



First Internet2 Joint Applications/ Engineering **QoS** Workshop

Enabling Advanced Applications Through

May 21–22, 1998
Santa Clara, CA

1998

**First Internet2 Joint Applications/
Engineering QoS Workshop**



ACKNOWLEDGEMENTS

UCAID is grateful to Bay Networks for its generous hospitality. Bay Networks provided conference facilities, logistical support, and refreshments for the workshop.

Additional thanks to the National Science Foundation, whose financial support (NSF Grant #ANI-9807142) has made both the workshop and this report possible.

ORGANIZERS

The First Joint Applications/Engineering Quality of Service Workshop was organized by the individuals listed below and by the Internet2 QoS Working Group. The workshop was hosted by the University Corporation for Advanced Internet Development (UCAID), which provides the formal organization to support Internet2 members as they develop the broadband applications, network engineering, and network management tools for next generation Internet use in research and education.

Douglas Van Houweling
President and CEO, UCAID

Van Houweling has been President and CEO since October 1997 when UCAID was formed. Prior to that, he served as Dean of Academic Outreach and Vice Provost for Information Technology at the University of Michigan. Dr. Van Houweling received his undergraduate degree from Iowa State University and his Ph.D. in government from Indiana University.

Guy T. Almes
Chief Engineer, Internet2

Almes has served as Chief Engineer of Internet2 since January 1997. He also continues to serve as the Vice President of Network Development at Advanced Network & Services, where he has been since 1991. He holds a BA and MEE from Rice University and a Ph.D. in computer science from Carnegie-Mellon University.

Ted Hanss
Director of Applications Development, Internet2

Hanss is Director of Applications Development for Internet2, on loan from the University of Michigan, where he served as Director of the Center for Information Technology Integration. He is an adjunct lecturer in the Computer and Information Systems department of the University of Michigan School of Business Administration. He has a BS from Boston College and an MBA from the University of Michigan.

Benjamin R. Teitelbaum (report editor)
Internet Engineer, Internet2

Teitelbaum is an Internet Engineer with Internet2 and Advanced Network & Services. He chairs the Internet2 QoS Working Group, and is also involved with engineering the Surveyor IP Provider Metrics (IPPM) measurement infrastructure. He holds a BS in mathematics from the Massachusetts Institute of Technology and an MS in computer science from the University of Wisconsin–Madison.

TABLE OF CONTENTS

<i>Preface</i>	3
Douglas Van Houweling	
<i>Quality of Service for Internet2</i>	5
Benjamin Teitelbaum, John Sikora, and Ted Hanss	
<i>Tele-Immersion: The Ultimate QoS-Critical Application</i>	17
Jaron Lanier	
<i>The Role of QoS in Wide Area Data Mining</i>	19
Robert Grossman	
<i>A Discrete and Dynamic Approach to Application / Operating System QoS Resource Management</i>	22
Scott Brandt, Gary Nutt, and Ken Klingenstein	
<i>Differentiated Services for the Internet</i>	26
Van Jacobson	
<i>Evolution of End-to-End QoS: A Design Philosophy</i>	32
John Wroclawski	
<i>A Scalable Resource Management Framework for Differentiated Services</i>	37
Lixia Zhang	
<i>NYSERNET QoS and Policy Routing</i>	40
Jim Grisham	
<i>Minimum-Complexity QoS for the Enterprise</i>	42
Terry Gray	
<i>Cross-Sectional Breakout Sessions</i>	45
<i>Specialist Breakout Sessions</i>	50
<i>Conclusions and Next Steps</i>	54
<i>Appendix One: Using Diffserv Premium Service to Provide Internet2 QoS</i>	58
Kathleen M. Nichols	
<i>Appendix Two: QoS Workshop Attendees</i>	63
<i>Acronyms</i>	<i>inside back cover</i>

PREFACE

The Internet2 project exists to help realize the promise of a new cycle of innovation in networking. The universities that initially set out on this endeavor understood that fundamental advances in network technologies would be needed to enable the advanced network applications that universities will require to fulfill their research and teaching missions in the 21st century. Today the Internet2 project represents an extraordinary partnership of over 130 universities, 40 corporations and 30 other organizations. These participants understand that in much the same way as the Internet has changed the world, Internet2 has the potential to transform again the way we work, learn, and communicate.

From the beginning, quality of service (QoS) has been essential to the Internet2 vision. The best-effort Internet is inherently unable to guarantee with any reliability the performance necessary to run even the more moderately ambitious advanced networked applications envisioned today. Moreover, as new technologies and applications demand ever more from the network and as users begin to rely on these applications, QoS will become increasingly mission critical. Successful deployment of production-quality QoS will allow doctors to interact in real time with distant patients, astronomers to move large amounts of data quickly from coast to coast, biologists to make use of remote electron microscopes, and instructors to offer distance learning courses and university library resources to students anywhere.

Quality of service will bring fundamental shifts in how the Internet is used and how we regard the service it provides. Users will be able to request and reserve the network resources that their applications require. Consequently, when the requested resources are not available along the path where they are needed, the network will have to gener-

ate a busy signal. In contrast, today's best-effort-only Internet always admits new traffic, with the expectation that in the face of congestion, applications will slow their transmission rates and adapt gracefully to the reduced performance. Another consequence of this is that, for mission-critical applications like distance learning, users will need to be able to schedule their network usage in advance to avoid the possibility of a busy signal.

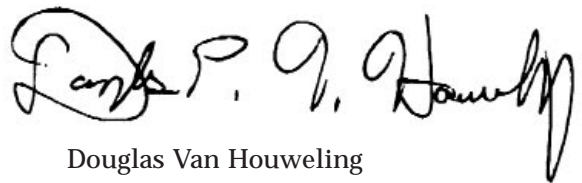
The resulting changes will shift the expectations of the Internet industry. Application developers will be freed to use a mixture of adaptive techniques and those that require absolute performance levels from the network, but will be tasked with characterizing more precisely the network requirements of their applications. Internet service providers will need to provision their networks to support multiple classes of traffic, providing QoS assurances to some flows, while allowing efficient sharing of bandwidth to provide adequate service to best-effort traffic. Finally, all of us will have to grapple with the problem of finding appropriate policies and business models to support these new network services.

The technical challenges are also significant. Although there has been much research into the signaling mechanisms and queuing disciplines needed to support QoS, and lately into highly scalable techniques to reduce the forwarding complexity of QoS-enabled routers, there are no QoS technologies proven to function in a large-scale internet-work. For a QoS technology to serve the needs of Internet2, it must scale to the high speeds and large numbers of flows found in the backbone. Further, it must also allow individual Internet2 network clouds to make independent engineering, provisioning, and implementation decisions while still achieving a well-defined end-to-end QoS for wide-area flows.

Nine months ago, UCAID assigned the Internet2 QoS Working Group the task of recommending a QoS strategy to meet the needs of Internet2. From a series of workshops that addressed the advanced applications and engineering QoS requirements of the Internet2 community, the working group synthesized a requirements document. Additionally, considering these requirements and the state of the technology and the standards process, the working group grappled to formulate a technical recommendation in this area. This first Internet2 Joint Applications/Engineering QoS Workshop was organized to present their recommendations and to solicit input from the broader Internet2 and networking research communities. The discussions that took place over the course of the two days significantly

helped to clarify and refine the direction for Internet2 QoS and reinforced our understanding of the difficult but rewarding road that lies ahead.

I invite you to read this report and encourage you to become engaged in the process of developing a production Internet2 QoS infrastructure that will enable the advanced applications of the future. This is your process. We look to you to criticize, correct and complete the ideas presented here. As with the development of the current Internet, a partnership among academia, industry and government is leading the cycle of innovation required to take today's Internet to a new plateau of uses. The work being undertaken in the area of QoS is an crucial part of this cycle.

A handwritten signature in black ink, appearing to read "Douglas P. Van Houweling". The signature is fluid and cursive, with a large initial 'D' and 'V'.

Douglas Van Houweling
President and CEO, UCAID

QUALITY OF SERVICE FOR INTERNET2

Benjamin Teitelbaum, *Internet2*
John Sikora, *AT&T Labs*
Ted Hanss, *Internet2*

ABSTRACT

The primary goal of Internet2 is to support the research and education missions of universities through the development of new advanced networked applications. Unfortunately, many of these applications simply are not feasible in today's Internet because the current best-effort delivery model is unable to provide the minimum end-to-end performance assurances that they require. To enable these applications, Internet2 networks must provide quality of service (QoS) functionality that allows applications to reserve key network resources without undue impact on best-effort traffic.

Through dialogue over the past year, the Internet2 applications and engineering communities have identified a set of requirements for Internet2 QoS based on application needs and engineering constraints. In turn, the Internet2 QoS Working Group has studied these requirements and has recommended that Internet2 adopt the evolving differentiated services (DiffServ) approach to QoS.

The DiffServ framework has emerged in the last year out of a desire for simple and scalable forms of QoS that provide meaningful end-to-end services across multiple, separately administered network clouds without necessitating complex interprovider business arrangements. These objectives are remarkably similar to the requirements and design goals identified for Internet2 QoS — in particular, the emphasis on simplicity, scalability, interoperability, and administrability.

Although the current push for DiffServ is largely being driven by the immediate needs of commercial network service providers

who wish to deploy class of service (CoS) functionality in their networks, the framework admits the implementation of a range of services that appear to meet the most pressing needs of advanced applications. Indeed, it is fortunate that most of the new functional components needed to implement CoS are identical with those that are required to implement the stronger services required by Internet2. The Internet2 QoS Working Group recommends that Internet2 emphasize experimental deployment of those particular services envisioned under the DiffServ framework that provide absolute transmission assurances — that is, assurances that are in no way relative to the current load on the network.

However wonderful the fit between DiffServ's design and Internet2's needs, and however sound the engineering principles behind the DiffServ framework, many hard technical and social problems remain that can only be solved through experience and iterative design refinement in a testbed environment. Therefore, Internet2 is launching a DiffServ testbed that will focus on interoperability among participating Internet2 networks and on collective problem solving among those experimenting with new services. Beyond providing a cooperative environment for resolving engineering issues, the testbed effort can serve as the focal point for continuing dialogue on the complex engineering, social, and policy changes that are required before production QoS services and regular use of advanced applications can become commonplace. The proposed Internet2 DiffServ testbed is described further in the final section of this report.

1. Problem Statement

In today's Internet, each network element along an IP packet's path makes nothing more than a good-faith effort to forward the

packet toward its destination. If a router's queue is overloaded, packets are dropped with little or no distinction between low-priority traffic and urgent traffic. This is known as best-effort service.

To function correctly, many advanced applications require guarantees of minimum bandwidth and maximum packet delay (latency) that best-effort networks are unable to make. For example, remote collaborative tools usually have requirements based on hard thresholds of human perceptual sensitivity that translate into very specific bandwidth and latency needs. Failure to meet these needs results in the application being unusable. Moreover, some important advanced applications, such as remote surgery or remote instrument control, have requirements that, if not met, result in unacceptable real-world consequences — for example, loss of life or damage to expensive remote equipment.

