

# Channel Provisioning and Flow Scheduling in Grid Overlay Networks

Dinil Mon Divakaran, Pascale Vicat-Blanc Primet

INRIA, LIP, ENS Lyon, France

Email: {Dinil.Mon.Divakaran, Pascale.Primet}@ens-lyon.fr

**Abstract**— Grid traffic characteristics are quite different from that of normal Internet traffic, as they impose more stringent constraints (such as packet delay and flow deadlines) on network resources. This paper explores differentiated channel provisioning in grid network, in order to schedule bulk data traffic. We propose a solution based on the two-phase routing scheme to provision the virtual channels, such that it will satisfy any valid traffic matrix. Then we detail a simple, online, preemptive algorithm for scheduling bulk data traffic.

## I. INTRODUCTION

An interesting and challenging problem in grid networks that hasn't received as much attention as it deserves is, *channel provisioning* and *data traffic scheduling* so as to improve network and application performances. From a user perspective, this means that the deadline-constrained data transfer tasks submitted, are able to meet the deadlines. From the perspective of a network operator, it means that the network is provisioned enough to meet the dynamic changes in network traffic, but still doesn't go underutilized.

Today, massive data sets are generated at different grid sites from scientific computing, high performance computer simulations, bioinformatics, satellites and so on. As the amount of data increases, more and more users tend to share and distribute them. Data movement tasks might be loosely (or even tightly) bound with job scheduling decisions; but, it has been shown that these two tasks can be efficiently addressed separately, thus decoupling computation and data scheduling in data-intensive applications [1]. Network bandwidth has been identified as one of important parameters that affect the scheduling of jobs in large scale distributed systems. Unfortunately, most often, there is no strategic scheduling or distributing scheme involved

for data transfers that takes care of dynamic network behaviours [2].

If the transfer of huge data sets generated in grid networks are constrained by deadlines, then the normal TCP transfer will not be of much help. This happens as TCP gives importance to fair sharing, and does not provide any guarantee on the completion time of transfer requests. Besides, for a single flow to sustain high flow rates, to saturate a high BDP (bandwidth delay product) link requires unrealistically low packet loss, while also incurring large, oscillatory queues [3].

Traffic in a grid network can be mainly classified into various types, which can be grouped into three main classes: bulk data, real-time (or control) data and the traditional best-effort data [4]. The first class of traffic demands throughput to meet the deadlines of data transfers, whereas the second class is more concerned about latency or delay. The third class is of least priority in the scenario of grid networks. The best-effort routing in the Internet is obviously not enough to take care of these requirements. The *transfer delay guarantee* feature required in the grid is what differentiates it from the 'best-effort' based network services. Instead, what is required is a multi-channel provisioning scheme, that provisions bandwidth in links such that, all the three different classes of traffic meet their demands.

In this paper, we focus not only on the problem of provisioning the virtual channels, but also on scheduling bulk data traffic on such a provisioned network. Once the network is provisioned to serve any valid traffic matrix, then the scheduling algorithm can aim at improving the acceptance ratio<sup>1</sup>.

The rest of the paper is organized as follows.

<sup>1</sup>Number of tasks accepted to the number of tasks arrived

The problem is described elaborately in Section II. In Section III, we use a two-phase routing scheme for provisioning virtual channels in grid networks. Scheduling of bulk data transfers is then discussed in Section IV. Related work is discussed in Section V. We conclude in Section VI.

## II. PROBLEM ANALYSIS

The problem in hand is: Given the three different kinds of traffic classes, how can we ensure that the links can serve the different traffic classes, without overloading the network? That is to say, we want to schedule the deadline-constrained bulk data transfers without causing any congestion in the network, at the same time route the real-time traffic without unpredictable delays. One possible solution is to dynamically route the traffic depending on the type of traffic. For bulk data traffic and real-time traffic, this means that the system should have information about link characteristics such as bandwidth and latency, which will require active measurements. Once such metrics are known, the system can take decisions on whether the user request can be met using the currently measured values or not.

But measuring these metrics is a difficult task. Measuring end-to-end bandwidth for a large network, of say  $N$  nodes, will result in generation of tremendous amount of traffic, as  $N^2$  such measurements have to be made in every measurement interval. Traffic matrix estimation using measurement techniques also gives errors of 20% or more [5].

