A Flexible Reservation Algorithm for Advance Network Provisioning *

Mehmet Balman[1], Evangelos Chaniotakis[2], Arie Shoshani[1], Alex Sim[1]
[1] *Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[2] *Energy Sciences Network, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
Email: {mbalman, echaniotakis, ashoshani, asim}@lbl.gov

April, 2010 †

Abstract

Many scientific applications need support from a communication infrastructure that provides predictable performance, which requires effective algorithms for bandwidth reservations. Network reservation systems such as ESnet's OSCARS, establish guaranteed bandwidth of secure virtual circuits for a certain bandwidth and length of time. However, users currently cannot inquire about bandwidth availability, nor have alternative suggestions when reservation requests fail. In general, the number of reservation options is exponential with the number of nodes n, and current reservation commitments. We present a novel approach for path finding in time-dependent networks taking advantage of user-provided parameters of total volume and time constraints, which produces options for earliest completion and shortest duration. The theoretical complexity is only $O(n^2r^2)$ in the worst-case, where r is the number of reservations in the desired time interval. We have implemented our algorithm and developed efficient methodologies for incorporation into network reservation frameworks. Performance measurements confirm the theoretical predictions.

1 Introduction

We are witnessing a new era that offers opportunities to conduct scientific research taking advantage of recent advancements in computational and storage technologies. Computationally intensive science spans multiple scientific domains, such as particle physics, climate modeling, and bio-informatics simulations. Scientific applications generate many terabytes and even petabytes of data. In addition to extreme storage requirement, these large-scale applications necessitate collaborators to access very large data sets resulting from simulations performed in geographically distributed institutions. Often, scientific experimental facilities generate massive data sets that need to be transferred to validate the simulation data in remote collaborating sites. For example, in high energy physics, the Large Hadron Collider (LHC) is expected to generate 100 gigabits per second in the near future. The generated data is propagated to other research sites for further analysis. Similarly, in the Earth System Grid (ESG) [2], 35 terabytes of data is shared by more than 16000 users worldwide; and the next generation climate data archive is expected to be more than 1 petabyte.

The need for transferring data chunks of ever-increasing sizes through the network shows no sign of abating. A major component needed to support these needs is the communication infrastructure which enables large-scale data replication, high performance remote data analysis and visualization, and also provides access to computational resources. In order to provide high-speed on-demand data access between collaborating institutions, national governments support next generation research networks such as Internet2 and the Energy Sciences Network (ESnet) [1]. Delivering network-as-a-service that provides predictable performance, efficient resource utilization and better coordination between compute and storage resources is highly desirable. Research institutions developed dedicated high-bandwidth networks which are able to provision the communication channels when the data, especially large-scale massive data, is ready to be transferred.

We study the network provisioning and advanced bandwidth reservation in ESnet for on-demand high performance data transfers. A reservation request from a user includes desired bandwidth allocation between end-points with duration and starting time information. The bandwidth reservation system, called On-demand Secure Circuits and Advance Reservation System (OSCARS) [3], serves as the network provisioning agent on ESnet. OSCARS checks network availability and capacity for the specified duration of time, and allocates it for the user if it is available. Otherwise, it reports to the user that it is unable to provide the required allocation. Accordingly, it

*SC10 November 2010, New Orleans, Louisiana, USA

† This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or The Regents of the University of California.

falls upon the user to search for a time-frame of a required bandwidth by trial-and-error, not having knowledge of the network's available capacity at a certain instant of time. We address the problem of improving the current ESnet advance network reservation system, OSCARS, by presenting to the clients possible reservation options and alternatives for earliest completion time and shortest transfer duration.

In this paper, we present a novel approach for path finding in time-dependent transport networks with bandwidth guarantees. We report an algorithm, where the user specifies the total volume that needs to be transferred, a maximum bandwidth that can be used and provisioned in the client sites, and a desired time window within which the transfer should be done. The proposed algorithm can find alternate allocation possibilities, including earliest time for completion, or shortest transfer duration - leaving the choice to the user. We describe the algorithm and show that its complexity is quadratic with number of nodes and existing reservations. It is therefore quite practical when applied to large networks with hundreds, even thousands of routers and links. We have implemented our algorithm for testing and incorporation into a future version of OSCARS. However, the algorithm is not specific to OSCARS, and can be used with any network reservation framework.

The organization of this paper is as follows. In Section 2, we highlight related work in the literature and compare with our new approach. In Section 3, we explain network reservation, define the problem and give details of a new network reservation service. In Section 4, we present challenges in time-dependent transport networks with bandwidth guarantees. In Section 5, we provide details about our methodology and discuss the efficiency of the proposed algorithm. In Section 6, we give implementation details and describe data structures developed to enhance the performance of the algorithm. Finally, we conclude with a brief summary and discussion on future work.

2 Related Work

Dedicated bandwidth channels are crucial requirements in distributed computing middleware to satisfy large scale data movement [14, 18]. On-demand bandwidth circuits provide predictable performance and data transfer duration. Advance network reservation helps users and client tools in the cooperating organizations to prepare for efficient and reliable data movement. This also enables well-organized resource utilization in which communicating parties can plan ahead and provision collaborating resources.

There are few studies in advance bandwidth reservation in the literature [11, 19, 6]. The network reservation problem and path computation with guaranteed bandwidth have been categorized into several domains in [20, 13]. One of those problems is to reserve a fixed slot in which we find a path from source to destination with a specific bandwidth requirement. Some other problems include finding the path with the largest bandwidth in a specific time slot, and finding the first time slot in which there is a path with the specified bandwidth requirement from source to destination. Those problems can be solved by extending and modifying known graph algorithms. They do not address flexible reservation, and do not provide alternative suggestions to the user. The proposed solution methods in [20, 13] represent time-dependent network topology by keeping available bandwidth information for every link in each time slot. This slotted time window model has high cost and it is not effective for real-life advance reservations systems.

Bandwidth scheduling problems for multiple data transfer requests are introduced in [15, 21, 10]. The main objective in [15] is to assign a network path for each reservation request with fixed bandwidth in a predetermined time period. A greedy heuristic is given in which requests consuming less resource are given preference in scheduling. These algorithms have high complexity and large space requirements. They do not compute an optimal reservation for a massive data transfer request, and do not suggest any allocation pattern.

In our approach, we discretize the time-dependent dynamic network topology by dividing the search interval into time steps. Each time step represents a stable status of the topology. We provide a methodology to calculate static snapshot graphs in each time steps and and apply max-bandwidth algorithm while traversing over the search interval. We show that the number of subsequent combinations of time steps is bounded by the number of reservations in the system. Searching the given time interval is accomplished in polynomial time. Hence, we provide an efficient algorithm to find possible advance network reservation options for the given data transfer requirements.

3 Advance Network Reservation: Background Information

ESnet provides high-bandwidth connections between more than 40 research laboratories and academic institutions for data sharing and video/voice communication. Experimental facilities, supercomputing centers and thousands of scientists are connected with ESnet. The ESnet's bandwidth reservation system, OSCARS, establishes guaranteed bandwidth of secure virtual circuits at a certain time, for a certain length of time and bandwidth. Though OSCARS operates within the ESnet, it also supplies end-to-end provisioning between multiple autonomous network domains. OSCARS gets reservation requests through a standard web service interface, and conducts a Quality-of-Service (QoS) path for bandwidth guarantees. Multi-Protocol Label Switching (MPLS) and the Resource Reservation Protocol (RSVP) enable ESnet to create a virtual circuit using Label Switched Paths (LSP's). It contains three main components: a reservation manager, a bandwidth scheduler, and a path setup subsystem [12]. The bandwidth scheduler needs to have information about the current and future states of the network topology in order to accomplish end-to-end bandwidth guaranteed paths.

