Journal of Grid Computing **1:** 329–343, 2003.

Quality of Service Networking for High Performance Grid Applications

Miguel Rio, Andrea di Donato, Frank Saka, Nicola Pezzi, Richard Smith, Saleem Bhatti and Peter Clarke
Networked Systems Centre of Excellence, Department of Physics and Astronomy, University College London, Gower Street, WC1E 6BT, London, United Kingdom
E-mail: M.Rio@cs.ucl.ac.uk

Key words: bandwidth broker, differentiated services, Grid networking, MPLS, quality of service, SLA

Abstract

This paper reports on different efforts to provide quality of service (QoS) Networking to Grid applications done in the context of the MB-NG, GRS and DataTAG EU projects. These are leading edge network research projects involving more that 50 researchers in the UK, Europe and North America, concerned with the development and testing of protocols and standards for the next generation of high speed networks. We have implemented and tested the Differentiated Services Architecture (DiffServ) in a multi-domain, 2.5 Gbits/s network (the first such deployment) defining appropriate Service Level Agreements (SLAs) to be used between administrative domains to guarantee end-to-end Quality of Service. We characterised several hardware implementations of DiffServ and concluded on their appropriateness for several network scenarios. Since current and future Grid applications will have to use modified mechanisms of congestion control we have evaluated old and new TCP implementations over a Differentiated Services Networks. These quality of service tests have also included innovative MPLS (Multi-Protocol Label Switching) experiments to establish guaranteed bandwidth connections to Grid applications in a fast and efficient way. We have also developed a software based bandwidth broker architecture for Grids based on IETF standards which allows applications to transparently request dynamic and advanced reservations and implemented it in a real experimental network. We finally report on experiences delivering Quality of Service networking to high performance applications like Particle Physics data transfer and High Performance Computation. This includes quantitative results on the performance improvements that QoS brought to real data transfers in the context of High Performance Computing.

1. Introduction

Recent years have seen the appearance of a wide range of scientific applications with extremely high demands from the network. Once more scientists require the network to be pushed to the limits and challenge the idea that bandwidth availability is not a problem any more.

Unlike traditional applications like email, WWW or even peer-to-peer systems, these applications require reliable file transfers on the order of the 1 Gbit/s and have often have tight delay deadlines for delivery to remote systems or low-delay requirements. High Energy Physics, Radio Astronomy or High Performance Steered Simulations cannot achieve their goals in a sustainable, efficient and reliable way with current production networks, which are almost totally based on a best-effort service model.

Although part of the networking community believe that bandwidth over-provisioning will always solve every network problem, our work shows that Quality of Service enabled networks provide a vital role to support high performance applications efficiently, inexpensively and with smaller additional configuration work.

QoS performance has been studied exhaustively through analytical work and simulation (see for example [1]). These works are of extreme importance and relevance but they have two drawbacks. On one hand simulation models have proved to be incomplete [2, 3]

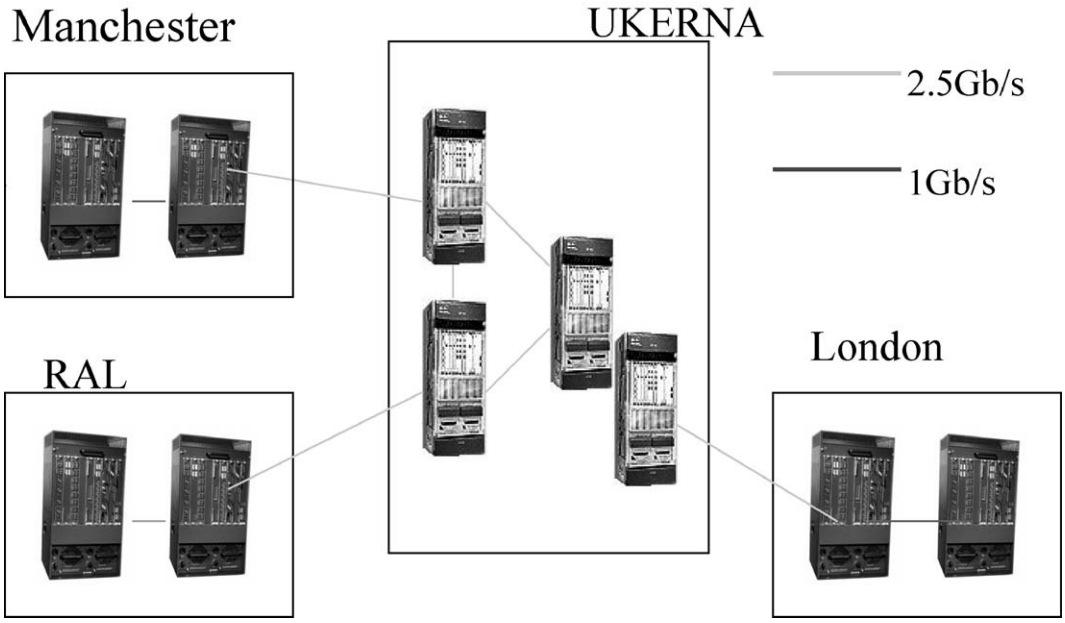

Figure 1. MB-NG network.

because they fail to represent all possible real configurations of the network. They also fail to account for several implementation details of 'real' networks. These include operating system tuning, driver configurations, memory and CPU overflows, feature interactions, etc. Therefore testbed networks play a vital role in network research as a way of consolidating technology and, through exhaustive debugging and testing, provide implementation guidelines to the QoS network user community. Indeed, testbed experience may help to improve simulation-based and analytical techniques.

The work reported here used two testbeds with different characteristics: a United Kingdom testbed used in the context of the MB-NG project and a Transatlantic one used in the context of the European DataTAG project.

The Managed Bandwidth – Next Generation (MB-NG) project [4] created a pan-UK-wide Networking and Grid testbed that focused upon advanced networking issues and interoperability of administrative domains.

The project addressed the issues which arise in the sector of high performance inter-Grid networking, including sustained and reliable high performance data replication and end-to-end advanced network services.

The MB-NG testbed can be seen in Figure 1. It consists of a triangle connecting RAL, University of Manchester and University College London at OC-48 (2.5 Gb/s) speeds using CISCO's 12000. Each of the edge domains is built with 2 Cisco's 7600 interconnected at 1 Gb/s.

The DataTAG project [5] created a large-scale intercontinental Grid testbed involving the European DataGrid project, several national projects in Europe, and related Grid projects in the USA. It involves more than 50 people and among other things is researching inter-domain quality of service and high throughput transfers of high delay networks (making use of an intercontinental link connecting Geneva to Chicago, which can be seen in Figure 2).

This paper describes work spanning several areas that was crucial to provide High Performance, QoS enabled networking to a variety of Grid applications, namely:

1. Hardware implementation issues.
2. Real implementation of a Differentiated Services Network and testing of TCP traffic over this network.
3. The use of MPLS for on-demand traffic engineering.
4. Control Plane software to automate the process of bandwidth reservation
5. SLA definition.

