An End-to-End QoS Mechanism for Grid Bulk Data Transfer for Supporting Virtualization*

Kashif Munir[1], Somera Javed[2], Michael Welzl[1], Humaira Ehsan[2], Tooba Javed[2]

[1] Institute for Computer Science, University of Innsbruck, Austria
{Kashif.Munir, Michael.Welzl}@uibk.ac.at
[2] National University of Computer and Emerging Sciences, Islamabad, Pakistan

Abstract. We consider sustainable and deterministic QoS a key ingredient for providing virtualization and hence introduce an end-to-end Quality of Service mechanism for Grid bulk data transfers. Our mechanism enables per-flow guarantees and efficiently utilizes available resources without requiring any router support except for the provisioning of a single high class traffic aggregate. This is attained by taking the specific requirements and environment conditions in common Grids into account. We document simulation results which illustrate how guarantees are realized by applying admission control, and by uniformly using a max-min fair congestion control mechanism for all flows.

Keywords: Quality of Service, QoS, Grid, Bulk Data Transfer, Advance Reservation, UDT

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

* The work described in this paper is supported by the Higher Education Commission (HEC) of Pakistan under the doctoral fellowship program for Austria.

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. This is all the support that we foresee from the ISP side in our mechanism. Note that, at this point, we exploit property 3 above: with no additional control of routers, path changes can always cause reservations to fail. Assuming that path changes are rare events, and the failure to provide a guarantee is not as severe as in the standard case of an end user requesting a multimedia service, we decided to accept this downside of our mechanism.

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).

In a standard Bandwidth Broker scenario, where signaling is used to ensure per-flow QoS, routers must constantly update the Bandwidth Broker about their current state, and at least the ingress router close to the newly joining flow must be informed about reservations in order to detect them and apply the right shaping or policing functions to ensure conforming behavior. Since our Resource Broker controls all the traffic, knowing when a flow enters and leaves the aggregate, there is no need for such traffic updates. Other information about the network is however needed, and would have to be communicated to the Resource Broker from a constantly active distributed measurement system in the Grid:

- Bottleneck link capacities[1] must be known for all bottlenecks of all end-to-end paths.
- Shared bottlenecks must be detected.

We assume that knowledge about bottleneck capacities and shared bottlenecks is available at the end systems, and point out that there are enough indications in the literature that obtaining such measurements would be feasible. This 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

2.1 Network Reservations

In general, there are two types of network resource reservations in computer networks [1]. One is immediate reservation which is made in a just-in-time manner and the other is advance reservation, which allows reserving network resources a long time before they are actually used.

[1] In what follows, the term “capacity” does not refer to the physical capacity of a link but the maximum transmission rate that it provides to users of the protected high-class traffic aggregate.

Early work on advance reservation focused on reservation protocols like RSVP [2] and ST-II [3], admission control mechanism [4] and routing algorithms for networks with advance reservations [5].

Grid applications need guarantees of Quality of Service (QoS) [6,7]. Targeting deadline support for bulk data transfers, the problem of network resource reservation [8] 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 [9]. Another example for the application of advance reservations is a distributed media server systems as described in [10], where a large number of media files are transmitted between the different servers.

In [12] if the latest call request is a malleable request, the method of [11] or [1] is used to adjust the bandwidth or duration to satisfy the requester. However for a fixed request, the bandwidth or duration of transmission can not be modified.

A general view of the network resources sharing in Grids and Grids traffic isolation are discussed in [13]. Optimization of bandwidth sharing among Grid flows is given [14] by manipulating the transmission windows of the flexible requests between minimum and maximum rates to maximize the acceptance rate of requests and to maximize the network utilization while still meeting their deadlines. The formulated optimization problem is proven to be NP-complete.

Two types of strategies for scheduling bulk data transfers are possible [15]. 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 former strategy.

A time-slot based approach for scheduling the elastic and streaming requests is described in [16]. 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.

