Optical Switching and Networking 9 (2012) 179–190

Contents lists available at SciVerse ScienceDirect

Optical Switching and Networking

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

Dynamic circuit provisioning in all-optical WDM networks using lightpath switching[☆]

Neal Charbonneau[a], Arush Gadkar[b], Bharath H. Ramaprasad[b], Vinod M. Vokkarane[b,*]

[a] The MITRE Corporation, Bedford, MA, United States

[b] Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA, United States

ARTICLE INFO

Article history:
Available online 13 December 2011

Keywords:
Holding-time-aware (HTA) lightpaths
Lightpath switching
Anycast
RWA
WDM

ABSTRACT

In this paper we investigate the problem of provisioning holding-time-aware (HTA) dynamic circuits in all-optical wavelength division multiplexed (WDM) networks. We employ a technique called *lightpath switching* (LPS) wherein the data transmission may begin on one lightpath and switch to a different lightpath at a later time. Lightpath switches are transparent to the user and are managed by the network. Allowing LPS creates a number of segments that can use independent lightpaths. We first compare the performance of traditional routing and wavelength (RWA) assignment to routing and wavelength assignment with LPS. We show that LPS can significantly reduce blocking compared to traditional RWA. We then address the problem of routing dynamic *anycast* HTA dynamic circuits. We propose two heuristics to solve the anycast RWA problem: anycast with continuous segment (ACS) and anycast with lightpath switching (ALPS). In ALPS we exercise LPS, and provision a connection request by searching for the best candidate destination node is such a way that the network resources are utilized efficiently. In ACS we do not allow a connection request to switch lightpaths. The lightpaths to each candidate destination node of a request are computed using traditional RWA algorithms. We first compare the performance of ACS to ALPS and observe that ALPS achieves better blocking than ACS. Furthermore, we also compare the performance of these two anycast RWA algorithms to the traditional unicast RWA algorithm. We show that the anycast RWA algorithms presented here significantly outperform the traditional unicast RWA algorithms.

1. Introduction

Optical wavelength routed networks are a promising candidate for future wide-area backbone networks as well as scientific Grid networks. In WDM networks, each fiber is partitioned into a number of wavelengths, each of which is capable of transmitting data simultaneously [1]. In order to transmit data over the network, a dedicated circuit is established when a user submits a connection request. When a connection request arrives at the network, the request must be routed over the physical topology and also assigned a wavelength. This is known as the *routing and wavelength assignment* (RWA) problem [2]. The combination of a route and wavelength is known as a *lightpath* [3]. The RWA problem is NP-complete, hence heuristics are typically used to solve the problem. As requests are accepted into the network, no two requests can use the same wavelength on the same link. As more requests arrive over time, new lightpaths must be allocated as long as there are enough wavelengths to establish them.

[☆] This paper is an extended version of a paper presented at the Fourth IEEE Symposium on Advanced Networks and Telecom Systems (ANTS) 2010 in Mumbai, India, in December 2010 [6] and a paper to be presented in 12th IEEE International Conference on High Performance Switching and Routing (HPSR) to be held in Cartagena, Spain, in July 2011 [19]. This work has been partially supported by the National Science Foundation (NSF) SOON project under grant CNS-0626798 and Department of Energy (DOE) COMMON project under grant DE-SC0004909.

[*] Corresponding author. Tel.: +1 5089106692; fax: +1 508 999 9144.
E-mail addresses: charbonneau@mitre.org (N. Charbonneau), agadkar@umassd.edu (A. Gadkar), bharath.hr@umassd.edu (B.H. Ramaprasad), vvokkarane@ieee.org (V.M. Vokkarane).

1573-4277/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.osn.2011.11.004

If an arriving request cannot find a lightpath, the request is rejected and it is said to be blocked. In an all-optical WDM system, once a path is set up, the signal is transmitted all-optically through the network.

A common traffic model for WDM networks is the dynamic unicast traffic model. Requests are assumed to arrive sequentially, according to a stochastic process, and have finite holding times. The goal is to minimize request blocking, where a user's request is denied due to lack of resources. We can consider two types of dynamic models. One is dynamic with unknown duration, where each request uses network resources for an unspecified amount of time, and the other is dynamic with known duration where requests specify a holding time when they arrive. This is also known as *holding-time-aware* (HTA) traffic [4]. There are classes of applications that are able to specify holding times, such as video distribution and large file transfers. This extra information allows the network to better optimize its resources and increase efficiency.

Grid applications, scientific experiments and large-scale e-Science projects generate tremendous amounts of data (in the order of Terabytes per second) and require transmission of this data set to a collaborating destination for purposes of storage, computation and processing. Such requirements call for a network with dynamic circuits which can provide efficient transmission of data instantaneously. While supporting such applications, a major challenge to be addressed is how to find such computing or storage resources distributed throughout the network. Anycast [5], is a communication paradigm that may provide an intelligent selection of a destination node for distributed applications. *Anycasting* is defined as the communication paradigm in which the user has the ability to choose a single destination from a group of candidate destinations, unlike deciding it a-priori as in the traditional *unicast* mode of communication. This provides the anycast communication paradigm more flexibility and results in the network resources being utilized efficiently.

It is already established that HTA can improve the performance of traditional dynamic traffic with unknown durations [4]. In this paper, we first consider HTA unicast traffic and show the benefits of employing *lightpath switching* (LPS) [6]. With LPS, a series of lightpath switches occur during the request's duration. For example, a request may use some lightpath x from time t_1 to t_5, then switch to a different lightpath y from time t_5 to t_8. We note that this is not the same as multihop routing. We still use single-hop routing, but the physical lightpath connecting the source to destination changes temporally. Furthermore, we consider *anycast* HTA traffic, with and without LPS and show that the anycast communication paradigm significantly outperforms the traditional unicast means of communication. To provision the anycast connection request, we try to find the requested number of time slots from the source node of the request, to any one member of the candidate destination set using LPS. We block the anycast request only if the requested number of time slots cannot be allocated on any path, and on any available wavelength, to any candidate destination member (node) of the request. To this end, we compare the performance of the RWA for anycast with lightpath switching (ALPS) to the performance of the RWA for anycast with continuous segment (ACS), wherein we restrict the data transmission to a single lightpath over a single wavelength (i.e., no lightpath switching). Our results show that the ALPS performs better than the ACS, and both anycast heuristics perform significantly better than the traditional unicast RWA algorithms.

The remainder of this paper is organized as follows: we discuss the related work in Section 2 and formally define the problems considered in this paper in Section 3. In Section 4, we present our heuristics to solve the unicast and anycast HTA traffic. In Section 5, we present the results of our performance evaluations and finally conclude the paper in Section 6.

2. Related work

In this section we provide a brief overview of the related work. There is a lot of work in the literature on routing and wavelength assignment of unicast traffic demands over wavelength-routed optical networks. We direct the reader to [7,8] and the references therein. In [5] the authors discuss several policies, based on which an anycast destination is selected in IP-over-optical networks. There are a number of papers that address HTA traffic, among others see [9]. Advance reservation, or scheduled demands, is related to HTA traffic. With advance reservation, in addition to specifying a holding time, requests book-ahead resources i.e., they reserve network resources in-advance for use at a later time [10,11]. Variations of advanced reservations such as book ahead with flexibility is discussed in [12].