Recent research works have come up with a static routing and bandwidth allocation scheme, which can cope up with dynamic traffic changes [6], [7]. These works propose a two-phase routing scheme, where traffic is initially split and routed to a set of intermediate nodes in the first phase, and from the intermediate nodes to the destination in the second phase. Depending on the ingress and egress constraints, the method proposes a pre-configuration of the network such that it satisfies any traffic matrix (respecting the ingress-egress capacities) of the network.

## III. PROVISIONING GRID TRAFFIC USING TWO-PHASE ROUTING

Consider a graph  $G(V, E)$ , of  $|V| = N$  nodes, and  $|E| = m$  edges. Let  $n_i$  represent node  $i$ , and

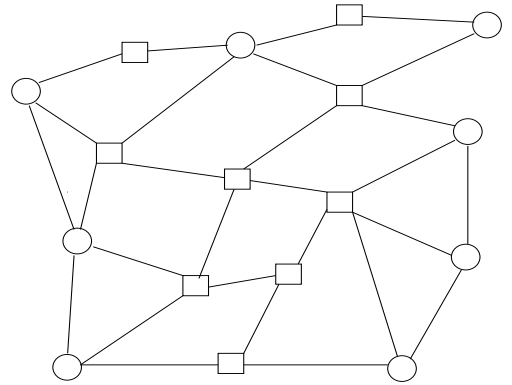


Fig. 1. An example showing a grid network. The circles represent the ingress-egress nodes, and the squares represent network elements connecting these nodes.

$e_{i,j}$  represent the link from  $n_i$  to  $n_j$ . A typical graph is shown in Fig. 1. The nodes in the graph represent the ingress-egress points connected to different sites that form a grid network.

A traffic matrix is represented by  $T = [t_{i,j}]$ , where  $t_{i,j}$  denotes the traffic from  $n_i$  to  $n_j$ . Let  $\rho_i$  represent the limit on the amount of traffic flowing into the network, as well as the traffic leaving the network at  $n_i$ . Before going to the details, we state our assumptions:

- 1) The access routers will be able to relay the traffic from other access routers.
- 2) Traffic matrices will be such that the inbound/outbound capacities of the nodes will be respected.

When it comes to bulk data, the second assumption will be enforced as part of the bulk data scheduling algorithm. But for real-time and best effort traffic classes, we assume that there is some policy in place, that makes sure that the inbound/outbound capacities are not violated.

We propose a solution based on the two-phase routing strategy. The core idea in the routing scheme is to split the traffic from a source node to a destination node, say from  $n_i$  to  $n_j$ , and send to all the nodes in the network, irrespective of the destination. This constitutes the first phase of routing. In the second phase, all the nodes route the traffic to the destination  $n_j$ . If the traffic from node  $n_i$  is split in equal ratios, then the traffic through the link  $e_{i,j}$  is  $\frac{\rho_i}{N}$  due the first phase routing. Since  $n_j$  can not receive more than  $\rho_j$  amount of traffic at any point in time, and since it receives this traffic in splits

of equal ratios from all the nodes, it will receive no more than  $\frac{\rho_j}{N}$  of the traffic through  $n_i$ . This essentially means that, the traffic through  $e_{i,j}$  due to second phase routing is  $\frac{\rho_j}{N}$ . Therefore, the total traffic through  $e_{i,j}$ ,

$$\lambda_{i,j} = \frac{\rho_i + \rho_j}{N} \quad (1)$$

As is obvious, every packet now takes a longer route to the destination. Hence the delay incurred will be longer, but bounded.

The grid traffic can be distinguished into three different classes as described earlier. If  $r_{i_1}$ ,  $r_{i_2}$  and  $r_{i_3}$  denote the rates of the bulk data traffic, real-time traffic and best effort traffic, respectively from node  $n_i$ , then,

$$\rho_i = r_{i_1} + r_{i_2} + r_{i_3} \quad (2)$$

The incoming traffic is also assumed to be split into  $r_{i_1}$ ,  $r_{i_2}$  and  $r_{i_3}$ , though this would still work for a different set of values. For convenience, let  $R_i$  represent  $r_{i_2}$ ; and let  $B_i$  represent the sum of bulk data and best effort traffic. That is,

$$B_i = r_{i_1} + r_{i_3} \quad (3)$$

We now look at an approach to route real-time traffic directly. That is, instead of splitting the traffic equally as well as treating them equally, we propose a split ratio such that, the real-time traffic is always routed directly. Let  $k_i = \frac{R_i}{\rho_i}$  denote fraction of the real-time traffic from  $n_i$ . The traffic from  $n_i$  is split such that,  $k_i \rho_i$  of traffic from  $n_i$  takes direct route to the destination, and the rest of  $(1 - k_i) \rho_i$  traffic is split equally among the  $N$  nodes.

