Path Computation with Variable Bandwidth for Bulk Data Transfer in High-performance Networks

Yunyue Lin, Qishi Wu
Department of Computer Science
University of Memphis
Memphis, TN 38152, USA
Email: {ylin1, qishiwu}@memphis.edu

***Abstract*—There are an increasing number of high-performance networks that provision dedicated channels through circuit-switching or MPLS/GMPLS techniques to support bulk data transfer in large-scale science or e-commerce applications. These dedicated links are typically shared by multiple users through advance reservations, resulting in varying bandwidth availability in future time periods. Therefore, efficient advance bandwidth reservation algorithms are needed to improve the utilization of network resources and meet the transport requirements of application users. We investigate the bandwidth-oriented path computation problem for two types of data transfer: (i) fixed path with variable bandwidth and (ii) variable path with variable bandwidth to minimize the transfer end time of a given data size. We prove that both problems are NP-complete and propose a heuristic algorithm for each of them. Extensive simulation results illustrate the performance superiority of the proposed heuristics over methods based on greedy strategies.**

***Index Terms*—path computation, data transfer, high-performance networks**

I. Introduction

A. Background

Many large-scale applications in science, engineering and business domains are generating colossal amounts of data, which must be transferred across wide areas for remote operations. Typical examples include next-generation computational sciences where large simulation datasets produced on supercomputers are shared by distributed collaborative scientists [1], [2]. Providing the capability of large data transfer over dedicated channels is critical to ensuring the success of these applications.

The significance of high-performance networks has been well recognized in the broad science and network research communities, and several projects are currently underway to develop such network capabilities, including UltraScience Net (USN) [3], Circuit-switched High-speed End-to-End Transport ArcHitecture (CHEETAH) [4], and many others. The dedicated links provisioned by these high-performance networks are typically shared by multiple users through advance reservations, resulting in varying bandwidth availability in future time periods. Therefore, finding appropriate paths that make full use of the "leftover" bandwidths is critical to maximizing the utilization of network resources and meeting the transport performance requirements of application users.

We formulate and investigate the bandwidth-oriented path computation problem for two types of data transfer: fixed path with variable bandwidth (FPVB) and variable path with variable bandwidth (VPVB), to minimize the transfer end time (TET) of a given data size. These two problems were first described and studied by Lin *et al.* in [5], where optimal algorithms are designed to solve them. We would like to point out that the solution to the FPVB problem in [5] is exponential with respect to the number of combined time slots, and hence may not scale well for large-scale networks. Furthermore, the VPVB problem in [5] does not consider path switching time. We prove that both FPVB and VPVB with path switching delay are NP-complete, and propose a heuristic algorithm for each of them. These heuristic approaches are implemented and tested in a large set of simulated networks with various data sizes. In small-scale networks, we use the optimal algorithm in [5] as a comparison base and the experimental results show that the performance of the proposed heuristics is close to the optimal one; while in large-scale networks, we illustrate the performance superiority of our approaches by comparing with methods based on greedy strategies.

B. Related Work

As dedicated networks are increasingly developed and deployed under different high-performance networking initiatives, many algorithms have been designed for in-

advance bandwidth reservation and scheduling. We provide below a brief background survey of such efforts.

In [6], Rao *et al.* describe four basic scheduling problems with different constraints on target bandwidths and time slots. Similar problems are further discussed in [7] with an in-depth explanation on the solution to each of these problems. These basic scheduling problems with several extensions are also investigated in [8] with a focus on increasing the flexibility of services. The scheduling algorithm proposed by Cohen *et al.* in [9] considers the flexibility of transfer start time and the capability of path switching between different paths during the course of a transport session to improve network utilization. In [10], Grimmell *et al.* formulate a dynamic quickest path problem, which deals with the transmission of a message from a source to a destination with the minimum end-to-end delay over a network with propagation delays and dynamic bandwidth constraints on links. In [11], files are transferred with varying bandwidths in different time slots in a simple case where the path is pre-specified. Ganguly *et al.* generalize the problems of finding an optimal path in a graph with varying bandwidth to minimize the total transfer time in [12]. In [13], Gorinsky *et al.* propose a Virtual Finish Time First algorithm to schedule incoming files in a preemptive manner to minimize total TET on a given dedicated channel. Most of the aforementioned work is primarily focused on instant scheduling of bandwidths provisioned by dedicated networks.

