11-16-2016

Scheduling Design for Advance Virtual Network Services

Hao Bai
University of South Florida, haobai@mail.usf.edu

Scheduling Design for Advance Virtual Network Services

by

Hao Bai

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.
Huseyin Arslan, Ph.D.
Srinivas Katkoori, Ph.D.
Biswanath Mukherjee, Ph.D.

Date of Approval:
October 27, 2016

Keywords: Network Virtualization,
Advance Reservation, Cloud Scheduling

Dedication

To my parents, for all their unconditional love

Acknowledgments

I would like to acknowledge and dedicate my gratitude to the following people who made the completion of this work possible: My supervisor, Dr. Nasir Ghani, for his continuous help and advisement through my Masters and Ph.D. study.

My dissertation committee members, Dr. Richard Gitlin, Dr. Huseyin Arslan, Dr. Srinivas Katkoori, and Dr. Biswanath Mukherjee, for their valuable suggestions and advice on this dissertation work.

My research colleagues, Dr. Feng Gu, Ms. Mahsa Pourvali, Mr. Ayan Ghayas, Mr. Diogo Oliveira, for their help and collaboration in my research.

All my friends, for their great support and encouragement.

And most especially my parents and my wife, for their endless love and support.

Table of Contents

List of Tables

List of Figures

Abstract

Network virtualization allows operators to host multiple client services over their base physical infrastructures. Today, this technique is being used to support a wide range of applications in cloud computing services, content distribution, large data backup, etc. Accordingly, many different algorithms have also been developed to achieve efficient mapping of client *virtual network* (VN) requests over physical topologies consisting of networking infrastructures and datacenter compute/storage resources. However as applications continue to expand, there is a growing need to implement scheduling capabilities for virtual network demands in order to improve network resource utilization and guarantee *quality of service* (QoS) support.

Now the topic of *advance reservation* (AR) has been studied for the case of scheduling point-to-point connection demands. Namely, many different algorithms have been developed to support various reservation models and objectives. Nevertheless, few studies have looked at scheduling more complex "topology-level" demands, including virtual network services. Moreover, as cloud servers expand, many providers want to ensure user quality support at future instants in time, e.g., for special events, sporting venues, conference meetings, etc.

In the light of above, this dissertation presents one of the first studies on advance reservation of virtual network services. First, the fixed virtual overlay network scheduling

problem is addressed as a special case of the more generalized virtual network scheduling problem and a related optimization presented. Next, the complete virtual network scheduling problem is studied and a range of heuristic and meta-heuristic solutions are proposed. Finally, some novel flexible advance reservation models are developed to improve service setup and network resource utilization. The performance of these various solutions is evaluated using various methodologies (discrete event simulation and optimization tools) and comparisons made with some existing strategies.

Chapter 1 Introduction

This dissertation studies advance reservation scheduling of virtual network services in cloud-based infrastructures. This chapter presents an overview of some recent developments in the field along with the key motivations for the work. Subsequently, the major contributions of the research are presented in a high-level manner along with an outline of the rest of the dissertation.

1.1 Background Overview

Recent decades have seen numerous developments in networking technologies, both wired and wireless. These many advances have delivered very high bandwidth scalability at reduced price-points, enabling ultra-high bandwidth connectivity across extended geographic domains, i.e., gigabits over 100-1,000 km ranges. At the same time computing and storage technologies have also seen rapid evolutions and much-improved cost efficiencies. These collective developments have facilitated high-bandwidth interconnection between large-scale storage/computing datacenter sites and led to the emergence of modern *cloud-computing* service paradigms. Namely, cloud-based services use advanced software virtualization techniques (at the server, storage, and network levels) to provision virtualized infrastructures/applications over physical substrate infrastructures consisting of datacenters and interconnecting high-speed networks. This provision allows organizations to "outsource"

their internal datacenter needs to software-based services hosted by external cloud provider organizations, Figure 1.1 [NI01]. Namely, these providers aggregate resources from *infrastructure providers* (InP) to provide end-to-end cloud services for clients and organizations.

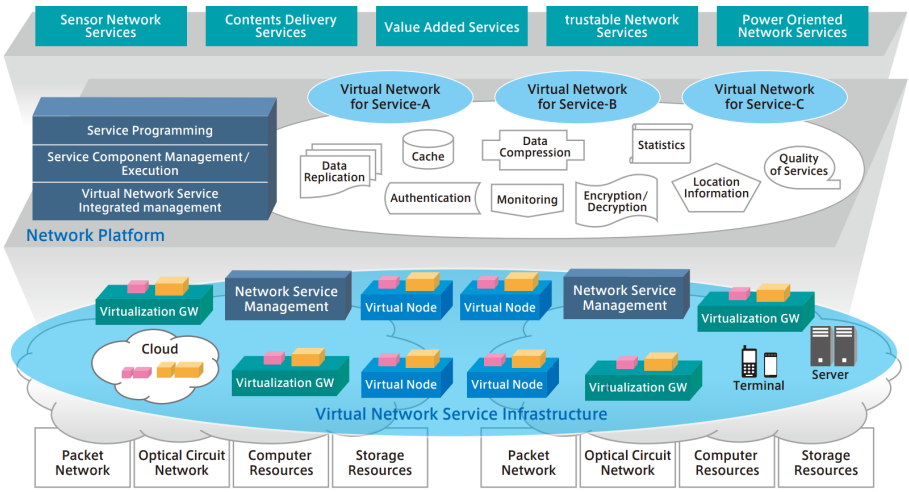


Figure 1.1: Virtual Network Service Infrastructure

Overall, cloud-computing services offer many benefits. Foremost, cloud providers can host multiple customers over a common physical infrastructure, leading to unprecedented cost efficiencies for their clients. Meanwhile, client organizations can heavily reduce (even eliminate) the higher costs associated with maintaining and interconnecting their own internal datacenter facilities, i.e., capital and operational expenditures. Furthermore, cloud users can also leverage abundant datacenter resources to scale their own service offerings, critical for increased market competitiveness. Finally, the distributed nature of the cloud allows client organizations to distribute their resource allocations and achieve much better responsiveness and reliability.

To date several key cloud-computing services have emerged, i.e., *infrastructure-as-a-service* (IaaS), *platform-as-a-service* (PaaS), and *software-as-a-service* (SaaS). Namely, IaaS offerings provide an interconnected set of virtualized storage/computing facilities, i.e., appropriately abstracting out details relating to physical locations, processor types, storage data partitioning, etc. Meanwhile PaaS goes a step further by provisioning a generalized development platform for clients, i.e., operating system, database, web server, etc. Finally, SaaS supports full turn-key applications and databases, i.e., highest level of specialization. Overall, the IaaS model is the closest to the networking layer and offers the most flexibility for users, i.e., in terms of hosting different operating systems, applications, etc. Some popular IaaS services are also shown in Figure 1.2.

Now consider the actual hosting of virtualized infrastructure (IaaS) services in distributed cloud environments. Here, end-user clients/organizations will usually specify a given set of computing and storage resources (to be interconnected across physical cloud substrates) to meet their business needs. On a high level, these demands can be formulated as *virtual networks* (VN) overlays which must be appropriately "mapped" or "embedded" onto underlying substrate infrastructures. Namely, a VN is a set of virtual nodes (VN nodes) interconnected by a set of virtual links (VN links). Here, VN nodes have specific storage and computation resource requirements which must be reserved at the mapped datacenter sites/nodes. Moreover, these mappings may be fixed or variable, depending upon specific customer needs. Similarly, VN links have specific bandwidth and delay requirements which

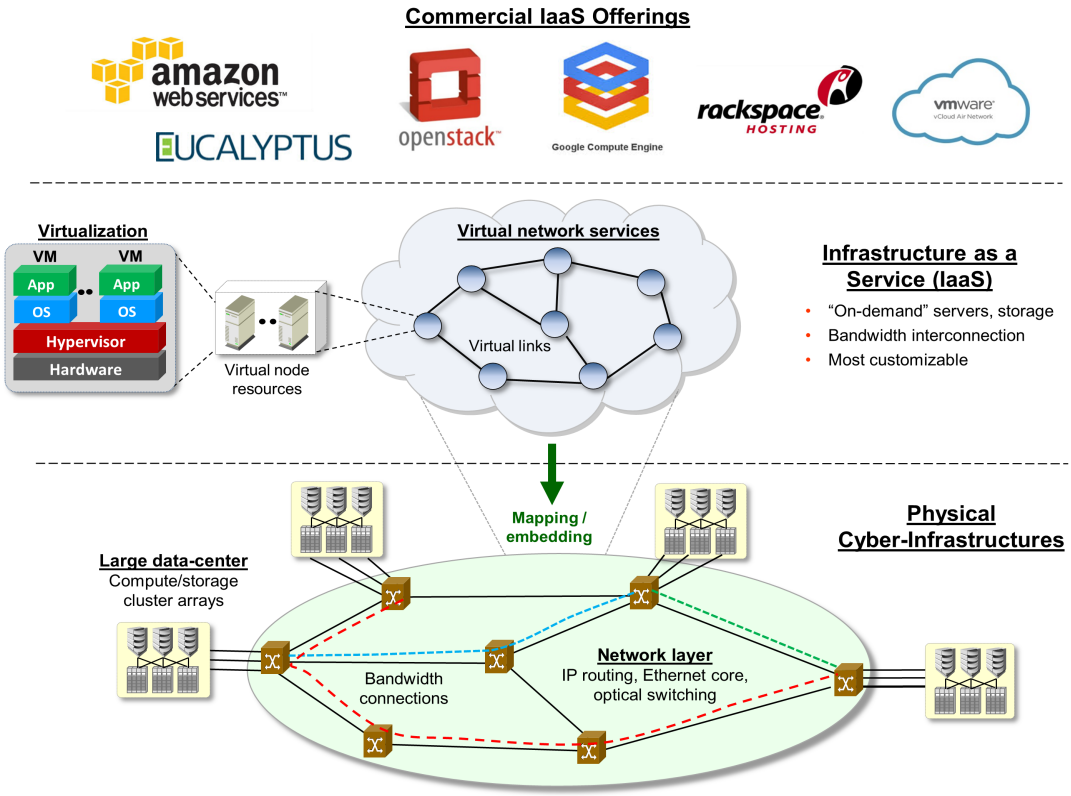


Figure 1.2: Cloud Infrastructure-as-a-Service (IaaS) Model

must be mapped to underlying network layer connections.

In light of the above, cloud providers must carefully provision user requests (VN demands) in order to maximize their revenues and lower costs. Now given the immense interest and focus on cloud-based service provisioning, the VN provisioning problem has been well-studied in recent years, i.e., also termed as *VN embedding* (VNE), see surveys in [AF01]. Additional studies have also looked at survivable VNE designs to improve (IaaS) service reliability under a range of single and multi-failure conditions [MR01]. In general,

these studies have used a wide range of techniques, including optimization, graph-theoretic heuristics, meta-heuristics, and various approximation strategies, see [MA01] for details.

1.2 Motivations

Current research work on network virtualization has focused on "on-demand" service provisioning, i.e., *immediate reservation* (IR). Namely, incoming requests are provisioned immediately based upon the current resource levels in the network. However as user needs continue to evolve, IR models can no longer suffice for all scenarios. For example, some users may want VN resources at very specific future points in time, i.e., fixed start/stop windows. Others may accept some flexibility as to when and how the VN requests are provisioned, e.g., when combined with a corresponding price incentive [LM01]. Indeed, traffic patterns measurements from [DX01] confirm that many popular cloud applications exhibit higher usage during specific intervals. Hence there is a growing need to support more time-sensitive VN demands, where users specify a desired time interval, i.e., *advance reservation* (AR). Namely, a request start time is typically specified at some time in the future and the holding time is finite. In general, this capability will provide new avenues to improve cloud infrastructure resource utilization and user performance guarantees.

Overall, AR scheduling is a well-studied area which has seen some notable contributions in recent years. For example, researchers have developed a range of service models for *point-to-point* (P2P) connection demands with fixed start/stop times, variable start/stop times, and fixed transfer volumes, see survey in [NC02]. Since many of these scheduling sub-

problems are known to be NP-complete, both optimization and heuristics-based solutions have been proposed as well. A few studies have also looked at scheduling more complex service types. For example, [TE01] studies multi-cast connection scheduling in wavelength-routing optical networks, whereas [FG01] introduces the *virtual overlay network scheduling* (VONS) problem to schedule fixed mesh topology overlays.

Building upon the above, this thesis is motivated by the growing need to develop network scheduling algorithms for more generalized cloud services, particularly those relying on VN designs (such as IaaS). These services will play a key role in supporting applications such as cloud backup or mirroring, scientific workflow computing, and event broadcasting (sports, conventions, etc). Although existing VNE schemes can be used as a guide here, the added timeline dimension (coupled with node placement concerns) presents a major challenge here. To date, this problem area remains largely unaddressed and there is significant room for new contributions. This need forms the key motivation for the research in this dissertation.

1.3 Problem Statement

Early AR studies have only looked at scheduling simpler point-to-point connection services, with some limited studies on multi-cast demands [TE01]. However, despite these contributions, few efforts have looked at scheduling more generalized "topology-level" IaaS demands, e.g., such as fixed overlay networks and virtual networks. Indeed there is a growing need to consider these expanded service models within the network scheduling context.

Foremost, new formalized optimization models are required to define the problem space along with scalable solution strategies to derive meaningful performance bounds. A key concern here is to handle high variable count and constraint complexities, i.e., as arising from the added timeline and node placement dimensions (for VN demands). Furthermore, computationally-efficient heuristic strategies also have to be developed in order to provide more viable solutions for realistic "on-demand" processing scenarios. Finally, most existing AR schemes simply provide binary (yes/no) acceptance decisions, leading to higher request rejection rates. Hence, flexible service models also need to be considered in order to improve customer satisfaction and operator's revenue.

To address these concerns, a range of AR schemes are presented for scheduling larger "topology-level" demands, including VN demands with fixed and variable node placement. The proposed strategies include optimization, heuristic, and meta-heuristic algorithms. Associated performances are also analyzed for a range of network testcase scenarios using discrete time event simulation and optimization techniques.

1.4 Proposed Work

This dissertation addresses a range of open issues in the area of VN demand scheduling. In particular, the key contributions here include the following:

1) Novel *integer linear programming* (ILP) optimization models and improved re-routing heuristic algorithms for the special case of virtual overlay network scheduling, i.e., fixed/pre-defined network site mappings.

2) Generalized VN scheduling problem formulation along with a range of meta-heuristic and graph-based heuristic provisioning schemes.

3) Novel advance reservation models for partial virtual network services.

The remainder of this dissertation is organized as follows. Chapter 2 presents a detailed survey of existing advance reservation techniques and related VN embedding schemes. Chapter 3 then studies the fixed overlay scheduling (VONS) problem and introduces some dynamic ILP schemes and improved re-routing strategies. Building upon this, Chapter 4 details the generalized VN scheduling problem formulation, as well as a range of solutions. Finally, Chapter 5 studies novel AR scheduling models to improve blocking performance and network resource utilization. Last but not least, conclusions and directions for future work are presented in Chapter 6.

Chapter 2 Background and Related Work

There is a growing need for scheduling network services at future time intervals. This approach allows providers to arrange/distribute incoming user requests over time and improve overall resource utilization. Service reservation is also critical for supporting live events and broadcasts. Now research in network service reservation has been going on for over a decade, and a range of schemes have been proposed here including heuristic and optimization-based strategies. Hence, this chapter reviews the main AR service models and presents some of the latest work in this area. Open challenges are also presented to motivate the overall research work in this dissertation.

In general, user service requests can be classified as either *immediate reservation* (IR) or *advance reservation* (AR) types. Namely, IR requests require "immediate" resource reservation to commence data transmission shortly after arrival. By contrast, AR requests require data transmission at the future time intervals. Hence AR resources can be reserved upon arrival, but do not need to be allocated until later (service activation). Overall, a range of IR and AR service provisioning schemes have been studied, and some of these works are classified in a high-level manner in Figure 2.1. Further details are now presented.

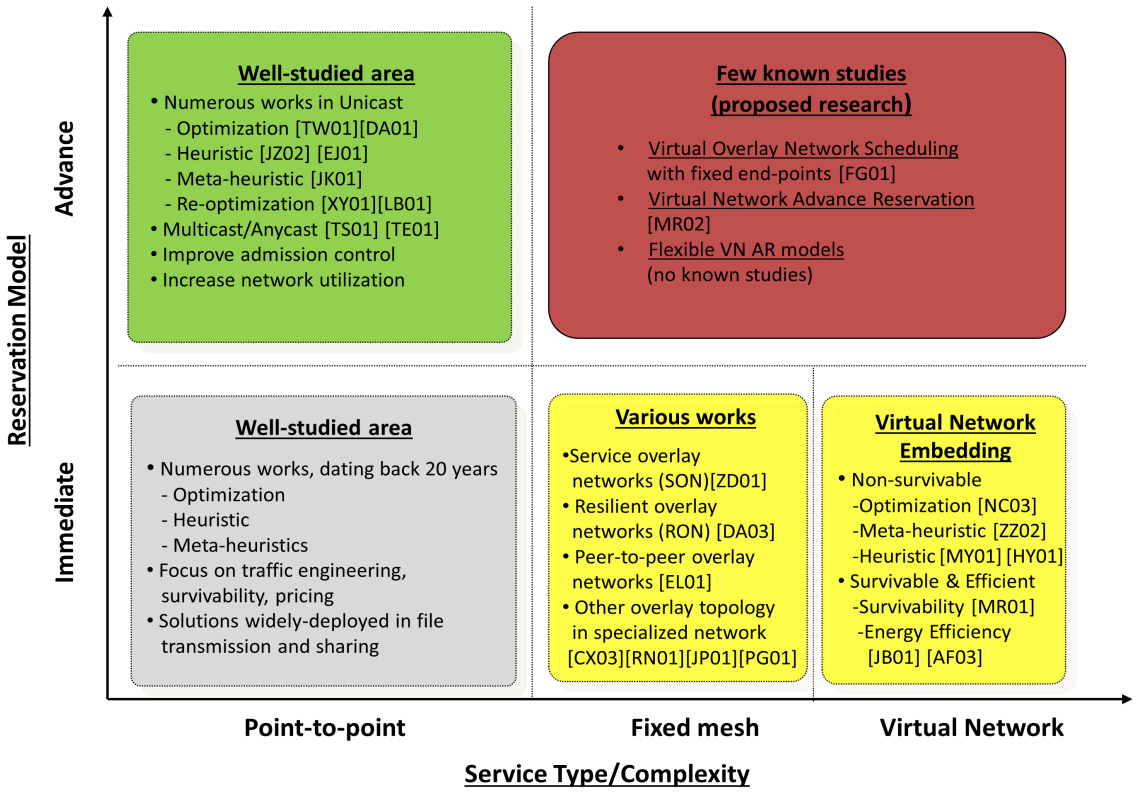


Figure 2.1: A Summary of Reservation Models and Service Type

2.1 Immediate Reservation

In general, IR service reservation is a very well-studied topic area. Namely researchers have looked at provisioning many different demand types, including point-to-point connections, multicast connections, and broader "topology-based" demands, e.g., virtual networks. Some of these contributions are now surveyed.

2.1.1 Point-to-Point Connection Routing

The point-to-point IR model basically requires bandwidth connection provisioning between two nodes in a network. Now given the history of contributions in this area, a full

survey would clearly be out of scope for the focus of this dissertation. In general, many IR studies have used graph-based heuristic methodologies such as Dijkstra's shortest path [ED01] and Eppstein's k-shortest path [DE01] as a basis to provide improved throughput or low latency. Furthermore, many solutions have also developed intelligent dynamic link weighting schemes to improve load distribution/resource efficiency [CC02]. More formal studies have also considered a wide range of optimization techniques to bound network performance when processing batch connection requests. Overall, this problem is termed as the *unsplittable flow problem* (UFP) and is known to be NP-complete. In general, point-to-point IR provisioning is a very mature field and many solutions are already deployed in operational network settings.

2.1.2 Fixed Overlay Network

As application demands have grown, more advanced service models have also been introduced, i.e., beyond basic point-to-point connections. For example, overlay services have been developed to support dedicated bandwidth connections between pre-specified client premise sites/locations. These services are commonly used by larger corporate clients to setup customized topology overlays and support specific application types, e.g., *voice over IP* (VoIP), video streaming, distributed datacenter backup, etc. For example, the earlier *service overlay network* (SON) [ZD01] study presents an optimization model to statically allocate bandwidth resources for overlay topologies (defined as sets of connection routes between gateway nodes). This formulation uses queueing models to incorporate oversubscription

(in terms of load parameters) and pursues a revenue maximization objective by solving an approximate solution. Meanwhile, the *resilient overlay network* (RON) scheme [DA02] adds failure provisions to improve resiliency and IP routing convergence behaviors. Namely, virtual links are provisioned between designated routers to run dedicated link-state routing protocols and achieve various objectives, such as delay or loss minimization, rapid recovery, etc. Overall findings confirm notable reduction in failure recovery times versus the legacy *border gateway protocol* (BGP).

Other studies have also considered overlay topology design for more specialized scenarios. For example, [CX02] presents several graph-based heuristic schemes to map Ethernet *local area network* (LAN) services over *synchronous optical networking* (SONET) and *synchronous digital hierarchy* (SDH) networks using full-mesh, star, and tree overlays. These algorithm also leverage the SONET/SDH inverse-multiplexing capability to implement path splitting and achieve partial (i.e, tiered) service recovery. Meanwhile, other efforts have proposed *optical* overlay provisioning in fiber-based networks with specialized physical-layer constraints, e.g., wavelength continuity, fixed/flexible spectrum, regeneration/amplification, etc [RN01]. For example, [JP01] presents ILP-based schemes for maximizing carried loads with both fixed and flexible spectrum grids. Findings indicate higher carried loads with the flexible spectrum approach, albeit the associated ILP model is less scalable.

2.1.3 Virtual Network

More recently, evolutions in cloud service paradigms have led to a blurring between traditional networking and datacenter domains. Hence evolved *virtual network* (VN) service types have emerged to provision more capable infrastructures, i.e., IaaS model. Namely these offerings comprise of distributed pools of computing/storage resources (VN nodes) interconnected by bandwidth connections (VN links). Clients can essentially leverage these scalable virtualized infrastructures to rapidly deploy their own services and applications across wide geographic domains. Now most VN users are not necessarily concerned about the exact physical location of their resources, i.e., as long as they meet required *quality of service* (QoS) and policy constraints. Hence this flexibility allows VN nodes to be mapped onto a range of potential data-center sites, as opposed to network overlay services where node mappings are fixed/pre-determined by client premise locations. Along these lines, researchers have studied many different *VN embedding* (VNE) schemes. Namely, this procedure involves mapping VN nodes onto data-center sites and routing VN links over substrate network connections. Now earlier work in [NC03] has shown the VNE problem to be NP-hard. Hence most studies have developed optimization and heuristic-based strategies to minimize substrate network resource usages or maximize carrier revenues, see [NC03], [ZZ02], [MY01], [HY01]. Some key contributions are now surveyed.

An early VNE optimization scheme is presented in [NC03] based upon a *mixed integer linear programming* (MILP) approach. Namely, each VN node here is treated as a

"meta-node" with infinite-capacity "meta-links" connecting to all substrate nodes. The optimization scheme then selects a meta-node to effectively "map" the VN node to a substrate node and also route its links. However, due to the intractability of the MILP formulation, the authors also propose two relaxation techniques using rounding algorithms to transform the formulation to a linear *multi-commodity flow* (MCF) problem. Overall results show that the node mapping phase combined with MCF-based link mapping outperforms various existing schemes in terms of acceptance ratios, revenue, and provisioning cost.

In general, finding optimal solutions for large network problems is quite difficult. Hence a wide range of scalable VNE heuristics have also been proposed to find "near-optimal" or acceptable solutions. For example, [ZZ02] uses a discrete particle swarm algorithm to solve the VNE problem. Here the position vector of a particle is defined as a possible VN embedding solution, and the velocity vector of the particle is used to guide the algorithm towards a better solution. The algorithm starts by randomly initializing the position and velocity of one particle within a set of candidate nodes, and then updates them in an iterative manner. If a VN link mapping fails, the position of the particle is re-initialized. After a pre-defined maximum number of iterations, the best position in the swarm is chosen as the optimal VNE solution. This particle swarm method is compared to the best relaxation solution in [NC03] via simulation, and results indicate higher average revenues and acceptance ratios. A more recent study in [SA01] also combines two meta-heuristic techniques, namely the *greedy randomized adaptive search procedure* (GRASP) and *reduced variable neighborhood search*

(RVNS) scheme. Here VNE provisioning is modeled as a GRASP minimization problem to reduce load imbalance (no path-splitting). A feasible solution is then found by solving two sub-problems, i.e., node embedding and link embedding. In order to apply the RVNS meta-heuristic, two neighborhood structures (to re-allocate a virtual link and re-allocate a node with its links) are devised to move to a better solution. Multiple structures are explored iteratively, until no improvement is achieved after five rounds. Overall simulation results show this combined meta-heuristic approach achieves higher acceptance ratios and average revenues as compared to the greedy and rounding scheme in [NC03].

Finally, a number of graph-based algorithms have also been developed for more realistic "on-demand" VNE provisioning scenarios. In general these schemes can be classified into two key categories, i.e., separate node/link mapping (two-stage) and joint node/link mapping (single-stage). Namely, two-stage VN embedding algorithms first map a subset of the VN nodes and then route their VN links. For example, [MY01] first embeds each VN node onto a candidate substrate node with the maximum amount of resources. Next, k-shortest path routing is used to find a path with sufficient bandwidth to route the corresponding virtual links. By contrast, single-stage mapping schemes jointly map VN nodes and all their adjacent VN links. Carefully note that the neighboring VN nodes (for the VN links) must already be mapped here. For example, the algorithm in [HY01] iterates and computes a set of candidate nodes with sufficient node resources and adjacent link bandwidth for each VN node. Each candidate node is then assigned a cost by considering its resources and the

communication costs from this potential location to already-mapped adjacent VN nodes. Finally the substrate node with the minimum cost is chosen for mapping, and adjacent VN link connections are routed using a shortest-path algorithm. Simulation results show that this single-stage scheme provides better resource efficiency and lower blocking versus the two-stage scheme in [MY01].

Finally, many studies have also looked at VNE survivability. For example, this work includes pre-protection provisioning schemes for single node [HY02] and single link [MR01], failure recovery. Others have also looked at more resource-efficient post-fault re-mapping restoration strategies [FG02], as well as large-scale disaster recovery solutions for VNE, see [FG03]. However, these schemes are not surveyed further as they are largely out of scope for the VN scheduling problem herein.

2.2 Advance Reservation

Research work on network scheduling (AR services) started over a decade ago and many contributions have emerged. Compared to traditional IR services, these services require users to provide some time-related bounds/parameters for their requests, i.e., desired arrival/departure times or windows, holding time durations, etc. In general staggering demands over future time intervals can allow operators to achieve improved resource utilization and revenues. Furthermore, users willing to accept delayed service start times can also realize improved QoS support at lower service price points.

