NETWORK QOS FOR GRID SYSTEMS

Saleem N. Bhatti[1]
Søren-Aksel Sørensen[1]
Peter Clark[2]
Jon Crowcroft[3]

Abstract

Grid users may wish to have fine-grained control of quality of service (QoS) guarantees in a network in order to allow timely data transfer in a distributed application environment. We present a discussion of the issues and problems involved, with some critical analysis. We propose possible solutions by making reference to and analysing existing work. Also, we describe the mechanisms being proposed as part of a work-in-progress (being conducted by the authors) that uses a peer-to-peer approach to micro-manage network capacity allocations at the edge of the network, at end-sites, in a multi-domain scenario. Scheduling controllers at the end-sites are employed, which are subject to local administrative controls and have flexibility in resource allocation based on user requests for network capacity. We highlight the issues in scaling such systems to large numbers of users and the issues concerning the interfaces available to applications and end-users for accessing such services.

Key words: Author: please provide key words

The International Journal of High Performance Computing Applications, Volume 17, No. 3, Summer 2003, pp. 000–000
© 2003 Sage Publications

1 Introduction and problem statement

Consider two users (applications) wishing to share some large data files and who have a specific deadline for the transfers of those files from one site to another. The users would like to signal the network to request some *protected capacity* to ensure the data are transferred by the deadline they have specified. This may be, for example, because the files are used as part of a large distributed processing application and so synchronization of input/output processing across the distributed system is required. Other users may have more traditional network resource requirements; for example, they wish to have a video conference between two sites using AccessGrid technology (AccessGrid, 2002). In both these cases, the users need to have some level of quality-of-service (QoS) guarantee. Additionally, these users may be geographically dispersed and so the QoS control mechanisms they use, both at the network level and application level, must operate across the boundaries of various administrative domains along the network path.

Normally, applications assume that the network capacity they require for such operations is always available to them, and do not explicitly model into the design of the application any mechanisms to deal with situations where such capacity is not available. This has not been a major issue for many applications used for high performance computing for a number of reasons: for example

- applications tend to be running on single-site operation with high-speed local area connectivity between any distributed processing units;
- applications have a mode of operation that can cope with highly asynchronous communication, e.g. message passing, transactional services (e.g. based on remote procedure calls);
- applications tend not to require *large* amounts of data (tens or hundreds of MB) to be transferred between them across wide-area networks.

For the last of these points, in a non-Grid environment, where such movement of data is required, it is dealt with out-of-band, e.g. by sending tapes or hard disks by courier between sites. One reason for the existence of Grid systems is to use network services to transfer data

[1] COMPUTER SCIENCE, UNIVERSITY COLLEGE LONDON (UCL), UK

[3] PHYSICS AND ASTRONOMY, UNIVERSITY COLLEGE LONDON (UCL), UK

[4] COMPUTER LABORATORY, CAMBRIDGE UNIVERSITY, UK

and so make such distributed processing easier. So, for high performance computing, using large datasets in a distributed environment we may need to ask the network for a high-capacity *reservation* in order to provide an assurance of timely data transfer. There are many users within the Grid community with such needs, e.g. astronomers, bio-informatics researchers and high-energy physicists. So, for such users (amongst others) we require network-level mechanisms to enable QoS services and application-level mechanisms to use such services. Additionally, in some cases, the applications themselves may need to adapt their operation in order to accommodate the interaction with the network that this will involve. Normally, we would like to provide QoS for a specific *flow* of information. In our case, we consider a flow to be a sequence of packets with some sort of semantic relationship at the application level.[1] Assuming communication using the Internet Protocol (IP) suite, a flow is identified in terms of header fields found in IP packets: for example destination and source addresses, destination and source port numbers. Whilst the identification of flows can be performed using a much more complex combination of header fields from the network-level, transport-level and application-level headers, in this paper, we will use only simple examples to illustrate our discussion. It is assumed that the reader is familiar with the basic functional aspects of IPv4, IPv6, TCP, and UDP.

1.1 QOS SERVICES ARCHITECTURE

IP-based networks, and the Internet itself will be used to allow communication across Grid Systems. The IP and the Internet were never designed to handle QoS-sensitive traffic and so the Internet community must evolve the network and enhance the Internet protocols in order to cater for the needs of these new and demanding applications (Bhatti, 1999).

Clark et al. (1992) speak of the Internet evolving to an *integrated services packet network* (ISPN), and identify four key components for an Integrated Services architecture for the Internet:

1. **service-level**, the nature of the commitment made by the network for a network flow, and a description of a set of control parameters to describe traffic patterns;
2. **service interface**, a set of parameters passed between the application and the network in order to invoke a particular QoS service-level, i.e. some sort of signalling protocol plus a set of data structures and parameter definitions;
3. **admission control**, for establishing whether or not a service-request can be honoured before allowing the flow to proceed;
4. **scheduling mechanisms within the network**, the network must be able to handle packets in accordance with the QoS service requested.

As an example, a simple description of the interactions between these components is as follows.

- A service-level is defined (e.g. within an administrative domain or, with global scope, by the Internet community). The definition of the service-level includes all the service semantics; descriptions of how packets should be treated within the network, how the application should inject traffic into the network as well as how the service should be policed. Knowledge of the service semantics must be available within routers and within applications.
- An application makes a service-request invocation using the service interface and a signalling protocol. This typically happens *before* data packets are transmitted. The service-request includes specific information about the traffic characteristics required for the flow, e.g. data rate. The network will indicate if the service invocation was successful or not. Also, once data packets start to flow, the network may also inform the application, via a well-defined interface and protocol, if there is a service violation, either by the application's (mis)use of the service, or if there is a network failure.
- Before the service invocation can succeed, the network must determine if it has enough resources to accept the service request. This is the job of admission control, which uses the information in the service request, plus knowledge about the other service requests it is currently supporting, to determine if it can accept the new request. The admission control function will also be responsible for *policing* the use of the service, making sure that applications do not use more resources than they have requested. This will typically be implemented within the network routers that handle the packets. Policing policies dictate the action to be taken when packets are found to be violating the parameters in the service request. Such actions include dropping the packet (easy to implement but may damage the flow), remarking the packet into a "worse" QoS class (easy to implement but could then affect "well-behaved" traffic using the other class) and re-shaping or delaying packets (harder to implement and may still spoil the performance of the flow due the additional delay).
- Once a service invocation has been accepted, the network must employ mechanisms that ensure that the packets within the flow receive the service that has been requested for that flow. This requires the use of scheduling mechanisms and queue management for

flows within the routers to ensure correct packet handling.

In this paper, we deal mainly with the application-level issues in points 1 and 2 above (service definitions and service interfaces), with some discussion of point 3 (admission control) as it affects the application. Issues related to point 4 (packet handling in the network) can be found in Bhatti and Crowcroft (2000).

2 Network-level QoS mechanisms

In our overall discussion, we intend to concentrate on the application-level aspects. However, we need to consider the network-level mechanisms, as the services offered at these levels must be understood in order for the applications to make correct and appropriate service-requests. The network-level mechanisms are concerned with enabling routers to treat certain packets with a "better" service than others. The definition of "better" constitutes the QoS *service-level* or in some cases *service-class* that is being requested for a flow.

There has been much work in the community for over two decades to address the problem of QoS control. Initially, work concentrated on the network-level issues and the kind of mechanisms required in the routers to allow correct packet handling (classification, queuing, scheduling and forwarding of packets). While work in this area continues, current research takes in areas of application-level and middleware QoS issues also. In this section, we present some related work in the network-related areas and discuss issues and problems that arise. Note that the presentation here is not exhaustive; we have selected and focused on work and activities that are current at the time of writing and relevant specifically to Grid systems.

2.1 IETF INTSERV

The IETF INTSERV WG (INTSERV) has proposed an architecture for evolving the Internet to an Integrated Services Network (ISN). To support the architecture, INTSERV have produced a set of specifications for specific QoS service-levels based on a general network service specification template (Shenker and Wroclawski, 1997a) and some general QoS parameters (Shenker and Wroclawski, 1997b). The template allows the definition of how network elements should treat traffic flows. With the present IP service enumerated as *best-effort*, two service-level specifications are currently defined:

- *controlled-load* service (Wroclawski, 1997a) – the behavior for a network element required to offer a service that approximates the QoS received from an unloaded, best-effort network;
- *guaranteed* service (Shenker et al., 1997) – the behavior for a network element required to deliver guaranteed throughput and delay for a flow.

Also specified is how to use a signalling protocol, RSVP (Braden et al., 1997), to allow the use of these two services to be signalled through the network (Wroclawski, 1997b). Part of the INTSERV work is the definition of an architecture for a QoS Manager (QM) entity that coordinates flow activities and resource usage at the end system. Note that this architecture requires that the network elements and applications have semantic knowledge about the service-levels for the application flows, as specified in the service templates.

