Parallel Circuit Provisioning in ESnet's OSCARS

Jeremy M. Plante*, Dylan A.P. Davis†, and Vinod M. Vokkarane*
*Department of Electrical and Computer Engineering, University of Massachusetts, Lowell
†Department of Computer Science, University of Massachusetts, Lowell
{Jeremy_Plante, Dylan_Davis}@student.uml.edu, Vinod_Vokkarane@uml.edu

***Abstract*—Large-scale science applications generate great volumes of data, which are frequently stored in remote data repositories or shared with cooperating laboratories across the network through the use of advance reservation connections. The groups that utilize these data transfers would benefit from having their applications simultaneously transmit data over multiple channels in parallel. Many of today's networks do not however provide the resource-aware scheduling to support this parallelization. As part of a framework for providing said applications with parallel resource-optimized provisioning of end-to-end requests, we propose and develop a scheduling enhancement to ESnet's On-demand Secure Circuits and Advance Reservation System (OSCARS). This enhancement comes in the form of a front-end client, the behavior of which we quantitatively evaluate to compare the performance of parallel resource-provisioning to serial resource usage for both unicast and anycast scenarios.**

I. Introduction

Data generation and archival rates are increasing dramatically as large-scale science applications grow in sophistication. Virtually all fields of science require inter-laboratory cooperation or data sharing infrastructures. Emerging petascale and exascale scientific applications demand investigation by teams of research scientists who may be distributed across nations and even continents. Optical networks have proven ideal for supporting the growing needs of scientific experiments which require large volumes of data to be distributed and analyzed by a number of geographically dispersed users.

There are a number of applications wherein *a priori* knowledge of a network's state may be beneficial to maintaining the health and availability of the network. Consider large-scale off-site backups or updates to remote data repositories. These backups may be regular and predictable, and so it makes sense to schedule them in advance, allowing applications to be scheduled around other existing connections. Most e-Science applications involve large quantities of data being transferred, to the point that if the transfers become blocked, the retransmission attempts may become costly. These Advance Reservation (AR) connections allow the network to better schedule and provision applications that benefit from window- or deadline-scheduled events [1]–[3].

ESnet's On-demand Secure Circuits and Advance Reservation System (OSCARS) has been largely adopted by the science and networking community to automate the process of establishing end-to-end dedicated Virtual Circuits (VCs) for guaranteed AR service. OSCARS circuits make up half of the 60 petabyte annual traffic load carried by the Department of Energy's backbone Energy Sciences Network (ESnet) [4]. OSCARS presently supports only directed end-to-end unicast transfers, limiting the degree of parallelism for any data transfer.

In current large- or extreme-scale science architectures, data is stored, processed, and analyzed in parallel. However, the underlying core network is not currently used to harness parallelism, instead necessitating serial resource usage. A discrepancy exists between the availability and usability of parallel technologies. The network is capable of parallel transfers, which can be used to increase the volume of transmissions by reserving resources along parallel paths for a single request. These parallelisms however, are not yet fully usable to researchers in today's networks, such as ESnet, due to the lack of a resource-aware, intelligent, and flexible scheduler. To this end, we have proposed a framework for parallel resource-optimized provisioning of end-to-end requests [5]. A major contribution of this DOE-funded project is a scheduling enhancement to OSCARS which allows for parallel resource provisioning across multiple link-disjoint paths for a single request. We enforce link-disjointedness due to the frequency of link failure in networks, as the resulting loss of data can be tremendous if several parallel circuits share a link. At present, applications or users must currently reserve additional network resources for additional paths through OSCARS individually. A multipath transfer service allows applications to utilize more than the maximum bandwidth available along a single network path for parallel storage and retrieval operations. In this work, we propose and develop a front-end client to enable multipath communication using OSCARS as the underlying provisioning controller. We further extend our proposed multipath solution to cooperate with an anycast enhancement to OSCARS, which was previously proposed and evaluated in detail [6].

