Computer Communications 82 (2016) 28–38

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Overlay network scheduling design

H. Bai[a], K. Shaban[b], M. Khodeir[c], F. Gu[d], J. Crichigno[e], S. Khan[f], N. Ghani[a,*]

[a] Electrical Engineering, University of South Florida, Tampa, FL, United States
[b] Computer Science and Engineering, Qatar University, Doha, Qatar
[c] Electrical Engineering, Jordan University of Science and Technology (JUST), Irbid, Jordan
[d] VMware Inc., Palo Alto, CA, United States
[e] Department of Engineering, Northern New Mexico College, Espaola, NM, United States
[f] Electrical & Computer Engineering, North Dakota State University, Fargo, ND, United States

ARTICLE INFO

Article history:
Received 7 July 2015
Revised 31 December 2015
Accepted 18 February 2016
Available online 26 February 2016

Keywords:
Network scheduling
Traffic scheduling
Advance reservation
Overlay networks

ABSTRACT

Advance reservation services are being used by a range of applications to schedule connection bandwidth resources at future time intervals. To date many different algorithms have been developed to support various point-to-point reservation models. However, with expanding data distribution needs there is a need to schedule more complex service types to provide connectivity between multiple sites/locations. In particular, these offerings can help improve network resource utilization and help expand carrier service portfolios. Along these lines, this paper presents a novel, scalable optimization solution to schedule (virtual) overlay networks with fixed end-point nodes. An improved re-routing heuristic scheme is also proposed and analyzed for comparison purposes.

1. Introduction

Network *advance reservation* (AR) allows users to schedule connection bandwidth at future time intervals [1]. These services types are being widely used by a range of applications in scientific computing, work flow process management, and data archival/backup. Overall, AR service models differ from more traditional *immediate reservation* (IR) models, in which incoming requests are provisioned in an immediate manner based upon current resource levels (network state). Namely, AR solutions must incorporate the time dimension in order to account for varying resource levels at future intervals.

In general, connection scheduling is a challenging topic area, and many AR solutions have been studied in recent years [2]. For example, researchers have developed a range of service models for point-to-point connection demands with fixed start/stop times, variable start/stop times, and fixed transfer volumes, see survey in [2]. Since most related scheduling sub-problems here are known to be NP-complete, both optimization and heuristics-based solutions have been proposed. However, as application paradigms expand, there is a growing need to schedule bandwidth interconnectivity between *multiple* end-points, e.g., for applications in cloud backup or mirroring, scientific workflow computing, and event broadcasting (sports, conventions, etc.). Along these lines, several studies have looked at more complex service request models. For example, Entel et al. [3] studied multicast connection scheduling in wavelength-routing optical networks. Meanwhile, Gu et al. [4] introduce the more generalized *virtual overlay network scheduling* (VONS) problem to schedule arbitrary mesh topologies. Consider the details.

Overlay network scheduling entails network resource (i.e., bandwidth) reservation between multiple mesh end-points. This is a very challenging topic since even the batch point-to-point connection scheduling problem is NP-complete [4], i.e., all known solutions are super-polynomial in time with the input size. Hence by extension the global/batch VONS problem is at least of polynomial degree higher complexity since multiple links are involved. In light of this, Gu et al. [4] specified a global integer linear programming (ILP) optimization model to schedule a full batch of incoming overlay demands. However, since the ILP is largely intractable due to high variable counts, the work in [4] proceeds to develop and analyze two basic heuristic schemes. Hence there is a need to further investigate the VONS problem and develop more formal performance bounds. Along these lines, this paper makes two key contributions:

(a) Develops and solves a new VONS ILP optimization model that only treats a subset of time-overlapping requests. Namely, each incoming overlay request is considered in isolation in a dynamic manner in order to lower variable count complexity. In contrast the work in [4] does not solve any ILP model or provide any sort of bounds for the heuristic schemes.

* Corresponding author. Tel.: +1 (813) 974-4772.
E-mail addresses: haobai@mail.usf.edu (H. Bai), khaled.shaban@qu.edu.qa (K. Shaban), makhodeir@just.edu.jo (M. Khodeir), fenggu@unm.edu (F. Gu), jcrichigno@ece.unm.edu (J. Crichigno), samee.khan@ndsu.edu (S. Khan), nghani@usf.edu, ghanin@yahoo.com (N. Ghani).

http://dx.doi.org/10.1016/j.comcom.2016.02.009
0140-3664/

(b) Develops an improved heuristic scheme that incorporates re-routing/re-scheduling strategies to lower request blocking rates and undo the effects of greedy allocation. Since overlay demands are only active at future intervals, prior re-routing of associated link connections will not cause service disruption – a key saliency. In contrast, Gu et al. [4] only present two basic VONS heuristics using minimum hop count and minimum distance routing.

Overall, this work provides a good basis from which to develop further scheduling solutions for cloud-based infrastructure services. Namely, these services require *virtual network* (VN) provisioning over cloud substrates consisting of computing/storage resource pools (datacenter sites) and interconnecting networks (switches, links), seeFig. 1. Now VN requests differ from overlay requests in that VN nodes also have (storage, computation) resource requirements and are not necessarily bound to a fixed network locations [5]. Hence *VN embedding* (VNE) requires both node mapping and connection link routing, adding further dimensionality to the provisioning problem (see Fig. 1). Although a range of VNE schemes have been proposed, only immediate reservation scenarios have been studied [6,7]. Therefore as cloud-based infrastructure services continue to expand, there will be a need to develop VN scheduling schemes as well, and these can leverage from the VONS solutions proposed herein.

This paper is organized as follows. Section 2 first presents a review of existing work in network virtualization design and AR scheduling. Section 3 then details an improved optimization formulation for the topology overlay scheduling problem, followed by a novel heuristic re-routing scheme in Section 4. Detailed performance analysis results are then presented in Section 5 along with conclusions and future directions in Section 6.

2. Background

AR schemes perform resource (bandwidth) reservation for future transfers. One of the key aspects of the AR problem is that user request durations are bounded over a given time interval, either fixed or variable. Accordingly, various service models have been proposed with fixed start/stop times, variable start/stop times, and variable start/fixed file sizes, etc. [1]. Furthermore, a wide range of point-to-point connection AR scheduling algorithms have also been developed to provision these service models, with many focusing on optical networks, see survey in [7]. These solutions include both optimization [8,9] and heuristic strategies [10,11]. The former types assume time-slotted arrivals/departures and pursue various objectives such as minimizing resource utilization, maximizing accepted requests. For example, Andrei et al. [8] formulates a *mixed ILP (MILP)* optimization model to jointly schedule and route lightpath requests with sliding start/stop times. Owing to the complexity of the problem, a Langrangean approximation is developed to maximize the scheduled demands. Meanwhile Zheng et al. [9] presents another optimization model to minimize wavelength resource utilization. However, optimization designs pose notable scalability limitations due to increased variable counts from the discretized time dimension. Hence a range of AR heuristics have also been proposed for on-line request arrivals. For example, Zheng and Mouftah [10,11] present graph-based schemes to schedule fixed start/stop time lightpaths demands (without wavelength conversion). These methods perform a greedy search using fixed-alternate routing methods.