RSVP is used by applications to make a *resource reservation*, by asking the network to provide a defined quality of service for a flow. The reservation request consists of a *FlowSpec* identifying the traffic characteristics and service-level required. One part of the *FlowSpec* is a *TSpec*, a description of the traffic characteristic required for the reservation. So it is possible for the same traffic characteristic to be used with different service levels. (This difference in QoS service-level could, for example, act as a way for offering cost differentials on the use of a particular application or service.)

To invoke a particular service (make a service-request), the application uses RSVP, for a particular communication session (which may consist of one or more flows). To make a resource reservation, an appropriate *FlowSpec* is used along with session IP destination address, the protocol number in the IP packet and, optionally, the destination port number in the service invocation. The reservation procedure is as follows. The sender transmits a *Path* message (effectively "advertising" the session QoS requirements) towards the destination IP address. All RSVP-enabled routers forwarding the *Path* message hold *soft-state* – information about the resource reservation required – until one of the following happens: a *PathTear* is sent from the sender cancelling the reservation, a *Resv* message is transmitted from a receiver effectively confirming the reservation, or the soft-state times out. A Resv message from a receiver is sent back along the same route as the *Path* message,[2] establishing the reservation and then the application starts sending data packets. *Path* and *Resv* messages are sent by the sender and receiver, respectively, during the lifetime of the session to refresh the soft-state and maintain the reservation. A *PathTear* or *ResvTear* message explicitly tears down the reservation and allows resources to be freed. It is possible for the reservation to be changed dynamically during the lifetime of the session. RSVP can be used for unicast or multicast (many-to-many) sessions.

Note that RSVP

- provides end-to-end signalling (between applications)
- RSVP sets up unidirectional reservations
- is specific to one session
- requires the applications and the network elements all to be RSVP and INTSERV aware.

2.2 PROBLEMS WITH INTSERV AND RSVP

The main issue concerning Integrated Services provisioning is the handling of the individual packets that make up a flow in order to honour the QoS requirements of that flow. The router has a non-trivial forwarding process for each packet:

- classify the packet in order to identify its QoS requirements (classification);
- determine when the packet should be forwarded (scheduling);
- manage the output queues under congestion conditions (queue management).

Note these activities are logically distinct from the *routing* functions that all routers must be able to perform in order to determine in which direction to *forward* a packet (i.e. which output interface should be used). Several schemes have been developed within the Internet community for performing classification, scheduling and queue management tasks, and they are currently undergoing experimentation and development (Bhatti and Crowcroft, 2000).

RSVP uses a soft-state technique with a two-pass protocol. We summarize the main problems with RSVP below (Wolf et al., 1997).

1. During reservation establishment if the first pass of each of two separate reservation requests are sent through the same network element, where one request is a super-set of the other, the lesser one may be rejected (depending on the resources available), even if the greater one eventually fails to complete (of course it is possible to re-try).
2. If the first pass does succeed, the router must then hold a considerable amount of soft-state for each flow and this burden of state maintenance increases as you approach the core of the network where many more flows pass through the network routers.
3. The routers must communicate with receivers to refresh soft-state, generating extra traffic, otherwise the reservation will time out.
4. If there are router failures along the path of the reservation, this results in IP route changes, so the RSVP reservation fails and the communication carries on at best-effort service, with the other routers still holding the original reservation until an explicit tear-down or the reservation times out or the reservation can be re-established along the new path.
5. The applications must be made RSVP aware, which is a non-trivial goal to realize for the many current and legacy applications that already exist, including multimedia applications with QoS sensitive flows.

Resource reservation could be expensive on router resources and adaptation capability is still required within the application to cope with reservation failures or lack of end-to-end resource reservation capability. Indeed, the Internet community has acknowledged the shortcomings of RSVP, especially with respect to scalability, and it is recommended for use only in restricted network environments (Mankin et al., 1997). Such concerns about resource reservation have directed the Internet community to consider alternatives; specifically *differentiated services*. Without resource reservation, we require some mechanisms to allow service differentiation within the network, but also we require a more flexible and *dynamic adaptation* capability within the application.

2.3 IETF DIFFSERV

The IETF DIFFSERV WG takes a different view of using network resources to that of the INTSERV WG (Blake et al., 1998). The general model is to define a class-based system where packets are effectively marked with a well-known value. This value identifies the aggregate service-level the packet will receive along with other packets with the same value, much like a letter can be marked as first-class or second-class delivery. This is a much coarser granularity of service, but reflects a well-understood service model used in other commercial areas. The DIFFSERV model is quite different to the INTESRV/RSVP model. A key distinction of the DIFFSERV model is that it is geared to a business model of operation, based on *administrative bounds*. Whereas RSVP can act on a per-flow basis, each of the DIFFSERV classes may be used by many flows simultaneously. Any packets within the same class must share resources with all other packets in that class. The packets are treated on a per-hop (per router) basis by *traffic conditioners*, functions in the router that determine the way a packet should be treated based on a policy that is selected by examining the value of the class marking of the packet. The policy could be applied to all the traffic

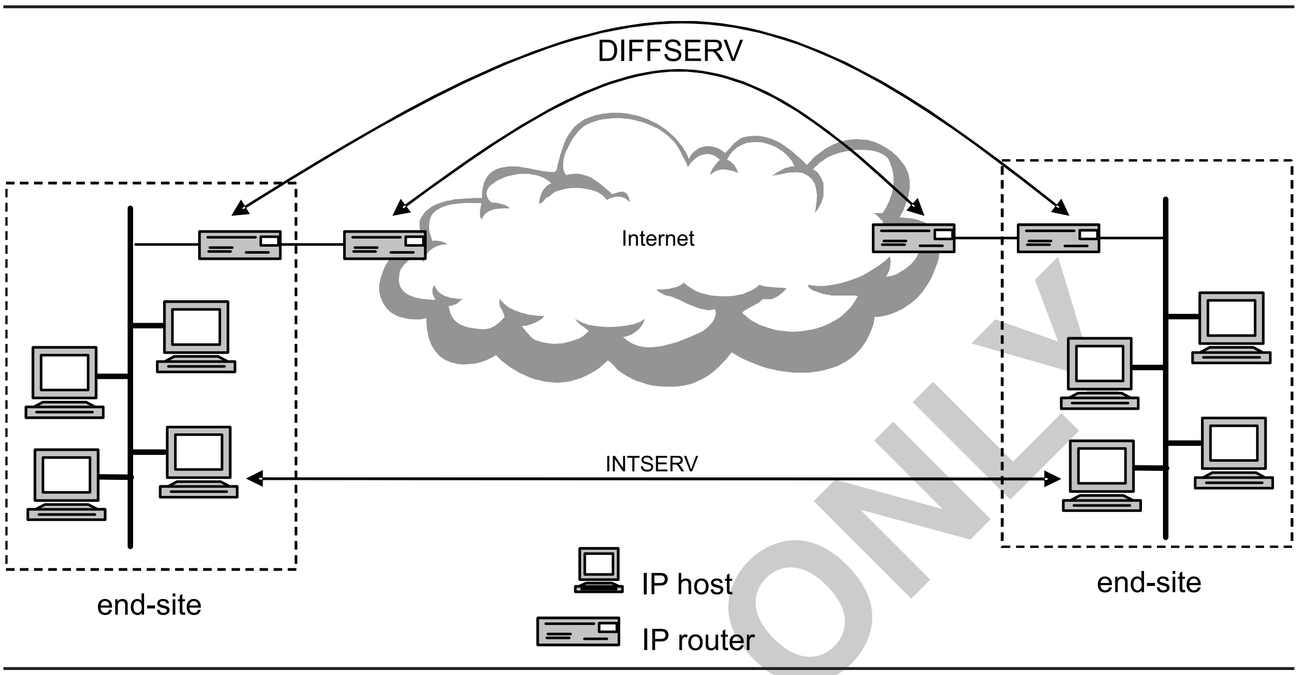


Fig. 1 Scope of INTSERV compared to DIFFSERV.

from a single user (or user group), and could be set up when subscription to the service is requested, or on a configurable profile basis. The DIFFSERV mechanisms would typically be implemented within the network itself, without requiring runtime interaction from the end-system or the user, so are particularly attractive as a means of setting up tiered services, each with a different price to the customer.

The INTSERV/RSVP mechanism seeks to introduce well-defined, end-to-end, per-flow QoS guarantees by use of a sophisticated signalling procedure. The DIFFSERV work seeks to provide a "virtual pipe" with given properties in which the user may require adaptation capability or further traffic control (if there are multiple flows competing for the same "virtual pipe" capacity).

