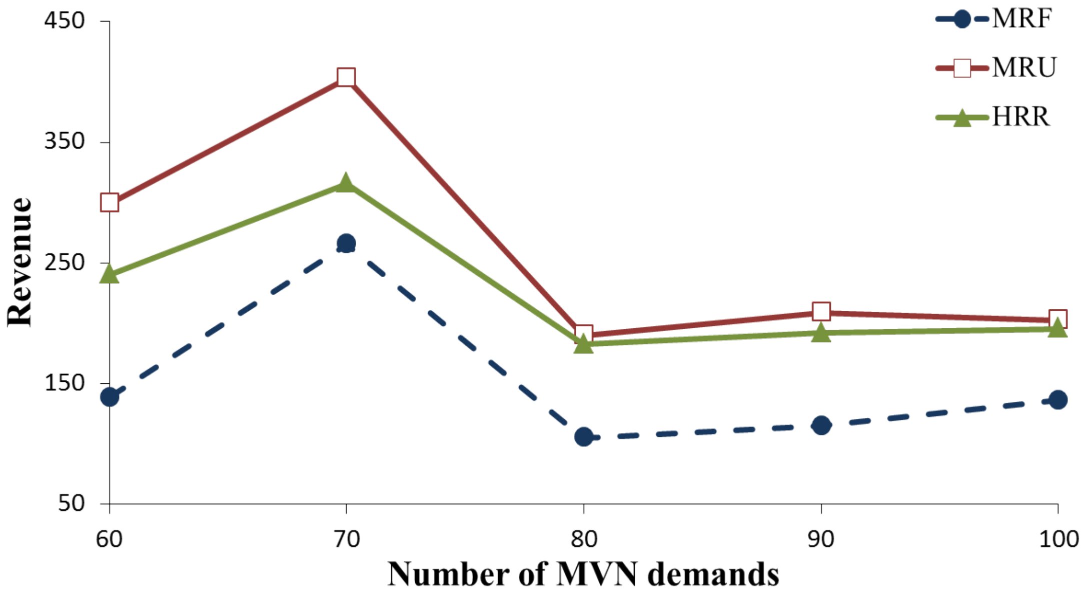


Figure 5.7: Revenue 24-node network

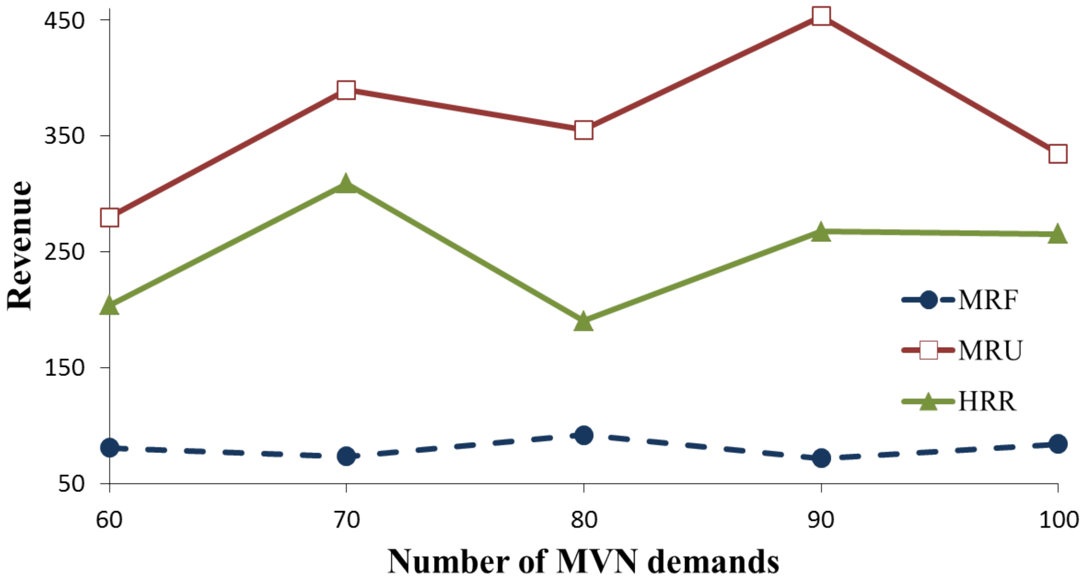


Figure 5.8: Revenue for 46-node network

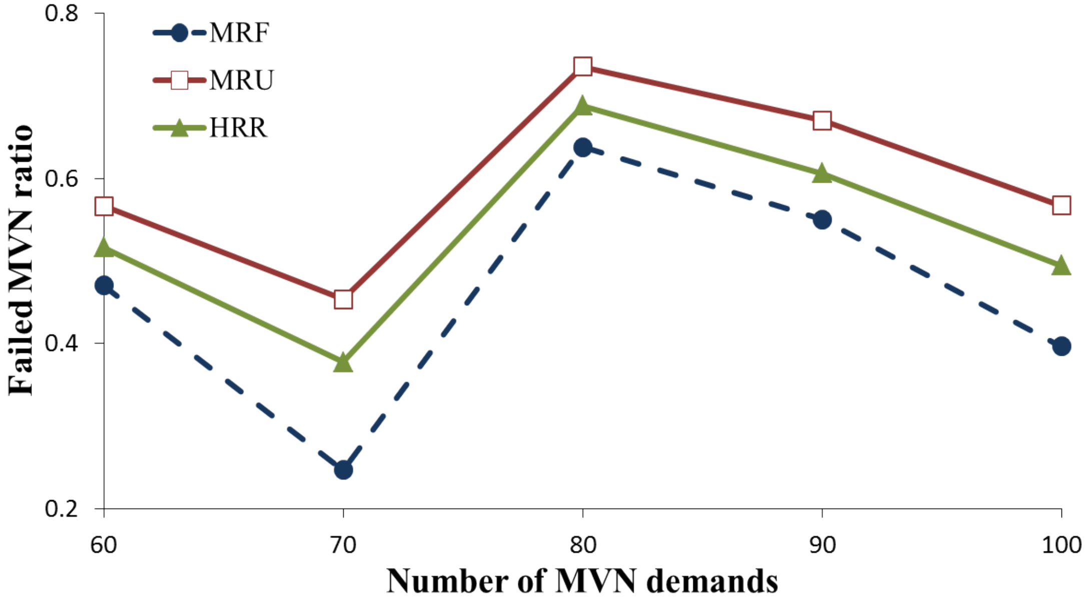


Figure 5.9: Failed MVN ratio for 24-node network

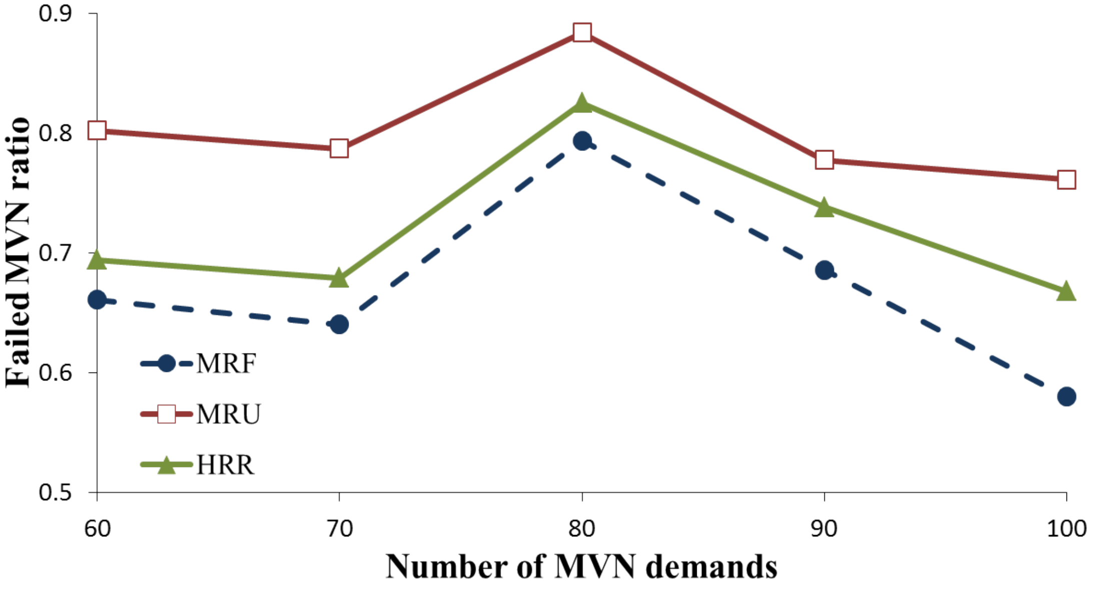


Figure 5.10: Failed MVN ratio for 46-node network

Chapter 6

Conclusions

The research in this dissertation focuses on cloud-based networks and presents a detailed study of post-fault repair strategies and reliable multicast embedding. First, Chapter 2 reviews some related work on *virtual network* (VN) and *multicast VN* (MVN) embedding, with a focus on survivability concerns. The existing body of work in progressive recovery (infrastructure repair) design is also surveyed here. Next, Chapter 3 presents an overall framework for progressive recovery and details a formal optimization-based model. Related metaheuristic solution strategies are also proposed and evaluated here. Building upon this, Chapter 4 presents a series of intelligent polynomial-time heuristic strategies to achieve more scalable operation in real-world settings. Finally, Chapter 5 details improved "risk-aware" mapping strategies for MVN service demands. The major findings and results from this dissertation effort are now summarized here and some important future research directions are also identified.

6.1 Summary of Research Findings

This dissertation initially focuses on the problem of progressive infrastructure repair for cloud-based VN services. To date most studies on VN disaster recovery have focused on pre-provisioned protection strategies. However these methods are resource-intensive and cannot guarantee full recovery at all times. Hence there is a critical need to develop further post-fault recovery schemes. Now most operators will usually re-build their damaged infrastructures in a staged and incremental manner. As a result, careful placement (scheduling) of repair resources is of crucial importance in order to minimize service down times and operator penalties/losses. Along these lines, Chapter 3 introduces a progressive repair framework and presents an in-depth optimization modeling of the repair scheduling problem. This setup uses a