On Periodic Scheduling of Bandwidth Reservations with Deadline Constraint for Big Data Transfer

Yongqiang Wang*, Chase Q. Wu†*, Aiqin Hou*
*School of Information Science and Technology, Northwest University, Xi'an, Shaanxi 710127, China
†Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA
Email: {yqwang, aiqinhou}@nwu.edu.cn, chase.wu@njit.edu

***Abstract*—The efficiency of bandwidth scheduling in high-performance networks is critical to the utilization of network resources and the satisfaction of user requests. In this paper, we formulate a periodic bandwidth scheduling problem to maximize the number of satisfied user requests for bandwidth reservation with deadline constraint on a fixed network path, referred to as multiple deadline-constrained bandwidth scheduling (M-DCBS). We show the NP-completeness of this problem and propose a Maximum User Number Resource Reservation Algorithm (MUNRRA). Extensive simulation results show that MUNRRA exhibits a superior performance over existing algorithms in terms of scheduling success ratio and execution time. Considering the popularity of the DCBS-based service model and the rapid expansion of high-performance networks in both speed and scope, the proposed scheduling algorithm has great potential to improve the network performance of big data applications that require the DCBS service for data transfer.**

***Keywords*-High-performance networks; bandwidth scheduling; big data; resource utilization**

I. Introduction

Many large-scale applications in science, engineering, and business domains are generating colossal amounts of data, now frequently termed as "big data", which must be transferred to remote sites for various purposes such as data storage, processing, and analytics. Unfortunately, the traditional shared public best-effort IP networks such as the Internet are not adequate to meet the unprecedented data transfer challenges posed by the sheer data volume of such scales.

In recent years, high-performance networks (HPNs) have emerged as a promising solution to big data movement and their significance has been well recognized in the broad science and network research communities [9]. An increasing number of HPNs have been deployed or are currently under development, including User Controlled Light Paths (UCLP) [1], UltraScience Net (USN) [2], On-demand Secure Circuits and Advance Reservation System (OSCARS) [3] of ESnet, and ION [4] of Internet2. Especially, the emerging Software-Defined Networking (SDN) technologies greatly facilitate HPN deployments, and many HPNs have incorporated SDN capabilities into their network infrastructures.

In general, there are two types of bandwidth scheduling, i.e. instant scheduling and periodic scheduling. The former is executed immediately for each incoming user request, while the latter is typically invoked periodically in a certain interval (i.e. scheduling period) to schedule a number of user requests accumulated during one scheduling period.

The majority of the existing work in this field is focused on instant scheduling [8]. Periodic scheduling for multiple reservation requests has attracted more attention in recent years due to the advantages in improving network utilization and meeting diverse requests from a global perspective. In [7], Li *et al.* proposed a new max-flow based greedy algorithm (GOS) to minimize the completion time for a single file transfer and a linear programming based algorithm (BATCH) to find an earliest-finishing schedule for a batch of file transfers. In [5], Divakaran *et al.* explored the problem of bandwidth sharing in data center networks to bring down the number of rejected requests through specifying flexible bandwidth demands. Particularly, Sharma *et al.* proposed a polynomial-time RRA algorithm to accommodate as many deadline-constrained reservation requests as possible between a pair of end sites while minimizing the total time needed to complete the data transfers [10]. This scheduling problem considers two optimization objectives: (i) maximize the number of satisfied requests, and (ii) minimize the total data transfer time required to meet all satisfied requests. However, the second objective does not really reflect the quality of bandwidth scheduling since it may yield a small number of satisfied requests.

In this paper, we consider a periodic bandwidth scheduling problem to maximize the number of satisfied user requests for bandwidth reservation with deadline constraint, referred to as multiple deadline-constrained bandwidth scheduling (M-DCBS). We show that this problem is NP-complete and propose a maximum user number resource reservation algorithm (MUNRRA). Extensive simulation results illustrate that MUNRRA has a superior performance over existing algorithms in terms of scheduling success ratio and execution time. Considering the increasing popularity of the DCBS-based service model and the rapid expansion of SDN-based HPNs in both speed and scope, the proposed scheduling algorithm has great potential to improve the resource utilization of network infrastructures and the network performance of many large-scale applications in a wide spectrum of domains with a need for big data transfer.

The rest of this paper is organized as follows. Section II presents the mathematical models and defines the scheduling problem under study. Section III details the algorithm design. Simulation-based comparative performance evaluation is conducted in Section IV. Section V concludes our work with a sketch of future plans.