The rest of the paper is organized as follows. We formulate FPVB and VPVB problems in Section II, and prove their NP-completeness in Section III. We present heuristic algorithms in Section IV and evaluate their performances in Section V.

II. Problem Formulation

We represent a dedicated network as a graph $G = (V, E)$, where each link $l \in E$ maintains a list of residual (available) bandwidths specified as segmented constant functions of time. We use a pair of time-bandwidth (TB) $(t[i], b_l[i])$ to represent the residual bandwidth $b_l[i]$ of link l at time slot $(t[i], t[i+1])$, $i = 1, 2, \ldots, T$, where T is the total number of time slots.

To make full use of the time-varying residual bandwidths, we formulate two data transfer problems with common objective of minimizing data transfer end time (TET) based on the constraint of path switching: Given a graph $G = (E, V)$ with time-bandwidth list TB_l for all links $l \in E$, path switching delay τ, source v_s and destination v_d, data size δ, we consider:

- FPVB: compute a fixed path (i.e. no path switching is allowed) from v_s to v_d with varying bandwidths across all time slots to minimize the data TET.
- VPVB: compute a set of paths from v_s to v_d and schedule path switchings at all time slots to minimize the data TET.

In FPVB, we restrict the switching path to be fixed during the entire data transfer period. The optimal solution with exponential complexity proposed in [5] can only be applied to small-scale networks with a limited number of time slots. While in VPVB, path switchings are allowed between two adjacent time slots to fully utilize bandwidth resources. However, since path switchings generally incur a certain amount of overhead, it may not be always beneficial to do so in some applications, especially when the switching overhead τ is comparable to the transfer time in a time slot. If the path switching overhead τ is negligible, we can find an optimal solution by computing the widest path in each time slot. On the other hand, if the path switching overhead τ is sufficiently large, VPVB reduces to FPVB. Since both FPVB and VPVB allow variable bandwidth during the entire course of data transfer, the transfer should always start immediately at the first available time slot.

III. Complexity Analysis

A. FPVB is NP-complete

We prove that FPVB is NP-complete by reducing from *0-1 Total Bandwidth (0-1 TB)* problem, whose NP-completeness is shown in [8]. We formulate the decision version of FPVB as follows: Given a graph $G = (E, V)$ with time-bandwidth list TB_l for all links $l \in E$, source v_s and destination v_d, data size δ, does there exist a fixed path from v_s to v_d with varying bandwidths across all time slots, such that the data can be completely transferred along the path during the interval $[0, T_{end}]$? Suppose the data transfer request is made at time $t = 0$.

Theorem 1: FPVB is NP-complete.

Proof: We first show that FPVB $\in$ NP. Given a solution (a path from v_s to v_d) to FPVB, one can verify in polynomial time the validity of the solution by checking whether or not the data size transferred on the path during the interval $[0, T_{end}]$ is greater than or equal to δ.

We now reduce the 0-1 TB problem [8] to FPVB. The 0-1 TB problem is defined as follows: Given a graph $G = (E, V)$ with either 1 or 0 available bandwidth on each link $l \in E$ at each time slot of unit length, source v_s and destination v_d, does there exist a path from v_s to v_d such that during the interval $[0, T_{end}]$, the number of time slots

in which all path links have a bandwidth value of 1 is at least β?

Let $(G, v_s, v_d, T_{end}, \beta)$ be an arbitrary instance of 0-1 TB. We construct an instance $(G', v'_s, v'_d, T'_{end}, \delta)$ of FPVB from the instance $(G, v_s, v_d, T_{end}, \beta)$ in polynomial time such that G' has a path from v'_s to v'_d and the data of size δ can be completely transferred along the path during the interval $[0, T'_{end}]$, if and only if there exists a path from v_s to v_d in G that the number of time slots in which all path links have a bandwidth value of 1 is at least β during the interval $[0, T_{end}]$. We set $G' = G$, $v'_s = v_s$, $v'_d = v_d$, $T'_{end} = T_{end}$, and $\delta = \beta$. Clearly, this instance construction can be done in polynomial time.