MINLP approach and tries to maximize the number of recovered demands in each stage by taking into account finite, i.e., limited, amounts of repair resources and partial VN demand recovery. Nevertheless this formulation is not readily-solvable owing to extreme intractability. Therefore more feasible metaheuristic strategies are also proposed based upon SA methods. These latter schemes use the same objective function as the optimization model and implement an iterative search process to generate and evaluate randomly-modified system state (resource placement) vectors. Specifically, two different SA renditions are developed here based upon selective (individual node and link) and distributed (multiple node and link) resource allocation strategies. Furthermore VN remapping schemes and VN playback (copy-back) methods are then used to restore portions of failed VN demands. The overall results from the analysis study indicate that:

- The non-linear MINLP optimization model poses very high intractability. For example, this formulation generates over 12,000 variables and 30,000 constraints for a sample 10-node/15-link physical substrate experiencing failure of 100 VN demands (with an average of 4 nodes and 6 links each).

- The performance of the SA schemes is affected by the initial temperature and cooling rate, especially the latter. Smaller values for the cooling rate examine large parts of the search space, leading to very high computational complexity. Conversely, high values for the cooling rate overly restrict the search space, yielding unacceptable performance.

- The distributed SA scheme yields significantly higher VN restoration ratios as compared to the selective SA variant. This improvement is due to the fact that this method distributes repair resources more evenly across the network.

- The VN remapping approach gives notably better recovery in the early repair stages, i.e., by actively re-computing failed mapping portions. However this approach cannot achieve full recovery since VN remapping is performed over a partially-working physical topology. By contrast, the VN playback approach provides full VN recovery by the final stage, but generally gives lower performances in the early-mid recovery stages.