In [6], we have proposed the usage of lightpath switching to route unicast traffic demands in optical networks. In [12] the authors discuss in detail the multiple constraints involved in Grid computing for immediate and advanced reservations. They also discuss deadline driven requests which is a form of HTA immediate reservation. As for non-continuous advance reservation, the authors in [13] consider a static traffic model wherein all the connection requests are known *a priori* and each request is divided into smaller segments that can use different lightpaths. While [14] develops several algorithms to solve the dynamic non-continuous routing, wavelength, and segment assignment (RWSA) problem. In [15], the authors propose a flexible reservation mechanism that can be segmented into smaller sets of reservations, but they do not consider the RWA problem. They approach the problem from a bandwidth scheduling perspective with the assumption that the lightpaths are already established. To the best of our knowledge, we are the first to propose LPS and anycast RWSA for all-optical wavelength-routed WDM networks.

3. Problem definition

In this section we formally define the problem and discuss the network assumptions used in this paper. We consider a dynamic traffic model wherein users' requests arrive according to some stochastic process. Given a network $G = (V, E, W, H)$, (V is the set of nodes, E is the set of edges, and W is the number of wavelengths per fiber), we consider a time-slotted network with fixed-size time slots. We define horizon (H) as the number of future time slots for which the state information is maintained in the scheduler. The horizon limits how large the holding times or duration of the requests can be. A centralized scheduler maintains the state information, which is updated for every new request. The state information consists of which timeslots are in use on all the wavelengths and on all edges across the network. It can be thought of as a three-dimensional list $U[E, W, H]$. A user's unicast request, R, can be defined as a three-tuple, (s, d, τ), where $s \in V$ is the source node, $d \in V$ is the destination, and τ is the duration in timeslots. Upon arrival of a dynamic circuit request, the scheduler must allocate resources to the request. The scheduler will then return a vector of *segments*, S, called the *schedule*. We define a *segment* as a lightpath used to transfer data between a specified start and end time. A segment can be defined as a four-tuple, (t, d, P, W), where t is the start time, d is the duration, P is the path, and W is the wavelength. We assume that t refers to a specific timeslot and d is specified in number of timeslots. The start time is inclusive, so the segment transmits data from $[t, t + d - 1]$. Each segment follows the wavelength continuity constraint on all links. Each new segment constitutes a lightpath-switch. Instead of generating a single route and wavelength for a given request as in traditional RWA problems, our heuristics can generate a schedule of one or more segments. We will call this *unicast routing, wavelength and segment assignment* (U-RWSA).

U-RWSA: Given a network, $G = (V, E, W, H)$, its current state, $U[E, W, H]$, and an incoming request, $R = (s, d, \tau)$, we must return a *schedule*, $S = \{(t_i, d_i, P_i, W_i)\}$ if the request can be accommodated, or *BLOCKED* otherwise. The segments should be selected in a way that they reduce blocking of future requests.

We have the following constraints for the schedule, S:

$$1 \le |S| \le \tau \tag{1}$$

$$t_1 = t_{now} \tag{2}$$

$$\sum d_i = \tau \tag{3}$$

$$t_i + d_i = t_{i+1}. \tag{4}$$

Assume the schedule has n elements. (1) specifies there must be at least one segment and there can be at most τ segments. (2) states that the first segment must start when the request arrives. (3) states that the summation of the segment durations is equal to the request's duration. (4) states that each segment must start when the previous one ends.

A user's anycast request R, can be defined as a three tuple, $(s, \{d_c\}, \tau)$ where $s \in V$ is the source node and d_c ($\{d_c\} \subset V - s$) is the set of candidate anycast destinations with $|d_c| = M$, where M is the cardinality of the anycast destination set and τ is the duration of the request in timeslots. Upon arrival of a dynamic anycast request, our heuristic must generate a schedule of one or more segments across k-shortest paths while selecting the best anycast candidate destination from an ordered destination set. We will call this *anycast routing, wavelength, and segment assignment* (A-RWSA).

A-RWSA: Given a network, $G = (V, E, W, H)$, its current state ($U[E, W, H]$), and an incoming dynamic anycast request $R = (s, \{d_c\}, \tau)$, we must return a *schedule*, $S = \{(t_i, d_i, P_i, W_i)\}$ if the request is accommodated or block the request due to unavailability of network resources. Note that the anycast request is blocked only if the requested number of time slots cannot be allocated on any path, and on any available wavelength, to any candidate destination member (node) of the request. Note also, that the segments should be selected in such a way that they reduce the blocking of future anycast requests and also adhere to the constraints of Eqs. (1)–(4).

3.1. Network architecture and assumptions

We consider HTA demands over Grid networks to support e-Science applications. We assume that there are no wavelength converters in the network, so any given lightpath segment must use the same wavelength on all links. It is possible, however, that different segments of a request can use different wavelengths, due to lightpath switching.

We assume the network is under centralized control by a network resource manager. This assumption is reasonable in the case of Grid networks because of the relatively small size. For more general networks, we assume the scheduler is implemented at the domain level for scalability. We consider the time-domain to be broken into discrete timeslots of fixed-size. There is no need for synchronization between the networking elements because the scheduler controls them directly. The storage complexity for the network state, U, is $\Theta(EWH)$, which can be stored as a bit-vector. A common architecture for Grids is to have a centralized resource broker that provides APIs for Grid applications or Grid middleware [16]. The lightpath switching is controlled by the resource broker. The API between the resource broker and grid application that can inform the application when it should transmit on different lightpaths. We assume that users request a single wavelength of bandwidth. The user (or client application) will determine how many timeslots are required given this bandwidth request.

We note that there will be a small overhead for requests that are segmented because of the time it takes to reconfigure the OXCs. We assume that the switching can be done in sub-second time, while the reservations are in the order of hours. The actual reconfiguration of OXCs and the lightpath switching can be done by using existing protocols, such as RSVP-TE [17]. We note that our algorithms are independent of the timeslot duration. Finally, we note that in order to achieve lightpath switching, in addition to the overhead caused due to signaling, there is a fixed cost associated with the migration of a lightpath.[1]

[1] In this paper, we do not consider the cost of migrating to a different lightpath while employing the LPS technique. Moreover in Section 5, we show that with a small number of lightpath switches (2–4), we can achieve considerable performance improvement.

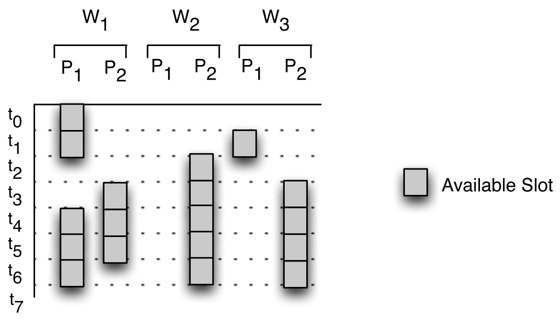


Fig. 1. Representation of the timeslot availability of different lightpaths. We assume that there are two precomputed paths (P_1 and P_2) and three available wavelengths. This leads to a total of six lightpaths, where each lightpath is either available or unavailable for any of the timeslots.

4. RWSA algorithms

In this section we present the dynamic unicast and anycast RWSA heuristics. Our algorithms employ a static k-shortest paths route computation, which are computed using Yen's algorithm [18]. For each source-destination pair, we pre-compute k-shortest paths using Yen's algorithm which finds k loop-less paths that may not be necessarily disjoint.

4.1. Unicast-routing, wavelength and segment assignment (U-RWSA)

In Fig. 1 we represent the state information maintained by the scheduler, wherein the shaded blocks represent the available timeslots. We assume that there are two pre-computed paths (P_1 and P_2) and three wavelengths (W_1, W_2, W_3) in the network for a total of six lightpaths between some source-destination pair. This can be computed for each arriving request based on the wavelength and timeslot availability information stored in the scheduler.