Suppose that there exists a path P from v_s to v_d in G that the number of time slots in which all path links have a bandwidth value of 1 is at least β during the interval $[0, T_{end}]$. We can find a corresponding path P' in G', where $P' = P$. Since one can transfer 1 unit of data in each unit time slot with a bandwidth of 1, the accumulated data size transferred along path P' during the entire interval is at least β. Therefore, the data of size $\delta = \beta$ can be completely transferred during $[0, T'_{end}]$ along path P'. Hence, P' composes a solution to FPVB.

Conversely, let P' be the path from v'_s to v'_d and the data of size δ can be completely transferred along the path during the interval $[0, T'_{end}]$. We can find a corresponding path P in G, where $P = P'$. Obviously, the number of time slots in which all path links have a bandwidth value of 1 is at least $\beta = \delta$ during the interval $[0, T_{end}]$, since transferring 1 unit of data requires one unit time slot in which all path links have a bandwidth value of 1. Hence, P composes a solution to 0-1 TB. Proof ends. ■

B. VPVB is NP-complete

We prove that VPVB is NP-complete by showing that FPVB is a special case of VPVB.

Theorem 2: VPVB is NP-complete.

Proof: We could restrict VPVB to FPVB by allowing only instances in which $\tau \geq \delta / min(b_l[i])$, where $l \in E$, $i \in [1, T]$. Since the path switching delay is sufficiently large compared to the available time slot, performing any path switching would negatively affect the data TET. The validity of NP-completeness proof by restriction is established in [14], where "restriction" constrains the given, not the question of a problem. Since FPVB is NP-complete, so is VPVB. ■

IV. Algorithm Design

we propose a heuristic path computation algorithm for FPVB and VPVB, respectively, and also design a naive greedy algorithm for performance comparison.

A. Heuristic Algorithms for FPVB

We first define several notations and operations to facilitate our explanation on the algorithm design:

- $t[v]$: TET through the computed path from v_s to v.
- Q: a min-priority queue of vertices, keyed by their TET values.
- $ExtractMin(Q)$: the operation of extracting the vertex with minimum TET in Q.
- $b_{(v_s,v)}[i]$: the bandwidth of the computed path from v_s to v in the i-th time slots, which is the bottleneck bandwidth of all component links on the path.

FPVB computes a fixed path with variable bandwidth from source to destination for data transfer. We first propose a greedy solution based on Dijkstra's algorithm, which takes as input a graph $G = (V, E)$, time-bandwidth list TB, source v_s and destination v_d, and data size δ. We refer to this algorithm as $GreedyFP(G, TB, v_s, v_d, \delta)$ with the objective to minimize the TET. The pseudo-code of the greedy algorithm is shown in Algorithm 1. The output of the algorithm is a fixed path from v_s to v_d. The algorithm first computes the TET for data of size δ over every single link, and assigns the TET as the weight to each link. It then computes the narrowest path in the updated graph by extending Dijkstra's algorithm. Here, the narrowest path is defined as the path from the source vertex to the destination vertex on which the maximum weight among all component links is minimized. The total computational complexity of this greedy algorithm is $O(|E| \cdot (T + lg|V|))$.

Algorithm 1 $GreedyFP(G, TB, v_s, v_d, \delta)$

```
for all (u, v) ∈ E do
    w_(u,v) = TET for data of size δ from u to v (or v to u) along link (u, v);
for all v ∈ V do
    t[v] = ∞;
t[v_s] = 0;
Q = V;
while Q ≠ ∅ do
    u = ExtractMin(Q);
    for all v ∈ Q, (u, v) ∈ E do
        if t[v] > max(t[u], w_(u,v)) then
            t[v] = max(t[u], w_(u,v)), prev(v) = u;
Construct the path P from v_s to v_d;
return P.
```

The greedy algorithm does not consider coordination of the component links on the computed path. Note that

the available bandwidth of the path is restricted by the bottleneck bandwidth of all component links at each time slot. We further propose another heuristic algorithm called $MinFP(G, TB, v_s, v_d, \delta)$ to address this issue by keeping track of the bottleneck bandwidth of the path. The pseudo-code of the $MinFP$ algorithm is shown in Algorithm 2. At each relaxation step, the bottleneck bandwidth from v_s to the current vertex is updated. The total computational complexity of the $MinFP$ algorithm is the same as that of the $GreedyFP$ algorithm.