DOI 10.1109/LCN.2016.48

II. MATHEMATICAL MODELS AND PROBLEM DEFINITION

In this section, we construct cost models and provide a formal definition of the M-DCBS problem.

A. Mathematical models

An HPN maintains a Time Bandwidth List (TBL) of residual bandwidths in the form of a 3-tuple of time-bandwidth (TB) $(t_P[i], t_P[i+1], b_P[i])$ to denote the residual/available bandwidth $b_P[i]$ of path P during time unit $(t_P[i], t_P[i+1])$, $i = 0, 1, 2, ..., T_P - 1$, where T_P is the total number of time-units on path P. When $i = 0$, $t_P[0]$ refers to the initial time point. For $t > t_P[T_P]$, we set the residual bandwidth of path P to be its full capacity as there is no bandwidth reserved on path P after $t_P[T_P]$.

HPNs may support different types of bandwidth reservations or service models to meet different data transfer needs. In this paper, we consider Deadline-Constrained Bandwidth Reservation Request (DCBRR), represented by (δ, B_{max}, T_S, T_E), where δ and B_{max} denote the data size to be transferred and the maximum bandwidth of the Local Area Network (LAN) connecting the edge router to the end host, and T_S and T_E denote the earliest possible data transfer start time and the end time by which the data must be transferred completely (i.e deadline).

Definition 1. Time Step Rectangle (TSR): TSR is defined as the longest time interval, within which the residual bandwidth of path P remains fixed.

A TSR_i is in the form of (TSR_i^s, TSR_i^e, TSR_i^b), where TSR_i^s, TSR_i^e and TSR_i^b denote the start time, end time and available bandwidth of the corresponding time interval of TSR_i, respectively.

Definition 2. Accommodation Region Rectangle (ARR): For each TSR, ARR is defined as the longest time interval during which the residual bandwidth is no less than that of TSR.

Similar to TSR_i, an ARR_i is in the form of (ARR_i^s, ARR_i^e, ARR_i^b), where ARR_i^s, ARR_i^e, and ARR_i^b denote the start time, end time and available bandwidth of the corresponding time interval of ARR_i, respectively.

B. M-DCBS Problem Definition

We provide a formal definition of the Multiple Deadline-Constrained Bandwidth Scheduling problem, referred to as M-DCBS, as follows:

Definition 3. M-DCBS: Given an established network path with a TBL of residual or available bandwidths specified as a segmented constant function of time, and k DCBRRs, our goal is to find a schedule that reserves the bandwidths of the network path for the given requests such that the number of satisfied DCBRRs is maximized.

To satisfy a DCBRR, a successful schedule should determine two time points t_s and t_e, and a fixed bandwidth b such that the data is completely transferred within the time slot $[t_s, t_e]$ at the data rate of b, where $t_s \geq T_S$ is the time point when the data transfer starts, $t_e \leq T_E$ is the time point when the data transfer ends, and $b \cdot (t_e - t_s) >= \delta$.

Once we obtain the i-th satisfied DCBRR, we need to update the time-bandwidth list of the path by subtracting the reserved bandwidth from the currently available bandwidth from the start time t_s^i to the end time t_e^i. Obviously, the high dynamics and large variations in the TBL would make it very challenging to find a feasible and efficient schedule, and the following four conditions would add more complexities to the problem: (i) different users may have different T_S and T_E, (ii) different reservations may have different data sizes δ to be transferred, (iii) different users may reside in different edge networks (which are all connected to the same reserved long-haul path in the backbone) and hence may have different B_{max}, and (iv) a different number of DCBRRs are accumulated in a different scheduling period. Obviously, depending on the availability of network resources, the system may or may not be able to satisfy all the requests. Our goal is to design a periodic bandwidth scheduling algorithm to maximize the number of satisfied bandwidth reservation requests under the current path and user statuses, including the time-bandwidth list of the path and the DCBRRs accumulated in the current scheduling period.

The NP-completeness of M-DCBS can be readily established based upon the computational complexity analysis in [10]. Similar to the proof in [10], we can convert an arbitrary instance of a known NP-hard variation of the generalized assignment problem (GAP) to a special instance of M-DCBS in polynomial time. In fact, the proposed M-DCBS problem is a generalized version of the $simple - SMR^3$ problem where all reservation requests are restricted to have the same start and end time and the same maximum bandwidth that they can use [10]. Since the $simple - SMR^3$ problem is NP-complete, so is M-DCBS, according to the principle of proof by restriction [6].