The paper is organized as follows. In the next section we describe the Differentiated Services Architecture and some experimental results with TCP traffic over DiffServ. In Section 3 we describe MPLS and

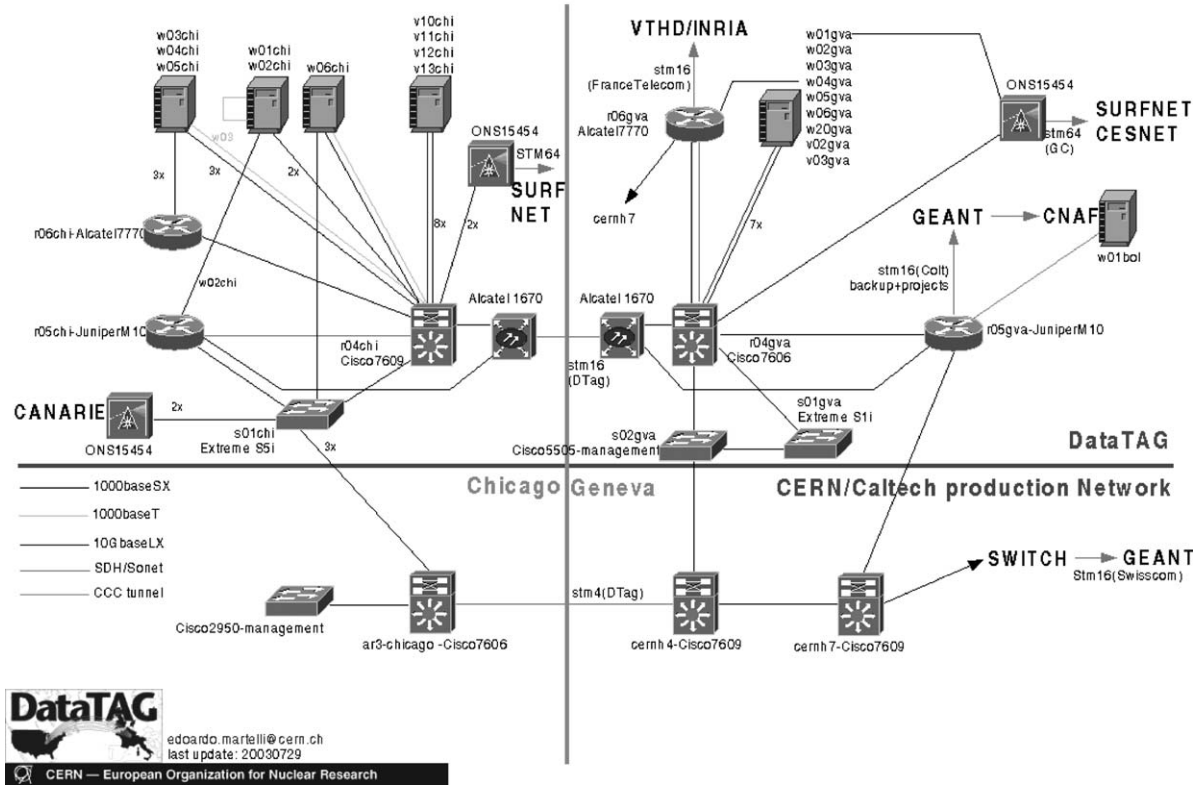

Figure 2. DataTAG network.

why it is a useful technology for Grid applications. In Section 4 we discuss Service Level Agreements definition between administrative domains followed by the description of our Control Plane software in Section 5. We describe some practical results when Grid applications used our QoS testbeds in Section 6 and present conclusions and further work in Section 7.

2. Differentiated Services

In the last two decades there have been many attempts to provide a Quality of Service enabled network that extends the current best-effort Internet by allowing applications to request specific bandwidth, delay or loss. These included complete new networks like ATM [6] or 'extensions' to IP networks like the Integrated Services Architecture [7]. Both these approaches required state to be stored in every router of the entire path for every connection. Soon it was realized that the core routers would not be able to cope with the large amount of state and signaling traffic and these architectures would not scale.

With the Differentiated Services Architecture [8] a simpler solution was proposed. Traffic at the edges is sorted into classes and routers in the core only have to schedule the traffic among these classes, i.e. traffic aggregates flows as opposed to individual application flows. Typically routers in the core only deal with a small number of classes, making it easier and manageable to implement. Routers at the edges may have to maintain per-flow state (especially the first router in the path) but since the number of flows that are being processed is several orders of magnitude smaller (e.g., a few tens of classes or fewer), this is not a significant problem.

However, applications can now only choose in which class they want their traffic to be placed, as opposed to a complete end-to-end traffic specification as in the IntServ architecture. This makes experimentation in testbeds crucial to understand how applications should make use of a DiffServ enabled network.

2.1. *DiffServ Hardware Implementation*

The deployment of IP-QoS in the Differentiated Services framework both in the access and in the core networks requires an in-depth knowledge of the performance of one of the most widely deployed router cards technology at the current time: POS_OC-48. The evaluation of its performance is based on how precisely a minimum bandwidth guarantee can be allocated to an aggregate of data traffic under interface congestion; it constitutes the foundations for the deployment of more complex IP QoS solutions.

We have studied the QoS performance of such technology provided by Cisco [9], Juniper [10] and Procket [11] in order to help the QoS engineering process of a multi-vendor network as well as to compare such performances throughout three amongst the main manufacturers for high speed core/access network routers.

The main objective is to understand the behaviour of the card in order to properly engineer the IP-level bandwidth allocation.

The approach is that of looking at both the utilisation of the link and the absolute as well as the relative bandwidth allocation errors. This double metric is necessary to have a complete picture of the performances of the routers under test.

In particular, observing poor link utilisation can be a quick method to detect possible operational anomalies but such observations need to be integrated with a proper error analysis. The latter results can be useful in detecting the presence of errors in correctly allocating bandwidth although it is useful to find out where the errors are when the link utilisation is adequate and it is also useful to find out where the errors are localised when the link utilisation already shows poor performances. This double metric is necessary since an adequate link utilisation is not only necessary, but also required, in order to have an equally good bandwidth allocation precision on an end-to-end basis.

Two classes were configured, BE (Best-Effort) and LBE (Less than Best-Effort). It is worth noticing noting that LBE is used here with the broadest meaning possible which is that of an IP class whose bandwidth allocation is complementary to 100% utilisation with that of BE; therefore it actually generalises the tight definition of LBE given by the Scavenger Service as defined in the work of Internet2 [12]. The bandwidth scheduler algorithm available on the three router manufacturers was always the same and is known with the term Weighted Fair Queuing (WFQ) [13].

The bandwidth allocation set chosen for the tests was mainly the following sequence of couples: BE-LBE = (99–1, 98–2, 97–3, 96–4, 97–3, 95–5, 94–6, 93–7, 92–8, 91–9, 90–10, 85–15, 80–20, 75–25, 70–30, 65–35, 60–40, 55–45, 50–50).

We refer to the sequence above as the ‘bandwidth allocation couples axis’ with the axis direction going from 99–1 to 50–50.

The type of traffic used to investigate the performance of the bandwidth scheduler algorithm (WFQ) was UDP since its dynamics are that of a constant rate whereas TCP’s are not.

We chose a packet size of 1470 bytes for our tests as it gave us the best performances when transmitting from Linux hosts (kernel version 2.4.20).

2.1.1. *Error Analysis Definition*

Since poor link utilisation (i.e. too much traffic on a link) is only sufficient but not necessary for poor bandwidth allocation precision, an error analysis is also required.

In particular, we present two types of errors, relative and absolute, as a measure of the precision in the allocation of bandwidth for both the BE and LBE class.