In general, it is difficult to implement optimization (and even metaheuristic) schemes in practical settings owing to their high computational complexity and non-deterministic run times. As a result, network operators will require tractable heuristic schemes to generate acceptable solutions within reasonable timeframes. Hence this dissertation also presents some intelligent polynomial-time resource scheduling schemes for progressive VN recovery in Chapter 4. In particular, these solutions implement resource placement based upon a range of measures, including random, physical network connectivity, virtual load, and required resources. Furthermore, both selective and distributed variants are also developed for the physical network connectivity and virtual load heuristics, akin to the SA schemes. Overall, detailed simulation studies here reveal the following findings:

- All heuristic algorithms (distributed and selective variants) outperform the baseline random scheme in almost all recovery stages in terms of the number of restored VN demands and service disruption penalties, regardless of the VN recovery approach.

- All schemes (including baseline random placement) give noticeably better performance in early stages with the VN remapping approach. Conversely, the VN playback approach gives full recovery by the final stage, i.e., since it simply copies back prior working mappings. However, the improvements with VN remapping are more noticeable with the selective heuristic variants.

- The distributed variants show better performance versus their selective counterparts regardless of the VN recovery approach, i.e., VN remapping or VN playback. This improvement is due to the fact that these methods tend to distribute resources more evenly in the network, and is particularly noticeable for VN playback recovery. The distributed variants also yield better resource efficiency.

- The physical network connectivity and virtual load heuristics show very close performance for both the selective and distributed variants for both VN recovery approaches, i.e., VN remapping and VN playback.

- The S-SA metaheuristic shows better performance compared to the selective physical degree and virtual load heuristics (in terms of recovered VN demands and service disruption penalties). Similarly, the D-SA metaheuristic shows better recovery performance as compared to the distributed physical degree and virtual load heuristics regardless of the VN recovery approach. This approach also gives higher resource usage as it restores more VN demands.

- The SRF scheme only gives improved performance with fewer numbers of affected VN demands, e.g., for the case of smaller network topologies or in later recovery stages in larger network topologies. The other heuristics (physical network connectivity, virtual load) and metaheuristic strategies are more effective otherwise.

Finally, this dissertation also presents some reliable *multicast VN* (MVN) embedding heuristics to handle large-scale multi-failure outage scenarios (Chapter 5). Specifically, the problem is treated as a special case of mapping multicast trees with further constraints on delay and geographic VN node placement. Probabilistic a-priori fault models are also used to incorporate risk vulnerabilities. Some novel MVN embedding heuristics are then proposed based upon resource usage minimization, failure risk minimization, and hybrid resource usage and failure risk. The main findings from this study include:

- The MRU scheme gives the best resource efficiency as it tries to minimize resource usage. However, this approach also gives the lowest post-fault reliability as it does not incorporate any risk vulnerabilities.

- The MRF approach gives the best overall reliability. However, this method also yields very high resource usage, leading to high blocking and reduced revenues.

- The hybrid HRR scheme achieves a good tradeoff between risk minimization (reliability) and resource usage (blocking).

6.2 Future Work

This work presents one of the first comprehensive studies on progressive recovery design for cloud-based infrastructure services as well as “risk-aware” mapping of multicast VN (cloud) services. As such, it provides a very solid basis from which to conduct further exploratory research. Foremost, new efforts can look at resolving the high intractability of the non-linear optimization formulation in Chapter 3, i.e., by applying a range of methodologies such as relaxation methods, dynamic programming, LP approximations, and column generation techniques. The performance of these near-optimum schemes can then be compared with the various heuristic and metaheuristic strategies developed herein to get a better ranking assessment of progressive repair scheduling methods.

Next, further research work can be done in the area of MVN embedding design. Foremost, detailed optimization formulations can be developed to try to bound mapping risks. Specialized progressive recovery schemes can also be investigated for MVN demand recovery. In particular, these service types feature tree-based topologies which embody a natural hierarchical structure, i.e., source and terminal nodes. Hence modified resource placement/scheduling solutions can be developed to recover the most damaged multicast tree branches or terminal endpoints first. Again, both optimization and heuristic-based methodologies can be formulated here.

Finally, the broader datacenter and networking sectors are seeing very notable evolutions with the advent of *software-defined networking* (SDN) and *network function virtualization* (NFV) technologies. In particular, SDN control frameworks are decoupling the data and control planes, eliminating the need for complex distributed routing frameworks and devices. Hence carriers can now deploy highly-customized provisioning and recovery solutions using condensed intelligent controllers running over commodity switches. Meanwhile, NFV concepts are revamping the data plane by reducing the need for dedicated network hardware systems. Specifically, NFV instantiates critical networking functions as software instances running on generalized servers, thereby allowing carriers to build customized service chains for their clients. Hence the overall NFV+SDN combination offers numerous possibilities for progressive recovery repair, i.e., by allowing operators to recreate damaged systems and re-route traffic to effectively reconstitute failed service demands. Overall, this is a very new area with much potential for new contributions.

References

[1] Eric Bauer and Randee Adams. *Reliability and availability of cloud computing*. John Wiley & Sons, 2012.

[2] Md Faizul Bari, Raouf Boutaba, Rafael Esteves, Lisandro Zambenedetti Granville, Maxim Podlesny, Md Golam Rabbani, Qi Zhang, and Mohamed Faten Zhani. Data center network virtualization: A survey. *IEEE Communications Surveys & Tutorials*, 15(2):909–928, 2013.

[3] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. Network virtualization: state of the art and research challenges. *IEEE Communications magazine*, 47(7):20–26, 2009.

[4] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906, 2013.

[5] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, 2010.

[6] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking (TON)*, 20(1):206–219, 2012.

[7] Mahshid Rahnamay-Naeini, Jorge E Pezoa, Ghady Azar, Nasir Ghani, and Majeed M Hayat. Modeling stochastic correlated failures and their effects on network reliability. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE, 2011.

[8] Muntasir Raihan Rahman, Issam Aib, and Raouf Boutaba. Survivable virtual network embedding. In *International Conference on Research in Networking*, pages 40–52. Springer, 2010.