The OSCARS bandwidth reservation system keeps track of changes in the network status and maintains a topology graph which can simply be described as follows. Every port in a router has a maximum bandwidth available for reservation, and each network link connecting two ports (providing communication from one router towards another one) has an 'engineering metric'. The engineering metric is used by network engineers to assign usage priority and preference to particular links to determine the most desirable paths to reserve. This is a common technique used in dedicated networks. Although we are not bounded by this metric and our algorithm works without taking it into account, we also consider the engineering metric in path computation.

A reservation request R to OSCARS consists of a source node v^s and destination node v^d, requested bandwidth $\mathtt{M}$, start time t^s and end time t^e: $R = (v^s, v^d, \mathtt{M}, t^s, t^e)$. Since there might be bandwidth guaranteed paths in the system that are already fully or partially committed, the reservation system needs to ensure availability of the requested bandwidth from source to destination for the requested time interval. In order to eliminate over commitment, committed reservations between start and end times are examined to extract available bandwidth information for each link in the time period. The shortest path is calculated based on the engineering metric on each link, and a bandwidth guaranteed path is set up from source to destination, to commit the reservation request for the given time period.

Problem Definition: Advance network reservation systems like OSCARS enable users to obtain guaranteed requested bandwidth for a certain duration of time. However, if the requested reservation cannot be granted, no further suggestion is returned back to the user, except a failure message. As mentioned above, in such a situation, users have to go through a trial-and-error sequence, and may need to try several advance reservation requests until they get an available reservation. These try-and-error attempts may also overload the system. Even if a user successfully reserves the network after several trials, the choice of the allocation might not be one of the optimal ones available in the system. Further, there is no possibility from the user's point of view to be aware of the other possibilities that might fit better into his/her requirements. In other words, users cannot, in general, make an optimal choice. Moreover, the current method of selecting a path may lead to ineffective use of the overall system such that network resources may not be used as optimally as possible.

Our goal is to enhance the OSCARS reservation system by extending the underlying mechanism to provide a new service in which users submit their constraints and the system suggests possible reservation options satisfying users' requirements.

Flexible Network Reservation: We developed a new methodology in which users submit constraints and the system suggests possible reservations options. In this approach, instead of giving all reservation details such as the amount of bandwidth to allocate between start/end times, users provide maximum bandwidth they can use, total size of the data requested to be transferred, the earliest start time, and the latest completion time. Moreover, users can set criteria such that they would like to reserve a path for earliest completion time or reserve a path for shortest transfer duration. Such a request can be represented as: $S = (v^s, v^d, \mathtt{M}^{max}, D, t^E, t^L)$, where D is the total size of data to be sent from v^s to v^d, and t^E the earliest start time, t^L is the latest end time. The flexible network reservation algorithm finds out a reservation $R = (v^s, v^d, \mathtt{M}, t^s, t^e)$ for the earliest completion or for the shortest duration where $\mathtt{M} \leq \mathtt{M}^{max}$ and $t^E \leq t^s < t^e \leq t^L$. The maximum bandwidth $\mathtt{M}^{max}$ is related to the capability of the client and server hosts between source and destination end-points. It also depends on intermediate hosts and routers (in the client sites) in order to achieve end-to-end optimization. Even if the network can provide a higher bandwidth than the maximum requested, there is no value in providing that since the user is not able to use all the available bandwidth due to limitations in the client and server sites. The focus of our work is to optimize bandwidth allocation in the wide-area backbone (between edge routers). Other projects such as TeraPath [4] and LamdaStation address reservations between the clients and edge routers.

4 Time-Dependent Transport Networks

In advance network reservation, we first need to ensure the availability of the requested bandwidth before committing a bandwidth allocation request. The foremost question is how to find the maximum bandwidth available for allocation from a source node to a destination node. The max-bandwidth path algorithm [17] is well known in quality-of-service (QoS) routing problems in which a path is constructed from source to destination whose bandwidth is maximized, given that each link is associated with an available bandwidth value.

The QoS condition is a bottleneck constraint in max-bandwidth path calculation. Alternatively, in shortest path calculation, we find a path whose sum of weights is minimized, and QoS constraint is additive (minimum delay path, or minimum hop count path). The max-bandwidth path algorithm is a slightly modified version of Kruskal and Dijkstra's algorithms with the same asymmetrical time complexity [17]. In the shortest path algorithm, the weight of a path is the sum of values added by each link in the path. On the other hand, the weight of a path in max-bandwidth is the minimum link bandwidth, the bottleneck link over the path. Those algorithms are very fast and efficient, and they have been adapted to deal with many problems in routing and gateway protocols. In a graph with n nodes, there is a total $n!$ paths from source to destination. The main advantage of those types of graph algorithms is that maximum n^2 paths are visited even in the worst-case.

We deal with a dynamic network such that the bandwidth value for every link is time dependent. While constructing a path and calculating the available bandwidth over a path, we need to consider another variable, time; therefore, the dimension of the problem is extended by adding the time variable such that the state of the topology depends on the time period. Graph algorithms for time-dependent dynamic networks has been studied in the literature especially for max-flow and shortest path algorithms [16, 9, 7]. The most common approach is the discrete-time algorithms in which the time is modeled as a set of discrete values and a static graph is constructed for every time interval. As an example, [8] uses time-expanded max flow for data transfer scheduling, and [16] presents various shortest path algorithms for dynamic networks with time-dependent edge weights.

Analogous Example: We need different types of algorithms to analyze time-dependent max-bandwidth path calculation. The following is given to clarify the advance bandwidth reservation in dynamic networks. Assume a vehicle travels from city A to city B where there are multiple cities between A and B connected with separate highways. Each highway has a specific speed limit but the speed is lower if there is high traffic load on the road, and we know the load on each highway for every time period. The first question is which path the vehicle should follow in order to reach city B as early as possible. Alternatively, we can delay our journey and start later if the total travel time would be shortened. Thus, for both questions, we need to find the route along with the staring time and end time. There is one more condition we need to satisfy, since we are dealing with bandwidth reservation where allocation should be set in advance when a request is received. If we apply this condition to the example problem described above, we have to set the speed limit before starting and cannot change that during the entire journey. Therefore, known algorithms do not fit into our problem domain. This distinguishes our path calculation from other time-dependent graph algorithms in the literature.

5 Methodology and Algorithm

We define the network topology as a time-dependent directed graph $G_T(\mathtt{V}, \mathtt{E}, \mathtt{X}_E(T))$, with a vertex set $\mathtt{V}$ of n nodes, and an edge set $\mathtt{E} \subseteq \mathtt{V} \times \mathtt{V}$ of m links between nodes. For every edge, $e_k : (v_i, v_j)$, there is a stepwise-constant function of available bandwidth $x^{e_k}(t)$ where t is a variable in time domain T. The available bandwidth $x^{e_k}(t)$ in G_T is time-dependent, nonnegative, and bounded by an upper limit u^{e_k}, where u^{e_k} is the maximum bandwidth available for allocation in e_k; such that, $0 \leq x^{e_k}(t) \leq u^{e_k}$ for any instance of time in T.

When an advance reservation $R_i = (v_i^s, v_i^d, \mathtt{M}_i, t_i^s, t_i^e)$ is found between start time t_i^s and end time t_i^e, we setup a path δ_i from source node v_i^s to destination node v_i^d that can satisfy the allocation of the requested bandwidth $\mathtt{M}_i$. For every edge along the path $\delta_i : (e_{k^i}, e_{k^j}, \ldots)$, we allocate $\mathtt{M}_i$ amount of bandwidth for the future use of reservation R_i. The available bandwidth x^{e_k} of each edge in δ_i is updated in the topology graph G_T for the time period of $[t_i^s, t_i^e]$.