Now unlike immediate reservation, advance reservation requests are not *active* until their future scheduled time intervals. Hence these services can be *re-routed* prior to start without disrupting any active transfers. Along these lines, several AR connection *re-routing* schemes have been proposed to improve blocking rates and accommodate more requests [12,13]. Most of these algorithms use graph-based heuristics to achieve a tradeoff between blocking reduction (carried load) and re-routing attempts (complexity). For example, Wallace et al. [12] present two algorithms to minimize the number of re-routed reservations, and findings show good blocking reduction versus non-re-routing schemes. Meanwhile Xie et al. [13] develops an ILP model to re-optimize scheduled demands and accommodate new requests. This algorithm is triggered if a regular (heuristic) scheduling algorithm fails and uses a *maximum look-ahead time* parameter to bound optimization time windows (complexity). Overall, this ILP scheme gives notably lower blocking as compared to some heuristic methods, but this solution is only feasible for relatively small network sizes (under 10 nodes).

Meanwhile, the overlay network design problem has also been investigated. The overall aim here is to embed dedicated client topologies (sets of virtual links) between select network nodes to provide improved performance, e.g., *quality of service* (QoS), survivability, etc. In general, network overlays can be used to support specific applications such as packetized voice, video streaming, datacenter interconnection, etc. For example, the *service overlay network* (SON) [14] study develops an optimization model to statically allocate bandwidth resources for overlay topologies (sets of connection routes between gateways). This formulation uses queuing models to incorporate oversubscription (in terms of load parameters) and tries to maximize revenues subject to oversubscription penalties. An approximate solution is developed and results are analyzed for different per-unit bandwidth prices and bandwidth levels. Meanwhile, the *resilient overlay network* (RON) project [15] uses overlays to improve network routing convergence and resiliency. Specifically virtual links are provisioned between designated routers running dedicated link-state routing protocols to achieve various objectives, e.g., delay or loss minimization, rapid recovery, etc. Results show notable reduction in failure recovery times versus the legacy *border gateway protocol* (BGP), i.e., seconds versus minutes.

In addition, other efforts have also considered overlay topology provisioning in more specialized networks. For example, Xie et al. [16,17] present several schemes for extending Ethernet *local area network* (LAN) services over SONET/SDH networks using full-mesh or star/hub overlays. These algorithms use shortest-path heuristics to route virtual link connections, and SONET/SDH inverse multiplexing capabilities are also leveraged for partial/tiered recovery via multi-path splitting. Other efforts have proposed *optical* overlay provisioning in fiber-based networks with specialized physical-layer constraints/capabilities, e.g., wavelength continuity, fixed/flexible spectrum, regeneration/amplification, etc. [18]. For example, Pages et al. [19] presents two ILP-based schemes for maximizing carried loads in optical networks with both fixed and flexible spectrum grids. The findings show higher carried loads with the flexible spectrum approach, albeit the associated ILP model is less scalable. Researchers have also studied the more generalized *virtual network* (VN) embedding problem [6,7] where the virtual node locations themselves are variable and subject to placement (Fig. 1).

Now as services expand, there is a need to *schedule* bandwidth connectivity between *multiple* sites, e.g., for applications in cloud/datacenter mirroring and backup, event broadcasting, scientific computing, etc. However, only a handful of studies have looked at scheduling *non-point-to-point* demands/services. For example, Entel et al. [3] studied multicast connection reservation in optical networks. Namely, several optimization and heuristics-based schemes are proposed to setup wavelength connection paths with/without multicast group members. However, the ILP models are only solvable for relatively small networks (under 20 nodes), and findings show improved wavelength conservation for multicast-aware routing schemes. Finally, Gu et al. [4] introduce

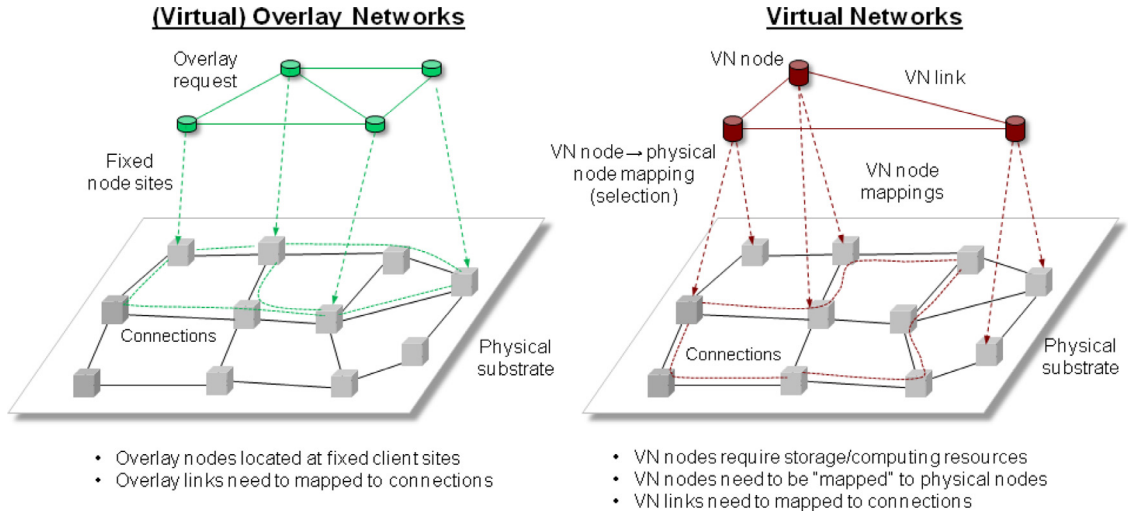


Fig. 1. Overlay networks and virtual networks (VN).

the *virtual overlay network scheduling* (VONS) problem for scheduling arbitrary mesh overlay topologies, i.e., sets of virtual links between designated network nodes. Namely, an ILP formulation is introduced to minimize resource usage across a set of overlay requests. However, due to excessive variable count complexity, only basic greedy heuristics are presented instead to schedule virtual link connections. Specifically, several link weighting schemes are tested in [4], including static (equal) link weights for resource minimization and dynamic bandwidth usage-based weights for load-balancing. Overall the results show reduced blocking rates with the latter weighting schemes.

Nevertheless, although Gu et al. [4] present some good findings, many further issues remain. Foremost, there is a need to develop and solve proper optimization solutions in order to bound overlay scheduling performance. Improved heuristics are also required to improved carried load (revenues) for operators. These concerns are now addressed.