[9] Hongfang Yu, Vishal Anand, Chunming Qiao, and Gang Sun. Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[10] Hongfang Yu, Chunming Qiao, Vishal Anand, Xin Liu, Hao Di, and Gang Sun. Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6. IEEE, 2010.

[11] Gang Sun, Hongfang Yu, Lemin Li, Vishal Anand, Hao Di, and Xiujiao Gao. Efficient algorithms for survivable virtual network embedding. In *Asia Communications and Photonics Conference and Exhibition*, pages 79890K–79890K. International Society for Optics and Photonics, 2010.

[12] Hyang-Won Lee, Eytan Modiano, and Kayi Lee. Diverse routing in networks with probabilistic failures. *IEEE/ACM Transactions on networking*, 18(6):1895–1907, 2010.

[13] Min Zhang, Chunming Wu, Ming Jiang, and Qiang Yang. Mapping multicast service-oriented virtual networks with delay and delay variation constraints. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.

[14] Dan Liao, Gang Sun, Vishal Anand, Hongfang Yu, and Kexiang Xiao. Opportunistic provisioning for multicast virtual network requests. In *Globecom Workshops (GC Wkshps), 2014*, pages 133–138. IEEE, 2014.

[15] Dan Liao, Gang Sun, Vishal Anand, and Hongfang Yu. Efficient provisioning for multicast virtual network under single regional failure in cloud-based datacenters. *TIIS*, 8 (7):2325–2349, 2014.

[16] Abdulaziz M Ghaleb, Tarek Khalifa, Sara Ayoubi, Khaled Bashir Shaban, and Chadi Assi. Surviving multiple failures in multicast virtual networks with virtual machines migration. *IEEE Transactions on Network and Service Management*, 13(4):899–912, 2016.

[17] Jianping Wang, Chunming Qiao, and Hongfang Yu. On progressive network recovery after a major disruption. In *INFOCOM, 2011 Proceedings IEEE*, pages 1925–1933. IEEE, 2011.

[18] Chen Ma, Jie Zhang, Yongli Zhao, and M Farhan Habib. Scheme for optical network recovery schedule to restore virtual networks after a disaster. In *Optical Fiber Communication Conference*, pages M3I–4. Optical Society of America, 2015.

[19] Long Gong, Yonggang Wen, Zuqing Zhu, and Tony Lee. Revenue-driven virtual network embedding based on global resource information. In *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 2294–2299. IEEE, 2013.

[20] Hongyan Cui, Fangjie Kong, and Yunjie Liu. A new algorithm of virtual network embedding based on minimum node stress and adjacent principle. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 792–796. IEEE, 2012.

[21] Sude Qing, Jianxin Liao, Jingyu Wang, Xiaomin Zhu, and Qi Qi. Hybrid virtual network embedding with k-core decomposition and time-oriented priority. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2695–2699. IEEE, 2012.

[22] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, 41(2):38–47, 2011.

[23] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, 2008.

[24] Muntasir Raihan Rahman and Raouf Boutaba. Svne: Survivable virtual network embedding algorithms for network virtualization. *IEEE Transactions on Network and Service Management*, 10(2):105–118, 2013.

[25] Tao Guo, Ning Wang, Klaus Moessner, and Rahim Tafazolli. Shared backup network provision for virtual network embedding. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[26] Bingli Guo, Chunming Qiao, Yongqi He, Zhangyuan Chen, Anshi Xu, Shanguo Huang, and Hongfang Yu. A novel virtual node migration approach to survive a substrate link failure. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference*, pages 1–3. IEEE, 2012.

[27] Chunming Qiao, Bingli Guo, Shanguo Huang, Jianping Wang, Ting Wang, and Wanyi Gu. A novel two-step approach to surviving facility failures. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, pages 1–3. IEEE, 2011.

[28] Bingli Guo, Chunming Qiao, Jianping Wang, Hongfang Yu, Yongxia Zuo, Juhao Li, Zhangyuan Chen, and Yongqi He. Survivable virtual network design and embedding to survive a facility node failure. *Journal of Lightwave Technology*, 32(3):483–493, 2014.

[29] Sara Ayoubi, Yiheng Chen, and Chadi Assi. Towards promoting backup-sharing in survivable virtual network design.

[30] Qian Hu, Yang Wang, and Xiaojun Cao. Towards survivable network virtualization. In *2013 IEEE International Conference on Communications (ICC)*, pages 2246–2250. IEEE, 2013.

[31] Ailing Xiao, Ying Wang, Luoming Meng, Xuesong Qiu, and Wenjing Li. Topology-aware virtual network embedding to survive multiple node failures. In *2014 IEEE Global Communications Conference*, pages 1823–1828. IEEE, 2014.

[32] Oussama Soualah, Ilhem Fajjari, Nadjib Aitsaadi, and Abdelhamid Mellouk. Pr-vne: Preventive reliable virtual network embedding algorithm in cloud's network. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 1303–1309. IEEE, 2013.

[33] Carlos Colman Meixner, Ferhat Dikbiyik, Massimo Tornatore, Chen-Nee Chuah, and Biswanath Mukherjee. Disaster-resilient virtual-network mapping and adaptation in optical networks. In *Optical Network Design and Modeling (ONDM), 2013 17th International Conference on*, pages 107–112. IEEE, 2013.

[34] Feng Gu, Hamad Alazemi, Ammar Rayes, and Nasir Ghani. Survivable cloud networking services. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 1016–1020. IEEE, 2013.

[35] Zhili Zhou, Tachun Lin, and Krishnaiyan Thulasiraman. Survivable cloud network mapping with multiple failures. In *2015 IEEE International Conference on Communications (ICC)*, pages 5491–5496. IEEE, 2015.

[36] Wai-Leong Yeow, Cédric Westphal, and Ulas C Kozat. Designing and embedding reliable virtual infrastructures. *ACM SIGCOMM Computer Communication Review*, 41(2):57–64, 2011.

[37] Feng Gu, Khaled Shaban, Nasir Ghani, Majeed Hayat, and Chadi Assi. Regional failure survivability for cloud networking services using post fault restoration. In *System of Systems Engineering (SoSE), 2013 8th International Conference on*, pages 229–234. IEEE, 2013.