The absolute (in Mbps) and relative (in %) errors for a generic class X are:

$$\begin{aligned} \text{AbsoluteError} &= \text{WhatclassXgets} \\ &\quad - \text{WhatclassXshouldget (Mbps)}, \\ \text{RelativeError} &= \text{AbsoluteError} \times 100/ \\ &\quad \text{WhatclassXshouldget (\%)}. \end{aligned}$$

One thing which is worth noticing is that most of the manufacturers set the accuracy in allocating bandwidth to a class to be around 95%, which in turn means that 2.5% is the maximum relative error acceptable. We refer to the region validating this bound as the ‘operating region.’

In order to bound an operating region out of the bandwidth allocation couples axis, a new metric consisting of the maximum value that the LBE and BE relative errors with sign assume all over the offered load axis is thus introduced. The errors are taken into account with their sign since the polarisation of the error is functional to understand its nature.

We refer to these metrics as the ‘Maximum LBE Relative Error with Sign’ (M-LREWS) and the ‘Maximum BE Relative Error with Sign’ (M-BREWS), respectively.

The MAX LBE/BE Relative Errors with Sign allow the evaluation of the bandwidth scheduler based solely on its worst performance over the offered_load/port_congestion_level axis.

In order to evaluate the performances of a card averaged over the whole offered load axis, or better, averaged over different card congestion levels, a different metric consisting of the average of the absolute values of LBE/BE relative errors which we refer to as A-ALRE and A-ABRE respectively, is introduced. The absolute values are used to avoid that the average of algebraic values could lead to a misleading error of $\sim 0\%$.

The main difference between the MAX-based and the Averaged-based metrics is that the latter takes account of the errors all over the 'offered_load'/'card_congestion_level' axis and not just of the maximum over this axis. This metric measures the magnitude of the spread of the error over the card congestion level axis.

We refer to the M-L/BREWS operating region as the 'max' operating region or simply as the operating region and to the A-AB/LRE operating region as the 'avg' operating region, this way highlighting the computation of the Max and Avg algebra, respectively.

Preliminary tests clearly showed that the BE relative error is always negligible as it is always $< 2.5\%$ throughout all the three cards while that of LBE is not. The MAX LBE Relative Error with Sign is therefore necessary to work out the boundary of the operating region which is used to compare the three manufacturers' cards in the next section.

2.1.2. *M-LREWS (Max LBE Relative Error with Sign)*

Apart from a glitch showed by Juniper, the entire bandwidth allocation couplet axis is a 'max' operating region for both Juniper and Procket with the latter performing slightly better.

It is difficult to determine a 'max' operating region for Cisco as the error is not monotonically decrescent but it is oscillating around the value 2.5. As a consequence, a conservative 'max' operating region is that which ranges from 75–24 to 50–49 which is 31.5% of the overall bandwidth allocation couple axis.

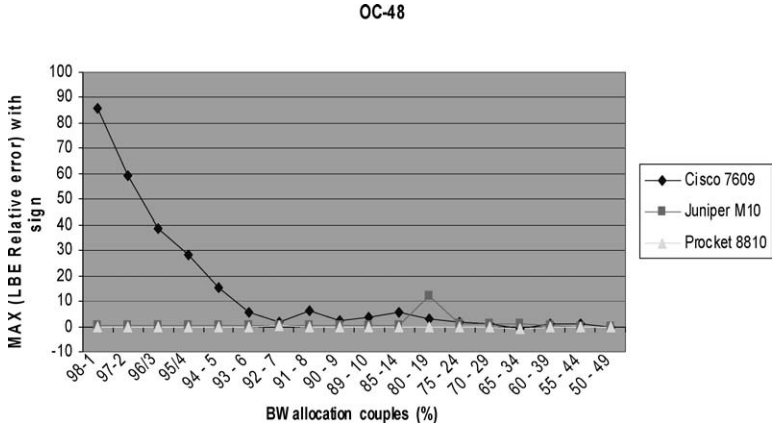

Figure 3. Max LBE relative error.

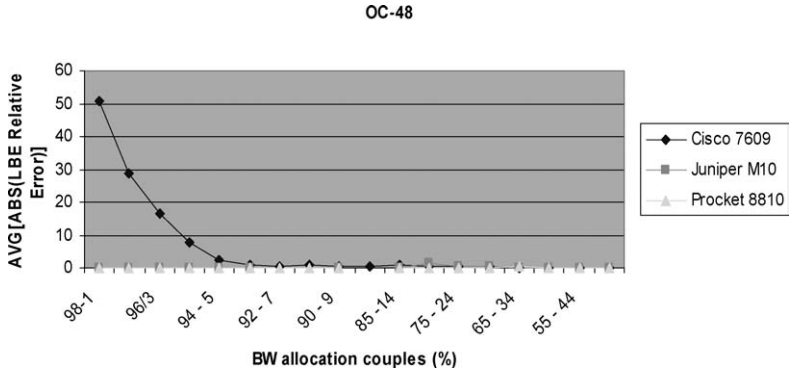

Figure 4. Average absolute LBE error.

2.1.3. *A-ALRE (Average Absolute LBE Relative Error)*

The above plot shows that the Cisco operating region averaged over the whole port congestion levels axis ('avg' operating region) ranges from 94–5 included to 50–49.

It is worth noticing how the average lowers the values but also acts as a low pass filter with the effect of smoothing out those oscillations that led before to a conservative and therefore rather poor performance evaluation.

A summarising Table 1 reporting both the errors and the percentage improvement (Δ) in passing from the 'max' to the 'avg' operating region is presented below.

The error is concentrated on LBE and presents positive polarity, this revealing, along with the negligible BE error and with the poor link utilisation, that the scheduler's issue is the inability in allocating the BE leftover bandwidth to LBE under a certain range of port congestion levels

The graphs clearly show that Procket has got the best performance out of the three and that Juniper is closer to Procket than to Cisco.

It is worth highlighting that Cisco 'avg' operating region shows a huge improvement of 117.2% over the 'max' operating region, which reveals that the magnitude of the per-bandwidth allocation couple max errors is actually rather localised over few port congestion levels. What follows is a plot showing the Cisco per bandwidth allocation couple LBE MAX error over the port congestion levels.

As expected, it is clear that each error curve presents a pronounced maximum which is notably greater than most of the values involved. In particular, this figure allows us to detect that the card presents the biggest errors in the region of the port congestion levels where BE is under-subscribed and LBE is over-subscribed, this corroborating the previous analysis according to which the nature of the errors is in the incapability of reallocating the BE left-over bandwidth to the over-subscribed LBE class.

Table 1. Comparative results.

OC-48	Cisco 7609	Juniper M10	Procket PRO 8801
Max op region	75–25 to 50–50 $6/19 = 31.5\%$	99–1 to 50–50 100%	99–1 to 50–50 100%
Avg op region	94–6 to 50–50 $13/19 = 68.42\%$ $\Delta = +117.2\%$	//	//

2.2. *TCP Stacks over DiffServ*

TCP's congestion control algorithm additive increase multiplicative decrease (AIMD) performs poorly over paths of high bandwidth-Round-Trip-Time product and the aggregate stability is even more precarious if this aggregate consists of only few flows [14].

