EuroFGI Workshop on IP QoS and Traffic Control
P. Pereira (Ed.)
Lisbon, Portugal, December 6-7, 2007

SCHEDULING NETWORK RESERVATIONS WITH A FLEXIBLE AND RELIABLE QOS MECHANISM IN GRIDS[1]

Kashif Munir*, Michael Welzl* and Somera Javed**

* Institute of Computer Science, University of Innsbruck, Austria
e-mails: kashif.munir@uibk.ac.at, michael.welzl@uibk.ac.at

** National University of Computer and Emerging Sciences, Islamabad, Pakistan
e-mail: somera11@gmail.com

Keywords: Grid, QoS, Advance Reservation, TCP, Resource Broker, UDT.

Abstract. *Bulk data transfers in Grids require end-to-end performance guarantees as well as a reliable and efficient transfer mechanism. The specific traffic requirements and environment conditions need to be taken into account as the focus of resource utilization is shifting from power to network resources in data Grids. This article investigates network resource sharing in Grids, especially data Grids. The scheduling of advance resource reservations in Grids was proven to be NP-complete. The existing heuristics for the scheduling of bandwidth requests do not take communication and computation delays and communication overheads into account which are involved in the reliable transfers of such reservations. This paper proposes a flexible, reliable and realistic QoS Mechanism which maximizes acceptance rate and network resource utilization and demonstrates the performance improvement by documenting results of simulations.*

1 INTRODUCTION

Grid computing enables the virtualization of distributed computing and data resources such as processing, storage capacity and network bandwidth to provide a user with a unified view of the system. It is therefore a major effort in Grid computing to hide some of the complexity from the programmers of Grid applications, which requires mechanisms to be in place for automatically distributing parts of applications – so-called "schedulers", which work best if the underlying system exhibits a deterministic behavior. This can be attained by reserving resources such as CPUs and memory on machines ("Advance Reservation"); the underlying connection infrastructure being the Internet (or a specific part thereof), fully deterministic behavior can only be seen if such reservations include the network. These reservations have properties which make them somewhat different from the classical per-flow guarantees that have been demanded for multimedia services – the service may not be used immediately after its reservation and the flows are elastic.

Realizing such per-flow QoS guarantees is not easy. Even when fine-grain QoS mechanisms like IntServ/RSVP would be available, providing them is an effort for an ISP, meaning that it will not be done for free. On the other hand, differentiating between a protected traffic aggregate and "all other traffic" is much easier, and can for instance be done by switching a pre-configured type of traffic (with classification via the DSCP, for instance) onto a leased line with MPLS or by treating it as "Expedited Forwarding" (EF) traffic with DiffServ.

[1] The work described in this paper is partially supported by the Higher Education Commission (HEC) of Pakistan under the doctoral fellowship program for Austria and through the FP6-IST-507613 project Euro-FGI.

In order to guarantee fine-grain QoS, traffic within the protected aggregate must be controlled – but, rather than involving routers, this can be done at the end systems by communicating with a Resource Broker (a common service in Grids where one can, for instance, request a machine with a certain CPU power; our intention is to extend this element with the ability to grant Advance Network Reservation).

There are two major issues related to quality of service (QoS) enforcement for Grid applications [1]. First, there is the separation of bulk data transfers from the other Grid traffic like SOAP and MPI messages. Second, there is the high speed transfer of bulk data because TCP reacts poorly to bulk transfer in large pipes. While a lot of work has been done for scheduling of computational resources, our focus is on incorporating network resource management into Grid environments. Per-flow bandwidth allocation is necessary and realistic in high-end Grid networks, which are characterized by high QoS requirement and low multiplexing level.

The literature will be surveyed in the next section. We explain how our mechanism works in Section 3, and support our explanations with simulation results in Section 4. Section 5 concludes.

2 RELATED WORK

Early work on advance reservation focused on reservation protocols like RSVP [2] and routing algorithms for networks with advance reservations [3]. Grid applications need guarantees of Quality of Service (QoS) [4,5]. Targeting deadline support for bulk data transfers, the problem of network resource reservation [6] has been proposed to be studied within the grid scope. An example for a Grid toolkit that supports such mechanisms is Globus with its GARA resource allocation component [7]. The issue of bandwidth fragmentation is discussed by Burchard et al. [8]. Bandwidth fragmentation may reduce acceptance percentage of requests arriving later. They propose the idea of malleable reservation to address the problem for which a start time and single rate value can be selected from a range of values.