The QoS service itself will be defined in terms of a *service level agreement* (SLA) that embodies the contract between the service user and service provider. The policy implemented by the SLA may include constraints other than QoS that must be met, e.g. security, time-of-day constraints, etc. Figure 1 highlights the main difference between INTSERV and DIFFSERV scope. INTSERV tries to provide, per application, end-to-end resource reservation. DIFFSERV aims to provide an SLA-based contract between service networks. One very attractive feature of DIFFSERV is that it can be introduced into existing networks in a piecewise manner, without having to modify current or legacy applications. The packets leaving a network are marked for DIFFSERV handling by DIFFSERV-capable routers that sit at administrative boundaries. Therefore, only these routers need to be updated and the applications themselves can remain unchanged. (However, this does not preclude individual hosts or individual applications being DIFFSERV-aware and marking packets accordingly as they leave the host.) The DIFFSERV-capable routers could be at the edge of the customer network or part of the provider's network. If the DIFFSERV-marking is performed within the customer network, then policing is required at the ingress router at the provider network in order to ensure that customer does not try to use more resources than allowed by the SLA.

The DIFFSERV work is aimed at providing a way of setting up QoS using policy statements that form part of a service level agreement between service user and service provider. The policy may use several packet header fields to classify the packet, but the classification marking is a simple identifier (currently a single byte) within the packet header. The classification is by way of a special value for a single header field, the DS (*differentiated services*) byte, which will be used in place of the

ToS (Type of Service) field in IPv4 packets or the traffic-class field in IPv6 packets. The DS byte will have the same syntax and semantics in both IPv4 and IPv6. There are already defined some global DS values – *DS codepoints* (*DSCPs*) – agreed for the DS field within the IETF but the intention is that the exact policy governing the interpretation of the DSCPs and the handling of the packets is subject to some locally agreed SLA. SLAs could exist between customer and Internet Service Provider (ISP) as well as between service providers. The DSCPs are used to identify packets that should have the same aggregate *per-hob behavior* (*PHB*) with respect to how they are treated by individual network elements.

The meaning of the DSCPs and the content of SLAs are established at subscription time and, although there will be scope for change by agreement between customer and provider, the kind of dynamic and flexible resource reservation that is described above for using RSVP is not envisaged for DIFFSERV.

DIFFSERV is not without its problems. However, these would appear to be less difficult to overcome than those for INTSERV and RSVP.

2.4 PRACTICAL USAGE OF THE IETF WORK

It would be possible to make use of both INTSERV and DIFFSERV without modifying the application. A network management tool could be used by network administrators to configure the network routers appropriately before data packets started to flow. However, such a mechanism is clearly not scalable to a large number of users (ignoring the other scaling problems of INTSERV) and does not give the level of control that Grid users might expect. Such users would prefer to have mechanisms that are directly accessible from the application via an API or specific middleware. However, we may need to apply local administrative controls (as well as security and access control) if general access to these services is permitted from users via client systems in this way.

The use of DIFFSERV has received great interest lately, as users and researchers begin to understand how to deploy and make use of SLAs across administrative domains. However, there still remain problems in trying to provision DIFFSERV. Some standard PHBs have been defined and these are being used provide specific services, such as Expedited Forwarding (EF; Davie et al., 2002) for low loss, low delay services, and Assured Forwarding (AF; Heinanen et al., 1999) for high throughput adaptive services that can tolerate some loss. There is also a Less-than Best Effort (LBE) PHB (Shalunov and Teitelbaum, 2001) which is intended to be used to provide services for long-lived flows using reliable transport for applications such as large file transfers and continuous-update applications. Such flows would be treated as low-priority or "background flows" so they could use any network capacity that remains unused by other flows, such as best-effort flows and EF-marked flows. However, in the presence of other traffic, LBE would have a fixed minimum allocation so that LBE-marked flows are not starved (e.g. 1% of total capacity) but have lower priority than best-effort flows.

There have been numerous activities within the research community to try to provide mechanisms for QoS control in IP-based networks, mostly focusing on the use of DIFFSERV, in projects such as AQUILA (1999), TEQUILA (2002), MESCALE (2002), Task Force – Next Generation Networking (TF-NGN, 2002) and QBone (2002). These have concentrated on the mechanisms and protocols needed to provide a network service which allows some kind of QoS control as well as aspects of admission control and management of the network resources. Indeed, AQUILA, TEQUILA and MESCALE are EU-funded projects that try to provide a practical solution for some of the problems and issues that arise in a DIFFSERV network trying of offer a real service, e.g. definition and management of Service Level Agreements (SLAs) and mapping of Service Level Specification (SLSs). TF-NGN and QBone look at the practical aspects of deploying the mechanisms in a service environment. Recently, the use of DIFFSERV QoS mechanisms at high-speeds is also being investigated by some of the authors in the Managed Bandwidth – Next Generation (MB-NG, 2002) project, which involves the UK Education and Research Networking Association (UKERNA) and Cisco as project partners.

A problem that is encountered in all these projects is the idea of allowing QoS reservations. The original INTSERV work is not capable of scaling to Internet-wide usage, so various communities have decided to build their own systems. The e-Science community uses General-purpose Architecture for Reservation and Allocation (GARA, 2002) and the authors are involved in a decentralized approach for DIFFSERV EF reservations within the Grid Resource Scheduling (GRS, 2002) project. We will provide more details of both of these later.

3 Application-level Grid access to network QoS services

For an application to be able to request QoS services, it must signal the network with information about (at the very least):

- the QoS service that is required;
- the identity of the flow that is to receive that QoS service;
- QoS-service-specific parameters, such as the data rate required, etc.

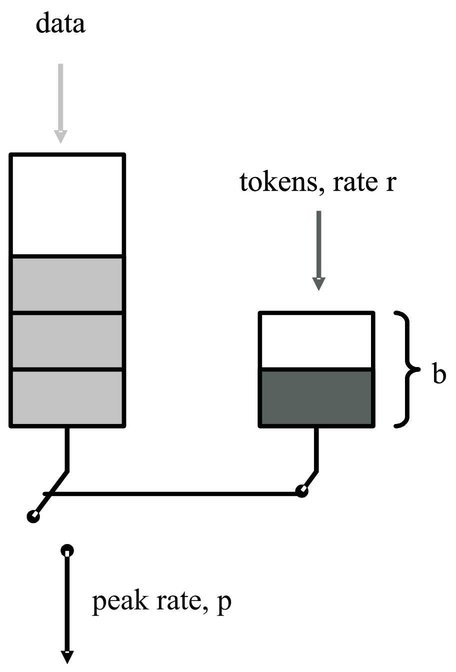
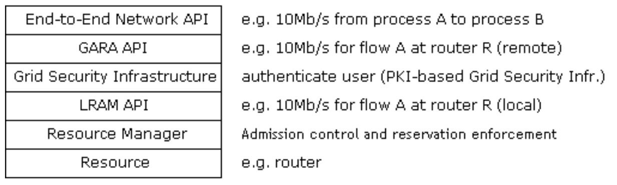


Fig. 2 The token-bucket traffic descriptor.

For DIFFSERV, the QoS service (or class) that is required for a flow is indicated by the DSCP value in the packets for that flow. For identifying a flow, we can use packet header fields such as source and destination IP address. For describing the traffic parameters, however, it is not immediately obvious what we should use. An obvious but naïve approach would be to ask for the expected mean data rate of the flow. Using a mean data rate to make a reservation does not account for the *burstiness* of data traffic (ratio of peak rate to mean rate). Asking for a reservation of the mean rate only would mean that peak rate bursts might be subject to policing policy (e.g. packets are dropped). Asking for a reservation of the peak rate may be inefficient if the flow does not transmit at the peak very often. The model used within the IETF work recognizes the bursty nature of traffic and uses the *token bucket* to describe traffic characteristics. The token bucket parameters are:

- *r*, the token rate;
- *b,* the bucket size;
- *p,* the peak rate;
- *m,* the minimum policed unit (effective minimum packet size);
- *M,* the maximum packet size.



End-to-End Network API	e.g. 10Mb/s from process A to process B
GARA API	e.g. 10Mb/s for flow A at router R (remote)
Grid Security Infrastructure	authenticate user (PKI-based Grid Security Infr.)
LRAM API	e.g. 10Mb/s for flow A at router R (local)
Resource Manager	Admission control and reservation enforcement
Resource	e.g. router

Fig. 3 The GARA architecture (source http://www-fp.mcs.anl.gov/qos/gara.htm).

The key parameters are *r*, *b* and *p*, and the relationship between them is explained with help of Figure 2.