There exist several destination-agnostic applications that can take advantage of anycast request provisioning, such as database replication and off-site backups. *Anycasting* refers to the transmission of data from a source node to any one member among a candidate destination set of D nodes. Such requests may be denoted as anycast $D/1$. By providing more than one candidate destination, the overall success rate of request provisioning can be dramatically improved by selecting the "best" destination at runtime based on the current network state [7]. Anycast can not only aid in load-balancing, but can select destinations based on various cost metrics, such as distance from the source, Grid/Cloud resource availability at the destination datacenters and supercomputing sites [8], or energy consumption along routes to each destination [9].

The remainder of this paper is organized as follows. Section II provides an overview of the OSCARS modules that are fundamental to the understanding of our proposed multipath client's operation, along with a summary of the previous work to create an anycast OSCARS deployment. A thorough description of the proposed multipath solution, as well as modifications required to the core OSCARS components in order to support the client, is offered in Section III-A. Section III-B details our multipath client's operation for anycast circuit requests, and offers two heuristics for selecting the appropriate destination according to the end-user's requirements. The client's behavior is quantitatively analyzed in Section IV for

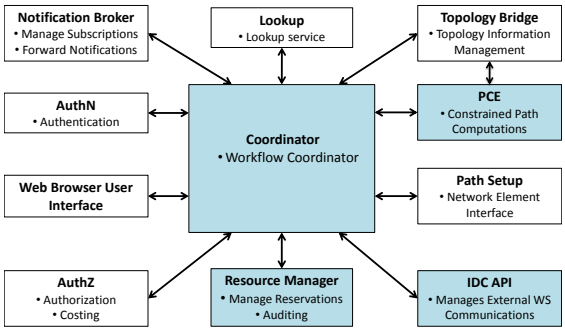


Fig. 1: OSCARS modular framework.

both unicast and anycast request scenarios. Section V discusses some areas of future work to the proposed and implemented multipath client solution, and Section VI concludes the paper.

II. OSCARS Overview

Our proposed and implemented multipath solution is a mostly front-end client that is dependent on the core OSCARS functionality.We present a brief overview of the circuit-provisioning process and interfaces provided by OSCARS. This section provides a high-level description of the OSCARS structure and the circuit request workflow, as well as a summary of the modifications that contributed to the anycast OSCARS deployment presented in [6].

A. Traditional OSCARS Deployment

OSCARS allows users to reserve network resources, such as bandwidth and VCs in both the space- and time-domains. Designed to support long-lived, large-scale data transfers, the OSCARS software package is the controller and operator for a network which configures the physical components and Virtual LANs (VLANs) for carrying and supporting dedicated AR VC reservations. OSCARS is an open-source project and can be modified as necessary based on network, ISP, or application requirements. The latest release of OSCARS, version 0.6, accomplishes its guaranteed service delivery goals through the use of a number of service modules, each of which is deployed as a WebService that communicates with the other modules, as shown in Fig. 1. Each service provided by OSCARS can be deployed, customized, and disabled separately due to the modular nature of the system. Given the nearly two dozen service modules currently incorporated into the system, only those that are directly related to the core circuit-provisioning functionalities will be covered in this subsection.[1]

Coordinator: As the key workflow engine for OSCARS, the *Coordinator* module serves as a central message passer, enabling communication between each other module to allow for establishing, canceling, or querying the status of VCs.

PCE: The *Path Computation Engine (PCE)* stack determines which path to reserve based on provided input constraints, such as start-time or required bandwidth, and is made up of sub-modules, each playing a different role in the routing process. User-defined constraints are combined with the up-to-date network topology information and a set of optional-constraints (further explained in Section II-B) as input into each sub-module. The sub-modules each output a pruned topology with elements, each given a Universal Resource Name (URN), removed based on the module's criteria. Each URN represents a combination of a node, port, and link in the topology. The constraints and modified topology are passed down the stack until a final network topology is returned, which will consist only of the selected path, established on a dedicated VLAN.

Resource Manager: Responsible for keeping track of up-to-date information on the currently available network resources, e.g., bandwidth or VLANs, the *Resource Manager* provides this critical information to the PCE stack during path computation to ensure that pruning and network state updating is performed correctly.

IDC API: The *Inter-Domain Controller (IDC)* API supplies an interface for front-end clients to make use of the VC services provided by OSCARS. Clients issue a request to OSCARS, which then performs its standard internal processing based on the client inputs and issues a reply with relevant response details. Considering this architecture, client applications, such as our proposed multipath client, can create, modify, query, and cancel circuit reservations using the underlying tools provided by OSCARS, while also performing additional front-end tasks independent of the core OSCARS system.