Algorithm 2 $MinFP(G, TB, v_s, v_d, \delta)$

for all $v \in V$ **do**
  $t[v] = \infty$;
$t[v_s] = 0$;
$Q = V$;
while $Q \neq \emptyset$ **do**
  $u = ExtractMin(Q)$;
  for all $v \in Q, (u, v) \in E$ **do**
    if $u \equiv v_s$ **then**
      $b'_{(v_s,v)}[i] = b_{(u,v)}[i], \forall\, i \in [1, T]$;
    else
      $b'_{(v_s,v)}[i] = min(b_{(v_s,u)}[i], b_{(u,v)}[i]), \forall i \in [1, T]$;
    Compute the TET t' of data of size δ from v_s to v with available bandwidth $b'_{(v_s,v)}$;
    if $t[v] > t'$ **then**
      $t[v] = t'$, $b_{(v_s,v)}[i] = b'_{(v_s,v)}[i], \forall\, i \in [1, T]$, $prev(v) = u$;
Construct the path P from v_s to v_d;
return P.

B. Heuristic Algorithms for VPVB

Again, we shall define several notations and operations to facilitate our explanation:

- $g(P_i, i)$: data size transferred on path P_i in the i-th time slot.
- $s(i)$: the time slot of the most recent path switching considered in the i-th time slot.
- $P^f_{[s(i),i]}$: the FPVB path from v_s to v_d that maximizes the throughput within time slot range $[s(i), i]$.
- $g(P^f_{[s(i-1),i-1]}, [s(i-1), i-1])$: the data size transferred on path $P^f_{[s(i-1),i-1]}$ within the time slot range $[s(i-1), i-1]$.

VPVB allows path switchings during data transfer. Due to the varying available bandwidth in the future time slots, path switchings allow data to be transferred along the widest path in each time slot to improve the utilization of the bandwidth resources. On the other hand, path switchings incur additional time overhead. We must take both the positive and negative effects of path switchings into consideration, and make a decision on whether a path switching between two adjacent time slots is worthwhile. We consider the case where the delay τ is smaller than the length of any time slot. Firstly, we introduce a naive greedy algorithm $GreedyVP(G, TB, \tau, v_s, v_d, \delta)$, which is shown in Algorithm 3. The algorithm computes the widest path in the first time slot P^w_1, and then moves to the second time slot and checks whether to switch to the widest path in the second time slot P^w_2 or continue to use the path computed in the first time slot P^w_1. The algorithm repeatedly performs this path switching check until the data have been completely transferred. The total computational complexity of $GreedyVP$ algorithm is $O(T \cdot |E| \cdot lg|V|)$.

Algorithm 3 $GreedyVP(G, TB, \tau, v_s, v_d, \delta)$

Compute the widest path P^w_1 and its transferred data size in the first time slot $g(P^w_1, 1)$, and $P_1 = P^w_1$, $\delta = \delta - g(P_1, 1)$;
for $(i = 2; i \leq T; i++)$ **do**
  Compute the widest path P^w_i in the i-th time slot;
  if $\frac{t_{i+1}-t_i-\tau}{t_{i+1}-t_i} g(P^w_i, i) > g(P_{i-1}, i)$ **then**
    $P_i = P^w_i$, $\delta = \delta - \frac{t_{i+1}-t_i-\tau}{t_{i+1}-t_i} g(P^w_i, i)$;
  else
    $P_i = P_{i-1}$, $\delta = \delta - g(P_{i-1}, i)$;
  if $\delta \leq 0$ **then**
    break;
return $\{P_1, P_2, ..., P_i\}$.