The bucket starts full with *b* bytes. The bucket is filled with tokens, which control the flow of data. If the token has *b1* tokens in it, then *b1* bytes of data may be transmitted. The bucket is then drained of *b1* bytes of tokens. This is the size of the burst that can be sent, at a maximum rate (peak rate) *p*. Note that *p* is not necessarily the maximum transmission rate of the network interface. The bucket fills again at the rate *r* bytes per second. At any time *t*, the amount of data sent, *d* is such that $d \leq rt + b$.

This acts as a traffic control at the application end but also allows the network to police the flow.

The parameters *m* and *M* are used by the network, as required and we do not discuss these details here. Typically, an application would set these values to correspond to application-specific needs. Note that *M* is considered to be a maximum, but packets smaller than *m* can be transmitted, but will, for policing purposes, be treated as being of size *m*.

3.1 GARA

GARA (2002) is popular among the Grid community as an application-level interface to the network QoS services. In fact, GARA is intended as a general purpose platform allowing reservation of numerous resources, including CPU cycles and disc space, not just network capacity. GARA provided a simple and useful platform for early adopters of the Grid who required reservation capability. Figure 3 shows the GARA architecture. The End-To-End API is a high-level API that is specific to a particular resource, e.g. network capacity, CPU, etc. Below that sits the main GARA API. This makes use of the Grid Security Infrastructure (GSI) for authentication and access control. A Local Resource Access Manager (LRAM) provides controlled access to the local resource (network, CPU, disc etc) via a Resource Manager (RM).

As used for network reservation, the End-to-End Network Reservation (EENR) API is currently under development to allow co-ordinated reservations across multiple domains. This requires that a reservation be made in each *domain* that the end-to-end network path will traverse. This is not necessarily as restrictive as the RSVP soft-state which requires each *router* along the network path to maintain state. However, the existence of the EENR will mean that the application need not contact each domain and make the reservation itself. Nevertheless, a GARA system must exist at each domain boundary and maintain state about all reservations.

The GARA API allows reservations to be created, modified and cancelled including the ability to have callbacks to notify the application of changes to any reservations that have been made. A simple string-based mechanism is used to specify the resource requirement. The attributes that can be specified for the network resource are:

- *endpoint-a, endpoint-b* – the IP addresses of the two end-points of the reservation;
- *Bandwidth* – network capacity required, in Kb s^{-1};
- *Directionality* – whether the reservation is uni-directional or bi-directional.

Although GARA does not specifically need Globus (2002) to function, invariably Globus is used. GARA has been used to demonstrate the provision of reservations using DIFFSERV (Sander et al., 2000).

3.2 PROBLEMS WITH GARA

GARA is not designed specifically to deal only with network QoS control so it has drawbacks. It allows reservations to be made for CPU cycles and disc space as well (for example). Such resources are fundamentally very different to network capacity. Resources such as CPU cycles and disc space are localized to certain end-systems and reservations can be made at the remote end-systems where such resources are located. Network capacity is a distributed resource requiring reservations at the local *and* remote end-systems as well as the *network path* between the local and remote systems.

So, the architecture requires GARA/Globus systems to be present in each domain that the network path will traverse. This is so that the some state about the reservation can be held on each of the domains. Also, the EENR is still being developed so in a multiple domain scenario, so it will be necessary to make explicit reservations along each domain of a network path. This means discovering the network path, locating the GARA server and then requesting a reservation. The same is true when reservations have to be modified or deleted. Of course, if a reservation is not deleted but not used (e.g. application is terminated or application changes its operation), the reservation remains in place and cannot be released for other users, although GARA does require that reservations are bound to specific flows when the reservation is required. Also, if the network paths changes, due to the dynamic routing used for IP, then the reservation will fail if the network path change is not detected and a new reservation established along the new path. In this case, the network flow might suffer even if the new path does have capacity available for the given service-class, just because the reservation state has not been established. So, maintaining the reservations state is troublesome. However, dynamic routing changes can be hard to deal with in any circumstance not just with GARA.

The API and Resource Specification Language (RSL) do not allow the specification of the policing mechanism for a request (e.g. drop, re-mark, re-shape). Also, the RSL does not allow the port numbers of the flow to be specified and they must be specified in a separate binding stage using the API. The reason for doing this is not clear, nor how this is to allow dynamic flows to be established but the way this is reflected down to the resource in a timely way. Additionally, it is not possible for the user to specify the token bucket parameters described above. A knowledgeable network applications programmer can adjust the values of these parameters to tune the end-to-end delay that the flow experiences across the network. Of course, it is only useful to allow these additional parameters to be passed across the interface if they can be communicated to and used by the underlying resource (e.g. router) in a meaningful way.

Another problem that exists, but may be resolved with the EENR, is that for a user to be able to contact all the GARA servers along a network path to establish an end-to-end reservation would require that user to have the correct security and access control at each GARA server. This could make management of users hard for large numbers of users and dynamically changing populations. Indeed, it is evident that this will not scale well for Grid users and Internet-wide usage. For GARA, this has been addressed in a model that allows inter-Bandwidth-Broker communication (Sander et al., 2001) but has yet to see wide usage.

4 The Network Resource Scheduling Entity

In the GRS project (2002) the authors are developing a Network Resource Scheduling Entity (NRSE) to perform a similar job to GARA but specific to reserving network capacity. We have used the excellent work in GARA to inform our own work. We intend that as many of the mechanisms that are we developing will be gen-

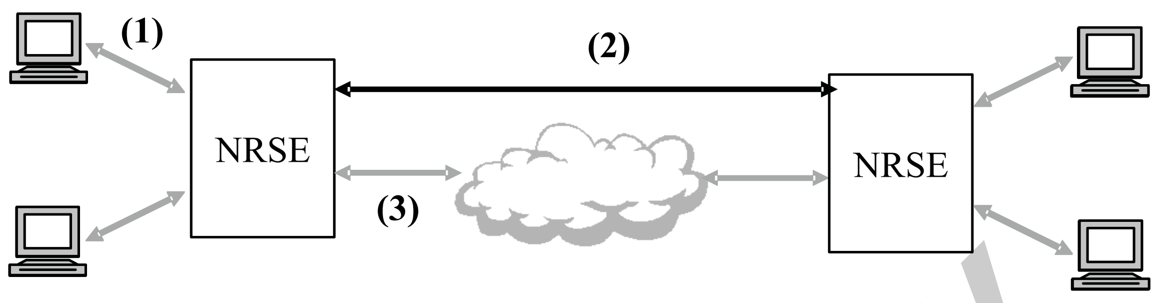


Fig. 4 The NRSE reservation model.

eral enough to be adopted for use for scheduling of other resources, not just network capacity. The main features planned for the NRSE are (in no particular order):

- allows reservations of network capacity across multiple domains based on SLAs;
- allows decentralized state for the reservations and does not require NRSEs to hold reservation state at all domain boundaries, just at end-sites which are involved in the reservation;
- allows creation, deletion and modification of reservations;
- allows configurable notifications from NRSEs to applications regarding violations to QoS;
- the flow identification can use most of the standard IP, TCP and UDP header fields;
- QoS service class, directionality and policing can be specified in the SLA;
- the NRSE uses a localized polling mechanism for the application holding the reservation (keep-alive) so that resources can be reclaimed if an application fails;
- allows flexible scheduling of jobs to deadlines for non-real-time service-requests (e.g. file transfers);
- can be configured with local policies that control the operation of the system, with such local policies being autonomously managed;
- has a hierarchical trust model so that security and access control information remains as localized as possible.

These features will be explained below, with further discussion about general issues in multi-domain reservation in a separate section.

4.1 GENERAL MODEL

We have seen with QoS reservations using INTSERV and RSVP and also with GARA, that signalling and per-flow state overhead can cause end-to-end QoS reservation schemes to scale poor to large numbers of users and multi-domain operation. We address this in our work by storing the per-flow/per-application state only at the end-sites that are involved in the communication. Service requests can be dynamic or can be in advance. The amount of capacity that is allocated to dynamic/advance reservations at each end site is controlled by local policy, and configured by local network administrators.

In Figure 4, Network Resource Scheduling Entities (NRSEs) at each end-site (each domain) are responsible for receiving user service-requests, checking if the request can be honoured site-to-site, and then issuing configuring instructions to the local network elements in order to provide the requested QoS. The arrows show the paths for signalling messages. Host end-systems may send to the NRSEs requests for a protected QoS allocation, in the form of a local; service level agreement (SLA). NRSEs then communicate to arrange "booking" of the request at both ends of the communication path (2). At the appropriate times, the NRSEs in each domain issue local instructions to program the requested QoS into the network elements. The signalling protocols (1) and (2) are application-level protocols, whilst (3) is a purely local protocol.