3. Optimization formulation

As noted earlier in [2], the point-to-point connection AR scheduling problem is NP-complete. Hence by extension the overlay scheduling problem is also of at least polynomial-degree higher complexity than NP-complete. In light of this, it is very difficult to find tractable optimization-based solutions, particularly for larger networks. For example, Gu et al. [4] present 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, i.e., as it tries to optimize all demands over the full time horizon (all timeslots).

To address these scalability limitations, a novel *dynamic* optimization scheme is presented for overlay scheduling (VONS problem). This approach adapts the ILP model in [4] and only optimizes a *single* overlay request over the *residual* substrate capacity graph using a shorter time window, i.e., fewer time slots. This contrasts with the optimization framework in [4] which tries to map all requests over the full capacity graph and complete time horizon. Namely, to achieve a tradeoff between performance and complexity, the dynamic ILP only optimizes a subset of the accepted (earlier scheduled) demands that overlap with the incoming request. Since these requests are not yet active, re-optimizing them does not cause any service disruptions. This revised optimization is now detailed.

First consider the requisite notation for substrate and overlay network topologies, as shown in Fig. 2. The substrate network here is 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 capacity, C, 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, $c_e(t)$, which represents its used (available) capacity as a function of time (future intervals). Meanwhile overlay network requests are denoted by the 5-tuple, $r^n = (S^n, L^n, t_s^n, t_e^n, b^n)$, where n is the request index, S^n is the set of node/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 end time. Without loss of generality, it is also assumed that all overlay links $l_{ij}^n \in L^n$ request b^n units of bandwidth, $b^n \leq C$. Note that a request can start at any time after it is received. Hence immediate reservation (IR) requests can be regarded as special cases where the t_s^n values are very close to the actual request arrival times, i.e., with some minor added delays for provisioning and setup.

Using the above notation, an overlay request is specified as a set of directional connections, i.e., nodes in S^n interconnected via a set of "virtual" overlay links given in L^n (see examples in Fig. 2). Carefully note that this formulation only incorporates bandwidth resources, although its can be extended to include datacenter (i.e., computing, storage) resources at the nodes as well. Also, in order to satisfy integer constraints, time is discretized into fixed timeslots 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: $(S^w, L^w, t_s^w, t_e^w, b^w)$
$R_{t_1}^{t_2}$	set of admitted 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 also the time when request r^w received
T_m	maximum look-ahead time allowed for ILP run $T_m = \max_n\{t_e^n \mid r^n \in R_{T_w}^{t_e^w} \cup \{r^w\}\}$
$v_i^n \in S^n$	the ith node in S^n
$L_{i,j}^n$	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 $L_{i,j}^n$ does (not) use link $e \in E$ in timeslot k
$v \to e$	egress node of link $e \in E$ if $v \in V$
$e \to v$	ingress node of link $e \in E$ if $v \in V$

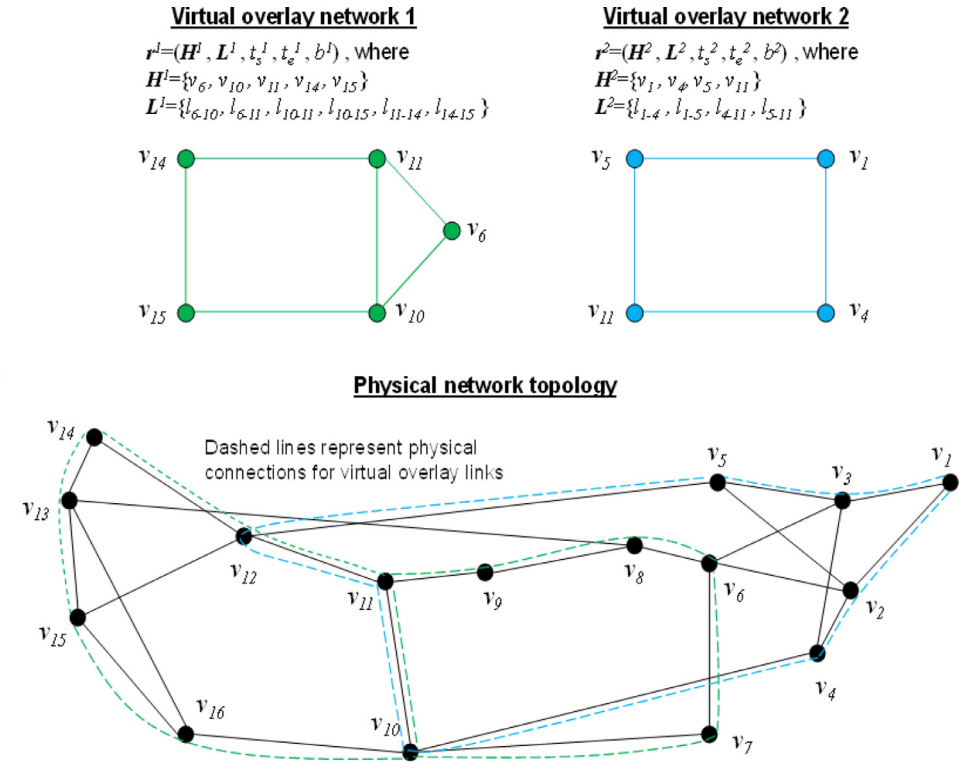


Fig. 2. Virtual overlay network services example.

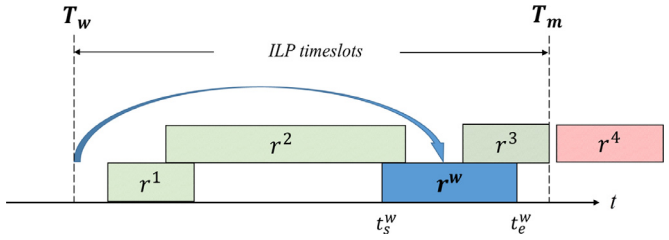


Fig. 3. Example set of admitted inactive reservations in ILP.

Now consider the reduction of the optimization time window, an example of which is shown in Fig. 3. Here, T_w denotes the arrival time of the new request r^w, and this also happens to be when the ILP computation is triggered. Meanwhile, T_m is defined as the *maximum look-ahead time* and is set to the maximum end time across all overlapping inactive reservations in set $R^{t_e^w}_{T_w} \cup \{r^w\}$. 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 $[T_w, T_m]$, i.e., only those requests that start after T_w and end before T_m. Hence for the case in Fig. 3, only requests r^1, r^2, and r^3 are included in the ILP formulation (and not request r^4). Alternatively, if request r^2 had ended before r^w arrived, then it would not be included in the re-optimization. Overall, reducing the number of timeslots in the optimization improves computational scalability, but can also result in a locally-optimal (sub-optimal) solution.