III. ALGORITHM DESIGN

In this section, we design a Maximum User Number Resource Reservation Algorithm (MUNRRA) to solve M-DCBS.

A. MUNRRA

Since the objective of MUNRRA is to maximize the number of satisfied DCBRRs, it is favorable to schedule the requests using less bandwidth resources first. If there are multiple ARRs that can be used for each DCBRR, we select the ARR with the largest coefficient.

We define several notations and operations as follows:

- $R, R^{'}$: the initial set of DCBRRs, and the current set of reservation requests that have not been scheduled.
- r: the current DCBRR, which is being scheduled.
- arr: the current ARR, which is being scheduled.
- $nTasks$: the initial number of DCBRRs accumulated in the current scheduling period.
- $nARRs$: the number of ARRs from time point T_S to T_E of the request r.
- $MinBValue$: the minimum bandwidth between B_{max} and ARR_i^b.
- $ARFLAG$: the subscript of the current ARR with the largest coefficient.
- $coefficient$: the ratio of the bandwidth to the time-slot from ARR_i^s to ARR_i^e.

Algorithm 1 MUNRRA

Input: a path with a time-bandwidth list TBL, and a set R of k DCBRRs, each of which specifies (δ, B_{max}, T_S, T_E)
Output: the maximum number N of satisfied DCBRRs

$N = 0, nARRs = 0, ARFLAG = -1$;
$coefficient = 0$;
Sort all DCBRRs in R in an increasing order of their data size δ and schedule them in this order;
$R^{'} = R$;
for $(i = 0; i < nTasks; i++)$ **do**
  Call Algorithm 2 FARR (TBL, T_S, T_E) to find all ARRs from time point T_S to T_E of the request r and set $nARRs$ to be the number of found ARRs;
  if $(nARRs == 0)$ **then**
    break;
  for $(k = 0; k < nARRs; k++)$ **do**
    if $((arr_k^e - arr_k^s) \cdot MinBValue \geq \delta_i^r)$ **then**
      if $(arr_k^b/(arr_k^e - arr_k^s)) > coefficient)$ **then**
        $coefficient = arr_k^b/(arr_k^e - arr_k^s)$;
        $ARFLAG = k$;
  if $(ARFLAG \neq -1)$ **then**
    $b_i = min(arr_{ARFLAG}^b, B_{max})$
    $t_s^i = arr_{ARFLAG}^s$;
    $t_e^i = arr_{ARFLAG}^s + \delta_i/b_i$;
    $ARFLAG = -1$;
    $N++$;
    $R^{'} = R^{'} - r$;
    Update the TBL of the path from t_s^i to t_e^i;
  $coefficient = 0$;
return N.

The pseudocode of MUNRRA is provided in Alg. 1. The MUNRRA first sorts all DCBRRs in an increasing order of data size δ and schedule them in that order. For each DCBRR, MUNRRA finds all ARRs that can be used for data transfer and selects the ARR with the largest coefficient for scheduling. If MUNRRA does not find any ARR, the corresponding DCBRR can not be satisfied. Once MUNRRA confirms the ARR, it computes b as the minimum of ARR_i^b and B_{max}, t_s as ARR^s, and t_e as $(ARR^s + \delta/b)$, respectively. MUNRRA then updates the TBL of the path from t_s^i to t_e^i for each satisfied DCBRR. The time complexity of MUNRRA in the worst case is $O(|R| \cdot \log|R| + |R| \cdot |T|^2)$, where R is the number of user requests.

B. FARR Algorithm

FARR, whose pseudocode is provided in Alg. 2, is to find all ARRs from T_S to T_E of each DCBRR. It first computes all TSRs, and then searches to the right and then to the left for the start and end time, at which the bandwidth is no less than TSR^b for each TSR as an ARR. If the ARR is different from any one in the current set of ARRs, FARR stores the start time, the end time, and TSR^b in an array as a new ARR. The time complexity of FARR in the worst case is $O(|T|^2)$.

IV. Performance Evaluation

We conduct a simulation-based performance evaluation of the proposed scheduling algorithm over a network path with various bandwidth capacities and different numbers of accumulated DCBRRs. We compare the performance of the proposed MUNRRA algorithm with that of the existing algorithm, namely RRA in [10], in terms of scheduling success ratio and execution time.