Internet2 strives for an integrated approach in which QoS-enabled streams coexist with normal best-effort traffic, sharing network resources in such a way that the benefits of circuit-switched networks (performance guarantees) and those of best-effort networks (low cost due to resource sharing) may simultaneously be achieved.

2. QoS Requirements for Internet2

With the help of input from a series of workshops held over the last year, the Internet2 QoS Working Group has identified several requirements that any approach to Internet2 QoS must meet. To be successful, any Internet2 QoS solution must:

- Enable advanced applications
- Be scalable
- Be administrable
- Be measurable
- Admit multiple, interoperable implementations of both individual pieces of equipment and clouds
- Have support from operating systems and middleware

- Be incrementally deployable starting in 1998

The following subsections outline each requirement in more detail. Readers familiar with these requirements may wish to skip to Section 3, “Differentiated Services: An Evolving QoS Solution.”

2.1 Enabling Advanced Applications

When asked what QoS they need from the network, application developers tend to smile and say something like, “I need as much bandwidth as you can give me with as little latency, jitter, and loss as possible.” This is not a completely tongue-in-cheek answer. The reality of developing networked applications for a best-effort Internet has compelled many application developers to become expert at writing adaptive applications that function correctly over a wide range of throughputs. Developers are consequently not accustomed to thinking about the baseline requirements of their applications.

To support the development of advanced applications, the primary Internet transport protocols — most importantly TCP — have been carefully designed and tuned over the past 25 years to degrade gracefully in the face of congestion and to hunt eagerly for bandwidth in its absence. In today's Internet, new connections are essentially never denied and every connection's performance degrades as the network load increases. There are no busy signals in a best-effort network.

In contrast, the user of a QoS-enabled network will perceive a service model more like that of the telephone system. There is first some sort of call setup, in which the user attempts to initiate a connection and reserve the required resources. Then, assuming that the call goes through, the user is assured a clear channel on which to communicate (barring equipment failures). Alternatively, at call setup time, the user may receive a busy signal and be denied the privilege of connecting at the desired level of QoS. Adapting to this new service

paradigm will require significant changes to the expectations of both users and developers.

Dialogue between the Internet2 applications and engineering communities over the past year has attempted to push beyond the initial stalemate to identify the specific network requirements of advanced applications. Together with an informal survey of developers, these discussions revealed a rough consensus on the kinds of transmission parameters that matter, but very few hard numbers and very little understanding of the nature of the network performance assurances required.

A fundamental dimension of any application's QoS requirements is the set of transmission parameters about which assurances are needed. The transmission parameters most commonly mentioned as requiring assurances are bandwidth and latency. For the next-generation applications currently envisioned, these requirements amount to bandwidth needs of some small number of megabits per second (<10Mbps) and maximum latencies of 30 to 500 msec. Some applications also require bounds on jitter (variability in packet delay), although most can mask jitter through receiver-based playback buffers. More esoteric transmission parameters like privacy and robustness were occasionally mentioned, but are currently considered out of scope by the Internet2 QoS Working Group.

An important class of applications requires that QoS assurances apply not only to unicast flows, but to multicast traffic as well. Because the engineering complexities of multicast technologies in general are notorious, the working group has decided to make the extension of QoS to multicast flows a long-term goal, but not a strict requirement. Early iterations of any QoS approach adopted will focus on providing reliable and robust QoS services to unicast flows. Attempts will be made to avoid taking steps that would preclude later extension of the adopted QoS technologies to the multicast case.

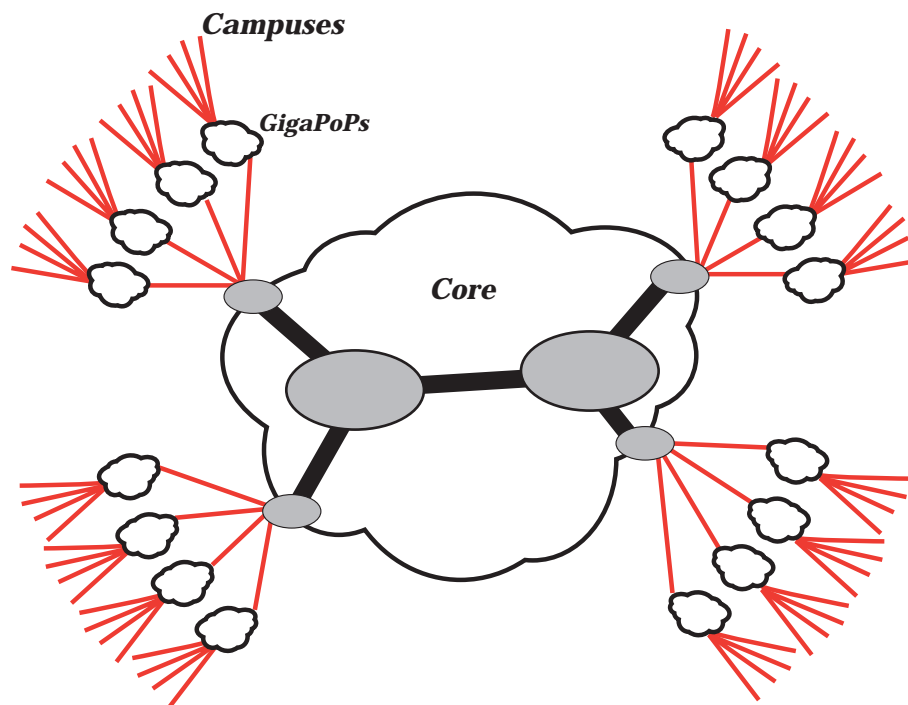
The other key component of any QoS assurance is the nature of the assurance itself. There is a great deal of confusion among developers (and among the networking community at large) about the kinds of assurances that are required. Closer scrutiny reveals a broad range of needs.

The most demanding applications have specific, absolute transmission requirements grounded in hard thresholds of human perceptual sensitivity, and are highly intolerant of network performance below these levels. Other applications have specific requirements, but are tolerant of occasional drops in performance or require only that the performance averaged over a certain time period be maintained. Finally, there is a desire for network functionality that can simply treat the traffic from certain applications or users "better" than other traffic. In the last case, the assurance made is relative to the network load imposed by other traffic, while in the first two, the required assurances are absolute.

Whereas attention is most often paid to the service "floor" provided to the application by QoS assurances, many notions of QoS also have consequences for the "ceiling" on the available resources. Over the past 25 years, a great deal of expertise has been developed around the engineering of so-called "adaptive applications." These applications are able to adjust gracefully and usefully to the varying availability of network resources. There is significant concern among some segments of the Internet2 community that any QoS approach allow for the continued use of adaptive techniques.

Finally, there is a temporal aspect to any QoS assurance — namely, when does the assurance begin and for how long does it last? An important class of applications, of which distance-learning applications and remote instrument control are superb examples, requires that users be able to schedule QoS-enabled sessions in advance. In the distance learning case, without such advance reservations, students might receive busy signals

FIGURE 1.
Scalability is a serious concern in the network core, where thousands of flows may pass through each router.



from the network at the time that a distance learning course is scheduled to meet.

2.2 Scalability

Certainly the greatest engineering challenge to providing end-to-end, per-flow QoS is scalability. QoS approaches that require substantial per-flow state or computational overhead in the forwarding engines simply will not scale as the number of QoS-enabled flows through the network grows large. Scalability is a particular concern in core routers, where the fan-in from the network edges can easily burden routers with forwarding thousands of flows at very high packet-per-second rates. (See Figure 1.)

Although the focus of the Internet2 QoS Working Group has been on finding highly scalable approaches to QoS, there may well continue to be small sets of high-end users who require very demanding or long-lived flows that are best supported by stateful QoS approaches like RSVP. However, it is the goal of Internet2 to move beyond isolated demonstrations of advanced applications and toward highly scalable architectures that

will enable the emergence of thriving user communities for these applications.

2.3 Administrability

As with any scarce resource, there will need to be mechanisms to allocate and account for QoS. These mechanisms must operate efficiently, providing end users ready access to QoS features without unduly burdening network planning and operations staff with additional management complexity. Furthermore, these administrative mechanisms must support a flexible set of policies and deter theft of QoS services.

2.4 Measurability

Since institutions (and eventually users) will pay for the QoS they receive, there must be a means for end users to measure and audit their network performance. The requirement of measurability implies not only a need for measurement tools, but also a need for well-understood network performance metrics. Network providers may need additional measurement tools to assist in the provisioning and debugging of QoS services, and possibly also to support auto-

mated measurement-based admission control mechanisms.

2.5 Admitting Multiple, Interoperable Implementations

Any Internet2 QoS approach must admit multiple, interoperable implementations of key functional components (i.e. packet forwarders, classifiers, and admission control elements), as well multiple, interoperable implementations of QoS services across individual network clouds.

2.5.1 Interoperability of Equipment

Any QoS approach adopted by Internet2 should be implementable by one or (preferably) many vendor partners. In a heterogeneous internetwork as large and diverse as Internet2, multi-vendor interoperability is absolutely essential. Fortunately, most equipment vendors and network service providers understand that standards-based interoperability of QoS technologies is key both to the success of end-to-end services and to their own financial success.

To ensure the wider success of QoS and to avoid technical isolation, Internet2 must

pursue a QoS strategy that is closely aligned with the directions of Internet standards organizations such as the Internet Engineering Task Force (IETF). Although it is very likely that initial Internet2 QoS deployment will begin before standards work is complete, the Internet2 QoS experience should provide invaluable early feedback to the ongoing standards work, including feedback on the interoperability or non-interoperability of alternative implementation techniques.

2.5.2 Interoperability of Network Clouds

QoS-enabled flows and call setup signaling should be treated in a standard and well-understood way at cloud-to-cloud boundaries, but clouds must be allowed to implement QoS internally in potentially very different ways. Internal implementations of QoS may vary depending on a cloud's underlying technologies, internal policies, and provisioning decisions.