B. Anycast PCE Design

The PCE stack, as previously described, is comprised of sub-modules that together compute paths given the current network topology, user-provided details, such as request start-time and bandwidth requirement, and a set of optional-constraints. Optional-constraints are loosely-defined input objects that can be used within the PCEs, with some modification, to manipulate their behaviors. For the purpose of performing anycast routing, the entire set of candidate-destinations can be passed in as a collection of optional-constraints, and then the modified PCE stack considers these candidate destinations rather than the traditionally expected single-destination. To this end, we have proposed and implemented an enhanced PCE stack implementation, composed of several new sub-modules to support anycast communication [6]. This subsection summarizes the proposed structure, which must be deployed as an alternative to the default unicast-only PCE stack. The anycast PCE, which is available via ESnet's OSCARS repository [10], processes a network topology composed of URNs, and returns a single path from the source to the destination selected via anycast routing. The anycast PCE is comprised of four ordered modules:

(i) **Anycast Connectivity PCE**: A network topology graph corresponding to the network connectivity graph between the source and candidate destination nodes is output by this PCE module. Every pair of nodes that is not physically connected is pruned, ensuring that the domain is accurately interpreted by the subsequent PCEs.

(ii) **Anycast Bandwidth PCE**: Any URNs that do not meet the bandwidth requirements specified by the user's anycast request, whether by design or due to competitive allocation, are pruned by this PCE.

[1]For greater detail on OSCARS modules, the reader is referred to [4].

(iii) **Anycast VLAN PCE**: This PCE module prunes out all URNs that do not have enough VLAN tags, a representation of the maximum number of accommodated VCs at a node, in order to support the intended circuit. This ensures that logical connection establishment for successful requests will be properly secured.

(iv) **Anycast Dijkstra PCE**: To select the final anycast destination, this PCE module computes the separate paths from the source to each destination that meet certain criteria (such as shortest path), and selects the destination that can be reached with the best available path.

Within each of the specified PCEs, each destination is examined to see if it can be reached under the specified constraints, with any infeasible candidates marked with a flag, enabling modules later in the chain to ignore those invalid destinations. The worst-case runtime complexity of the anycast PCE implementation is increased over the standard unicast procedure by a factor of D, the number of destinations in the anycast set. Requests sent through the anycast PCE stack with only a single destination will be serviced identically to the default unicast OSCARS implementation.

III. Proposed Multipath OSCARS Extension

In this section we detail the behavior of our proposed multipath OSCARS client (code available [11]) and detail modifications to the OSCARS PCE stack necessary to support the expected multipath behavior for both unicast and anycast requests.

A. Unicast Multipath

OSCARS only supports unicast VCs between a single source and destination, thereby placing the burden of managing several related parallel paths on our client. The proposed multipath client groups together several independent OSCARS unicast requests, each of which is issued a unique OSCARS ID, into a single multipath request with a shared multipath group ID that can be operated on as a unit. OSCARS will have no inherent knowledge about the relationships between members of a multipath group. Our client's API is modeled on the OSCARS IDC, allowing users to use nearly identical processes to query, cancel, and modify existing VCs or request new VCs, as if they were working directly with OSCARS. The notable difference between the two interfaces is that multipath users will be prompted to include the number of parallel paths they wish to reserve for each circuit.

Figure 2 provides a detailed view of the clients interaction with OSCARS when creating a multipath group reservation. One iteration through the client-API loop represents the result of creating a single VC reservation in OSCARS. The multipath client interacts directly with the OSCARS API, forwarding the request to the Coordinator, which invokes a call to the PCE stack to reserve the shortest available path. The details of the reservation, successful or not, are passed back through the OSCARS API to the multipath client. The first iteration from the client through the OSCARS PCE is identical to the procedure for establishing a unicast OSCARS request. Note however, that we *have* modified the Dijkstra PCE to support