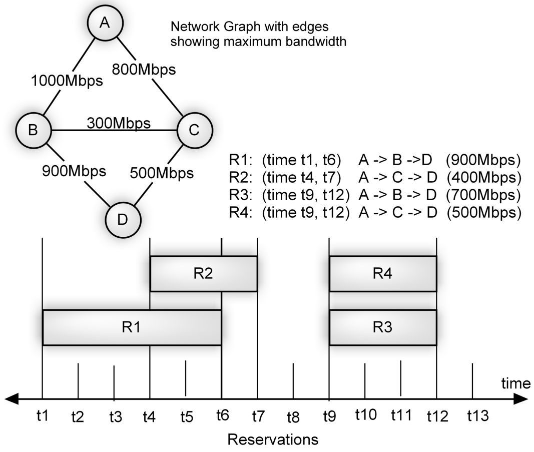

A
Network Graph with edges
showing maximum bandwidth
1000Mbps
800Mbps
B
300Mbps
C
900Mbps
500Mbps
D
R1: (time t1, t6) A -> B ->D (900Mbps)
R2: (time t4, t7) A -> C -> D (400Mbps)
R3: (time t9, t12) A -> B -> D (700Mbps)
R4: (time t9, t12) A -> C -> D (500Mbps)
R2
R4
R1
R3
time
t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13
Reservations

Figure 1: Example for Advance Network Reservation

The example in Figure 1 is given to clarify the underlying mechanism in advance network reservation. The top part shows maximum bandwidth of each edge of the network graph. At some point of time, assume that there are four reservations confirmed and active in the system; $R_1 = \{A \rightarrow B \rightarrow D, 900Mbps, t_1, t_6\}$, $R_2 = \{A \rightarrow C \rightarrow D, 400Mbps, t_4, t_7\}$, $R_3 = \{A \rightarrow B \rightarrow D, 700Mbps, t_9, t_{12}\}$, $R_4 = \{A \rightarrow C \rightarrow D, 500Mbps, t_9, t_{12}\}$. Thus, the first reservation, R_1, is for 900Mbps between t_1 and t_6 from source A to destination D. The system calculated a path based on engineering metric satisfying requested allocation, and allocated bandwidth over $A \rightarrow B \rightarrow D$. R_2, R_3, and R_4 are interpreted similarly. The bottom part of Figure 1 shows the extent over time of these four reservations. Figure 2 shows the available bandwidth and allocated bandwidth in link $A \rightarrow B$ over time.

The first graph in Figure 3 represents the status in $[t_1, t_4]$ and the second represents the status in $[t_4, t_6]$. Every link in Figure 3 shows available, allocated, and total capacity values of bandwidth, in the order given. We can confirm a new reservation request from source A to

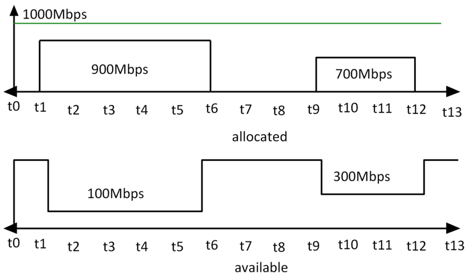


Figure 2: Available bandwidth and allocated bandwidth in link $A \rightarrow B$ over time

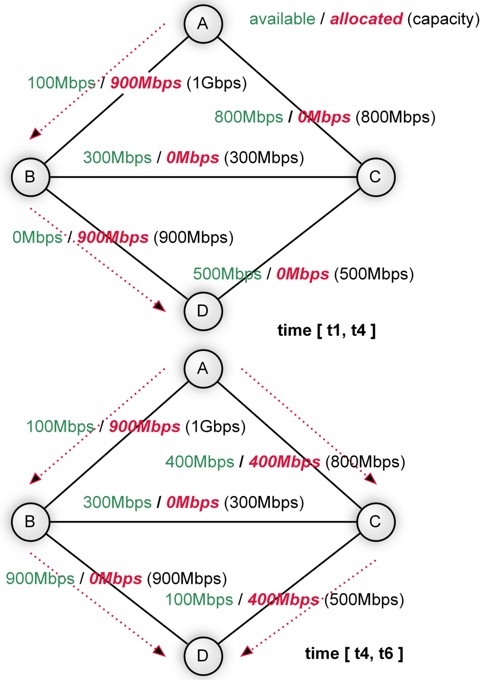

available / allocated (capacity)
A
100Mbps / 900Mbps (1Gbps)
800Mbps / 0Mbps (800Mbps)
B
300Mbps / 0Mbps (300Mbps)
C
0Mbps / 900Mbps (900Mbps)
500Mbps / 0Mbps (500Mbps)
D
time [t1, t4]
A
100Mbps / 900Mbps (1Gbps)
400Mbps / 400Mbps (800Mbps)
B
300Mbps / 0Mbps (300Mbps)
C
900Mbps / 0Mbps (900Mbps)
100Mbps / 400Mbps (500Mbps)
D
time [t4, t6]

Figure 3: Network Flow in specific time periods($[t_1, t_4]$, $[t_4, t_6]$)

destination D with start time t_1 and end time t_4, with 500Mbps guaranteed bandwidth, since we can allocate path $A \rightarrow C \rightarrow D$ for the $[t_1, t_4]$ time period. We can only allocate 100Mbps between t_4 and t_6 over $A \rightarrow C \rightarrow D$. We can allocate 300Mbps with start time t_4 and end time t_6 over $A \rightarrow C \rightarrow B \rightarrow D$. There is a possibility to send 300Mbps over $A \rightarrow C$ in $[t_1, t_4]$ and 300Mbps over $C \rightarrow B \rightarrow D$ in $[t_4, t_6]$. However, we cannot split the allocation among separate time periods. Therefore, the maximum available between $t1$ and $t6$ from A to D is 100Mbps because the maximum amount of bandwidth we can get during the entire period of $[t_1, t_6]$ is 100Mbps. Additionally, we cannot split the bandwidth among separate paths. For example, there is an opportunity to send 500Mbps from A to C. The maximum flow from A to C is 500Mbps in $[t_4, t_6]$, 100Mbps over $A \rightarrow B \rightarrow C$ and 400Mbps over $A \rightarrow C$. However, we make a reservation for a specific path. Therefore, the maximum amount of bandwidth we can allocate for a single reservation from A to C is 400Mbps in time period $[t_4, t_6]$.

A service request is defined as $S_i = (v_i^s, v_i^d, \mathtt{M}_i^{max}, D_i, t_i^E, t_i^L)$; with total size of data D_i to be sent from v_i^s to v_i^d, and a period of time between earliest start time t_i^E and latest end time t^L such that, this data transfer need to be accomplished in this given time interval. $\mathtt{M}_i^{max}$ is the maximum bandwidth provided by the user based on constraints of storage systems at both ends. If there exists bandwidth between v_i^s and v_i^d within the time constraints in G_T, a new reservation $R_{earliest}$ for earliest completion time or $R_{shortest}$ for shortest transfer duration is generated. Consequently, we create a reservation $R_j = (v_i^s, v_i^d, \mathtt{M}_j, t_j^s, t_j^e)$ where $\mathtt{M}_j \leq \mathtt{M}_i^{max}$ and $t_i^E \leq t_j^s < t_j^e \leq t_i^L$. We also compute a path δ_j satisfying reservation R_j.

In order to satisfy the given criteria, the amount of bandwidth allocation $\mathtt{M}_j$ and the time interval $[t_j^s, t_j^e]$ need to be sufficient to transmit the data volume D_i using the path δ_j allocated for reservation R_j. We can simply say $D_i = \mathtt{M}_j \times d$ where d is the duration between start time t_j^s and end time t_j^e. Note that our focus is to find possible reservation options according to given user criteria.