Consider the path between the two hosts in Figure 2. This path flows from the source host into the campus network, then on to the regional gigaPoP, a national interconnect

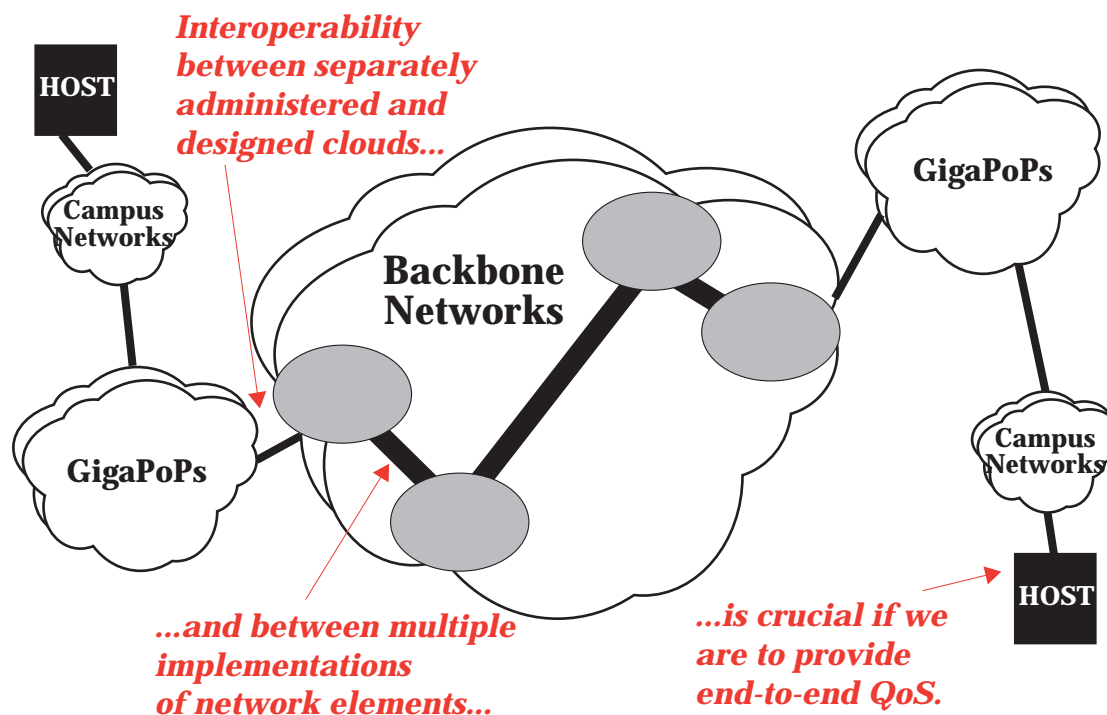


FIGURE 2. Interoperability. Separately administered clouds must be able to provide QoS services that concatenate to achieve end-to-end QoS.

(e.g. the vBNS), another gigaPoP, another campus network, and finally the destination host. This is the typical situation in the Internet2 environment. Since campuses, gigaPoPs, and interconnects are each under separate administrative control, there is a need to standardize a notion of QoS across a cloud, so that a sequence of clouds may provide meaningful end-to-end QoS services.

2.6 Support from Operating Systems and Middleware

Eventually, hosts should be able to initiate QoS requests for their flows. In the short term, however, QoS services may need to be statically configured on a per-wall-jack, per-port-number, or per-protocol basis. Hosts must also be able to identify themselves or their users appropriately to the network for purposes of authentication, authorization, and accounting. Additionally, to provide true end-to-end QoS, host operating systems will need to support QoS-enabled flows. This kind of real-time functionality is not present in the operating systems of most current Internet2 hosts, where packets may experience bottlenecks within the network stack, the memory system, or even the process scheduler.

2.7 Incrementally Deployable, Starting in 1998

It would be unrealistic to adopt a QoS approach that required a “flag day” for changing all network elements. A valuable service can arise from even a partial deployment — for example, only at the most congested points or only along certain paths. This could have immediate benefits for a number of important existing applications that require QoS.

Additionally, since no off-the-shelf QoS solution exists that will meet the pressing advanced applications requirements of our members, there is a need for immediate experimentation and iterative refinement of the most promising QoS technologies and services. At the time of this workshop, these appear to be DiffServ and a set of proposed services that offer non-relative (absolute) assurances to applications.

3. The DiffServ Architectural Framework

The goal of the evolving IETF DiffServ framework is to provide a means of offering a spectrum of services in the Internet without having to maintain per-flow state in every router.

By bringing together a multitude of QoS-enabled flows into a small number of aggregates that are given a small number of differentiated treatments within the network, DiffServ eliminates the need to recognize and store information about each individual flow in core routers, and thus achieves scalability. The essential trick is to combine a small number of simple aggregate packet treatments with a larger number of per-flow policing policies to provide a broad and flexible range of services.

Each DiffServ flow is policed and marked at the first trusted downstream router. This is done according to a contracted service profile, usually a token bucket filter. When viewed from the perspective of a network administrator, the first trusted downstream router is a leaf router at the periphery of the trusted network. Downstream from the nearest leaf router, a DiffServ flow is mingled with similar DiffServ traffic into an aggregate. All subsequent forwarding and policing is performed on aggregates.

Another important benefit of handling traffic aggregates is the simplification of the business relationships required to construct end-to-end services. In the DiffServ model, individual network clouds contract with neighboring clouds to provide differentiated service contracts for different traffic aggregates. Like per-flow contracts, aggregate contracts are characterized by profiles (again, often based on token bucket filters). By strictly enforcing the aggregate traffic contracts between clouds and by ensuring that new calls that would exceed aggregate traffic capacity are not admitted, the DiffServ architecture provides well-defined end-to-end service over chains of separately administered clouds. Furthermore, since each

aggregate's contracts exist only at the boundaries between clouds, the result is a set of simple bilateral service level agreements that mimics current interprovider exchange agreements.

In addition to DiffServ-enabled packet forwarders, required components include classifiers, policers, markers and a new kind of network component known as a bandwidth broker. These are shown in the figure below.

Per-flow policing and marking is performed by the first trusted leaf router downstream from the sending host. When a local admission control decision has been made by the sender's cloud, the leaf router is configured with the contracted per-flow service profile. Downstream from the first leaf router, all traffic is handled as aggregates.

On cloud ingress, incoming traffic is classified by the per-hop behavior (PHB) bits into aggregates, which are policed according to the aggregate profiles in place. Depending on the particular DiffServ service model in question, out-of-profile packets are either dropped at the edge or marked with a differ-

ent PHB. As in-profile traffic traverses a cloud, it may experience induced burstiness caused by queuing effects or increased aggregation. Consequently, clouds may need to shape on egress to prevent otherwise conforming traffic from being unfairly policed at the next downstream cloud.

Finally, to make appropriate internal and external admission control decisions and to configure leaf and edge device policers correctly, each cloud is outfitted with a bandwidth broker (BB). When a sender signals its local bandwidth broker to initiate a connection, the user is authenticated and the user's request is submitted to a local policy-based admission control decision making mechanism. On behalf of the sender, the BB then initiates an end-to-end call setup along the chain of bandwidth brokers representing the clouds to be traversed by the flow. The bandwidth broker abstraction is critically important because it allows separately administered network clouds (possibly implemented with very different underlying technologies and policies) to manage their network resources as they see fit.

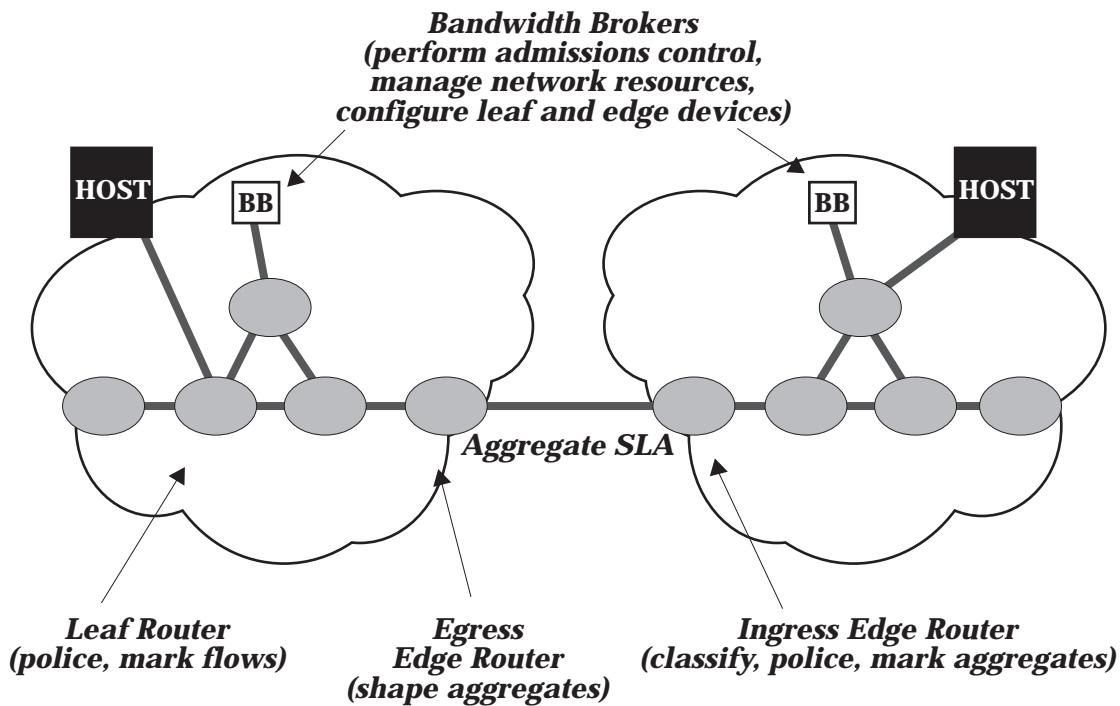


FIGURE 3.
The Differentiated Services Architectural Framework.

Although the DiffServ architectural framework described here is fully consistent with the framework described in Jacobson's presentation at this workshop and in several recent IETF Internet Drafts, it is not currently under standards consideration in the IETF. The current scope of the IETF DiffServ working group charter is to define several initial PHBs and code points in the IPv4 and IPv6 packet headers to represent them.

4. DiffServ Evaluated

The objectives guiding the evolution of DiffServ are remarkably similar to the requirements and design goals for Internet2 QoS. From a network engineering perspective, the Internet2 QoS Working Group likes DiffServ's emphasis on simplicity, scalability, interoperability, and administrability. Even more compelling is that the framework supports a range of implementable services that appears to span the space of advanced application requirements. What follows is a point-by-point analysis of how the DiffServ framework meets the quality of service requirements of Internet2.

4.1 Enables Advanced Applications

In judging how well a QoS approach meets the needs of advanced applications, the primary criterion must certainly be the nature of the end-to-end transmission guarantees that can be offered by the approach. We conjecture that a set of four proposed DiffServ services will suffice for Internet2 application requirements:

- **Premium** — emulates a leased line. Peak bandwidth guarantee with very low queuing delay, loss, and jitter. Appropriate for intolerant applications. See Jacobson's extended abstract on page 26.
- **Assured** (or a similar predictive service) — emulates a lightly loaded network. Similar to Controlled Load; appropriate for tolerant and adaptive applications. See Wroclawski's extended abstract on page 32.
- **Class of Service (CoS)** — relative, precedence-based service classes.

Better best-effort service to meet coarse user and institutional priorities.

- **Default** — best-effort service.