Algorithm 2 FARR

Input: a path with a time-bandwidth list TBL, and T_S and T_E
Output: a set $ARRSet$ of ARRs

$StartTime = 0$;
$EndTime = 0$;
$Bandwidth = 0$;
$FLAG = 0$;
Find all TSRs from TS to TE and set $nTSRs$ to be the number of found TSRs;
for $(i = 0; i < nTSRs; i++)$ **do**
  $Bandwidth = tsr_i^b$;
  $StartTime = tsr_i^s$;
  $EndTime = tsr_i^e$;
  for $(j = tsr_i^e; j <= T_E; j++)$ **do**
    $Endtime = j$;
    if $(Bandwidth > t_l[j]$ or $Bandwidth == 0)$ **then**
      break;
  for $(k = tsr_i^s; k >= T_S; k--)$ **do**
    $StartTime = k$;
    if $(Bandwidth > t_l[k]$ or $Bandwidth == 0)$ **then**
      break;
  $TempARR.Starttime = Starttime$;
  $TempARR.Endtime = Endtime$;
  $TempARR.B = Bandwidth$;
  for $(m = 0; m < nARR; m++)$ **do**
    if $(TempARR.Starttime == ARR_m.Starttime$ and $TempARR.Endtime == ARR_m.Endtime$ and $TempARR.B == ARR_m.Bandwidth)$ **then**
      $FLAG = 1$;
  if $(TempARR.B! = 0$ and $FLAG \neq 1)$ **then**
    Store ARR in an array $ARRSet$ with $StartTime$, $EndTime$ and $Bandwidth$;
  $FLAG = 0$;
return $ARRSet$.

A. Simulation Setup

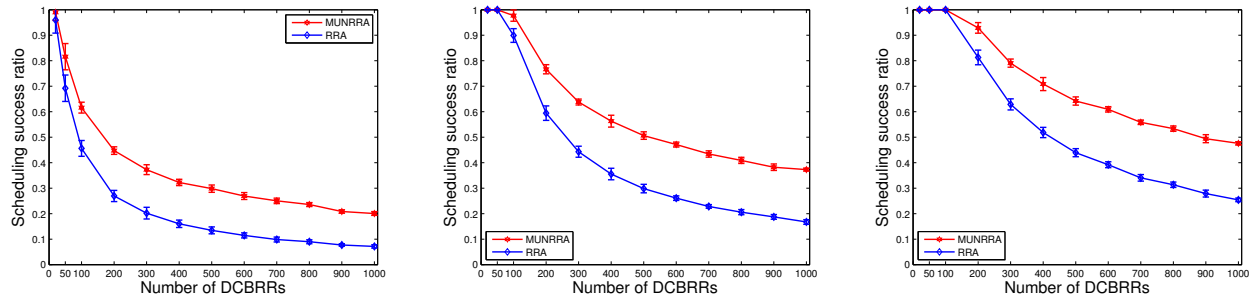
In the simulation, we consider a set of initial bandwidths of a network path, i.e. 20Gbps, 60Gbps, and 100Gbps, and schedule a number of randomly generated DCBRRs accumulated in each scheduling period of $[0, 60s]$, in which their arrivals follow a Poisson distribution. For a randomly generated DCBRR (δ, B_{max}, T_S, T_E), we set δ, B_{max}, T_S, and T_E to be a random integer, which is less than 500GBytes, 5Gbps, 60m, and 60m, respectively, and follows a uniform distribution, where $T_E > T_S$ and $(T_E - T_S) \cdot B_{max} \geq \delta$. All the algorithms are implemented and executed on a PC equipped with Intel(R) Core(TM) i3-2350M and 6GB memory to process the same batch of DCBRRs in each set of simulations. The simulations run across contiguous one-hour scheduling periods for about one week.

We define the scheduling success ratio ssr as the number of satisfied requests divided by the total number of requests:

$$ssr = \frac{\text{the number of satisfied DCBRRs}}{\text{the total number of DCBRRs}}.$$