These well known facts pose a serious question on how to effectively deploy data-intensive Grid applications given that multi-gigabit per second capacity can be provisioned on trans-continental links for few users/flows at any one time. So, although there is in principle spare capacity, TCP as it is now, will impact negatively on the performance of these new applications and will compromise the whole concept of high performance computational Grids working across wide geographical regions at high-speeds.

This work proofs as necessary the use of both new TCP stacks and IP-QoS and investigates in order to

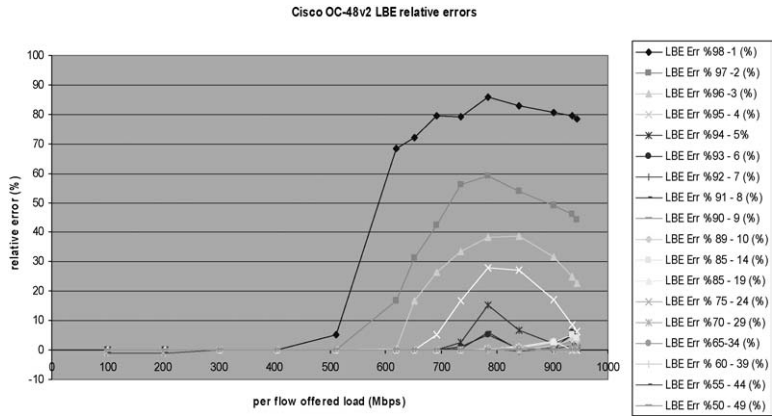

Figure 5. LBE relative errors for CISCO OC-48 card.

find the conditions under which IP-QoS and new TCP stacks can coexist in such a way that high utilisation as well as good stability and inter-class protection is achieved.

The figure below shows how poor is the throughput that standard (vanilla) TCP (on Linux 2.4.20 SMP) gets, even when the competing background CBR traffic is very small which, in principle, would free a big fat pipe out of the 1GE link to take advantage of. On the contrary, the performances showed by the new TCP implementations are notably better.

But we still need a mechanism for segregating traffic sharing the same packet-switched path. This is in order to protect traditional BE traffic from the possible aggression of the new TCP stacks and to guarantee a level of end-to-end service predictability for the new TCP proposals which is sufficient to enforce a network resources reservation system through the use of the Grid middleware.

Three IP QoS classes are therefore configured: BE for traditional Best Effort traffic (WEB-like traffic), and AF whose traffic is preferentially treated over BE.

The QoS configurations we used have the following rationale: since the numerous BE traffic aggregate utilises only a small pipe while the AF class, which has a much smaller number of flows, uses a much larger pipe, it follows that the bandwidth-delay product of the former is much smaller than that of the latter. The BE pipe is therefore a much more responsive and stable aggregate than the AF pipe. Thus, all the QoS configurations adopted are tailored to protect the AF traffic as much as possible.

Moreover, the minimum bandwidth guaranteed for BE is always smaller than that allocated to AF. This practically reflects the short/medium term Grid scenario where a lot of new capacity is not followed by a significant increase in the traditional BE usage of the bandwidth.

2.2.1. *Vanilla TCP in the AF Class*

We start this first QoS test by giving a chance to Vanilla TCP to use the preferentially treated AF class.

Standard TCP in the AF class competing with CBR UDP traffic in the BE class is presented below.

A plot of Vanilla TCP competing against 2 × 904 Mbps BE flows aggregate is presented below.

It is evident that, as soon as BE traffic is introduced, AF Vanilla flow drops to zero and does not recover anymore.

This result shows that even when QoS is configured, standard-TCP is still unable to take advantage of the privileged treatment the AF class provides and this is mostly due to the fact that the standard-TCP behaviour is so erratic that the percentage of time the interface is in congestion is too small to have QoS active full time.

Two very differently behaving flows feed the scheduler: a highly persistent UDP flow and an extremely erratic and elastic TCP vanilla flow. This mix causes the port and thus the scheduler to spend very little time in congestion state, which in turn causes QoS not to be sufficiently active. We refer to it as the 'scheduler pre-emption incapability' which is entirely due to the erratic vanilla TCP behaviour.

Vanilla TCP in AF still exhibits what is already known in high bandwidth delay product links as a lack of reactivity, which leads to not even reaching the equilibrium throughput.

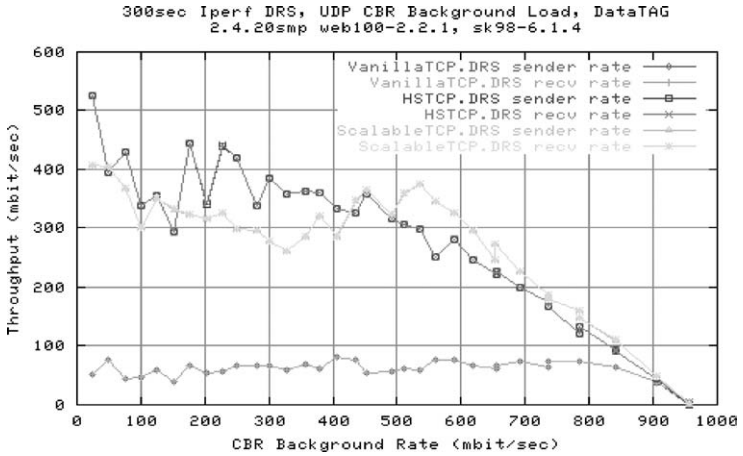

Figure 6. TCP implementation comparison with background traffic.

From this result, we draw the important conclusion that new TCP proposals should be hosted in the AF class while BE should host the standard-TCP flows.

This new scenario is investigated in the following section.

2.2.2. *Scalable TCP in the AF Class*

Scalable TCP [15] is taken as a representative example of the new TCP stacks family. This choice was made because Scalable TCP is the most extreme proposal up to now, this way allowing the investigation at the edge of the parameters space.

Keeping the same approach as for the test before, the load of the CBR BE traffic is lowered from 940 to 904 Mbps, which is exactly the same load used for testing Vanilla TCP in the AF class and that gave very poor performances.

Surprisingly, Scalable TCP clamps at the throughput expected and the same happens for BE. The joint flow dynamic is that the AF traffic hits the QoS pipe it was reserved and backs-off. As soon as it reduces, the interface is not congested anymore and CBR BE traffic can temporarily fill the AF leftover bandwidth until Scalable flow is back again.

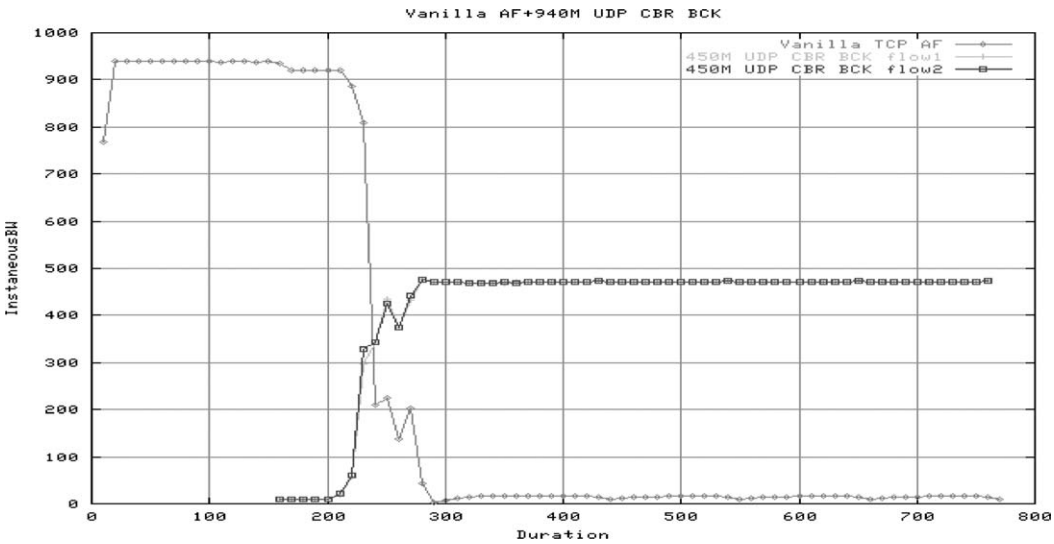

Figure 7. Standard TCP in AF class vs. two CBR flows in BE class.