In this example, let the transmission window start at $\alpha = t_0$ and $\tau = 7$. From the figure, there is no single lightpath available starting at t_0 that is also available for seven slots. With lightpath switching, however, we could select P_1 on W_1 during timeslots $[t_0, t_2)$, and P_2 on W_2 during $[t_2, t_7)$, creating schedule $S = \{(t_0, 2, P_1, W_1), (t_2, 5, P_2, W_2)\}$. Lightpath switching allows us to provision this request that would otherwise be blocked. In what follows, we describe the two heuristics (with and without lightpath switching) to solve the U-RWSA problem.

4.1.1. Unicast continuous segments (UCS)

In order to evaluate the performance of lightpath switching, we compare our proposed heuristics with a simple HTA heuristic that does not allow lightpath switching. The heuristic scans all path and wavelength combinations and records all of the available segments with duration of τ. If there are multiple segments found, it will select the segment on the lowest index wavelength. The algorithm, Unicast Continuous Segments (UCS), is shown in Algorithm 1. The *available* function determines how many consecutive slots are available starting at the current time (t_{now}) on the specified lightpath. Determining the number of consecutive slots that are available takes $O(V\tau)$. The complexity of the UCS is then $O(WkV\tau)$.

Algorithm 1: Unicast Continuous Segments (UCS)

```
  Input: $R = (s, d, \tau), G = (V, E, W, H), U[E, W, H]$
  Output: Schedule, $S = \{(t, \tau, P, W)\}$
1 schedule $= \phi$
2 for $w = 1$ to $W$ do
3     for $k = 1$ to $K$ do
4         if available$(P_k, w, t_{now}) \geq \tau$ and lowest index then
5             schedule $= (t_{now}, \tau, P_k, w)$
6 return schedule
```

4.1.2. Lightpath switching (LPS)

In this section we propose a lightpath switching heuristic (LPS) that fills voids on wavelengths in increasing order of wavelength index. LPS starts with the lowest-index wavelength and scans it for unused slots, which are turned into segments. Once all paths on the current wavelength are scanned, it moves to the next higher index wavelength again looking for unused slots that do not overlap in time with previously selected slots and adds these to the schedule.

Algorithm 2 Lightpath Switching (LPS)

```
  Input: $R = (s, d, \alpha, \tau, \omega), G = (V, E, W, H)$,
         $U[E, W, H]$
  Output: Schedule, $S = \{(t_i, d_i, P_i, W_i)\}$
1 schedule $= \phi$
2 for $w$ in $W$ do
3     for $k$ in $K$ do
4         validTimes = findFreeTimes(schedule)
5         for $v$ in validTimes do
6             find segments for $P_k$, $w$ between
              $[v.start, v.end]$
7             insert segments into schedule
8 return schedule
```

The algorithm maintains a list of currently selected segments in the *schedule*. Because we assume simultaneous transmission on multiple lightpaths is not possible, any new segments that will be added to *schedule* cannot overlap in time with anything currently in *schedule*. The *findFreeTimes* function on line 4 (Algorithm 2) returns the gaps in time between segments already in *schedule*. For example, if the current schedule is $S = \{(t_3, 2, P_1, W_1), (t_5, 2, P_1, W_1)\}$, the free times are $[\alpha, t_3), [t_7, \alpha + \tau)$. The algorithm scans these free times for unused slots on the next wavelength, adding new segments to *schedule* as it finds them.

For each lightpath, the *findFreeTimes* function can execute in $O(\tau)$, since there are at most τ segments in the

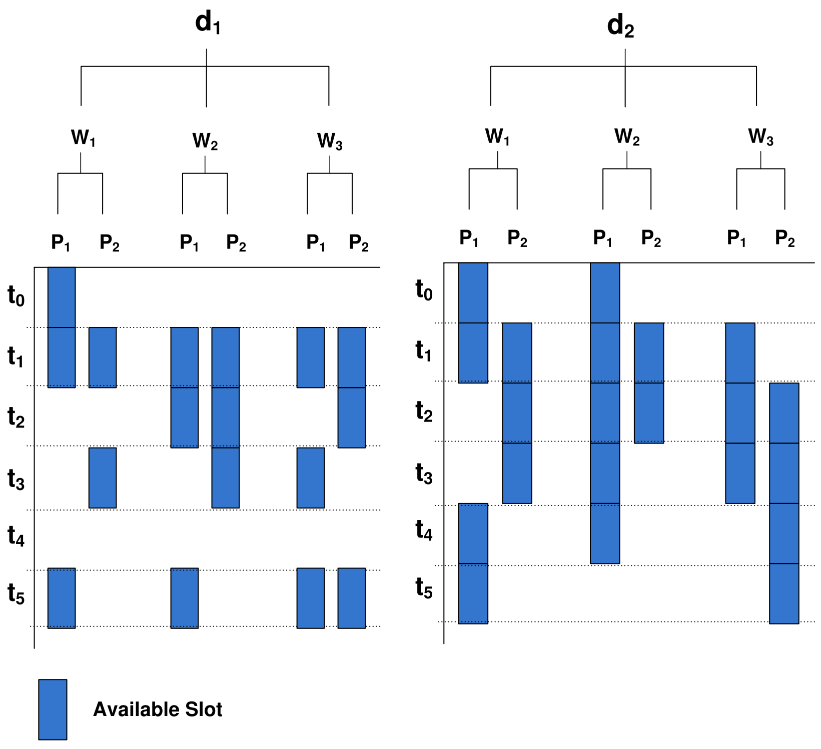


Fig. 2. Representation of timeslot availability for destinations d_1 and d_2.

schedule. There are also at most τ valid timeslots to scan on at most V links of the path. This leads to a runtime of $O(WkV\tau)$, which is the same as the UCS heuristic.

4.2. Anycast-routing, wavelength and segment assignment (A-RWSA)

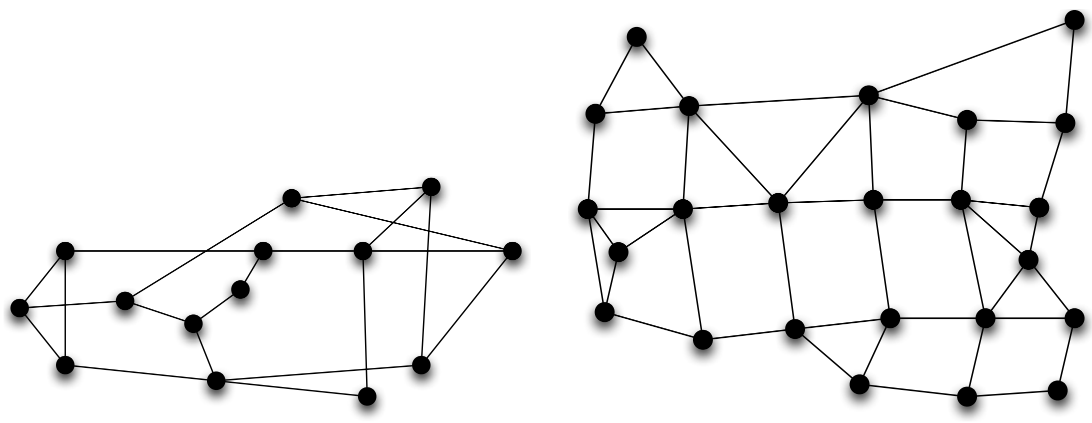
In this section, we first show with the help of an example, the benefit of anycasting as compared to the traditional unicast communication paradigm and then formally describe the algorithms.