Premium offers a peak-limited service with extremely low loss and jitter and minimal queuing delay. It requires a PHB equivalent to strict priority queuing, as well as admission control policies that never oversell the Premium capacity of any router. Per-flow and aggregate Premium policers drop all out-of-profile packets. Premium service is appropriate for the most intolerant applications and may appeal to an even broader community of providers, developers, and users due to its easily understood service contract. (For a more in-depth analysis of Premium's applicability to internet2's needs, see Nichols' appendix on page 58.)

Assured service is an example of the predictive services that inhabit the middle portion of the service spectrum. With Assured, users contract for a specific service profile and are guaranteed that in-profile traffic will experience service consistent with a "lightly loaded network" even in the presence of congestion. In-profile packets are given a PHB that indicates that they should be dropped last if packets need to be dropped. Out-of-profile packets are re-marked by policers to have the default, best-effort PHB. Predictive services like Assured are likely to be much less expensive to provide than Premium and may well meet the needs of applications that don't require the strength of the Premium assurance. Such services may be appropriate for applications that can tolerate some packet loss and only require transmission assurances that are averaged over long periods of time. Additionally, the Assured service model has a soft ceiling that allows for continued use of adaptive application techniques.

Finally, CoS will likely become important to Internet2 member institutions that want coarse differentiation of network traffic to meet institutional priorities or to support applications that are merely in need of better service in the short term. For example, a

university might give email traffic from the bursar's office higher priority than chat traffic from dormitory networks, but lower priority than email between medical researchers. However, as network traffic continues to grow faster than bandwidth can be provisioned, CoS approaches will never meet the stronger requirements of many important advanced applications.

The Internet2 QoS Working Group recommends immediate testbed deployment of the stronger, "absolute" forms of DiffServ — specifically, the Premium and Assured services described above. Additionally, we recommend that Internet2 networks support CoS functionality as it becomes commercially available.

Other application requirements mentioned in Section 2.1 include eventual QoS support for multicast flows and the ability to schedule QoS-enabled sessions in advance. Although neither piece of functionality has been developed for DiffServ, there is nothing in the framework that would appear to preclude either one. Extending the PHB treatments to the multicast case should be straightforward, with most complexity arising in the extension of the bandwidth broker admission control algorithms. Supporting scheduled reservations is again a matter of extending the bandwidth broker admission control algorithms.

4.2 Scalability

Scalability is one of the most serious engineering requirements for QoS. Happily, it is also one of the primary concerns driving the design of the DiffServ architecture.

4.2.1 Scaling to Large Numbers of Flows

Core routers are routinely burdened with forwarding thousands of flows. Consequently, any QoS approach that requires substantial per-flow state or computational overhead in the forwarding engines will not scale well. The DiffServ approach attempts to push per-flow complexity away from the network core and toward the network periphery where both the forwarding speeds and the fan-in of

flows are smaller. The aim is to provide per-flow service assurances by policing flows at leaf routers and servicing only a small number of traffic aggregates in the core. As has been discussed, aggregates are characterized by only a small number of simple per-hop forwarding behaviors (PHBs), which greatly reduces the per-packet forwarding complexity for core routers.

4.2.2 Scaling to High Speeds

In addition to the need to keep PHBs simple so that aggregates may be handled at high speeds, there will be an increasing need for high-speed handling of individual flows. This is especially true in environments where increases in available network bandwidth create a perception that bandwidth is plentiful. Dramatic increases in available bandwidth will inspire and enable new kinds of applications that will increasingly depend on the existence of fat pipes. When these pipes eventually become congested, the new applications will require QoS assurances that scale to very high speeds. The differentiated services approach of pushing all per-flow complexity as close to the end-hosts as possible represents the best hope for scaling per-flow QoS to very high speeds.

4.3 Administrability

In the differentiated services framework, each cloud is free to set its own arbitrarily complex and baroque local policies and administrative procedures as long as the bilateral agreements that it makes with other clouds are honored. Since resource allocation policies are often sensitive, it is appealing that the bandwidth broker admission control abstraction handles external requests without extensive sharing of policy information.

Because internal policies and procedures vary so widely among domains, the administrative flexibility of DiffServ is especially important for clouds that contain end hosts. The advent of production quality QoS will inevitably be accompanied by a need to control access to it according to locally appropriate policies. The negotiation between a host and its local bandwidth broker is the

ideal place to enforce such local policies. Within a domain, the bandwidth broker is the centralized point for performing authentication, authorization, and accounting (AAA) and as such is the point where policies are administered.

Some campuses within Internet2 may have decentralized or hierarchical administrative structures that would be better served by a distributed bandwidth broker architecture. A distributed bandwidth broker architecture would be fully consistent with the DiffServ architectural framework and could also have desirable performance and fault tolerance characteristics. Other campuses may choose to avoid the complexities of highly dynamic AAA by selling statically configured, subscription-based services to users. This approach is explored further in Gray's extended abstract.

4.4 Measurability

DiffServ allows a contracted QoS service profile to be compared with well-defined IP performance metrics. A user with suitable measurement tools will be able to audit the service received against the contracted service. Measurement tools could also be used by network service providers to audit their aggregate traffic contracts.

Additionally, network service providers may rely on measurement tools for network planning, engineering, and debugging. For example, measurements might reveal unexpected congestion points that need to be either relieved or managed carefully by the local bandwidth broker.

A particularly nice consequence of the DiffServ architectural framework is that much useful measurement data falls naturally out of the architecture's packet classifiers and policers.

4.5 Admitting Multiple, Interoperable Implementations

Standards-based interoperability of forwarding equipment and support for end-to-end services have been primary design goals since the

beginning of DiffServ and are among its key strengths. The IETF is currently focused on specifying standards for PHBs and the packet header code points that signify them; this should allow for interoperability at the lowest level. Cloud-to-cloud interoperability will be achieved by global PHB standards and simple bilateral service level agreements that can be concatenated to form end-to-end services.

4.5.1 Interoperability of Equipment

Although much of the IETF DiffServ working group's work to date has been on specifying a few standard PHBs and on finding agreement on how to assign the bits of the DiffServ field while maintaining backward compatibility with routers that use the ToS field, additional work will soon be needed to define standard protocols for

- Hosts to signal their requests to their local bandwidth broker.
- Bandwidth brokers to authenticate users.
- Bandwidth brokers to signal other bandwidth brokers for end-to-end call setup.
- Bandwidth brokers to signal leaf routers with the per-flow policing parameters that should be enforced.

4.5.2 Interoperability of Network Clouds

The need to concatenate the separately administered and engineered services of multiple network clouds into meaningful end-to-end services has been a primary motivator of the DiffServ architecture since its inception. The DiffServ framework meets this need by defining services only at cloud edges and by allowing for flexibility in how services are implemented within each cloud. Bandwidth brokers abstract the differences between how clouds are engineered internally, and provide a common meeting point for admission control negotiation. Within a DiffServ cloud, network engineers have considerable flexibility in choosing among competing technologies, network equipment vendors, and provisioning alternatives. In particular, DiffServ should be implementable either completely at the IP level, or, in the

case of an IP-over-ATM cloud, by mapping DiffServ PHBs to ATM service classes.

The differentiated services approach is particularly well suited to meeting the administrative challenges of extending a QoS service across separately administered clouds. The framework results in simple, bilateral agreements between service providers that are easy to comprehend, enforce, and audit.

Additionally, by hiding the internal provisioning and engineering details behind the bandwidth broker abstraction, network service providers are afforded the privacy and flexibility they need to operate independently and competitively. DiffServ's separation of differentiated forwarding from intradomain resource management and admission control has been compared with the historically successful separation of forwarding (simple) from routing (complex). If the parallel abstraction is equally successful, it should result in increasingly sophisticated support for policies and efficient resource management techniques that occur independently of the underlying differentiated forwarding mechanisms.

4.6 Support from Operating Systems and Middleware

A DiffServ architecture can be deployed initially using today's QoS-blind applications together with simple email requests. It is expected, however, that there will quickly develop a need for applications to be able to signal their QoS requirements dynamically with the support of appropriate APIs and middleware.

RSVP is increasingly available in modern operating systems and could be used as the signaling protocol between hosts and their local bandwidth broker. An architecture for mating RSVP to DiffServ has been outlined in a recent Internet Draft.

To facilitate the development of DiffServ-enabled applications, the Internet2 applications group has proposed that differences among existing APIs and among operating system QoS signaling protocols be abstracted by a new API and appropriate middleware.

4.7 Incrementally Deployable, Starting in 1998

There is a happy coincidence in the concurrent evolution of DiffServ and Internet2 QoS. While the push for differentiated services is being driven by commercial network service providers' immediate need to offer CoS, the set of new functional components that are needed for this overlaps significantly with that required to implement the experimental, absolute, per-flow services required by Internet2. Traffic shapers, classifiers, policers, and parts of the proposed bandwidth broker functionality either exist today or will exist in next-generation routers.

An attractive quality of the DiffServ framework is that it may be deployed incrementally and offer useful pre-production services before a complete system is in place. The incremental deployments of the proposed Premium service and a predictive service like Assured are likely to complement each other nicely. Premium deployment would initially have a rigid service definition, but with very limited availability that would incrementally expand. In contrast, Assured would initially have nearly universal availability, but with a very weak service definition that would incrementally strengthen.

5. Summary and Conclusions

Over the past year, the Internet2 QoS Working Group has worked with the Internet2 applications and networking communities to identify the requirements for Internet2 QoS and the best technologies for meeting them. Although there are no off-the-shelf solutions that will fulfill the demanding array of requirements identified, the evolving differentiated services approach to QoS appears to offer the most promising approach.

Nevertheless, many hard technical and social problems remain that can be solved only through experience, research, and iterative design refinement in a testbed environment. In the coming months, Internet2 will facilitate a DiffServ testbed where new services can improve incrementally in quality

and availability and where technical experience can be gained for feedback to the relevant research, engineering, and standards efforts. On the social side, it is particularly important to begin to develop broader experience with QoS, so that developers, users, and administrators can begin to understand the expectational and policy transitions that are required before production QoS services can flourish.

We invite the reader to enjoy the following report from the First Internet2 Joint Applications/Engineering QoS Workshop. The two-day workshop was held at Bay Networks in Santa Clara, California, on May 21-22, 1998. It attracted an unusual and exciting diversity of participants. Campus network planners, advanced application developers, representatives from federal agency networks, university chief information officers, gigaPoP planners, and members of the network research community all met to investigate the many challenges and opportunities of getting Internet2 started on a concerted and multi-disciplinary effort to deploy a robust inter-domain QoS infrastructure.