Conceptually, the major difference between IR and AR services is the time between the arrival of the request and the start of the service/transmission, termed as *book-ahead time*. In this regard IR can be classified as a "zero book-ahead" AR service. Additionally, the holding times for IR requests are usually not known in advance and/or assumed to be infinite. By contrast, most AR service holding times are known in advance, which allows operators to achieve more efficient resource allocation. Hence AR services can be classified into several types according to their start/stop times or durations see [JZ01]. For example, demands that request a specified start time and duration are denoted as *specified start specified duration* (STSD) Figure 2.2(a). Meanwhile, a slight variation of this scheme is also defined to allow variable start times, i.e., termed as flexible STSD, Figure 2.2(b). Note that this service type can also be represented by an earliest start-time and latest end-time. In general, flexible request timings can provide improved support for large file transfers and result in better network resource usage. Furthermore, [JZ01] also presents a *specified start unspecified duration* (STUD) service which defines a fixed start time but no request duration, Figure 2.2(c). Expectedly, this service model is best suited for users requiring long-steady transfers. Finally, the *unspecified start specified duration* (UTSD) service model specifies a fixed duration but no start times, Figure 2.2(d). These services may suit users who want to achieve a given amount of throughput/transfer in a variable time interval. Overall, most AR studies assume the STSD model.

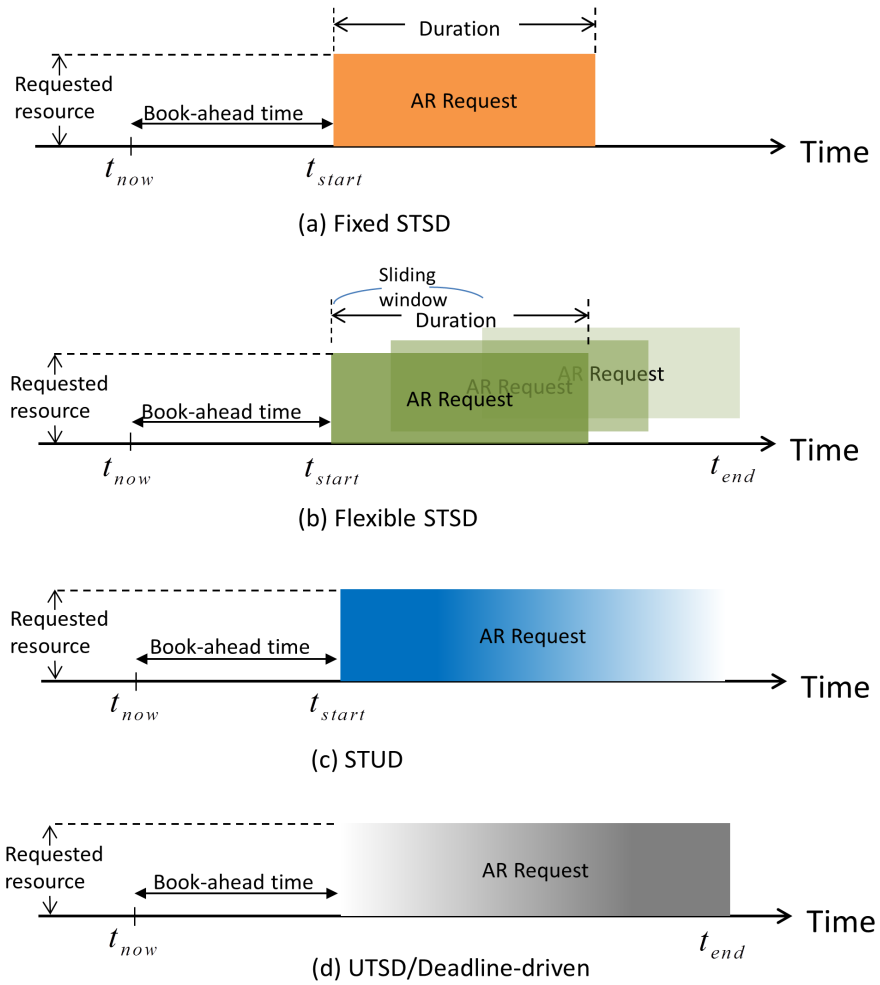


Figure 2.2: AR Service Models

Now earlier work in [RG01] has shown that the basic AR connection scheduling (routing) problem is NP-complete, owing to the additional "time-line" dimension in the provisioning process. As a result most solutions have proposed optimization and heuristic-based strategies to either derive theoretical bounds or improve scalability. Furthermore, studies have also been done for regular bandwidth provisioning, i.e., *multi-protocol label switching*

(MPLS) networks and optical *dense wavelengths division multiplexing* (DWDM) networks. Some of these works are now surveyed.

2.2.1 Point-to-Point Connection Scheduling

Earlier work on network scheduling in [TW01] proposes a novel *mixed integer programming* (MIP) model for optical lightpath connection reservation, i.e., *routing and wavelength assignment* (RWA). Although the requests follow an STSD model, services can start at any time after a basic request's start time. Here the objective function is defined to minimize the time difference between the service start time and the request time. However due to high MIP computational complexity, the optimization is only solved and tested for a small 4 node network. Findings show the MIP scheme can achieve improved acceptance ratios and resource utilization versus other strategies using fixed AR service models. Meanwhile a non-linear scheduling optimization is also proposed in [DA01] to maximize the acceptance ratios for a fixed number of wavelengths. This model tries to jointly solve the request scheduling and RWA problems by using a set of pre-computed paths. Furthermore two heuristic schemes are also proposed, i.e., one using k-shortest path selection and the other using a Lagrangian relaxation of the integer constraints. Overall simulation results show improved resource efficiency with joint scheduling/RWA heuristic versus other "two-step" algorithms.

In general, optimization methods impose sizable computational complexities for larger network sizes or longer lookahead windows. Hence various heuristic AR scheduling algorithms have also been developed. Many of these schemes apply graph-theoretic techniques

and are designed for "on-line" provisioning. For example, [JZ02] presents a simple heuristic for STSD services arriving/departing at discretized time-slots. Namely, a k-shortest path algorithm is used to compute the route for each connection over a reduced graph (that precludes any links that are not available during the required fixed window). However this scheme is not analyzed. Meanwhile, the authors in [EJ01] study the STSD reservation model and assign flexible sliding windows to reduce blocking probability. This model assumes continuous time and maintains reservation state for each link using a time-bandwidth list. A start-time vector is also defined for each wavelength on a link to record its usage at any possible start time. Two initial heuristics are then proposed to find the path with the least start time first, along with two others to find the shortest (min-hop) path in a scheduling window. Overall results show improved performance with the latter scheme.

Finally, some researchers have also proposed meta-heuristic techniques to achieve near-optimal results for the RWA scheduling problem in optical DWDM networks. For example, the authors in [JK01] propose two algorithms for optical networks with wavelength continuity constraints and continuous time requests. Namely, the first algorithm pre-computes k-shortest paths for connections and uses a branch-and-bound algorithm to find the best routes. Meanwhile, the second algorithm uses a tabu-search meta-heuristic to reduce the excessive complexity of the branch-and-bound approach and find a set of feasible routes. A graph coloring heuristic is then used to select the wavelength. Overall results indicate that the meta-heuristic approach gives notably better results than greedy heuris-

tic schemes and even approaches optimal performance (in terms of setup success rates and reduced wavelength usage).

Accepted AR reservations are only active at future time intervals. As a result, these demands can also be re-routed beforehand without causing any service disruptions. Along these lines, researchers have also investigated AR *re-routing* schemes by leveraging from earlier work on IR re-routing and restoration. For example, some early AR re-routing studies have looked at survivable connection requests. Namely, [LB01] proposes a "load-based" re-routing scheme to improve connection survivability against link failures with unknown durations. Results show lower termination/failure rates with increased levels of flow re-routing (but overheads are also higher). Meanwhile, [CX01] considers regular (working-only) AR requests and develops a re-routing algorithm to improve acceptance ratios while reducing the amount of capacity perturbed (during re-routing). Namely, a "dynamic" ILP optimization is formulated for each arriving request with the goal of re-optimizing some existing reservations in conjunction with the incoming request, i.e., maintain a level of "optimally". To improve scalability, this scheme only considers accepted, inactive, time-overlapped requests within a maximum look-ahead time, i.e., versus all demands (global optimization). The authors also propose some additional heuristic schemes which are triggered if a regular scheduling algorithm fails. Namely, these methods identify a subset of inactive time overlapping connections for re-routing and use shortest path computations. Overall results confirm that both of these schemes perform notably better than existing "non-re-routing" strategies.

Finally, studies here have also looked at scheduling more complex multicast connections. For example, [TE01] studies the multicast lightpath RWA problem in optical wavelength routing networks. Here the authors propose two solutions that allow different point-to-point requests in the same multicast request to re-use the same wavelength amongst each other at destination node/any node. Hence multicast connections are setup using multiple "logical" hops instead of being treated independently. This approach prevents a single request from competing with itself for wavelength links. Both ILP optimization and heuristics are proposed here, and overall results show a notable reduction in wavelength usage compared to a naive approach that routes multiple independent unicast connections. Meanwhile, [TS01] studies advance reservation of anycast workflows. The model here assumes deadline-driven service requests, where transfers must be completed before a given deadline. The authors first transform the problem of finding a route and destination node into a multi-cost routing problem and consider both IR and AR requests. For the latter case, an optimal algorithm is proposed to jointly schedule communication (routing) and computation (destination selection) of tasks. The overall algorithm consists of three phases: non-dominated path computation between sources and all the network nodes; candidate path selection; and final path pair selection. However due to exponential complexity, a polynomial-time heuristic is also proposed and evaluated, and findings show that AR scheduling gives notably lower blocking versus IR provisioning (with a trade-off in end-to-end delay).

2.2.2 Virtual Network Service Scheduling

As customer requirements continue to evolve, there is a further need to schedule more complex "topology-level" demands. Key examples here include *virtual overlay networks* (VON) interconnecting a set of network (or datacenter) sites or more flexible *virtual network* (VN) demands with added flexibility of node placement. However, only a handful of studies have looked at actually scheduling these service types. For example, [FG01] introduces the VONS problem for scheduling arbitrary mesh overlay topologies, i.e., sets of virtual links between designated (fixed) network nodes. A "global" ILP formulation is presented here but not solved due to excessive complexity. Instead, some more practical greedy heuristics are presented to handle the "on-line" case, i.e., one incoming VONS request at a time. Also several link weighting schemes are tested here, including static (equal) link weights for achieving resource minimization and dynamic usage-based weights for load-balancing. Overall results show reduced blocking rates with the latter strategy.

Finally, some researchers have done initial work on the more generalized VN scheduling problem. Recall, VN demands impose additional node-level constraints (storage, computation) and hence require placement of VN nodes as well as routing of VN link connections. As a result, [MR02] formulates a *temporal VNet embedding problem* (TVNEP) for a set of a-priori VN requests, each specifying an earliest start time and latest end time (flexible window). The authors introduce the concept of abstract event point model in order to deal with continuous time requests. Namely, each event (VN start or stop time) causes a state change

in the substrate network, which in turn is reflected in the constraints of the optimization formulation. Three different types of state models are also proposed here, i.e., Δ-model representing only state changes, Σ-model representing each state explicitly, and cΣ-model maintaining state for each VN request start event while reducing unnecessary variable numbers from the Σ-model. To further increase scalability, a greedy heuristic based upon the cΣ-model is also proposed. Nevertheless, these algorithms are only evaluated for the limited case of fixed (pre-mapped) VN nodes, i.e, equivalent to VONS problem with sliding window.

2.3 Open Challenges

Although [FG01] and [MR02] present some initial contributions on "topology-level" demand scheduling, many future problems and challenges remain. For example, proper optimization solutions are required for the simpler VONS scheduling problem with static/fixed nodes. Improved heuristics also be developed to extend the baseline greedy schemes in [FG01]. Moreover, generalized VN request scheduling is an even lesser-explored topic area, i.e., in terms of optimization, heuristic, or meta-heuristic solution strategies. Indeed there is much room for new contributions here. Furthermore, VN scheduling can also benefit from *partial* demand provisioning strategies, i.e., as opposed to binary all/nothing provisioning strategies. Hence more flexible provisioning models need to be investigated in order to improve network operator revenues and lower customer blocking rates.

Chapter 3 Virtual Overlay Network Scheduling [1]

As surveyed in Section 2.2, the ability to schedule bandwidth connectivity between multiple sites is becoming an important concern, i.e., as it can benefit applications in cloud computing, data backup, scientific computing, etc. Along these lines, the VONS study [FG01] has looked at scheduling connections between fixed endpoint nodes, equivalent to a special case of VN scheduling. Namely, three basic greedy heuristic schemes are analyzed along with an unsolved ILP formulation.

Building upon this preliminary work, this chapter presents a more detailed study of the VONS problem and provides some improved bounds/benchmarks for overlay network scheduling. Specifically a novel "dynamic" ILP optimization formulation is presented and solved, greatly reducing overall complexity compared to the global optimization in [FG01]. In addition, some more capable heuristics are also developed using *re-routing* strategies to improve resource utilization and acceptance ratios. The work herein provides a good basis from which to develop further generalized VN scheduling schemes for cloud-based services. The details are now presented.

[1] This chapter was previously published in [HB01]. Permission is included in Appendix B.

3.1 Network Model and Description

Consider a substrate network modeled as a graph, $G(V, E)$, where V is the set of router/switches nodes and E is the set of network links. Without loss of generality, all links are assumed to have fixed bandwidth capacity (bandwidth), B, and connectivity is bi-directional, i.e., there are two opposing uni-directional links between neighboring nodes. Each link $e \in E$ also has an associated capacity function, $rem_e(t)$, which represents its available (used) capacity as a function of time (future intervals). Meanwhile an overlay network request is denoted by the 5-tuple, $r^n = (S^n, L^n, t_s^n, t_e^n, b_l^n(t))$, where n is the request index, S^n is the set of nodes/sites ($S^n \subseteq V$), L^n is the set of overlay (virtual) links, t_s^n is the request start time, and t_e^n is the request stop time. Note that this is equivalent to a STSD demand model, as shown in Section 2.1 (Figure 2.2). It is also assumed that all overlay links in a VN$\{l = l_{ij}^n | i, j \in S^n\} \in L^n$ request $b_l^n(t)$ units of bandwidth at time t, $b_l^n \leq B$ and $t_s^n < t < t_e^n$. Technically, the bandwidth request for each overlay link can also be varied as a function of time, i.e., users request different amounts of bandwidth at different times in the request interval. However, to simplify the VONS model, it is assumed that $b_l^n(t)$ is constant during the whole interval $t_s^n < t < t_e^n$, i.e., resulting in the 5-tuple request $r^n = (S^n, L^n, t_s^n, t_e^n, b_l^n)$.

An example of an overlay network is shown in Figure 3.1. Here, two overlay networks (one with 5 nodes and the other with 4 nodes) are mapped onto the 16-node substrate network. Meanwhile, the virtual nodes are already assigned to substrate nodes, and hence

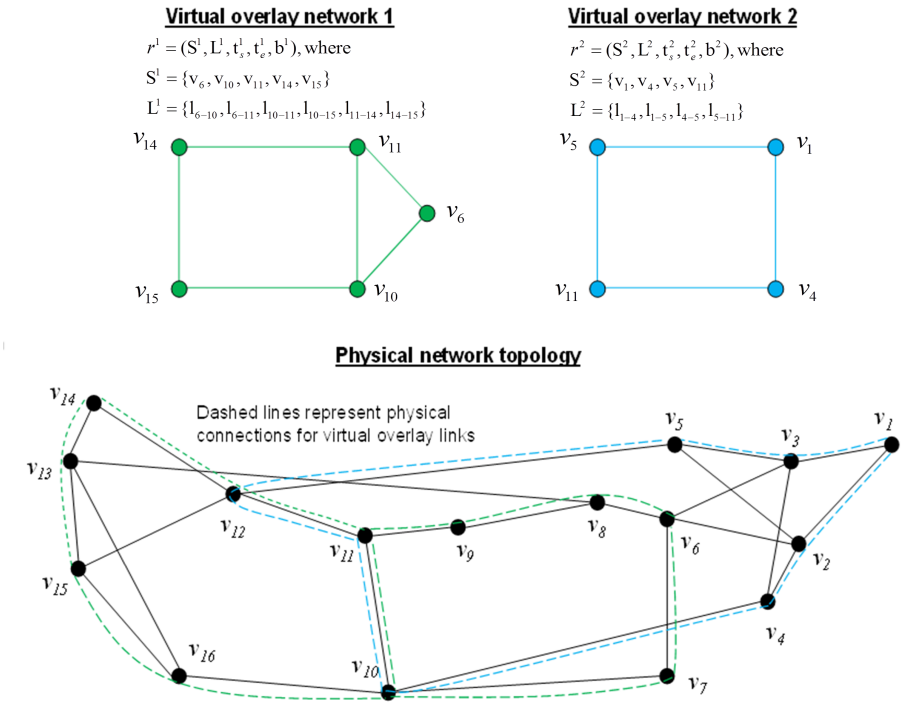


Figure 3.1: Virtual Overlay Network Services Overview

the main objective is to route the link inter-connections between the virtual nodes.

3.2 Optimization Formulation

As noted in [NC02], the point-to-point connection AR scheduling problem is *NP*-complete. Hence by extension the overlay scheduling problem is also at least polynomial-degree higher complexity than *NP*-complete. In light of this, it is very difficult to find tractable global optimization-based solutions, particularly for larger network sizes. For example, [FG01] presents a "global" ILP formulation to schedule a complete batch of a-priori overlay requests, but this model cannot be solved due to excessive variable counts. To ad-

dress these scalability limitations, a novel "dynamic" optimization scheme is now presented for the VONS problem.

Using the notation in Section 3.1, an overlay request is specified as a set of directional connections, i.e., nodes in S^n are interconnected by a set of "virtual" overlay links given in L^n (see examples in Figure 3.1). Carefully note that this formulation only incorporates bandwidth resources, but can also be extended to include data-center (i.e., computing, storage) resources at the nodes. Now in order to satisfy integer constraints, time is discretized into fixed time-slots of duration T, and all t_s^n and t_e^n values are chosen as integral multiples thereof. In addition, several other variables are also defined here as follows:

- r^w: Incoming overlay request: $(S^w, L^w, t_s^w, t_e^w, b_l^w(t))$
- $v_i^n \in S^n$: The i-th node in S^n
- $v \rightarrow e$: Egress node of link $e \in E$ if $v \in V$
- $e \rightarrow v$: Ingress node of link $e \in E$ if $v \in V$
- $R_{t_1}^{t_2}$: Set of admitted but inactive reservations from timeslot t_1 to t_2, i.e., $r^n \in R_{t_1}^{t_2}$ iff start time $t_s^n \geq t_1$ and $t_s^n \leq t_2$
- T_w: Current time at which ILP is triggered and time when request r^w arrives
- T_m: Maximum look-ahead time allowed for ILP run $T_m = \max_n \{t_e^n | r^n \in R_{T_w}^{t_e^w} \cup \{r^w\}\}$
- $L_{i,j}^n$: Virtual link between v_i^n and v_j^n, $v_i^n \neq v_j^n$, $v_i^n \in S^n$, $v_j^n \in S^n$
- $p_{i,j}^{n,e,k}$: Binary flag for overlay link usage in time slot k, i.e., $p_{i,j}^{n,e,k} = 1(0)$ if $l = L_{i,j}^n$ does (not) use link $e \in E$ in timeslot k

Now consider a reduced optimization time window, an example of which is shown in Figure 3.2. Here, T_w denotes the arrival time of the new request r^w, which also happens to be when the ILP computation is triggered. Meanwhile, T_m is defined as the maximum look-ahead time, and this value is set to the maximum stop time across all overlapping inactive reservations in set $R_{T_w}^{t_e^w} \cup \{r^w\}$. Hence any request that starts after request r^w ends will not impact it. Based upon this maximum look-ahead time, the ILP (re)optimization only considers accepted but inactive reservations in the interval $[T_w, T_m]$. For the example in Figure 3.2, only requests r^1, r^2 and r^3 are included in the ILP formulation (but not request r^4). Overall, reducing the number of time-slots in the optimization greatly improves computational scalability, but can also result in a locally-optimal (sub-optimal) solution.

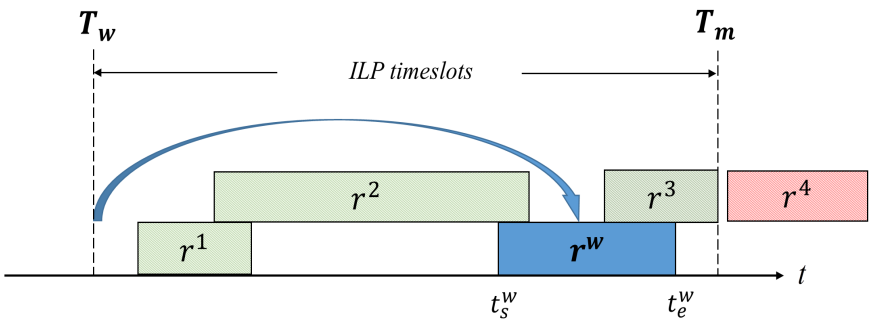


Figure 3.2: Example Set of Admitted Inactive Reservations in ILP

Using the above notation, the overall objective function is defined as:

$$min \sum_{r^n \in R_{T_w}^{t_e^w} \cup \{r^w\}} \sum_{v_i^n \in S^n} \sum_{v_j^n \in S^n} \sum_{e \in E} \sum_{T_w < k \leq T_m} b_l^n p_{i,j}^{n,e,k} \quad \text{(Eq. 3–1)}$$

subject to the following constraints:

$$\sum_{v_i^n \to e} p_{i,j}^{n,e,k} = 1 \quad t_s^n \leq k \leq t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–2)}$$

$$\sum_{e \to v_i^n} p_{i,j}^{n,e,k} = 0 \quad t_s^n \leq k \leq t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–3)}$$

$$\sum_{e \to v_j^n} p_{i,j}^{n,e,k} = 1 \quad t_s^n \leq k \leq t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–4)}$$

$$\sum_{v_j^n \to e} p_{i,j}^{n,e,k} = 0 \quad t_s^n \leq k \leq t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–5)}$$

$$\sum_{e \to v} p_{i,j}^{n,e,k} = \sum_{v \to e} p_{i,j}^{n,e,k}$$

$$t_s^n \leq k \leq t_e^n, v \in V \setminus \{v_i^n, v_j^n\}, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–6)}$$

$$\sum_{r^n \in R_{T_w}^{t_e^w} \cup \{r^w\}} \sum_{v_i^n \in S^n} \sum_{v_j^n \in S^n} b_l^n p_{i,j}^{n,e,k} \leq B$$

$$e \in E, T_w \leq k \leq T_m \quad \text{(Eq. 3–7)}$$

$$p_{i,j}^{n,e,k} = p_{i,j}^{n,e,k+1}$$

$$r^n \in R_{T_w}^{t_e^w} \cup \{r^w\}, e \in E, t_s^n \leq k < t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–8)}$$

Here the objective function in Eq. 3-1 tries to minimize resource utilization across all requests in set $R_{T_w}^{t_e^w} \cup \{r^w\}$, and thereby reduce request blocking rates. Meanwhile, the remaining equations specify some necessary constraints. For example, Eqs. 3-2 and 3-3 (Eqs. 3-4 and 3-5) ensure flow conservation at the respective substrate source (destination) nodes. Meanwhile, Eq. 3-6 ensures input/output flow conservation at the transit nodes. Finally, Eq. 3-7 limits the total provisioned bandwidth on a link to below its maximum capacity, whereas Eq. 3-8 ensures route consistency, i.e., a connection between two overlay nodes must follow the same route in the request interval.

The pseudo-code listing of the dynamic optimization solution is also shown Figure 3.3. The scheme first identifies the set of time-overlapping scheduled reservations to (re)optimize when a new request arrives and then frees up their reserved capacities. To do this, a temporary working copy of the residual bandwidth graph, i.e., $G'(V, E)$, is generated and the maximum look-ahead time window is used to identify the overlapping demands. An ILP formulation is then run for the request along with its set of overlapping reservations over the substrate graph $G'(V, E)$. If this ILP is successful in finding a valid mapping for all reservations in the set, then the request is accepted and the respective resources are reserved in $G(V, E)$. Otherwise the incoming request is dropped.

Overall, the dynamic ILP model greatly reduces run-time complexity versus the "global" ILP formulation in [FG01]. For example, consider a 10-node mesh substrate network with 100 overlay requests. If each request has an average holding times of 10 timeslots

1: Given new overlay request $r^w = (S^w, L^w, t_s^w, t_e^w, b^w)$
2: Identify set of accepted *inactive* reservations $R_{T_w}^{t_e^w}$
3: Generate temporary graph, $G'(V,E)$=$G(V,E)$, remove all resource reservations in $R_{T_w}^{t_e^w}$
4: Generate ILP formulation for set $R_{T_w}^{t_e^w} \cup \{r^w\}$ in $G'(V,E)$
5: **if** ILP solution found
6: Setup success, reserve resources in $G'(V,E)$, copy $G'(V,E)$ to $G(V,E)$
7: **else**
8: Drop request r^w and discard $G'(V,E)$

Figure 3.3: ILP Formulation Framework

and the average request inter-arrival time is 5 timeslots, then approximately 500 timeslots are required for the global optimization scheme in [FG01]. Furthermore, if each overlay request demands 3 nodes and 3 links, the total number of variables in the global optimization is approximately 15,000,000 (i.e., 3 $\times$ 100 total links, 10 $\times$ 10 node-to-node topology, and 500 timeslots). Clearly this value poses insurmountable complexity for most modern servers. Now assume instead that on average, only 2 requests will overlap in time. Hence by using a maximum look-ahead time of 10 timeslots in the dynamic optimization, the total number of optimization variables drops to about 9,000, i.e., 3 $\times$ 2 $\times$ 10 $\times$ 10 $\times$ 15 = 9,000 (i.e., 3 $\times$ 2 total links, 10 $\times$ 10 node-to-node topology, and 15 timeslots). This figure is more than three orders of magnitude lower than that for the global optimization and makes the solution much more tractable.

3.2.1 State-Based Abstract Model

In general, the timeslotted model for AR service demands, i.e., as used in [CX01] and [JZ02], can pose notable concerns for users and network providers. Namely, using longer/more granular time-slots can result in a loss of precision and flexibility for users. For example, defining time-slot durations in days will obviously be less efficient for users only wanting a few hours of service. On the other hand, using smaller time-slot durations will impose high computational complexity when dealing with long-standing requests, i.e., even though user requests remain flat while only substrate network resource levels change.