Next, we once again compute the traffic demand through  $e_{i,j}$ . The traffic due to direct routing of real-time traffic from  $n_i$  to  $n_j$  is  $k_i \rho_i$ . The traffic due to first phase routing from  $n_i$  to any node  $n_k$  that is routed through  $n_j$  is  $\frac{1-k_i}{N} \rho_i$ . Similarly, node  $n_j$  will receive  $\frac{1-k_l}{N}$  of traffic destined to  $n_j$  from  $n_l$  through  $n_i$ . Therefore the total traffic it will receive through  $n_i$  due to second phase routing is  $\sum_{l=1}^N (\frac{1-k_l}{N}) t_{l,j}$ . Hence, the total traffic through  $e_{i,j}$  is,

$$\lambda_{i,j} = k_i \rho_i + \frac{1 - k_i}{N} \rho_i + \sum_{l=1}^N (\frac{1 - k_l}{N}) t_{l,j} \quad (4)$$

The last term in the above equation has an upper bound: the fraction of the non-real-time traffic due

to the second phase routing through  $e_{i,j}$  destined for  $n_j$ , which is  $\frac{B_j}{N}$ . Therefore, the equation in simplified readable form is,

$$\lambda_{i,j} = R_i + \frac{B_i + B_j}{N} \quad (5)$$

The total link capacities required in this case is,

$$L = (N - 1) \left[ \sum_i^N R_i + 2 \frac{\sum_i^N B_i}{N} \right] \quad (6)$$

If  $\forall i, k_i = k$ , then the above equation reduces to,

$$L = (2(1 - k) + kN) \left[ \sum_i^N \rho_i \right] \frac{N - 1}{N} \quad (7)$$

In [7], the authors prove that the minimum total link capacities required to serve any valid traffic matrix is  $2 \sum_i^N \rho_i - \max_i \rho_i$ . Comparing this with Eq. 7, the deviation from the minimum total bandwidth increases with the value of  $k$ . But depending on the traffic of real-time data, the value of  $k$  can be very small.

The traffic demand matrix,  $\Lambda = [\lambda_{i,j}]$ , which specifies the capacity required for every link in the network, is computed using Eq. 5. The knowledge of the upper bounds of the different traffic classes helps us to determine the bandwidth that have to be allocated between the nodes in the network. Once the traffic demand between the endpoints are computed, the network operator can provision the network accordingly. In such a provisioned network, real-time traffic will be routed directly with minimum delay. The bulk data traffic and best effort traffic will be routed in two phases. The best effort traffic can be rate-limited by enforcing a policy that will not only cap the rate of best effort traffic to the specified value of  $r_{i_3}$  for each node  $n_i$ , but will also allocate the unused share for bulk data transfer when not utilized completely.

The scheduling algorithm that follows in the next section is unaware of the traffic splitting and aggregation that happen at egress and ingress points, respectively. Instead of routing packets of a single flow through different routes (causing packet re-ordering at the destination), a better approach is to divide a flow into chunks of packets, and then route these chunks independently over different routes. Thereby only the chunks have to be re-ordered.

---

**Algorithm 1** ScheduleTransfer( $\tau$ )

**Require:** Task  $\tau = \{s$  (src),  $d$  (dst),  $v$  (volume),  $\eta$  (start time),  $\psi$  (end time) $\}$

```
1:  $min\_rate = \frac{v}{\psi - \eta + 1}$ 
2:  $t = \min(\eta, current\_time)$ 
3:  $time\_left = \psi - t + 1$ 
4:  $avail\_bw = find\_avail\_bw(t, \psi, s, d)$ 
5: if  $avail\_bw \geq min\_rate$  then
6:    $sched = allocate\_max\_rate(r)$ 
7: else
8:    $avail\_bw = push\_find\_avail\_bw(t, \psi, s, d)$ 
9:   if  $avail\_bw \geq min\_rate$  then
10:     $sched = push\_allocate\_max\_rate(r)$ 
11:   else
12:    reject request
13:   end if
14: end if
```

---

Splitting a huge flow into multiple small chunks, will also improve the link utilization. This happens as the *normalized aggregate TCP throughput* increases (approaches 1) as the number of flows increases [8].