6. Authors' Addresses

Ben Teitelbaum
Internet Engineer
Internet2
200 Business Park Drive, Suite 307
Armonk, NY 10504
Phone: 914-765-1118
Fax: 914-273-1809
Email: ben@internet2.edu

John Sikora
Technical Manager, Broadband and
Advanced Networking
AT&T Labs
101 Crawfords Corner Rd, Room 1J-307
Holmdel, NJ 07733
Phone: 732-949-1828
Email: jjs@arch4.att.com

Ted Hanss
Director of Applications
Internet2
3025 Boardwalk, Suite 100
Ann Arbor, MI 48108
Phone: 734-913-4256
Fax: 734-913-4255
Email: ted@internet2.edu

TELE-IMMERSION: THE ULTIMATE QoS- CRITICAL APPLICATION

Jaron Lanier, *The National Tele-immersion Initiative*

1. Introduction

A coalition of research universities is working to create a dramatic advance in virtual reality technology in order to provide a test application for Internet2.

Tele-immersion in its early years will serve as the toughest customer for new ideas from the Internet2 community, before those ideas can be widely adopted and eventually “locked in.” If the network can handle tele-immersion, it can handle anything.

2. Telecubicles, The First Tele-immersion Application

Tele-immersion differs from prior efforts in virtual reality in a variety of ways. In a tele-immersive interface the physical environment, including people, is continuously measured and transmitted as viewer-independent three-dimensional data. This opens up the possibility of photorealistic “avatars.” Furthermore, as a result of this continuous

measurement, features of people and other objects in the environment will be recognized and tracked, and displays will be dynamically autocalibrated. A tele-immersive interface will therefore be able to achieve an unprecedented integration of physical and virtual objects.

This is in turn part of a larger goal. Tele-immersive interfaces are being designed to fit into the larger workflow of users rather than being exceptional devices. The first such interface, the “telecubicle,” will allow immersive networking to be undertaken without giving up more familiar office tools. In order to achieve this ambitious vision, new research goals have been defined in the areas of network software architecture, sensors, display devices, high level authoring tools, database design, and image processing. Early applications will be in mechanical CAD (Computer Aided Design) and medicine.

3. QoS Requirements for Telecubicles

One of the purposes of the initiative is to provide a challenging empirical test of advanced network research.

TABLE 1.
Needs for a Successful Tele-immersive Experience

Type	Latency	Bandwidth	Reliable	Multicast	Security	Streaming	Dynamic QoS
Control	<30 ms	64 kb/s	yes	no	high	no	low
Text	<100 ms	64 kb/s	yes	no	med	no	low
Audio	<30 ms	N x 128 kb/s	no	yes	med	yes	med
Video	<100 ms	N x 5 Mb/s	no	yes	low	yes	med
Tracking	<10 ms	N x 128 kb/s	no	yes	low	yes	med
Database	<100 ms	>1 GB/s	yes	maybe	med	no	high
Simulation	<30 ms	>1 GB/s	mixed	maybe	med	maybe	high
Rendering	<30 ms	>1 GB/s	no	maybe	low	maybe	med

Source: Rick Stevens, Argonne National Laboratory

TABLE 2.
Estimates for Database Updates in Telecubicles

Function	Bandwidth	Loss	Latency	Jitter
New Object	1 MB/object	none	tolerant	TCP-like
Change Object				
Dance	216 kb/s	okay	low	okay
Bulldozer Plow	600 kb/s	okay	not critical	okay
Environment Scanning	15 GB/s	okay	minimal	okay
Facial Changes	*5–10 MB/s	okay	moderate	okay
Secondary Head Motion	low	none	low	none
Wavelet	43 kb/s	none	low	none
Haptic Data	1 kb/s	none	low	none

**with compression*

Source: Jaron Lanier, The National Tele-immersion Initiative

Telecubicles combine an intolerance for latency (because of the real-time interactions required for collaborative mechanical design) with a substantial bandwidth requirement (because of the need to transmit the current state of the physical environments).

Furthermore, each cubicle must multicast some, but not all, aspects of its current state to other cubicles. Because latency can never be completely eliminated, predictive techniques will be brought into play to reduce the apparent latency between cubicles.

At this point, it's not possible to firmly predict the requirements for bandwidth, latency, jitter, and packet loss required for suc-

cessful tele-immersive applications, such as the Telecubicles project. In May 1997, Rick Stevens of the Argonne National Laboratory made the projections shown in Table 1.

The Telecubicles project has created some estimates for database updates required for a tele-immersive experience, shown in Table 2.

4. Summary and Conclusions

Telecubicles are the ultimate test for QoS. Top networking and tele-immersion researchers will work together to implement and test what will be one of the most ambitious network applications to date.

THE ROLE OF QOS IN WIDE AREA DATA MINING

Robert Grossman
*National Center for Data Mining
University of Illinois at Chicago
and Magnify, Inc.*

1. Introduction

Data mining is the automatic discovery of patterns, associations, changes and anomalies in large data sets ([1] and [2]). Data mining has traditionally been focused on data which is located in a central warehouse and analyzed within memory. On the other hand most data is distributed. Developing technologies to mine distributed data is a fundamental challenge ([7], [6] and [8]).

It is important to distinguish searching for data on networks and data mining. An example should make this distinction clear. Finding all documents containing the key word sunspots is a typical search. On the other hand, consider the problem of searching for correlations between 25 years of sunspot data archived on a server in Boulder and 80 years of Southern night marine air temperature data archived on a server in Maryland. The goal of this data mining query is to understand whether sunspots might be correlated with climatic shifts in temperature.

Concretely, the data mining *process* can be viewed as the process of applying data mining *algorithms* to a learning set extracted from a data warehouse to produce a predictive model [6]. More accurate predictive models can always be obtained by moving distributed data to a central data warehouse. The problem is that this can be so expensive that most data is never mined. A fundamental trade-off in distributed data mining is whether to (a) move data or (b) keep the data in place, analyze the data separately, and merge the results. More precisely, the challenge is move as little data as possible while still maintaining acceptable accuracy.

QoS plays two different types of roles in distributed data mining:

1. Today we have a good understanding of distributed databases. The challenge with distributed databases is whether to use a strategy which returns data or one which returns the results of queries applied to data. Data mining is more complex in that, in addition, the possibility also exists of returning predictive models. The primary role of QoS in distributed data mining is to provide the necessary guarantees so that an appropriate strategy for either returning (a) data, (b) the results of queries, or (c) predictive models can be developed when mining data which is geographically distributed.
2. Visualization is an important technique in data mining, but today is limited primarily to mining local data. With QoS, visualization can also be used when mining geographically distributed data.

2. Four Examples

Interactive Exploration of Data. Both ftp and http can move files. But with web browsers and http people begin to work with networked documents in a fundamentally new way. Today the software infrastructure on the Web is focused on exploring (multimedia) documents. Distributed data mining is about the interactive exploration of networked data. For this to be possible, the data mining system must determine whether to move data, the results of queries, or predictive models. Given the data size, the accuracy required, and QoS guarantees, this becomes possible.

Decision Support. Data mining has proven successful for a variety of decision support applications such as detecting fraud and extending credit to consumers. The goal is to make decisions in near real time using as

much data as possible. Generally speaking, the more data which is available, the more accurate the predictive models. The problem is that as the number of data sources increases, so does the possibility that the entire decision will be delayed due to delays of accessing a few of the sites. With QoS, decisions can be reached even if it means not using some of the data due to the cost in accessing it.

Supporting Visualization in Data Mining.

Visualization is an important technique in data mining, but today is by and large limited to analyzing local data. Next generation networks supporting QoS would allow the combination of visualization with data analysis and data mining for understanding networked and distributed data.

Mining Image and Continuous Media Data.

The three applications above describe how QoS is an important enabling technology for mining distributed data. Working together QoS and data mining can be used to enhance remote access to digital libraries of image and continuous media data. As users interact with these types of digital libraries, agents can observe their queries and the responses of the system. By applying machine learning algorithms to this information, precision and recall can be improved, as well as intelligent caching and prefetching of image and continuous media data. Today, intelligent caching and prefetching is possible for local data — QoS enables a similar strategy for distributed data.

3. Clusters, Meta-Clusters and Super-Clusters

There are several approaches to distributed data mining. In this section, we sketch the approach described in [4, 1998b and 1998c], from which this section is adapted. Data mining can be computationally intensive and clusters of workstations connected by high performance networks are emerging as a popular platform for data mining [3]. Distributed data mining requires mining data across clusters connected by networks.

The following terminology is useful:

Workgroup cluster. A workgroup cluster refers to mining data resident on a cluster of workstations in a single location connected with a high performance network or other specialized hardware.

Meta-cluster. A meta-cluster consists of two or more distributed work group clusters connected with a network.

Super-cluster. A super-cluster consists of two or more distributed work group clusters connected with a high performance network.

For example, we have completed several experimental studies with a super-cluster consisting of three workgroup clusters in Chicago, Philadelphia, and College Park, Maryland. See Figure 1. It is important to distinguish between meta-clusters and super-clusters because the performance of the network often dictates different strategies for distributed data mining.

To return to the example in the first section, a scientist using a workstation which is part of the workgroup cluster in College Park may wish to engage in exploratory data analysis and correlate the sunspot data with a variety of data sets in Philadelphia and Chicago. A traditional approach would require moving the relevant files to College Park with ftp, building a common data set, and applying statistical programs. With distributed data mining and QoS, the appropriate data mining software could make intelligent decisions about returning data, the results of queries, or predictive models to supply approximate results to correlation queries and still be interactive. The strategies would obviously be different depending upon the size of the data sets, the QoS being supported, the time required to reformat the data, and the complexity of the computation required.

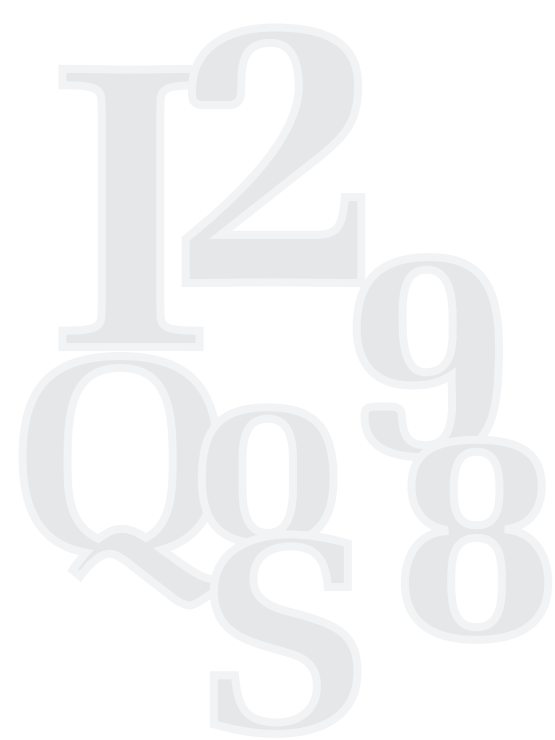
4. Summary

In this note, we have given a brief description of the role of QoS in distributed data

mining. Next generation networks supporting QoS should enable users to engage in exploratory data analysis, visualization, and data mining of data which is geographically distributed. The challenge today is not only to design and deploy networks with the appropriate QoS capabilities, but also to develop corresponding algorithms and middleware for distributed data mining. This should fundamentally change the way scientists and engineers analyze data.