In all the above approaches, a flow sends at a fixed rate in a time slot. The residual network capacity gets wasted in the approaches which do not use explicit signaling for the manipulation of data transfer rates of individual flows. The above approaches are based on offline scheduling of network reservations. Furthermore the above approaches are not reliable and realistic as they do not take into account the communication losses and overheads which occur in real networks. Our mechanism is reliable and realistic and it takes into account all communication and computation overheads that are involved in a reliable transfer of data in a network. Our mechanism provides online scheduling of network reservations. Furthermore in our proposed scheme the residual capacity is opportunistically and fairly shared by all existing flows, which results in the early completion times of flows and which consequently leads to higher percentage of admission of flows in the network.

2.2 Network Measurements

The information about the network that is needed for our architecture can be obtained via an end-to-end measurement system such as the one described in [17]. This system

could send probe traffic, or require the sender to cooperate by time stamping the packets or sending them back-to-back. Active methods for deducing bottleneck capacities via so-called “packet pairs” have been studied for a long time, starting with [18], and led to a large number of measurement tools. An example of such tool is “NetTimer” [19]. Recently, strictly passive methods were investigated, where the fact that TCP itself sends packet pairs if receivers use “Delayed ACKs” (as the specification suggests) is exploited [20].

Detecting shared bottlenecks in the network is also not a new problem; various techniques were proposed in [21,22]. In [23], a completely passive approach for learning about shared bottlenecks was introduced.

2.3 Congestion Control

Common admission control schemes assume all flows to use a certain fixed (or maximum) rate. It is a key feature of our mechanism that it manages to efficiently utilize network resources in a scalable manner because flows automatically increase their rates as bandwidth becomes available. This is attained by using a 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.

Most end-to-end congestion control schemes in the literature converge to a rate which depends on the round-trip time (RTT). One particular fairness measure that would suit our needs is called "max-min fairness". The authors of [24, 25, 26] showed that the well-known TCP variants FAST TCP, Scalable TCP (STCP), HighSpeed-TCP, BIC, CUBIC, H-TCP are not "RTT-fair". There are however exceptions: UDT [27] is designed to be max-min fair. Because it is designed for high-speeds and particularly convenient in a Grid setting, we chose UDT for our mechanism, but stress that any max-min fair congestion control scheme could be used in its place.

3 Our Quality of Service Mechanism: Introduction, Design and Implementation

3.1 Introduction

The QoS mechanism provides strict network guarantee to a flow, i.e., it admits a flow with an average rate (we call this the “Average Required Rate (ARR)”) of x bits per second to make it possible for it to meet its deadline. After admission, a fair allocation is provided to flows using 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 average rate requirement.

The admission and termination of a flow is controlled through the Resource Broker (RB) residing on any 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, and distributing the Resource Broker with a scheme as in [28] would not change anything about our mechanism.

It is assumed that an efficient technique for measuring the bottleneck capacity and shared bottlenecks is used in the Grid network. There are existing techniques which achieve that (see section 2). Further, we assume that all the QoS traffic is isolated from any other traffic – that is, the Resource Broker has complete knowledge of all flows that enter and leave the system in our QoS mechanism.

The design goals of such a QoS Mechanism include providing strict network guarantees, higher percentage of admission of flows in the network, Advance Reservation of flows and max-min fair allocation to flows irrespective of their RTTs.

3.2 Design/Operation of the QoS Mechanism

The basic idea is to divide the bandwidth into weights of some predefined rate value (e.g. each weight is *y* bits per second). So a flow requiring an ARR of *x* bits per second takes *x/y* share of the bottleneck capacity.

The following example explains the scenario. Let us assume that we have bottleneck capacity of 4 Gbps and we have 4 flows at the start sharing the bottleneck. Let one weight be of 1 Gbps and each flow requires an ARR of 1 Gbps and each flow has a different completion time. As the required ARR for all the flows is available, all the flows are admitted. Each flow informs the RB about its desired admission in the network and it will also inform the RB as soon as it terminates so that its entry is deleted by the RB and the resources owned by the flow are relinquished. In the case of congestion or loss in the network the rates of all flows are reduced to their average required rates.

After *t1* seconds flow-1 terminates. This means that 1 Gbps of the bottleneck bandwidth is now available. This available bandwidth will be divided fairly among the existing flows, i.e. among flows 2, 3 and 4, using a max-min fair mechanism. So each remaining flow (2, 3 and 4) will reach the sending rate of 1.33 Gbps. Increased rate of 1.33 Gbps will make the flows 2, 3 and 4 terminate earlier than their deadlines. The pictorial representation of the 3 flows at that particular moment is shown in figure 1.