After a reservation has been made, the NRSE can inform the application if there are violations to the QoS, including if the reservation has had to be aborted for some reason.

Additionally, during the lifetime of the reservation, a scheme of keep-alive polls can be used between the NRSE and the application so that reservations are never left in place if an application is no longer active. Potentially, such polling could create a lot of traffic if used naively. The use and frequency of these keep-alive polls is subject to local policy. We would expect these to have a frequency of several 10s of seconds to several minutes. This can be determined by local policy, e.g. if it is a relatively large reservation, then we would poll frequently so that the resource can be reclaimed quickly if the application fails or stops. If the reservation is quite small, the polling period will be less frequent. Also, as the polling is in the local area, where network capacity is not normally so scarce, the polling traffic should not raise too much cause for concern.

```
<id> <!-- this will be used to identify the
                  request, along with
                  user_info -->
  <timestamp>2002-10-02-10150000</timestamp>
  <req_no>42</req_no> <!-- e.g. 32-bit ran-
                  dom number -->
</id>

<user_info> <!-- administrative info -->
  <user_name>Saleem Bhatti</user_name>
  <user_credentials>TBA<user_credentials>
</user_info>
```

4.2 THE SLA FROM THE USER

The SLA from the user contains some administrative information. The <user_credentials> are a certificate or other mechanism as specified by policy local to the NRSE. It is not required that two NRSEs have the same policy local to their site. Here we show a human user but the certificate could easily identify an application instead. The <timestamp> and <req_no> are local to the NRSE. The whole of the SLA is also signed (according to local policy) so that SLAs cannot be forged, modified or replayed.

In our scheme, a domain is defined by the presence of an NRSE. All resources that are controlled via that NRSE form part of the same domain. Local policy will dictate whether this is coincident with, for example, a DNS domain, e.g. cs.ucl.ac.uk.

```
<time_span> <!-- this is optional: if it is
                  not present then this
                  SLA should remain in place
                  until explicitly removed
            -->
    <!-- one of start_time and end_time must
                  be present, but both also
                  OK -->
    <start_time>2002-11-01-0000</start_time>
                  <!-- nearest minute is
                  fine here for now -->
    <end_time>2002-11-02-0000</end_time>
                  <!-- nearest minute is
                  fine here for now -->
</time_span>
```

Effectively, this SLA is between the user and the local NRSE. The local NRSE establishes a logically separate SLA between itself and the remote NRSE on behalf of the user. This separation between the inter-domain and intra-domain application-level relationships has some advantages, as we shall discuss later. Also, in the local scope, it provides a point at which local policy can be enforced and cannot be by-passed by users.

The SLA also contains an indication of when in time this reservation is to be executed; if <time_span> is not present, then the SLA is to be executed immediately and remain in place until it is explicitly removed. Only the user who creates an SLA or a local administrator can delete or modify the SLA.

The SLA also, of course contains information about the reservation to be made, including the identity of the flow, the traffic description of the flow, and the QoS service-class required:

```
<filter>
  <!-- could also have other level3 or
                  level4 header fields
     but just stick with these for now
  -->
  <src_ipv4>128.16.10.1</src_ipv4> <!-- could
                  be name or address -->
  <src_port>1284</src_port>
  <dst_ipv4>128.16.10.11</dst_ipv4>
  <dst_port>8080</dst_port>
</filter>

<tspec>
  <!-- all rates in Kbps -->
  <peak_rate>1000</peak_rate>
  <token_rate>800</token_rate>

  <!-- all sizes in bytes -->
  <bucket_size>2048</bucket_size>
  <min_policed_unit>48</min_policed_unit>
  <max_pkt_size>1024</max_pkt_size>
</tspec>

<qos>
  <quality>premium</quality>  <!-- "premium"
                  mapped to EF, "low" mapped
                  to LBE -->
```

```
    <policing>  <!-- for future, also "delay" or
                     "remark" are possible -->
      <action>drop</action>
    </policing>
    <direction_mode> <!-- {uni|bi}directional
                     (multicast for future) -->
      bidirectional
    </direction_mode>
    <flow_type> <!-- real_time or non_real_time -->
      non_real_time
    </flow_type>
  </qos>
```

4.3 NOTIFICATIONS

```
  <notifications> <!-- this is optional -->

    <notification_sink>  <!--multiple instances
                     of this are possible -->
      <dst_ipv4>morland.cs.ucl.ac.uk</dst_ipv4>
      <dst_port>4242</dst_port>
    </notification_sink>

    <start_notification> <!-- this is optional -->
      1 <!-- number of second before start of
                     reseravtion -->
    </start_notification>

    <end_notification> <!-- this is optional -->
      1 <!-- number of seconds before end of
                     reservation -->
    </end_notification>

    <notification_flags service_qos_viola-
                     tion="on" <!-- on or off -->
                   user_qos_violation="on"
                   abnormal_termination="on"
                   administrator_interven-
                     tion="on" />
  </notifications>
```

The SLA also allows a user to specify if they would like notifications sent from the NRSE under certain conditions. The user can ask for a notification to be sent just before the reservation starts and ends, as well as when the reservation is terminated due to QoS violations, and also when the SLA is modified or deleted through administrator intervention. Here, <notification_sink> specifies where the notifications are to be sent and can have multiple instances. <start_notifcation> specifies the number of seconds before the reservation actually starts that a notification should be sent; <end_notification> is similar but gives "warning" of the end of the reservation. This might be used, for example, by an application that may run over its reservation time to try to modify the reservation for a longer duration as required. It is also useful for distributed applications because, although the reservation may have been made for a certain time, fine-grained clock synchronization would be required to ensure that reservations and the start and end of data flows are coordinated. The use of the <start_notification> and <end_notification> allows well-defined synchronization points without the need for a distributed clock synchronization mechanism. The notifications can be implemented as callbacks or asynchronous events at the API and so are easily integrated into the application.

4.4 COMMUNICATION BETWEEN NRSES

```
  <id> <!-- this will be used to identify the
                     request, along with
                     user_info -->
    <timestamp>2002-10-02-10180000</timestamp>
    <req_no>19</req_no> <!-- e.g. 32-bit ran-
                     dom number -->
  </id>

  <user_info> <!-- administrative info - TBC -->
    <user_name>nrs001@cs.ucl.ac.uk</user_name>
    <user_credentials>TBA<user_credentials>
  </user_info>
```

Between NRSEs, and thus between domains, a trust relationship must be established. The administrative information used here reflects the communication between the NRSEs and does not identify individual users at the end sites. This means that the trust relationship between NRSEs is independent of the trust relationship between local users and their respective local NRSE. For the SLA above, when the information it contains is sent to the remote NRSE to check that he reservation is possible, the only real difference will be the identity and credentials.

Note that, here, the value of <id> differs from the user-side SLA as does the <user_name>. This is the local NRSE sending a request, on behalf of the user, to the remote NRSE. So trust relationships should be more manageable than when end-users have to contact remote-sites (or intermediate sites) directly. It is left for the local administrators of each domain/NRSE to agree the nature of the credentials used between them to establish trust, and this is in keeping with the peer-to-peer model of the interaction between NRSEs.

When QoS violations or administrator intervention occurs, notifications are sent between NRSEs involved in the reservation.

4.5 THE NRSE INTERNAL ARCHITECTURE

The NRSE itself has similar functional blocks to GARA (see Figure 5). The NRS-A is the application-facing interface and protocol, allowing SLAs to be created,

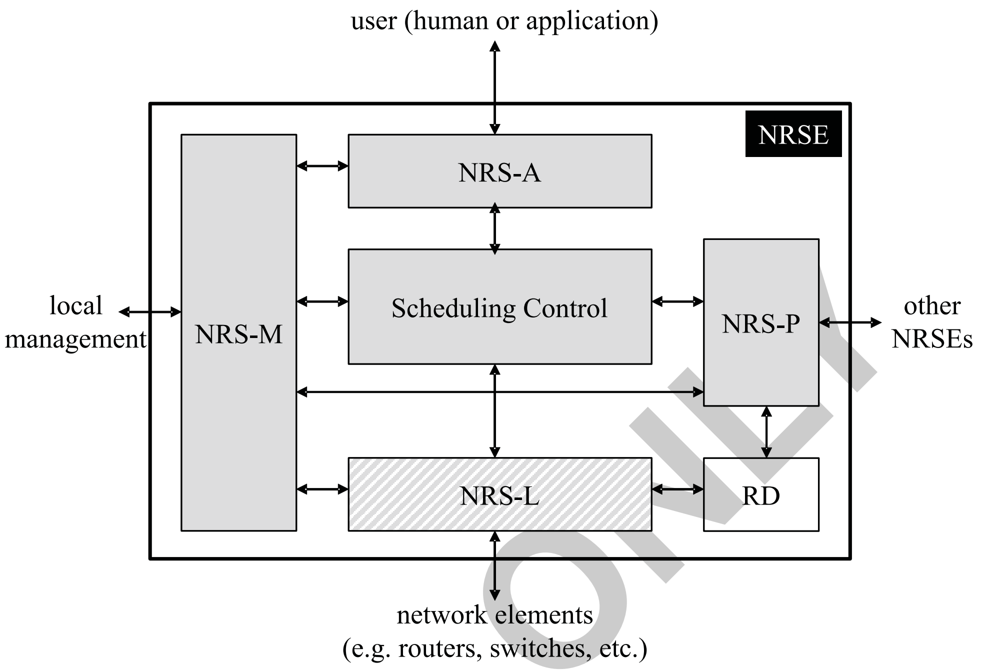

user (human or application)
NRSE
NRS-A
Scheduling Control
NRS-M
NRS-P
NRS-L
RD
local management
other NRSEs
network elements (e.g. routers, switches, etc.)