For malleable requests in [10], the method of [8] or [9] is used to adjust the bandwidth or duration to satisfy the requester. However for a fixed request in [10], the only way to avoid being rejected is to adjust the bandwidth of admitted malleable requests. The trouble with this scheme is the extra overhead in finding and adjusting the admitted reservations which may be modified. The Multi-Interval scheme, presented in [11], avoids this trouble. The scheme is based on the concept that a request should not be rejected if there is at least one feasible solution to accept it. If there are multiple solutions, the one which yields the minimum flow time is chosen and is not changed after that.

In [12] a general view of the network resources sharing in Grids and Grids traffic isolation is presented. Optimization of bandwidth sharing among Grid flows is given [1] by manipulating the transmission windows of the flexible requests between minimum and maximum rates. The formulated optimization problem is proven to be NP-complete.

Two types of strategies for scheduling bulk data transfers are possible [13]. One strategy is to immediately grant or reject admission to a reservation request on its arrival time. In the other strategy, if a reservation request can not be granted or rejected at the time of its arrival, it is put in a queue to explore its possible admission later. Our mechanism is based on the latter strategy.

A time-slot based approach for scheduling the elastic and streaming requests is described in [14]. However, the effect of the extra signaling overhead, which is due to the manipulation of the data transfer rates of individual flows, is not taken into account in this approach.

All the above approaches assume loss-free networks and no computation and communication overhead of admission control. Our mechanism is reliable and realistic in which the residual network capacity is opportunistically, quickly and fairly shared by all existing flows, which minimizes flows completion times and maximizes acceptance of reservation requests in the network. Our mechanism includes all communication and computation delays and overheads into account.

Common admission control schemes assume all flows to use a certain fixed rate. It is a key feature of our mechanism that it manages to efficiently utilize network resources in a scalable manner because flows opportunistically increase their rates as bandwidth becomes available. This is attained by using a high-speed congestion control mechanism for all end-to-end flows; moreover, we use a mechanism that is

designed for high-speed networks (networks with a large bandwidth-delay product), where standard TCP congestion control is known not to yield satisfactory performance. Because it is designed for high-speeds and particularly convenient in a Grid setting, we chose UDT [15] for our mechanism, but stress that any max-min fair congestion control scheme could be used in its place.

3 THE QOS MECHANISM

3.1 Architecture

Admission control is a central element of our architecture; we provide QoS solely by controlling what enters the network. Our Resource Broker based architecture is shown in figure 1. We require no prioritization and therefore no state in routers; instead, we use a high-speed, stable, fair and end-to-end congestion control mechanism like UDT. Admission control in our mechanism is different from the traditional Bandwidth Broker in DiffServ as it provides per-flow end-to-end guarantees to a Grid flow keeping in view its deadline constraints. Also the Resource Broker element which is used for admission control does bandwidth monitoring besides bandwidth brokerage, which means that a flow can be admitted dynamically at any time depending on the current availability of resources as well as its own deadline and average rate that is needed for meeting it (we call this the "Average Required Rate (ARR)").

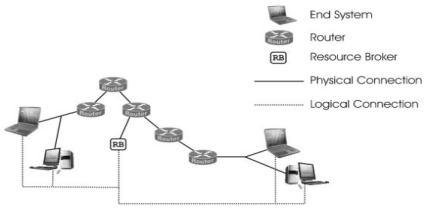

Figure 1. Resource Broker based architecture showing logical and physical flows

As shown in figure 2, the Resource Broker maintains an internal repository. The repository contains the global topology of the Grid as well as information about all flows currently in the network and the link capacities allocated for Grid traffic. The Resource Broker, installed as a separate server on any node in the Grid network, is a software instance used to manage resource reservation requests and to control the access to the Grid for each flow. It handles two types of requests:

1. *Resource Reservation Request:* The Resource Broker gets a request having a start time, its ARR and the duration of the transfer. The Resource Broker checks the availability of the requested resources, permits or denies access for that flow, and finally saves the information in the repository. In case it does not permit access to a flow, it adds it in a waiting queue and continuously monitors the bandwidth till it becomes possible to admit the flow. If the bottleneck capacity becomes less than the ARR or the termination time of the flow becomes equal to the current time, the Resource Broker finally rejects the admission of the flow. The flows in the waiting queue are checked periodically for possible admission in the network.
2. Cancel Request: When the broker gets a cancel request it updates the network information by relinquishing the resources owned by the flow.