[38] Anisha Mazumder, Chenyang Zhou, Arun Das, and Arunabha Sen. Progressive recovery from failure in multi-layered interdependent network using a new model of interdependency. In *International Conference on Critical Information Infrastructures Security*, pages 368–380. Springer, 2014.

[39] Yangming Zhao, Mohammed Pithapur, and Chunming Qiao. On progressive recovery in interdependent cyber physical systems. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016.

[40] Chen Ma, Jie Zhang, Yongli Zhao, M Farhan Habib, S Sedef Savas, and Biswanath Mukherjee. Traveling repairman problem for optical network recovery to restore virtual networks after a disaster [invited]. *Journal of Optical Communications and Networking*, 7(11):B81–B92, 2015.

[41] Laxman H Sahasrabuddhe and Biswanath Mukherjee. Multicast routing algorithms and protocols: A tutorial. *IEEE network*, 14(1):90–102, 2000.

[42] EN Gilbert and HO Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[43] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[44] Michael R Garey and David S Johnson. Computers and intractability: a guide to the theory of np-completeness. 1979. *San Francisco, LA: Freeman*, 58, 1979.

[45] L Kou, George Markowsky, and Leonard Berman. A fast algorithm for steiner trees. *Acta informatica*, 15(2):141–145, 1981.

[46] Vachaspathi P Kompella, Joseph C Pasquale, and George C Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking (TON)*, 1(3): 286–292, 1993.

[47] George N. Rouskas and Ilia Baldine. Multicast routing with end-to-end delay and delay variation constraints. *IEEE Journal on Selected Areas in communications*, 15(3):346–356, 1997.

[48] Sara Ayoubi, Chadi Assi, Khaled Shaban, and Lata Narayanan. Minted: Multicast virtual network embedding in cloud data centers with delay constraints. *IEEE Transactions on Communications*, 63(4):1291–1305, 2015.

[49] Dan Liao, Gang Sun, Vishal Anand, and Hongfang Yu. Survivable provisioning for multicast service oriented virtual network requests in cloud-based data centers. *Optical Switching and Networking*, 14:260–273, 2014.

[50] Xiujiao Gao, Zilong Ye, Jingyuan Fan, Weida Zhong, Yangming Zhao, Xiaojun Cao, Hongfang Yu, and Chunming Qiao. Virtual network mapping for multicast services with max–min fairness of reliability. *Journal of Optical Communications and Networking*, 7 (9):942–951, 2015.

[51] Sara Ayoubi, Chadi Assi, Yiheng Chen, Tarek Khalifa, and Khaled Bashir Shaban. Restoration methods for cloud multicast virtual networks. *Journal of Network and Computer Applications*, 78:180–190, 2017.

[52] Abdulaziz M Ghaleb, Tarek Khalifa, Sara Ayoubi, and Khaled Bashir Shaban. Surviving link failures in multicast vn embedded applications. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 645–651. IEEE, 2016.

[53] Mahsa Pourvali, Hao Bai, Feng Gu, Khaled Shaban, M Naeini, Jorge Crichigno, M Hayat, S Khan, and Nasir Ghani. Virtual network mapping for cloud services under probabilistic regional failures. In *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, pages 407–412. IEEE, 2014.

[54] Oscar Diaz, Feng Xu, Nasro Min-Allah, Mahmoud Khodeir, Min Peng, Samee Khan, and Nasir Ghani. Network survivability for multiple probabilistic failures. *IEEE Communications Letters*, 16(8):1320–1323, 2012.

Appendix A

Glossary

BFS	Breath First Search
CSP	Cloud Service Provider
DFRDM	Dynamic Failure Region Disjoint Mapping
DFS	Depth First Search
D-PD	Distributed Physical Degree
D-SA	Distributed Simulated Annealing
D-VL	Distributed Virtual Load
DWDM	Dense Wavelength Division Multiplexing
FRGBM	Failure Region Group-Based Mapping
HRR	Hybrid Resource and Risk
IaaS	Infrastructure as a Service
ILP	Integer Linear Programming
InP	Infrastructure Provider
IOCM	Incremental Optimization with Constrained Mapping
IP	Internet Protocol
MCF	Multi-Commodity Flow

MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
MIP	Mixed Integer Programming
MPLS	Multi-Protocol Label Switching
MRF	Minimum Risk of Failure
MRU	Minimum Resource Usage
MST	Minimal Spanning Tree
MVN	Multicast Virtual Network
MVNE	Multicast Virtual Network Embedding
NFV	Network Function Virtualization
NSVIM	Non-Survivable Virtual Infrastructure Mapping
PaaS	Platform as a Service
PD	Physical Degree
QoS	Quality of Service
RD	Random
SA	Simulated Annealing
SaaS	Software as a Service
SDN	Software Defined Networking
SOUM	Separate Optimization with Unconstrained Mapping
S-PD	Selective Physical Degree
SRF	Smallest Request First

SRG	Shared Risk Group
SRLG	Shared Risk Linked Group
S-SA	Selective Simulated Annealing
S-VL	Selective Virtual Load
SVNE	Survivable Virtual Network Embedding
VL	Virtual Load
VM	Virtual Machine
VN	Virtual Network
VNE	Virtual Network Embedding
WMD	Weapons of Mass Destruction

Appendix B

Variable Definitions

A summary listing of all variable definitions is provided here for reference sake.