Fig. 5 The NRSE internal architecture.

deleted and modified, to send notifications and to allow keep-alive polling. The NRS-P is the peer-entity facing interface and protocol. This allows peer-to-peer communication between NRSEs and allows (via the NRS-L) management of the trust relationships between NRSE peers.

The NRS-L is the local interface to resources, allowing, for example, configuration information for new flows to be passed to routers. This will be specific to the resource, e.g. with Cisco routers, one would expect to generate some CiscoIOS commands to control and configure the router in response to SLA requests. It is also used to monitor the progress of the flow and detect any QoS violations. Here, we will be re-using mechanisms being developed in other projects, such as DataTAG (2002), EU-DataGrid (2002), NetLogger (2002) and GridProbe (2002).

The RD function is resource discovery, allowing NRSEs to find each other. Here again, we intend to re-use mechanisms developed elsewhere. We are likely to provide mechanisms to allow multiple resource discovery protocols to be used as there is no consensus on how to provide service location and discovery within a Grid environment. So, various mechanisms could be used including, directory services ala Globus, UDDI (2002), the DNS (a similar mechanism to allow Internet mail-servers to be located) and the IETF Service Location Protocol (Guttman et al., 1999). The RD module could also be used by the NRSE to discover local network resources, which would need to be configured for providing QoS through localized resource discovery mechanisms.

The NRS-M is the local management interface allowing administrators to access to all aspects of the NRSE operation and configuration. End-sites typically employ their own local administrative controls. In order to allow users to have common services across a multi-domain communication path, there will need to be some common semantics at the end-sites in the offered QoS services. However, we should respect the autonomy of the local systems and allow them to impose any local controls they see fit on how those QoS services should be used, e.g. allocation of protected capacity (EF) versus best-effort capacity. So, the QoS control system must be manageable through local interfaces with local policy.

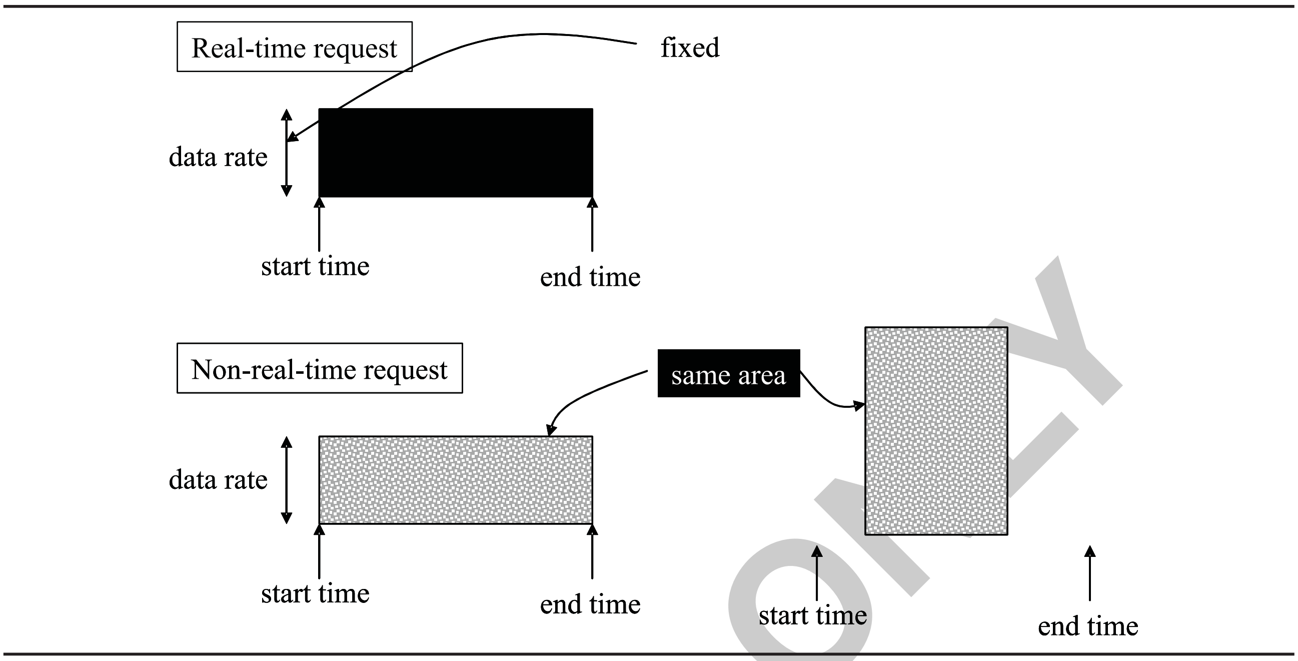


Fig. 6 Real-time and non-real-time jobs.

We will explain the role of the Scheduling Control separately, below.

4.6 SCHEDULING CONTROL AND RESERVATION TYPE

The Scheduling Control shown in Figure 5 is initially configured, subject to local policy and constraints, to allow some capacity allocation for both real-time and non-real time service requests. In both cases, a protected service is required, but there are different semantics to the booking. As shown in Figure 6, a real-time SLA request represents a booking for a fixed data-rate with specific start and end times, e.g. for a video call. A non-real time SLA request is for a protected and fixed data transfer capacity; the start time and end time are fixed constraints but the Scheduling Control is permitted, within these time constraints, to negotiate with the user as to how the request is actually fulfilled, e.g. for a file transfer for the distributed application example above. This allows some flexibility in how the non-real-time requests are scheduled within the protected service provisioning, allowing better distribution of service requests and more even use of the QoS services. The user can indicate the reservation type in the SLA – “real-time” or “non-real-time”.

4.7 USE OF DIFFSERV

Currently, the architecture assumes that DIFFSERV is being used. Specifically, we are currently aiming to allow reservations for EF capacity. We enable decentralized reservations with per-flow state held only at the edge sites by exploiting the DIFFSERV architecture with the use of SLAs between service provider and customers at the edges, but also the chain of SLAs that exist in the provisioning to the core within a DIFFSERV scenario. There are some difficulties in certain multi-domain scenarios (explained later) but the assumption is that a provider will not sell downstream a service that they cannot provide from their own resourcing, i.e. the services they have purchased upstream. How this is achieved within a particular service provider’s network is a decision for that provider. For example, a pilot, nationwide DIFFSERV service currently being used within UKERNA uses a conservative “sum of peak rates” policy for admission control. With such an assumption, the job of the NRSEs becomes to micro-manage the EF service for a domain, i.e. decide which packets originating from end-systems within that domain get marked as being EF, based on the SLA service-requests that are received. This means that the QoS state information regarding a flow need only be maintained at the edge-sites as this is the only place it is actually required. Indeed, for now, we may make a stronger assumption (at least in the current usage scenarios of research networks); that the core is over-provisioned.

4.8 BENEFITS TO GRID USERS

The main benefit that could be seen for Grid users is that of a "one-stop-shop" approach to making network reservations. When a user requires network capacity, even in a multi-domain scenario, as long as users have established a trust relationship within their local domain, they are then permitted, subject to local policy, to make network resource reservations for their applications based on SLA requests made to local entities, without requiring knowledge of the administrative details and trust relationships that must be set up with the remote sites. Taking the examples introduced earlier – the video conference (e.g. for example using AccessGrid) and the distributed application – we can show how the use of a QoS reservation service implemented using NRSE approach benefits Grid users.

For the video conference between two sites capacity could be reserved, separately, for the each of the media streams – audio and video. Two separate SLAs would be required, one for the audio flow and one for the video flow. A bi-directional reservation could be set-up by one end of the conference or two separate uni-directional reservations could be set up as required. These can be adjusted independently, via a single point of access (the NRSE), depending on the audio and video quality considered appropriate for the conference.