In Fig. 2, we depict the state information maintained by the scheduler wherein the shaded blocks represent available timeslots. For the sake of convenience, we assume $k = 2$, pre-computed shortest paths (P_1, P_2) and three wavelengths (W_1, W_2, W_3) in the network. We represent the ith member of the anycast destination set as d_i. For every anycast request, the scheduler computes the wavelength-timeslot availability for each member of the destination set. In this example (Fig. 2), we assume that: (a) the anycast destination set has a total of two members, (i.e., $|i| = 2$), (b) the transmission window starts at $\alpha = t_0$ and (c) the anycast request has a duration of 5 time slots ($\tau = 5$). It can be easily verified that there is no single lightpath available which can accommodate this request. Considering candidate destination 1 (d_1), we note that even if we allow lightpath switching we still cannot provision this request due to unavailability of a wavelength slot at t_4. Since the summation of the segments' duration is not equal to the request's duration, it is deemed *blocked* by the scheduler, i.e., the request cannot be provisioned to destination member d_1.

The scheduler then checks if the request can be provisioned to the next destination member in the set d_2. For this destination, it can be verified that one possible, valid schedule to provision the request is as follows: $S = \{(t_0, 2, P_1, W_1),\ (t_2, 2, P_2, W_1),\ (t_4, 1, P_1, W_1)\}$. The total segment duration equals the duration of the request which satisfies the scheduler constraint. Note that another valid schedule for this request is $S = (t_0, 5, P_1, W_2)$. If we are unsuccessful in finding a valid schedule for the request at any candidate destination member, the scheduler deems the request as blocked. In what follows, we first describe the A-RWSA with Continuous Segments (ACS) heuristic and then outline the criteria that the A-RWSA with Lighpath Switching (ALPS) heuristic uses to find a valid schedule.

4.2.1. Anycast with continuous segment (ACS)

The purpose of proposing ACS in this paper is twofold. First is to compare the performance improvement achievable in blocking as compared to the traditional unicast RWA problem. The second is to make a comparison of the ACS to the ALPS which will render the benefit one achieves by using lightpath switching. The ACS heuristic finds the shortest paths to all the M anycast destinations and sorts them based on the shortest path length (hop count) in increasing order. The heuristic scans all k shortest paths and wavelength combinations to find all the available segments with duration τ for each destination in the anycast destination set. If there are multiple segments found, it selects the segment on the lowest index wavelength. The ACS heuristic is outlined in Algorithm 3.



(a) 14-node NSFnet.

(b) 24-node network.

Fig. 3. Networks used for heuristic evaluation.

```
Algorithm 3: Anycast with Continuous Segment (ACS)
   Input: $R = (s, d_{c(i)}, \tau)$, $G = (V, E, W, H)$, $U[E, W, H]$
   Output: Schedule, $S = \{(t, \tau, P, W)\}$
1  schedule = $\phi$
2  Find shortest path to all $M$ Anycast destinations
3  Sort Anycast Destinations based on the shortest path
4  then
5  for $d = 1$ to $M$ do
6      for $w = 1$ to $W$ do
7          for $k = 1$ to $K$ do
8              if available($P_k, w, t_{now}$) $\geq \tau$ and lowest index then
9                  schedule = $(t_{now}, \tau, P_k, w)$
10     return schedule
11 return
```

The *available* function (Line 8) determines how many consecutive slots are available starting at the current time (t_{now}) on a specific lightpath. For any of the wavelengths on the k shortest paths, if the available number of slots returned by this function is greater than or equal to the holding time specified by the user's request then the request is deemed to be provisioned.

If the requested number of timeslots (from the source to a candidate destination D_i), are not available, the algorithm moves to the next destination and repeats the above procedure. If the requested number of time slots cannot be found on any path (on a particular wavelength) to any member of the destination set, the scheduler blocks the request. The complexity of the ACS can be computed as follows: determining the number of consecutive slots that are available for a particular source-destination pair takes $O(V\tau)$. The complexity of ACS in the worst case is then $O(MWkV\tau)$ wherein ACS has to search through all the single segments equal to request's duration to find the schedule for the request.

4.2.2. Anycast with lightpath switching (ALPS)

The ALPS algorithm aims to reduce the number of wavelengths used to schedule lightpaths so as to reduce the number of optical sources (e.g., lasers) needed to provision these lightpaths. ALPS fills voids on wavelengths by packing the user requests as closely as possible on the available wavelengths. The heuristic packs the wavelengths in increasing order of wavelength index. The ALPS heuristic finds the shortest paths to all the M anycast destinations and sorts them based on the shortest path length (hop count) in increasing order. For a particular candidate destination the ALPS starts with the lowest index wavelength and searches for unused slots on all paths, which are converted into segments. If the requested number of slots are not found on a particular wavelength, it scans the next wavelength for unused slots (which do not overlap in time with previously selected segments) and adds them to the schedule. If the summation of the duration of the segments added thus far equals the request's duration then the request is said to be provisioned and the schedule is added to the scheduler. Otherwise the schedule is deleted and the ALPS algorithm then starts the same procedure as described above for the next best anycast candidate destination pair (s, D_{i+1}). After scanning for all the possible segments and trying to create a schedule for the request on all the (s, D_i) anycast pairs, if the lightpath cannot be provisioned then the request is deemed to be blocked. We outline the ALPS heuristic in Algorithm 4.

As we assume simultaneous transmission on multiple lightpaths is not possible for a particular request, any new segments that are added to the schedule cannot overlap in time with any segment currently in the schedule. The *findFreeTimes* function in ALPS algorithm (Line 8) returns the voids in time between segments already in schedule. For instance, consider the first destination node D_1 in Fig. 2. The current schedule for the request is $S = (t_2, 1, P_1, W_3), (t_4, 1, P_1, W_3)$. The free times returned by the free time function are $(t_3, 1), (t_5, 1)$. The algorithm scans these free times for unused slots on the next higher index wavelength to create new segments and adds them to the schedule as it find them.

For each lightpath, the findFreeTimes function can execute in $O(\tau)$, since there are at most τ segments in the schedule. For each of the M possible candidate

Algorithm 4: Anycast with Lightpath Switching (ALPS)

```
    Input: $R = (s, d_{c(i)}, \alpha, \tau, \omega)$, $G = (V, E, W, H)$,
           $U[E, W, H]$
    Output: Schedule, $S = \{(t_i, d_i, P_i, W_i)\}$
1   $schedule = \phi$
2   Find the shortest path to all $M$ Anycast destinations
3   Sort Anycast Destinations based on the shortest path
4   then
5   for $d = 1$ to $M$ do
6       for $w$ in $W$ do
7           for $k$ in $K$ do
8               $validTimes = findFreeTimes(schedule)$
9               for $v$ in $validTimes$ do
10                  find segments for $P_k$, $w$ between
                    $[v.start, v.end]$
11                  insert segments into $schedule$
12      return $schedule$
13  return
```

destinations, there are at most τ time slots to search on at most V links of the k shortest paths on every wavelength. Therefore the complexity of ALPS in the worst case is $O(MWkV\tau)$.