The widest path varies at different time slots. Therefore, using the widest path in the previous time slot to transfer data in the current time slot may lead to a bad performance. We propose a more sophisticated algorithm $MinVP(G, TB, \tau, v_s, v_d, \delta)$, which calls the $MinFP$ algorithm designed for FPVB problem, by changing the inputs as $MinFP(G, TB, v_s, v_d, p, q)$, where p and q are the given start and end time slots, respectively. $MinFP$ described in Algorithm 2 is to compute a path from v_s to v_d to minimize the TET for a given data size, which is essentially the same as computing a path from v_s to v_d to maximize the transferred data size within a given time range. The pseudo-code of the algorithm is shown in Algorithm 4. When moving to the next time slot, say the i-th time slot, the algorithm makes a decision on whether to switch to the widest path in the i-th time slot or stick to the FPVB path computed for time slots ranged from $s(i-1)$ to i. An example of $MinVP$ is shown in

Algorithm 4 $MinVP(G, TB, \tau, v_s, v_d, \delta)$

Compute the widest path in the first time slot P_1^w;
$s(1) = 1$, $P^f_{[s(1),1]} = P_1^w$;
for $(i = 2; i \leq T; i++)$ **do**
  Compute the widest path P_i^w in the i-th time slot;
  Compute the FPVB path $P^f_{[s[i-1],i]}$ from time slot $s[i-1]$ to i by calling $MinFP$ algorithm;
  if $\frac{t_{i+1}-t_i-\tau}{t_{i+1}-t_i} g(P_i^w, i) + g(P^f_{[s(i-1),i-1]}, [s(i-1), i-1]) > g(P^f_{[s(i-1),i]}, [s(i-1), i])$ **then**
    $P_j = P^f_{[s(i-1),i-1]}$, $j \in [s(i-1), i-1]$;
    $\delta = \delta - g(P^f_{[s(i-1),i-1]}, [s(i-1), i-1])$;
    $t_i = t_i + \tau$;
    $s(i) = i$;
    if $\delta - \frac{t_{i+1}-t_i-\tau}{t_{i+1}-t_i} g(P_i^w, i) \leq 0$ **then**
      $P_i = P_i^w$;
      break;
  else
    $s(i) = s(i-1)$;
    if $\delta - g(P^f_{[s(i-1),i]}, [s(i-1), i]) \leq 0$ **then**
      $P_j = P^f_{[s(i-1),i]}$, $j \in [s(i-1), i]$;
      break;
return $\{P_1, P_2, ..., P_i\}$.

Fig 1, where the most recent path switching occurs in time slot $i-3$. When path switching is performed at time slot i, the transferred data size on path $P^f_{[i-3,i-1]}$ within the time slot range $[i-3, i-1]$ and the transferred data size on path P_i^w within time range of $[t_i + \tau, t_{i+1}]$ are $g(P^f_{[i-3,i-1]}, [i-3, i-1])$ and $\frac{t_{i+1}-t_i-\tau}{t_{i+1}-t_i} g(P_i^w, i)$, respectively. When path switching is not performed at time slot i, we compute the FPVB path $P^f_{[i-3,i]}$ from time slot $i-3$ to time slot i, on which the transferred data size within time slot range $[i-3, i]$ is $g(P^f_{[i-3,i]}, [i-3, i])$. If path switching yields a larger data transfer, the paths computed before the i-th time slot are recorded, and the data size δ is updated by subtracting the transferred data size on path $P^f_{[s(i-1),i-1]}$ between time slot $s(i-1)$ and $i-1$. During the time $[t_i, t_i+\tau]$, the system is performing path switching, and there is no data transfer during this time range, so t_i is updated to $t_i + \tau$. Meanwhile, $s(i)$ is updated to i, which means that the most recent path switching occurs at the i-th time slots. On the other hand, if the FPVB path in $[s(i-1), i]$ time slots provides a better solution, we do not record the path information since the FPVB path may change when moving to the next time slot $i+1$. The total computational complexity of the $MinVP$ algorithm is $O(T \cdot |E| \cdot (T + lg|V|))$.