The other example given at the start of this paper was a high-speed data transfer. Consider a situation where a Grid user wishes data to be generated at P from a specific application then to be transferred to Q for processing and then to be transferred to R for visualization, displaying the result on his/her on terminal at S. The user may, of course, have to reserve computational and storage resources at P, Q, R and S in order to complete his/her task. The network resources required can then be reserved, independently, once the reservations for these other resources have been confirmed, specifying data transfer deadlines between P, Q, R and S based on the timing of the reservations for the other resources at these end-points.

Note that in both these scenarios, the co-ordination of the reservations of the various resources is considered to be an application-level issue. Indeed, we might consider that such co-ordinated, multi-domain, multi-resource scheduling could (and should) be something that is handled outside of the application. This would then become the function of a middleware substrate to co-ordinate activities between, say, the NRSE and other resource scheduling mechanisms. However, it is also evident that, with a very distributed application, such distributed scheduling co-ordination activities could become quite complex. So, expressing the requirements, dependencies and relationships between resources policies and SLAs in such scenarios could also be complex (and such issues are currently being researched within the Grid community).

4.9 CURRENT STATUS

The work on the NRSE is ongoing. Currently, we are investigating the provision of micro-management of access to a DIFFSERV EF service. Effectively, the NRSEs receive SLA requests containing a packet filter specification and a token bucket specification. A peer-to-peer relationship between NRSEs at end-sites is used to allow negotiation of the reservation for user requests and when the request is agreed, the state is stored at each end-site by the NRSEs – the core network incurs no extra overhead for state storage other than that which is required for normal DIFFSERV EF operation. There is some additional signalling between the NRSEs at each site, but this is application-level traffic and does not need to be inspected/checked by routers. The test-bed is being built on RedHat Linux 7.2 and TC (see http://diffserv.sourceforge.net/). The implementation is in Java and will use BEEP (Rose, 2001) for the communication with and between NRSEs. The work will be realized in four phases, starting with a single-domain, homogeneous scenario and extending to a multi-domain, heterogeneous scenario. There are a number of options being investigated for resource discovery. There are plans to offer an Open Grid Services Architecture (OGSA) interface (see http://www.globus.org/ogsa/) to this service towards the end of the project (Q2/2004).

The source and documentation will be made freely available to the research community from the GRS project Web page at http://www.cs.ucl.ac.uk/research/grs/ (Q2/2004) as and when stable releases become available.

5 Multi-domain issues in QoS reservation

The biggest problem that we see for any reservation system is that of dealing with reservations across multiple domains. Here, we must coordinate resource usage across multiple administrative boundaries. This is eased somewhat with the use of DIFFSERV and the NRSE holding

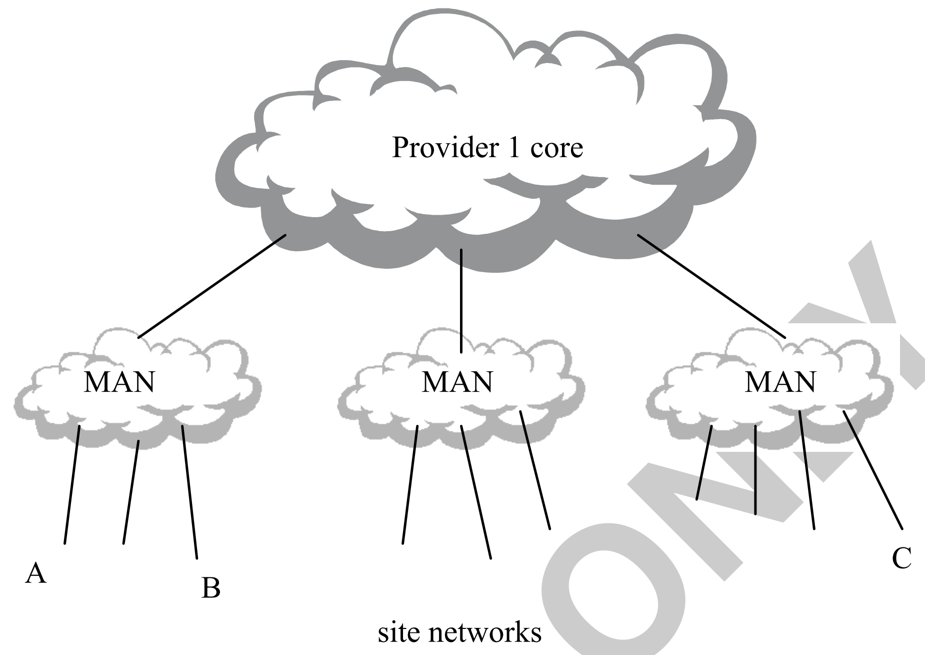


Fig. 7 A multi-domain scenario.

state only at the edges of the network. However, there are still some technical problems, which we highlight with examples from our ongoing involvement in various other collaborative projects.

5.1 MULTI-DOMAIN EF SERVICE CAPACITY

In Figure 7 we see a simplified version of the scenario currently being piloted in the UK by UKERNA. Universities and research institutes typically connect to the UKERNA core via Metropolitan Area Networks (MANs). These MANs operate autonomously (as do indeed the site networks) and will usually also have additional commercial customers connecting to other core providers. Where site networks share the same MAN (e.g. A and B), local routing can be arranged and the EF provision is via a single provider. So, the model for the NRSE operation described above, where the NRSEs simply micro-manage their respective ends of the EF service, will hold. However, when the EF service is required between sites that are on separate MANs, then there are several problems (e.g. between A and C).

Firstly, let us say that the MAN connecting A and B has provisioned 10 Mb s^{-1} EF services to each of A and B. In principle, it should then be possible to get 10Mb/s EF services from A to B. Now let us say that site C also has a 10 Mb s^{-1} EF service provided by its MAN. Using the NRSE model naïvely, we might expect to get 10 Mb s^{-1} EF between A and C. In fact, it may be that the connection from A’s MAN to the core has only a 5 Mb s^{-1} EF service and the MAN connecting C has only 2 Mb s^{-1} EF provisioned to the core. In this case, it is not sufficient simply to use the NRSE model described above and rely on the edge systems alone to make the reservation. There has to be some mechanism of discovering the EF capability along the entire path from A to C. In fact, we need to find the EF provisioning bottleneck. This problem is compounded when you consider international connectivity, where one national core provider would have peering arrangements with another national core provider.

GARA avoids this problem by insisting the reservations be made in each domain. However, we wish to avoid this for the reasons of poor scaling and difficulties in management discussed earlier. It may be sufficient to have resource management entities in each domain that maintain information about the EF *resource pool* on ingress/egress links as a whole. That is, when a reservation is made, there is *no need to maintain per-flow state*, just to record how much of the available pool is currently in use and how much is available. When a reservation is requested, the NRSE must use its resource discovery mechanisms to query the resource pools and discover if the reservation is possible or not.

5.2 ADVANCED RESERVATIONS

There are still problems here with *advanced reservations* – reservations made some time before they are actually needed. At the time the service request is made, there is no way of knowing how many other advance reservations may be competing for the same bottleneck resource pool. We are currently investigating the use of resource pools via a combination of administrative controls and dynamic signalling. This will try to address the same problem as the GARA inter-Bandwidth-Broker communication (Sander et al., 2001), but will try to prevent signalling state being stored within the core as much as possible, with signalling traffic for reservation setup originating from the edge sites (local and remote) only and not propagating further than is required in order to update the appropriate resource pool.

5.3 A PROPOSAL FOR AN INTERIM SOLUTION

At the time of writing, even without these more complex inter-domain issues being considered, there would be some increase in utility to Grid users simply by employing resource reservation mechanisms (GARA or the NRSE) at the edge sites. Let us consider that, for the relatively small number of "heavy duty" (high capacity and real-time) Grid users on the research networks, we can assume that the core is over-provisioned and that the bottlenecks are at the links that connect the site networks to their service providers. So, managing only these bottleneck links would allow some increase in the QoS control that is offered to traffic. Such a system is easy to implement, requiring only the end-sites to deploy the correct software. In the UKERNA scenario discussed above (Figure 5), the situation is helped further in that the core network does indeed recognize the EF codepoint and provides correct packet handling as well as being over-provisioned. However, we expect that within the next 12–24 months, many of these heavy duty users (certainly the researchers at our respective institutions) will require such network QoS control mechanisms.