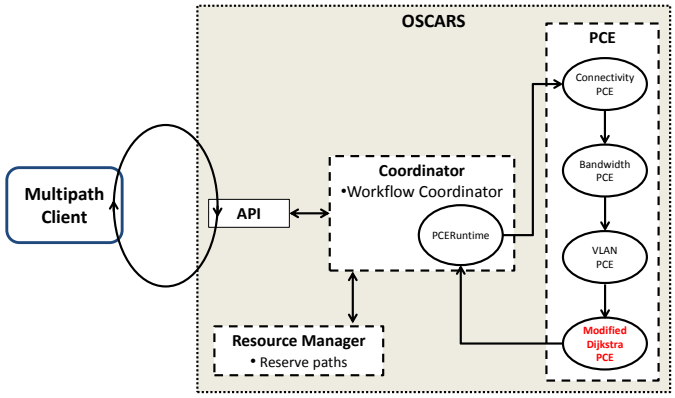


Fig. 2: Interaction between the Multipath client and OSCARS.

the expected behavior for all following iterations through the reservation loop.

The Dijkstra PCE modifications make use of the optional-constraint objects offered by OSCARS, as detailed in Section II-B. The client populates the optional-constraints with the ordered set of URNs that comprise the path reserved in the previous iteration(s) of the reservation loop. When a primary path reservation is unsuccessful, the client exits without attempting to secure additional parallel paths and blocks the request. When a request is successful however, an additional iteration of the loop occurs using a new VC request object containing the first path's list of URNs. The modified Dijkstra PCE uses this list to prune out all of the included URNs from the current topology so that they are unavailable for this iteration, which results in a completely link-disjoint second VC. Note that during URN-pruning, the source/destination URNs must remain in the topology in order to establish a circuit between them. This reservation looping continues until network resource availability prohibits the establishment of further link-disjoint paths, or the user-specified number of paths have been successfully established.

Multipath requests may be specified as either flexible or inflexible. In flexible scenarios the client will perform best-effort VC establishment, such that K paths may be requested, and $M \leq K$ paths will be successfully provisioned. For example, in heavily loaded networks, it is possible for $M = 1$, which is still considered a successful flexible multipath reservation.

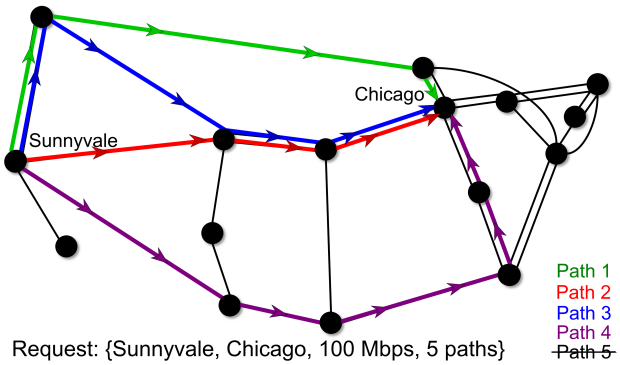


Fig. 3: Given the nodal degree of the source node, only four of the five requested parallel paths may be successfully provisioned. Note that the topology includes separate, parallel links between some node pairs.

Blocking occurs only when $M = 0$. The value of M depends on network resource availability and network connectivity. Consider the example shown in Fig. 3 where five parallel paths are requested on the ESnet topology from Sunnyvale to Chicago, each with a bandwidth requirement of 100 Mbps. The source node has a nodal degree of five, but only four of those adjacent links are connected to Chicago. Using link-disjoint routing, there is no solution for routing the fifth path that does not intersect the resources allocated to the first four paths.

For inflexible requests, the client will attempt to reserve the specified number of parallel paths, and if that quantity cannot be reached, the entire request is deemed a failure. For example, if a request demands three parallel paths, but only two can be provisioned, the multipath client will then issue a cancellation order to OSCARS for the first two paths in the group and mark the entire multipath request as blocked. In either scenario, the worst-case runtime complexity for multipath reservation is increased by a factor of K, the number of parallel paths requested.