B.1 MINLP Optimization Variables

The overall MINLP model is presented in Chapter 3. The associated variable definitions are as follows:

$\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$	Physical cloud substrate topology
$\mathbf{V}_s$	Set of datacenter nodes in $\mathbf{G}_s$
$\mathbf{E}_s$	Set of network links in $\mathbf{G}_s$
R_n^{max}	Maximum resource capacity of substrate node $n \in \mathbf{V}_s$
$B_{(m,n)}^{max}$	Maximum bandwidth of substrate link $e = (m, n) \in \mathbf{E}_s$
$\mathbf{G}_a = (\mathbf{V}_a, \mathbf{E}_a)$	VN request
$\mathbf{V}_a$	Set of VN nodes in $\mathbf{G}_a$
$\mathbf{E}_a$	Set of VN links in $\mathbf{G}_a$
$r(a)$	Amount of node resources required by VN node in demand a
$b(a)$	Amount of bandwidth capacity required by VN link in demand a
T_0	Time of initial outage event
T_k	Time of recovery stage k, $k > 0$

$\mathbf{F}_v^k$	Set of affected (eligible) physical nodes in stage k
$\mathbf{F}_e^k$	Set of affected (eligible) physical links in stage k
X_0	Aggregate datacenter node resource loss
Y_0	Aggregate network link capacity loss
$\mathbf{A}$	Set of affected VN demands
a	An affected VN demand in $\mathbf{A}$
$\mathbf{V}_a^{'}$	Set of affected virtual nodes in VN demand a
$\mathbf{E}_a^{'}$	Set of affected virtual links in VN demand a
X_k	Datacenter repair resources in stage k, $k > 0$
Y_k	Link repair resources in stage k, $k > 0$
R_n^k	Current resource level of physical node n in stage k
$R_n^{'k}$	Available resource level of physical node n in stage k
$B_{(m,n)}^k$	Current resource level of physical link (m, n) in stage k
$B_{(m,n)}^{'k}$	Available resource level of physical link (m, n) in stage k
$\beta_{(m,n)}^k$	Amount of link repair resources allocated to link (m, n) in stage k
α_n^k	Amount of node repair resources allocated to node n in stage k
$\rho_n^{p,a,k}$	Indicates if VN node p from a is restored onto physical node n in stage k
$f_{(m,n)}^{(p,q),a,k}$	Indicates if VN link (p, q) is restored onto physical link (m, n) in stage k
$\tau^{a,k}$	Indicates if the VN request a is restored in stage k
$\eta_{(m,n),a}^{(p,q)}$	Indicates if physical link (m, n) is assigned to VN link (p, q) in VN demand a
$\gamma_n^{p,a}$	Indicates if VN node p from VN demand a is mapped onto physical node n

B.2 Simulated Annealing (SA) Metaheuristic Variables

The SA metaheuristics schemes are presented in Chapter 3. The associated variable definitions are as follows:

$prob$	SA acceptance probability
$temp$	SA temperature
Δr	Difference in the objective value between current and new state
$\mathbf{V}^s_{cur}$	Current selective SA node solution vector
$\mathbf{E}^s_{cur}$	Current selective SA link solution vector
$\mathbf{V}^s_{new}$	New selective SA node solution vector
$\mathbf{E}^s_{new}$	New selective SA link solution vector
$\mathbf{V}^d_{cur}$	Current distributed SA node solution vector
$\mathbf{E}^d_{cur}$	Current distributed SA link solution vector
$\mathbf{V}^d_{new}$	New distributed SA node solution vector
$\mathbf{E}^d_{new}$	New distributed SA link solution vector
Δ^v_i	Portion of node repair resources for i-th node in $\mathbf{F}^k_v$
Δ^e_i	Portion of link repair resources for i-th link in $\mathbf{F}^k_e$
α	SA cooling rate
$P(G_a)$	Penalty for VN demand a
$\mathbf{P}(e)$	Unit VN link penalty cost
$\mathbf{P}(n)$	Unit VN node penalty cost

μ	Relative scaling factor to weight $\mathbf{P}(e)$ and $\mathbf{P}(n)$
P_{total}	Long term penalty
$Overhead$	VN restoration overhead
$\mathbf{V}_r$	Set of failed VN nodes that are successfully migrated
$\mathbf{E}_r$	Set of failed VN links that are successfully remapped
ζ	Relative scaling factor to weight $\mathbf{V}_r$ and $\mathbf{E}_r$

B.3 Polynomial-Time Heuristic Variables

The polynomial-time heuristics schemes are presented in Chapter 4. The associated variable definitions are as follows:

$\mathbf{F}_v^{\prime k}$	Candidate node set
$\mathbf{F}_e^{\prime k}$	Candidate link set
w_n	Weight assigned to physical node n in distributed (PD,VL) heuristics
Δ_n	Portion of X_k assigned to physical node n in distributed (PD,VL) heuristics
w_e	Weight assigned to physical link e in distributed (PD,VL) heuristics
Δ_e	Portion of Y_k assigned to physical link e in distributed (PD,VL) heuristics
nd_i	Physical node i degree in original working substrate topology, $\mathbf{G}_s$
VL_i	Virtual load at physical node/link i in original working substrate topology, $\mathbf{G}_s$
r^a	Total amount of resources (node and link) required by affected VN demand a
Γ	Relative scaling factor to weight node and link required resources

B.4 Multicast VN (MVN) Embedding Variables

The MVN embedding heuristic schemes are presented in Chapter 5. The associated variable definitions are as follows:

$R_v^{'}$	Available resources of physical node v
$B_e^{'}$	Available bandwidth of physical link e
$c(e)$	Cost associated with link e (static, additive)
$d(e)$	Delay associated with link e (static, additive)
$c(\mathbf{P})$	Cost associated with path $\mathbf{P}$
$d(\mathbf{P})$	Delay associated with path $\mathbf{P}$
$\xi(\mathbf{P})$	Modified probabilistic risk associated with path $\mathbf{P}$
$(s, \mathbf{D}, \delta, \gamma, r, b)$	MVN demand tuple
s	MVN source node
$\mathbf{D}$	Set of MVN terminal nodes
d_i	MVN terminal node in $\mathbf{D}$
δ	Maximum delay bound for MVN demand
γ	Maximum delay variation bound for MVN demand
r	MVN node resource requirement
b	MVN bandwidth capacity requirement
$\mathbf{loc}_s$	Set of physical location nodes for MVN source node s
$\mathbf{loc}_{d_i}$	Set of physical location nodes for MVN terminal node d_i