5. References

- [1] Machine Learning Research: Four Current Directions, to appear.
- [2] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery: An Overview," in *Advances in Knowledge Discovery and Data Mining*, edited U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI Press/MIT Press, pp. 1–34, 1996.
- [3] R. L. Grossman, S. Bailey and D. Hanley, "Data Mining Using Light Weight Object Management in Clustered Computing Environments," *Proceedings of the Seventh International Workshop on Persistent Object Stores*, Morgan-Kaufmann, San Mateo, 1997, pp. 237–249.
- [4] R. L. Grossman, S. Bailey, A. Ramu and B. Malhi, P. Hallstrom, I. Pulleyn and X. Qin, The Management and Mining of Multiple Predictive Models Using the Predictive Modeling Markup Language (PMML), IST, to appear.
- [5] R. L. Grossman, Supporting the Data Mining Process with Next Generation Data Mining Systems, *Enterprise System Journal*, to appear, <http://www.lac.uic.edu/~grossman/papers/four-gen-dm-v7.htm>.
- [6] Robert Grossman, Stuart Bailey, Simon Kaisf, Don Mon, Ashok Ramu and Balinder Malhi, "The Preliminary Design of Papyrus: A System for High Performance, Distributed Data Mining over Clusters, Meta-Clusters and Super-Clusters," *Proceedings of the 1998 KDD Workshop on Distributed Data Mining*, AAAI Press, Menlo Park, California, 1998.
- [7] Y. Guo, S. M. Ruger, J. Sutiwaraphun and J. Forbes-Millott, Meta-Learning for Parallel Data Mining, submitted for publication.
- [8] S. Stolfo, A. L. Prodromidis and P. K. Chan, "JAM: Java Agents for Meta-Learning over Distributed Databases," in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, AAAI Press, Menlo Park, 1997.



A DISCRETE AND DYNAMIC APPROACH TO APPLICATION/OPERATING SYSTEM QoS RESOURCE MANAGEMENT

Scott Brandt, Gary Nutt, and Ken Klingenstein
University of Colorado at Boulder

1. Introduction

One of the primary goals of Internet2 is to enable widespread use of high-bandwidth, time-sensitive, distributed applications, such as video over IP or distributed virtual cubicles. Such “real-time” applications typically combine high resource demands with a low tolerance for failures or delays, yet they must also share finite resources from both the network and from the host computer through its operating system.

For these real-time applications to be successful, they must negotiate with both networks and operating systems for the maximum possible service levels available, and then adapt appropriately to fluctuations in resources. At the operating system layer, service requirements may include CPU, memory, and buffer commitments. At the network layer, resources include bandwidth, delay, and jitter.

While a great deal of emphasis has been placed on QoS from the network, QoS mechanisms for the operating system may yet prove to be the most important factor in facilitating such real-time applications. It is at this level our work is aimed.

Existing approaches to achieving quality of service from an operating system dictate that system resource usage outside the worst case requires enforcement. In contrast to this strict enforcement regime, we are exploring a middleware QoS solution that negotiates service levels between operating systems and applications. Our aim is to provide

assurances for service rates without making hard QoS guarantees and thereby “wasting” resources when full QoS allotments are not in use.

It appears that this QoS method could also be used to negotiate service levels between the application and the network.

2. Background

Operating systems designers have been creating mechanisms to support QoS-based soft real-time application execution. These mechanisms provide a variety of interfaces for determining the amount of resources allocated to an application, allowing a process to do three things:

1. Negotiate with the operating system for a specific amount of resources, as in RT Mach [7] and Rialto [5,6]
2. Specify a range of resource allocations, as in MMOSS [3]
3. Specify a measure of application priority that can be used to compute a fair resource allocation, as in SMART [8,9]

These systems all provide a mechanism that can be used to dynamically reduce the resource allotment granted to running applications. In the extreme, applications may be forced to dynamically adapt to a resource allocation lower than that required for average-case execution.

In creating these resource management mechanisms, operating systems developers have assumed that it is possible for applications to adjust behavior according to the availability of resources. OS developers have not, however, provided a general

model of application development for such an environment.

We are exploring a middleware solution that allows applications to cooperate with the operating system in their use of system resources, adapting to the current state of the system to maximize the benefit obtained from the available resources. This approach is in contrast with the operating system approach in which utilization outside the worst case requires enforcement. This distinction — negotiation and adaptation versus strict enforcement — is a major philosophical difference in our approach when compared to existing approaches to achieving quality of service from an operating system.

In previous papers [1,4,10,11] we presented execution levels, a method for dynamically managing soft real-time application execution in environments with varying QoS allocations. We also demonstrated the feasibility of cooperative middleware-based QoS management, discussed a prototype middleware execution-level-based QoS resource manager called the Dynamic QoS Resource Manager (DQM), and examined a set of representative QoS allocation algorithms within this context.

In continuing our research in this area, we have extended the DQM in several ways that bring it significantly closer to our goal of having a viable middleware QoS resource management agent [2]. In earlier work we used synthetic programs to experiment with the DQM, though now our tests are driven using working applications (two MPEG players). We have also expanded the adaptive capabilities of the DQM with a technique called Dynamic Estimate Refinement.

It is important to note that while these techniques directly address only the QoS for application to operating system interactions, it appears that the approach can be extended to application to network QoS interactions. One possible extension would be for applications to use similar tools to negotiate lev-

els for QoS with network layer devices, and, in turn, receive updates from the network for application adaptation. An alternative approach would be to include network layer interactions as part of the operating system interaction, and consider network capacity as one of the several resources that the operating system and application negotiate.

The latter approach would be very appealing if it is true that operating system QoS, rather than network QoS, is the greatest factor in end-to-end performance, as many have speculated.

3. Execution Levels and the DQM

Our research has focused on supporting soft real-time processes on general-purpose operating systems. Most soft real-time systems soften the real-time behavior of the applications by moderating the percentage of missed deadlines or the amount by which deadlines are missed; smaller amounts are considered better. This is adequate for the class of soft real-time processes for which missed deadlines are acceptable, but not all such processes fall into this category. For example, desktop playback of a fixed-bandwidth network-based continuous media stream does not allow for all deadlines to be missed by a certain amount because eventually the OS will run out of buffer space to hold the queue of frames slowly backing up. In this case a preferable solution would be one such as dropping frames or reducing the amount of processing for each frame so that the hard deadlines (enforced by the arrival of new data) can still be met.

In support of application-controlled adaptive behavior, we have developed the execution-level-based application execution model. With execution levels, each application is constructed using a set of algorithms for achieving its goals, ordered by their relative resource usage and the relative quality of their output. The execution levels are the application specification of the soft real-time policy it implements. The QoS manager provides the mechanism for managing the soft

real-time execution of the applications by dynamically adjusting the level of each running application. A variety of QoS allocation algorithms exist that adjust the applications according to the currently available resources.

In order to examine QoS-based soft real-time processing with the execution level model, we have developed a prototype system consisting of a middleware DQM and a library of DQM interface and soft real-time support functions called the Soft Real-Time Resource Library (SRL). This prototype system has allowed us to experiment with execution-level-based adaptive soft real-time processing. Like the flexible QoS systems cited above, the current implementation of our DQM works solely with the CPU resource. However, we believe that the concepts described in this paper can be extended in a straightforward manner to encompass other resources such as network bandwidth and memory.

The DQM dynamically determines a level for the running applications based on the available resources and the specified benefit of the application, and changes the level of each running application until all applications run without missing deadlines, the system utilization is above some predetermined minimum, and stability has been reached. Resource availability (or the lack thereof) is determined in a few different ways. CPU overload is determined by the incidence of deadline misses in the running applications. The SRL linked into each application notifies the DQM each time an application misses a deadline. CPU under-utilization is determined by examining CPU idle time. System idle time can be determined in several ways including via the operating system, through the /proc filesystem, by measuring the CPU usage of a low priority application, and by taking the complement of the CPU usage measurements (or estimates) of the running applications. If the operating system provides idle time information, this information is the most reliable.

4. Results

In order to examine the operation of the DQM, we have used both synthetic and real applications. The synthetic applications consume resources according to their level specifications without performing any useful work. They have allowed us to exercise the DQM with a wide variety of level specifications.

The two real applications are MPEG video players, but they differ in the way that their real-time behavior has been softened. The first application changes the frame rate of the displayed image from 0 frames per second to 10 frames per second. This particular application required no algorithmic changes other than the inclusion of the three SRL functions: `dqm_init()`, called once at the beginning of the application; `dqm_loop()`, called each time through the main loop; and `dqm_exit()`, called at application exit. The second application dynamically adjusts the size of the image displayed on the screen. Since the amount of work is related to how much time is spent drawing the pixels on the screen, this results in a reasonable range of CPU usage numbers over the different levels. `Dqm_loop()` returns the level at which the application should execute, so the main control loop of the application contains a switch which sets the size of the displayed image accordingly.

As shown in previous papers [1,2], our results to date have been very promising. The applications adapt quickly to changing resource availability and stabilize at appropriate execution levels as determined by the DQM.

5. Conclusions

Internet2-based applications will make tremendous demands for the use of host computer and network resources. Information must flow from an application through the host operating system, over the Internet, up through a second host operating system, and into a receiver application. Because of the heavy demand for various

resources to support this data movement, these applications must be prepared to participate in the way the resources are managed across the application's components (as well as across applications). The execution level model and DQM demonstrate a sound approach for managing the resources in a soft real-time environment within a host computer.

The contribution of this work to the Internet2 community is clear. Execution levels are a simple and natural model for developing resource-sensitive adaptive applications, and the DQM demonstrates one way in which a host can support the execution of such applications. We have illustrated the utility of this new approach to resource management, an approach that can be applied to network bandwidth as well as CPU resources.

Just as host machines are oversubscribed for their resources, the Internet is (and will continue to be) oversubscribed for its bandwidth. In such an operating environment, host computers must be able to administer the network bandwidth so as to provide assurances for service rates without making hard QoS guarantees. We believe execution levels and the DQM can be the basis of a sound approach for managing the allocation of Internet2 bandwidth.

5. Acknowledgements

Scott Brandt and Gary Nutt were partially supported by NSF grant number IRI-9307619.