5. Performance evaluation

We now provide the simulation results for our proposed heuristics. We have used the following parameters: the arrival process is a Poisson process with an exponentially distributed holding time. The horizon is large enough (equal to 2000 timeslots) so that none of the requests are blocked due to their holding time. The time slot size vastly depends upon the type of traffic that a network operator expects. It could be fine tuned by the operator to range from seconds to hours. The primary performance metric we look at is the blocking probability of a connection, which is defined as the fraction of connections that cannot be scheduled. We also determine in our simulations, the hop count (defined as the number of physical hops that a request takes to reach its destination) and average number of lightpath switches defined as the ratio of number of lightpath switches to the number of successfully switched requests. We simulate 10^6 requests and take the average of 30 runs.[2] We use different values of k ($k = 1, 2, \ldots, 4$) for pre-computing shortest paths using Yen's k-shortest path algorithm. We evaluate our heuristics on the 14-node NSFnet and the 24-node network as shown in Fig. 3, and consider that each link in the network is equipped with $W = 8$ wavelengths.

In Figs. 4 and 5 we show the performance of the two heuristics – UCS and LPS, used to solve the U-RWSA problem for the NSFnet and 24-node networks respectively. From Figs. 4(a) to 5(a), we can observe that the blocking probability is significantly reduced when lightpath switching is allowed. At lower loads we observe that the blocking probability is reduced by approximately two orders of magnitude for the NSFnet (approximately an order of magnitude for the 24-node network), and LPS lowers the blocking by about 3%–5% at higher loads. With $k = 1$, the performance improvement between UCS and LPS is not as significant because here only wavelength switching would be possible. For all other values of k, the relative improvement of LPS to UCS is about the same.

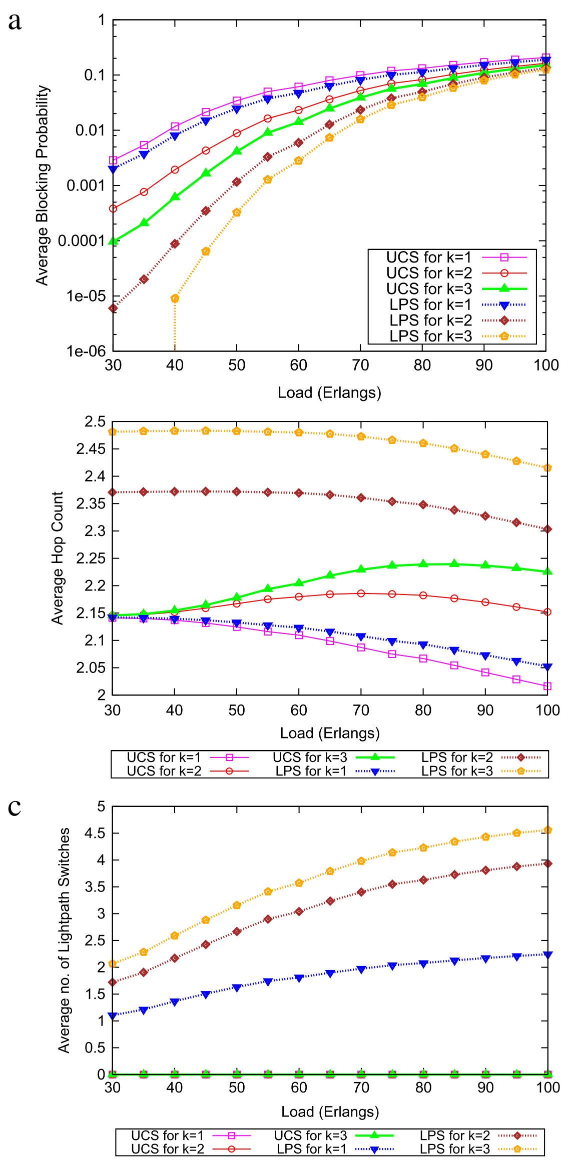


Fig. 4. Comparison of UCS and LPS for 14-node NSFnet (a) average blocking probability, (b) average hop count, and (c) average number of lightpath switches.

In Figs. 4(b) and 5(b) we show that the hop count for UCS is quite lower than LPS, as LPS switches paths heavily to achieve higher blocking performance. Fig. 4(c) depicts the average number of lightpath switches that occur for each request. The mean holding time for each request is twelve timeslots. The number of lightpath switches ranges

[2] We also compute confidence intervals, but do not plot them here on any of the curves to avoid cluttering of the figures. However, we noticed that the 95% confidence intervals were quite narrow in all cases.

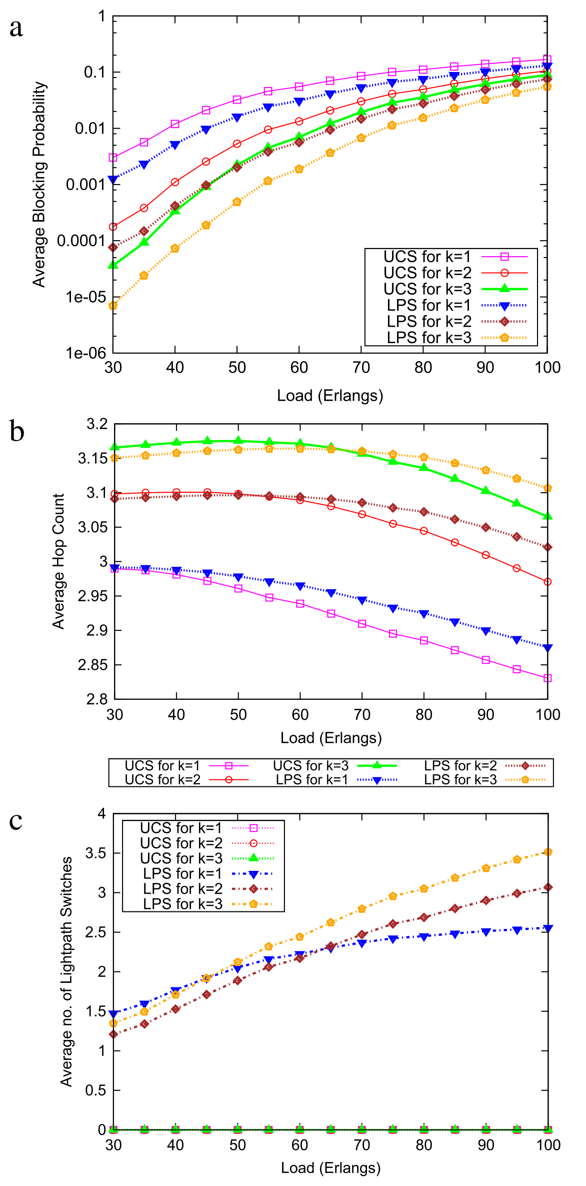


Fig. 5. Comparison of UCS and LPS for 24-node network (a) average blocking probability, (b) average hop count, and (c) average number of lightpath switches.

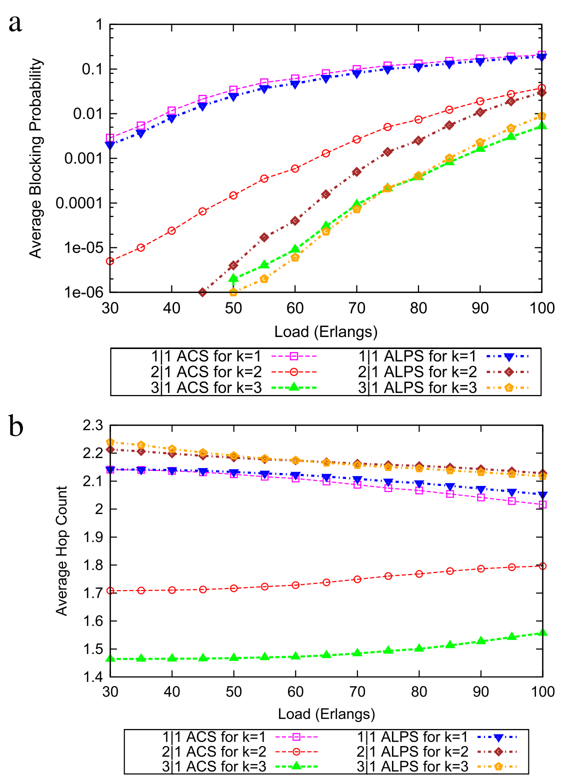


Fig. 6. Comparison of ACS and ALPS for 14-node NSFnet (a) average blocking probability and (b) average hop count.