To overcome these obstacles, a modified state-based abstract model is also developed to further reduce computation complexity while bypassing the time-slot granularity concern. Namely, instead of defining start/stop times-slot intervals, this model only computes state changes at request start/stop times. This is shown further in Figure 3.4 for three sample time-overlapping requests, $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$. Here an event is defined for each AR request start or stop time. Clearly, the resource levels in the underlying substrate network will not change in the interval between any two sequential events here, i.e., regardless of its duration. Hence state changes (events) only need to be defined for specific time-slots on a given link. This implies a total of up to $2|\mathcal{R}| - 1$ states given a total of $|\mathcal{R}|$ overlapped requests. Note that another auxiliary array list is also needed here to map the abstract state timeline to real time in order to reserve bandwidth resources in substrate network (if the ILP is solved). Furthermore, if the start/stop times of one or more requests coincide, then multiple events

can be combined into one. Now consider a modified ILP formulation using this state-based abstract model. Here, overlapping inactive requests will be calculated the same way as before, and the associated states and auxiliary list will be formulated accordingly afterwards. Hence consider some new variables:

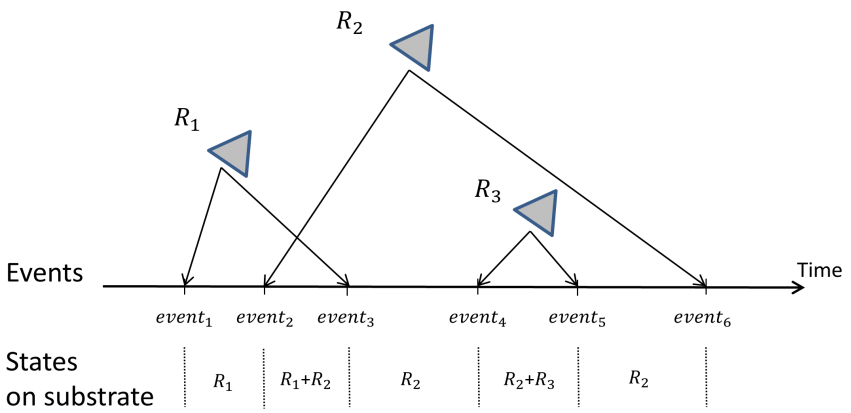


Figure 3.4: State-Based Abstract Model: 3 Requests Mapped to 5 States, 6 Events

- $\mathbb{S} = \{s_1, s_2, ...s_{|\mathcal{R}|-1}\}$: Sets of states abstracted from overlapping requests
- $\mathbb{A}_{r^n}$: Auxiliary state array for overlay request, i.e., $\mathbb{A}^-_{r^n} \in \mathbb{S}$ is the start state for request r^n, and $\mathbb{A}^+_{r^n} \in \mathbb{S}$ is the end state for request r^n
- $p^{n,e,\bar{s}}_{i,j}$: Binary flag for overlay link usage in state $\bar{s}$, i.e., $p^{n,e,\bar{s}}_{i,j} = 1(0)$ if $l = L^n_{i,j}$ does (not) use link $e \in E$ in state $\bar{s}$.
- $\mathbb{B}^{\bar{s}}_{i,j}$: Bottleneck bandwidth (minimum available bandwidth) on link $l = L^n_{i,j}$ during state $\bar{s}$

Based upon this, a new objective function is defined as:

$$min \sum_{r^n \in R^{t^w_e}_{T_w} \cup \{r^w\}} \sum_{v_i^n \in S^n} \sum_{v_j^n \in S^n} \sum_{e \in E} \sum_{T_w < k \leq T_m} b_l^n p_{i,j}^{n,e,\bar{s}} \quad \text{(Eq. 3–1a)}$$

Accordingly the new constraints are:

$$\sum_{v_i^n \to e} p_{i,j}^{n,e,\bar{s}} = 1 \quad \mathbb{A}^-_{r^n} \leq \bar{s} \leq \mathbb{A}^+_{r^n}, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–2a)}$$

$$\sum_{e \to v_i^n} p_{i,j}^{n,e,\bar{s}} = 0 \quad \mathbb{A}^-_{r^n} \leq \bar{s} \leq \mathbb{A}^+_{r^n}, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–3a)}$$

$$\sum_{e \to v_j^n} p_{i,j}^{n,e,\bar{s}} = 1 \quad \mathbb{A}^-_{r^n} \leq \bar{s} \leq \mathbb{A}^+_{r^n}, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–4a)}$$

$$\sum_{v_j^n \to e} p_{i,j}^{n,e,\bar{s}} = 0 \quad \mathbb{A}^+_{r^n} \leq \bar{s} \leq \mathbb{A}^+_{r^n}, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–5a)}$$

$$\sum_{e \to v} p_{i,j}^{n,e,\bar{s}} = \sum_{v \to e} p_{i,j}^{n,e,\bar{s}}$$

$$\mathbb{A}^-_{r^n} \leq \bar{s} \leq \mathbb{A}^+_{r^n}, v \in V \setminus \{v_i^n, v_j^n\}, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–6a)}$$

$$\sum_{r^n \in R^{t^w_e}_{T_w} \cup \{r^w\}} \sum_{v_i^n \in S^n} \sum_{v_j^n \in S^n} b_l^n p_{i,j}^{n,e,\bar{s}} \leq \mathbb{B}_{i,j}^{\bar{s}}$$

$$e \in E, \mathbb{A}^-_{r^n} \leq \bar{s} \leq \mathbb{A}^+_{r^n} \quad \text{(Eq. 3–7a)}$$

$$p_{i,j}^{n,e,\bar{s}} = p_{i,j}^{n,e,\bar{s}+1}$$

$$r^n \in R^{t^w_e}_{T_w} \cup \{r^w\}, e \in E, \mathbb{A}^-_{r^n} \leq \bar{s} < \mathbb{A}^+_{r^n}, v_i^n \in S^n, v_j^n \in S^n \quad \text{(Eq. 3–8a)}$$

Again, the objective function in Eq. 3-1a tries to minimize resource utilization across all considered requests and all abstracted states, just as the previous ILP formulation. Meanwhile,

Eqs. 3-2a - 3-6a ensure flow conservation at the respective substrate source and destination nodes, as well as transit nodes. Also, Eq. 3-7a constrains the total provisioned bandwidth on a link to below its maximum remaining capacity during each state time. Finally, Eq. 3-8a ensures route consistency in the request interval state(s).

The state-based abstract optimization model follows the same pseudo-code as shown in Figure 3.3. Namely if the ILP is successfully solved, the request is accepted and respective resources are reserved. Otherwise the incoming request is dropped and the temporary graph is discarded. Overall, the new state-based model further reduces run-time complexity by replacing the number of timeslots with abstract states. Namely, the total number of variables and computation overheads are now independent of the real duration of each request in ILP formulation. As an example, consider a 10-node mesh topology with 100 overlay requests. With the new state-based model, the request durations do not matter any more, i.e., only the number of overlapping requests count. Hence if only 2 requests (3 nodes and 3 links) overlap in time, the total number of optimization variables will drop from 9,000 to about 1,800, i.e., 3 $\times$ 2 $\times$ 10 $\times$ 10 $\times$ 3 = 1,800 (3 $\times$ 2 total links, 10 $\times$ 10 node-to-node topology, and 2$\times$2-1=3 states). This approach makes it much easier to handle larger request holding times/durations and also increased network sizes. More importantly, the state-based abstract model can also be used as an extension to continuous-time requests, with no need to round continuous time to the nearest timeslot.

3.3 Re-Routing Heuristic

Although dynamic optimization greatly reduces run-time complexity compared to a global optimization, it is still very high. Hence some novel overlay scheduling heuristics are also presented here to improve upon the greedy schemes in [FG01]. The overall goal here is to use re-routing techniques to re-map accepted overlay requests in order to free up resources for new demands, i.e, much in the same way the dynamic ILP operates. This approach is motivated by earlier work on AR connection re-routing, which has shown good blocking reduction, see [CX01]. Consider the details below.

A high-level view of the proposed re-routing framework is shown in Figure 3.5 along with a more detailed psuedocode description in Figure 3.6. Overall, this heuristic is comprised of two stages. The first stage (Stage 1) attempts a regular VONS setup for a new request. If this stage fails, then the second stage (Stage 2) tries to re-route a subset of time-overlapped (virtual link) VONS requests to create enough free resources to accommodate the new request. One of the key objectives here is to achieve a balance between computational complexity (i.e., number of re-routing attempts) and request blocking rates (i.e., network usage utilization). The two stages are now detailed further.

3.3.1 Baseline Scheduling (Stage 1)

The initial stage simply runs a baseline heuristic to setup an incoming VONS request (steps 4-5, Figure 3.6). Although any overlay scheduling heuristic can be re-used here, the load-balancing strategy from [FG01] is chosen as it gives lower blocking (higher carried loads)

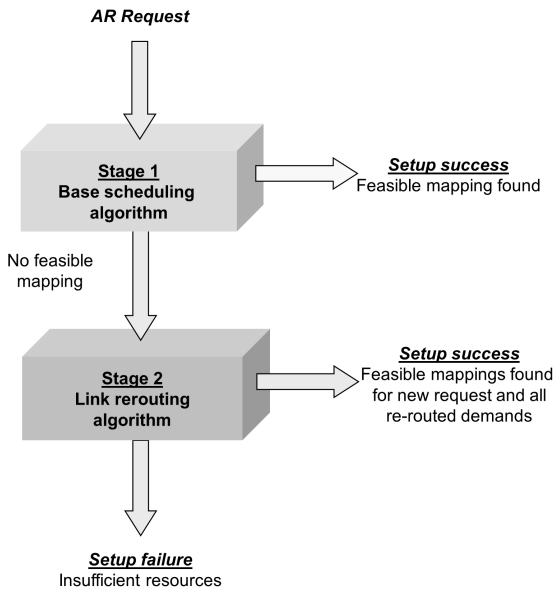


Figure 3.5: Two-stage Overlay Network Scheduling Re-routing Strategy

versus the resource minimization strategy. Specifically, this approach assigns dynamic "load-based" weights to links in $G(V, E)$ and then uses them to compute minimum cost routes for virtual link connections using Dijkstra's algorithm (sequential manner). Namely, the weight of link e in request r^w is computed as inversely-proportional to its bottleneck capacity in the request interval, as follows:

$$\omega_e = 1/(b_e^{min} + \epsilon) \qquad \text{(Eq. 3–9)}$$

where the bottleneck capacity for link e, i.e., b_e^{min} is defined as:

$$b_e^{min} = \min_{t \in [t_s^w, t_e^w]} rem_e(t) \qquad \text{(Eq. 3–10)}$$

where $rem_e(t)$ is the earlier-defined link capacity function, and ϵ is a small value chosen to avoid division errors. Figure 3.7 shows an example of time-varying capacity levels on a link,

1: Given new overlay request $r^w = (S^w, L^w, t_s^w, t_e^w, b^w)$
2: Generate temporary graph, $G^*(V,E)$=$G(V,E)$
/ Loop and provision all overlay links in request */*
3: **for** $j = 1$ to $|L^w|$
4: */* Stage 1: Regular attempt */*
Generate another temporary graph $G'(V,E)$=$G^*(V,E)$, remove all non-feasible links, $b_e^{min} < b^w$ in $[t_s^w, t_e^w]$
5: Run connection scheduling algorithm for j-th virtual link over $G'(V,E)$
6: **if** success
7: Reserve path resources for j-th link in $G^*(V,E)$
8: **else**
9: */* Stage 2: Re-routing */*
Compute candidate path for j-th virutal link in $G^*(V,E)$ via MHR, MNR, or THR approach
10: **if** fail
11: Drop r^w, discard $G'(V,E)$ and $G^*(V,E)$, exit
12: **else**
13: - Sort reservations on candidate path (decreasing order)
14: - Compute re-routing connection set
15: - Free resources for reservations in re-routing set
16: - Reserve path resources for j-th virtual link
17: - Re-route each connection in re-routing set
18: **if** success
19: Reserve candidate path resources in $G^*(V,E)$ for j-th virtual link
20: **else**
21: Drop r^w, discard $G'(V,E)$ and $G^*(V,E)$, exit
22: **if** all overlay (virtual) link connections $l_{ij}^w \in L^w$ routed
23: Setup successs, copy $G^*(V,E) \rightarrow G(V,E)$

Figure 3.6: Overlay Demand Re-routing Heuristic

$rem_e(t)$, as well as the bottleneck capacity, b_e^{min}, if the new request r^w overlaps with some existing requests r^1 and r^2.

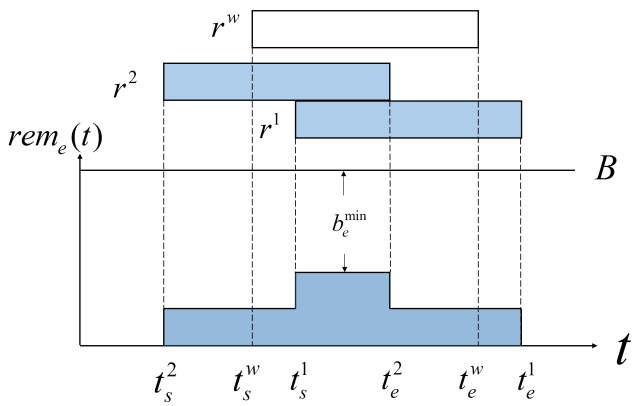


Figure 3.7: Sample Link Capacity Function $rem_e(t)$ and Bottleneck Link Capacity b_e^{min}

Note that all the above computations are done using a temporary copy of the residual bandwidth graph, termed $G^*(V, E)$, see steps 1-2, Figure 3.6. This copy is used to track/store all accepted connection reservations, i.e., virtual links, while the request is being processed. Now if the overall setup is successful, then the original capacity graph $G(V, E)$ is replaced by $G^*(V, E)$, i.e., steps 22-23, Figure 3.6. To further reduce computational complexity, Dijkstra's algorithm only considers substrate links in $G^*(V, E)$ with sufficient capacity to provision the requested bandwidth for r^w, see also Figure 3.7 (step 4, Figure 3.6). Here all links with $b_e^{min} < b^w$ in $G^*(V, E)$ are removed to generate a reduced sub-graph for routing computation, labeled as $G'(V, E)$. This approach tries to achieve better load distribution by preventing specific substrate links from becoming overloaded.

3.3.2 Re-routing Scheduling (Stage 2)

The re-routing stage is triggered if the virtual link connection attempt in Stage 2 is unsuccessful, i.e., steps 9-21, Figure 3.6. Specifically, a candidate path is first computed for

the failed overlay link request. Next, a subset of the existing reservations on the candidate path links are re-routed to free up capacity for the failed request. However since multiple reservations can be perturbed by this re-routing/re-scheduling phase, it is important to limit the number of re-routing attempts, i.e., generally a divergent objective versus blocking reduction. Hence several candidate path selection approaches are proposed here:

- Minimum Hop Re-Routing (MHR): This scheme selects the candidate path with the shortest hop count path between the (failed) overlay link end-point nodes using Dijkstra's algorithm. Choosing the shortest path indirectly tries to minimize re-routing disruption.
- Minimum Number Re-Routing (MNR): This scheme selects a candidate path to minimize the disruption of scheduled demands (re-routing complexity). Namely, the *k-shortest paths* (k-SP) between requested overlay link's end-point nodes are computed, and the path with the fewest number of (virtual link connection) re-routings is chosen. This is done by ordering all connection reservations on a link by decreasing bandwidth and then counting the minimum number needed to meet the requested capacity, b^w.
- Threshold Re-Routing (THR): This scheme tries to minimize the amount of perturbation by selecting a path that already has a fraction ρ ($0 \leq \rho < 1$) of the requested overlay link capacity (in the request interval). Namely, Dijkstra's shortest-path algorithm is re-run over $G^*(V, E)$, and all links with bottleneck capacity below the fractional amount are precluded from consideration, i.e., link e with $b_e^{min} \geq \rho b^w$ in $[t_s^w, t_e^w]$ is kept. This approach is similar to the connection-level AR re-routing scheme in [CX01].

Once a candidate path has been chosen, a subset of the existing (overlay link) connection reservations on its links are selected for re-routing, termed as the *re-routing connection set*. Carefully note that these reservations can include multiple overlay demands. Hence to minimize the disruption of accepted demands, connections on the candidate path (links) are first sorted in terms of decreasing bandwidth size. The re-routing procedure then loops through all candidate path links, and for each, iteratively moves a sufficient number of scheduled connections (with time-overlapping durations) to the re-routing connection set to free up capacity. Specifically, for each iteration at a candidate path link, the bottleneck link bandwidth, b_e^{min}, is recomputed until enough capacity is freed for the new request (step 15, Figure 3.6).

Finally, the heuristic scheme tries to sequentially re-schedule all reservations in the re-routing connection set (steps 17-21, Figure 3.6). This step basically re-runs the regular Stage 1 setup algorithm (from Section 3.3.1) for each request over the temporary $G^*(V, E)$ graph. If all reservations in the re-routing connection set can be successfully re-scheduled, then re-routing is deemed successful and the request is accepted. Otherwise, the request is rejected and the setup attempt terminated.

3.3.3 Complexity Analysis

Now consider the overall run-time complexity of the heuristic approach, starting with Stage 1. Foremost, the bottleneck link capacity filtering step is of $O(|E|)$ complexity. Meanwhile, Dijkstra's shortest path algorithm (used in the min-distance scheme) is

of $O(|E| + |V|log(|V|))$ complexity [TC01]. Hence the aggregate run-time complexity for scheduling *each* virtual link is $O(|E| + |V|log(|V|))$.

Next consider the re-routing stage, Stage 2. The first step focuses on candidate path selection where both the MHR and THR schemes use Dijkstra's shortest path algorithm to compute a candidate path, i.e., $O(|E| + |V|log(|V|))$ complexity, akin to Stage 1. However, the MNR strategy is more involved since it first computes k-shortest paths and then selects one with the smallest number of overlay links to re-route. This latter step requires sorting all reservations along each of the k-shortest paths. Now k-SP computation is of $O(|E| + |V|log(|V|) + k)$ complexity [DE01]. Hence in the worst case, the MNR scheme may have to process all $O(|E|)$ links in $G^*(V, E)$, and each link may have up to $O(|V|^2)$ reservations [CX01]. Sorting these reservations in decreasing order of bandwidth size adds an additional $O(|E||V|^2log|V|)$ complexity, yielding an aggregate $O(|E| + |V|log(|V|) + |E||V|^2log|V|)$ bound for the MNR scheme.

Finally, to compute the re-routing connection set, both the MHR and THR schemes have to sort the previously-scheduled reservations in decreasing order, i.e., $O(|E||V|^2log|V|)$ complexity. However since the MNR scheme already performs this sorting step earlier, it has *constant* time complexity here. Leveraging the above, the total number of re-routing attempts (across all three candidate path selection strategies) is upper-bounded by $O(|V|^2)$, i.e., the same as the maximum number of reservations on each link. This yields a computational complexity bound of $O(|V|^2|E| + |V|^3log|V|)$ for all three heurisitcs. In practice,

however, the number of active connections on a link will be well below $|V|^2$, and this will give much lower run-time complexity. These overall bounds are summarized in Table 3.1.

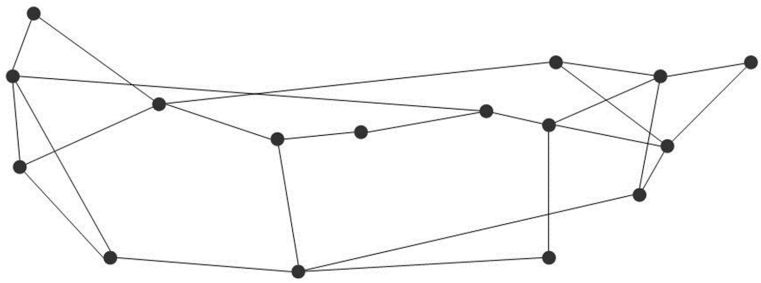
Table 3.1: Heuristic Complexity Comparison

Heuristic		MHR	MNR	THR
Stage 1		$O(\|E\| + \|V\|log\|V\|)$		
Stage 2	Candidate path selection	$O(\|E\| + \|V\|log\|V\|)$	$O(\|E\| + \|V\|log(\|V\|) + \|E\|\|V\|^2 log\|V\|)$	$O(\|E\| + \|V\|log\|V\|)$
	Re-routing connection set selection	$O(\|E\|\|V\|^2 log\|V\|)$	$O(1)$	$O(\|E\|\|V\|^2 log\|V\|)$
	Re-routing computation	$O(\|V\|^2\|E\| + \|V\|^3 log\|V\|)$		

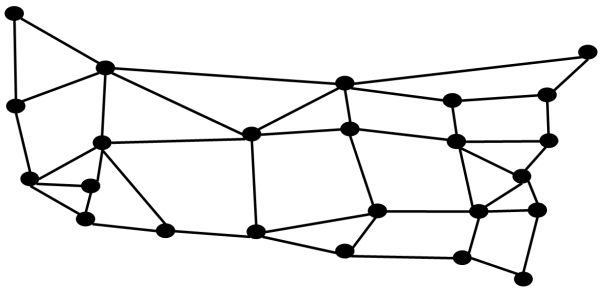
3.4 Performance Evaluation

The performance of the various overlay scheduling schemes is now analyzed using an Intel(R) $Core^{TM}$ i5-4300U CPU @1.90GHz server with 8.00GB RAM. Namely, advanced discrete event simulation models are developed in the $OPNET\ Modeler^{TM}$ toolkit to generate and process overlay requests/demands. Meanwhile, the dynamic optimization model (Section 3.2) is also solved by generating external file-driven calls to the *CPLEX* optimization solver tool. To further prevent this optimization tool from consuming all usable memory or taking too long to find an optimal solution, at most 2.00 GB memory is allocated and the maximum time limit for each iteration of the optimization is limited to 600 seconds. This constraint implies that a request is turned down if an optimization solution is not found in time. Moreover, two topologies are tested here, including the NSFNet backbone with 16 nodes/25 links (3.12 node degree) and a larger network with 24 nodes/43 links (3.58 node degree),

see Figure 3.8. All nodes are assumed to be regular packet-switching routers with advance bandwidth provisioning capabilities.



(a)



(b)

Figure 3.8: Test Network Topologies, (a) 16 Nodes/25 Links, (b) 24 Nodes/43 Links

Meanwhile, all overlay requests are randomly generated with 4-6 nodes each and an average node degree of 2.5. The corresponding overlay link capacities are also chosen in a random manner, ranging uniformly between 100 and 1,000 Mbps. Furthermore, incoming requests have exponentially-distributed holding and inter-arrival times with means μ and λ, respectively. In particular, the mean holding time is set to 600 time units, and the mean

inter-arrival time is varied according to the desired input load. Carefully note that network load is commonly measured using the dimensionless Erlang metric, defined as the ratio of the service rate to the arrival rate [RW01]. However, to properly account for varying numbers of virtual links (i.e., connections) in an overlay request, a slightly-modified load metric is proposed here as follows [FG01]:

$$\text{Modified Erlang load} = \frac{1}{3}\sum_{n=4}^{6}(n-1) \times \mu/\lambda \qquad \text{(Eq. 3–11)}$$

for overlay topologies ranging from n=4-6 nodes.

Sensitivity tests are first done to select the fractional bandwidth parameter ρ, for the THR scheme. Namely, three different ρ values are tested for both network topologies, i.e., ρ=0.1, 0.5, and 0.9. These tests are done using 500,000 random VONS requests for both 16-node and 24-node topologies. The overall blocking results (not shown) indicate very minimal variations between the different ρ values, i.e., smaller ρ values give approximately 1% lower blocking versus larger ρ settings at any given input load. Meanwhile the average overlay connection path lengths are also shown in Figure 3.9 and show slightly higher utilization with smaller ρ values (particularly at higher loads). Given the nearly identical blocking rates here, shorter average path lengths (for larger ρ values) indicate more efficient resource usage. Nevertheless, the respective differences between the ρ values still fall within 1% of each other at any given input load.

Next, the number of re-routed reservations and re-routing success rates are also plotted in Figure 3.10 for varying ρ values. These results show notably higher overheads (total number of re-routing attempts) with smaller ρ values, i.e., due to reduced re-route triggering thresholds. Finally, re-routing success rates are also plotted in Figure 3.11 and indicate improved setup performance with larger ρ values. This is expected since these values will result in fewer, more successful re-routing attempts. Based upon these findings, a median value of ρ=0.5 is chosen to maintain a balance between re-routing overheads and blocking reduction. The various schemes are now tested, including the dynamic ILP approach.

3.4.1 Blocking Rate Performance

The overall request blocking rates are shown in Figure 3.12 for all other schemes. A total of 1,000 VONS requests are simulated for both topologies. Foremost these findings confirm that the dynamic ILP scheme gives the highest setup success rates, averaging about 20% higher than the re-routing schemes, especially at lower loads. Similar separation can also be seen in the larger topology (24-node). In general, these behaviors are expected since the ILP scheme can re-configure more routing resources in a larger topology, giving better overall performance. Moreover as a trade-off, larger topologies also yield more optimization variables, resulting in additional computation time (for the same number of states and overlapping requests as compared to the smaller topology). Meanwhile, the separation between the re-routing and non-re-routing heuristics is generally lower, but still notable, i.e., averaging around 10% lower blocking depending upon input load. However, these results also

indicate very little separation between the individual re-routing schemes, with the respective blocking ratios falling to within 2% of each other for both topologies.

The corresponding run times for the dynamic ILP scheme are also shown in Table Table 3.2. Although these values increase with load, they mostly fall within the tens of seconds range, indicating good applicability in real-world on-line settings.

Table 3.2: Average Run-Time Comparison

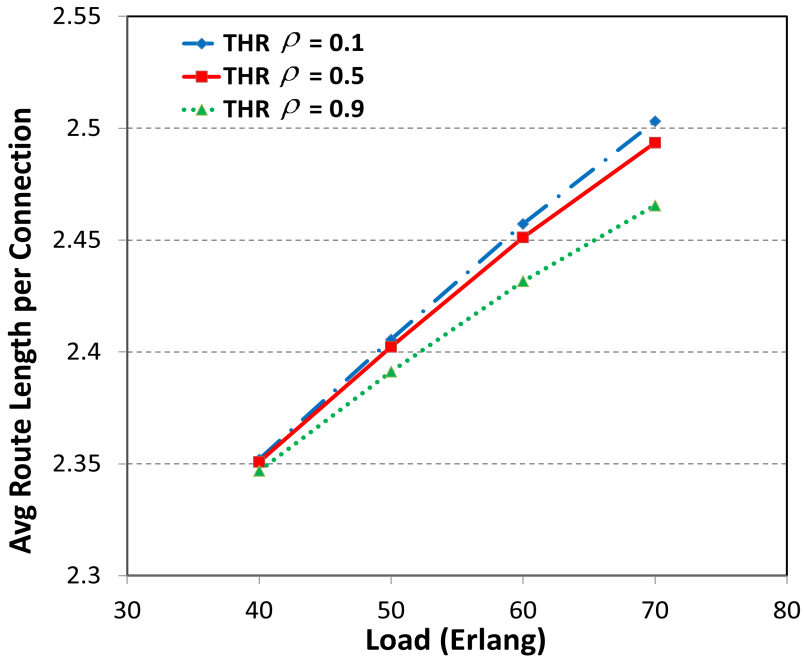
Schemes	Heuristic No Re-Routing	Heuristic Re-Routing	ILP Optimization
Single Request	<1 second	<1 second	3-10 seconds
All Requests	15-20 seconds	20-30 seconds	30-180 minutes

3.4.2 Average Overlay Connection Path Lengths

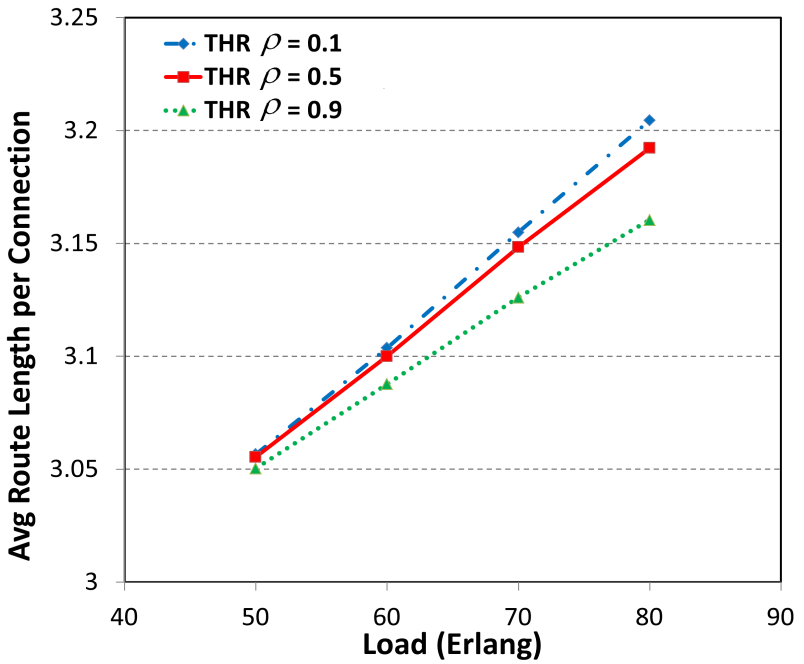
The average overlay connection path lengths are also plotted in Figure 3.13. As expected, the dynamic ILP scheme gives the lowest resource utilization, followed by the non-re-routing heuristic schemes. In particular, the optimization approach gives about 5-10% lower hop count utilization than the baseline min-distance scheme. By contrast, the re-routing schemes give slightly longer path lengths, indicating higher resource utilization. In general, this is expected since re-routing procedures result in longer detour routes to accommodate new demands, i.e., trade-off between blocking and resource efficiency.