6. References

- [1] S. Brandt, G. Nutt, T. Berk, and M. Humphrey. "Soft Real-Time Application Execution with Dynamic Quality of Service Assurance." *Proceedings of the Sixth IEEE/IFIP International Workshop on Quality of Service*, pp. 154–163, May 1998.
- [2] S. Brandt, G. Nutt, T. Berk, and J. Mankovich. A Dynamic Quality of Service Middleware Agent for Mediating Application Resource Usage. Submitted for publication, May 1998.
- [3] C. Fan. "Realizing a Soft Real-Time Framework for Supporting Distributed Multimedia Applications." *Proceedings of the 5th IEEE Workshop on the Future Trends of Distributed Computing Systems*, pp. 128–134, August 1995.
- [4] M. Humphrey, T. Berk, S. Brandt, G. Nutt. "The DQM Architecture: Middleware for Application-centered QoS Resource Management." *IEEE Workshop on Middleware for Distributed Real-Time Systems and Services*, December 1997, pp. 97–104.
- [5] M. Jones, J. Barbera III, and A. Forin. "An Overview of the Rialto Real-Time Architecture." *Proceedings of the Seventh ACM SIGOPS European Workshop*, pp. 249–256, September 1996.
- [6] M. Jones, D. Rosu, M. Rosu. "CPU Reservations & Time Constraints: Efficient Predictable Scheduling of Independent Activities." *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, October 1997.
- [7] C. Mercer, S. Savage and H. Tokuda. "Processor Capacity Reserves: Operating System Support for Multimedia Applications." *Proceedings of the International Conference on Multimedia Computing and Systems*, pp. 90–99, May 1994.
- [8] J. Nieh and M. Lam. "The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Applications." *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, October 1997.
- [9] J. Nieh and M. Lam. "Integrated Processor Scheduling for Multimedia." *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, April 1995.
- [10] G. Nutt, T. Berk, S. Brandt, M. Humphrey, and S. Siewert. "Resource Management of a Virtual Planning Room." *Proceedings of the Third International Workshop on Multimedia Information Systems*, September 1997.
- [11] G. Nutt, S. Brandt, A. Griff, S. Siewert, M. Humphrey, and T. Berk. Dynamically Negotiated Resource Management for Data Intensive Application Suites. *Transactions on Knowledge and Data Engineering*, to appear.

DIFFERENTIATED SERVICES FOR THE INTERNET

Van Jacobson,¹
Lawrence Berkeley National Laboratory

1. Architecture Requirements

Let us be clear about what problem we are solving. The objective with QoS is to give better service to some traffic at the expense of giving worse service to the rest. ATM marketing fantasies to the contrary, QoS is a zero-sum game. QoS does not create bandwidth and, since bandwidth allocated to some is bandwidth not available to others, QoS does not guarantee that you get better service.

1.1 Service Models

QoS is “quality-of-service.” Therefore, let us also be clear about what “service” we are talking about. Basically, there are two camps: The ISP community, which is most vocal in the IETF, is very interested in a “better best effort” approach based on relative classes of service. ISPs want this control because they see a simple way to sell high-paying customers a “better” service. This may be implemented with drop preference or weighted-round-robin queuing disciplines.

At odds with this, is what the users themselves want. Many applications have absolute bandwidth requirements. For example, Jaron Lanier may want to run a tele-immersion experiment, but cannot afford to pull a wire between hither and yon. However, the nature of the service that he needs is exactly equivalent to a wire. It does Jaron no good at all to have 10 times better service than anyone else — that may mean 10 bps or may mean 10 Mbps! His application has certain bandwidth and latency needs and, if they are not met, it simply does not work. This “virtual leased line” service model may be implemented with priority queuing and strict policing.

1.2 Target Applications

So, should we design an Internet2 QoS architecture for tele-immersion? What are the other target applications for which we should be designing? These are bad questions! The killer networked application has always changed over time. In 1978, there was remote job entry (RJE); in 1988, it was email and FTP; now, it is certainly the Web. But that too will change. The strength of the Internet has been that its protocols work for any application.

1.3 Policy Control

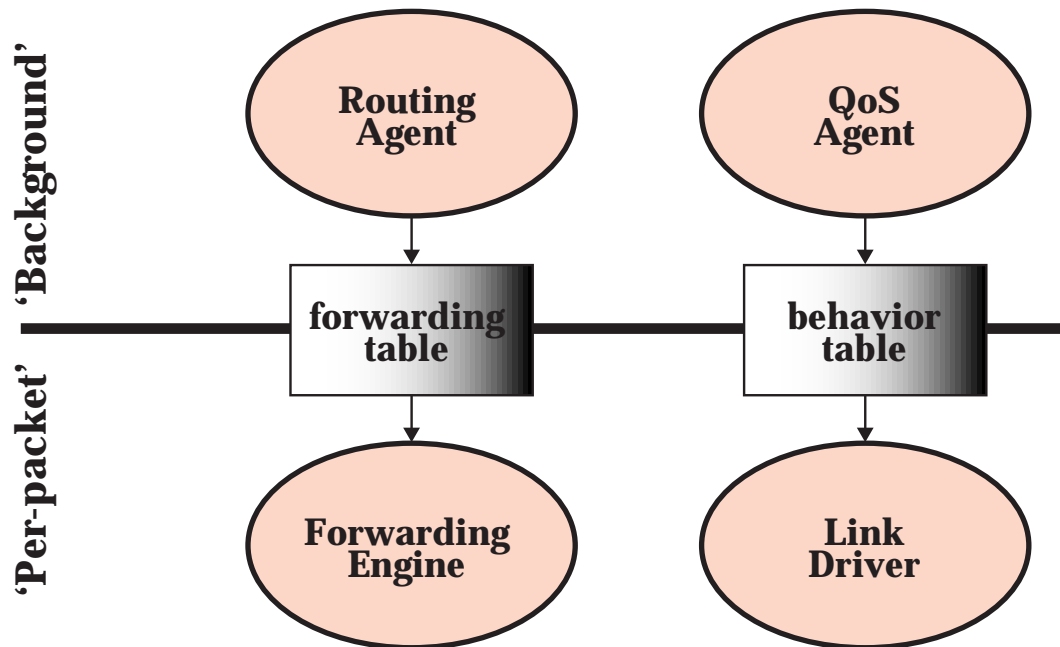
Since institutions pay the bills, QoS must give each institution control over how its bandwidth is shared among different users. There are three consequences to this. First, there must be some way for an institution to communicate its sharing policy to the network. Second, since QoS will create a better service, it will be tempting to steal it and, therefore, any base-level architecture must include security and workable trust models. Finally, if a QoS architecture attempts to limit the state in the network (as does DiffServ), there might be only one or two places in the network where policy can be enforced.

1.4 Scaling

The Internet is growing at 20% per-month and works across 10 orders of magnitude - from 300 baud links to multi-gigabit speeds. There is such huge fan-in to the core and the growth rate is so much faster than Moore's Law, that router companies will not be able to build boxes that can keep up and support significantly more per-packet forwarding complexity. Therefore, for an Internet QoS architecture to succeed, it must have the same scaling properties as the rest of the Internet. This scaling is possible only if we push all per-packet state and all per-conversation policing and shaping to the network

1. This abstract was ghostwritten by the editors, but is closely based on a recording of Jacobson's talk and on his slides.

FIGURE 1.
*IP Gateway
Architecture*



edge. Then, as the Internet grows, so does the edge. And, as the edge grows, so do the number of places and cycles available to perform computationally intensive per-packet work.

2. The Differentiated Services Architecture

2.1 Consequences of the Need to Scale

There are three consequences to the scaling imperative. First, edge-only state suggests that the special/normal service indication must be carried in the packet. Second, administrative diversity and high speed forwarding both argue for a very simple semantics for that indication (e.g. a space of possibilities that could be represented by a few bits). Third, the absence of state in the center means that the network sees only aggregates, which could lead to fairness problems if the sharing control is not done carefully.

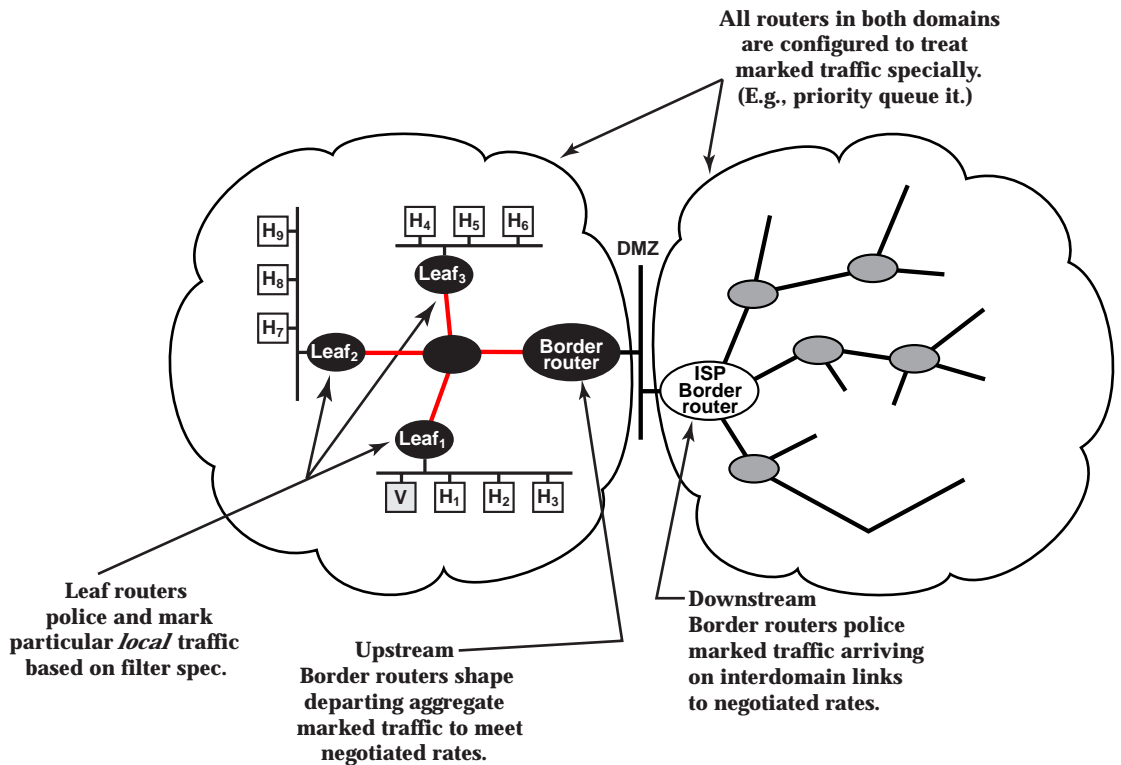
Almost all Internet traffic crosses many administrative boundaries. To achieve a meaningful end-to-end service, all administrative domains along a QoS flow's path must agree to treat its traffic as special. However, multilateral agreements rarely work. ISPs are competitive enterprises that

act in their own best interests and, where necessary, against the interest of their competitors. To scale the administrative complexity of inter-domain QoS, end-to-end services must instead be constructed from concatenations of bilateral agreements. These agreements cannot realistically require extending trust or control across administrative boundaries and there must also be fault isolation — customers should be able to trash only their own networks.