$R_{shortest}$ has the minimum duration $d = |t^s, t^e|$ among all other possible reservations satisfying S_i. The objective for earliest completion time is to select a reservation R_j satisfying the criteria given in S_i which has the earliest end time t^e. We would favor a reservation with a shorter duration if there are more than one possible reservations completing at the same earliest time. For reservation $R_{earliest}$, $\forall R_j$ satisfying S_i : $t_{earliest}^e \leq t_j^e$, and $\forall R_j$ with $t_j^e = t_{earliest}^e$: $t_{earliest}^s \geq t_j^s$.

5.1 Search Interval between Earliest-Start and Latest-End times

The outline of our approach is as follows. We divide the given search interval into time steps. The search interval $[t_i^E, t_i^L]$ is the time period between earliest start time t_i^E and latest end time t_i^L in which the data needs to be transmitted. A **time step** represents the longest duration of time in which we have a stable discrete status in terms of available bandwidth over the links. A time period $[t_i, t_j]$ is considered as a time step if $\forall e_k \in G_T : x^{e_k}(t) = c_k$ where $t_i \leq t \leq t_j$, and c_k is a constant. We obtain a static directed graph that keeps information about the available bandwidth status for every link. This information is updated on-the-fly every time a reservation request is committed and stored for further processing during the path calculation phase. A snapshot graph of G_T in time step $ts(t_i, t_j)$ is defined as $G(ts_i)$, with the same vertex set and same edge set. For every edge $e_k : (v_i, v_j)$ in $ts(t_i, t_j)$, the available bandwidth $x^{e_k} = c_k$ stands for the value of $x^{e_k}(t)$ in G_T between t_i and t_j in time step $ts(t_i, t_j)$. This help us discretize the dynamic graph and apply known graph algorithms efficiently.

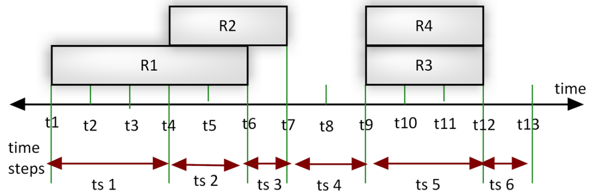


Figure 4: Time steps between t_1 and t_{13}

Figure 4 shows time steps between t_1 and t_{13}, for the example given in Figure 1 with four committed reservations. We have six time steps: $ts_1(t_1, t_4)$, $ts_2(t_4, t_6)$, $ts_3(t_6, t_7)$, $ts_4(t_7, t_9)$, $ts_5(t_9, t_{12})$, $ts_6(t_{12}, t_{13})$. Every time step corresponds to a static snapshot of the network topology. Figure 5 shows $G(ts_1)$, $G(ts_2)$, $G(ts_3)$, $G(ts_4)$, $G(ts_5)$, and $G(ts_6)$, where every link is labeled with the available bandwidth.

We analyze the search interval $[t^E, t^L]$ with a set of consecutive time steps covering the entire period. The set of confirmed reservations in the system characterize time steps since they change the available bandwidth values in the network topology. If two reservations partially overlap in terms of time period, they split the total period of time into either two or three time steps. If they do not overlap, they split into three time steps. In other words, the number of time steps in the search interval is bounded by the number of committed reservations within the given period $[t^E, t^L]$. If there are r committed reservations falling into the period, there can be a maximum of $2r + 1$ different time steps in the worst-case. Figure 4 shows the general idea behind time steps and reservations.

The next step is to traverse these time steps to check whether we can find a reservation satisfying the given criteria. For the example given in Figures 4 and 5, first ts_1, and then ts_2 will be examined; then, if both cannot satisfy the request, time window $tw(t_1, t_6)$, a combination of ts_1 and ts_2, will be examined. A **time window** consists of subsequent time steps. tw_k is a time window which corresponds to the time period in ts_k. $tw_{k_1-k_2}$ is a time window including all time steps between ts_{k_1} and ts_{k_2}. If there are s time steps in a given search interval, there are $(s \times (s+1))/2$ time windows since time windows are subsequent combinations of time steps.

We search through these time windows in a sequential order to check whether we can satisfy the requested allocation in that time window. For a bandwidth allocation with the shortest duration, we can sort time windows according to their length, and start checking with the smallest one. For a bandwidth allocation with the earliest completion time, we can benefit from a specific search pattern. The search pattern for earliest completion time in the given example will be as follows: $tw_1, tw_2, tw_{1-2}, tw_3, tw_{2-3}, tw_{1-3}, tw_4, tw_{3-4}$,$tw_{2-4}$, tw_{1-4},.... The algorithm will stop searching when it finds a time window satisfying the given criteria. In most cases, we do not need to check all possible time windows. In the worst-case, we may require to search all time windows, which consists of $(s \times (s + 1))/2$ searches, where s is the number of time steps.

5.2 Examining Time Windows to Find Possible Reservations

While checking a time window to verify whether it can satisfy the request, we first look at the total duration of the time window. We know the max bandwidth $\mathtt{M}^{max}$ user can support, and the total size of data D. Therefore, we first determine the duration of a time window and simply ensure whether this time window is large enough to satisfy the user request. The length of a time window $d_{tw} = |tw_{k1-k2}|$ should be larger than the minimum amount of time, $D/\mathtt{M}^{max}$, required to transmit data if $\mathtt{M}^{max}$ bandwidth can be allocated.

Then, we calculate the maximum bandwidth available from source v^s to destination v_d in time window tw. We use max-bandwidth path algorithm over static snapshot graph $G(tw)$. $G(tw)$ can easily be computed using snapshots of time steps that form this time window. $G(tw_k) = G(ts_k)$, and $G(tw_{k_1-k_2}) = G(ts_{k_1}) \circ G(ts_{k_1+1}) \circ G(ts_{k_1+2}) \ldots \circ G(ts_{k_2})$, where $\circ$ is a newly defined operator that intersects static snapshot graphs. $G_1 \circ G_2$ forms a new graph with the same vertex and edge set as in G_1 and G_2. For each edge e_k, the available bandwidth is the minimum of x^{e_k} both in G_1 and G_2. Thus, $\forall e_k \in G_1 \circ G_2$: $x^{e_k} = min\{x_1^{e_k}, x_2^{e_k}\}$, where $x_1^{e_k}$ is the available bandwidth of e_k in G_1 and $x_2^{e_k}$ is the available bandwidth of e_k in G_2. This property makes the process easy, since we only need to store one graph snapshot for each starting time window; for example, to obtain $G(tw_{1-3})$, we only need $G(tw_{1-2})$ and $G(tw_3)$, $G(tw_{1-3}) = G(tw_{1-2}) \circ G(tw_3)$.

Figure 6 shows static snapshot graphs for time windows tw_{1-2}, tw_{3-4}, tw_{5-6}, and tw_{1-6}. $G(tw_{1-2}) = G(ts_1) \circ G(ts_2)$, $G(tw_{3-4}) = G(ts_3) \circ G(ts_4)$, $G(tw_{5-6}) = G(ts_5) \circ G(ts_6)$, and $G(tw_{1-6}) = G(tw_{1-2}) \circ G(tw_{3-4}) \circ G(tw_{5-6})$. R_1 and R_2 are active in time interval $[t_1, t_6]$, so links associated with both R_1 and R_2 are updated in $G(tw_{1-2})$. Only R_2 is active in time interval $[t_6, t_9]$, so links associated with R_2 are updated in $G(tw_{3-4})$.