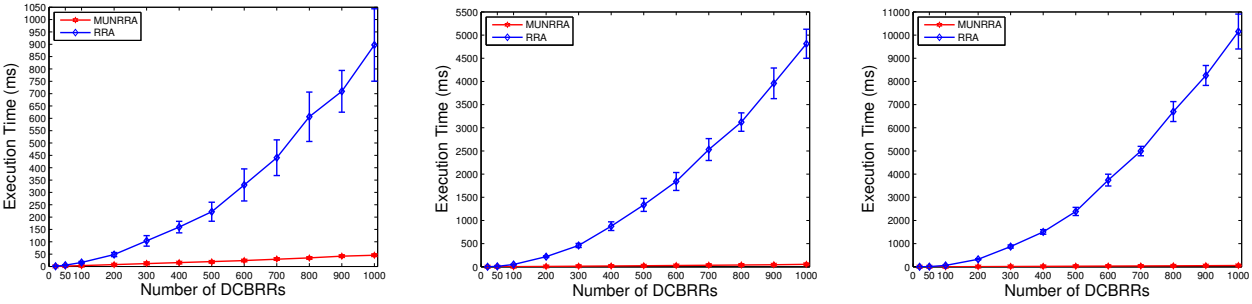
B. Performance Analysis

We collect two performance measurements: (i) scheduling success ratio, and (ii) execution time. We plot these average measurements with their corresponding standard deviations obtained by MUNRRA and RRA with different numbers of DCBRRs under different initial bandwidth capacities in Figs. 1(a) – 1(c) and Figs. 2(a) – 2(c), respectively. The performance measurements of MUNRRA has smaller standard deviations, indicating its higher robustness than RRA.



(a) (b) (c)

Figure 1. Comparison of scheduling success ratio of MUNRRA and RRA under an initial path capacity of (a) 20Gbps, (b) 60 Gbps, and (c) 100Gbps.



(a) (b) (c)

Figure 2. Comparison of execution time of MUNRRA and RRA under an initial path capacity of (a) 20Gbps, (b) 60 Gbps, and (c) 100Gbps.

These simulation results show that MUNRRA consistently outperforms RRA in all the cases we studied. In particular, MUNRRA improves the scheduling success ratio by more than 20% over RRA on average. In general, the more intensive the network resource contention is, the more advantage MUNRRA exhibits: under the same initial path bandwidth capacity, more user requests lead to higher performance improvement; while with the same set of requests, smaller initial path bandwidth capacities lead to higher performance improvement, as clearly illustrated by the performance curves in Figs. 1(a) – 1(c). We also observe that MUNRRA takes much less time than RRA to process the same batches of DCBRRs. The average execution time of MUNRRA is about 200 times faster than that of RRA, as illustrated by the performance curves in Figs. 2(a) – 2(c).

V. Conclusion

We formulated an NP-complete periodic bandwidth scheduling problem, M-DCBS, to maximize the number of satisfied deadline-constrained bandwidth reservation requests in SDN-based high-performance networks. The extensive simulations showed that the proposed algorithm for this problem has the best overall scheduling performance and execution time statistics in comparison with the existing algorithm.

It would be of our future interest to incorporate the proposed scheduling algorithm into the control plane of real-life high-performance networks connecting the Data Transfer Nodes deployed at collaborating national laboratories.

Acknowledgment

This research is sponsored by U.S. National Science Foundation under Grant No. CNS-1526134 and Department of Energy's Office of Science under Grant No. DE-SC0015892 with New Jersey Institute of Technology, and by National Nature Science Foundation of China under Grant No. 61472320 with Northwest University, P.R. China.

References

[1] UCLP: User Controlled LightPath Provisioning. http://www.uclp.ca, 2011.

[2] USN: DOE UltraScience Net. http://www.csm.ornl.gov/ultranet/, 2015.

[3] ESnet: On-demand Secure Circuits and Advance Reservation System. http://www.es.net/engineering-services/oscars/, 2013.

[4] Internet2 Interoperable On-Demand Network (ION) Service. http://www.internet2.edu/ion.

[5] D. Divakaran, M. Gurusamy, and M. Sellamuthu, "Bandwidth allocation with differential pricing for flexible demands in data center networks," *Computer Networks*, pp. 84–97, 2014.

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

[7] Y. Li, S. Ranka, and S. Sahni, "In-advance path reservation for file transfers in e-science applications," *The Journal of Supercomputing*, vol. 59, no. 3, pp. 1167–1187, 2012.

[8] Y. Lin and Q. Wu, "Complexity analysis and algorithm design for advance bandwidth scheduling in dedicated networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 1, pp. 14–27, 2013.

[9] N. Rao, Q. Wu, S. Ding, S. Carter, W. Wing, A. Banerjee, D. Ghosal, and B. Mukherjee, "Control plane for advance bandwidth scheduling in ultra high-speed networks," in *Proceedings of INFOCOM*. IEEE, 2006, pp. 1–5.

[10] S. Sharma, D. Katramatos, and D. Yu, "End-to-end network QoS via scheduling of flexible resource reservation requests," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–10.