5.4 DIFFSERV CODEPOINT VALUES

Other issues for DIFFSERV provisioning include use of DIFFSERV codepoint (DSCP) values across domains. Whilst some DSCP values are agreed (e.g. EF and LBE), many people use other values in the DSCP field in the IP header. This may be proprietary usage or usage of a historic Type of Service (ToS) mechanism for IP. This causes problems at administrative boundaries as the DCSP values have to be mapped, i.e. the packet is re-marked. This may be done unilaterally by the provider receiving the packets and so the end-to-end service is disrupted if the service semantics are not maintained. This remains an issue for further study.

6 Summary

High performance distributed applications may need to use network QoS control mechanisms to guarantee that data are available at remote systems when required for processing. In order to do this, mechanisms are required within the network to provide QoS services, but also we need application-level mechanisms in order to allow applications to make QoS service requests as well as manage and monitor network usage during the course of the operation of the application. For example, a distributed application may have made reservations of CPU and disk space at remote sites, and so wishes to ensure that the network capacity is available to provide the data for the use of these resources at the appropriate times.

The GARA system is widely used and offers some vital core functionally for providing network resource reservation (as well as reservation of other resources such as CPU and disc). However, we see that it has some limitations with respect to managing reservations especially in a multi-domain scenario.

We have given some details of work-in-progress within the GRS project, which seeks to provide a network QoS reservation system that attempts to overcome some of the drawbacks of GARA as well as allowing decentralized state for the reservations, configurable notifications to applications regarding violations to QoS and administrator intervention, localized management mechanisms and flexible scheduling of jobs to deadlines for non-real-time service-requests (e.g. file transfers).

Multi-domain issues, especially dealing with information about service provisioning and usage along a multi-domain network path, remain a major issue, which we are currently considering in our work.

AUTHOR BIOGRAPHIES

Dr Saleem N. Bhatti is Lecturer in the Department of Computer Science and is the coordinator of the UCL

NETSYS Group (http://www.netsys.ucl.ac.uk/). He is involved in several Grid/e-Science projects at UCL, specifically in the area of networked systems. His work has covered a wide range of networked systems topics, including multi-service networking, tele-working, multicast, network and systems management, network protocol design, network and application security, IPv6 and QoS adaptability support for Internet applications and adaptive systems.

Dr Søren-Aksel Sørensen is a Senior Lecturer in the Department of Computer Science at UCL. He works in the fields of parallel, distributed, large-scale, modelling and simulation. His current interests relate to autonomous, migrating processes in large-scale Grid and cluster environments. He is Co-I on the UK *eProtein* project (official title: A Distributed Pipeline for Structure-based Proteome Annotation using Grid Technology), a BBSRC flagship project with partners from UCL, Imperial College and EBI. Prior activities have included development of commercial LAN products and CORBA-based service networks.

Professor Peter Clarke is Professor of Physics at UCL and is involved in several Grid/e-Science project and initiatives. He is also Chair of the PPNCG (Particle Physics Network Coordinating Group) and Deputy Chair of the UK Grid Network Team (GNT). He is a member of the Global Grid Forum Governing Body (the Grid Forum Steering Group) and Director of the Data Area for the GGF. Prof. Clarke is also WP7 Deputy Chair (Networking) in the EU-DataGRID project and involved with networking activities in the DataTAG project.

Professor Jon Crowcroft is Professor of Communication Systems at the Cambridge Computer Laboratory and has been involved in many collaborative projects in networked and distributed systems. He is a Fellow of the Royal Academy of Engineering, a Fellow of the IEE and a Fellow of the ACM. He is a member of the UK Grid Network Team (GNT), the Technical Advisory Group (TAG) and is also on the on the UK EPSRC Technical Opportunities Panel (TOP). He has served on the programme committees of various conferences and workshops including ACM SIGCOMM, IWQoS, QoFIS, NetGames and INFOCOM; on the Internet Architecture Board (IAB); and is currently involved with the Grid High Performance Networking Research Group of the GGF.

NOTES

[1] Unfortunately, the literature contains many definitions of what constitutes a *flow*. We have chosen a definition here that is suitable for this discussion, but is also applicable to the IETF DIFFSERV QoS architecture discussed later.

[2] It is assumed that routes are symmetrical and relatively stable, but this is not always true in the wide area (Paxon 1997).

REFERENCES

AccessGrid. 2002. http://www-fp.mcs.anl.gov/fl/accessgrid/.

AQUILA. 1999. http://www.ist-aquila.org/.

Bhatti, S. N. 1999. IP and Integrated Services. In R. Osso, ed., *Handbook of Communications Technologies: The Next Decade*, pp. 217–238, CRC Press, Boca Raton, FL.

Bhatti, S. N., and Crowcroft, J. 2002. QoS Sensitive Flows: Issues in IP Packet Handling. *IEEE Internet Computing*, 4(4):48–57.

Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. 1998. An Architecture for Differentiated Services, RFC2475, IETF DIFFSERV WG.

Braden, R., Zhang, L., Berson, S., Herzog, S., and Jamin, S. 1997. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, RFC2205, IETF INTSERV WG.

Clark, D. D., Shenker, S., and Zhang, L. 1992. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *Proceedings of ACM SIGCOMM'92*, pp14–26.

DataTAG. 2002. http://datatag.web.cern.ch/datatag/.

Davie, B., Charny, A., Bennett, J. C. R, Benson, K., Le Boudec, J-V., Courtney, W., Davery, S., Firoiu, V., and Staliadis, D. 2002. An Expedited Forwarding PHB (Per-Hop Behavior), RFC3246, IETF DIFFSERV WG.

EU-DataGrid. 2002. http://eu-datagrid.web.cern.ch/eu-datagrid/.

GARA (General-purpose Architecture for Reservation and Allocation). 2002. http://www-fp.mcs.anl.gov/qos/.

Globus. 2002. http://www.globus.org/.

GridProbe. 2002. http://umbriel.dcs.gla.ac.uk/NeSC/action/projects/ project_action.cfm?title=63.

Grid Resource Scheduling (GRS). 2002. http://www.cs.ucl.ac.uk/research/grs/.

Guttman, E., Perkins, C., Veizades, J., and Day, M. 1999. Service Location Protocol, Version 2.RFC2608, IETF.

Heinanen, J., Baker, F., Weiss, W., and Wroclawski, J. 1999. An Assured Forwarding PHB Group, RFC2597, IETF DIFFSERV WG.

Mankin, A., Baker, F., Braden, B., Bradner, S., O'Dell, M., Romanow, A., Weinrib, A., and Zhang, L. 1997. Resource ReSerVation Protocol (RSVP) – Version 1 Applicability Statement Some Guidelines on Deployment, RFC2208, IETF INTSERV WG.

Managed Bandwidth – Next Generation (MB-NG). 2002. http://www.mb-ng.net/.

MESCAL. 2002. http://www.mescal.org/.

NetLogger. 2002. http://www-didc.lbl.gov/NetLogger/.

Paxson, V. 1997. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615.

Qbone. 2002. http://qbone.internet2.edu/.

Rose, M. 2001. The Blocks Extensible Exchange Protocol Core, RFC3080, IETF.

Sander, V., Foster, I., Roy, A., and Walker, L. 2000. A Differentiated Services Implementation for High Performance TCP Flows. In *Proceedings of IWQoS'2000.*

Sander, V., Adamson, W. A., Foster, I., and Roy, A. 2001. End-to-End Provision of Policy Information for Network QoS. http://qbone.internet2.edu/bb/BB-to-BB_Security_Model.pdf.

Shalunov, S., and Teitelbaum, B. 2001. QBone Scavenger Service (QBSS) Definition, Internet2 Technical Report. http://qos.internet2.edu/wg/wg-documents/qbss-definition.txt.

Shenker, S., Partridge, C., and Guerin, R. 1997. Specification of Guaranteed Quality of Service, RFC2212, IETF INTSERV WG.

Shenker, S., and Wroclawski J. 1997a. General Characterization Parameters for Integrated Service Network Elements, RFC2215, IETF INTSERV WG.

Shenker, S., and Wroclawski, J. 1997b, Network Element Service Specification Template, RFC2216, IETF INTSERV WG.

TEQUILA. 2002. http://www.ist-tequila.org/.

Task Force – Next Generation Networking (TF-NGN). 2002. http://www.terena.nl/tech/task-forces/tf-ngn/.

UDDI. 2002. http://www.uddi.org/.

Wolf, L., Gridwodz, C., and Steinmetz, R. 1997. Multimedia Communication. *Proceedings of the IEEE*, 85(12):1915–1933.

Wroclawski, J. 1997a. Specification of the Controlled-Load Network Element Service, RFC2211, IETF INTSERV WG.

Wroclawski, J. 1997b. The Use of RSVP with IETF Integrated Services, RFC2210, IETF INTSERV WG.