While exploring a time window, a max-bandwidth path δ is calculated in $G(tw)$ in which $\mu_{tw}(v^s, v^d)$ is the maximum amount of bandwidth we can allocate in time window tw. $d_{tw} \times \mu_{tw}$ simply gives the amount of data that can be transmitted if a reservation is made in time window tw, where d_{tw} is the length of the time window. A time window $tw(t_i, t_j)$ is selected if it can provide enough resources to satisfy the user criteria. For such a time window, we consider the length of usable period of time overlapping with $[t^E, t^L]$, time period between earliest start time and latest end time; $d'_{tw} = |max\{t_i, t^E\}, min\{t_j, t^L\}|$ is the maximum duration

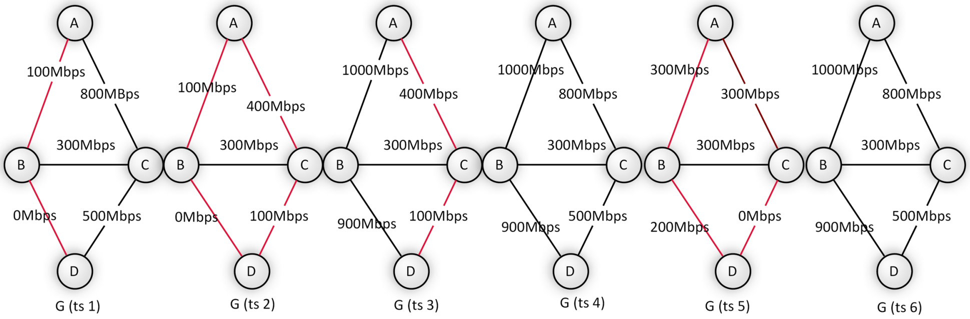


Figure 5: Static Graphs for time steps ts_1, ts_2, ts_3, ts_4, ts_5, ts_6

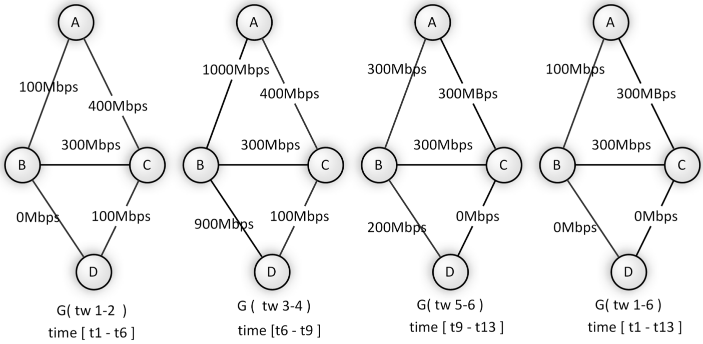
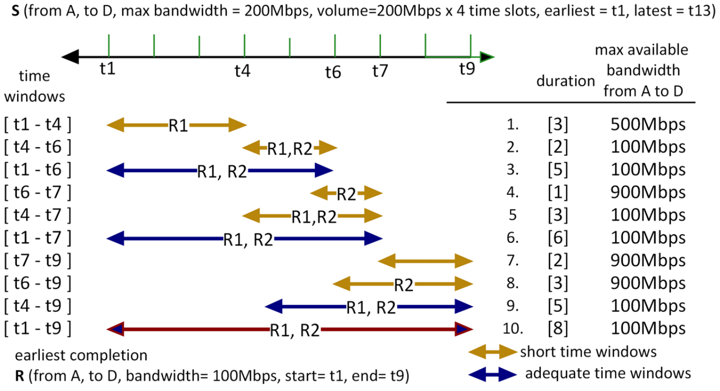


Figure 6: Static Graphs for time windows tw_{1-2}, tw_{3-4}, tw_{5-6}, and tw_{1-6}

we can use to make a reservation. $\mu_{tw} = \mu_{tw}(v^s, v^d)$ is the maximum amount of bandwidth we can allocate from source to destination. Note that we need to consider the amount of bandwidth we can use which is also limited by the maximum set by the user, $\mu'_{tw} = min\{\mu_{tw}, \mathtt{M}^{max}\}$. Therefore, the product $\mu'_{tw} \times d'_{tw}$ should be greater than the requested volume size D.

When a satisfactory window is found, we generate a reservation $R = (v^s, v^d, \mathtt{M}, t^s, t^e)$ and a path from source to destination to be used for this reservation in the network. The start/end times and M are calculated based on the given user criteria and available resources in the time window. A straightforward strategy to generate a reservation when a time window tw is selected to satisfy the user criteria is as follow: $t^s = max\{t_i, t^E\}$, $\mathtt{M} = min\{\mu_{tw}, \mathtt{M}^{max}\}$, and $t^e = t^s + |D/\mathtt{M}|$.

Figure 7 shows the search pattern to find a reservation for the earliest completion time, for the example given in Figure 1. Assume that we have a service request $S = (A, D, 200Mbps, 200 \times 4t, t_1, t_{13})$, and we want to find a reservation satisfying the given criteria. Time window $tw(t_1, t_4)$ with length $3t$, and time window $tw(t_4, t_6)$ with length $2t$, are short in duration to conform to the requirements of this request. The maximum bandwidth allowed is 200Mbps, so we need at least a time window with length $4t$. $tw(t_1, t_6)$ satisfies the time requirement, so we proceed and calculate the maximum bandwidth available in $G(tw(t_1, t_6))$. The maximum bandwidth we can reserve from A to D between t_1 and t_6 is 100Mbps. Total size of data we can transfer is $100 \times 5t$. Therefore, $tw(t_1, t_6)$ can not satisfy the bandwidth requirement. We keep searching through time windows until we find $tw(t_1, t_9)$ which satisfies both time and bandwidth requirements. Time window $tw(t_1, t_9)$ is selected for the earliest completion time. We generate $R_{earliest} = (A, D, 100Mbps, t_1, t_9)$ with start time t_1 and end time t_9.


S (from A, to D, max bandwidth = 200Mbps, volume=200Mbps x 4 time slots, earliest = t1, latest = t13)

time windows			duration	max available bandwidth from A to D
[t1 - t4]	R1	1.	[3]	500Mbps
[t4 - t6]	R1,R2	2.	[2]	100Mbps
[t1 - t6]	R1, R2	3.	[5]	100Mbps
[t6 - t7]	R2	4.	[1]	900Mbps
[t4 - t7]	R1,R2	5	[3]	100Mbps
[t1 - t7]	R1, R2	6.	[6]	100Mbps
[t7 - t9]		7.	[2]	900Mbps
[t6 - t9]	R2	8.	[3]	900Mbps
[t4 - t9]	R1, R2	9.	[5]	100Mbps
[t1 - t9]	R1, R2	10.	[8]	100Mbps

earliest completion
R (from A, to D, bandwidth= 100Mbps, start= t1, end= t9)

Figure 7: Example for earliest completion

If we want to find a reservation for the shortest transfer duration, we need to continue searching until we cover the entire interval between t_1 and t_{13}. As shown in Figure 8, $tw(t_9, t_{12})$, $tw(t_7, t_{12})$, $tw(t_6, t_{12})$, $tw(t_4, t_{12})$, $tw(t_1, t_{12})$, $tw(t_{12}, t_{13})$, and $tw(t_9, t_{13})$ ... are searched next. Time window $tw(t_9, t_{13})$ satisfies the given bandwidth and time requirements. All other time windows coming after this in the search pattern, are longer in terms of duration. Therefore, $tw(t_9, t_{13})$ gives the reservation $R_{shortest} = (A, D, 200Mbps, t_9, t_{13})$ with shortest duration.

If the total volume of data was $175 \times 4t$, $S = (A, D, 200Mbps, 175 \times 4t, t_1, t_{13})$, then we would obtain reservation $R_{shortest} = (A, D, 200Mbps, t_9, t_{12.5})$ for shortest duration and $R_{earliest} = (A, D, 100Mbps, t_1, t_8)$ for earliest completion, as also shown in Figure 8.