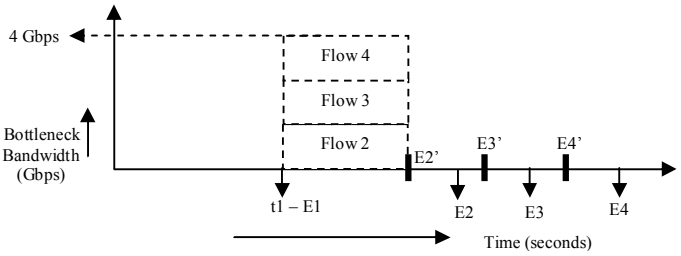

Fig. 1. At time *t1*, *flow 1* terminates. The black marks along the x-axis only show the (logical) expected completion times *E2', E3'* and *E4'* for the termination of flows 2, 3 and 4 respectively according to the rate of the flows at that particular moment of time t1.

Suppose that after *t2* seconds a new flow (flow-5) requiring 1 Gbps ARR wants to be admitted in the network before any of the existing flows (i.e. flows 2, 3 and 4) terminates, as shown in figure 2.

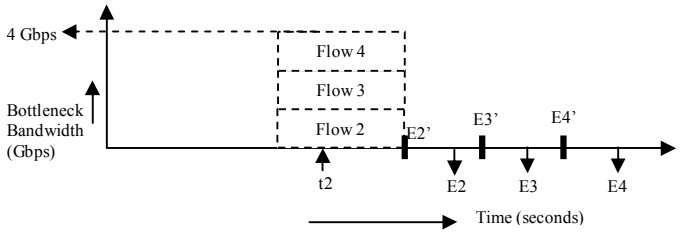

Fig. 2. At time *t2, Flow-5* asks the Resource Broker for admission

The new flow is admitted in the network as its ARR is available. All flows (flows 2, 3, 4 and 5) will now be sending at the rate of 1 Gbps. The pictorial representation of the 3 previous flows (flows 2, 3 and 4) and the new flow (flow 5) at the time of admission of flow-5 in the network is shown in figure 3. Note that the pictures 1, 2 and 3 show ideal adjustment of rates and are just used to explain the example; in actual simulation the adjustment of rates takes some time according to the congestion control protocol.

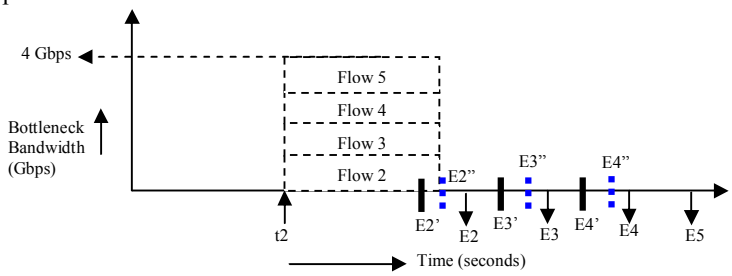

Fig. 3. The dotted marks along the x-axis show the expected completion times *E2"*, *E3"* and *E4"* for the termination of *flows 2*, *3* and *4* respectively according to the rate of the flows at that particular moment of time *t2*.

3.3 Implementation of the QoS Mechanism

One of the key components of our proposed QoS mechanism is the Resource Broker which is designed for admission control and to maintain the current state of network (i.e. all information about existing flows, shared bottleneck links and their capacities and paths). To admit a flow a sender sends a message to the RB for its possible admission and upon completing the transfer of a flow, 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 huge amount of data is transferred. In the simulations the FTP application protocol is used over the UDT high-speed data transfer protocol. The Admission Control Algorithm for the RB is given below.