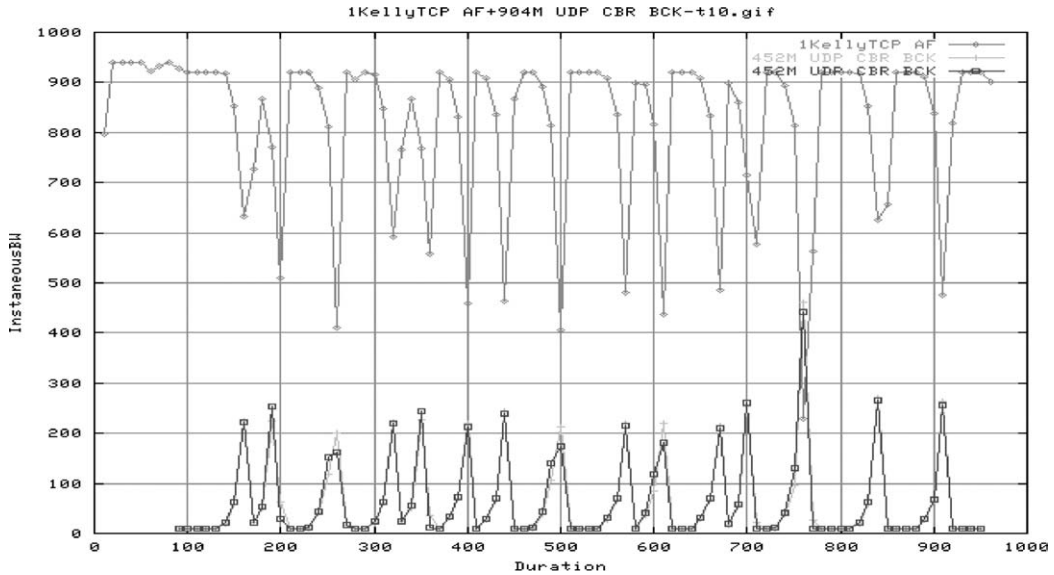

Figure 8. Scalable TCP in AF class vs. two CBR flows in BE class.

In summary, the short/medium term envisaged Grid scenario highlights the design limitations of the current (standard) TCP. New TCP proposals are therefore necessary. Moreover, in a packet switched network paradigm, a way of segregating traffic is also necessary. This is in order to protect traditional BE traffic from the possible aggression of the new TCP stacks and to guarantee a level of end-to-end service predictability for the new TCP proposals, which is sufficient to enforce a network resources reservation system through the use of the Grid middleware.

3. MPLS

MPLS – Multiprotocol Label Switching [16] has its origins in the IP over ATM effort. People soon realized that a label switching technology could be extended to other layer 2 technologies and complement IP at a global scale.

MPLS is considered by some as a layer 2.5 technology since it resides between IP and the underlying medium. It adds a small label to each packet and the forwarding decision is made solely based on this label. To establish these labels a signaling protocol, like RSVPTE [17], must be used. These signaling protocols allow specifying explicitly the route that the MPLS flows will take bypassing normal routing tables (see Figure 9). Here we can see Label Edge Routers (LER) classifying traffic into a specific label and Label Switch Routers forwarding traffic according to these labels.

MPLS has two major uses: Traffic Engineering and VPNs – Virtual Private Networks. In this work we were mainly concerned with the former. The ability to switch based on a label, as opposed to traditional IP forwarding based solely in the IP destination address, allows us, in certain scenarios, to manage available bandwidth in a more efficient way. Since Grid applications have traffic flows orders of magnitude bigger than traditional applications but will represent a small percentage of the flows in the network, it becomes cost efficient to select dedicated paths for their flows. This way we can be sure that the network complies with the QoS constraints and that the bandwidth available in the network is used in a more efficient way.

Unfortunately, at the time of writing, MPLS implementations we used do not allow MPLS traffic to be isolated in case of congestion. Although the signaling protocol allows specifying a bandwidth for the flow, this will not be a guaranteed bandwidth in the case of congested link(s). To force this bandwidth guarantee, traffic needs to be policed at any possible congestion point in the network.

In MB-NG we executed tests to verify if MPLS could be used to reserve bandwidth for a given TCP flow. In Figure 10 we can see the result of reserving an MPLS tunnel for a given TCP connection using explicit routes. Not only can we optimise the network

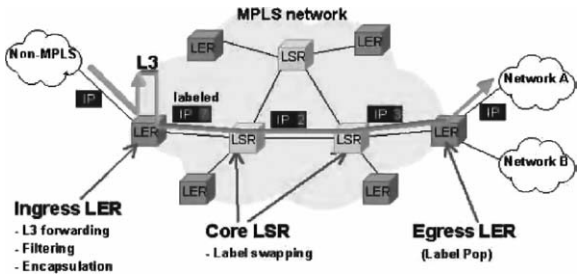

Figure 9. MPLS example.

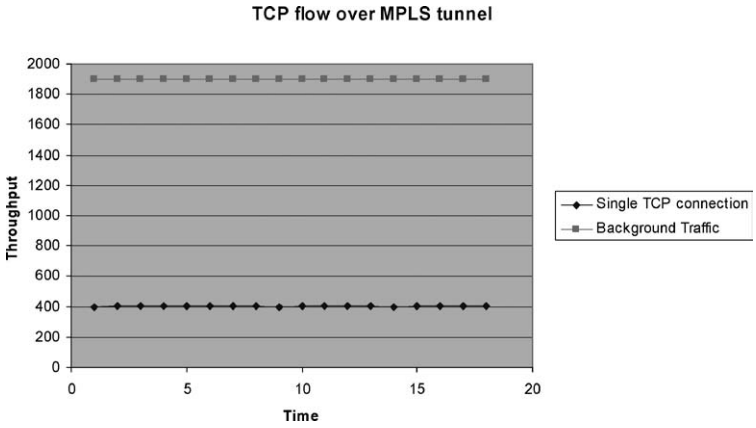

Figure 10. MPLS for flow reservation.

bandwidth but we could guarantee with very good precision a 400 Mbits/s connection to our application (in this case just simulated traffic with *iperf*). The typical TCP saw-tooth behaviour did not prevent us from having a very stable TCP connection in a congested network.