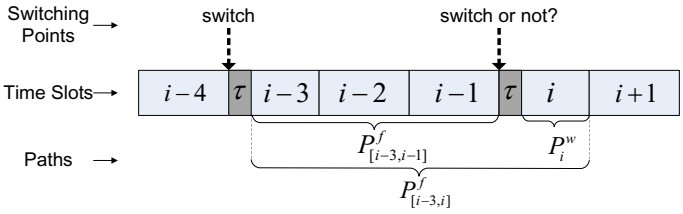

Fig. 1. $MinVP$ algorithm illustration.

V. Performance Evaluation

We present the simulation-based performance comparisons between the heuristics designed for FPVB and VPVB under various network sizes. Each simulated network has a randomly generated network topology for a given number of nodes and links, and the time-bandwidth list of each link is also randomly generated with residual bandwidth ranging from 0.2 Gbps to 10 Gbps at each time slot with the same length of 1 second.

A. Comparison of Algorithms for FPVB

We conduct performance evaluation of $MinFP$, $GreedyFP$, and the optimal algorithm in [5] for FPVB problem using various simulated network and data sizes. Since the computational complexity of the optimal algorithm is exponential, we first compare these three algorithms in a small-scale network with 8 nodes and 16 links for small data sizes ranging from 3 to 8 GBytes. The performance in terms of TET in response to data sizes is shown in Fig. 2, where we observe that $MinFP$ consistently outperforms $GreedyFP$ and is close to the optimal algorithm in all the cases we studied. We further compare $MinFP$ with $GreedyFP$ in a larger network with 50 nodes and 200 links for larger data sizes ranging from 10 to 55 GBytes. The performance of $MinFP$ and $GreedyFP$ in Fig. 3 shows that $MinFP$ is able to achieve about 2 times transfer speedup of $GreedyFP$.

B. Comparison of Algorithms for VPVB

We compare $MinVP$ with $GreedyVP$ for VPVB problem with a path switching delay of 0.2 seconds in the same large network used for FPVB. The performance in Fig. 3 shows that $MinVP$ outperforms $GreedyVP$ in all the cases with different data sizes. Fig. 3 also provides the insight that VPVB scheduling has better performance than FPVB scheduling, since path switching flexibility improves the utilization of network bandwidth resources. We also study the effect of the path switching delay on the data TET. As shown in Fig. 4, the data

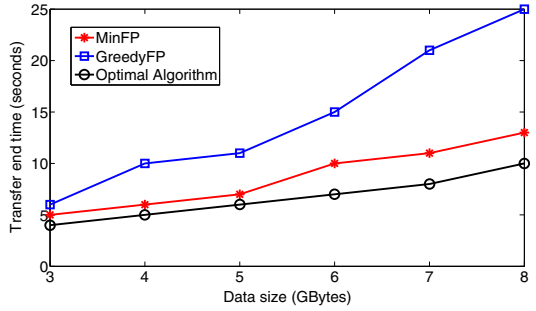

Fig. 2. Comparison of algorithms for FPVB in a network with 8 nodes and 16 links for varying data sizes.

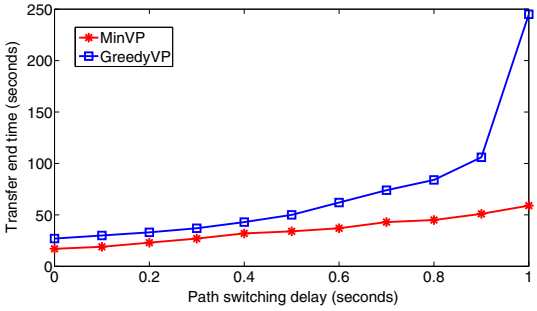

Fig. 4. Comparison of algorithms for VPVB in a network with 50 nodes and 200 links with varying path switching delays.