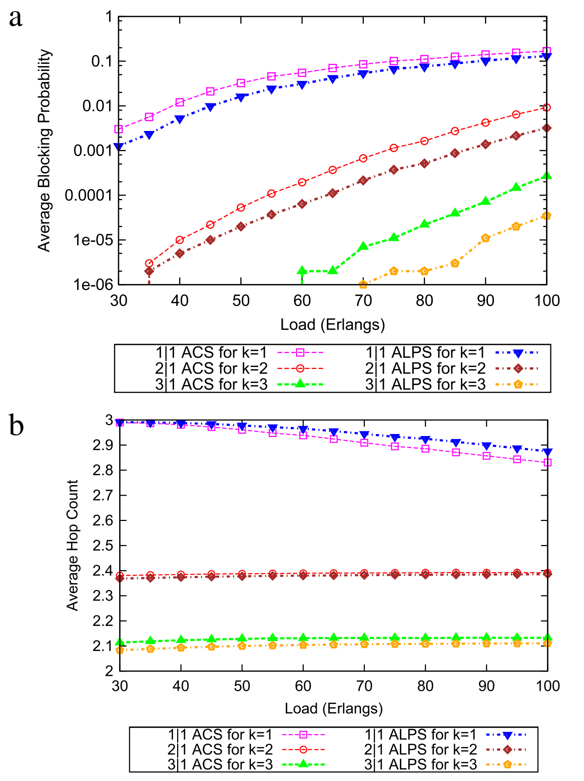


Fig. 7. Comparison of ACS and ALPS for 24-node network (a) average blocking probability and (b) average hop count.

from two to four, meaning each segment is on average four timeslots at low loads and over two timeslots at higher loads. A similar trend was observed for the 24-node network as shown in Fig. 5(c). These results show there is a trade off between reduced blocking and increased network signaling (number of lightpath switches). As the value of k increases, the number of lightpath switches also increases while the blocking probability decreases with LPS. We also note that depending on the network's average nodal degree, there is a maximum value of k for which no further improvement occurs.

In Fig. 6, we show the performance of the two heuristics – ACS and ALPS, used to solve the A-RWSA problem for the NSFnet. We use the notation $P|1$ to represent an anycast request wherein P is the total number of candidates in

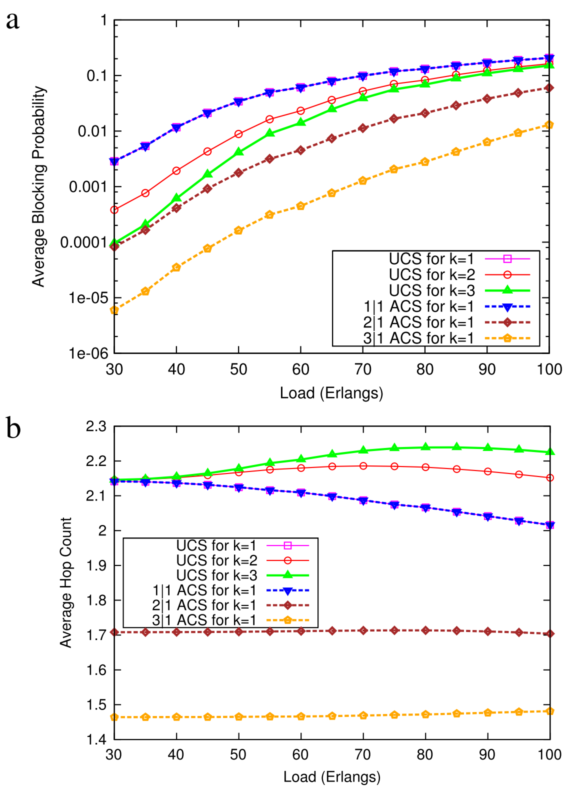


Fig. 8. Comparison of UCS and ACS for 14-node NSFnet (a) average blocking probability and (b) average hop count.

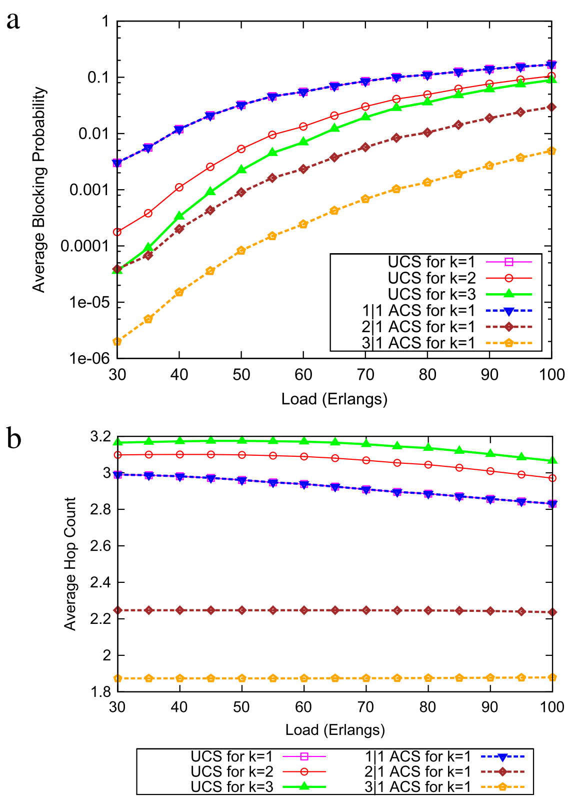


Fig. 9. Comparison of UCS and ACS for 24-node network (a) average blocking probability and (b) average hop count.

the destination set, out of which we have to schedule the anycast request to any 1 destination member. In Fig. 6(a) we show that for 2|1 ALPS with $k = 2$ reduces blocking by approximately 32% when compared to 2|1 ACS with $k = 2$ at lower loads and achieves about a 3%–5% reduction at higher loads. However, 3|1 ALPS with $k = 3$ and 3|1 ACS with $k = 3$ behave quite identically across different load values. In Fig. 6(b) we show that ACS reduces hop counts by 65% when compared to ALPS for $M \geq 2$ and $k \geq 2$ which does not result in any notable blocking improvement. A similar trend was observed for the 24-node network as shown in Fig. 7.

In Fig. 8 we compare the performance of the traditional unicast (i.e., $M = 1$) RWSA with continuous segments (UCS) to that of anycast RWSA with continuous segments (ACS) for the NSFnet. Fig. 8(a) shows that ACS with $M \geq 2$, lowers blocking by about 50% as compared to UCS for lower traffic loads, and by about 40% for higher traffic loads. Fig. 8(b) shows that ACS with $M \geq 2$, significantly reduces hop count by about 50% when compared to UCS. The significant decrease in blocking and hop count is due to the fact that ACS has higher chances of successfully establishing lightpaths from source node to the best candidate destination node from the ordered anycast destination set. Note that we also compared the performance of the ALPS to that of the UCS. It was observed that ALPS (just as the ACS) outperforms the UCS (by over two orders of magnitude) in blocking performance. A similar trend in performance was observed for the 24-node network as shown in Fig. 9.