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

$$min \sum_{r^n \in R^{t_e^w}_{T_w} \cup \{r^w\}} \sum_{v_i^n \in S^n} \sum_{e \in E} \sum_{T_w < k \le T_m} b^n p_{i,j}^{n,e,k} \quad (1)$$

subject to the following constraints:

$$\sum_{v_i^n \to e} p_{i,j}^{n,e,k} = 1 \qquad t_s^n \le k \le t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad (2)$$

$$\sum_{e \to v_i^n} p_{i,j}^{n,e,k} = 0 \qquad t_s^n \le k \le t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad (3)$$

$$\sum_{e \to v_j^n} p_{i,j}^{n,e,k} = 1 \qquad t_s^n \le k \le t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad (4)$$

$$\sum_{v_j^n \to e} p_{i,j}^{n,e,k} = 0 \qquad t_s^n \le k \le t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad (5)$$

$$\sum_{e \to v} p_{i,j}^{n,e,k} = \sum_{v \to e} p_{i,j}^{n,e,k} \quad t_s^n \le k \le t_e^n, v \in V \setminus \{v_i^n, v_j^n\}, v_i^n \in S^n, v_j^n \in S^n \quad (6)$$

$$\sum_{r^n \in R^{t_e^w}_{T_w} \cup \{r^w\}} \sum_{v_i^n \in S^n} \sum_{v_j^n \in S^n} b^n p_{i,j}^{n,e,k} \le C \quad e \in E, T_w \le k \le T_m \quad (7)$$

$$p_{i,j}^{n,e,k} = p_{i,j}^{n,e,k+1} \quad r^n \in R^{t_e^w}_{T_w} \cup \{r^w\}, e \in E, t_s^n \le k < t_e^n, v_i^n \in S^n, v_j^n \in S^n \quad (8)$$

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

The pseudocode listing of the dynamic optimization solution is shown in Fig. 4. The scheme first identifies the set of overlapping

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, $G'(V,E)$=$G(V,E)$ and remove all resources for reservations in $R_{T_w}^{t_e^w}$
4: Generate ILP formulation for set $R_{T_w}^{t_e^w} \cup \{r^w\}$
5: **if** ILP solution found
6: Setup success, reserve resources in $G'(V,E)$ and copy $G'(V,E) \rightarrow G(V,E)$
7: **else**
8: Drop request r^w

Fig. 4. ILP formulation framework.

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 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 valid mappings for *all* reservations in the set, then the request is accepted and the residual resource graph $G(V, E)$ is replaced by the temporary working graph $G'(V, E)$. Otherwise the incoming request is dropped.

Overall, the dynamic ILP model greatly reduces run-time complexity versus the "global" ILP formulation in [4]. For example, consider a 10 node mesh topology with 100 overlay requests. If the requests have average holding times of 10 timeslots and average inter-arrival times of 5 timeslots, then approximately 500 timeslots are required for the global optimization scheme in [4]. 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 × 100 total links, 10 × 10 node-to-node topology, and 500 timeslots). Clearly this value poses insurmountable complexity for most modern servers. Now consider the fact that on average, only 2 requests will overlap in time. Hence assuming a maximum look-ahead time of 10 timeslots in the dynamic optimization, the total number of optimization variables drops to about 9000, i.e., 3 × 2 × 10 × 10 × 15 = 9000 (i.e., 3 × 2 total links, 10 × 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.

4. Re-routing heuristic

Some novel overlay scheduling heuristics are also presented to improve upon the greedy schemes in [4]. The overall goal here is to use *re-routing* techniques to re-map *accepted* overlay requests in order to make room for a new demands, i.e., much in the 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 [13]. Consider the details.

A high-level view of the proposed re-routing framework is illustrated in Fig. 5 along with a more detailed psuedocode description in Fig. 6. Overall, this heuristic comprises of two stages. The first stage (Stage 1) attempts a regular 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) connections 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. The two stages are now detailed further.

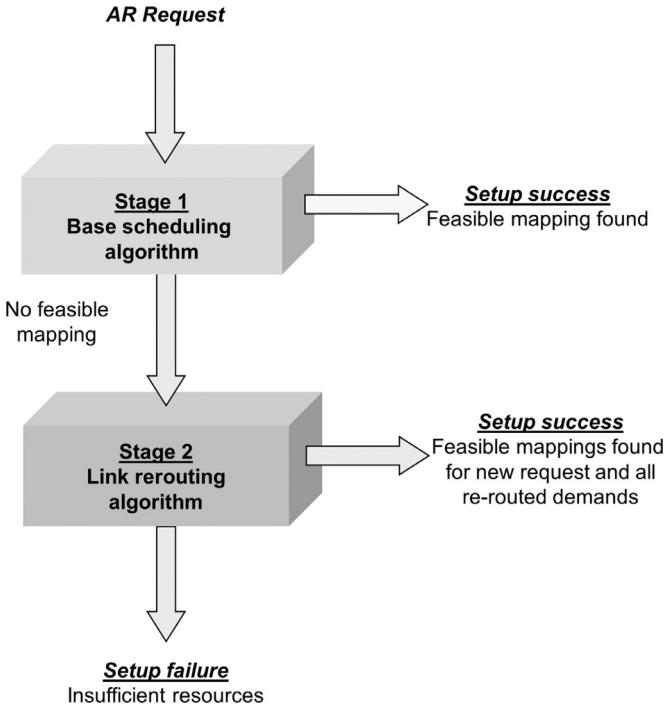


Fig. 5. Two-stage overlay network scheduling re-routing strategy.

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 **then**
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 **then**
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 **then**
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 $G^*(V,E) \rightarrow G(V,E)$

Fig. 6. Overlay demand re-routing heuristic.

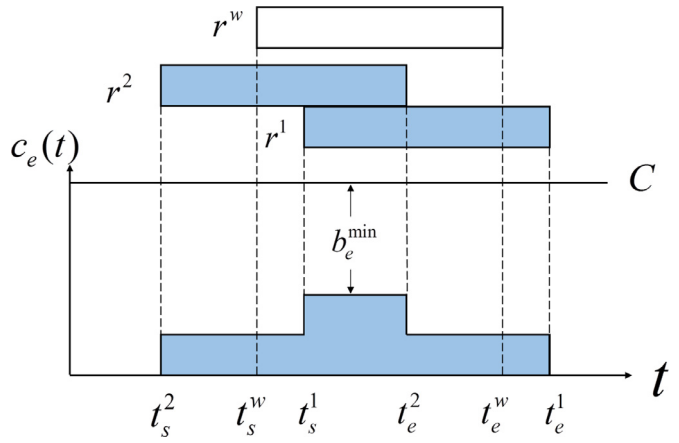


Fig. 7. Sample link capacity function $c_e(t)$, two reservations (r^1, r^2) when r^w arrives.