A big advantage of using MPLS is the traffic engineering feature that allows fault recovery in a much more efficient way. The fact that you have a signaling mechanism for bandwidth allocation in the control plane coupled with a matching QoS configuration in the forwarding plane, allows for, in the case of a link going down, a secondary path to be found much more quickly preserving bandwidth allocation.

We have confirmed this experimentally with several tests in the MBNG network. We injected traffic at 588 Mbits/s and at some point removed a link in the normal data path. With normal IP routing traffic was rerouted but with a loss of more than half a million packets. With MPLS failover recovery the loss was just 19 packets.

This can be of extreme importance when providing guaranteed bandwidth services to applications with high reliability requirements. Not only guaranteed paths can be allocated on demand with possible explicit routes but failover paths are found more quickly, with obvious advantages for the applications.

4. Service Level Agreements

An important issue in the implementation of an end-to-end Differentiated Services enabled network is the definition of Service Level Agreements (SLAs) between Administrative Domains. Because individual flows are only inspected on the edge of the network, strong, enforceable agreements about the traffic aggregates need to be made at each border of every pair of domains, so that all the guarantees made by all the providers can be met.

One of the goals of MB-NG as a leading multi-domain DiffServ experimental network is to provide guidelines for the definition and implementation of Service Level Agreements.

We are trying to standardise the definition of IP Premium (or EF – Expedited Forwarding [18] in the DiffServ literature). This is to be used by applications that require tight delay bounds. The SLA is divided in two parts: an administrative part and a Service Level Specification part (SLS). The SLS contains information about:

- Scope – defines the topological region to which the IP premium service will be provided.
- Flow Description – this indicates for which IP packets the QoS guarantees of the SLS will be applied.
- Performance Guarantees – the performance guarantee field depicts the guarantees that the network offers to the customer for the packet stream described by the flow descriptor over the topological extent given by the scope value. The suggested performance parameters for the IP Premium are:
  - One-way delay.
  - Inter-Packet delay variation.
  - One way packet loss.
  - Capacity.
  - Maximum Transfer Unit.
  - Traffic Envelope and Traffic Conformance.
  - Excess treatment.
  - Service Schedule.
  - Reliability.
  - User visible SLS metrics.

This SLA template is inspired by the one DANTE defined in [19] and can be read in more detail in [20].

5. The Control Plane Software

The final piece for providing the potential of QoS enabled networks to the applications is a usable and efficient control plane with APIs accessible to end-system applications. Even when the network is configured to support Differentiated Services and/or MPLS it is unreasonable to assume human intervention for configuring every flow request. There has to be a way for applications to request resources from the network. In the Integrated Services Architecture, applications would use a signaling protocol like RSVP [21] to signal the network and enable appropriate allocation of resources. In a DiffServ network resources are not allocated along the entire path for a specific application level flow, only for an aggregate. The literature [22] describes two mechanisms to achieve this: in the first RSVP is used and DiffServ clouds are seen as single hops. In the second a Bandwidth Broker [23] per domain is used. The application contacts the Bandwidth Broker, which is responsible for checking, guaranteeing and possible reserving resources.

We have instigated a project to develop such a solution, which we call Grid Resource Scheduling (GRS),

following the Bandwidth Broker model. The initial design was detailed in [24]. Our goal is to enable Grid users (or applications acting on their behalf) to micromanage network capacity allocations at the edge of the network. In recognition of the current usage within the Grid community, we simplify the the problem somewhat and assume that the bottlenecks are edge links and the core is over-provisioned. While, in general, this simplification may be considered too strong, in our particular experiments so far – within the UK and running over the MB-NG network and only for EF traffic, it provides a way to make a fast-start with experiments and so gain experience for use in more complex network scenarios.

Our Bandwidth Broker is called the Network Resource Scheduling Entity (NRSE) and we currently require one per edge domain. Client systems in an edge domain send reservation requests to their local NRSE. The local domain will have agreements with their upstream domain to carry DIFFSERV EF traffic. The NRSE knows how much DIFFSERV bandwidth is available, given existing reservations and checks whether the new reservation can be accommodated. If it can, it contacts the NRSE in the remote edge domain where the same check is made, before the reservation is admitted. When the reservation is due to begin, the NRSEs at each edge domain instructs the edge router to begin marking packets belonging to the specified flow as DiffServ EF.

We have the notion of *non-real-time* traffic. This is, essentially, data traffic that can be modeled as a large file transfer with strict deadlines for arrival at the remote system.

Reservations for *non-real-time* traffic only may be modified by the NRSE. For example, local policy might dictate that large file transfers happen overnight. If it is not possible to schedule a continuous block of bandwidth, the NRSE may choose to split a non-real-time reservation into multiple smaller reservations. As long as the deadline is met (i.e. the file is transferred by the deadline), the operation of the application should not be perturbed.

The NRSE protocol uses human readable XML for signaling between clients and NRSEs (*a* in Figure 11), as well as between NRSE peers (*b*). SLA requests contain token bucket parameters, filter specifications, authentication credentials, start times, etc. As well as booking reservations, the protocol supports administrative functions (e.g., querying the NRSE's scheduling table) and signaling operations (e.g., signaling to a client that a reservation is about to end).

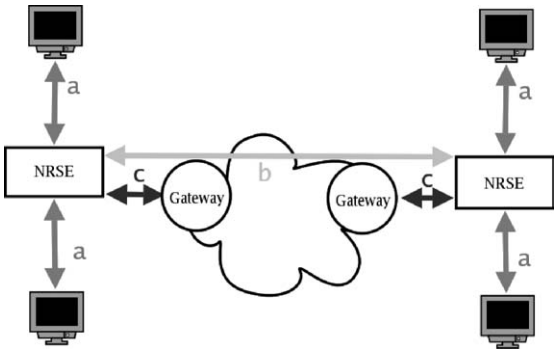

Figure 11. GRS architecture.

The NRSE, in the current context, scales well because reservation state is only stored at the edges of the network. Also, authentication is peer-to-peer. Users authenticate with their local NRSE, which in turn authenticates itself with a remote NRSE, so there is no need for a global database of users. Domain administrators are free to choose their own policies for authentication as well as reservation scheduling.

5.1. *Implementation*

We have now implemented the NRSE for the Java platform. We have used the BEEP application protocol framework [25] to carry our XML-based GRS protocol. Reservations are stored in a PostgreSQL database, providing long-term persistence for advance reservations. Authentication is performed currently using PGP signatures, but this mechanism is extensible and we intend to add other types of authentication.

The interface between the NRSE and the edge router(s) is modular (*c* in diagram), and a module to support Cisco routers is now being written and is being tested on MB-NG.

We have a low-level client library which exposes the full functionality, as well as a higher level libraries (in progress) designed to be easy to bolt on existing applications. We have produced an FTP client which uses this library to make reservations automatically for its file transfers.

Java has performed acceptably on our current testbeds, but we have not yet investigated scaling NRSE to a large number of domains. We intend to implement clients and servers for other platforms. We are also looking at the case where the bandwidth bottleneck is not in either of the edge domains and also at the problem of advance reservations.

Note that the NRSE is not dependent on the Globus platform or any other Grid middleware.

5.2. *Application Performance*

The concluding part of the work reported here was to execute demonstrations of high performance applications on our QoS testbed. These kinds of applications will drive future network research and are, therefore, a vital piece to our work. We have conducted tests with three applications: High Performance Visualisation, Radio Astronomy and High Energy Particle Physics. Here we present quantitative results on the High Performance part.