3.4.3 Re-Routing Heuristics Comparison

The individual re-routing heuristics are also compared here. First, Figure 3.14 plots the number of successful and failed re-routed request attempts for each scheme in order to gauge overall run-time durations. These findings indicate that the THR (MHR) algorithm gives the lowest (highest) re-routing overheads. Hence in light of the relatively close blocking performances for all re-routing heuristics (Figure 3.12), it can be concluded that the THR scheme gives the most competitive re-routing performance. Meanwhile, re-routing success rates are also plotted in Figure 3.15 to gauge the efficiency of the re-routing schemes. These findings show that re-routing is much more effective at lower load regimes since there are fewer contending users and more available network resources (resulting in fewer re-routing attempts). Last but not least, the THR scheme gives the highest overall re-routing success rates. Hence this heuristic is deemed more efficient than the MHR and MNR re-routing strategies, i.e., even though all variants yields very close blocking and resource utilization results (see Figure 3.12 and Figure 3.13).

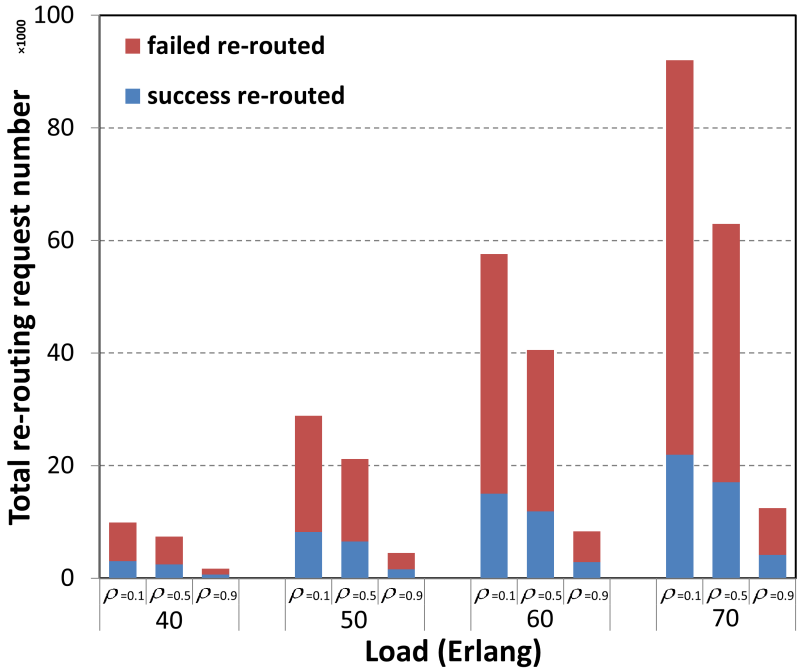


(a)

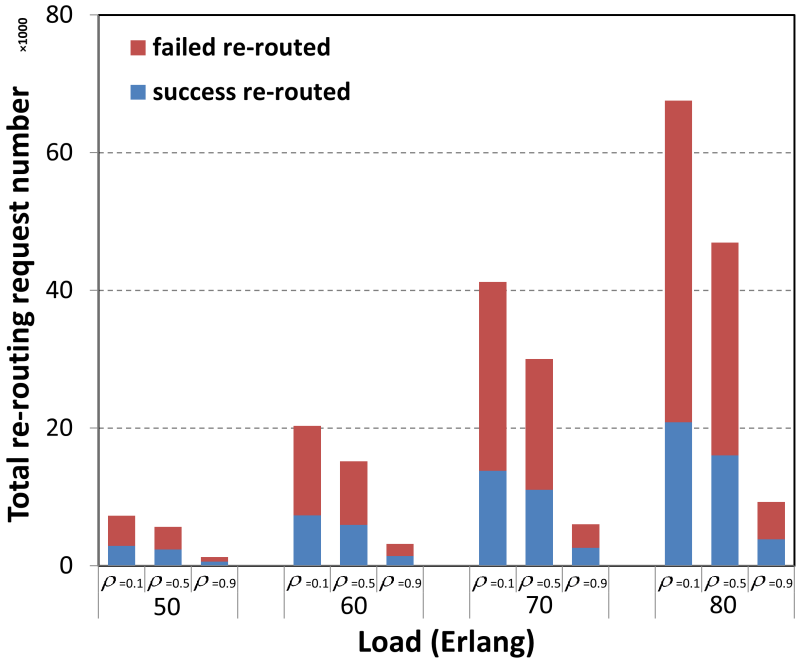


(b)

Figure 3.9: Average Overlay Link Connection Length: a) 16-Node Topology, b) 24-Node Topology

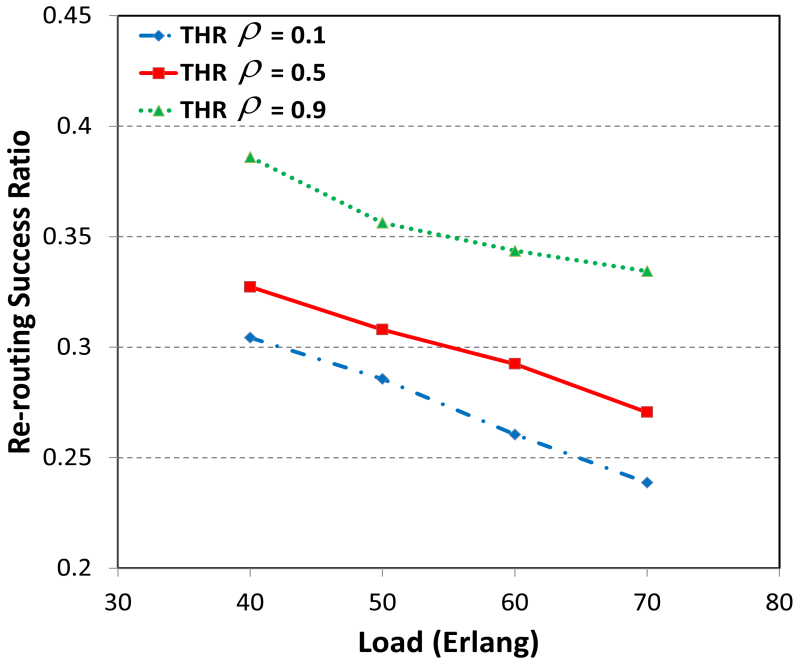


(a)

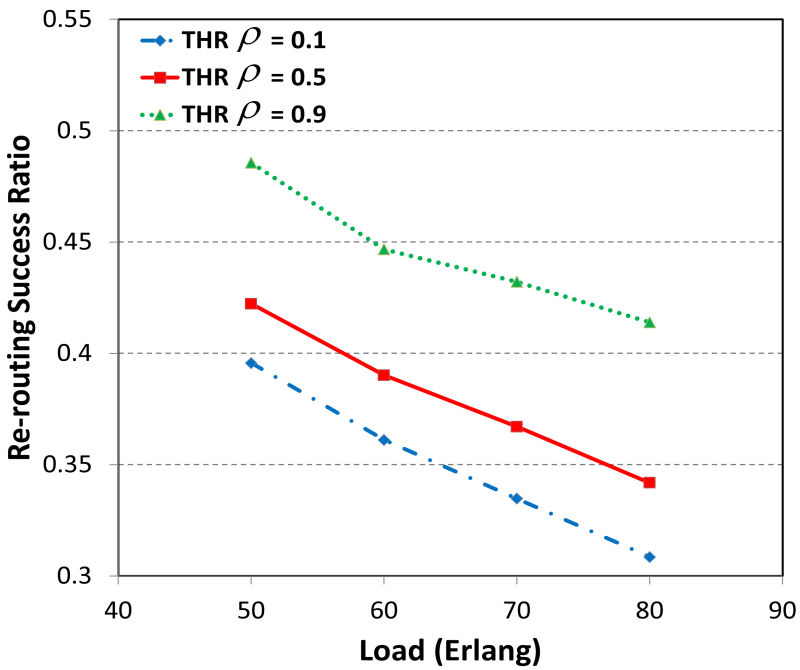


(b)

Figure 3.10: Number of Re-routing Requests (Success & Failed): a) 16-Node Topology, b) 24-Node Topology

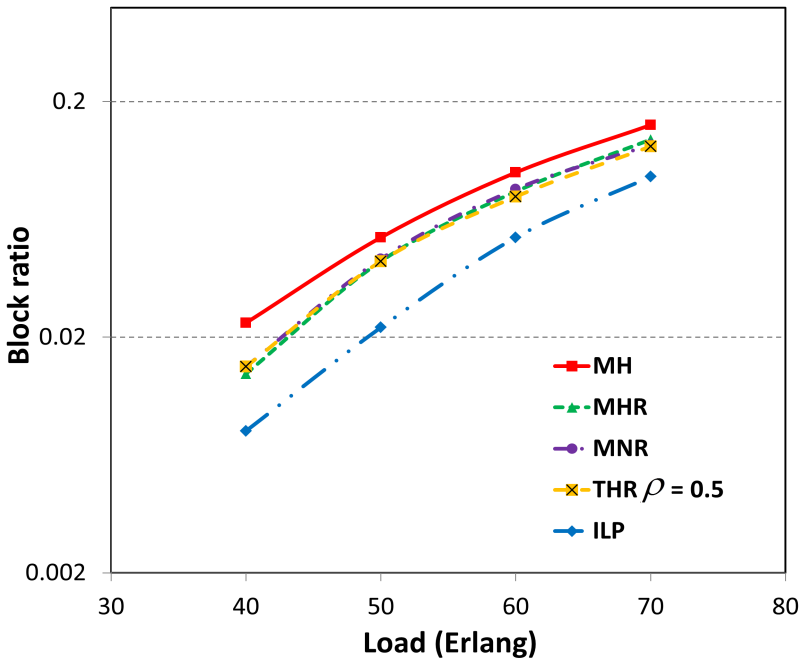


(a)

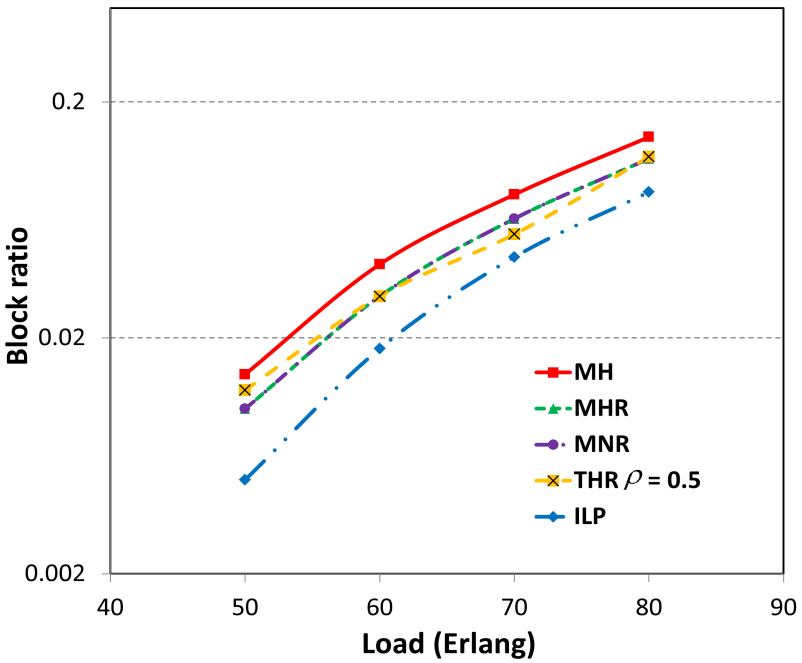


(b)

Figure 3.11: Re-routing Success Ratio: a) 16-Node Topology, b) 24-Node Topology

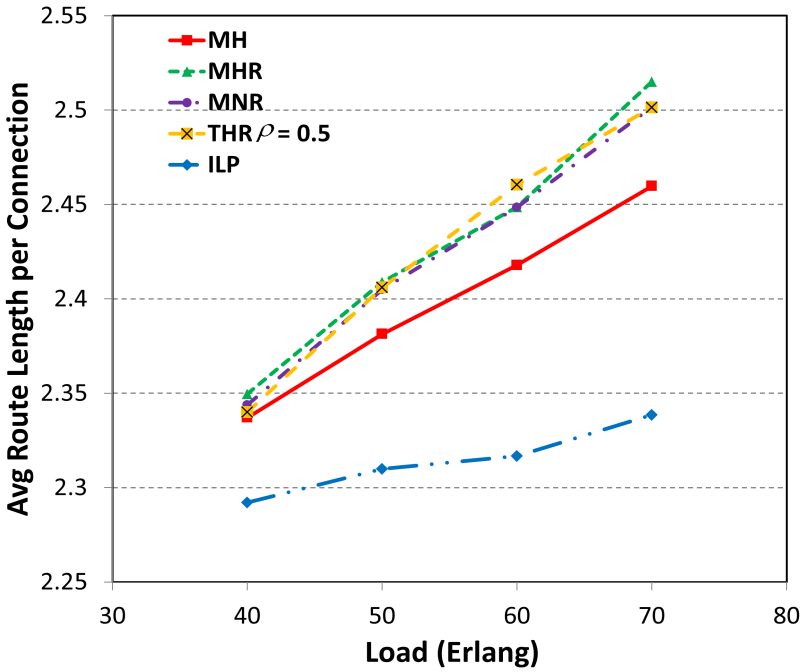


(a)

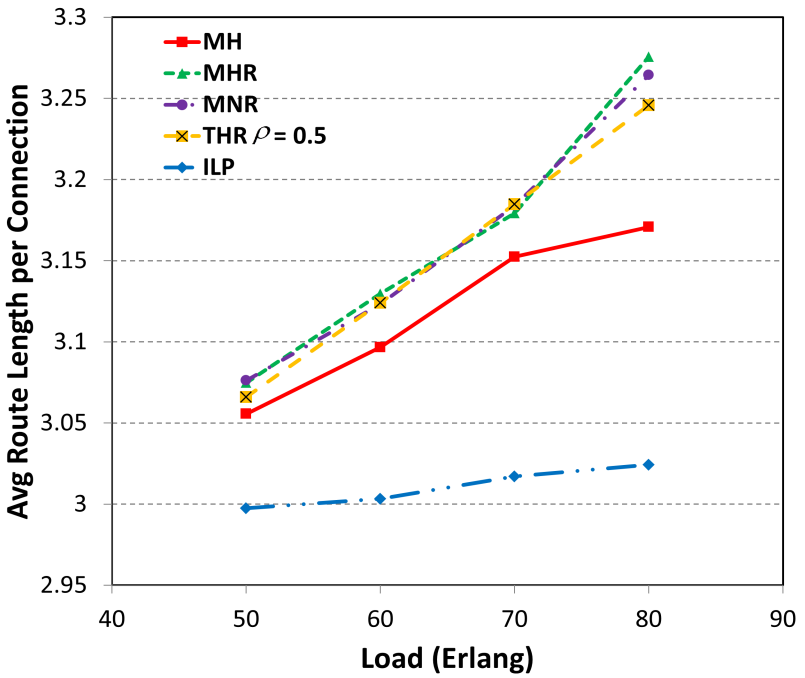


(b)

Figure 3.12: Request Blocking Rate: a) 16-Node Topology, b) 24-Node Topology

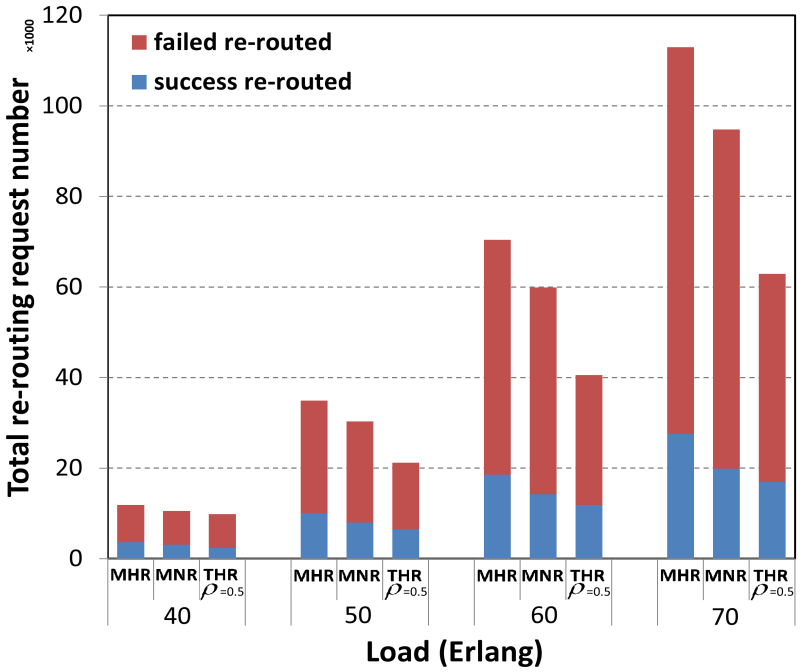


(a)

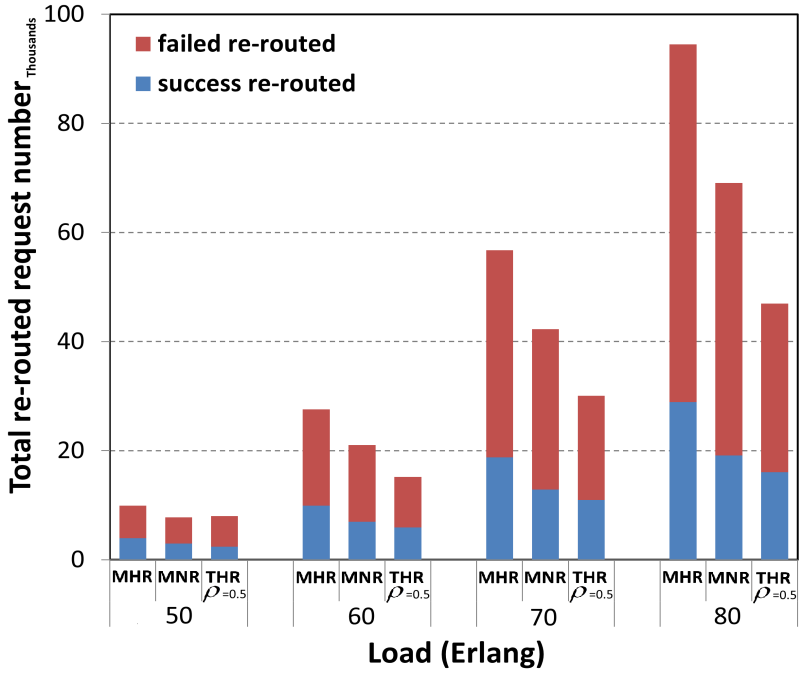


(b)

Figure 3.13: Average Overlay Link Connection Length: a) 16-Node Topology, b) 24-Node Topology

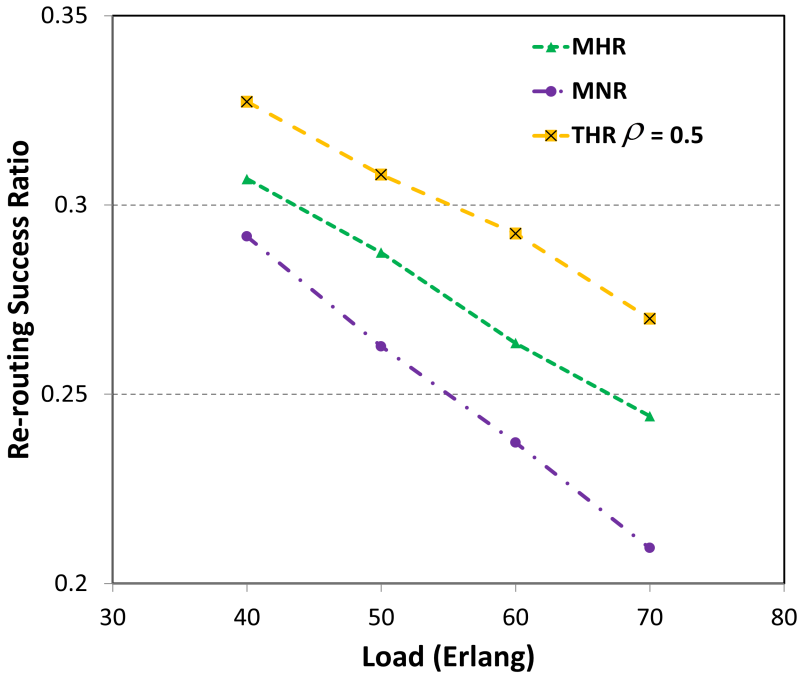


(a)

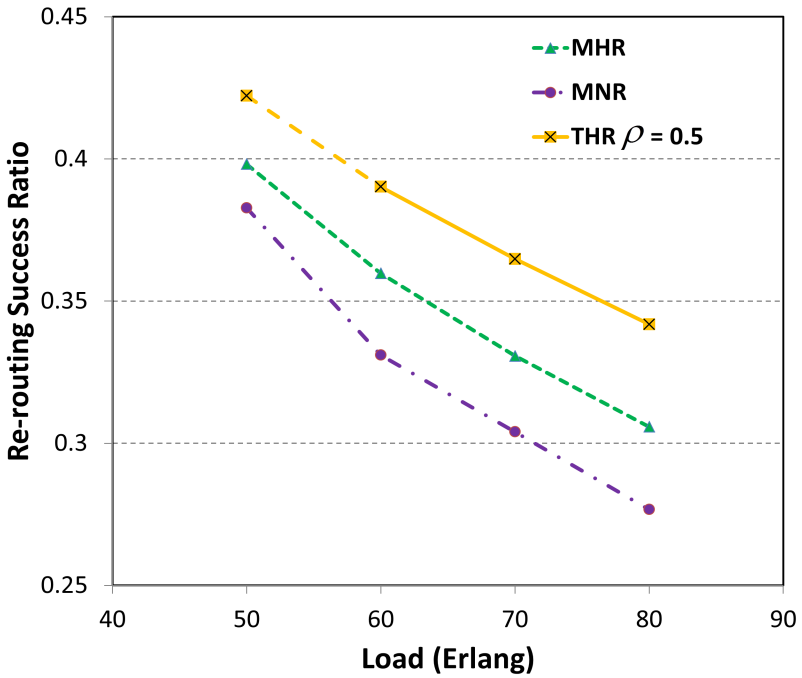


(b)

Figure 3.14: Re-routing Requests: a) 16-Node Topology, b) 24-Node Topology



(a)



(b)

Figure 3.15: Re-routing Success Ratio Among Different Heuristics: a) 16-Node Topology, b) 24-Node Topology

Chapter 4 Virtual Network Advance Reservation [1]

The VONS study in Chapter 3 basically addressed bandwidth demand scheduling between *fixed* end-point nodes. However, many organizations using cloud-based applications will want more flexible service options. For example, many users may require some form of data center (substrate) resources at the network node sites as well, e.g., computing or storage. Furthermore, most cloud customers may not necessarily care about exactly where their services are located or executed, i.e., as long as they meet necessary QoS, policy and security requirements. As a result, variable VN node placement can greatly improve service flexibility and help satisfy different user needs.

In light of above, this chapter presents a novel advance reservation framework for handling more generalized VN requests (with variability of node placement). The goal here is to provide a baseline model for future related work. Foremost, the VN scheduling problem is introduced and a global optimization formulation is presented for admitted but inactive time-overlapping VN requests without a-priori knowledge of upcoming requests. Owing to high computation complexity (due to the added timeline-dimension), a meta-heuristic solution is also developed using a *simulated annealing* (SA) approach to provide a near optimal solution. Moreover, inspired by the VNE problem, several single- and two-phase

[1] This chapter was previously published in [HB02]. Permission is included in Appendix B.

heuristic VN scheduling schemes are also developed to provide a baseline for future work. Complete details are now presented.

4.1 Network Model and Description

Before detailing the proposed scheduling strategies, the overall model for VN AR services is presented along with the requisite notation. Note that the overall notation and network model here is similar to that introduced in Section 3.1, with some additions and modifications to handle VN node placement/scheduling.

4.1.1 Substrate Network

The substrate network is modeled as an undirected graph $G_s = (V_s, E_s)$, where $V_s = \{v_s^1, v_s^2, ..., v_s^{|V_s|}\}$ is the set of substrate nodes and $E_s = \{e_s | e_s = (v_s^i, v_s^j) : v_s^i, v_s^j \in V_s\}$ is the set of substrate links connecting substrate nodes. Additionally each substrate node $v_s \in V_s$ has a fixed amount of computing and storage resources, C units, and each substrate link $e_s \in E_s$ has a fixed bandwidth capacity, B units. Moreover, in order to schedule VN node/link requests, time-varying capacity levels are also defined for nodes and links, i.e., $rem_v(t)$ and $rem_e(t)$, respectively. Overall, a sample 10-node substrate network hosting two 3-node VN requests is shown in Figure 4.1. Here the numbers next to each substrate node (link) represent the minimum node resource (link bandwidth) levels within the requested time-interval. Meanwhile, the numbers next to each VN node (link) represents the amount of requested node resource (link bandwidth).