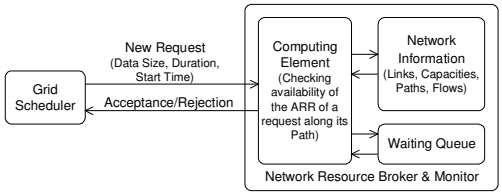

Figure 2. High level architecture of the QoS mechanism

3.2 Operation

The Resource Broker admits a flow with its average required rate of x bits per second to make it possible for it to meet its deadline. After admission, the rates of flows start increasing according to a max-min fair congestion control scheme in such a way that at any time the rate of any flow does not go below its ARR. In a Grid we can also exploit the knowledge that deadlines are sometimes known in advance, and it is important to have a network Resource Broker which can reserve flows in advance. The admission and termination of a flow is controlled through the Resource Broker residing on a node in the network and by having a Sender – Resource Broker signaling mechanism. Note that we only assume a single node for the sake of simplicity; our architecture is scalable, as distributing the Resource Broker with a scheme as in [16] would not change anything about it.

When a request is served, the sender sends a termination message to the RB. This message passing takes only a few milliseconds on average, which is quite negligible as compared to a typical Grid flow transfer time in which a huge amount of data is transferred. In the simulations the FTP application protocol is used over the UDT high-speed data transfer protocol.

The flow chart of the QoS mechanism is shown in figure 3.

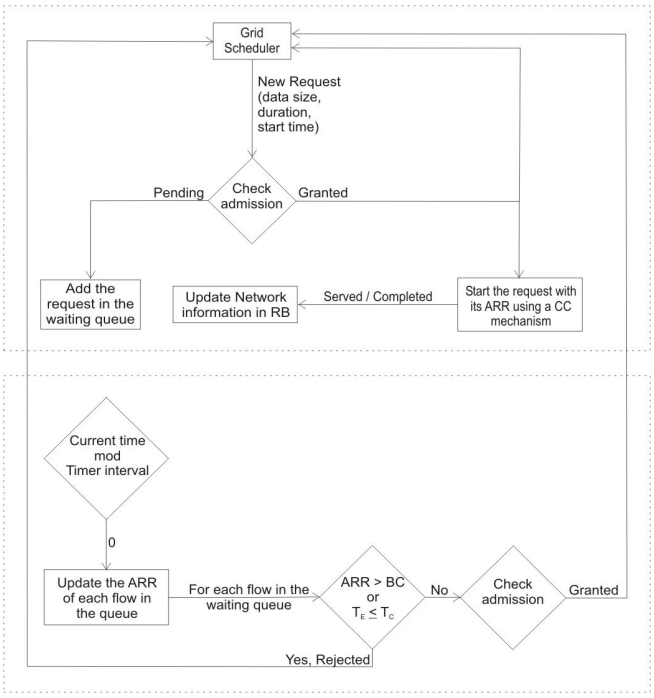

Figure 3. Flow chart of the QoS mechanism. In the figure, CC stands for Congestion Control, BC is the Bottleneck Capacity; T_E is the deadline of a request and T_C is the current time.

The Admission Control Algorithm of for the RB is given below.