#### IV. SCHEDULING BULK DATA TRAFFIC

In this section, we detail an algorithm for scheduling bulk data traffic over such a provisioned grid network. Since network flows following valid traffic matrices will not cause congestion in the network, the scheduling algorithm need not worry about congestion. Instead a simple online preemptive scheduling algorithm that maximizes the acceptance ratio of tasks can be used for bulk data transfer.

An algorithm to schedule as many requests as possible, by allocating maximum possible rates to transfer requests is given in Algorithm 1. Lines 1-3 compute the minimum rate required for the transfer, and also the maximum time left to transfer the data. Line 4 finds out the available bandwidth within the requested time, from the specified source node to the destination node. The schedule is obtained in Line 6, if the available bandwidth is enough for data transfer; or else *push\_find\_avail\_bw* is called to check if the tasks scheduled in the interval  $[\eta, \psi]$  (that transfers data from  $s$  to  $d$ ) can be preempted and postponed (pushed) to the future such that

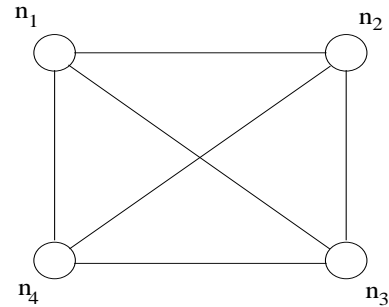


Fig. 2. An overlay network of four nodes.

their deadlines are still not violated. If so, the total available bandwidth achieved after *pushing* the minimum possible transfer tasks ahead is returned in line 8. If the available bandwidth  $\geq$  the minimum required rate, the schedule is obtained by calling *push\_allocate\_max\_rate* in line 10, or else the request is rejected in line 12. The schedule returned by the functions are of the format  $\{n_l, n_k, ([t_1, t_2], r_1), ([t_3, t_4], r_2), \dots\}$ , where the rate  $r_i$  is specified for each interval  $[t_j, t_{j+1}]$  for transfer of data from  $n_l$  to  $n_k$ . The length of the interval can be greater than or equal to 1 unit time. Note that a task can be rescheduled in future to incorporate other tasks, but with a guarantee to meet its deadline.

We illustrate the algorithm using a simple example. Consider the overlay network of Fig. 2 that has been provisioned after computing the demand matrix. Assume that each node is able to send and receive bulk data traffic of 800 Mbps. The transfer requests are given in Table I.

The first task requests for a transfer of 300 Mb (Megabits) from  $n_1$  to  $n_3$ . Since there is no transfer active during the requested time, the schedule returned by the algorithm for task  $\tau_1$  is  $\{n_1, n_3, ([2,2], 300)\}$ . The schedule for  $\tau_2$  will be  $\{n_1, n_3, ([3,3], 800) ([4,4], 400)\}$ . The task  $\tau_3$  that arrives at  $t = 3$  can not be scheduled unless the transfer of  $\tau_2$  at  $t = 4$  is preempted and postponed to  $t = 5$ . This does not affect the deadline of  $\tau_2$ . Therefore,  $\tau_2$  is preempted, and the new schedule for  $\tau_2$  is  $\{n_1, n_3, ([5,5], 400)\}$ . The schedule for  $\tau_3$  will be  $\{n_1, n_3, ([4,4], 800)\}$ . Next, task  $\tau_4$  will obtain the schedule  $\{n_2, n_4, ([5,6], 800)\}$ .  $\tau_5$  also requests for a transfer of 450 Mb of data to  $n_4$  at  $t = 6$ . Since  $n_4$  will receive data at its maximum inbound capacity at  $t = 6$ , the algorithm can not schedule  $\tau_5$ , and therefore will reject it.

TABLE I  
BULK DATA TRANSFER REQUESTS

Task No.	Arrival time	src (s)	dst (d)	Volume (Mbits)	Start time( $\eta$ )	End time( $\psi$ )
$\tau_1$	1	$n_1$	$n_3$	300	2	5
$\tau_2$	2	$n_1$	$n_3$	1200	3	7
$\tau_3$	3	$n_1$	$n_3$	800	4	4
$\tau_4$	4	$n_2$	$n_4$	1600	5	6
$\tau_5$	5	$n_1$	$n_4$	450	6	6

## V. RELATED WORK

There has been a wealth of research work on provisioning bandwidth in VPN hose models. For symmetric inbound/outbound capacities, and finite link capacities, the problem of optimal single-path routing and tree routing have been shown as NP-hard [9]. In [10], an optimal polynomial-time algorithm for computing bandwidth reservations at minimum cost using multi-path routing is presented. Whereas, [6] and [7] proposed the two-phase routing scheme discussed earlier.