4.1. Baseline scheduling (Stage 1)

The initial stage simply runs a baseline heuristic to setup an incoming overlay request (steps 4 and 5, Fig. 6). Now technically any overlay scheduling heuristic can be re-used here. However, for the purposes herein, the load-balancing *min-distance* strategy is chosen as it gives the best performance out of all the schemes in [4], i.e., reduced blocking (higher carried loads) 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 all virtual link connections using Dijkstra's algorithm (sequential manner). Namely, the weight for link e is computed as inversely-proportional to its lowest residual capacity in the request interval:

$$w = 1/(\chi + \epsilon) \tag{9}$$

where

$$\chi = \min_{t \in [t_s^w, t_e^w]} c_e(t) \tag{10}$$

where $c_e(t)$ is the earlier-defined link capacity function, and ϵ is a small value chosen to avoid division errors. Note that all the above computations are done using a *temporary* copy of the residual bandwidth graph, termed $G^*(V, E)$, see steps 1 and 2, Fig. 6. This copy is used to track/store all accepted connection reservations (virtual links) in the request 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 and 23, Fig. 6. To further improve setup success, Dijkstra's algorithm only considers substrate links in $G^*(V, E)$ with sufficient capacity to provision the requested bandwidth for r^w, b^w (step 4, Fig. 5). Specifically, based upon the $c_e(t)$, the *bottleneck* link capacity for link e in the interval $[t_s^w, t_e^w]$ is defined as follows:

$$b_e^{min} = \min c_e(t), t_s^w \le t \le t_e^w \tag{11}$$

Based upon the above, 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. An example of bottleneck link capacity selection is also shown in Fig. 7.

4.2. Scheduling re-routing (Stage 2)

The re-routing stage is triggered if a virtual link connection attempt in Stage 1 is unsuccessful, i.e., steps 9–21, Fig. 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 divergent objective versus blocking reduction. Hence several candidate path selection approaches are proposed here including:

- *Minimum Hop Re-Routing (MHR):* This scheme selects the candidate path as 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 the overlay link request's end-point nodes are computed, and the path giving the fewest number of (virtual link connection) re-routings is chosen. Specifically, this is done by ordering all connection reservations on a link by decreasing bandwidth size and then counting the minimum number needed to meet the requested capacity, b^w.
- *Threshold Re-Route (THR):* This scheme minimizes the amount of perturbation by selecting a path that already has a fraction ρ ($0 \le \rho < 1$) of the requested overlay link capacity in the 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 $b_e^{min} \ge \rho b^w$ in $[t_s^w, t_e^w]$ is kept. This approach is similar to the connection-level AR re-routing scheme in [13].

Once a candidate path has been chosen, a subset of the existing (overlay link) connection reservations on its links are selected for re-routing, i.e., termed as the *re-routing connection set*. Clearly these reservations can span across multiple overlay demands. Hence to minimize 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 overlapping durations) to the re-routing connection set to free up capacity. Specifically, upon 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, Fig. 6).

Finally, the heuristic tries to sequentially re-schedule all reservations in the re-routing connection set (steps 17–21, Fig. 6). This step basically re-runs the regular Stage 1 setup algorithm (from Section 4.1) 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. Note that additional improvements can also be devised here. For example, the reservations in the re-routing connection set can be sorted based upon increasing or decreasing capacity sizes. This will have a potential impact on the future requests. However, such modifications are left for future study.

4.3. Computational complexity

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

Table 1
Heuristic complexity comparison.

	Heuristic	MHR	MNR	THR
Stage 1	Baseline scheduling	$O(\|E\| + \|V\| \log \|V\|)$		
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	$O(\|E\|\|V\|^2 \log \|V\|)$	$O(1)$	$O(\|E\|\|V\|^2 \log \|V\|)$
	Rerouting computation	$O(\|V\|^2\|E\| + \|V\|^3 \log \|V\|)$		

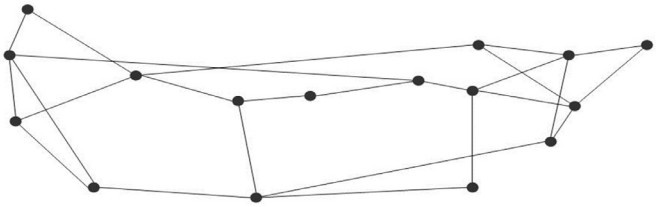
Next consider the re-routing stage, Stage 2. The first step focuses on candidate path selection, and here both the MHR and THR schemes use Dijkstra's shortest path algorithm to compute a candidate path, i.e., $O(|E| + |V| \log(|V|))$ complexity, akin to Stage 1. However the MNR strategy is more involved 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-shortest paths computation can be solved in time $O(|E| + |V| \log(|V|) + k)$ [21]. 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 [13]. Sorting these reservations in decreasing order of bandwidth size adds an additional $O(|E||V|^2 \log |V|)$ complexity, yielding a total bound of $O(|E| + |V| \log(|V|) + |E||V|^2 \log |V|)$ for the MNR scheme.

Finally, to compute the re-routing connection set, both MHR and THR schemes must sort the previously-scheduled reservations in decreasing order, i.e., $O(|E||V|^2 \log |V|)$ 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 $O(|V|^2|E| + |V|^3 \log |V|)$ for all three heuristics. 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 also summarized in Table 1.

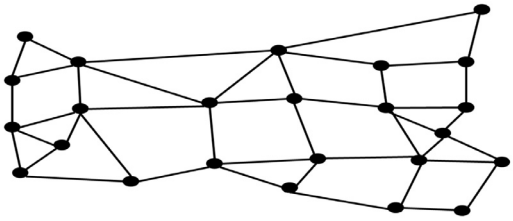
5. Performance analysis

The performance of the overlay scheduling schemes is now analyzed. Namely, advanced discrete event simulation models are developed in the *OPNET Modeler*™ toolkit to generate and process overlay requests/demands. Meanwhile, the dynamic optimization model (Section 3) is also solved by generating external file-driven calls to the *CPLEX* optimization solver tool. Furthermore, 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 Fig. 8. All nodes are assumed to be IP *multiprotocol label switching* (MPLS) routers with 10 Gbps links and advanced bandwidth provisioning capabilities.

Meanwhile, the Inet topology generator is used to generate random overlay requests (topologies) 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 1000 Mbps. Furthermore, incoming requests have exponentially-distributed holding and inter-arrival times with means μ and λ, respectively (rounded to nearest timeslot value). In particular, the mean holding time is set to 10 timeslots, 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 [22]. However, in order 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:



(a) NSFNET backbone topology (16 nodes, 25 links)



(b) 24-node test topology (24 nodes, 43 links)

Fig. 8. Test network topologies.

$$\text{Modified Erlang load} = \frac{1}{3}\sum_{n=4}^{6}(n-1) \times \mu/\lambda \tag{12}$$

for overlay topologies ranging from $n = 4$–6 nodes. Consider the findings now.