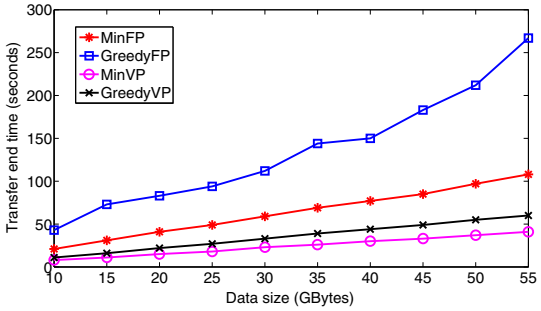

Fig. 3. Comparison of algorithms for FPVB and VPVB in a network with 50 nodes and 200 links with a switching delay of 0.2 seconds.

TET almost linearly increases when the path switching delay increases. When the path switching delay is set to 1, i.e. the length of each time slot, no path switching is performed, which conforms to our argument: VPVB reduces to FPVB when the switching delay is sufficiently large. We observe that $MinVP$ consistently outperforms $GreedyVP$ with different path switching delays.

VI. Conclusion

We considered two path computation problems: FPVB and VPVB with path switching delay. We proved these two problems to be NP-complete and designed heuristic algorithms to minimize data transfer end time. The performance superiority of these heuristics was verified by extensive experimental results in a large set of simulated networks in comparison with greedy strategies.

Acknowledgments

This research is sponsored by National Science Foundation under Grant No. CNS-0721980 and Oak Ridge National Laboratory, U.S. Department of Energy, under Contract No. PO 4000056349 with University of Memphis.

References

[1] Terascale Supernova Initiative (TSI). http://www.phy.ornl.gov/tsi.

[2] Terascale High-Fidelity Simulations of Turbulent Combustion with Detailed Chemistry. http://scidac.psc.edu.

[3] N. Rao, W. Wing, , S. Carter, and Q. Wu, "Ultrascience net: Network testbed for large-scale science applications," *IEEE Communications Magazine*, vol. 43, no. 11, pp. s12–s17, 2005, an expanded version available at www.csm.ornl.gov/ultranet.

[4] CHEETAH: Circuit-switched High-speed End-to-End Transport ArcHitecture, http://www.ece.virginia.edu/cheetah.

[5] Y. Lin, Q. Wu, N. Rao, and M. Zhu, "On design of scheduling algorithms for advance bandwidth reservation in dedicated networks," in *The 2008 INFOCOM High-Speed Networks Workshop*, Phoenix, Arizona, Apr. 13 2008.

[6] N. Rao, Q. Wu, S. Carter, W. Wing, D. G. A. Banerjee, and B. Mukherjee, "Control plane for advance bandwidth scheduling in ultra high-speed networks," in *INFOCOM 2006 Workshop on Terabits Networks*, 2006.

[7] S. Sahni, N. Rao, S. Ranka, Y. Li, E. Jung, and N. Kamath, "Bandwidth scheduling and path computation algorithms for connection-oriented networks," in *Proc. of Int. Conf. on Networking*, 2007.

[8] R. Guerin and A. Orda, "Networks with advance reservations: the routing perspective," in *Proc. of the 19th IEEE INFOCOM*, 2000.

[9] R. Cohen, N. Fazlollahi, and D. Starobinski, "Graded channel reservation with path switching in ultra high capacity networks," in *Proc. of Broadnets*, San Jose, CA, 2006.

[10] W. Grimmell and N. Rao, "On source-based route computation for quickest paths under dynamic bandwidth constraints," *Int. J. on Foundations of Computer Science*, vol. 14, no. 3, pp. 503–523, 2003.

[11] M. Veeraraghavan, H. Lee, E. Chong, and H. Li, "A varying-bandwidth list scheduling heuristic for file transfers," in *Proc. of IEEE Int. Conf. on Comm.*, 2004.

[12] S. Ganguly, A. Sen, G. Xue, B. Hao, and B. Shen, "Optimal routing for fast transfer of bulk data files in time-varying networks," in *Proc. of IEEE Int. Conf. on Comm.*, 2004.

[13] S. Gorinsky and N. Rao, "Dedicated channels as an optimal network support for effective transfer of massive data," in *INFOCOM 2006 Workshop on High-Speed Networks*, 2006.

[14] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York: W.H. Freeman and Company, 1979.