The pseudo-code for finding the desired reservation is

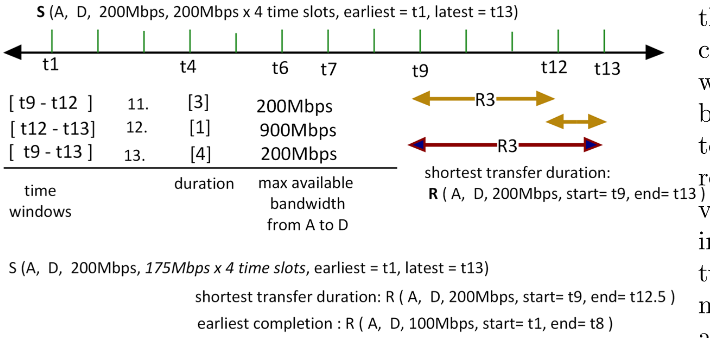

S (A, D, 200Mbps, 200Mbps x 4 time slots, earliest = t1, latest = t13)
t1
t4
t6
t7
t9
t12
t13
[t9 - t12]
11.
[3]
200Mbps
[t12 - t13]
12.
[1]
900Mbps
[t9 - t13]
13.
[4]
200Mbps
time windows
duration
max available bandwidth from A to D
R3
R3
shortest transfer duration:
R (A, D, 200Mbps, start= t9, end= t13)
S (A, D, 200Mbps, 175Mbps x 4 time slots, earliest = t1, latest = t13)
shortest transfer duration: R (A, D, 200Mbps, start= t9, end= t12.5)
earliest completion : R (A, D, 100Mbps, start= t1, end= t8)

Figure 8: Example for shortest transfer duration and earliest completion

given in Algorithm 1.

Algorithm 1: A sample search pattern to find a network reservation with earliest completion or shortest duration

```
Input: A request
       with user constraints, $S = (v^s, v^d, \mathtt{M}^{max}, D, t^E, t^L)$
Output: A reservation
        $R$ for earliest completion or shortest duration

Get
the set of time steps in the search interval $\{ts_1, ts_2, \ldots, ts_n\}$ ;
for $i = 1$ to $n$ do
    for $j = i$ to 1 do
        Get time window $tw = tw_{j-i}$
        which contains all time steps between $ts_j$ and $ts_i$;
        if the given criteria
        can fit into the time window $tw = ts_j \ldots ts_i$ then
            Obtain
            static snapshot graph $G(tw)$ for time window $tw$;
            Calculate
            max-bandwidth $\mu_{tw}$ from source to destination;
            if we can satisfy
            request in time window tw (Examine $\mu_{tw}$) then
                select $tw$ ;
        if goal
        is to find a reservation with Earliest completion then
            if there is any selected time window tw then
                Get $tw$ with
                shortest duration to satisfy the given request;
                Generate a Reservation
                and a Path, Return for earliest completion;
if goal is to find a reservation with Shortest duration then
    if there is any selected time window tw then
        Get
        $tw$ with shortest duration to satisfy the given request;
        Generate a
        Reservation and a Path, Return for shortest duration;
Return: No reservation found (no
possible option available satisfying the given user constraints);
```

5.3 Evaluation of the Proposed Algorithm

The max-bandwidth path algorithm is bounded by $O(n^2)$, where n is the number of nodes in the topology graph. In the worst-case, we may require to search all time windows, $(s \times (s+1))/2$, where s is the number of time steps. If there are r committed reservations in that period, there can be a maximum of $2r+1$ different time steps in the worst-case. Overall, the worst-case complexity is bounded by $O(r^2n^2)$. However, r is relatively very small compared to the number of nodes n, in the topology. Bandwidth reservation is used for large-scale data transfers and it is very unlikely to have thousands of committed reservations in a given time period. Also, the path calculation from two end-points does not span to all nodes in a real network; therefore, we can trim the topology graph and perform calculation on a reduced data set while calculating path from source to destination. Moreover, time windows that are too short in duration to transmit the requested amount of data are eliminated from consideration beforehand. Max bandwidth and shortest path algorithms are quite efficient and the search process over time windows is scalable and practical, considering that the number of reservations in practice is limited. Furthermore, there are usually less than a hundred node in a typical network topology like ESnet. We have tested the performance of the algorithm by simulating very large graphs (with 10K nodes) and have observed that the computation time is usually in the order of seconds.

6 Implementation Details

The network topology graph in OSCARS includes routers, ports, and unidirectional links between two ports, $G =< n_{router}, v_{port}, e_{link} >$. Each router has a list of attached ports, $n_{router} =< v^1_{port}, v^2_{port}, .. >$, and each port has a maximum available bandwidth for advance allocation. A link connects two ports in one direction, $e^1_{link} =< v^1_{port}, v^2_{port} >$, $e^2_{link} =< v^2_{port}, v^1_{port} >$; such that, a separate reservation request is established for each direction. Every port in a router has a maximum bandwidth value available for reservation. Furthermore, engineering metric is assigned to each port by network system administrators. A link provides communication from one router towards another one over two in/out ports in each. In general, the engineering metric represents the preferred routing pattern; it is related to the latency of the link. The link with the smaller value is favored over another with a higher value. Based on this information, each link has an absolute value of maximum bandwidth available for reservation and an absolute engineering metric which is used later to establish a path and compute the routing pattern from two end-points, $e_{link}(v^1_{port}, v^2_{port}) =< M_{linkBandwidth}, m_{engMetric} >$.

The current web service interface in OSCARS enables users to request a fixed amount of bandwidth for a time period between two end-points in the network. The source and destination end-points are usually the host/IP names of the client machines; they are converted to the corresponding router addresses in the network topology. The reservation system needs to ensure availability of the requested bandwidth from the source to the destination for the requested time interval. Therefore, it needs to have the topology information and current active reservations in the system. In other words, we need to have knowledge about the network structure and the committed bandwidth guaranteed paths in order to examine whether a reservation request can be satisfied.

We have applied our algorithm as a new service, called Flexible Network Reservation Service, in which we find out and return alternative reservation options to the user. In the rest of this section, we state some crucial implementation issues that we came across during designing and developing the libraries for this new service. We provide practical methods to traverse time windows in the given search space. We have also developed a modular organization in terms of software components to evaluate bandwidth availability and possible reservations options in an effective manner. We need well-organized and re-usable data structures in order to minimize the computation time and eliminate unnecessary data duplication during the retrieval of snapshot network structures, which are used to compute resource availability between source and destination end-points in every time window. We present effective methodologies since we might need to search all time windows that fall into the given search interval in the worst-case. For each time window, we need to evaluate resource availability. In order to eliminate duplicate information, we only store available bandwidth values in each step. The rest of the network topology does not vary over time, but time dependent bandwidth availability needs to be queried to calculate a reservation path.

6.1 Deployment and Integration

OSCARS's reservation system enables users to make reservations and to query their currently active and committed reservation requests. For administrative and security purposes, we provide minimum amount of information to the user about the current load and future allocations in the system. The new service, called Flexible Network Reservation Service, enables us to return alternative reservation options to the users. We require topology information and committed reservations in the search interval, between earliest start time and latest completion time, in order to calculate time steps, examine time windows, and find out possible reservation options. These include current load in links, active advance reservations, capacity of links and available bandwidth in each link. It is impractical to expose the topology information and the state of network and reservations to users for administrative and security concerns. Since we need administrative interface to query topology and reservation information, we have implemented this new service, Flexible Reservation Service, inside OSCARS.

The Flexible Reservation Service acts as a suggestion agent and generates a reservation request based on user constraints. We have developed a new interface where users submit their constraints and the system suggests a reservation option if it can find one in the given search interval. As has been described in the previous section, the algorithm is able to locate all possible reservation options. However, we limit the choices provided to the user, and instead require users to specify whether they desire to make a reservation based on earliest completion time or shortest transfer duration. If the system can find a reservation according to the given user criteria, it notifies the user; and, if confirmed by the user, it makes the reservation by allocating resources over the path found. Figure 9 shows a diagram of the user interfaces and the overall structure for integration and deployment in OSCARS.