In Fig. 10, we compare the performance of ALPS to LPS for the NSFnet. In Fig. 10(a), we show that ALPS (3|1, with $k = 1$ shortest path) achieves over two-orders of magnitude improvement in blocking as compared to LPS with $k = 3$ shortest paths. For ALPS (2|1, with $k = 1$ shortest path) it is observed that over an order of magnitude improvement in blocking is achieved as compared to LPS with $k = 2$ shortest paths. In Fig. 10(b) we show that for M|1 ALPS, with $M \geq 2$, reduces the hop count by approximately 16%–24% as compared to LPS with $k \geq 2$. In Fig. 10(c) we compare the number of lightpath switches required for M|1 ALPS ($M \geq 2$) to LPS with $k \geq 2$. It is observed that to achieve the blocking shown in Fig. 10(a), ALPS requires almost 50% lower average number of lightpath switches as compared to LPS. In Fig. 11, we show the corresponding results for the comparison between ALPS and LPS for the 24-node networks. In Fig. 11(a) we observe that ALPS achieves approximately an order of magnitude improvement in blocking as compared to LPS, while utilizing 32%–38% lower average number of physical hops (Fig. 11(b)) and approximately 62%–65% lower average number of lightpath switches (Fig. 11(c)).

Finally in Figs. 12 and 13 we compare all the proposed heuristics under similar network conditions for the NSFnet and 24-node network respectively. It can be observed that both M|1 ACS and M|1 ALPS show significant blocking improvement when compared with UCS and LPS. We can also see that ALPS outperforms LPS across all traffic loads.

From the results of our performance evaluations we note that, both the ALPS and ACS show that anycasting

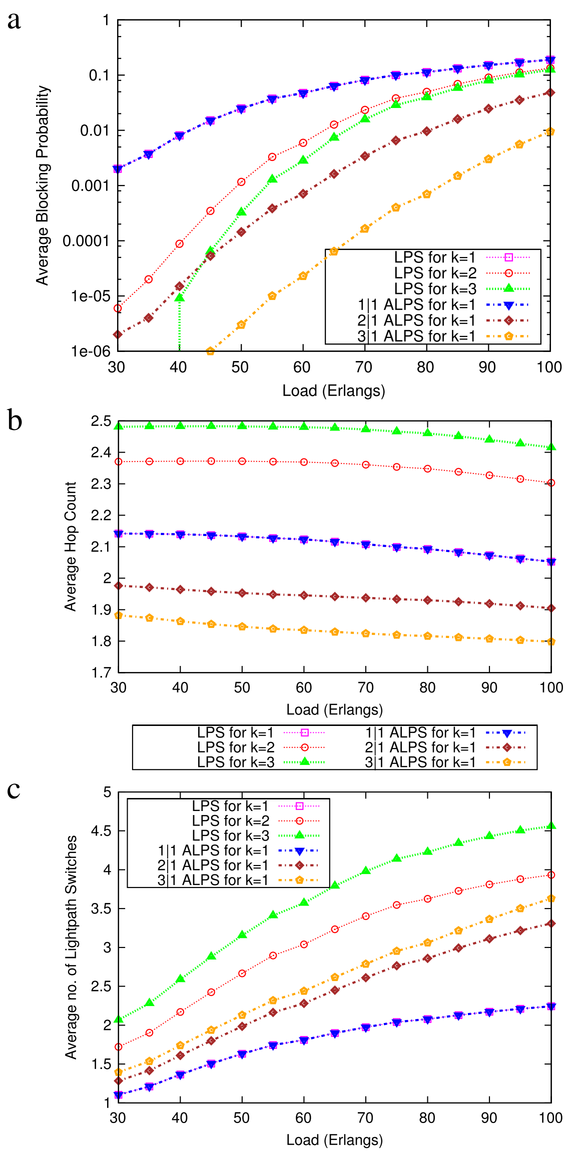


Fig. 10. Comparison of LPS and ALPS for 14-node NSFnet (a) average blocking probability, (b) average hop count, and (c) average no. of lightpath switches.

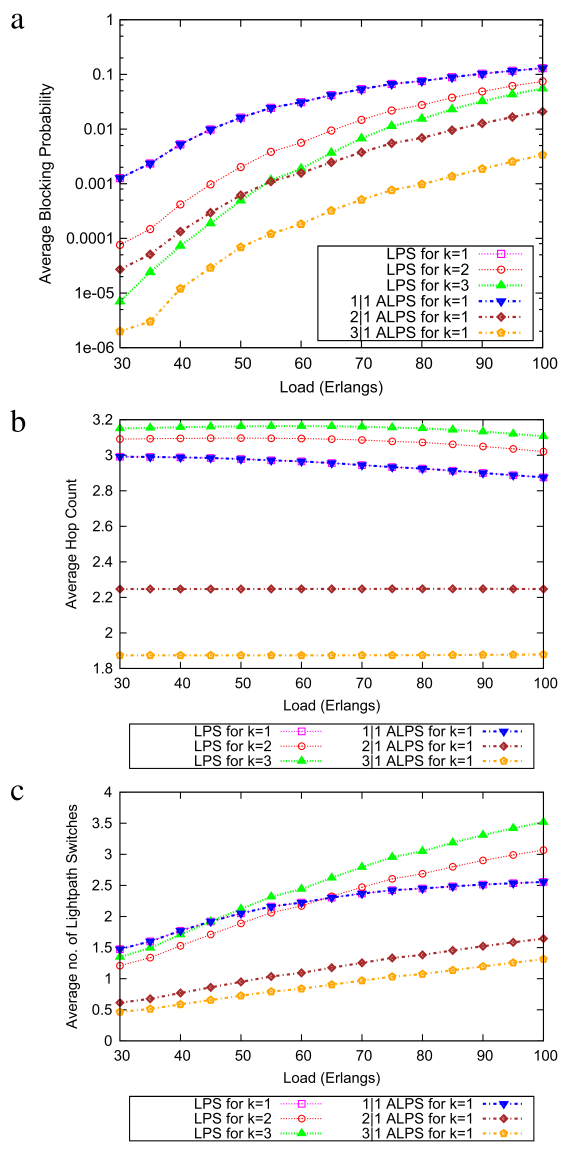


Fig. 11. Comparison of LPS and ALPS for 24-node network (a) average blocking probability, (b) average hop count, and (c) average no. of lightpath switches.

with lightpath switching on the shortest path to the destinations can lead to a significant improvement over unicast lightpath switching with k shortest paths and the traditional RWA (UCS). We also show that the increase in lightpath switches is linearly proportional to the value of k in ALPS. We note that as the average nodal degree of the 14-node NSFnet topology is 3 (3.37 for the 24-node network). Any value of k greater than 3 does not lead to significant performance improvements.

Finally, in Table 1 we compare the time-complexities of the anycast heuristics implemented in this paper for the NSFnet and 24-node network. We show a sample comparison for the following values: $k = 3$ shortest path computations (using Yen's algorithm), $M = 3$ candidate destination nodes and $\tau = 2000$ timeslots. It can be observed that the ACS and ALPS heuristics take an order of M times longer than the UCS and LPS heuristics to complete.

Table 1
Time complexity of heuristics.