B. Anycast Multipath

We have considered two alternative approaches for the incorporation of multipath reservation with anycast OSCARS requests.[2] The first is *Anycast Before Multipath (ABM)*, wherein the nearest anycast destination is selected from the candidate set, and then multiple parallel paths are provisioned to that node. The alternative approach is *Anycast Multipath with Minimum Hops (AMMH)* which calls for an iterative assessment of each anycast candidate in turn. In each iteration, a set of parallel paths will be established to a single candidate destination, and the total cost in terms of physical hop count is calculated. The destination which yields the lowest total cost is selected as the best destination and the corresponding set of parallel paths is provisioned. AMMH only considers the cost as a tie-breaker in cases where the same quantity of parallel paths can be established from the source, otherwise the solution which best conforms to the user's requested degree of parallelization is preferred. Note that AMMH is, in essence, an iterative variation of ABM, wherein the simpler heuristic is repeated D (the number of destinations in the anycast set) times. After the D^{th} iteration, the best destination is selected and the same set of parallel paths is finally provisioned as a unicast multipath request. The AMMH heuristic therefore has a worst-case run-time complexity greater than ABM by a factor of $D + 1$.

IV. Performance Evaluation

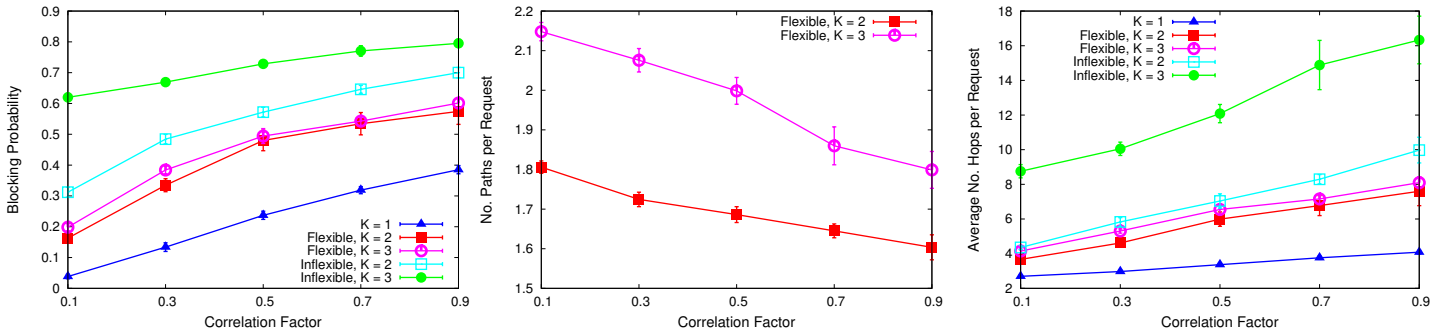
We have evaluated our client on the ESnet science data network core topology, shown in Fig. 3, and considered various multipath traffic scenarios for both unicast and anycast requests. The considered topology is assumed to have bidirectional links, all with a uniform capacity of 10 Gbps. The source and destination(s) are uniformly distributed amongst all nodes, and the request bandwidth is uniformly distributed in the interval [1 Gbps, 5 Gbps], in increments of 1 Gbps. K paths are requested for each multipath request, where $K \geq 1$, and each of those paths is expected to have the same bandwidth and duration as the next. To-date, OSCARS has been responsible for approximately $5,000$ VC reservations on ESnet, when considering all demonstration circuits [4], and so we have considered small, realistic traffic sets of one hundred requests to evaluate our client's behavior when interacting with OSCARS. All requests are scheduled to reserve, transmit, and release network resources within a two-hour time window. A request set's *correlation factor* corresponds to the probability that requests overlap during this time window. As the correlation factor increases, more requests overlap in time; a correlation factor of zero provides a set of completely time-independent reservations. The formula for calculating the correlation factor for a set of requests is given as $\sum_j C_j/n(n-1)$, where n is the number of requests to schedule, and C_j is the number of requests which overlap in time with request j [12]. Please note that the correlation factor does not directly represent load on the network, as the overlapping requests are in fact multipath requests targeting multiple parallel paths at any given point in time. All results shown in this section represent the average of 30 unique sets of reservation requests, and we have included the 95% confidence intervals, which are quite narrow, on each graph.