6.2 Examining Time Windows

An important difficulty in designing an advance reservation system is to find an appropriate data structure to keep bandwidth availability in a time-dependent network. The common approach presented in the literature is to divide the entire time period into time slots and store available bandwidth over a link for each time slot. When a new reservation is committed and added to the system, we proceed and update the bandwidth availability and time slot information for every link on the allocated path. Using such a technique, in which we accumulate resource availability for time slots for every link in a network, enables straightforward evaluation since all resource availability has been pre-computed. On the other hand, the total data size increases dramatically especially for a large network with many reservations committed. There have been several studies analyzing data structures for network routing with advance reservation [5, 23, 22]. Further, we need an effective methodology which we can also benefit in calculating static snapshot graphs $G(t_w)$, during maximum bandwidth calculation and time window evaluation. Since we already have reservation information, which includes path information and allocated bandwidth value, we can automatically generate the bandwidth availability for all links for a specific time period if we know the reservations that are active in the time window we are evaluating. Simply, we deduct allocated bandwidth

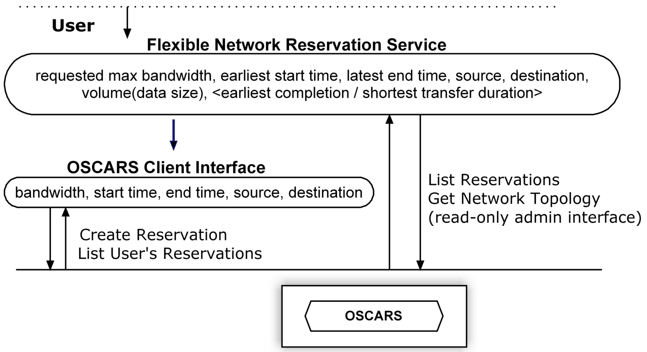

User
Flexible Network Reservation Service
requested max bandwidth, earliest start time, latest end time, source, destination, volume(data size), <earliest completion / shortest transfer duration>
OSCARS Client Interface
bandwidth, start time, end time, source, destination
Create Reservation
List User's Reservations
List Reservations
Get Network Topology
(read-only admin interface)
OSCARS

Figure 9: Integration and deployment in OSCARS

of a reservation from the available bandwidth of a link if the path for this reservation uses the link we consider.

We present a specialized linked-list data structure that holds time steps and a set of active reservation identifiers associated for each time step. This information is updated on the fly. When a new reservation is committed, the data structure is updated and new time steps are added automatically if necessary. If a reservation is canceled, its identifier is removed from the set of reservations that belongs to time steps the canceled reservation spans over. The main purpose of this data structure is to query time windows quickly and retrieve a list of active reservations in time windows. We only need time steps that fall into the given search interval. Time steps are indexed for faster operations and a set of reservations is returned for each time window. Figure 10 gives a brief overview of this data structure and shows how it is updated when reservations are added.

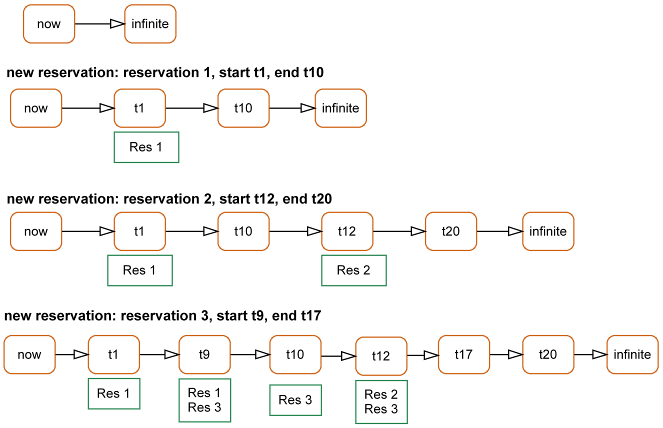

now
infinite
new reservation: reservation 1, start t1, end t10
now
t1
t10
infinite
Res 1
new reservation: reservation 2, start t12, end t20
now
t1
t10
t12
t20
infinite
Res 1
Res 2
new reservation: reservation 3, start t9, end t17
now
t1
t9
t10
t12
t17
t20
infinite
Res 1
Res 1
Res 3
Res 3
Res 2
Res 3

Figure 10: List of time steps and associated reservations

6.3 Maximum Hop Count

Next, we discuss another important parameter, hop count. Although we might find a path satisfying a given user criteria, we do not want to reserve a path passing through too many routers in the network. Hop count is crucial in terms of network engineering, especially in network reservations where we need to configure every router over the path to setup a secure circuit for guaranteed bandwidth. It is more costly and less desirable to arrange a path along many routers. The maximum allowed hop count is usually set by the system. For our case in OSCARS, we configure the system to allow a maximum of 10 hops for a network reservation path. In addition to this default value, we also permit users to specify a maximum hop count value for the path they would like to reserve between source and destination nodes.

The maximum hop count parameter enables us to optimize maximum bandwidth calculation. We can eliminate network nodes which are not accessible with the given maximum hop count. In other words, we should not consider a path which would be discarded due to the maximum hop count parameter. We take advantage of using the maximum hop count parameter, and we prepare a **reachable set** at first, before traversing time windows. We examine bandwidth availability and compute a path by considering nodes in this reachable set. Calculating a reachable set based on a maximum hop count parameter has the same asymptotic complexity with maximum bandwidth and shortest path algorithms. The QoS constraint in finding a reachable set with maximum number of hops from a source node is additive; and, the algorithm stops traversing over nodes if the hop count is more than the maximum. The main difference is that we return a set of nodes which are selected and traversed, instead of finding a path. Applying the maximum bandwidth algorithm to the reachable set is significantly more efficient, especially in a sparse graph structure since many of the nodes will not be in the reachable set, so will not be considered.

Another improvement is a modification of the maximum bandwidth algorithm that takes advantage of the fact that the number of ports is usually much larger than the number of routers/nodes. For example, ESnet's basic network has 75 routers and 587 ports. Therefore, we use a specialized maximum bandwidth path algorithm in which we pass through a node (router) in each iteration instead of iterating over ports. While visiting a node, we select an unvisited node that is connected over a port which provides maximum available bandwidth. A visited node will not be visited again while traversing the network graph for maximum bandwidth calculation from source node to destination node.

6.4 Experiments

We have developed a simulator to experiment our approach by generating large random graphs. The performance of the proposed algorithm depends not only on the number of nodes in the network, but the number of currently active reservations in the given search interval directly affects the number of time windows we might need to evaluate in order to find a reservation. In order to test the performance, we have generated random requests asking for a reservation within a 200hrs time interval. User parameters such as data volume, earliest start time and latest end time are all set randomly. Those requests ask for a reservation for earliest completion time. If a reservation is found, we allocate the path and admit the reservation. If there are no resources available for the current request, we continue and generate a new random request. We test the Flexible Network Reservation Service by gradually iterating until 1000 reservations are committed. Note that it is very unlikely in real life to have thousands of committed reservations in a short period. We ignored the maximum hop count parameter and set it to infinite in order to evaluate performance in large and complex system. As can