High Performance Computing (HPC) Visualisation applications have different requirements than pure data transfer applications. Because they are frequently interactive, they have tight delay constraints in both directions of the communication. When these requirements are coupled with high transfer rates (between 500 Mbits/s and 1 Gb/s) the necessity of new network paradigms becomes evident.

5.2.1. *Introduction*

The high performance computing (HPC) application used was OpenGL VizServer (http://www.sgi.com/software/vizserver/). This is a component of SGI's visual area network (VAN) solution. It is the product on which the Reality-grid (http://www.realitygrid.org/) is based. OpenGL Vizserver is a remote graphics processing system which allows users to view and interact with large data sets from a remote server and to collaborate with multiple colleagues using these same applications and data.

OpenGL Vizserver provides a variety of visualisation demo applications. For our tests, we used the demo called 'smoke and mirrors.' This demo is a 3-dimentional computer generated image of a room with mirrors on the walls. The viewer's position in the room is constantly changing and various objects appear, undergo a series of rotations and translation and then disappear. The user can interact in real time by interactively changing the position of the viewer.

For this experiment, we measured the throughput and inter-packet time for the initial 15 or so seconds of the application. The server ran on an SGI Onyx based in London and the client was on a PC running the linux operating system based in Manchester.

The tests were done with and without a QoS policy applied in the network. The policy allocated 20% of a 2.5 Gbit/s bandwidth to EF traffic, that is 500 Mbit/s. The scheduler served EF traffic in strict priority, that is, if EF traffic was present, it was served before any other traffic up to the 500 Mbits/s limit.

5.2.2. *Performance without QoS*

Figure 12 shows the average throughput of the application in the presence of varying degrees of background traffic. As expected, the throughput of the application decreases with increasing background traffic.

The maximum total received rate (application and background) was 2316 Mbit/s. The average application throughput between 65 Mbit/s and 75 Mbit/s was

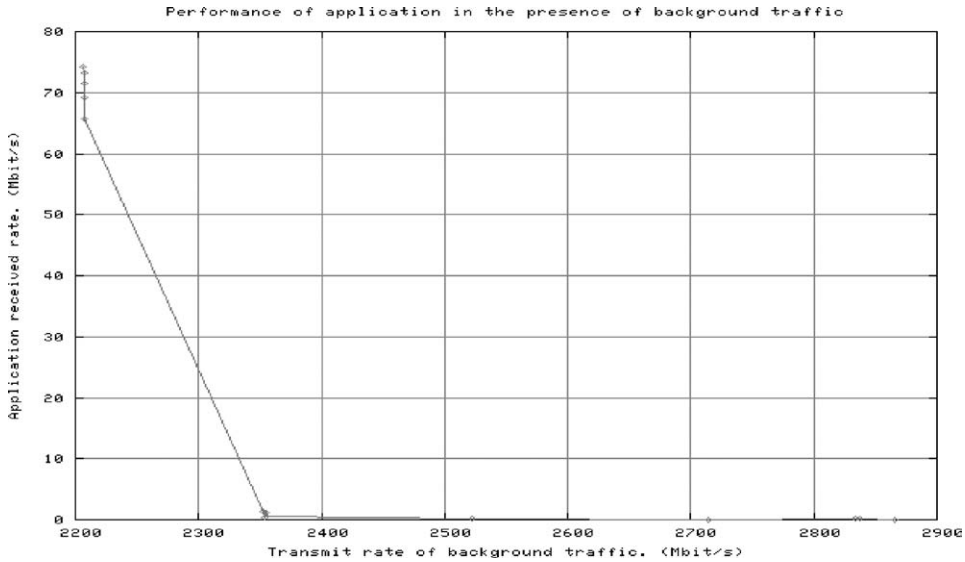

Figure 12. The achieved throughput of the application as a function of the background traffic.

sufficient for a usable refresh rate. The corresponding background rate was 2.208 Gbit/s. However, given a maximum achievable rate of 2316 Mbit/s, the application has 108 Mbit/s available. Due to the granularity of the background traffic generator [26], it was difficult to assess the performance of the application in the region where the background traffic lies between 2.2 Gbit/s and 2.3 Gbit/s.

Figure 13 shows how the normalised cumulative distribution of the inter-packet time (jitter) varies with different amounts of background traffic. This plot shows the percentage of packet with a given inter-packet time or less. As expected, with larger background the inter-packet time of the application traffic grows.

5.2.3. *Performance with QoS Enabled*

To show the performance of QoS, we ran the application with maximum background (2.87 Gbit/s) and without background traffic. The throughput achieved by the application without and with the maximum background is shown in Figure 14. In both cases, the average throughput was around 75 Mbit/s. This demonstrates QoS working to protect the application traffic from the background.

Figure 15 shows the normalised cumulative frequency distribution of the inter-packet time when the experiment was run with maximum and without background. The figure shows that in both cases, the majority of the packets arrives with a minimum inter-packet time of 12 microseconds less than that of Figure 13.

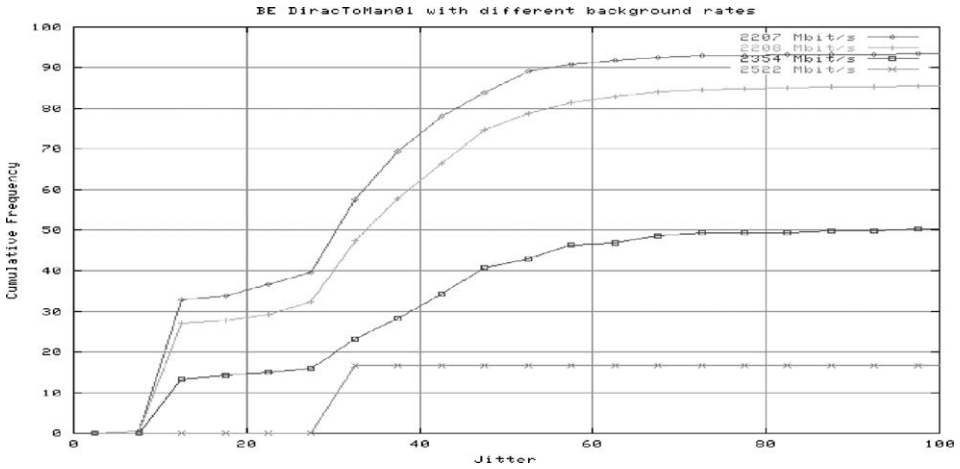

Figure 13. The normalised cumulative inter-packet time (jitter) of the application with different background traffic rates.