A. Unicast Multipath Evaluation

We first evaluate our multipath client for unicast-only reservations. This is a quantitative evaluation of our client in a configuration that can be deployed alongside the default OSCARS system, with a modified Dijkstra PCE as detailed in Section III. We have considered traffic wherein requests aim to provision either two or three parallel paths. We have not considered four-path or five-path requests because the physical connectivity of the topology and the nodal degrees of most nodes would not support that many simultaneous paths. We have not distinguished between requests that are blocked due to bandwidth oversubscription and requests that are blocked as a result of exceeding the physical nodal degree limitations of a node. We investigate these multipath request scenarios for both flexible and inflexible routing and compare them to traditional single-path unicast OSCARS requests.

Figure 4a shows the blocking comparison between the considered request types. Unsurprisingly, when more paths are requested, there is more blocking due to bandwidth oversubscription. For inflexible requests, a two-path scenario blocks as much as 30% of the incoming requests, even at low correlation factors. The three-path requests are blocked twice as often at low loads. Note however that given that two-path AR necessitates a 100% increase in bandwidth requested, a 30% increase in blocking is quite attractive. There is not much observed difference between the two-path and three-path flexible requests. By observing Fig. 4b, we can see that this is because as more requests enter the system, the number of provisioned paths decreases. The average number of paths reserved per request (excluding the blocked requests) for flexible two-path requests is only 1.8 paths, and the corresponding value for flexible three-path requests is just over 2.1 paths for low correlation factors. Even when the load is not high, requests are not being completely fulfilled. It can be observed however that even at high correlation factors, the average number of paths per successful request is still quite a

[2] In order to support anycast multipath reservations, the multipath client must be used with the anycast OSCARS deployment, which is separate from the default unicast OSCARS.



(a) Blocking probability. (b) Average path count per successful request. (c) Average hop count per successful request.

Fig. 4: Unicast multipath evaluation for requests with K paths.

bit higher than 1, indicating that the multipath functionality is not going to complete waste and many early requests are still being satisfied to their maximum potential.

The total hop cost for multipath requests is shown in Fig. 4c, which has been normalized against blocking. OSCARS reserves the shortest available primary path, and each additional disjoint path must be at least as long as the previously allocated route. For each additional path reserved, the relative hop count for the request will increase, with the second or third paths taking inefficient routes in order to avoid the previously allocated links. We can see that traditional unicast routes are between two and four hops long for all correlation factors, whereas the addition of another (inflexible) path doubles the number of traversed hops at low loads and triples it at high loads. This trend is scaled higher when $K = 3$.

B. Anycast Multipath Evaluation

Like we have done for the unicast evaluation, we have also considered both flexible and inflexible anycast requests using both the ABM and AMMH routing heuristics described in Section III-B. Each request is given a candidate destination set of size two or three, i.e. anycast 2/1 and 3/1.[3]

Figure 5a depicts the blocking probability for anycast 2/1 with flexible multipath reservation. Similar trends to those observed for unicast scenarios are apparent, though unsurprisingly, all blocking rates are lower than those corresponding to the unicast multipath reservations evaluated in Fig. 4a. Blocking does increase as more paths are requested, but the flexibility in the degree of required parallelism prevents any noticeable difference between two-path and three-path results. Furthermore, AMMH outperforms ABM by only the slightest of margins, and since the behavior of the heuristics is identical for single-path scenarios, there is no difference in their relative blocking. Note also that AMMH reserves on average, fewer paths successfully than ABM for each multipath request, as shown by Fig. 5b. The addition of a second candidate destination has not however improved the degree of parallelism for flexible reservations over corresponding unicast analogues when $K = 2$. Setting $K = 3$ does result in slightly more paths on average for the anycast scenario. Allowing destination-selection flexibility for anycast also decreases the total hop cost of multipath requests as shown in Fig. 5c. Here we can observe that the inclusion of a candidate destination set significantly reduces path length over comparable unicast requests; at a correlation factor of 0.9, the anycast solutions reduce overall hop cost by more than 2.5 hops for both $K = 2$ and $K = 3$. Furthermore, AMMH successfully reserves shorter paths than ABM for all values of K, thereby using the network more efficiently.