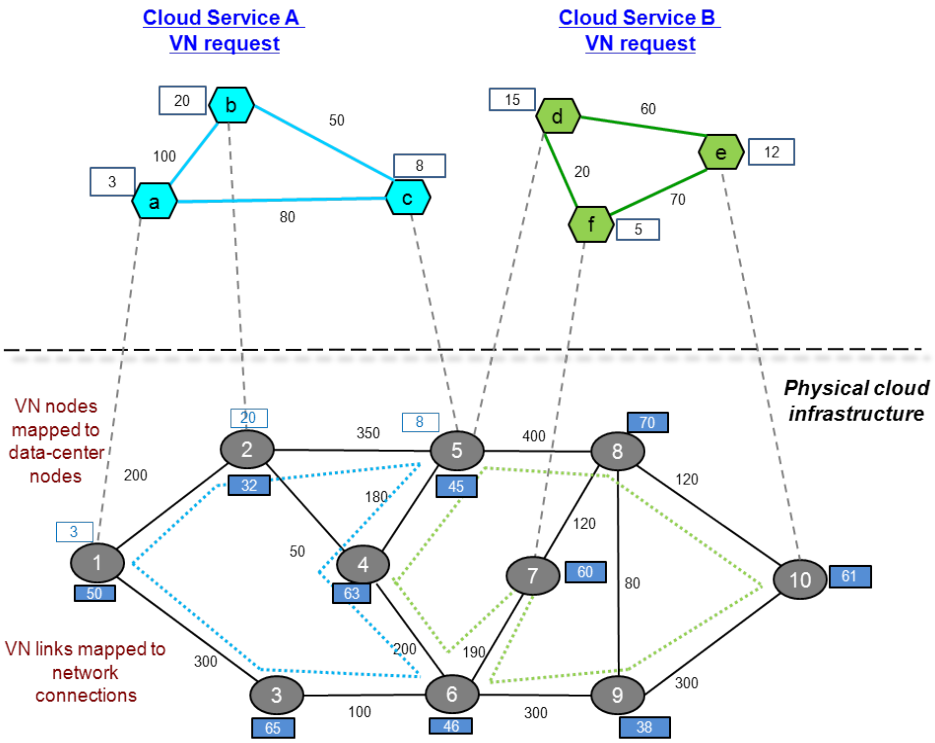


Figure 4.1: Physical Substrate Network with Embedded Virtual Networks

4.1.2 VN Scheduling Request

A VN scheduling request is defined by the 5-tuple, i.e., $r^n = (G_v^n, c^n, b^n, t_s^n, t_e^n)$, where n is the request index, $G_v^n = (V_v^n, E_v^n)$ is an undirected virtual graph containing the set of VN nodes $v_v^n \in V_v^n$ and virtual links $e_v^n \in E_v^n$, c^n is the requested amount of node resources ($c^n < C$), b^n is the requested amount of bandwidth ($b^n < B$), t_s^n is the start time, and t_e^n is the stop time. Similar to a substrate link, a VN link connects two VN node end-points. Furthermore, a VN node-to-substrate mapping is also denoted as $< v_v, v_s >$, i.e., VN node v_v mapped to physical substrate node v_s. The mapped substrate node for a VN node v_v is also denoted as $\eta(v_v)$. Carefully note that all t_s^n and t_e^n values occur at integral multiples of the

discrete time-slot value/duration. Furthermore, all requests arrive in an "on-line" fashion and are processed in a *first come first serve* (FCFS) manner. Again, Figure 4.1 shows two sample VN requests with their associated node resource and link bandwidth requirements. For example VN request A is assigned node mappings $\{< a, 1 >, < b, 2 >, < c, 5 >\}$ and request B is assigned node mappings $\{< d, 5 >, < e, 10 >, < f, 7 >\}$. Additionally, virtual link "ac" in VN request A is mapped to a connection along substrate links $\{< 1-3 >, < 3-6 >, < 6-4 >, < 4-5 >\}$.

4.1.3 Problem Description and Objectives

The VN scheduling problem is defined by a mapping $M : G_v(V_v, E_v) \rightarrow G_s(V'_s, E'_s)$ from G_v to a subset of G_s, where $V'_s \subset V_s$ and $E'_s \subset E_s$. The main objective here is to efficiently schedule VN demands in their requested time intervals. Furthermore, sliding time-windows are not assumed here.

4.1.4 Performance Evaluation Metrics

The key metrics used for VN scheduling performance are now presented. Foremost, resource efficiency and revenue generation (cost reduction) are two major operator concerns [GS01] [XC01]. Hence some related metrics are defined here. First, the modified net revenue associated with provisioning a VN request is defined as:

$$REV(G_v^n) = (t_e^n - t_s^n) * (\sum_{e_v^n \in E_v^n} b^n/\mathbb{B} + \rho \sum_{v_v^n \in V_v^n} c^n/\mathbb{C}) \qquad \text{(Eq. 4-1)}$$

where ρ is the fraction of node resource revenue, and $\mathbb{B}$ and $\mathbb{C}$ are large numbers to normalize node/link resources. Unlike [MC01], here the modified revenue is dependent upon request durations, i.e., a request with longer duration but the same node/link resources has larger net revenue. Next, the cost of accepting a VN request is defined as:

$$COST(G_v^n) = (t_e^n - t_s^n) * (\sum_{e_s \in E_s} \mathcal{F}_{e_s}^{G_v^n}/\mathbb{B} + \pi \sum_{v_s \in V_s} \mathcal{N}_{v_s}^{G_v^n}/\mathbb{C}) \quad \text{(Eq. 4-2)}$$

where π is the fraction of node resource cost, $\mathcal{F}_{e_s}^{G_v^n}$ is the total amount of bandwidth allocated on substrate link e_s for mapping the VN, and $\mathcal{N}_{v_s}^{G_v^n}$ is the total amount of node resources allocated at substrate node v_s for mapping the VN. A similar duration-sensitive modification is also used here, as compared to [MC01].

Now from an operator's point of view, an efficient and effective on-line VN scheduling algorithm should maximize long-term revenue in the substrate network. Hence, akin to [MY01], the long-term average revenue is defined as:

$$\lim_{T \to \infty} \frac{\sum_n REV(G_v^n)}{T}, \quad \forall G_v^n \in \mathbb{A} \quad \text{(Eq. 4-3)}$$

where $\mathbb{A}$ is the set of all accepted VN requests, and T is the total time duration. Similarly, the long-term average cost is defined as:

$$\lim_{T \to \infty} \frac{\sum_n COST(G_v^n)}{T}, \quad \forall G_v^n \in \mathbb{A} \quad \text{(Eq. 4-4)}$$

Finally the long-term revenue-cost ratio is also defined to quantify the efficiency of resource utilization in the substrate network as follows:

$$\frac{\sum_n REV(G_v^n)}{\sum_n COST(G_v^n)}, \quad \forall G_v^n \in \mathbb{A} \qquad \text{(Eq. 4-5)}$$

Carefully note that several other performance metrics are also considered here. For example, the VN request blocking ratio is a well-established measure of lost revenue, and hence minimizing this figure is a major objective as well. Additionally, the average route length (of embedded VN links) is also considered in order to gauge network substrate link utilization.

4.1.5 Load Balancing

In general, most embedding or routing schemes use static node and link weights. However, these values cannot account for real-time dynamic resource variations at the substrate level and may yield increased congestion at specific nodes or links. Therefore, a modified time-sensitive *load balancing* strategy is proposed to alleviate such concerns here. Namely, substrate link cost (weights) are set as inversely proportional to the load over the time interval $[t_s^n, t_e^n]$, as follows:

$$C(e) = \frac{B \times (t_e^n - t_s^n)}{\int_{t_s^n}^{t_e^n} rem_e(t) + \epsilon} \qquad \text{(Eq. 4-6)}$$

where B is the full (maximum) capacity of a substrate link and ϵ is a small value (to avoid division errors). In particular, the denominator term in Eq. 4-6 represents the average amount of free resources in the interval. Similarly, node resource costs can be defined as:

$$C(v_s) = \frac{C \times (t_e^n - t_s^n)}{\int_{t_s^n}^{t_e^n} rem_v(t) + \epsilon} \quad \text{(Eq. 4-7)}$$

where C is the full (maximum) capacity of a substrate node. Overall, the substrate link (or node) cost is defined as the ratio of total capacity to available bandwidth (capacity) during requested interval, i.e., links (nodes) with more available resources have smaller cost and are more likely to be chosen.

Meanwhile Dijkstra's shortest path using minimum load balance cost (Eq. 4-6) is denoted as $P(v_s^1, v_s^2)$. Hence the cost between two substrate nodes v_s^1 and v_s^2 is defined as $cp_{v_s^1,v_s^2}$:

$$cp_{v_s^1,v_s^2} = \sum_{e \in P(v_s^1,v_s^2)} C(e) \quad \text{(Eq. 4-8)}$$

Similarly, the cost between two virtual nodes v_v^1 and v_v^2 is also defined as $cp_{v_v^1,v_v^2}$, respectively:

$$cp_{v_v^1,v_v^2} = cp_{\eta(v_v^1),\eta(v_v^2)} = \sum_{e \in P(\eta(v_v^1),\eta(v_v^2))} C(e) \quad \text{(Eq. 4-9)}$$

4.2 Optimization Formulation

A detailed ILP optimization for VN advance reservation is now presented using the above notation. This approach pursues a single-objective function to minimize overall resource usage per VN request. Akin to the VONS optimization (Section 3.2), this approach also performs dynamic optimization over a reduced batch of admitted, inactive time-overlapping VN requests (without a-priori knowledge of upcoming requests). Time is also discretized into fixed time-slots of duration T. Building upon the VONS model, here each virtual node $v_v^n \in V_v^n$ is mapped to a unique substrate node $v_s \in V_s$ with sufficient

node resources, and two virtual nodes in the same request cannot be mapped to the same substrate node. This enforces a one-to-one mapping between virtual and physical nodes within a VN request. At the same time, VN links (connections) must be routed between the mapped substrate nodes. Hence each VN link e_v^n is treated as a directional single flow running between two virtual nodes with no path splitting. VN link bandwidth reservation is also done in both link directions during $[t_s^n, t_e^n]$. As per the formulation, the following variables are defined:

- $\mathbb{R}$: Set of requests treated in the optimization formulation including the incoming request and admitted inactive time-overlapping reservations, i.e., $\forall r^n = (G_v^n, c^n, b^n, t_s^n, t_e^n) \in \mathbb{R}$ will be considered in the optimization
- $f_{i,j}^{q,n,k}$: Binary link mapping variable, i.e., $f_{i,j}^{q,n,k} = 1(0)$ if virtual link $q^n \in E_v^n$ does (not) use link $(i,j) \in E_s$ (direction $i \rightarrow j$) in time-slot k
- $s_u^{p,n,k}$: Binary node mapping variable, i.e., $s_u^{p,n,k} = 1(0)$ if virtual node $p^n \in V_v^n$ is (not) mapped to substrate node $u \in V_s$ in time-slot k
- $v_{src}^{q,n}$: Source node (virtual) of virtual link q^n which is mapped to egress node of mapped substrate link
- $v_{dst}^{q,n}$: Destination node (virtual) of virtual link q^n which is mapped to ingress node of mapped substrate link.

Using the above definitions, the objective function is defined as:

$$min \sum_{r^n \in \mathbb{R}} \sum_{t_s^n \leq k \leq t_e^n} (\sum_{i \in V_s} \sum_{j \in V_s} b^n * f_{i,j}^{q,n,k} / \mathbb{B} + \sum_{u \in V_s} c^n * s_u^{p,n,k} / \mathbb{C}) \quad \text{(Eq. 4-10)}$$

Accordingly, the related constraints are:

$$f_{i,j}^{q,n,k} = 0, \forall r^n \in \mathbb{R}, \forall q^n \in E_v^n, \forall i,j \in V_s, (i,j) \notin V_s, \forall k \in [t_s^n, t_e^n] \quad \text{(Eq. 4-11)}$$

$$\sum_{r^n \in \mathbb{R}} \sum_{q^n \in E_v^n} b^n * (f_{i,j}^{q,n,k} + f_{j,i}^{q,n,k}) \leq B, \forall i,j \in V_s, \forall k \in [t_s^n, t_e^n] \quad \text{(Eq. 4-12)}$$

$$\sum_{r^n \in \mathbb{R}} \sum_{p^n \in V_v^n} c^n * s_u^{p,n,k} \leq C, \forall u \in V_s, \forall k \in [t_s^n, t_e^n] \quad \text{(Eq. 4-13)}$$

$$\sum_{u \in V_s} s_u^{p,n,k} = 1, \forall r^n \in \mathbb{R}, \forall p^n \in V_v^n, \forall k \in [t_s^n, t_e^n] \quad \text{(Eq. 4-14)}$$

$$\sum_{p^n \in V_v^n} s_u^{p,n,k} \leq 1, \forall u \in V_s, \forall r^n \in \mathbb{R}, \forall k \in [t_s^n, t_e^n] \quad \text{(Eq. 4-15)}$$

$$\sum_{j \in V_s} f_{i,j}^{q,n,k} - \sum_{j \in V_s} f_{j,i}^{q,n,k} - s_i^{v_{src}^{q,n},n,k} + s_i^{v_{dst}^{q,n},n,k} = 0, \forall r^n \in \mathbb{R}, \forall q^n \in E_v^n, \forall i \in V_s, \forall k \in [t_s^n, t_e^n] \quad \text{(Eq. 4-16)}$$

$$f_{i,j}^{q,n,k} = f_{i,j}^{q,n,k+1}, \forall r^n \in \mathbb{R}, \forall q^n \in E_v^n, \forall i,j \in V_s, \forall k \in [t_s^n, t_e^n) \quad \text{(Eq. 4-17)}$$

$$s_u^{p,n,k} = s_u^{p,n,k+1}, \forall r^n \in \mathbb{R}, \forall p^n \in V_v^n, \forall u \in V_s, \forall k \in [t_s^n, t_e^n) \quad \text{(Eq. 4-18)}$$

$$f_{i,j}^{q,n,k} \in \{0,1\}, \forall r^n \in \mathbb{R}, \forall q^n \in E_v^n, \forall i,j \in V_s, \forall k \in [t_s^n, t_e^n] \quad \text{(Eq. 4-19)}$$

$$s_u^{p,n,k} \in \{0,1\}, \forall r^n \in \mathbb{R}, \forall p^n \in V_v^n, \forall u \in V_s, \forall k \in [t_s^n, t_e^n] \quad \text{(Eq. 4-20)}$$

Overall, the objective function in Eq. 4-10 tries to minimize the cost of scheduling the overlapping subset of VN requests, i.e., sum of all VN node and VN link mapping costs normalized by two large numbers across all substrate nodes. Meanwhile, Eq. 4-11 appropriately constrains the link mapping variables. Next, Eq. 4-12 and Eq. 4-13 bound link capacity and node resources, respectively, at any given time-slot. Namely, summing $f_{i,j}^{q,n,k}$ and $f_{j,i}^{q,n,k}$ ensures that the total flows in both directions do not exceed the total available bandwidth on link (i,j). Based on this, Eq. 4-13 sums all node mappings to make sure they do not exceed node capacity. Moreover, Eq. 4-14 ensures that a virtual node is only mapped to one substrate node, whereas Eq. 4-15 ensures that one substrate node can be allocated to at most one virtual node in the same VN request. More importantly, Eq. 4-16 ensures flow conservation at a source, destination, or transit nodes by combining the node and link mapping variables. Specifically, consider any virtual link q^n in the optimization set. Here if node v_i is chosen as the substrate node to map the source node of q^n, i.e., $s_i^{v_{src}^{q,n},n,k} = 1$ and $s_i^{v_{dst}^{q,n},n,k} = 0$, then the total flow for q^n leaving node v_i is unity. Meanwhile, if v_i is chosen as

1: Given new VN AR request
2: Identify set of accepted *inactive* reservations $\mathbb{R}$
3: Remove substrate reserved resources in $G_s = (V_s, E_s)$ to generate temporary graph $G'_s = (V'_s, E'_s)$
4: Generate ILP formulation for set $\mathbb{R}$
5: **if** ILP solution found
6: Setup success, reserve resources in $G'_s = (V'_s, E'_s)$, copy G'_s to G_s
7: **else**
8: Drop request and discard $G'_s = (V'_s, E'_s)$

Figure 4.2: Modified ILP Formulation for VN Scheduling

the substrate node to map the destination node of q^n, i.e., $s_i^{v_{src}^{q,n},n,k} = 0$ and $s_i^{v_{dst}^{q,n},n,k} = 1$, then the total flow for q^n entering node v_i is also unity. On the other hand, if v_i is a transit node, i.e., $s_i^{v_{src}^{q,n},n,k} = 0$ and $s_i^{v_{dst}^{q,n},n,k} = 0$, the total flow entering and leaving v_i is zero. Meanwhile , Eq. 4-17 and Eq. 4-18 ensure VN link connection and VN node mapping consistency during the duration of the request interval, i.e., no time-varying mappings. Finally, Eq. 4-19 and Eq. 4-20 specify the necessary binary constraints on variables $f_{i,j}^{q,n,k}$ and $s_u^{p,n,k}$.

The overall pseudo-code for the ILP formulation is shown in Figure 4.2. This scheme identifies the set of overlapping reservations and formulates the reduced dynamic optimization problem for the incoming request. Now the actual ILP is run over a temporary substrate graph by removing all reserved resources for accepted but inactive requests in the set. If this ILP is successful in finding a valid mapping for all requests in the set, then the new request is accepted and its respective resources are reserved in $G_s = (V_s, E_s)$. Otherwise the request is dropped.

Overall, the above ILP poses significant complexity. For example, consider a 5-node/10-link VN request with 10 time-slots over a 20-node mesh substrate. The total number of variables here will be approximately 41,000 (i.e., 40,000 $f_{i,j}^{q,n,k}$ variables, 1,000 $s_u^{p,n,k}$ variables). Now the state-based method in Section 3.2.1 can also be applied to reduce this variable count. For example, on average this reduces the 10 time-slots to only 2 states. Hence the revised variable count will now drop to 8,200. However, adding any overlapping requests will quickly increase this total by multiple factors. Hence more tractable heuristic schemes are also pursued herein.

4.3 Graph-Based Heuristic Solution

Owing to the high computational complexity of the optimization formulation, more tractable graph-based heuristic solutions are presented for scheduling VN requests. These algorithms include several two-stage schemes (which first map VN nodes onto substrate nodes and then route the VN links) as well as a single-stage scheme that jointly maps VN nodes and links.

4.3.1 Two-Stage Heuristic

The overall pseudo-code for the two-stage heuristic is shown in Figure 4.3. Here the VN nodes are first sorted according to their node degree and mapped sequentially, i.e., nodes with more adjacent VN nodes are mapped first (steps 3-12, Figure 4.3). Next, the virtual links between the mapped nodes are routed (steps 13-15, Figure 4.3) using Dijkstra's shortest path algorithm. Specifically, the link weights here are selected using the earlier-detailed LB

link weighting strategy defined in Section 4.1.5. Finally, the heuristic only returns success if all nodes and links are successfully mapped (scheduled). Now consider the virtual node mapping stage first, where four different strategies are proposed:

- Random Pick (RP): This scheme picks a random substrate node with enough free resources in the VN request interval. This approach essentially provides a baseline for comparison purposes.
- Maximum Neighbors (MN): This scheme chooses the substrate node with the maximum number of available neighbors in $G'_s(V'_s, E'_s)$. This approach tries to assign the most-connected substrate nodes first in order to (subsequently) increase the likelihood of successfully routing virtual links.
- Maximum Bandwidth (MB): This scheme chooses the substrate node with the most available bandwidth on its adjacent links. This approach tries to achieve link load-balancing when mapping virtual nodes by avoiding heavily-loaded substrate links.
- Maximum Product (MP): This scheme pursues a median between balancing node and link loads. Namely, the product of the remaining node resources and total adjacent link bandwidth levels is calculated for each substrate node in $G'_s(V'_s, E'_s)$. Each virtual node is then mapped to the available substrate node with the maximum product.

Carefully note that a temporary copy of the network graph, $G'_s(V'_s, E'_s)$ is used to perform all node/link mapping computations, i.e., Figure 4.3, step 2. This graph basically removes any substrate nodes or links which cannot accommodate a requested VN node or VN link, i.e., all substrate links with $b_e^{min} < b^n$ and all substrate nodes with $b_v^{min} < c^n$. Again, the

bottleneck link is defined as per Eq. 3-10 (Figure 3.7). Similarly, the bottleneck capacity for a substrate node is also defined as:

$$b_v^{min} = \min rem_v(t), t_s^n \leq t \leq t_e^n \qquad \text{(Eq. 4-21)}$$

Based upon above, all non-feasible links with $b_e^{min} < b^n$ and nodes with $b_v^{min} < c^n$ are removed to generate the reduced sub-graph.

In general, the above heuristics can only provide a baseline for developing further improved algorithms and do not provide any guarantee of optimal or even near-optimal result. Now consider the overall computational complexity here. Foremost, removing non-feasible nodes and links to build the temporary graph (steps 1-2) requires checking all substrate nodes and links, yielding a complexity of $O(|E_s| + |V_s|)$. Unlike the (IR) VNE mapping problem, here the time-varying capacity levels during overlapping VN request intervals also need to be considered. In fact, the average number of overlapping VN requests depends on the request inter-arrival and holding time. Hence assuming that there are M overlapping requests when a new arrival request arrives, at most $N = 2M + 1$ time blocks have to be considered when computing the bottleneck capacity while generating temporary graph. In addition, sorting the VN nodes requires an average complexity of $O(|V_v|log|V_v|)$. Next, each VN node mapping approach (steps 4-12) requires $O|V_s|$, resulting a total complexity of $O|V_v||V_s|$. Moreover, assigning link costs in $G'_s(V'_s, E'_s)$ (step 14) has $O(N \times |E_s|)$ complexity. Finally, routing a virtual link connection only requires one instance of the Dijkstra's algorithm since the VN nodes have already been mapped, i.e., $O(|E_s| + |V_s|log|V_s|)$. Therefore, the runtime

1: Given an incoming request $r^n = (G_v^n, c^n, b^n, t_s^n, t_e^n)$, generate temporary graph copy $G'_s(V'_s, E'_s) = G(V, E)$
2: Remove non-feasible nodes and links in $G'_s(V'_s, E'_s)$, i.e., $b_v^{min} < c^n$, $b_e^{min} < b^n$ in $[t_s^n, t_e^n]$;
/ Loop and map all virtual nodes in the request*/*
3: Sort the virtual nodes based upon node degree
4: **for** every node in V_v^n
5: Pick substrate node from V'_s based on given node mapping (RP, MN, MB, or MP)
6: **if** failed
7: VN request r^n failed
8: Discard $G'_s(V'_s, E'_s)$ and temporary node mapping array
9: Exit loop
10: **else**
11: Remove substrate node from V'_s
12: Save node mapping in temporary node mapping array
/ Loop and map all virtual nodes in the request*/*
13: **if** All virtual nodes mapping successful
14: Assign load balancing weight to links in $G'_s(V'_s, E'_s)$
15: Run Dijkstra's shortest-path for each virtual link between mapped substrate nodes
16: **else**
17: VN request r^n failed
18: Drop r^n, discard $G'_s(V'_s, E'_s)$ and node mapping array
19: **if** all VN link connection routed
20: Setup successful
21: Reserve mapped node resources from node mapping array onto $G_s(V_s, E_s)$
22: Reserve routed link resources onto $G_s(V_s, E_s)$

Figure 4.3: Two-Stage Virtual Network Advance Reservation Heuristic Algorithm

complexity of two-stage VN scheduling algorithm is bounded by $O(2N|E_s|+N|V_s|+|V_v|(|V_s|+log|V_v|)+|E_v|(|E_s|+|V_s|log|V_s|))$.

4.3.2 Single-Stage Heuristic

As noted above, pre-selecting the node mappings without taking into account link concerns can yield poor performance, i.e., incorporating link traffic load information into the

node selection process may not suffice (MB, MP schemes). Therefore a more sophisticated algorithm is required to jointly coordinate between node and link scheduling and provide improved blocking and revenue performance. Along this lines, a single-stage VN scheduling scheme is proposed here, as shown in Figure 4.4.

1: Given incoming request $r^n = (G_v^n, c^n, b^n, t_s^n, t_e^n)$, generate temporary graph copy $G'_s(V'_s, E'_s) = G(V, E)$
2: Remove non-feasible nodes and links in $G'_s(V'_s, E'_s)$, i.e., $b_v^{min} < c^n$ and $b_e^{min} < b^n$ in $[t_s^n, t_e^n]$;
/ Loop and map all virtual nodes in the request*/*
3: Sort the virtual nodes based upon node rank
4: **for** every node in V_v^n
5: Compute a set of candidate substrate nodes
6: Compute candidate node cost and pick one with minimum total node cost (Eq. 4-22)
7: Route VN links from this selected substrate node to its mapped neighbor VN nodes
8: **if** all VN nodes and links mapped and routed
9: Setup successful
10: Reserve node and link resources onto $G_s(V_s, E_s)$
11: **else**
12: VN request r^n failed
13: Drop r^n, discard $G'_s(V'_s, E'_s)$ and node mapping array

---

Figure 4.4: Single-Stage Virtual Network Advance Reservation Heuristic Algorithm

Overall the single-stage VN scheduling heuristic leverages from existing single-stage VNE algorithms [HY01]. Specifically, VN node and VN link mappings are done jointly, and VN node mappings also consider potential (future) link mapping costs. The algorithm starts by generating a temporary copy of the network graph and removing any non-feasible nodes and links (step 2 in Figure 4.4). The requested VN nodes are then sorted and mapped in

decreasing order of their (virtual) node degrees (step 3 in Figure 4.4). For each unmapped VN node, v_v, a set of candidate substrate nodes is then computed. Here the total amount of available node computational resources cannot be lower than the requested amount c^n. In addition, the maximum and total bandwidth requested between a candidate VN node and its adjacent VN node(s) must also be considered. Namely, for any candidate substrate node, v_s, the maximum amount of available bandwidth on the links adjacent to v_s must not be less than b^n. Additionally, the total available bandwidth on all links adjacent to v_s also has to be satisfied. Based upon this selection, unavailable substrate nodes will be removed from future consideration.

Now consider minimum cost mapping. Here the cost of a candidate substrate node, v_s, is calculated as the sum of three factors, i.e., node mapping cost $C(v_s)$, average link cost to already mapped neighbor VN nodes, $C^0 < v_v, v_s >$, and average potential link cost to unmapped neighbor VN nodes, $C^1 < v_v, v_s >$, i.e.,

$$COST < v_v, v_s >= C(v_s) + C^0 < v_v, v_s > + C^1 < v_v, v_s > \qquad \text{(Eq. 4-22)}$$

Namely node mapping cost is calculated using load balancing (as in Section 4.1.5) to avoid congestion. Now assume V_0' is the set of neighboring nodes (for candidate node v_v) that have already been mapped, and the corresponding mapping set is $\eta(V_0') \in V_s$. Additionally, let V_1' be the set of neighboring nodes that have not been mapped yet. As a result, $C^0 < v_v, v_s >$ is computed as the sum of path costs between substrate candidate v_s and a set of substrate nodes that have been allocated to V_0', see Eq. 4-23. Meanwhile, $C^1 < v_v, v_s >$ is computed

as the sum of path costs between v_s and a set of substrate nodes that may be allocated to V_1', see Eq. 4-24.

$$C^0 < v_v, v_s >= \sum_{\forall v_v' \in V_0'} cp_{v_v', v_v} = \sum_{\eta(v_v') \in V_s} cp_{v_s, \eta(v_v')} \quad \text{(Eq. 4-23)}$$

$$C^1 < v_v, v_s >= \sum_{\forall v_v' \in V_1'} cp_{v_v', v_v} = \sum_{\eta(v_v') \in V_s \setminus \eta(V_0')} cp_{v_s, \eta(v_v')} \quad \text{(Eq. 4-24)}$$

Based on the above, the candidate node with the least cost (Eq. 4-22) is selected. Subsequently, the VN links between the chosen substrate node and the nodes corresponding to its mapped VN neighbors will be routed and scheduled using the minimum cost path (step 7). The VN request is successful if and only if all VN nodes and their adjacent VN links are mapped. Otherwise, this request is dropped and the temporary copy of the graph discarded. Overall, the single-stage mapping algorithm tends to choose feasible substrate nodes that take into account both node and link costs to mapped and unmapped VN nodes.

Finally, consider the time complexity for single-stage VN scheduling scheme. The overall procedure in Figure 4.4 starts by generating a temporary graph copy, i.e., akin to the two-stage scheme, $O(N \times (|E_s| + V_s))$. Similarly, VN node sorting requires $O(|V_v| log |V_v|)$ time complexity. It is also necessary to compute load-balancing link costs for each substrate link, i.e., $O(N|E_s|)$. Meanwhile, the single-stage VN mapping has similar time-complexity to the NSVIM algorithm in [HY01], i.e., $O(|E_v|(2+|V_s|)|V_s||E_s| log |V_s|)$. Therefore the total run-time complexity of the single-stage VN scheduling is bounded by $O(N(2|E_s| + |V_s|) + |V_v| log |V_v| + |E_v|(2 + |V_s|)|V_s||E_s| log |V_s|)$.