D_S, D_F, T_S, ARR, ID, R_T: data size, duration, start time, average required rate, ID and reservation type of the request/flow for which a reservation is requested
$R_T \in$ {IR, AR}: IR = immediate reservation and AR = advance reservation
Record of a request/flow: {R_T, T_S, T_E, D_F, D_S, ARR, ID}
***Φ:** Set of records of the currently accepted requests sharing the bottleneck link*

ψ: (Waiting Queue) Set of those requests which do not get acceptance right at the time of their arrivals; the admission of requests from this set is based on a greedy acceptance policy i.e., after every periodic interval of time, any request whose deadline can be met is granted admission however the checking of admission of the requests is prioritized on the basis of their arrival times.
C_T: The total capacity of the bottleneck link
T_C: The current time
T_i: The timer interval

```
Procedure ARR_CC_Queue(Network_Topology_Information)

While (All requests are processed)

        If (a new reservation is requested)
                ARR = ⌈D_S/D_F⌉
                ID = {generate ID for the new request}
                If (Admission(Φ,ID,ARR,R_T,D_F,D_S,T_S,T_C,C_T) = YES) Then
                        {Accept the request by sending an acceptance message to the
                        sender and start the flow with its ARR at its start time using a
                        max-min fair Congestion Control protocol}
                Else
                        ψ = ψ + flow
                        // Insert the request in the waiting queue
                        // flow = flow_record(R_T,T_S,T_E,D_F,D_S,ARR,ID)
        If (a served request is completed) Then
                Termination(Φ,ID)
End While

End Procedure

Procedure Admission (Φ,ID,ARR,R_T,D_F,D_S,T_S,T_C,C_T)

Set C_R to 0
// C_R is the reserved capacity

If (R_T = IR) Then
        T_S = T_C
        T_E = T_C + D_F
        // T_E is the End Time of a flow
Else
        // R_T = AR; this is an Advance Reservation
        T_E = T_S + D_F

For each flow ∈ Φ
        If ((Φ(flow).T_S < T_E) AND (Φ(flow).T_E > T_S)) Then
                C_R = C_R + Φ(flow).ARR
End For

If (C_T - C_R) > ARR Then
        Φ = Φ + flow    // admit the flow
        // flow = flow_record(R_T,T_S,T_E,D_F,D_S,ARR,ID)
        Return "Yes"
Else
        Return "No"

End Procedure

Procedure Termination (Φ,ID)

For each flow ∈ Φ
        If (Φ(flow).ID = ID) Then
                Φ = Φ  - flow
                Break
End For

End Procedure
```

```
Procedure Periodic_checking_of_the_waiting_queue(ψ,Φ)

If (T_C % T_i = 0)
        For each flow ∈ ψ        // remove flows which cannot be admitted
                If (ψ(flow).T_E ≤ T_C)
                        ψ = ψ - flow
                        // Reject the request by sending a rejection message to
                        // the sender
                If (ψ(flow).T_S ≤ T_C)
                        ψ(flow).T_S = T_C
                        ψ(flow).D_F = ψ(flow).D_F - T_i
                        ψ(flow).ARR = ⌈ψ(flow).D_S/ψ(flow).D_F⌉
                        If (ψ(flow).ARR > C_T)
                                ψ = ψ - flow
                                // Reject the request by sending a rejection
                                // message to the sender
        End For

        For each flow ∈ ψ
                ID = ψ(flow).ID
                ARR = ψ(flow).ARR
                R_T = ψ(flow).R_T
                D_F = ψ(flow).D_F
                D_S = ψ(flow).D_S
                T_S = ψ(flow).T_S
                If (Admission(Φ,ID,ARR,R_T,D_F,D_S,T_S,T_C,C_T) = YES) Then
                {Accept the request by sending an acceptance message to the sender and
                start the flow with its ARR at its start time using a max-min fair
                Congestion Control protocol}
                        ψ = ψ - flow
        End For

End Procedure
```

4 PERFORMANCE EVALUATION

A single bottleneck link dumbbell network configuration is used for the simulations using ns-2. The bottleneck capacity is 1 Gbps and the bottleneck delay is set to 50ms. Drop Tail routers are used. The buffer size of the bottleneck link is set to 100% of the Bandwidth-Delay product. The packet size is set to 1500 bytes. The capacity of side links is 10 Gbps and the delay of each side link is set to 2ms.

We have compared the results of our mechanism with the unreliable *Fixed_Rate* transfer scheme that represents a traditional QoS architecture such as IntServ/RSVP. In a *Fixed_Rate* scheme, a flow can only send its data at a constant ARR. We have also compared the results of our scheme, *ARR_UDT_Queue*, with *ARR_IdealTCP_Queue* scheme, which uses an ideal TCP instead of UDT. Using an ideal TCP means that there is no loss in the network and the residual bottleneck bandwidth is fairly shared among the flows. The *ARR_IdealTCP_Queue* scheme gives an upper bound of the admission percentage for our practical mechanism, *ARR_UDT_Queue*.