First, sensitivity tests are 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., $\rho = 0.1$, 0.5, and 0.9. Overall blocking results (not shown) indicate very minimal variations between the different ρ values, i.e., blocking rates for different ρ values within 1% of each other at any given input load. Meanwhile the average overlay connection path lengths are shown in Fig. 9 and indicate slightly higher utilization with smaller ρ values (particularly at higher loads). 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 is plotted in Fig. 10, and these results show notably higher overheads with smaller ρ values, i.e., due to reduced re-route triggering thresholds. Finally, the re-routing success rates are also plotted in Fig. 11 and indicate improved setup performance with larger ρ values. This is expected since larger ρ values result in fewer, more successful re-routing attempts. Based upon these findings, a median value of $\rho = 0.5$ is chosen to maintain a balance between re-routing overheads and blocking reduction.

Next, detailed tests are done to compare the performances of the dynamic optimization and heuristic schemes (MHR, MNR, THR). The baseline scheduling heuristic in Section 4.1 is also tested here in order to provide a "non-re-routing" reference. As mentioned earlier, this baseline is the same as the best-performing min-distance heuristic in [4]. Furthermore, since the ILP optimization approach has notably higher run-time complexity, all tests

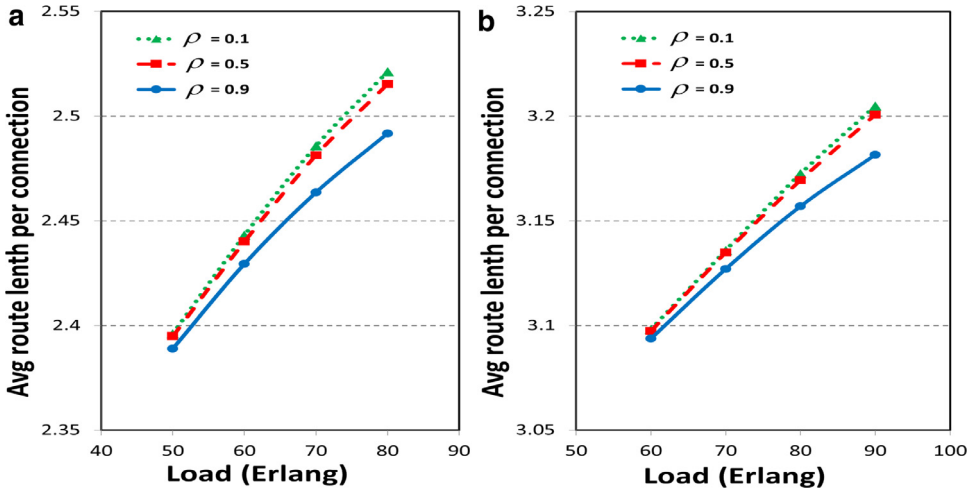


Fig. 9. Average overlay link connection length (a) NSFNET and (b) 24-node network.

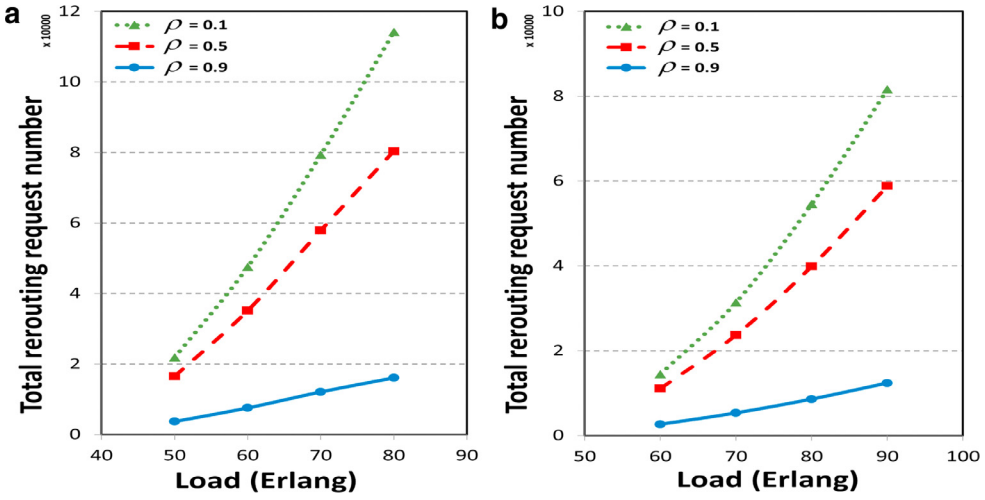


Fig. 10. Total number of re-routed requests (a) NSFNET and (b) 24-node network.

comparing its performance are done using 1000 random overlay requests. Otherwise, all runs are averaged over 100,000 random requests.

Foremost, blocking results are shown in Fig. 12 and confirm that the dynamic ILP scheme gives the highest setup success rates. For example, at higher input loads this scheme gives almost 25% lower blocking versus the baseline min-distance heuristic and close to 15% lower blocking versus the re-routing heuristics (for both network topologies). Meanwhile, the separation between the re-routing and non-rerouting heuristics is generally lower, but still notable, i.e., averaging around 10% depending upon input load. However, only minor differences are seen between the various re-routing (candidate path selection) schemes, with the respective blocking ratios falling within 2% of each other. The corresponding run times for the dynamic ILP scheme are also shown in Table 2 (for 1000 requests processed on a 3.7 GHz quad-core processor with 8 gigabytes of memory). Although these values increase with load, they mostly fall within the 10s of seconds rage, indicating good applicability in on-line settings.

Next, the average overlay connection path lengths are plotted in Fig. 13. As expected, the dynamic ILP scheme gives the lowest resource utilization, followed by the non-re-routing heuristic. In particular, the optimization approach gives about 5–10% lower hop count values 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 as re-routing procedures force longer detour routes to accomodate new demands, i.e., tradeoff between blocking and resource efficiency.

Table 2
Average run-time comparison.