4.4 Simulated Annealing Meta-Heuristic Solution

In general, graph-based heuristics cannot provide any sort of optimal performance guarantee or bounds. Furthermore, the optimization models are not very tractable either (as per Section 4.2). Hence in order to achieve a balance between performance and computation complexity, additional meta-heuristic strategies are also developed here to achieve a near-optimal solution in a reasonable and controllable amount of time. Now as discussed in Section 2.1.3, various studies have used genetic algorithms, ant colony optimization, particle swarm optimization, and tabu search meta-heuristics for VNE mapping. Along these lines, a further *simulated annealing* (SA) solution is introduced to solve the VN AR scheduling problem by considering link costs to adjust node placement, see high-level overview in Figure 4.5.

Overall, SA methods have been widely used to solve global optimization problems with large search spaces. This approach defines a system temperature and interprets slow cooling as a slow decrease in the probability of accepting worse solutions as it searches the solution base. Hence by establishing the cooling phase and cooling rate, it is relatively straightforward to control the total number of searches, as well as the overall computation time. In general, SA schemes are easy to program when the search space is discrete. Consider some further detailed here.

First let vector $M = \left(\eta(v_1), \eta(v_2), ..., \eta(v_{|V_v|})\right)$ define a VN nodes mapping solution, i.e., $v_i \in V_v^n, i = 1, ..., |V_v^n|$. The feasibility of this solution is also verified by using Dijkstra's

shortest path to route all VN links whenever a new M is updated. The SA scheme also defines a cost $COST(M)$ for each feasible mapping based upon Eq. 4-22, i.e., the smaller the cost, the better the mapping. Moreover, two mapping solutions are also recorded throughout the SA process, i.e., the current solution ($sCurr$), and the best solution ($sBest$). The overall SA algorithm in Figure 4.5 entails three key steps:

1) Computing an initial feasible solution

2) Setting up the annealing phase to find a feasible neighboring solution

3) Accepting a better (or possibly worse) solution with a certain acceptance probability based upon the annealing temperature

Now with regards to initial feasible solution computation (step 2, Figure 4.5), the SA scheme calls the *find feasible solution* (FFS) algorithm for the first time, see Figure 4.6. This algorithm basically re-uses the two-stage heuristic in Section 4.3.1 to compute an initial feasible solution. In order to avoid the possibility that the basic algorithm does not find a feasible solution in a single attempt, another parameter $iterNum$ is also introduced to run the algorithm multiple times until a feasible solution is found. If no feasible solution is found after $iterNum$ attempts, the request is rejected, otherwise the algorithm proceeds to set up the annealing phase, i.e., steps 7-9 Figure 4.5.

Next, consider the setup for the annealing phase (steps 7-9, Figure 4.5) Here the FFS computation routine is called again to generate a new neighbor solution, i.e., steps 9-11 Figure 4.6. Each node mapping in $sCurr$ is first converted into a binary number, and

1: Given incoming request $r^n = (G_v^n, c^n, b^n, t_s^n, t_e^n)$, Create three null solution set, $sCurr$ and $sBest$;
2: Call FFS computation routine to find a feasible mapping (see Figure 4.6);
3: **if** FFS computation fails
4: VN request r^n failed
5: Drop r^n
6: **else**
7: **for** $Temp = T_{max}; Temp > T_{min}; Temp = \mu Temp$
8: Call FFS computation routine again to find feasible mapping from neighbors of $sCurr$
9: Update $sCurr$ and $sBest$ for each loop
10: Reserve node mapping of $sBest$ and reserve routed link resources

Figure 4.5: Simulated Annealing Meta-Heuristic Algorithm

a neighboring solution is then generated by randomly shifting bits in $sCurr$. Finally, the feasibility of this new (candidate) solution is checked and its acceptability decided upon. Namely, Dijkstra's shortest path algorithm is run for each VN link, and the cost of this new solution is also computed (if all of VN link connections can be routed). As noted above, the SA algorithm can also accept solutions with higher (worse) cost in a probabilistic manner. Namely, the acceptance probability depends upon the cost and the "temperature" parameter, $Temp$, as follows:

$$p = exp\Big(\frac{COST(sCurr) - COST(M)}{Temp}\Big) \qquad \text{(Eq. 4-25)}$$

where $sCurr$ is the current solution and M is the new neighbor feasible solution of $sCurr$. Based upon the acceptance decision, $sCurr$ is updated if the new solution is accepted, and $sBest$ is also updated if the new solution is better than before. Overall, the system

```
1: Given sCurr and sBest, set iteration number iterNum
2: if sCurr = NULL
3:    for iterNum times loop
4:       Find solution M using two-stage Random Pick (RP) algorithm in Section 4.3.1
5:       if Find feasible solution
6:          Save M as sCurr and sBest
7:          break
8: else
9:    for iterNum times loop
10:      Convert each node mapping in sCurr to binary coding
11:      Randomly shift bits of sCurr to get a new neighbor solution M
12:      if (M is feasible) && (COST(M) is acceptable)
13:         Save M as sCurr
14:         if COST(M) < COST(sBest)
15:            Save M as sBest
16:         break
17: if Find feasible solution M
18:    Return TRUE;
19: else
20:    Return FALSE;
```

Figure 4.6: Find Feasible Solution (FFS) Computation Algorithm

"temperature" is initialized to T_{max}. When this value reaches T_{min}, the node mappings in $sBest$ are saved and reserved along with their routed link resources. Note that both the loop iteration numbers, i.e., parameter $iterNum$, and annealing phase $Temp = \mu Temp$ can be specified manually at each step in order to control the total compute time.

4.5 Performance Evaluation

The performance of proposed VN advance reservation schemes are now tested using *OPNET Modeler*^{TM} for the same 16-and 24-node topologies shown in Figure 3.8. All substrate nodes have 1,000 units of generic resource capacity and all physical links have 10,000 units of bandwidth. Also, client VN request sizes vary between 4-7 nodes with an average VN node degree of 2.6, i.e., computed as the ratio of VN links to VN nodes. Meanwhile, the average requested VN node capacity is uniformly distributed between 1-30 units, and the average requested VN link capacity is uniformly distributed between 100-1,000 units. The constants $\mathbb{B}$ and $\mathbb{C}$ in Eq. 4-1 and Eq. 4-2 are also set to 1,000 and 30 respectively, and ρ and π are set to unity. As per Chapter 3, all requests arrive in a "on-line" manner and have exponential holding and inter-arrival times, with means μ and λ. Namely, a value of $\mu = 600$ time units is chosen here and λ is varied per desired load. Based on this, the total load is defined by using a modified Erlang definition (as per Eq. 3-11, Section 3.4) as follows:

$$\text{Modified Erlang load} = \frac{1}{4}\sum_{n=4}^{7}(n-1) \times \mu/\lambda \qquad \text{(Eq. 4-26)}$$

Furthermore, all tests are done for 10,000 randomly-generated "on-line" VN requests, i.e., no a-priori knowledge of upcoming requests. Carefully note that the *CPLEX* optimization tool is also incorporated with *OPNET Modeler*^{TM} in order to solve the ILP formulation in Section 4.2. However, due to the massive computational complexity, this ILP is only formulated for the new incoming VN request. Finally, the SA scheme chooses T_{max} and T_{min} values

of 100 and 0, respectively. Furthermore, the annealing phase is set to $Temp = 0.9Temp$, and for each iteration $iterNum = 10$, i.e., the FFS computation routine is run at most 10 times to find an acceptable solution for each temperature. Detailed results and findings are now presented.

4.5.1 Blocking Rate Performance

The request blocking rates for the various VN scheduling schemes are shown in Figure 4.7. Foremost the findings for both the 16-node and 24-node topologies indicate that the ILP optimization and SA meta-heuristic schemes give the lowest blocking of all. This behavior is expected since the ILP is a local optimal strategy. More importantly, the SA meta-heuristic scheme gives almost identical performance to the optimization scheme. This indicates that the SA model can achieve a near-optimal performance.

Meanwhile, the results for the graph-based heuristic schemes also indicate improved blocking rates with the single-stage scheme. Namely, the findings indicate very little separation between the MN, MP and RP schemes in the 16-node network, i.e., with MN and MP giving about 15% lower blocking than the RP scheme. Meanwhile, the more selective (load-aware) MB and single-stage schemes also give better overall performance. More importantly, the blocking gains with these schemes are even more pronounced in the larger 24-node topology, i.e., about 90% lower blocking than the RP, MN and MP heuristics at low-medium loads. These gains can be attributed to the fact that the larger topology has more substrate nodes and links, and this allows load-aware schemes to utilize network re-

sources in a more effective manner. Meanwhile, the MP scheme does slightly better than the RP and MN schemes. These results clearly indicate that node mapping schemes can have a huge impact on VN scheduling performance, i.e., jointly taking into account node and link load-balancing constraints into the node selection process yields better (lower) blocking ratios. Finally, the SA scheme also gives lower blocking than all the graph-based heuristics across all input loads, with only slightly longer computation times as a trade-off. Hence the meta-heuristic strategy offers a very competitive solution to the VN scheduling problem.

4.5.2 Long Term Revenue

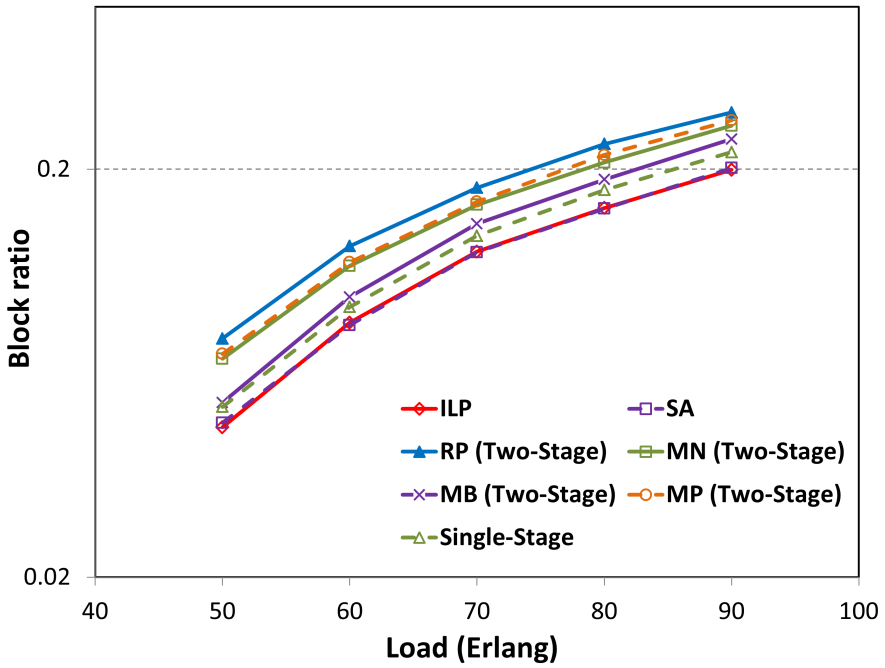
The long term revenues are also plotted in Figure 4.8 and again confirm improved values with the ILP optimization and SA meta-heuristic strategies. For example, in the smaller 16-node topology, the ILP and SA solutions give over 20% higher revenues than the two-stage RP scheme (and this separation increases further in the larger topology, see Figure 4.8b). In addition, the single-stage heuristic scheme also gives significantly higher revenues versus the MB scheme (the best two-stage scheme). Meanwhile, the two-stage RP and MN schemes tend to give the lowest revenues in both networks. Since revenue is directly related to the VN topology, these findings also indicate that the ILP and SA meta-heuristic schemes can support larger VN request sizes versus graph-based heuristic methodologies.

4.5.3 Network Resource Utilization Efficiency

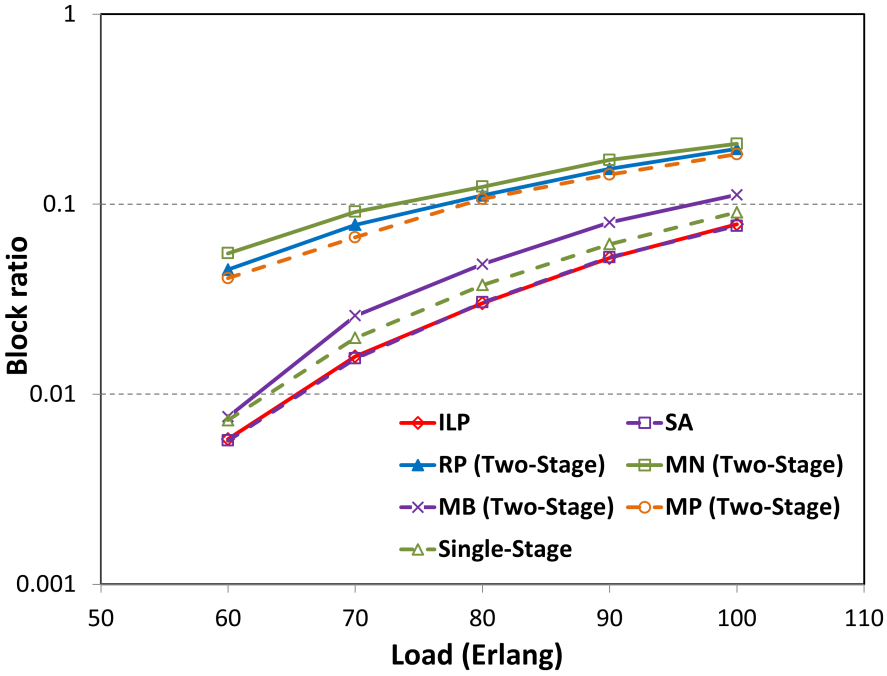
The efficiency of network resource utilization is also measured by plotting the average path length for each VN link connection in Figure 4.9. These results indicate that the

ILP optimization scheme achieves much lower bandwidth utilization cost per successfully-scheduled VN. However the SA meta-heuristic gives slightly longer VN link routes compared to the ILP optimization, i.e., about 40% longer in both topologies. On the other hand, the single-stage scheme gives the lowest utilization amongst all the heuristic methods, i.e., approximately 15% longer VN link routes than SA scheme in the 16-node network and less than 10% longer VN link routes in larger 24-node network. Furthermore, the two-stage MN scheme also gives the shortest VN link routes across all two-phase schemes. This is expected since choosing nodes with the maximum number of neighbors tends to schedule the whole VN in a smaller localized area (which on the other hand also results in congestion and higher blocking, see Figure 4.7 (a), (b)). Carefully note that the VN link length utilization does not increase much with larger VN request loads, indicating algorithmic consistency across all tested schemes.

Finally, to show network resources utilization more clearly, revenue-cost ratios are also plotted in Figure 4.10, i.e., higher ratios indicating increased network resource utilization. The findings here show the highest revenue-cost ratio for the ILP scheme, followed by the SA meta-heuristic and single-stage heuristic. For example, the ILP approach achieves an average revenue-cost ratio of 90%, whereas the SA meta-heuristic scheme yields one closer to 70% across all loads. These results confirm the same network resources utilization efficiency as VN link length results (see Figure 4.9). Finally, these findings also indicate that the ILP optimization and SA meta-heuristic schemes can support more VN requests (customers).

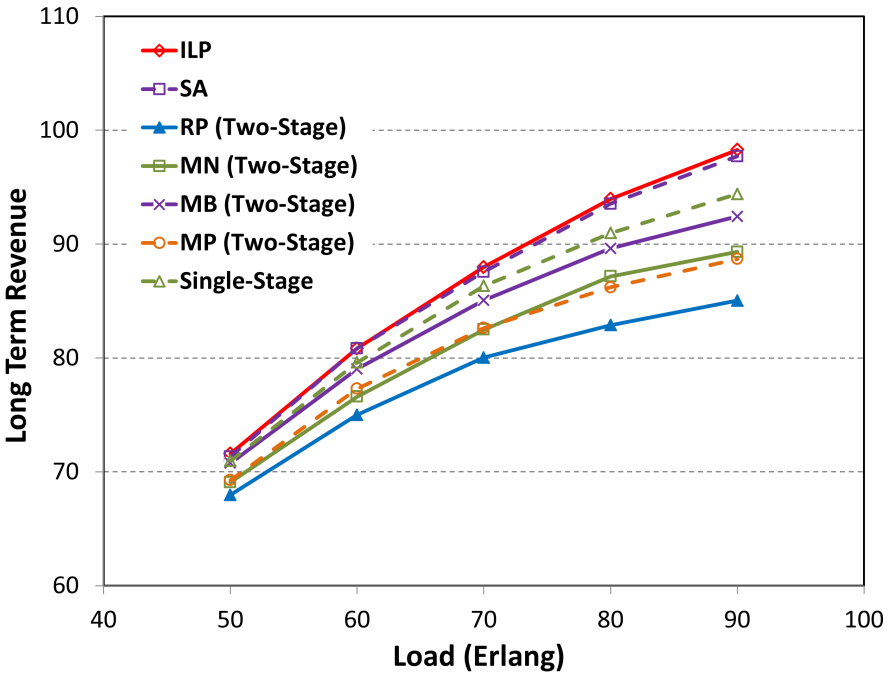


(a)

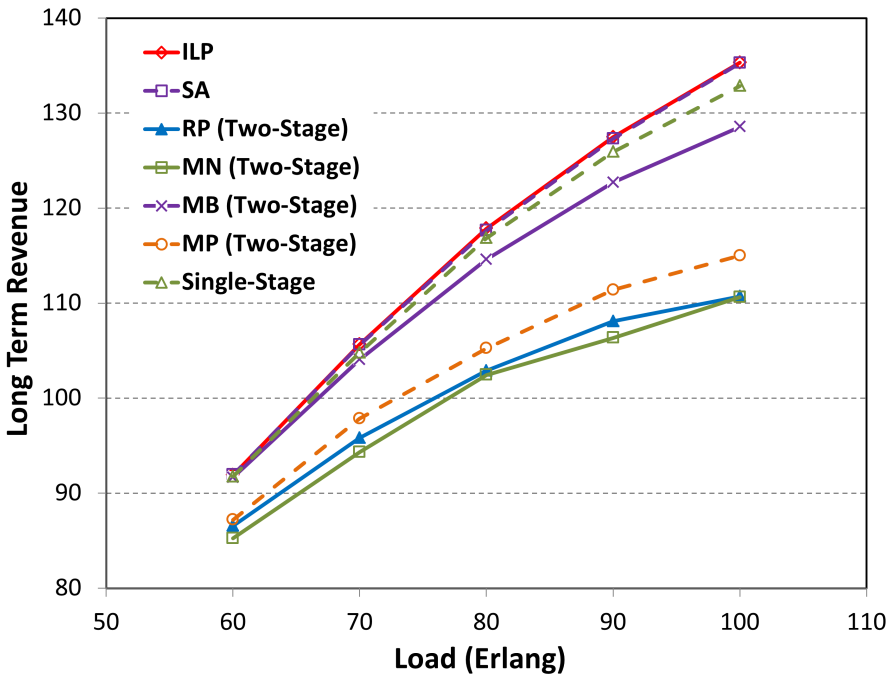


(b)

Figure 4.7: Request Blocking Rate: a) 16-Node Topology, b) 24-Node Topology

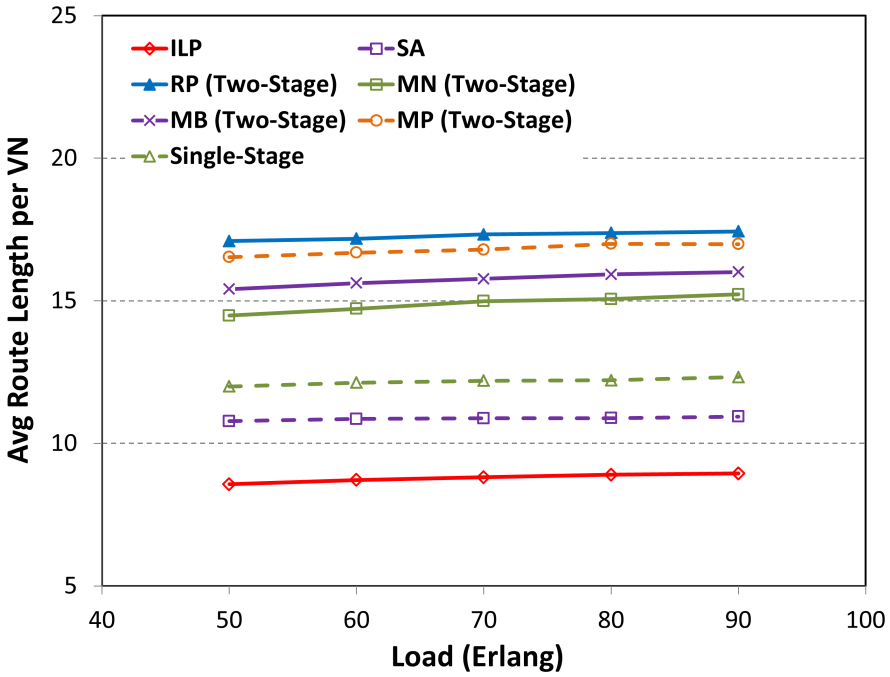


(a)

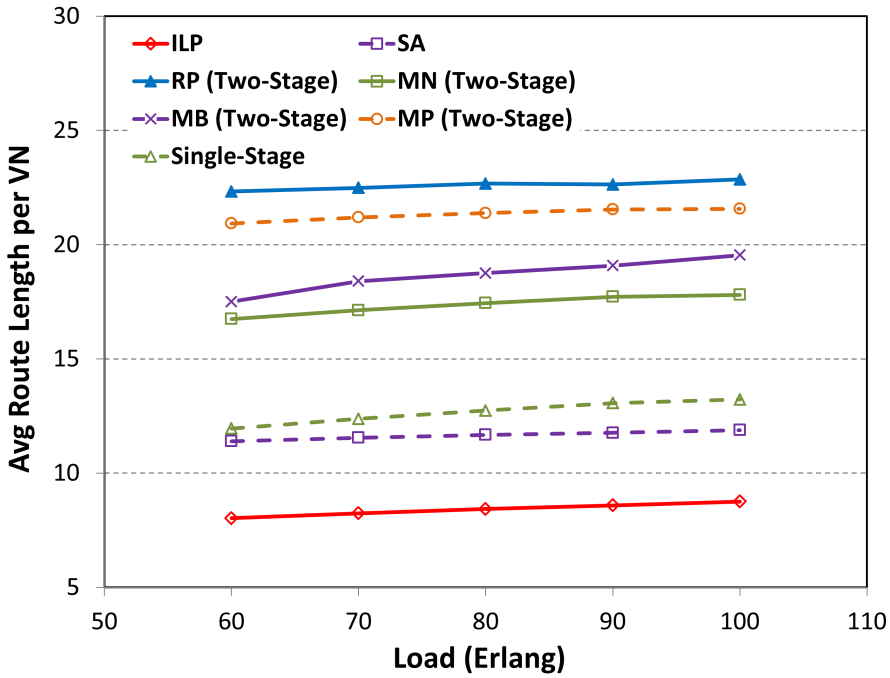


(b)

Figure 4.8: Long Term Revenue: a) 16-Node Topology, b) 24-Node Topology

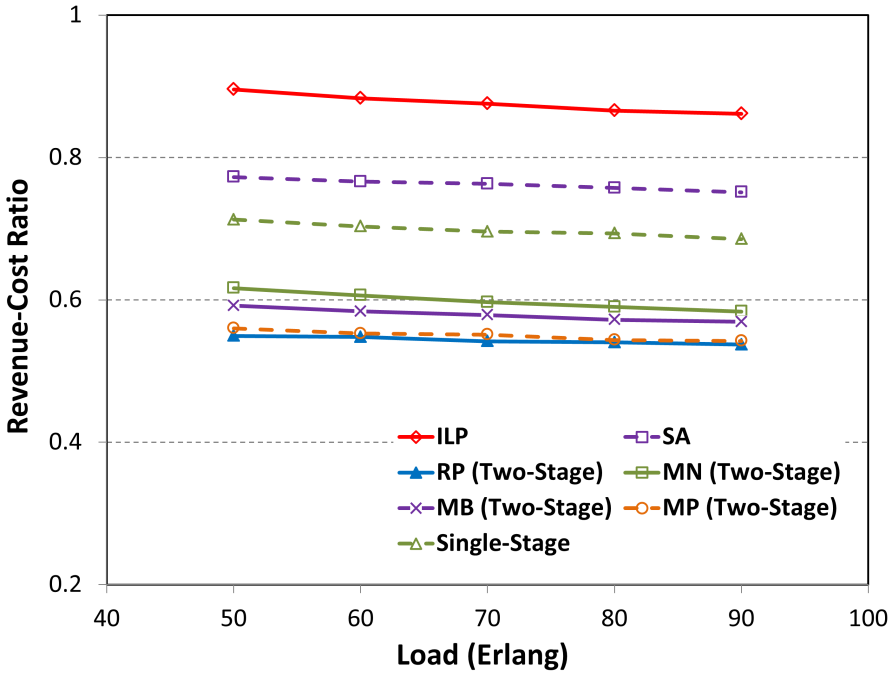


(a)

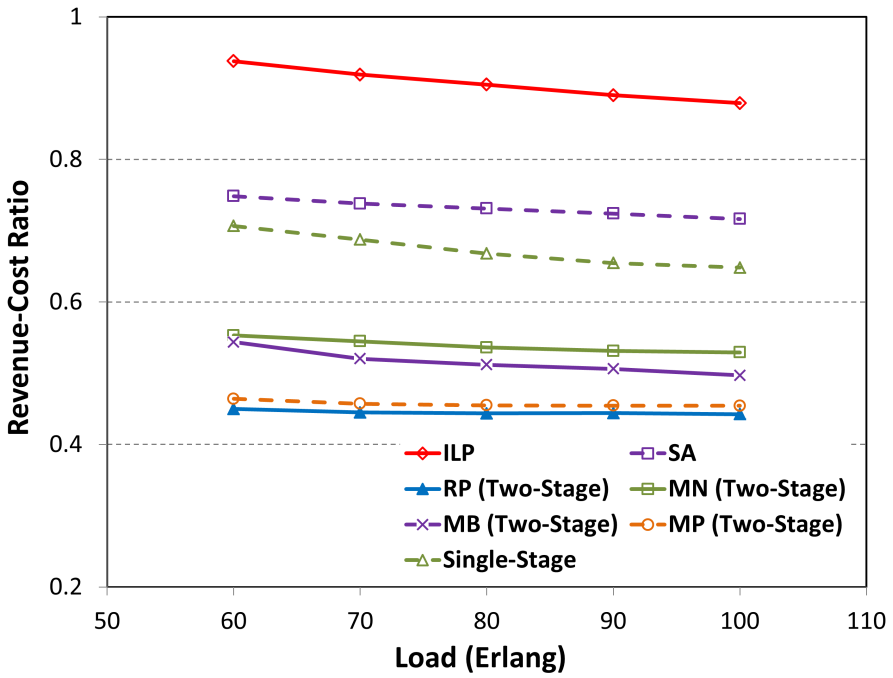


(b)

Figure 4.9: Average VN Link Length: a) 16-Node Topology, b) 24-Node Topology



(a)



(b)

Figure 4.10: Revenue-Cost Ratio: a) 16-Node Topology, b) 24-Node Topology

Chapter 5 Flexible AR Models for Virtual Network Scheduling [1]

In general, simple binary accept/reject decisions may not provide the most flexible option for service providers and their clients. For example, many customers may be willing to accept some form of "partial" VN service setup if their full request cannot be met, i.e., reduced VN node or VN link resource levels, even truncated VN topologies, etc. At the same time, service providers can realize improved revenues and infrastructure utilization by providing partial services support.