In one experiment, 5 sets of 10 simulations are performed. In each simulation 100 flows are run. Within each set the size of all flows is the same, however the size of a flow varies from 500 MB to 1500 MB from one set to the other. The mean inter arrival time of flows is 4 seconds and the transfer duration of each flow is 50 seconds. For each simulation within a set, there are mixed types of randomly generated reservation requests, immediate and advance reservations. In each simulation the arrival time of a new reservation is also randomly chosen in each 4 seconds interval. The start time of an advance reservation request is also selected randomly in the interval [25,75] relative to the time of the arrival of the flow. The results of this experiment are shown in figure 4. Each point in this figure represents an average of the results of 10 simulations of a set. The standard deviation of the results of all sets is less than 2.

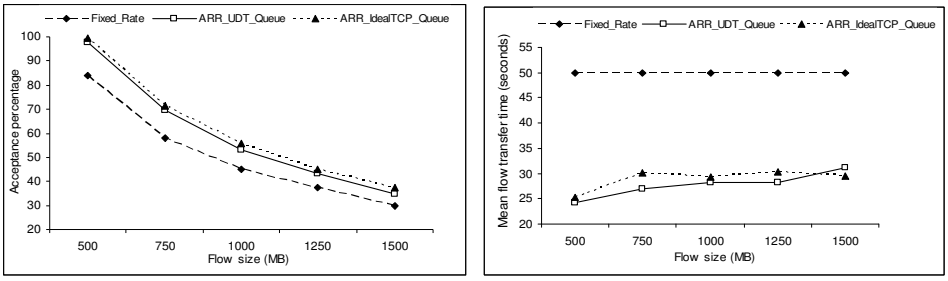

Figure 4. The average inter arrival time is held constant however the flow size varies.

In another experiment, 3 sets of 10 simulations are performed with 100 flows. Other than a flow size and the mean inter arrival time of flows, all other simulation parameters are the same as are used in the previous experiment. In this experiment the flow size remains the same in each simulation of all sets however the mean inter arrival time of flows varies from 2 seconds to 6 seconds. The results of the experiment are shown in figure 5. Each point in this figure 6 represents an average of the results of 10 simulations of a set. The standard deviation of the results of all sets is less than 2.

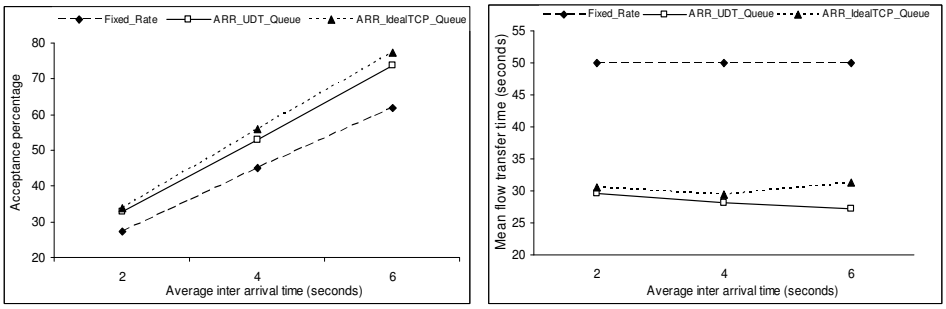

Figure 5. The flow size is held constant however the average inter arrival time varies.

The figures 4 and 5 show that the acceptance rate of *ARR_UDT_Queue* scheme is always higher than *Fixed_Rate* scheme. The mean flow transfer time scheme of *ARR_UDT_Queue* scheme is always less than the *Fixed_Rate* scheme which has a flow transfer time equal to the duration of a flow. The ideal approach, *ARR_IdealTCP_Queue*, always gives an upper bound of acceptance rate for *ARR_UDT_Queue* scheme. As UDT quickly consumes the available bandwidth, the results show that the acceptance percentage of flows with our mechanism is quite close to the ideal case. In general, the mean flow transfer time of *ARR_IdealTCP_Queue* is more than the *ARR_UDT_Queue* scheme because in the *ARR_IdealTCP_Queue* scheme the flows share the residual network capacity fairly whereas in *ARR_UDT_Queue* scheme it takes some time for flows to converge to the fair share upon admission of a new flow in the network, making the flows complete earlier.