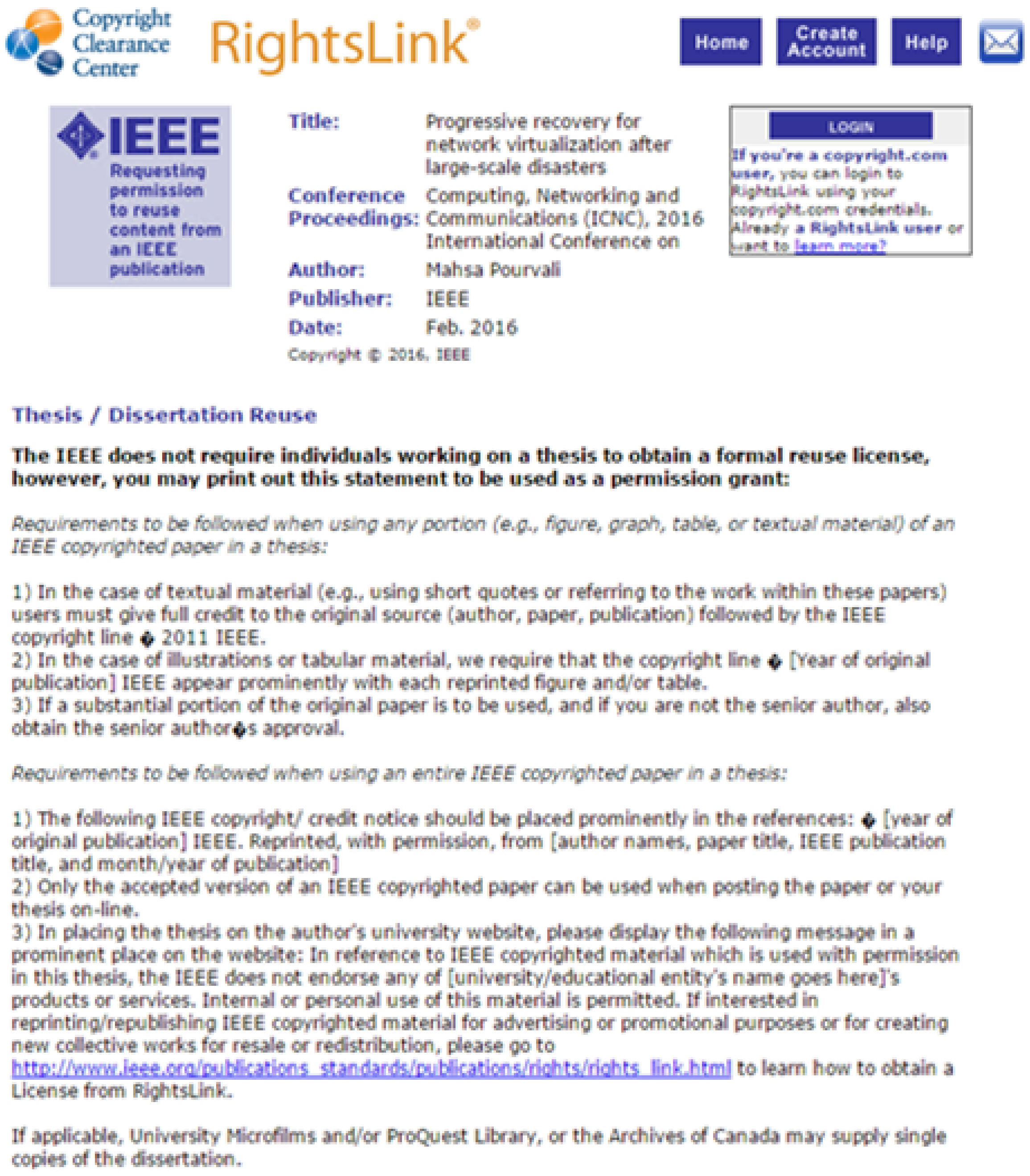
$\mathbf{U}$	Set of a-priori outage events
u_i	Outage event i
$p(u_i)$	Occurrence probability of outage event u_i
$\omega(v)$	Conditional failure probability for physical node v
$\omega(e)$	Conditional failure probability for physical link e
$\xi(v)$	Modified logarithmic failure probability for physical node v
$\xi(e)$	Modified logarithmic failure probability for physical link e
$\mathbf{T} = (\mathbf{T}_v, \mathbf{T}_e)$	Multicast tree computed for MVN demand
$\mathbf{T}_v$	Set of physical nodes in $\mathbf{T}$
$\mathbf{T}_e$	Set of physical links in $\mathbf{T}$
$delay_{min}^{T}$	Minimum source-terminal delay in $\mathbf{T}$
$delay_{max}^{T}$	Maximum source-terminal delay in $\mathbf{T}$
$cost_m$	MVN demand mapping cost in MRU and MRF schemes
$cost_{m,1}$	First MVN demand mapping cost in HRR scheme
$cost_{m,2}$	Second MVN demand mapping cost in HRR scheme
$cost_v^n$	MVN node mapping cost for substrate node v in MRU and MRF schemes
$cost_v^t$	MVN tree update cost for substrate node v in MRU and MRF schemes
$cost_v^{n,1}$	First MVN node mapping cost for substrate node v in HRR scheme
$cost_v^{n,2}$	Second MVN node mapping cost for substrate node v in HRR scheme
$cost_v^{t,1}$	First MVN tree update cost for substrate node v in HRR scheme

$cost_v^{t,2}$	Second MVN tree update cost for substrate node v in HRR scheme
ε	Relative scaling factor to weight node mapping and tree update cost
$REV(\mathbf{T})$	Revenue generated by provisioning multicast tree $\mathbf{T}$
$\mathbf{I}(e)$	Revenue per unit of bandwidth
$\mathbf{I}(v)$	Revenue per unit of node resource
ψ	Relative scaling factor to weight bandwidth and node resource revenues
$COST(\mathbf{T})$	Cost of mapping multicast tree $\mathbf{T}$
$\chi_e^{\mathbf{T}}$	Amount of allocated bandwidth to multicast tree link e
$\Upsilon_v^{\mathbf{T}}$	Amount of node resource allocated to multicast tree node v
π	Relative scaling factor to weight node and link allocated resources
$P(\mathbf{T})$	Penalty for affected multicast tree $\mathbf{T}$
$\mathbf{P}(e)$	Unit node link penalty cost
$\mathbf{P}(n)$	Unit link penalty cost
ϱ	Relative scaling factor to weight node and link penalty costs
$REV_N(\mathbf{T})$	Net revenue for multicast tree $\mathbf{T}$
$\mathbf{A}_{mvn}$	Set of successfully-mapped MVN demands
$\mathbf{T}^i$	The i-th multicast tree in $\mathbf{A}_{mvn}$
$\mathbf{F}$	Set of affected multicast trees
$\mathbf{T}^j$	The j-th multicast tree in $\mathbf{F}$
$Time$	Total simulation run-time

Appendix C

Copyright Permissions

Below are the permissions for the use of material in Chapters 3 and 4 of this dissertation.