Network	V	W	k	M	τ	UCS/LPS	ACS/ALPS
NSFnet	14	8	4	2000	3	896,000	2,688,000
24-node	24	8	4	2000	3	1,536,000	4,608,000

6. Conclusion

In this paper we propose lightpath switching for holding-time-aware demands. We show that allowing a request to use multiple lightpaths over its duration can significantly decrease blocking probability. We also addressed

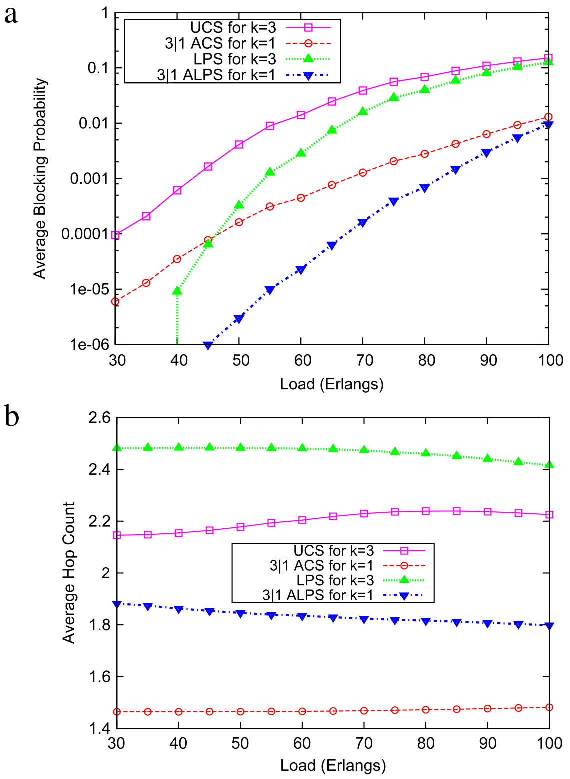


Fig. 12. Comparison of UCS, ACS, LPS, and ALPS for 14-node NSFnet (a) average blocking probability and (b) average hop count.

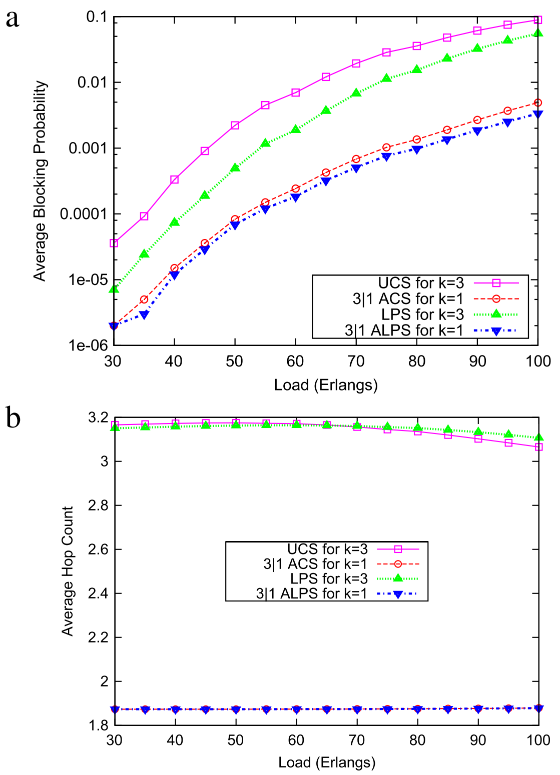


Fig. 13. Comparison of UCS, ACS, LPS, and ALPS for 24-node network (a) average blocking probability and (b) average hop count.

the problem of routing anycast-holding-time-aware demands in all-optical wavelength routed networks. We proposed two heuristics – ACS and ALPS, to solve the anycast RWSA problem. We first compared the performance of these heuristics to the traditional unicast RWSA heuristic (UCS), for provisioning lightpaths. We observed that both the ALPS and ACS, showed significant improvement (40%–50%) in blocking as compared to UCS. We further compared the performance of the ALPS to that of the ACS and observed that ALPS achieves almost an order of magnitude improvement in blocking, while using a significantly lower number of physical hops. These results suggest the benefits rendered by using the lightpath switching technique to route the traffic demands. Although the technique of lightpath switching aids in lowering the blocking performance, we note that there is a tradeoff between reduced blocking and increased network signaling which is incurred due to an increase in the number of lightpath switches. Areas of future work include anycast RWSA for advanced reservation requests and developing heuristics for dynamic routing.

References

[1] G.E. Keiser, A review of WDM technology and applications, Optical Fiber Technology 5 (1) (1999) 3–39.
[2] R. Ramaswami, K. Sivarajan, Routing and wavelength assignment in all-optical networks, IEEE/ACM Transactions on Networking 3 (1995) 489–500.
[3] I. Chlamtac, A. Ganz, G. Karmi, Lightpath communications: an approach to high-bandwidth optical WANs, IEEE Transactions on Communications 40 (7) (1992) 1171–1182.
[4] M. Tornatore, A. Baruffaldi, H. Zhu, B. Mukherjee, A. Pattavina, Holding-time-aware dynamic traffic grooming, IEEE Journal on Selected Areas in Communications 26 (3) (2008) 28–35.
[5] E. Zegura, M. Ammar, Z. Fei, S. Bhattacharjee, Application-layer anycasting: a server selection architecture and use in a replicated web service, IEEE/ACM Transactions on Networking 8 (4) (2000) 455–466.
[6] N. Charbonneau, V.M. Vokkarane, Dynamic circuits with lightpath switching over wavelength routed networks, in: Proceedings of IEEE ANTS, Mumbai, India, 2010.
[7] D. Banerjee, B. Mukherjee, A practical approach for routing and wavelength assignment in large wavelength routed optical networks, IEEE Journal on Selected Areas in Communications 14 (5) (1996) 903–908.
[8] H. Zang, J. Jue, B. Mukherjee, A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks, Optical Networks Magazine 1 (2000) 47–60.
[9] D. Lucerna, M. Tornatore, B. Mukherjee, A. Pattavina, Dynamic routing of connections with known duration in WDM networks, in: Proceedings of the 28th IEEE GLOBECOM, Honolulu, Hawaii, USA, 2009.
[10] J. Kuri, N. Puech, M. Gagnaire, E. Dotaro, R. Douville, Routing and wavelength assignment of scheduled lightpath demands, IEEE Journal on Selected Areas in Communications 21 (8) (2003) 1231–1240.
[11] J. Zheng, B. Zhang, H. Mouftah, Towards automated provisioning of advance reservation service in next generation optical internet, IEEE Communications Magazine 44 (12) (2006) 68–74.
[12] T. Stevens, et al., Multi-cost job routing and scheduling in Grid networks, Future Generation Computer Systems 25 (8) (2009) 912–925.
[13] Y. Chen, A. Jaekel, A. Bari, Resource allocation strategies for a non-continuous sliding window traffic model in WDM networks in: Proc. of ICST Broadnets, 2009.
[14] N. Charbonneau, V.M. Vokkarane, Dynamic non-continuous advance reservation over wavelength-routed networks, in: Proceedings, 20th IEEE ICCCN 2011, Maui, USA, 2011.
[15] S. Naiksatam, S. Figueira, Elastic reservations for efficient bandwidth utilization in lambda-Grids, Future Generation Computer Systems 23 (2007) 1–22.
[16] C. Guok, D. Robertson, E. Chaniotakis, M. Thompson, W. Johnston, B. Tierney, A user driven dynamic circuit network implementation, in: Proceedings of IEEE GLOBECOM Workshops, 2008.

[17] RFC 3209: RSVP-TE extensions to RSVP for LSP tunnels.
[18] J. Yen, Finding the *k* shortest loopless paths in a network, Management Science 17 (11) (1971) 712–716.
[19] B. Ramaprasad, A. Gadkar, V.M. Vokkarane, Dynamic anycasting over wavelength routed networks with lightpath switching in: Proc. of IEEE Conf. on High Performance Switching and Routing, 2011.