D_S,D_F,T_S,ARR,ID,R_T: Data size, duration, start time, ARR, ID and reservation type of the flow for which a reservation is requested
$R_T \in$ {IR,AR}: IR = Immediate reservation and AR = Advance reservation
Record of a flow: {T_S,T_E,D_F,D_S,ARR,ID}
***Φ**: Set of records of the currently accepted flows sharing the bottleneck link*
C_T: The total capacity of the bottleneck link
T_C: The current time

```
Procedure ARR_CC(Network_Topology_Information)

While (All flows are processed)
  If (a new reservation is requested)
       ARR = ⌈D_S / D_F⌉
       ID = generate ID for new request
       If (Admission(Φ,ID,ARR,R_T,D_F,T_S,T_C,C_T) = YES) Then
       {Accept the flow and start the flow with its ARR at its start
       time using a max-min fair Congestion Control protocol}
       Else
              {Reject the flow}
       If (a served request is completed) Then
              Termination(Φ,ID)
End While

End Procedure

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

Set C_R to 0            // C_R is the Reserved Capacity

If (R_T = IR) Then      // Immediate reservation request
       T_S = T_C
       T_E = T_C + D_F    // T_E is the End Time of a flow
Else                    // Advance reservation request
       T_E = T_S + D_F

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

If (C_T − C_R) > ARR Then
       Φ = Φ + flow     // flow = flow_record(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 (flow.ID = ID) Then
       Φ = Φ  - flow
       Break
End For

End Procedure
```

4 Simulations and Analysis

A single bottleneck link dumbbell network configuration is used for the simulation using ns-2. The bottleneck capacity is 1Gbps 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 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. Due to space constraints, only the most important results are included in order to show that our QoS mechanism meets all its goals.

4.1 Simulation: Higher Acceptance Percentage of Flows in the Network

A series of experiments is performed with 25, 50, 75, 100 and 125 flows. In each experiment the arrival time of a new flow is random in each five seconds interval, the data size of each flow is randomly chosen between 100 MB to 2 GB and the duration of each flow is randomly chosen between 50 to 300 seconds. Without our QoS mechanism (ARR-CC), a flow would send its data at a fixed rate equal to its ARR and would complete by its deadline. Figure 4 shows the acceptance percentages with our QoS mechanism as opposed to Fixed-Rate data transfers.

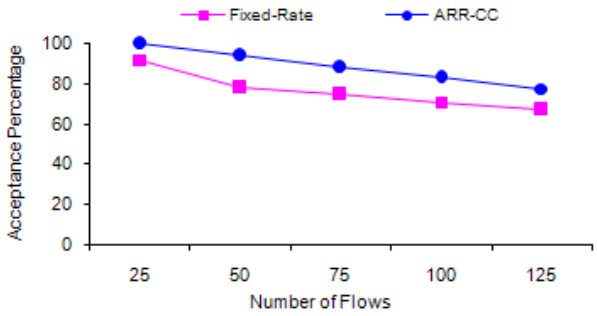

Fig. 4. Flows Acceptance Percentage with and without our QoS mechanism

4.2 Simulation: Mixed Types of Reservation Requests

For this experiment we extended our mechanism with an advance reservation capability. In this simulation ten flows are started with different data sizes and starting times. The first and second flows are granted admission at their admission time as their ARR is available. The fourth flow joins the network at 10 seconds of simulation time and reserves bandwidth for 30 seconds in advance, starting at 40 seconds of simulation time and so on. All accepted flows utilize the maximum available bandwidth and finish earlier than their deadlines. Table 1 shows that the average rate achieved by a flow for data transfer is higher than the ARR of that flow. It has become possible due to the use of the high-speed congestion control protocol, UDT, which makes the flows to quickly fill up the residual capacity of the network.

Table 1: Immediate and Advance Reservations: AT is the Admission Time, ARST is the Advance Reservation Start Time, CT is the Completion Time and ARA is the Average Rate Achieved, IR is the Immediate Reservation and AR is the Advance Reservation

Flow #	File Size (GB)	AT (Sec)	Duration (Sec)	ARST (Sec)	ARR (Mbps)	Status	CT (Sec)	ARA (Mbps)
1	1	0	40	--	200	IR Accepted	10	800
2	3.75	20	100	--	300	IR Accepted	72	577
3	6	30	120	--	400	IR Rejected	--	--
4	1.875	10	30	40	500	AR Accepted	69	517
5	0.438	15	35	40	100	AR Accepted	62	160
6	3.125	115	50	--	500	IR Rejected	--	--
7	1.125	145	30	--	300	IR Rejected	--	--
8	4	20	40	150	800	AR Accepted	186	889
9	0.500	25	40	150	100	AR Accepted	182	125
10	2	50	40	250	400	AR Accepted	268	889