Schemes	Heuristic no re-routing	Heuristic re-routing	ILP optimization
Single request	<1 s	<1 s	5–10 s
All requests	15–20 s	20–30 s	100–180 min

The individual re-routing heuristics are also compared here. First, Fig. 14 plots the total number of re-rerouted link connections for each scheme in order to gauge overall run-time durations. These findings indicate that the THR (MHR) algorithm gives the lowest (highest) amount of re-routing. In light of the relatively close blocking performances across all heuristics (Fig. 12), it can be concluded that the THR scheme gives the most compet-

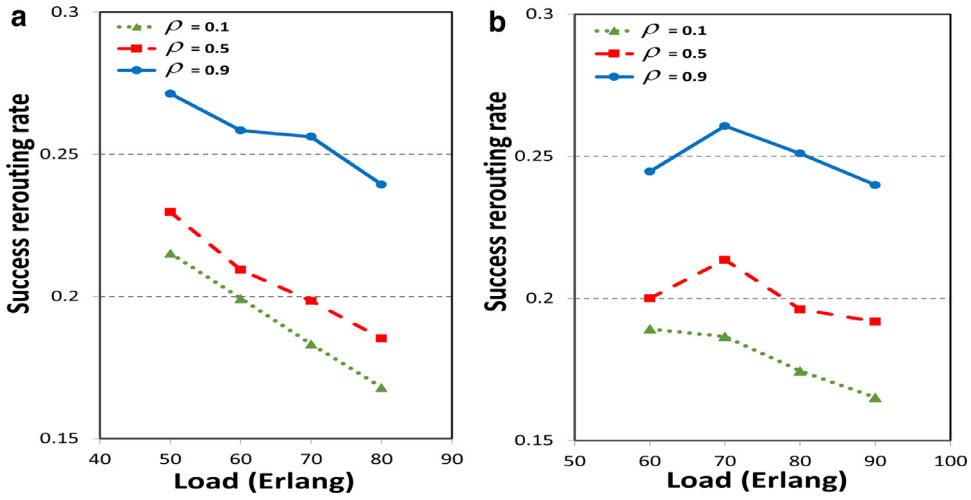


Fig. 11. Re-routing success rate (a) NSFNET and (b) 24-node network.

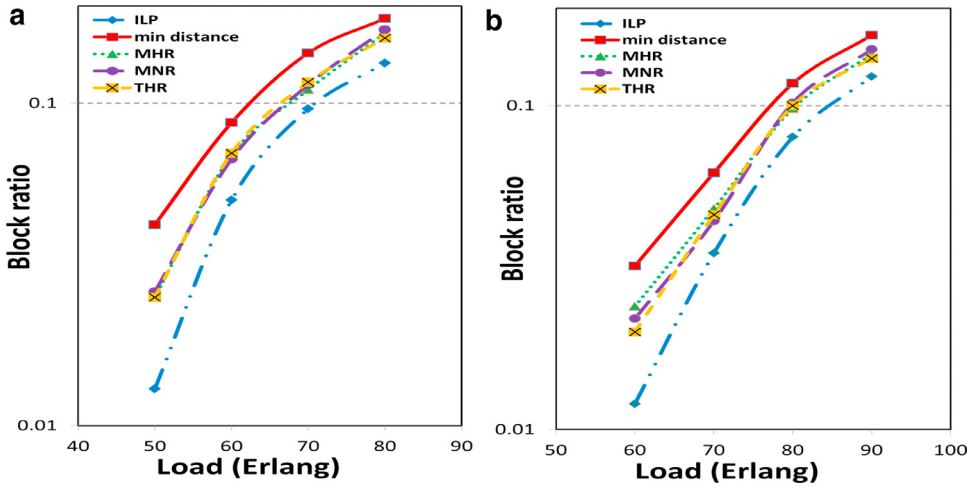


Fig. 12. Request blocking rate versus load (a) NSFNET and (b) 24-node network.

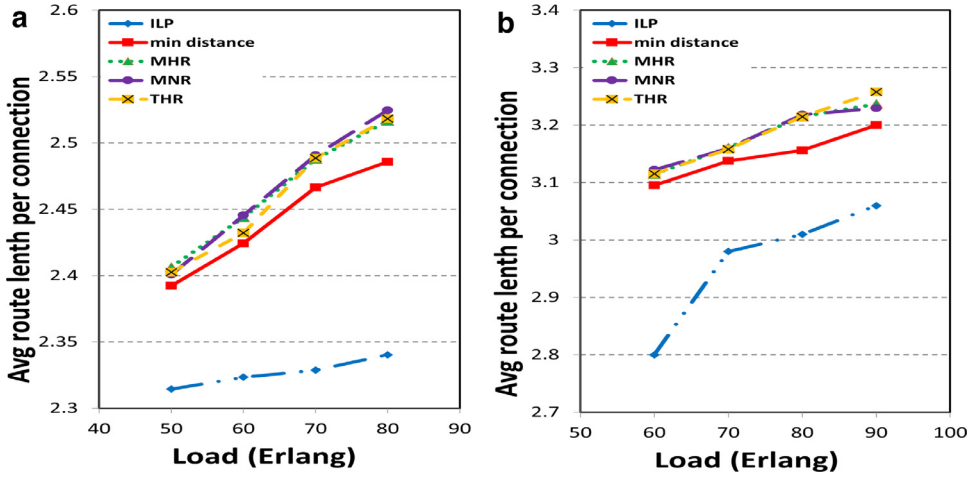


Fig. 13. Average overlay link connection length (a) NSFNET and (b) 24-node network.

itive re-routing performance. Meanwhile, re-routing success rates are also plotted in Fig. 15 to gauge the efficiency of the re-routing schemes. These findings show that re-routing is much more effective at lower load regimes, as there are fewer contending users and more available network resources (resulting in fewer re-routings). Despite some fluctuations here, 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 schemes, i.e., even though all variants yield very close blocking and resource utilization results (see Figs. 12 and 13).

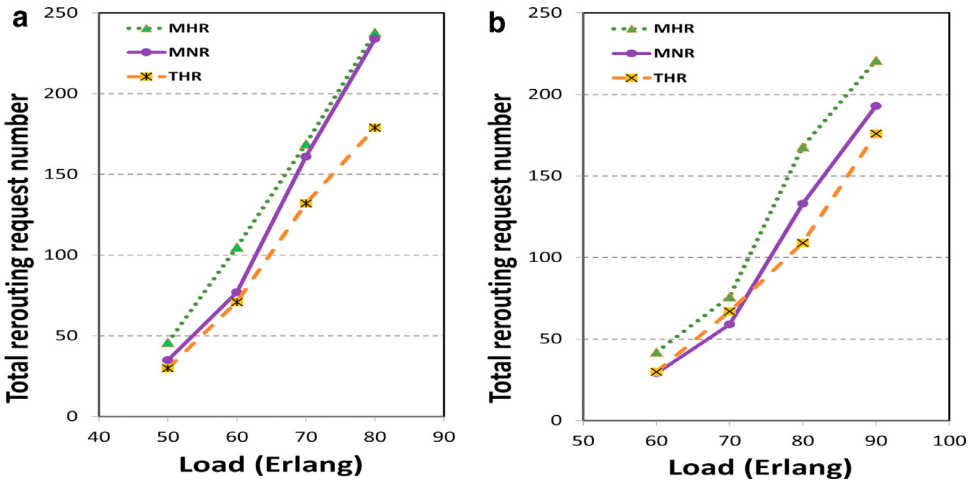


Fig. 14. Total number of re-routed requests (a) NSFNET and (b) 24-node network.