Jones et al. proposed methods to improve the inter-cluster network performance by using bandwidth-aware co-allocating meta-schedulers, for a network with known topology and predictable performance characteristics [11]. Zang et al. presented mechanisms based on traffic prediction, rate-limiting and priority based adaptation to improve the quality of data transfer in the Internet. A concept of agreement was used that would guarantee transfer of data within a specific period, with a certain level of confidence [12]. In [13], the authors investigate network bandwidth sharing in datagrids. They consider a specific topology with a well-provisioned core network, the maximum transmission rate limited only at the end hosts. The off-line scheduling problem is formulated with the objective of maximizing acceptance ratio and network utilization by manipulating both the start time and transfer rate. The problem of transferring huge amounts of data in grid networks was handled exclusively in [14]. Bulk data transfer scheduling, proposed in this paper, assigns a multi-interval bandwidth allocation profile for each task, such that the network congestion is minimized.

## VI. CONCLUSIONS

In this paper, we introduce new ideas for solving both the problems of *virtual channel provisioning*

and *flow scheduling* in grid networks. In future, we want to implement the proposed solutions and study the various practical problems that might arise due to splitting and aggregation of chunks, preemption, relaying of traffic classes etc. We also plan to do an analysis study before deciding on the transport protocol for transferring data; a protocol such as *high speed SCTP* might be a good option.

## REFERENCES

- [1] K. Ranganathan and I. Foster, "Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications," in *HPDC '02: Proceedings of the 11th IEEE Int'l Symposium on High Performance Distributed Computing*, 2002, p. 352.
- [2] S. Al-Kiswany, M. Ripeanu, A. Iamnitchi, and S. Vazhkudai, "Are P2P Data-Dissemination Techniques Viable in Today's Data-Intensive Scientific Collaborations?," in *Euro-Par*, Aug. 2007, pp. 404–414.
- [3] A. Falk, T. Faber, J. Bannister, A. Chien, R. Grossman, and J. Leigh, "Transport Protocols for High Performance," *Commun. ACM*, vol. 46, no. 11, pp. 42–49, 2003.
- [4] P. Vicat-Blanc Primet and J. Zeng, "Traffic isolation and network resource sharing for performance control in grids," in *ICAS/ICNS*, Oct. 2005, p. 87.
- [5] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: existing techniques and new directions," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 161–174, Aug. 2002.
- [6] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Efficient and robust routing of highly variable traffic," in *Third Workshop on Hot Topics in Networks (HotNets-III)*, Nov. 2004.
- [7] Zhang-Shen Rui and Nick McKeown, "Designing a Predictable Internet Backbone with Valiant Load-Balancing.," in *IWQoS*, 2005, pp. 178–192.
- [8] L. Qiu, Y. Zhang, and S. Keshav, "On Individual and Aggregate TCP Performance," *ICNP*, vol. 00, pp. 203, 1999.
- [9] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, "Provisioning a virtual private network: a network design problem for multicommodity flow," in *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, 2001, pp. 389–398.
- [10] T. Erlebach and Maurice Rügge, "Optimal Bandwidth Reservation in Hose-Model VPNs with Multi-Path Routing," in *INFOCOM*, Mar. 2004.
- [11] W. M. Jones, L. W. Pang, W. B. Ligon, and D. Stanzione, "Bandwidth-aware co-allocating meta-schedulers for mini-grid architectures," in *CLUSTER '04: Proceedings of the 2004 IEEE Int'l Conf. on Cluster Computing*, 2004, pp. 45–54.
- [12] H. Zhang, K. Keahey, and W. Allcock, "Providing Data Transfer with QoS as Agreement-Based Service," in *SCC '04: Proceedings of the 2004 IEEE Int'l Conf. on Services Computing*, Aug. 2004, pp. 344–353.
- [13] L. Marchal, P. Vicat-Blanc Primet, Y. Robert, and J. Zeng, "Optimal Bandwidth Sharing in Grid environment," in *IEEE HPDC 15th IEEE Int'l Conf. on High Performance Distributed Computing*, June 2006.
- [14] Bin Bin Chen and P. Vicat-Blanc Primet, "Scheduling deadline-constrained bulk data transfers to minimize network congestion," *Seventh IEEE Int'l Symposium on Cluster Computing and the Grid (CCGrid 07)*, vol. 0, pp. 410–417, May 2007.