Figure 5 shows the above simulation of the ten flows. All flows utilize the maximum bandwidth and their rates do not go below their respective ARR at any time during the simulation. The total rate curve shows that, throughout the simulation time, when there is at least one flow in the network, its rate stays around 1Gbps which indicates maximum utilization of the network.

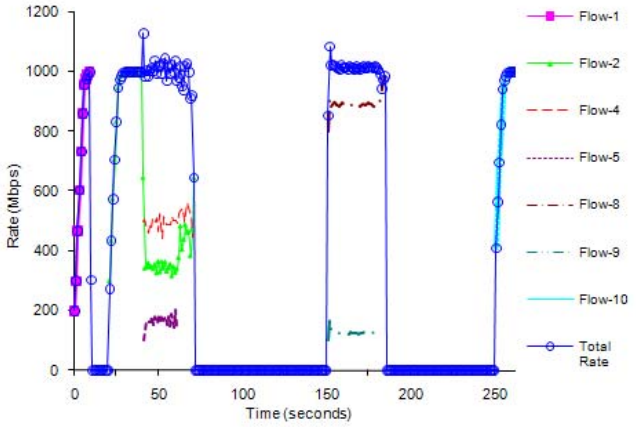

Fig. 5. The QoS mechanism with mixed type of reservation requests

5 Conclusion and Future Work

At the beginning of this paper, we made the point that end-to-end virtualization requires deterministic and sustainable QoS guarantees from lower layers. The reason for this requirement is easy to see: in a performance oriented system like the Grid, it only makes sense to hide complexity if such hiding does not come at the cost of

reduced efficiency. QoS guarantees must therefore be a part of the little information that is kept about lower layers – and failure to deliver the necessary QoS raises the question whether virtualization makes sense for the system. For this reason, saying "yes" to requests as often as possible must be the main design goal of a Grid QoS system.

Our results show that, by using a fair and stable bandwidth allocation mechanism like UDT to provide network reservation guarantees for elastic flows, the network can be fully utilized, resulting in earlier completion of a long-lived flow which consequently makes it possible to admit more flows earlier than it would have been possible without using a congestion control mechanism. Clearly, the goal of saying "yes" as often as possible was reached. Our contribution is that we have shown the design and the implementation of a reliable and realistic approach which takes the computation and communication overheads into account.

Our next step is to extend the mechanism in such a way that it will become possible to admit some of the new flows in the network even if the required bandwidth is not available at the cost of decreasing the rates of some already existing flows even below their ARRs. This will increase the acceptance percentage of flows. Since this requires the Resource Broker to choose which flows are to decrease their rates, this scheme will require some signaling between the Resource Broker and end nodes.

References

1. Burchard, L., Heiss, H., Rose, D.: Performance issues of bandwidth reservations for grid computing. Proceedings of Computer Architecture and High Performance Computing, pp. 82–90, (2003)
2. Schill, A., Breiter, F., Kuhn, S.: Design and Evaluation of an Advance Reservation Protocol on Top of RSVP. In IEIP 4th International Conference on Broadband Communications (BC '98), Stuttgart, Germany, IFIP Conference Proceedings 121, Chapman & Hall, pp. 23–40, (1998)
3. Reinhardt, W.: Advance Resource Reservation and its impact on Reservation Protocols. In Proceedings of Broadband Islands '95, Dublin, Ireland, (1995)
4. Ferrari, D., Gupta, A., Ventre, G.: Distributed Advance Reservation of Real-Time Connections. Multimedia Systems, Vol.5. Springer-Verlag, Berlin Heidelberg New York, pp. 187-198, (1997)
5. Guerin, R., Orda, A.: Networks with Advance Reservations: The Routing Perspective. In Proceedings of IEEE INFOCOM 2000, pp. 118–127, (2000)
6. Zhang, H., Keahey, K., Allcock, B.: Providing Data Transfer with QoS as Agreement-Based Service. International Conference on Services Computing (SCC 2004), Shanghai, China, (2004)
7. Foster, I., Roy, A., Sander, V.: A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation. 8th International Workshop on Quality of Service. June 2000, (IWQoS 2000), pp. 181–188, (2000)
8. Foster, I., Fidler, M., Roy, A., Sander, V., Winkler, L.: End-to-end quality of service for high-end applications. Computer Communications, vol. 27, no. 14, pp. 1375–1388, (2004)
9. Foster, I., Kesselman, C., Lee, C., Lindell, R., Nahrstedt, K., 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)