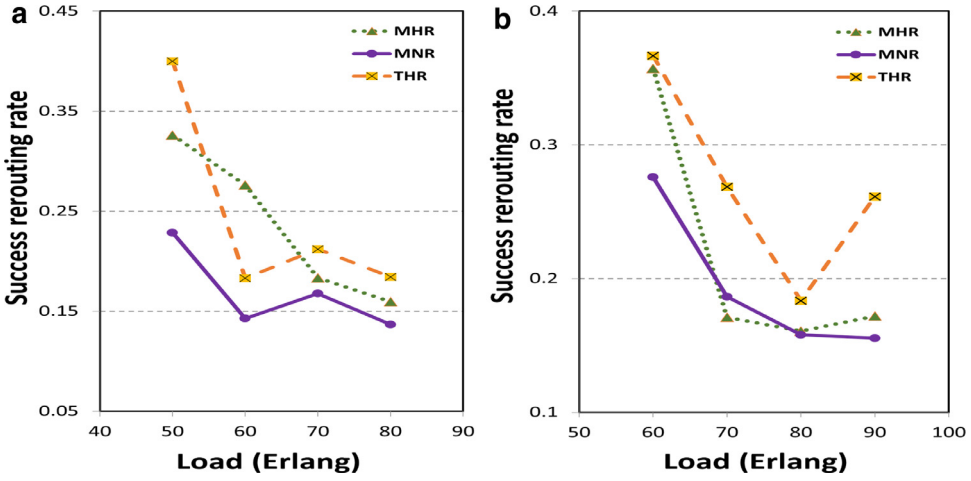


Fig. 15. Re-routing success rate (a) NSFNET and (b) 24-node network.

6. Conclusions and future work

Network advance reservation (traffic scheduling) is an important area, and a range of studies have looked at reserving point-to-point connection demands. However, as user applications continue to evolve, there is a further need to schedule more complex overlay topologies at future intervals. This paper addresses this concern and presents a novel dynamic optimization scheme for overlay scheduling. In addition, some advanced heuristic solutions are also developed by using connection re-routing strategies. Overall findings confirm that the optimization approach gives the lowest blocking and resource utilization. However, the proposed re-routing heuristics also provide very good gains over a baseline greedy heuristic scheme. In particular, strategies which use thresholding to minimize the amount of re-routed capacity give the highest request acceptance rates. Future efforts will look at extending these schemes to address the more general virtual network scheduling problem for cloud-based settings.

Acknowledgment

This work was made possible by the NPRP 5-137-2-045 Grant from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

[1] R. Guerin, A. Orda, Networks with advance reservations: the routing perspective, in: Proceedings of the IEEE International Conference on Computer Communications, IEEE INFOCOM, vol. 1, 2000, pp. 118–127, doi:10.1109/INFCOM.2000.832180.
[2] N. Charbonneaum, V. Vokkarane, A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks, IEEE Commun. Surv. Tutor. 14 (2011) 1037–1064, doi:10.1109/SURV.2011.111411.00054.
[3] T. Entel, et al., Scheduled multicast overlay for bandwidth-intensive applications, Opt. Netw. Des. Model. (2012) 1–6, doi:10.1109/ONDM.2012.6210278.
[4] F. Gu, et al., Virtual overlay network scheduling, IEEE Commun. Lett. 15 (2011) 893–895, doi:10.1109/LCOMM.2011.060811.110819.
[5] Y. Zhu, M. Ammar, Algorithms for assigning substrate network resources to virtual network components, in: Proceedings of the IEEE International Conference on Computer Communications, IEEE INFOCOM, 2016, pp. 1–12, doi:10.1109/INFOCOM.2006.322.
[6] N. Mosharaf, et al., A survey of network virtualization, Comput. Netw. 54 (2010) 862–876, doi:10.1016/j.comnet.2009.10.017.
[7] A. Fischer, et al., Virtual network embedding: a survey, IEEE Commun. Surv. Tutor. 15 (2013) 1888–1906, doi:10.1109/SURV.2013.013013.00155.
[8] D. Andrei, et al., Integrated provisioning of sliding scheduled services over WDM optical networks, IEEE/OSA J. Opt. Commun. Netw. 1 (2009) A94–A105, doi:10.1364/JOCN.1.000A94.
[9] J. Zheng, et al., Toward automated provisioning of advance reservation service in next-generation optical internet, IEEE Commun. Mag. 44 (2007) 68–74, doi:10.1109/MCOM.2006.273102.
[10] J. Zheng, H.T. Mouftah, Supporting advance reservations in wavelength-routed WDM networks, in: Proceedings of the Tenth International Conference on Computer Communications and Networks, 2001, pp. 594–597, doi:10.1109/ICC.2002.997338.

[11] J. Zheng, H.T. Mouftah, Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks, in: Proceedings of the IEEE International Conference on Communications, ICC, vol. 5, 2002, pp. 2722–2726.
[12] T. Wallace, et al., Connection management algorithm for advance lightpath reservation in WDM networks, in: Proceedings of the International Conference on Broadband Networks, IEEE BROADNETS, 2007, pp. 837–844, doi:10.1109/BROADNETS.2007.4550520.
[13] C. Xie, et al., Rerouting in advance reservation networks, Comput. Commun. 35 (2012) 1411–1421, doi:10.1016/j.comcom.2012.02.011.
[14] Z. Duan, et al., Service overlay networks: SLAS, QOS, and bandwidth provisioning, IEEE/ACM Trans. Netw. 11 (2004) 870–883, doi:10.1109/TNET.2003.820436.
[15] D. Anderson, et al., Resilient overlay networks, in: Proceedings of ACM Symposium on Operating Systems Principles, 35, 2001, pp. 131–145, doi:10.1145/502034.502048.
[16] C. Xie, et al., Traffic engineering for Ethernet over SONET/SDH: advances and frontiers, IEEE Netw. 23 (2009a) 18–25, doi:10.1109/MNET.2009.4939259.
[17] C. Xie, et al., Multi-point ethernet over next-generation SONET/SDH, in: Proceedings of the IEEE International Conference on Communications, ICC, 2009b, pp. 1–6, doi:10.1109/ICC.2009.5199207.
[18] R. Nejabati, et al., Optical network virtualization, in: Proceedings of the International Conference on Optical Network Design and Modeling (ONDM), 2011, pp. 1–5.
[19] A. Pages, et al., Optimal allocation of virtual optical networks for the future internet, in: Proceedings of the International Conference on Optical Network Design and Modeling (ONDM), 2012, pp. 1–6, doi:10.1109/ONDM.2012.6210209.
[20] T. Cormen, et al., Introduction to Algorithms, 3rd edition, The MIT Press, 2009.
[21] D. Eppstein, Finding the k shortest paths, SIAM J. Comput. 28 (2) (1998) 652–673.
[22] R. Wolff, Stochastic Modeling and the Theory of Queues, Prentice Hall, 1989.