Now earlier studies in AR scheduling have looked at more flexible service models for point-to-point connections. For example , [JX01] modifies requests by varying both the time (start, stop) and resource (requested bandwidth) dimensions. The overall aim here is to introduce added flexibility but still guarantee the total amount of data transfered. Overall results show significant improvements in acceptance rates and bandwidth usage. However, as noted this model only considers point-to-point connection demands (single-path). Indeed, multi-node and multi-link VN requests allow much more flexibility in terms of service modification.

In light of the above, this chapter presents two generalized AR schemes based upon more flexible service models. First, a *priority-based reservation* (PBR) scheme is proposed

[1]This chapter was previously published in [HB03]. Permission is included in Appendix B.

to assign different priority levels to VN nodes and VN links within a VN request. Next, a *capacity-based reservation* (CBR) scheme is also developed to modify requested VN resource levels and achieve a trade-off with request durations. Consider the details below.

5.1 Network Model and Description

Overall, the proposed flexible VN scheduling framework re-uses the same notation and network model introduced in Section 4.1. Furthermore, many of the performance evaluation metrics defined for VN scheduling are also applicable here, particularly modified revenue/cost and long term revenue and cost, see Eqs. 4-1 - 4-4. However, some further VN request definitions are still required in order to properly differentiate the flexible AR service models. These modifications will be detailed along with their associated provisioning algorithms.

5.2 Flexible AR Model

As mentioned earlier, many clients will be willing to accept some level of service provisioning flexibility. For example, it may be amendable to only provision a portion of an incoming VN request under high-load or failure conditions. Along these lines, two *partial* VN scheduling policies are also proposed here. Namely, the PBR scheme assigns different priorities to VN nodes and VN links, and schedules these priorities in separate stages. Meanwhile, the CBR scheme adjusts request durations to meet revenue expectations.

5.2.1 Baseline Algorithm

Technically, any of the schemes introduced in Chapter 4 can be used as a baseline VN scheduling algorithm. Here the simple RP heuristic scheduling heuristic in Section 4.3.1 is chosen here and used for subsequent comparison purposes.

5.2.2 Priority-Based Reservation (PBR) Scheme

This scheme assumes that all VN nodes/links are assigned two different priority levels, i.e., high or low, as specified by operators or their clients themselves. The overall pseudocode for the PBR solution is shown in Figure 5.1. The scheme tries to schedule all higher priority nodes and links first. If this procedure is unsuccessful, the VN request is turned down. Subsequently, attempts are made to schedule as many of the lower priority VN nodes and VN links as possible. Specifically, unmapped VN nodes are first sorted in decreasing order of their node rank. A set of candidate nodes is also derived for each of these VN nodes by selecting physical nodes with sufficient available resource levels and available bandwidth levels on adjacent links. Using this information, an unmapped VN node is mapped to a candidate physical node with the lowest link costs to each mapped VN neighbor node (see Eq. 4-8 and 4-9). The VN link connections between this selected node and already-mapped neighboring VN nodes are then routed/scheduled. Note that evaluation metrics (such as revenue or cost) can only be calculated for successfully provisioned portions of a VN request instead of the whole request.

1: Given incoming request $r^i = (G_v^i, c^i, b^i, t_s^i, t_e^i)$, generate temporary graph copy $G'_s(V'_s, E'_s) = G(V, E)$
2: Remove non-feasible nodes and links in $G'_s(V'_s, E'_s)$, i.e., nodes $rem_v(t) < c^n$ and links $rem_e(t) < b^n$ in $[t_s^i, t_e^i]$;
3: Attempt to schedule higher priority VN nodes any baseline algorithm
4: **if** Failed
5: VN request r^i failed
6: Discard $G'_s(V'_s, E'_s)$ and temporary node mapping array
7: **else**
8: Setup Successful
9: Sort the virtual nodes based upon node rank
10: **for** Each unmapped VN node
11: Compute candidate physical substrate nodes
12: Compute cost (route length) from candidate nodes to mapped VN nodes
13: Select the candidate node with the minimum cost
14: Run Dijkstra's shortest-path between chosen node and mapped substrate nodes
15: **if** All VN link connections routed
16: Reserve mapped node and link resource in $G_s(V_s, E_s)$

Figure 5.1: Priority-Based Reservation (PBR) Scheme

5.2.3 Capacity-Based Reservation (CBR) Scheme

This approach pursues a compromise between assigned resource levels and request durations. Namely, this algorithm is only triggered when an incoming VN request cannot be reserved using the baseline algorithm. Specifically, all the VN node and VN link resources requested in r^i are scaled by a fraction, σ, $0 < \sigma < 1$. To compensate for this reduction, the request duration is appropriately extended by a factor of $1/\sigma$ as well. Here the baseline algorithm is used to initially schedule the VN request. If this setup fails, the VN request is adjusted from r^i to $r_{mod}^i = (G_v^i, \sigma c^i, \sigma b^i, t_s^i, t_s^i + 1/\sigma(t_e^i - t_s^i))$, and the baseline algorithm is re-

run. Note that this reservation model is very amendable to data transfer services requesting variable or extended transfer times.

1: Given incoming request $r^i = (G_v^i, c^i, b^i, t_s^i, t_e^i)$, generate temporary graph copy $G'_s(V'_s, E'_s) = G(V, E)$
2: Remove non-feasible nodes and links in $G'_s(V'_s, E'_s)$, i.e., $rem_v(t) < c^n$ and $rem_e(t) < b^n$ in $[t_s^i, t_e^i]$;
3: Attempt to schedule the VN request r^i using baseline algorithm
4: **if** Failed
5: Adjust resource and duration by σ and $1/\sigma$, $r_{mod}^i = (G_v^i, \sigma c^i, \sigma b^i, t_s^i, t_s^i + 1/\sigma(t_e^i - t_s^i))$
6: Generate another temporary copy of network $G''_s(V'_s, E'_s)$ where $rem_v(t) < \sigma c^n$ and $rem_e(t) < \sigma b^n$ in $[t_s^i, t_s^i + 1/\sigma(t_e^i - t_s^i)]$
7: Attempt to schedule modified VN request r_{mod}^i using baseline algorithm
8: **if** Failed
9: VN request failed
10: Discard $G'_s(V'_s, E'_s)$ and temporary node mapping array
11: **else**
12: Setup Successful
13: Reserve mapped nodal resources (σc^i) from node mapping array in $G_s(V_s, E_s)$
14: Reserve routed link resources (σb^i) in $G_s(V_s, E_s)$
15: **else**
16: Setup Successful
17: Reserve mapped nodal resources from node mapping array onto $G_s(V_s, E_s)$
18: Reserve routed link resources in $G_s(V_s, E_s)$

Figure 5.2: Capacity-Based Reservation (CBR) Scheme

5.2.4 Complexity Analysis

Now consider the overall run-time complexity of the flexible AR service provisioning schemes. As shown in Section 4.3.1, the two-stage baseline scheme has $O(2N|E_s| + N|V_s| + |V_v|(|V_s| + log|V_v|) + |E_v|(|E_s| + |V_s|log|V_s|))$ computation complexity. Similarly, provisioning high-priority VN nodes and VN links gives similar complexity for the PBR scheme.

Moreover, scheduling low-priority VN nodes and VN links also entails similar computational complexity as the single-stage VN scheduling scheme (Section 4.3.2), i.e., bounded by $O(|E_v|(2+|V_s|)|V_s||E_s|log|V_s|)$, where VN node and VN link resources are reserved at the same stage per un-mapped VN node. Therefore, depending upon the number of high and low priority VN nodes (links) in a VN, the total run-time complexity of the PBR scheme is bounded between $O(2N|E_s|+N|V_s|+|V_v|(|V_s|+log|V_v|)+|E_v|(|E_s|+|V_s|log|V_s|))$ and $O(N(2|E_s|+|V_s|)+|V_v|log|V_v|+|E_v|(2+|V_s|)|V_s||E_s|log|V_s|)$.

Meanwhile, the CBR scheme basically runs the two-stage baseline algorithm twice with two different VN requests. Therefore the total run-time complexity of this scheme is the same as the two-stage baseline scheme, i.e., bounded by $O(2N|E_s|+N|V_s|+|V_v|(|V_s|+log|V_v|)+|E_v|(|E_s|+|V_s|log|V_s|))$.

5.3 Performance Evaluation

The two flexible VN scheduling models are now tested in *OPNET Modeler*^{TM} using the earlier-defined 16-node and 24-node substrate topologies shown in Figure 3.8. Again, all substrate nodes have 100 units of capacity and all subtrate links have 10,000 units of bandwidth. Meanwhile, VN requests are also generated by composing random graphs with 4-7 nodes each, with an average node degree of 2.6. The requested VN node capacities are also distributed uniformly between 1-30 units, and VN link capacities are distributed uniformly between 100-1,000 units. All VN requests have exponentially-distributed holding and inter-arrival times, with means μ and λ. Again a value of $\mu = 600$ time units is used

here, and λ is adjusted accordingly per load as Section 4.5. Accordingly, $\mathbb{B}$ is set to 1,000, and $\mathbb{C}$ is set to 30 to normalize revenue and cost computation. Both ρ and π in Eq. 4-1 and Eq. 4-2 are also set to unity. Furthermore, in the PBR scheme up to 3 VN nodes and VN links are assigned high priority levels, see Figure 5.3. Meanwhile, for the CBR scheme, 3 different values of σ are chosen to gauge performance sensitivity, i.e., 0.2, 0.5, 0.8.

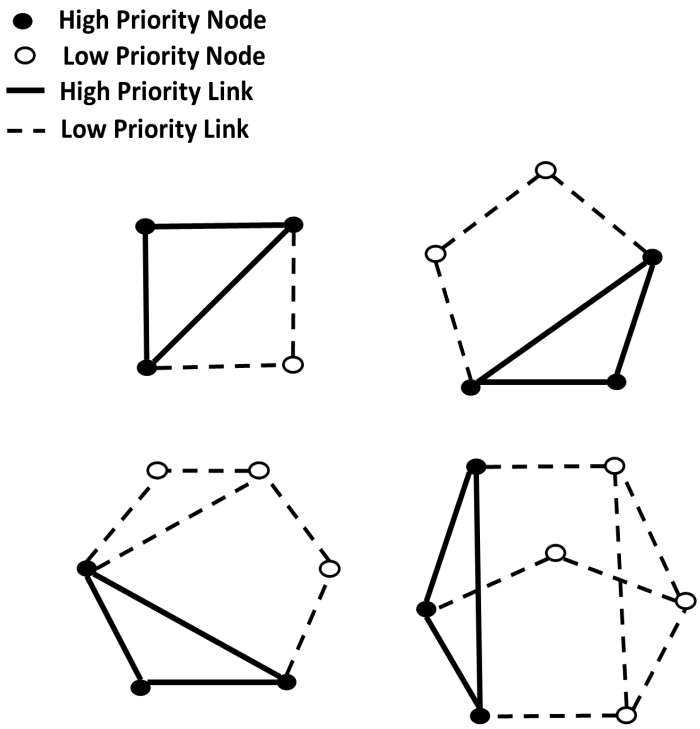


Figure 5.3: Prioritized VN Requests

5.3.1 Blocking Rate Performance

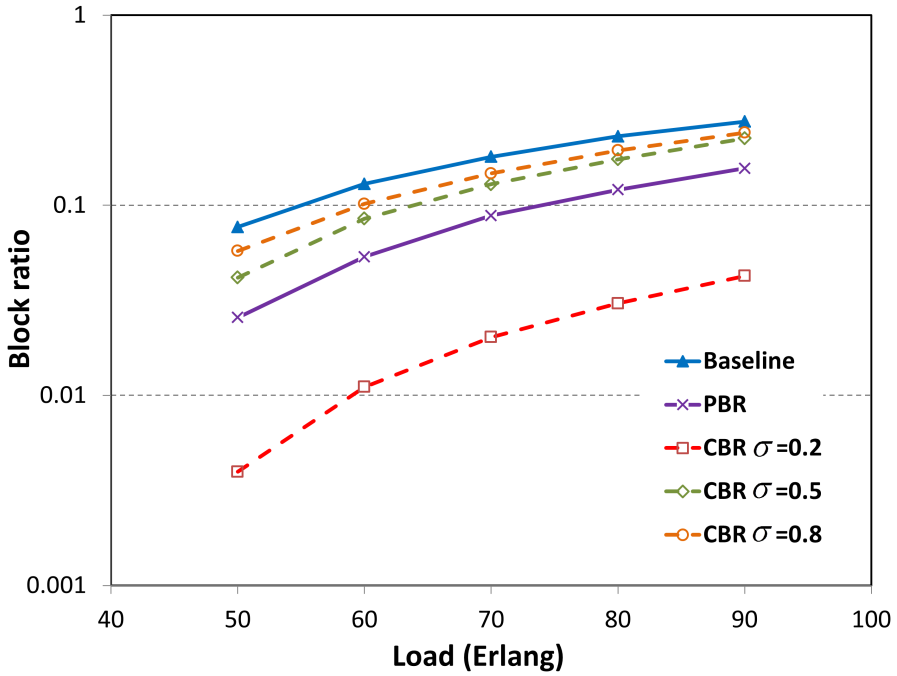
Initial tests are done to measure VN request blocking rates, see Figure 5.4. Overall findings show much lower blocking performances with both the PBR and CBR schemes. Specifically, the PBR scheme yields nearly half the blocking rate of the baseline scheme in the 16-node topology, and this difference increases further in the larger 24-node topology, see Figure 5.4 (b). Meanwhile, the CBR scheme gives improved blocking reduction for the smaller value of σ. Here decreasing requested VN resource levels also creates more available substrate resources for new requests to be accepted.

5.3.2 Long Term Revenue

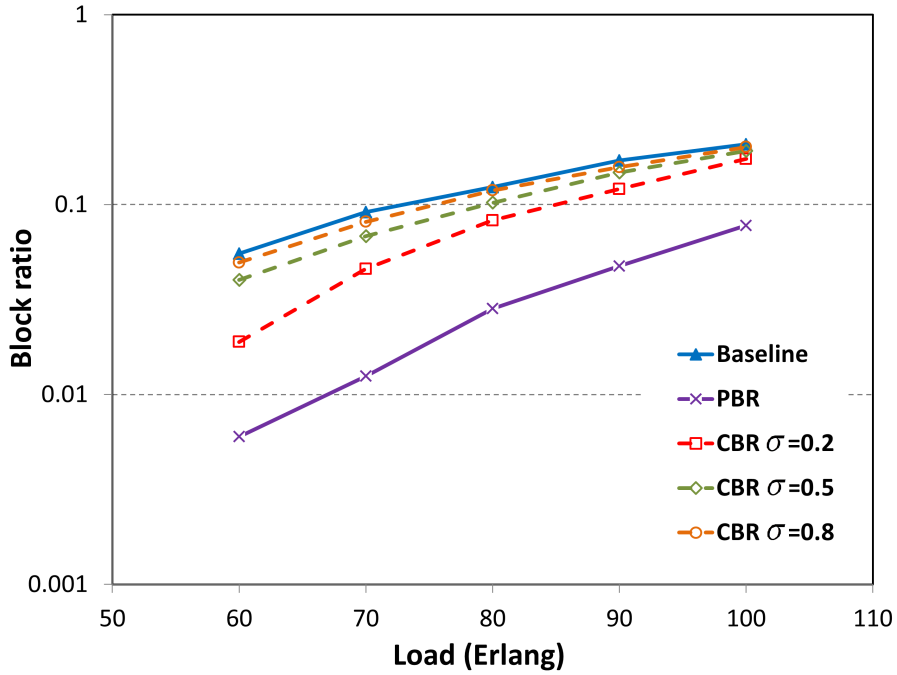
The long term revenues are also plotted in Figure 5.5 and show much improved performance with PBR and CBR schemes. Foremost, smaller σ values for the CBR scheme yield the highest revenues. Meanwhile, the long term revenues for the PBR scheme in the smaller 16-node topology are slightly less than those with the CBR scheme (with $\sigma = 0.2$). This reduction occurs since the smaller network can only schedule the high priority portions of incoming VN requests under higher loads. However, with more available substrate resources in a larger network, more VN requests can likely be accommodated in the original setup attempt. These results indicate that the PBR scheme is more effective in larger networks. Regardless, both of these flexible VN service schemes provide significant performance and revenue improvements versus the baseline strategy.

5.3.3 Network Resource Utilization Efficiency

Finally, the average link lengths for successful VN requests (link connections) are also plotted in Figure 5.6, along with revenue-cost ratios in Figure 5.7 (to gauge network resource utilization efficiency). These findings indicate that the PBR scheme achieves the highest overall resource utilization. Namely, this scheme gives approximately 50% shorter VN link routes, and over 20% higher revenue-cost ratios compared to the RP baseline scheme. Conversely, the CBR scheme shows very small separation for different values of σ. However, these values still show notably lower resource utilization (than the baseline scheme) for both flexible VN AR schemes.

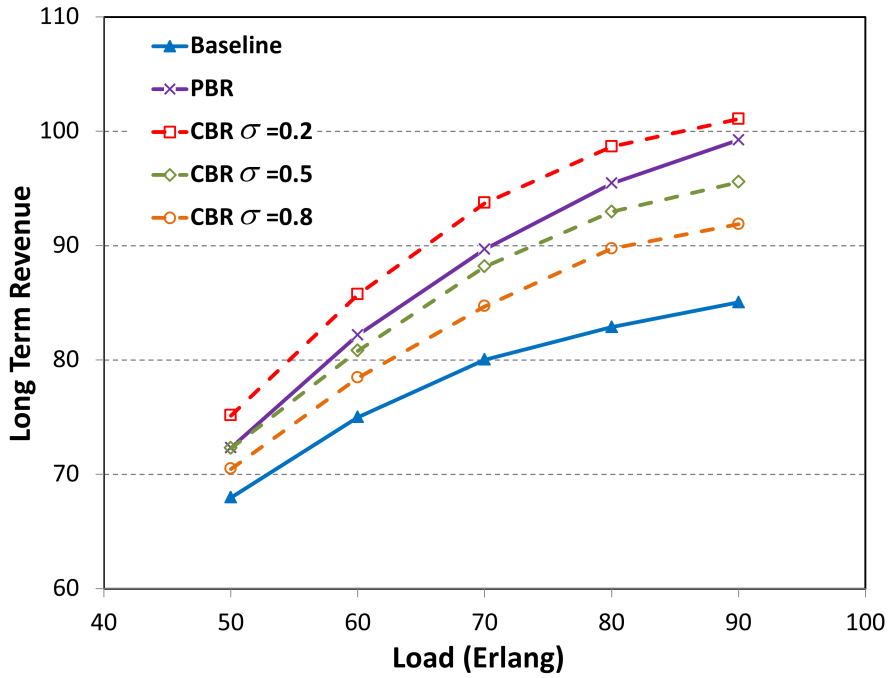


(a)

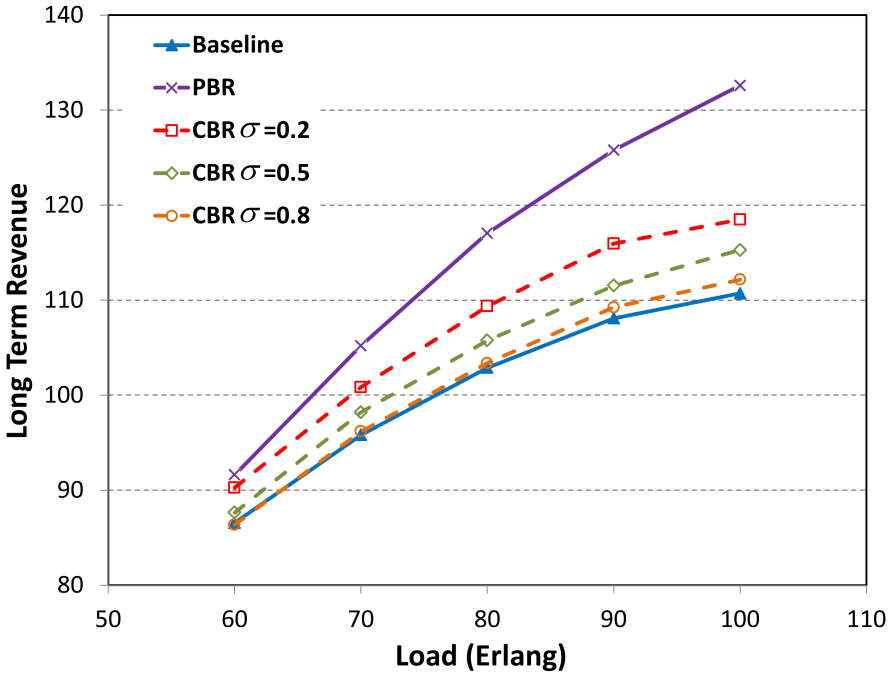


(b)

Figure 5.4: Request Blocking Rate: a) 16-Node Topology, b) 24-Node Topology

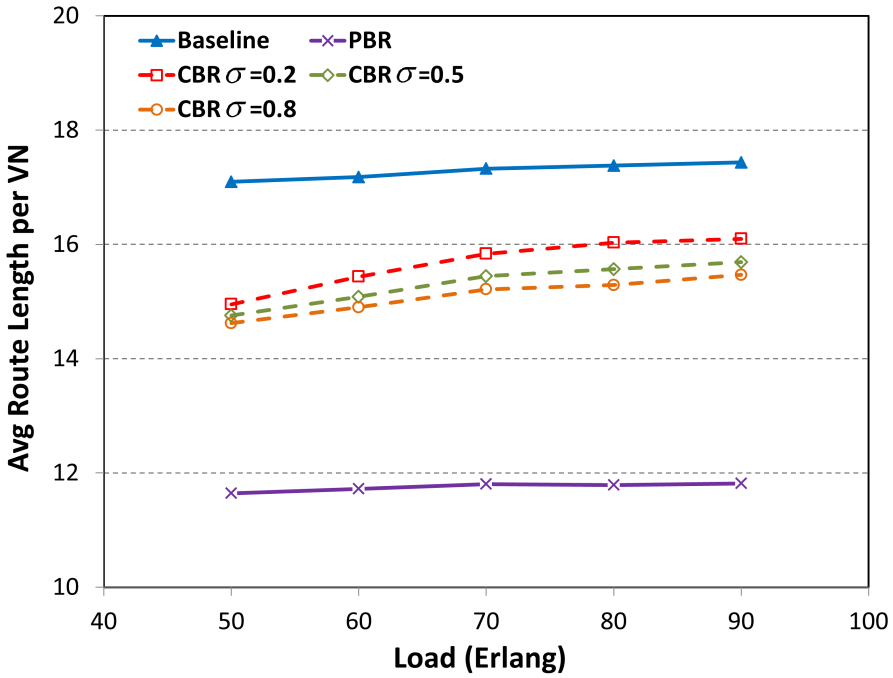


(a)

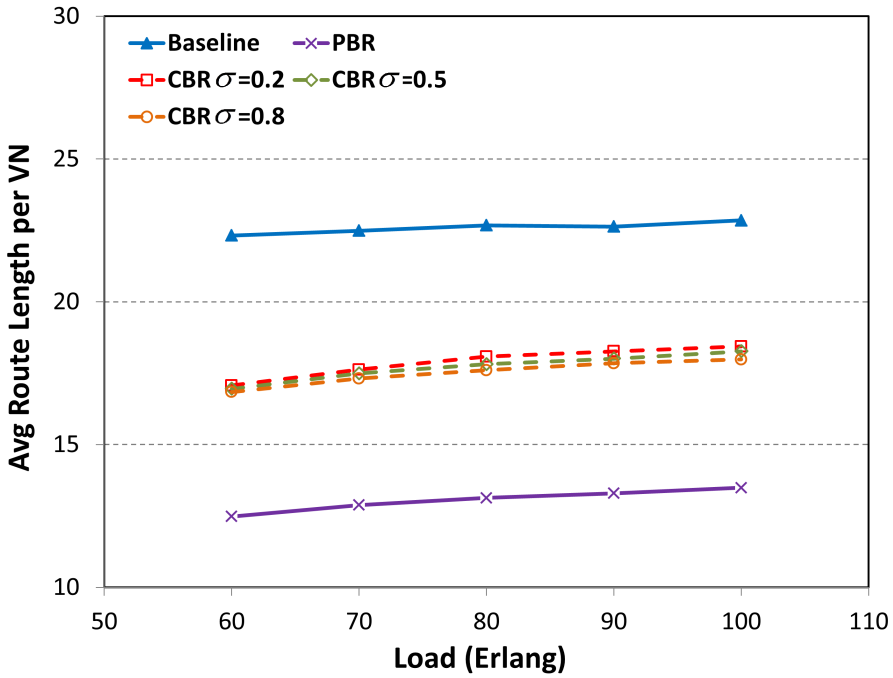


(b)

Figure 5.5: Long Term Revenue: a) 16-Node Topology, b) 24-Node Topology

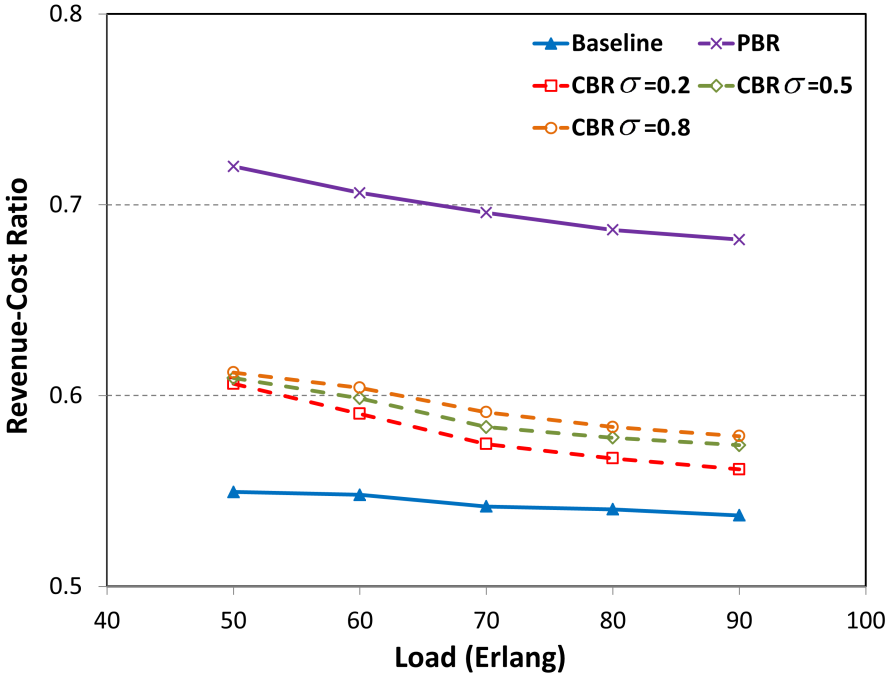


(a)

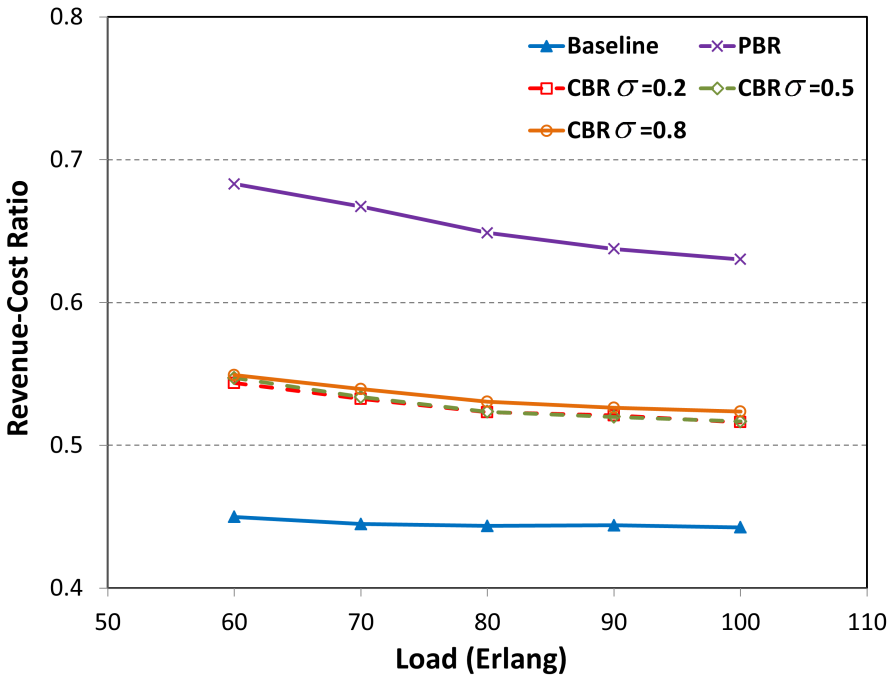


(b)

Figure 5.6: Average VN Link Length: a) 16-Node Topology, b) 24-Node Topology



(a)



(b)

Figure 5.7: Revenue-cost Ratio: a) 16-Node Topology, b) 24-Node Topology

Chapter 6 Conclusions and Future Work

This dissertation presents one of the first studies on AR scheduling for virtual network (VN) services. First, Chapter 2 presents a review of some related background work in the field. Next, the VN scheduling problem with fixed node locations is studied in Chapter 3 and several new optimization and heuristic methods are proposed. Subsequently, the more generalized VN scheduling problem is introduced/formulated in Chapter 4, and several solution strategies are proposed and evaluated. Namely, these solutions include optimization, meta-heuristic, and graph-based heuristic methods. Finally, Chapter 5 presents some flexible VN service scheduling models in order to lower request blocking ratios and improve operator revenues. The major findings from this research are now presented.