10. Burchard, L., Luling, R.: An Architecture for a Scalable Video-on-Demand Server Network with Quality-of-Service Guarantees. In 5th International Workshop on Distributed Multimedia Systems and Applications (IDMS), vol. 1905 of Lecture Notes in Computer Science (LNCS), pp. 132–143. Springer, (2000)
11. Xing, J., Wu, C., Tao, M., Wu, L., Zhang, H.: Flexible Advance Reservation for Grid Computing. GCC 2004, pp. 241-248, (2004)
12. Wu, L., Xing, J., Wu, C., Cui, J.: An Adaptive Advance Reservation Mechanism for Grid Computing. PDCAT 2005, pp. 400-403, (2005)
13. Primet, P., Zeng, J.: Traffic Isolation and Network Resource Sharing for Performance Control in Grids. ACNS'05, USA (2005)
14. Marchal, L., Primet, P., Robert, Y., Zeng, J.: Optimal Bandwidth Sharing in Grid environment. IEEE HPDC, Paris, France, (2006)
15. Kaushik, N., Figueira, S., Chiappari, S.: Flexible Time-Windows for Advance Reservation in LambdaGrids, ACM SIGMETRICS/Performance, (2006)
16. Naiksatam, S., 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)
17. Yousaf, M. M., Welzl, M.: A Reliable Network Measurement and Prediction Architecture for Grid Scheduling. IEEE/IFIP International Workshop on Autonomic Grid Networking and Management (AGNM'05), Barcelona, Spain, (2005)
18. Keshav, S.: Congestion Control in Computer Networks. (http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/91/thesis/keshav.th.tar.Z) PhD Thesis, published as UC Berkeley TR-654, (1991)
19. Lai, K., Baker, M.: "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, California, (2001)
20. Barz, C., Frank, M., Martini, P., Pilz, M.: Receiver-Based Path Capacity Estimation for TCP. In Proceedings of KIVS'05, Kaiserslautern, Germany, (2005)
21. Shriram, A., Kaur, J.: Identifying Bottleneck Links Using Distributed End-to-end Available Bandwidth Measurements. In the First ISMA Bandwidth Estimation Workshop (BEst'03), San Diego, USA, (2003)
22. Kim, M. S., Kim, T., Shin, Y., Lam, S., Powers, E. J.: A Wavelet-Based Approach to Detect Shared Congestion. In Proceedings of ACM SIGCOMM 2004, (2004)
23. Katabi, D., Bazzi, I., Yang, X.: A passive approach for detecting shared bottlenecks. In Proceedings of the 10th IEEE International Conference on Computer Communications and Networks, (2001)
24. Li, Y. T., Leith, D., Shorten, R.: Experimental Evaluation of TCP Protocols for High-Speed Networks. Technical report, Hamilton Institute, (2005)
25. Tan, K., Song, J., Zhang, Q., Sridharan, M.: Compound TCP: A Scalable and TCP-friendly Congestion Control for High-speed Networks. 4th International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2006), Nara, Japan, (2006)
26. Ha, S., Kim, Y., Le, L., Rhee, I., Xu, L.: A Step toward Realistic Performance Evaluation of High-Speed TCP Variants. PFLDnet 2006, Nara, Japan, (2006)
27. Gu, Y., 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)
28. Müller, J. A., Hessler, S., Irmscher, K.: Class of Service Concepts in Autonomous Systems. Terena networking conference 2004 (Terena2004), Rhodes, Greece, (2004)