Requesting permission to reuse content from an IEEE publication

Title: Progressive recovery for network virtualization after large-scale disasters

Conference Proceedings: Computing, Networking and Communications (ICNC), 2016 International Conference on

Author: Mahsa Pourvali

Publisher: IEEE

Date: Feb. 2016

Copyright © 2016, IEEE

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

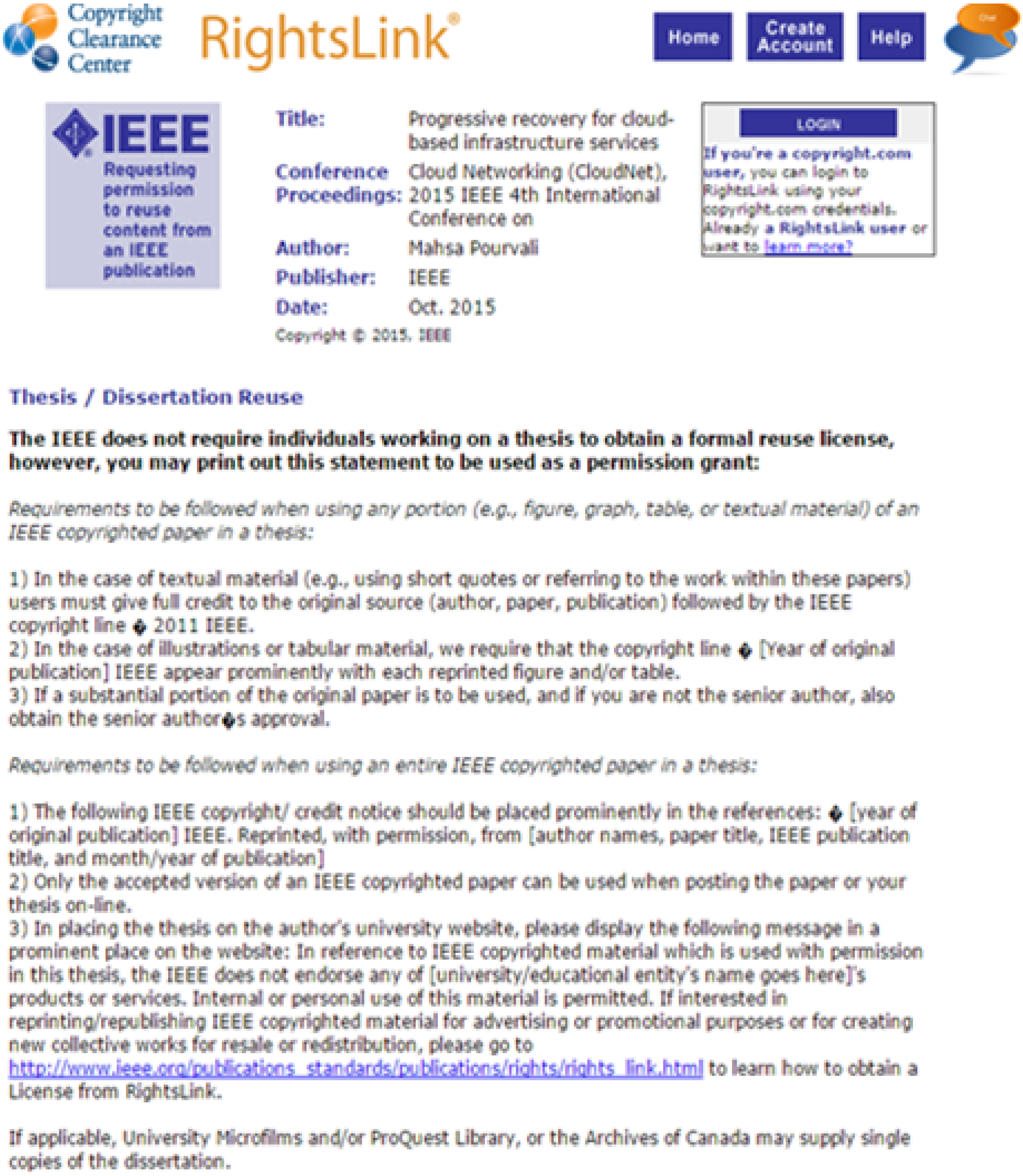
Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line � 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line � [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author�s approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: � [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.



Copyright Clearance Center
RightsLink®
Home
Create Account
Help
IEEE
Requesting permission to reuse content from an IEEE publication
LOGIN

Title:	Progressive recovery for cloud-based infrastructure services
Conference Proceedings:	Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on
Author:	Mahsa Pourvali
Publisher:	IEEE
Date:	Oct. 2015

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. **Already a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line � 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line � [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author�s approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: � [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

About the Author

Mahsa Pourvali obtained her Bachelor of Science degree in Computer Engineering from Azad University of Lahijan (Guilan, Iran) in 2008. She obtained her Master of Science degree in Computer Engineering from Ferdowsi University of Mashhad (Khorasan Razavi, Iran) in 2012. She started her doctoral studies in 2014, in the Department of Electrical Engineering at the University of South Florida under the supervision of Prof. Nasir Ghani. Her research interests include cloud networks, network virtualization, and network resiliency.