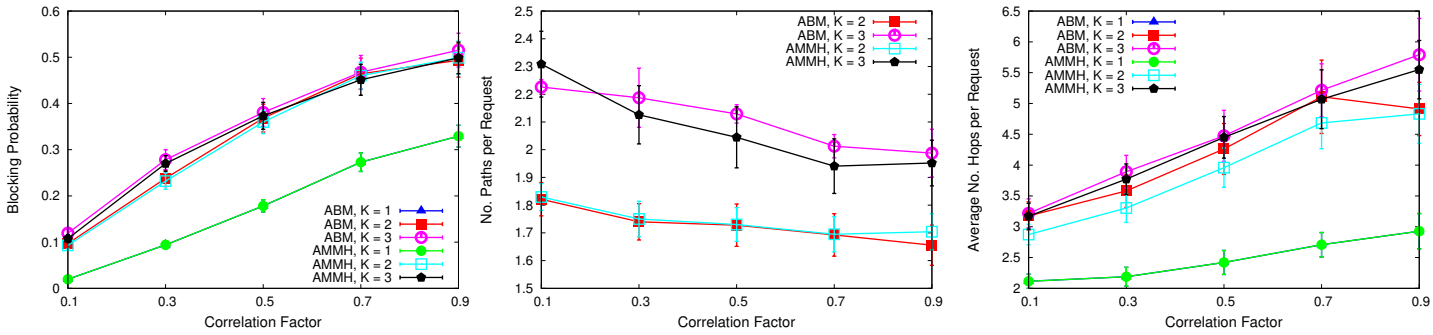
Given these results, we are confident that we can provide some guarantee for the degree of parallelism on the network. In future works, we will examine the potential for guaranteeing a minimum quality of parallelism wherein a user desiring parallel paths can be offered a guarantee of some number of parallel paths given the current state of the network.

Figure 6 offers a blocking analysis of inflexible anycast multipath reservations. Similarly to flexible multipath requests, AMMH yields nearly identical results to ABM; we therefore omit the AMMH results from this figure. For these reservations, as the degree of desired parallelism increases, so too does overall blocking. Note that ABM solutions result in blocking rates slightly lower than unicast blocking for *flexible* scenarios for $K = 2$ at correlation factors above 0.1. This demonstrates the enormous advantage of destination-selection. The impact of this advantage is diminished however when $K = 3$, which results in blocking performances only slightly better than the *inflexible* unicast results for $K = 3$.

V. Discussion and Future Enhancements

Multipath solutions offer issues such as differential latency, packet reordering, and failovers, which can be evaluated as future work through deployment in a real world testbed. There are also some design limitations to the current multipath client. As mentioned in Section III-A, on each iteration through the multipath reservation loop, all previously allocated URNs are pruned from the topology, except for the source and destination, which must remain in order to define a VC's origin and termination points. Consider the specific case in which these URNs are just one physical hop apart. Here, the entire shortest path consists of just two URNs which cannot be pruned out. Thus, each subsequent iteration would merely return the same single-hop path from source to destination until bandwidth or VLAN resources are exhausted. We have taken care to disallow single-hop requests in our evaluations, but an area of future enhancement includes the proposal of an alternate multipath solution for single-hop scenarios.

[3]We have omitted the anycast 3/1 results, as they exhibit similar trends to anycast 2/1. This decision is also informed by our findings in [6], wherein we learned that adding more candidate destinations does not severely impact performance.



(a) Blocking probability. (b) Average path count per successful request. (c) Average hop count per successful request.

Fig. 5: Anycast 2/1 multipath evaluation for flexible requests with K paths.

An additional area of future work is the evaluation of our multipath solution for fault tolerances. If a fiber-link were cut, or a serious failure occurred at a node or datacenter along the primary route, our multipath client could provide survivable backup solutions to maintain the user's expected quality of service and meet the specified, guaranteed deadline. Our multipath solution emphasizes the link-disjoint nature of the provisioned parallel paths, and thus each path may serve as a survivable solution to the next. We therefore plan to evaluate our client on fault-prone systems for path protection.