6.1 Conclusions

The work starts by taking a more formal look at the VONS problem in Chapter 3. This problem is a special case of VN scheduling problem in which virtual nodes locations are pre-determined and do not require any resources, i.e., only batch scheduling of virtual link connections. Namely, a modified dynamic ILP formulation is first outlined to pursue an optimization strategy that minimizes resource usage over a reduced time window. Specifically, this window spans the request interval and any time-overlapping (accepted, inactive) requests. To further simplify variable count (computational) complexity, a further state-

based abstract ILP model is also presented. However, since these optimization formulations still pose notable scalability challenges, three different heuristic techniques are proposed to improve upon the greedy schemes presented in [FG01]. In particular the goal here is to use re-routing techniques to re-map accepted overlay requests in order to free up more resources for new demands, i.e., much in the same way the dynamic ILP operates. Overall, some of the key contributions and findings from this study are as follows:

- The modified dynamic ILP optimization model treats a subset of time-overlapping requests. Here each incoming VONS request is considered in isolation over a reduced time windows in order to lower variable count complexity.
- The state-based ILP solution greatly improves the computational scalability by reducing the number of considered time-slot variables. This approach significantly outperforms all heuristic methods and can solve relatively larger network sites in a reasonable time, i.e., up to 24 nodes and 43 links.
- The proposed (VN link) re-routing heuristics give notable improvements over the basic minimum hop-count heuristic in [FG01]. In particular, the threshold re-routing (THR) scheme yields slightly lower blocking performance and less re-routing overheads across all three heuristic strategies, i.e., up to 5% lower blocking rates and 50% less total re-routing attempts.

In general, bandwidth provisioning between multiple fixed end-points can impose many service limitations. For example, users may want more capable services that also

provide some data-center resources at the network node sites. Hence Chapter 4 presents a novel scheduling framework for more generalized VN requests. In particular, a "dynamic" ILP optimization formulation is first detailed to process all accepted but inactive AR demands. However, due to excessive complexity by – particularly with the added node placement dimension – alternate heuristic methods are also proposed, i.e., single-stage and two-stage. Since heuristic methodologies cannot guarantee any type of performance bounds, a meta-heuristic solution is also proposed based upon *simulated annealing* (SA) method. This approach achieves a better trade-off between performance and computation complexity. Overall, the findings here indicate:

- The ILP-based solution gives the best performance in terms of setup success rates, long-term revenue, and network resource utilization efficiency. In particular, results from two sample topologies (16 and 24 nodes) indicate up to two times the number accepted requests versus the two-stage heuristic solution. These findings indicate that there is room to further develop improved heuristic strategies.
- The SA meta-heuristic gives almost identical blocking and revenue performance as the ILP optimization scheme with notably lower computation times. Hence this approach provides a very competitive solution, especially as network sizes increase.
- The single-stage heuristic outperforms the simpler two-stage heuristic since it jointly incorporates node and link mapping. Furthermore, the two-stage maximum-bandwidth (MB) scheme (which takes into account link load balancing for node selection) also outperforms the other two-stage heuristics. This improvement occurs since that load-

aware schemes tend to distribute VN requests across the network and avoid localized congestion.

Finally, Chapter 5 develops more flexible service models by modifying VN request profiles. Namely, the priority-based reservation (PBR) scheme assigns different priorities to VN nodes and VN links in a request, i.e., in order to improve setup success rates for critical portions of a VN request. On the other hand, the capacity-based reservation (CBR) scheme pursues a compromise between the assigned resources and request durations to satisfy total transfer demands. Overall, some of the key findings here include:

- Both the PBR and CBR schemes provide significant gains in request acceptance rates and long term revenue. Namely, the PBR scheme accepts two times the number of requests versus the baseline algorithm across all loads in two sample topologies. Meanwhile, the CBR scheme gives the highest acceptance rates for smaller networks.
- The PBR scheme gives the highest overall network resource utilization, i.e., shorter average route lengths and higher revenue-cost ratios, especially in larger network sizes. These results indicate that this scheme is very amendable to realistic environments.

6.2 Future Work

This dissertation introduces some new problems in the area of VN service reservation and opens up several new avenues for future research. Foremost, ILP relaxation methodolo-

gies can be investigated to handle larger network sizes. In addition more flexible VN demand models can also be studied by extending the work in Chapter 5 and considering flexible start and stop windows.

Furthermore, as cloud scheduling paradigms gain traction, related service survivability concerns are also starting to come to the fore. Along these lines, pre-fault protection and post-fault restoration strategies can be developed for the VN scheduling problem. Furthermore, the added timeline-dimension can also be leveraged here to develop appropriate migration strategies.

Finally, it is noted that all the VN scheduling schemes presented in this dissertation only consider single-domain network scenarios. Indeed, there may be an emerging need for multi-domain VN scheduling across different operator domains as well. This extension will require more advanced distributed VN scheduling solutions that operate with reduced levels of "global" network state information. Provisioning requests across multiple domains will also require further protocol interactions between multiple domains, and these topics can be investigated further.

References

[AF01] A. Fischer, *et al,* "Virtual Network Embedding: A Survey". *Communications Surveys & Tutorials, IEEE* 15.4 (2013), pp. 1888–1906.

[CC02] Cheng-Shang Chang, *et al,* "Load Balanced Birkhoff–von Neumann Switches, part I: One-stage Buffering". *Computer Communications* 25.6 (2002), pp. 611–622.

[CX01] C. Xie, *et al,* "Rerouting in Advance Reservation Networks". *Computer Communications* 35.12 (2012), pp. 1411–1421.

[CX02] C. Xie, *et al,* "Traffic Engineering for Ethernet over SONET/SDH: Advances and Frontiers". *Network, IEEE* 23.3 (2009), pp. 18–25.

[DA01] D. Andrei, *et al,* "Integrated Provisioning of Sliding Scheduled Services over WDM Optical Networks [Invited]". *Journal of Optical Communications and Networking* 1.2 (2009), A94–A105.

[DA02] D. Andersen, *et al, Resilient Overlay Networks*. Vol. 35. 5. ACM, 2001.

[DE01] D. Eppstein, "Finding the K Shortest Paths". *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*. IEEE. 1994, pp. 154–165.

[DX01] D. Xie, *et al,* "The Only Constant Is Change: Incorporating Time-varying Network Reservations in Data Centers". *ACM SIGCOMM Computer Communication Review* 42.4 (2012), pp. 199–210.

[ED01] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs". *Numerische mathematik* 1.1 (1959), pp. 269–271.

[EJ01] E. Jung, *et al,* "Performance Evaluation of Routing and Wavelength Assignment Algorithms for Optical Networks". *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*. IEEE. 2008, pp. 62–67.

[FG01] F. Gu, *et al,* "Virtual Overlay Network Scheduling". *IEEE Communications Letters* 15.8 (2011), pp. 893–895.

[FG02] F. Gu, *et al,* "Regional Failure Survivability for Cloud Networking Services Using Post Fault Restoration". *System of Systems Engineering (SoSE), 2013 8th International Conference on*. IEEE. 2013, pp. 229–234.

[FG03] F. Gu, *et al,* "Survivable Cloud Network Mapping for Disaster Recovery Support". *IEEE Transactions on Computers* 64.8 (2015), pp. 2353–2366.

[GS01] G. Sun, *et al,* "Optimal Provisioning for Virtual Network Request in Cloud-based Data Centers". *Photonic Network Communications* 24.2 (2012), pp. 118–131.

[HB01] H Bai, *et al,* "Overlay Network Scheduling Design". *Computer Communications* 82 (2016), pp. 28–38.

[HB02] H Bai, *et al,* "Virtual Network Advance Reservation". *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE. 2015, pp. 84–86.

[HB03] H Bai, *et al,* "Flexible Advance Reservation Models for Virtual Network Scheduling". *Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th*. IEEE. 2015, pp. 651–656.

[HY01] H. Yu, *et al,* "Survivable Virtual Infrastructure Mapping in a Federated Computing and Networking System under Single Regional Failures". *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE. 2010, pp. 1–6.

[HY02] H. Yu, *et al,* "Cost Efficient Design of Survivable Virtual Infrastructure to Recover from Facility Node Failures". *2011 IEEE International Conference on Communications (ICC)*. IEEE. 2011, pp. 1–6.

[JK01] J. Kuri, *et al,* "Routing and Wavelength Assignment of Scheduled Lightpath Demands". *Selected Areas in Communications, IEEE Journal on* 21.8 (2003), pp. 1231–1240.

[JP01] J. Perelló, *et al,* "Optimal Allocation of Virtual Optical Networks for the Future Internet". *Optical Network Design and Modeling (ONDM), 2012 16th International Conference on*. IEEE. 2012, pp. 1–6.

[JX01] J. Xing, *et al,* "Flexible Advance Reservation for Grid Computing". *Grid and Cooperative Computing-GCC 2004*. Springer, 2004, pp. 241–248.

[JZ01] J. Zheng, *et al,* "Supporting Advance Reservations in Wavelength-routed WDM Networks". *Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*. IEEE. 2001, pp. 594–597.

[JZ02] J. Zheng, *et al,* "Routing and Wavelength Assignment for Advance Reservation in Wavelength-routed WDM Optical Networks". *Communications, 2002. ICC 2002. IEEE International Conference on*. Vol. 5. IEEE. 2002, pp. 2722–2726.

[LB01] L. Burchard, *et al,* "Rerouting Strategies for Networks with Advance Reservations". *e-Science and Grid Computing, 2005. First International Conference on*. IEEE. 2005, 8–pp.

[LM01] L. Mai, *et al,* "Exploiting Time-malleability in Cloud-based Batch Processing Systems". *Proceeding of the ACM SIGOPS LADIS Workshop*. Vol. 13. 2013.

[MA01] M. Arregoces, *et al, Data Center Fundamentals*. Cisco Press, 2003. ISBN: 1-58705-023-4.

[MC01] M. Chowdhury, *et al,* "ViNEYard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping". *IEEE/ACM Transactions on Networking (TON)* 20.1 (2012), pp. 206–219.

[MR01] M. Rahman, *et al,* "SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization". *Network and Service Management, IEEE Transactions on* 10.2 (2013), pp. 105–118.

[MR02] M. Rost, *et al,* "It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities". *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE. 2014, pp. 17–26.

[MY01] M. Yu, *et al,* "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration". *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 17–29.

[NC02] N. Charbonneau, *et al,* "A Survey of Advance Reservation Routing and Wavelength Assignment in Wavelength-routed WDM Networks". *Communications Surveys & Tutorials, IEEE* 14.4 (2012), pp. 1037–1064.

[NC03] N. Chowdhury, *et al,* "Virtual Network Embedding with Coordinated Node and Link Mapping". *INFOCOM 2009, IEEE*. IEEE. 2009, pp. 783–791.

[NI01] Virtual Network Service Infrastructure Project. http://www.nict.go.jp/en/nrh/nwgn/nwgn-virtualnetwork.html.

[RG01] R. Guérin, *et al,* "Networks with Advance Reservations: The Routing Perspective". *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 1. IEEE. 2000, pp. 118–127.

[RN01] R. Nejabati, *et al,* "Optical Network Virtualization". *Optical Network Design and Modeling (ONDM), 2011 15th International Conference on*. IEEE. 2011, pp. 1–5.

[RW01] Ronald W Wolff, *Stochastic Modeling and the Theory of Queues*. Pearson College Division, 1989.

[SA01] S. Araujo, *et al,* "A Metaheuristic Approach for the Virtual Network Embedding Problem". *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*. IEEE. 2015, pp. 68–75.

[TC01] T. Cormen, *Introduction To Algorithms*. MIT press, 2009.

[TE01] T. Entel, *et al,* "Scheduled Multicast Overlay for Bandwidth-intensive Applications". *Optical Network Design and Modeling (ONDM), 2012 16th International Conference on*. IEEE. 2012, pp. 1–6.

[TS01] T. Stevens, *et al,* "Multi-cost Job Routing and Scheduling in Grid Networks". *Future Generation Computer Systems* 25.8 (2009), pp. 912–925.

[TW01] T. Wallace, *et al,* "Scheduling Advance Reservation Requests for Wavelength Division Multiplexed Networks with Static Traffic Demands". *Communications, IET* 2.8 (2008), pp. 1023–1033.

[XC01] X. Cheng, *et al,* "Virtual Network Embedding Through Topology-Aware Node Ranking". *ACM SIGCOMM Computer Communication Review* 41.2 (2011), pp. 38–47.

[ZD01] Z. Duan, "Service Overlay Networks: SLAs, QoS, and Bandwidth Provisioning". *IEEE/ACM Transactions on Networking (TON)* 11.6 (2003), pp. 870–883.

[ZZ02] Z. Zhang, *et al,* "A Unified Enhanced Particle Swarm Optimization-based Virtual Network Embedding Algorithm". *International Journal of Communication Systems* 26.8 (2013), pp. 1054–1073.

Appendix A: Glossary

AR	Advance Reservation
BGP	Border Gateway Protocol
CBR	Capacity-Based Reservation
DHT	Distributed Hash Table
DWDM	Dense Wavelengths Division Multiplexing
EC2	Elastic Compute Cloud
FCFS	First Come First Serve
FFS	Find Feasible Solution
GRASP	Greedy Randomized Adaptive Search Procedure
IaaS	Infrastructure as a Service
ILP	Integer Linear Programming
InP	Infrastructure Provider
IR	Immediate Reservation
ISP	Internet Service Provider
k-SP	K-Shortest Paths
LAN	Local Area Network
LB	Load Balancing
LP	Linear Programming
MANETs	Mobile Ad-hoc Networks
MB	Maximum Bandwidth

MCF	Multi-Commodity Flow
MHR	Minimum Hop Re-Routing
MILP	Mixed Integer Linear Programming
MIP	Mix Integer Programming
MN	Maximum Neighbors
MNR	Minimum Number Re-Routing
MP	Maximum Product
MPLS	Multiprotocol Label Switching
NSVIM	Non-Survivable Virtual Infrastructure Mapping Algorithm
P2P	Peer-to-Peer
PaaS	Platform as a Service
PBR	Priority-Based Reservation
QoS	Quality of Service
RON	Resilient Overlay Network
RP	Random Pick
RVNS	Reduced Variable Neighborhood Search
RWA	Routing and Wavelength Assignment description
SaaS	Software as a Service
SDH	Synchronous Digital Hierarchy
SON	Service Overlay Network
SONET	Synchronous Optical Networking
SP	Service Provider
STSD	Specified Start Specified Duration
STUD	Specified Start Unspecified Duration
THR	Threshold Re-Routing

UFP	Unsplittable Flow Problem
UTSD	Unspecified Start Specified Duration
VM	Virtual Machine
VN	Virtual Network
VNE	Virtual Network Embedding
VoIP	Voice over IP
VON	Virtual Overlay Network
VONS	Virtual Overlay Network Scheduling
VPN	Virtual Private Network

Appendix B: Copyright Permissions

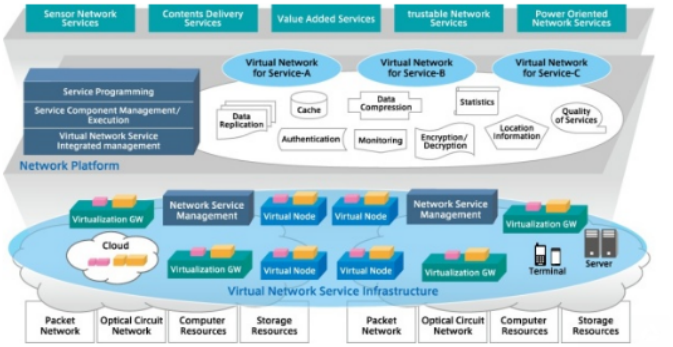
Below is the permission for the use of Figure 1.1 in Chapter 1.

Permission to use one of your figures in my thesis Inbox x

Hao Bai <[redacted]>
to publicity

Dear sir or madam

I am a graduate student from University of South Florida in US. I am working on my final thesis. I am writing to you to get your permission for me to use one of the figures to describe the virtual network service infrastructure from http://www.nict.go.jp/en/nrh/nwgn/nwgn-virtualnetwork.html.



Would you please kindly reply to me with your permission?

Thank you.

Best Regards

清川 雅博 <kiyokawa@nict.go.jp>
to publicity, nwgn-mado, me

Dear Bai Hao,

Thank you for your inquiry.

I confirmed with people concerned, and we are pleased for your using the figure from http://www.nict.go.jp/en/nrh/nwgn/nwgn-virtualnetwork.html.
in usual way of citation; putting the URL, etc.

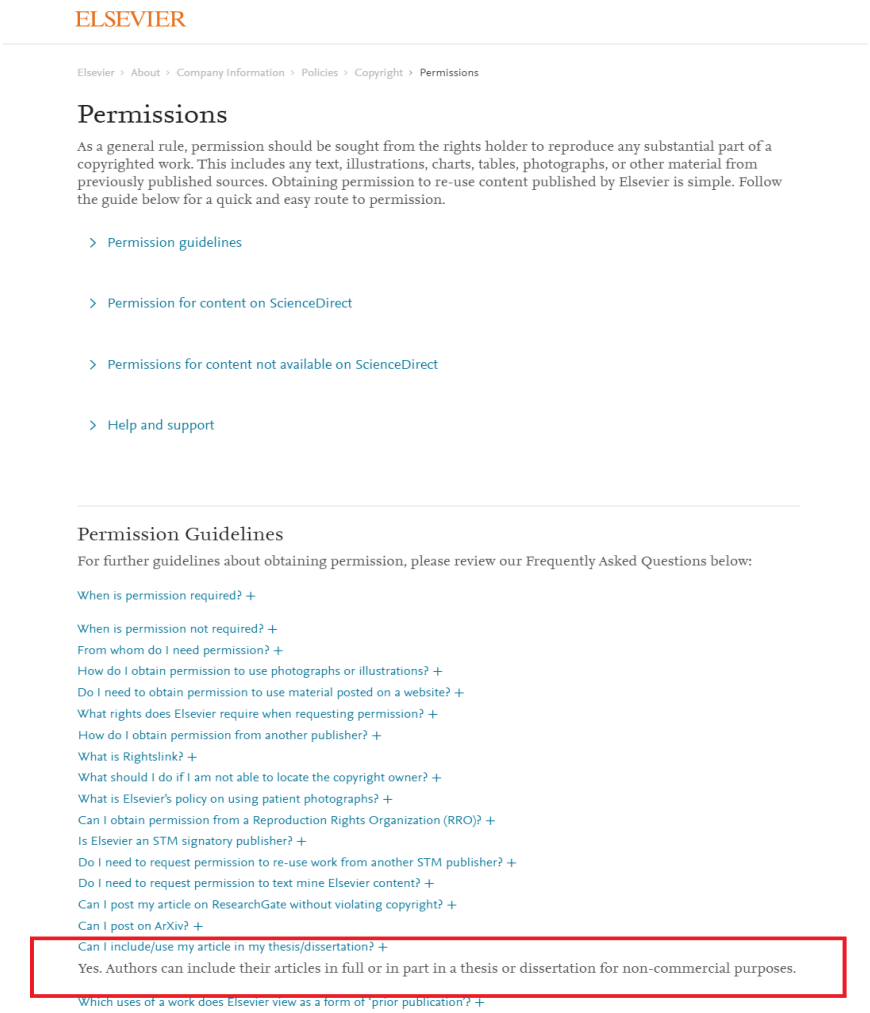
This is just a remark, but the Virtual Network Service Infrastructure Project was over in March 2015.

Best regards,

Masahiro Kiyokawa

Public Relations Department, National Institute of Information and Communications Technology
4-2-1, Nukui-Kitamachi, Koganei, Tokyo, 184-8795, Japan

Below is permission for the use of material in Chapter 3.



ELSEVIER

Elsevier › About › Company Information › Policies › Copyright › Permissions

Permissions

As a general rule, permission should be sought from the rights holder to reproduce any substantial part of a copyrighted work. This includes any text, illustrations, charts, tables, photographs, or other material from previously published sources. Obtaining permission to re-use content published by Elsevier is simple. Follow the guide below for a quick and easy route to permission.

› Permission guidelines

› Permission for content on ScienceDirect

› Permissions for content not available on ScienceDirect

› Help and support

Permission Guidelines

For further guidelines about obtaining permission, please review our Frequently Asked Questions below:

When is permission required? +

When is permission not required? +
From whom do I need permission? +
How do I obtain permission to use photographs or illustrations? +
Do I need to obtain permission to use material posted on a website? +
What rights does Elsevier require when requesting permission? +
How do I obtain permission from another publisher? +
What is Rightslink? +
What should I do if I am not able to locate the copyright owner? +
What is Elsevier's policy on using patient photographs? +
Can I obtain permission from a Reproduction Rights Organization (RRO)? +
Is Elsevier an STM signatory publisher? +
Do I need to request permission to re-use work from another STM publisher? +
Do I need to request permission to text mine Elsevier content? +
Can I post my article on ResearchGate without violating copyright? +
Can I post on ArXiv? +
Can I include/use my article in my thesis/dissertation? +

Yes. Authors can include their articles in full or in part in a thesis or dissertation for non-commercial purposes.

Which uses of a work does Elsevier view as a form of 'prior publication'? +

Below is permission for the use of material in Chapter 4.

Copyright Clearance Center RightsLink®

Home | Create Account | Help

IEEE Requesting permission to reuse content from an IEEE publication

Title: Virtual network advance reservation
Conference Proceedings: Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on
Author: H. Bai
Publisher: IEEE
Date: Oct. 2015
Copyright © 2015, IEEE

LOGIN
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:

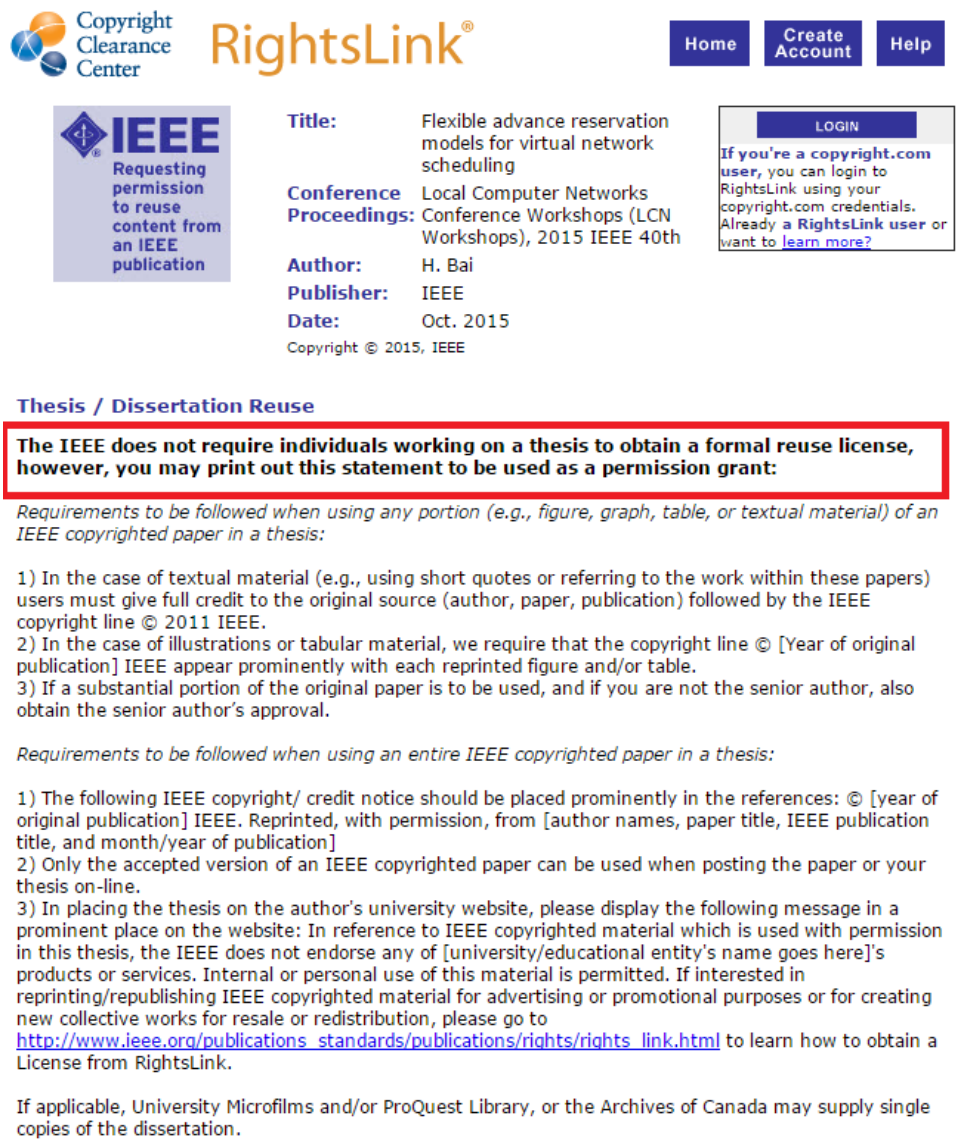
1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that 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.

Below is permission for the use of material in Chapter 5.



Copyright Clearance Center RightsLink®

Home | Create Account | Help

IEEE Requesting permission to reuse content from an IEEE publication

Title: Flexible advance reservation models for virtual network scheduling
Conference Proceedings: Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th
Author: H. Bai
Publisher: IEEE
Date: Oct. 2015
Copyright © 2015, IEEE

LOGIN
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.