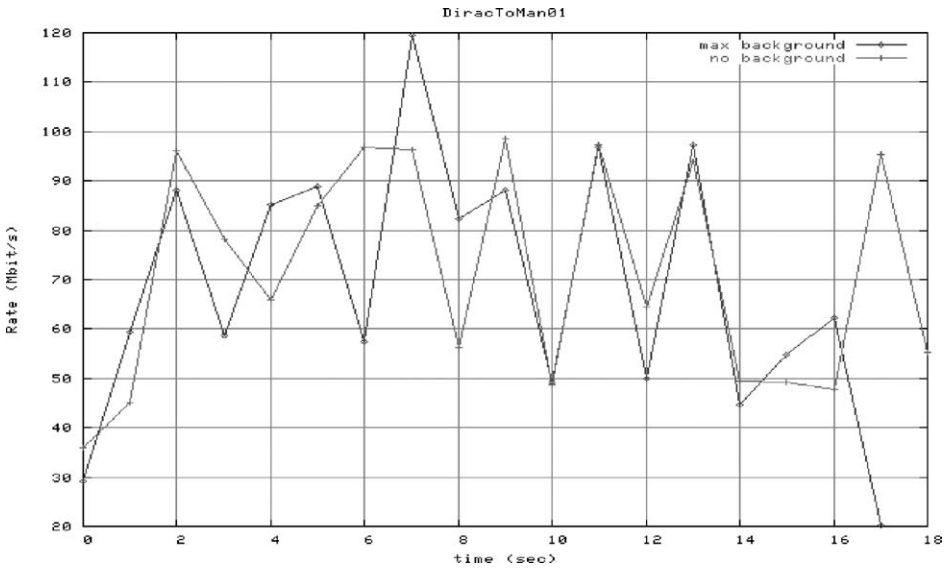

Figure 14. The performance of the application with and without the maximum background traffic. QoS enabled.

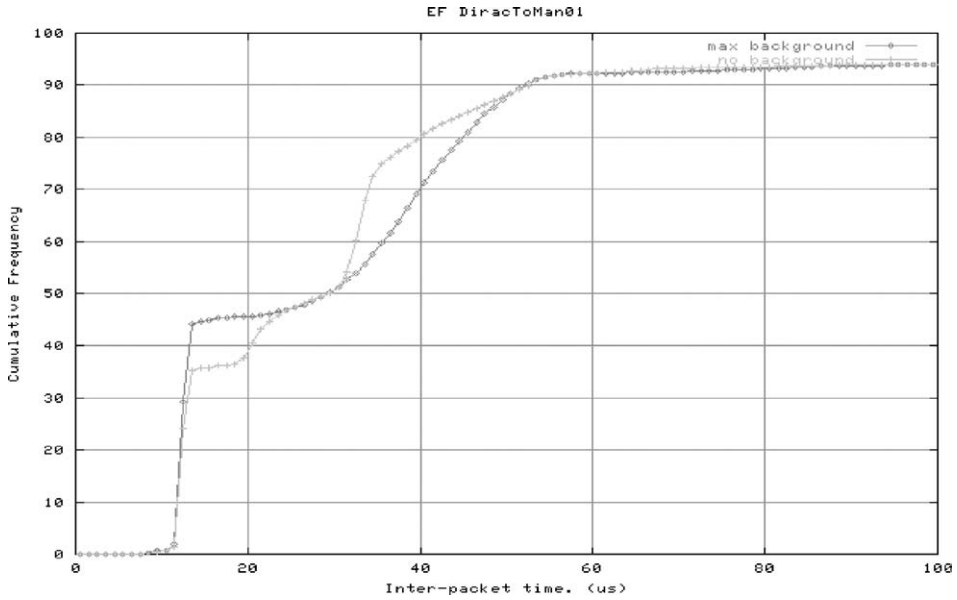

Figure 15. The normalised cumulative frequency distribution of the inter-packet time, without background and with a maximum background of 2.87 Gbit/s.

6. Conclusions

Experimental work in network testbeds plays a crucial part in network research. Many problems are undetected by analytical and simulation work but which practical experiments find in its early stages. They also provide good feedback for new topics of theoretical research.

In our experiments we concluded that quality of service networks will play an important role in future Grids and IP networks in general. Bandwidth overprovisioning of the core network does not solve all the problems and it will be impossible to guarantee end to end.

Both Differentiated Services and MPLS provide for the creation of valuable services for the scientific community with no major extra administration effort. We have shown that the choice of hardware to implement a Diffserv network is crucial and should take into account the specific bandwidth allocation that each class will have.

Since Grid applications will have to use modified TCP stacks we have studied the interactions between current and new versions of these stacks and DiffServ implementation. More aggressive TCP proposals should be mapped into an AF-like class with a significant percentage of bandwidth allocation.

We have implemented a Bandwidth Broker architecture using IETF standards that has been proven to work in real development networks. We are currently investigating the possibility of a NRSE-OGSA gateway [27].

We have demonstrated that QoS in our network is able to protect a real application from the effect of background traffic.

References

1. L. Breslau and S. Shenker, "Best-Effort versus Reservations: A Simple Comparative Analysis", in *Proceedings of SIGCOMM 98*, September 1998.
2. S. Floyd and V. Paxson, "Difficulties in Simulating the Internet", *IEEE/ACM Transactions on Networking*, Vol. 9, No. 4, 2001.
3. S. Floyd and E. Kohler, "Internet Research Needs Better Models", in *Proceedings of HOTNETS-1*, October 2002.
4. http://www.mb-ng.net.org
5. http://www.datatag.org
6. O. Ibe, *Essentials of ATM Networks and Services*. Addison Wesley, 1997.
7. R. Braden, D. Clark and S. Shenker, *RFC 1633 – Integrated Services in the Internet Architecture: an Overview*. June 1994.
8. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, *RFC 2475 – An Architecture for Differentiated Services*. December 1998.
9. http://www.cisco.com
10. http://www.juniper.com
11. http://www.procket.com
12. http://www.internet2.edu
13. C. Partridge, *Gigabit Networking*. Addison Wesley, 1993.
14. S.H. Low, F. Paganini, J. Wang and J.C. Doyle, "Linear Stability of TCP/RED and a Scalable Control", *Computer Networks Journal*, Vol. 43, No. 5, pp. 633–647, 2003.

15. T. Kelly, “Scalable TCP: Improving Performance in High Speed Wide Area Networks”, presented at *First International Workshop on Protocols for Fast Long-Distance Networks*, February 2003.
16. B.S. Davie and Y. Rekhter, *MPLS: Technology and Applications*. Morgan Kaufmann Series on Networking, May 2000.
17. “RSVP Signaling Extensions for MPLS Traffic Engineering”, White Paper, Juniper Networks, May 2001.
18. V. Jacobsen, K. Nichols and K. Poduri, *RFC 2598 – An Expedited Forwarding PHB*. June 1999.
19. C. Bouras, M. Campanella and A. Sevasti, “SLA Definition for the Provision of an EF-based Service”, Technical Report.
20. “SLA Definition – MB-NG”, Technical Report (work in progress).
21. L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala, “RSVP – A New Resource Reservation Protocol”, *IEEE Network*, Vol. 5, No. 5, September 1993.
22. Z. Wang, *Internet QoS: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann, San Francisco, 2001.
23. I. Foster, A. Roy and V. Sander, “A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation”, in *Proceedings of the 8th International Workshop on Quality of Service (IWQoS2000)*, June 2000.
24. S.N. Bhatti, S.A. Sorenson, P. Clarke and J. Crowcroft, “Decentralised QoS Reservations for Protected Network Capacity”, in *Proceedings of TERENA Networking Conference, May 2003*, 2003, pp. 19–22.
25. M. Rose, *RFC 3080 – The Blocks Extensible Exchange Protocol Core*. March 2001.
26. “The Iperf Traffic Generator”. http://dast.nlanr.net/Projects/Iperf/
27. I. Foster, C. Kesselman, J.M. Nick and S. Tuecke, “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration”. www.globus.org/research/papers/ogsa.pdf