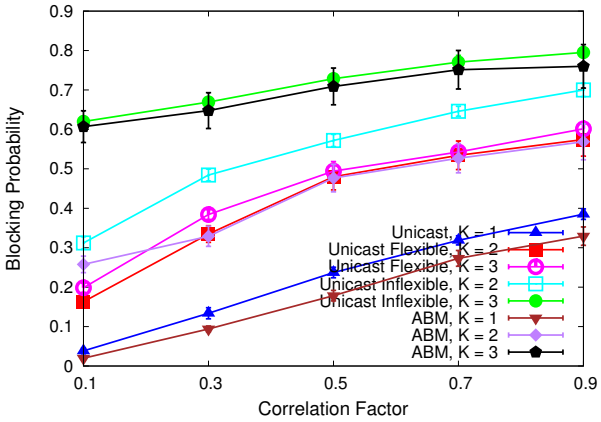


Fig. 6: Blocking probability of anycast 2/1 requests with inflexible multipath routing of K paths.

Routing the shortest available set of paths may be too greedy when available capacity on the selected links is limited. In future extensions, we will further analyze the selected path sets based on cumulative bandwidth consumption to try and uniformly maintain some level of unused capacity for future requests across the entire network. We also aim to enhance our multipath client to support sets of paths with heterogeneous transfer rates and bandwidth needs. Such a study will benefit greatly from cumulative bandwidth consumption and capacity-driven evaluations.

VI. Conclusion

With the ever-increasing amount of data produced by scientific communities, solutions for efficient and safe transmission are increasing in importance. Large file transfers can benefit from simultaneous transmissions over parallel channels. In this paper, we introduced a multipath solution to ESnet's OSCARS VC provisioning software and have quantitatively and qualitatively compared it to both the traditional unicast and enhanced anycast deployments of the system. With only minimal modification to the OSCARS Dijkstra PCE module, we have developed our solution as a mostly front-end client that provides end-users and applications with familiar and flexible interfaces by which to reserve multipath VCs. The work we have presented herein has resulted in a novel contribution to the scientific community, and has yielded the first parallel multipath solution that cooperates with OSCARS.

Acknowledgment

This work has been supported by the Department of Energy (DOE) PROPER project under grant DE-SC0012115TDD [5].

References

[1] J. Zheng and H. T. Mouftah, "Supporting advance reservations in wavelength-routed WDM networks," in *Proc., IEEE Int. Conf. on Computer Communications and Networks (ICCCN)*, 2001, pp. 594 – 597.

[2] ——, "Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks," in *Proc., IEEE Int. Conf. on Communications (ICC)*, vol. 5, 2002, pp. 2722–2726.

[3] N. Charbonneau and V. M. Vokkarane, "A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 1037–1064, Dec. 2012.

[4] "OSCARS," Apr. 2014, [Online]. Available: http://www.es.net/services/oscars/read-more/.

[5] V. M. Vokkarane, "PROPER: Parallel Resource-Optimized Provisioning of End-to-end Requests," [Online]. Available: http://faculty.uml.edu/Vinod_Vokkarane/proper/.

[6] M. Boddie, T. Entel, C. Guok *et al.*, "On extending ESnet's OSCARS with a multi-domain anycast service," in *Optical Network Design and Modeling (ONDM)*, 2012, pp. 1–6.

[7] D. Din, "A hybrid method for solving ARWA problem on WDM network," *Elsevier Computer Communications*, vol. 30, no. 2, pp. 385–395, Jan. 2007.

[8] T. Stevens, M. D. Leenheer, C. Develder *et al.*, "Anycast routing algorithms for effective job scheduling in optical grids," in *Proc. of European Conference on Optical Communication (ECOC) 2006*, Cannes, France, Sep. 2006, pp. 371–372.

[9] B. G. Bathula, M. Alresheedi, and J. Elmirghani, "Energy efficient architectures for optical networks," in *Proc. London Communications Symposium (LCS) 2009*, London, UK, Sep. 2009.

[10] UMass Dartmouth and ESnet, "Anycast Multi-Domain OSCARS," [Online]. Available: https://oscars.es.net/repos/oscars/branches/common-anycast/.

[11] J. M. Plante, D. A. P. Davis, and V. M. Vokkarane, "Multipath service OSCARS client," [Online]. Available: http://faculty.uml.edu/Vinod_Vokkarane/proper/multipath_oscars/.

[12] N. Charbonneau and V. M. Vokkarane, "Static routing and wavelength assignment for multicast advance reservation in all-optical wavelength-routed WDM networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 1–14, 2012.