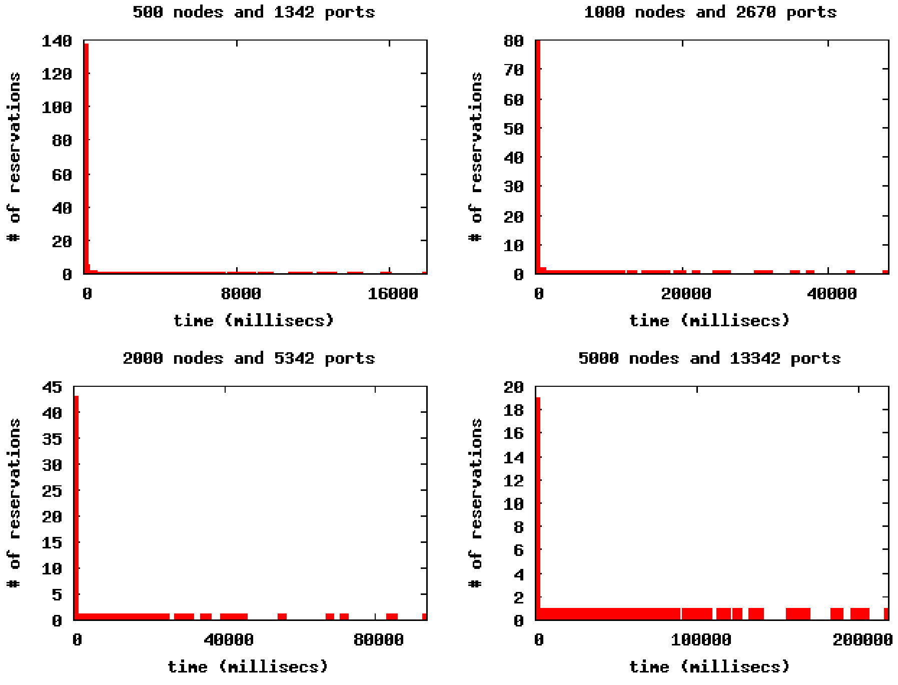


Figure 11: Histogram for execution time (milliseconds): search performance to find a reservation in a network with 1000 reservations applied

be seen in Figure 11, most of the reservation requests are completed by finding a reservation in less than a second. These test are conducted on a workstation with 2.4GHz Intel CPU and 8G RAM. Our software is implemented in JAVA and tests are performed with JVM version 1.6.0_11. Even for very large graphs with many reservations committed, we were able to process and search all related time windows to find a reservation in a timely manner.

7 Summary and Future Work

In this paper, we have studied advance network reservation and provisioning for on-demand high performance data transfers. Advance reservation systems enables users to allocate a fixed amount of bandwidth for a time period between two end-points in a network. If the requested reservation cannot be granted, no further suggestion is returned back to the user. In order to enhance advance network reservation systems, we have developed a new methodology in which users submit constraints and the system suggests possible reservation options satisfying requirements. We have reported a polynomial-time algorithm, where the user specifies the total volume that needs to be transferred, a maximum bandwidth that he/she can use, and a desired time period within which the transfer should be done. The proposed algorithm can find alternate allocation possibilities, including earliest time for completion, or shortest transfer duration - leaving the choice to the user. The proposed algorithm is quite practical when applied to large networks with thousands of routers and links. We have implemented our algorithm as a new service extending the current underlying mechanisms. In order to take advantage of the available network bandwidth, clients need to provision other resources such as storage capacity and bandwidth from/to the storage system. According to the storage allocation policy and available storage bandwidth in the source and destination ends, users may need to adjust the network reservation requests. Our future work includes integration of the algorithm into the future version of ESnet OSCARS, and the coordination of storage and network resource provisioning.

Acknowledgments

We would like to thank David Robertson and Mary Thompson from ESnet for their generous help in OSCARS client interface during the development and testing of the Flexible Network Reservation Service.

This work was funded by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract no. DE-AC02-05CH11231.

References

[1] Energy Sciences Network. http://www.es.net.

[2] ESG: Earth System Grid. www.earthsystemgrid.org.

[3] OSCARS: On-demand secure circuits and advance reservation system. www.es.net/oscars.

[4] TeraPaths: Configuring end-to-end virtual network paths with QoS guarantees. https://www.racf.bnl.gov/terapaths.

[5] Lars-Olof Burchard. Analysis of data structures for admission control of advance reservation requests. *IEEE Trans. on Knowl. and Data Eng.*, 17(3):413–424, 2005.

[6] Lars-Olof Burchard. Networks with advance reservations: Applications, architecture, and performance. *J. Netw. Syst. Manage.*, 13(4):429–449, 2005.

[7] Ismail Chabini. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Records*, 1645:170–175, 1998.

[8] William C. Cheng, Cheng fu Chou, Leana Golubchik, Samir Khuller, and Yung-Chun (Justin) Wan. Large-scale data collection: a coordinated approach. In *in Proceedings of IEEE INFOCOM*, pages 218–228, 2003.

[9] Bolin Ding, Jeffrey Xu Yu, and Lu Qin. Finding time-dependent shortest paths over large graphs. In *EDBT '08: Proceedings of the 11th international conference on Extending database technology*, pages 205–216, New York, NY, USA, 2008. ACM.

[10] S. Ganguly, A. Sen, G. Xue, B. Hao, and B. Shen. Optimal routing for fast transfer of bulk data files in time-varying networks. IEEE Int. Conf. on Communications, 2008.

[11] R. Guerin and A. Orda. Networks with advance reservations: the routing perspective. IEEE INFOCOMM, 2000.

[12] Chin Guok, D. Robertson, M. Thompson, J. Lee, B. Tierney, and W. Johnston. Intra and interdomain circuit provisioning using the oscars reservation system. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–8, Oct. 2006.

[13] Eun-Sung Jung, Yan Li, Sanjay Ranka, and Sartaj Sahni. An evaluation of in-advance bandwidth scheduling algorithms for connection-oriented networks. In *ISPAN '08: Proceedings of the The International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 133–138, Washington, DC, USA, 2008. IEEE Computer Society.

[14] Zhaoming Li, Qiang Song, and Ibrahim Habib. Cheetah virtual label switching router for dynamic provisioning in ip optical networks. *Optical Switching and Networking*, 5(2-3):139–149, 2008. Advances in IP-Optical Networking for IP Quad-play Traffic and Services.

[15] Yunyue Lin and Qishi Wu. On design of bandwidth scheduling algorithms for multiple data transfers in dedicated networks. In *ANCS '08: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 151–160, New York, NY, USA, 2008. ACM.

[16] Ariel Orda and Raphael Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *J. ACM*, 37(3):607–625, 1990.

[17] Marek Piotrów. A note on constructing binary heaps with periodic networks. *Inf. Process. Lett.*, 83(3):129–134, 2002.

[18] N. S. V. Rao, W. R. Wing, S. M. Carter, and Q. Wu. Ultrascience net: network testbed for large-scale science applications. *Communications Magazine, IEEE*, 43(11):S12–S17, 2005.

[19] N.S.V. Rao, Qishi Wu, Song Ding, S.M. Carter, W.R. Wing, A. Banerjee, D. Ghosal, and B. Mukherjee. Control plane for advance bandwidth scheduling in ultra high-speed networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–5, April 2006.

[20] Sartaj Sahni, Nageshwara Rao, Sanjay Ranka, Yan Li, Eun-Sung Jung, and Nara Kamath. Bandwidth scheduling and path computation algorithms for connection-oriented networks. In *ICN '07: Proceedings of the Sixth International Conference on Networking*, page 47, Washington, DC, USA, 2007. IEEE Computer Society.

[21] M. Veeraraghavan, H. Lee, E.K.P. Chong, and H. Li. A varying-bandwidth list scheduling heuristic for file transfers. In *Communications, 2004 IEEE International Conference on*, volume 2, pages 1050–1054 Vol.2, June 2004.

[22] Tao Wang and Jianer Chen. Bandwidth tree - a data structure for routing in networks with advanced reservations. In *PCC '02: Proceedings of the Performance, Computing, and Communications Conference, 2002. on 21st IEEE International*, pages 37–44, Washington, DC, USA, 2002. IEEE Computer Society.

[23] Qing Xiong, Chanle Wu, Jianbing Xing, Libing Wu, and Huyin Zhang. A linked-list data structure for advance reservation admission control. *Networking and Mobile Computing*, 2005.