5 CONCLUSION

Our results show that, by using a fair and a stable congestion control mechanism like UDT and by using a bandwidth allocation mechanism with admission control to provide network reservation guarantees for elastic flows, the network can be fully utilized, resulting in early completion of a long-lived flow which consequently enables us to admit more flows earlier than it would have been possible without using our mechanism. Our contribution is that we have shown the design and the implementation

of an approach which is reliable and realistic, taking computation and communication overheads and delays into account.

We are planning to develop an implementation of *ARR-CC-Queue* for usage in real Grids, and test it in the context of the European project "EC-GIN" ("Europe-China Grid InterNetworking") using the French Grid'5000 testbed in collaboration with INRIA.

REFERENCES

[1] Marchal L., Primet P., Robert Y. and Zeng J., "Optimal Bandwidth Sharing in Grid environment", *IEEE HPDC*, Paris, France, 2006.

[2] Schill A., Breiter F. and Kuhn S., "Design and Evaluation of an Advance Reservation Protocol on Top of RSVP", *In IFIP 4th International Conference on Broadband Communications (BC '98)*, Stuttgart, Germany, IFIP Conference Proceedings 121, Chapman & Hall, 23–40, 1998.

[3] Guerin R. and Orda A., "Networks with Advance Reservations: The Routing Perspective", *In Proceedings of IEEE INFOCOM 2000*, 118–127, 2000.

[4] Zhang H., Keahey K. and Allcock B., "Providing Data Transfer with QoS as Agreement-Based Service", *International Conference on Services Computing (SCC 2004)*, Shanghai, China, September 15 - 18, 2004.

[5] Foster I., Roy A. and Sander V., "A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation", *In 8th International Workshop on Quality of Service (IWQoS 2000)*, pp. 181–188, 2000.

[6] Foster I., Fidler M., Roy A., Sander V. and Winkler L., "End-to-end quality of service for high-end applications", *Computer Communications*, vol. 27, no. 14, pp. 1375–1388, 2004.

[7] Foster I., Kesselman C., Lee C., Lindell R., Nahrstedt K. and Roy A., "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", *In 7th International Workshop on Quality of Service (IWQoS)*, London, UK, pages 27–36, 1999.

[8] Burchard L., Heiss H. and Rose D., "Performance issues of bandwidth reservations for grid computing", *Proceedings of Computer Architecture and High Performance Computing*, pp 82–90, 2003.

[9] Xing J., Wu C., Tao M., Wu L. and Zhang H., "Flexible Advance Reservation for Grid Computing", *GCC 2004: 241-248*, 2004.

[10] Wu L., Xing J., Wu C. and Cui J., "An Adaptive Advance Reservation Mechanism for Grid Computing", *PDCAT 2005: 400-403*, 2005.

[11] Chen B.B. and Primet P., "Supporting bulk data transfers of high-end applications with guaranteed completion time", *IEEE ICC2007 International conference on computer communication*, 2007.

[12] Primet P. and Zeng J., "Traffic Isolation and Network Resource Sharing for Performance Control in Grids", *ACNS'05*, USA, 2005.

[13] Kaushik N., Figueira S. and Chiappari S., "Flexible Time-Windows for Advance Reservation in LambdaGrids", *ACM SIGMETRICS/Performance*, 2006.

[14] Naiksatam S. and Figueira S., "Elastic Reservations for Efficient Bandwidth Utilization in LambdaGrids", *Elsevier's FGCS - The International Journal of Grid Computing: Theory, Methods and Applications*, vol. 23, issue 1, pp. 1-22, 2007.

[15] Gu Y. and Grossman R., "UDT UDP-based data transfer for high-speed wide area networks", *Computer Networks, special issue on Hot topics in transport protocols for very fast and very long distance networks*, 2007.

[16] Müller J.A., Hessler S. and Irmscher K., "Class of Service Concepts in Autonomous Systems", *Terena networking conference 2004*, Rhodes, Greece, 2004.