2.2 Engineering Absolute Services

For new internet services to be economically viable, customers have to know what they're buying. The service delivered cannot depend on other people's traffic and the customer must be able to measure conformance to a service agreement specified in absolute terms. There are two engineering consequences to this. First, an ISP must not overbook its premium service — simple queuing theory dictates that if the aggregate inflow is greater than the outflow, the delivered bandwidth, drop rate and/or delay could be arbitrarily bad. Second, even if the allocated service is limited to capacity, aggregation can cause some users to get poor service

FIGURE 2.
DiffServ Architectural
Components



unless traffic shaping is done at the borders. This is because, unlike voice traffic, data traffic is not Poisson. Burstiness increases when you aggregate and, therefore, one needs to think a lot about preserving the smoothness of aggregates even over relatively small time scales.

2.3 Services Versus Behaviors

Do we standardize services or per-hop packet forwarding behaviors? If the answer is “services”, then everyone has to agree on what constitutes a useful service and every router has to implement the machinery for it. In order to deploy a new service this way, you would have to upgrade the world. This makes no sense. Since a router cannot actually do that many different things to a packet, it makes more sense to standardize forwarding behaviors. A few simple forwarding behaviors like “send this packet first” or “drop this packet last” can be combined with rules enforced at the edge to provide a broad spectrum of services.

One way to think about this is that it is analogous to the way that an IP router separates forwarding from routing (see Figure 1). For forwarding, the router relies on a forwarding

table on which a longest-prefix match is performed to discover the interface out which a packet is to be forwarded. There is nothing end-to-end about forwarding; it is a completely simple, local, and high-speed decision. The magic necessary to get an end-to-end service is all done by routing agents which work in the background to set up the forwarding tables in a globally consistent way. Analogously with DiffServ, a QoS agent builds a behavior table for the link driver to use. In both cases, one piece of software provides the necessary background information for another piece of software to make rapid per-packet decisions.

This is a really powerful separation and one that the telcos never figured out. They built the notion of voice calls into the very lowest levels of their networks and lost the opportunity to build a flexible and adaptable network. With this separation of background and per-hop processes, new services can be achieved by appropriately munging the forwarding or behavior tables. As evidence of this, note the continual evolution of routing (e.g. static routing, shortest-path-first and distance-vector algorithms, policy based routing, BGPvs-IGPs) contrasted with the total lack of evolution in IP forwarding.

Similarly, there will likely be much evolution in the QoS policy and resource management agents once a few per-hop forwarding behaviors have been standardized.

2.4 Architectural Components

Ten years ago, Dave Clark (MIT LCS) and I proposed “edge-tagging” as a scalable way of offering differentiated services. With this scheme (illustrated in Figure 2), a leaf router adjacent to the source has a traffic signature for “special” traffic and a “profile meter” giving the characteristics of that traffic. The leaf router marks all special traffic that conforms to its profile meter and unmarks all other traffic. Marking is accomplished by setting bits in the IP header’s ToS field. Downstream border routers police marked traffic arriving on interdomain links to negotiated aggregate rates. Upstream border routers (which are aware of the bilateral service level agreements being enforced downstream) shape departing traffic aggregates to meet the negotiated rates. Finally, all routers in both domains are configured to treat marked traffic specially — for example, to give it priority queuing.

This architecture still leaves many questions unanswered. Who decides which users are allowed to request special services? Where is institutional policy implemented? Who tells the edge router what to tag? Who makes sure that simultaneous users of special services fit within allocation?

To address these needs, we need somewhere to put a policy database of priorities and limits for user and project access to special bandwidth. For want of a better name, we will call the repository for this information a bandwidth broker (BB). The bandwidth broker includes user credentials so that requests can be authenticated and is part of the network infrastructure, so that it can have a trusted and secure association with all local routers. Requests go from the user to the BB so that it can record use and resolve conflicts. Only then, could a request result in configuring a leaf router policer, so the security model is well-founded.

2.5 Signaling and Allocating for End-to-End QoS

Per-flow end-to-end QoS is established by the local BB on behalf of a user by signaling

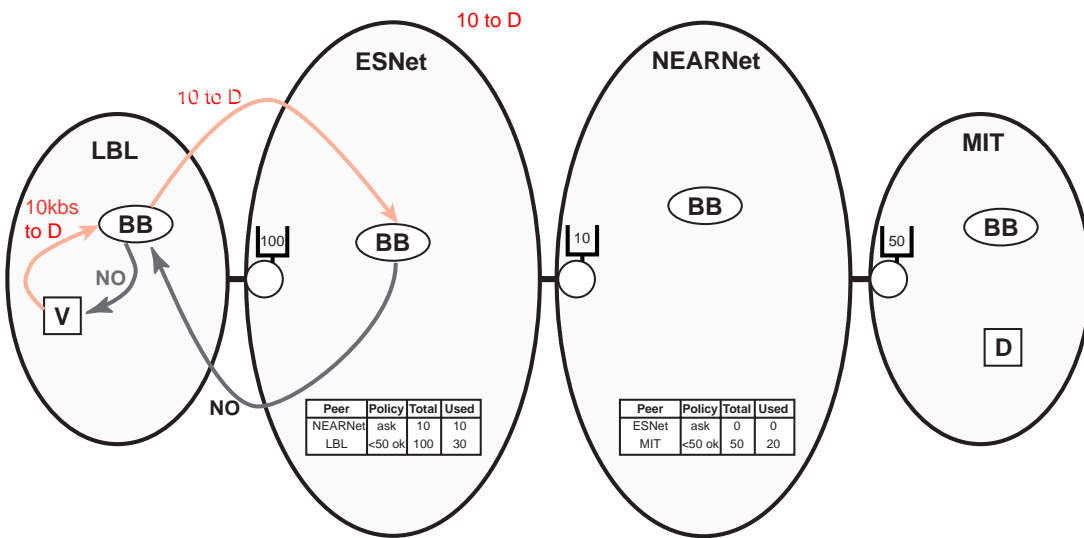
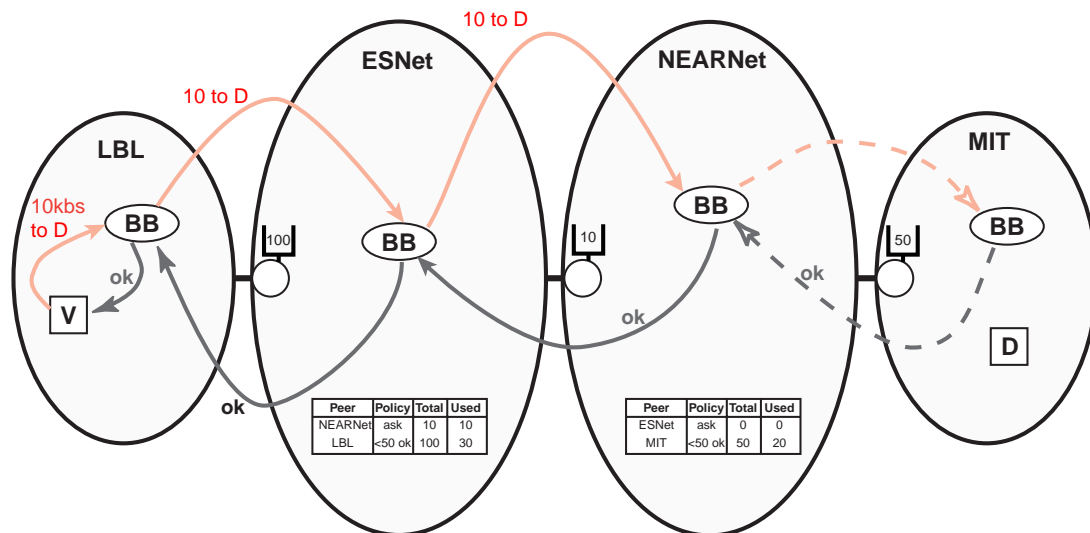


FIGURE 3.
Example: Interdomain Request With All Resources in Use

FIGURE 4.
Example: Successful
End-to-End
Interdomain Request



the cloud-by-cloud chain of BBs that manage the bandwidth along the end-to-end path (see Figure 4). But, who talks to the local BB and how do they do it? Because there is no need to constrain the architecture on this point, the answer is: “whatever works.” Certainly site network administrators will need to talk to the BB and one could have hosts making direct requests. However, hosts probably do not want to know about their BB directly — hosts more traditionally know about the network through their first-hop routers. One likely possibility is that hosts (or even applications) would signal their local BB for QoS through their first-hop router via a “lite” version of RSVP.

Should bandwidth be allocated across a boundary or along a path? There is no one right answer here either. However, path allocation is much, much harder, as it forces the allocator to know the full topology and possibility to subvert routing to “pin” the path. Knowing the full topology of even a single non-trivial network cloud is very hard, as it changes from millisecond to millisecond. Furthermore, knowing how to allocate bandwidth optimally along paths is provably NP-hard. Luckily, there is no need to solve these really hard problems because most topologies allow us to stick with very simple solutions.

3. Examples

3.1 Topologies

Consider a switched LAN campus infrastructure with at least 10Mbps everywhere internally and priority queuing in the switches. By accounting for allocation from a single 10Mbps “premium” bandwidth pool with no topological knowledge whatsoever, a bandwidth broker may conservatively allocate the resources of the campus network. Note that even with this extremely conservative allocation, there is room within the 10Mbps for 300 simultaneous voice/video sessions.

If you have two such campuses connected by a low-speed WAN link like a T1 line, you can add a T1-sized pool of bandwidth for external conversations to the previous allocation scheme. Again, the BB knows nothing about topology—it only knows that there are two bottlenecks and keeps counters for each. Finally, if you have multiple campuses connected to a common ISP by low-speed tails, each campus can run a simple allocator for its campus-to-ISP link, while the ISP runs a simple allocator for each of its ISP-to-campus links.

3.2 Requests

The bandwidth broker abstraction exists in order to make bandwidth sharing decisions

reflect institutional priorities and policies, but it also provides a natural vehicle for dynamic interdomain setup and control. For example, say that between LBL and MIT lie two other domains, ESNet and NearNet, and that all four domains are equipped with BBs (see Figure 3). Each domain has an agreement with each of its neighbors to accept and deliver packets at some specified rate. In Figure 3, LBL's remaining LBL-ESNet allocation is 100kbs, while ESNet's remaining ESNet-NearNet allocation is 0Kbps. Suppose that Van at LBL wants to establish a 10Kbps connection to Dave at MIT. His workstation asks the LBL BB for the allocation.

Since 10Kbps is fine with it, it passes the request on to ESNet's BB, which sees that this would overdraw its account with NearNet. ESNet's BB returns this information to LBL's, which in turn passes a "Reject" message back to Van's workstation.

On the other hand, if there is room within the contracted ESNet-to-NearNet profile, as shown in Figure 4, NearNet's BB could pass the request on to MIT. However, Figure 4 shows a slightly different possibility that illustrates the diversity of possible bilateral

agreements that might flourish under DiffServ. In the example above, the NEARNet-to-MIT agreement specifies that for aggregate premium loads under 50Kbps, NEARNet need not actually ask MIT's BB, but can simply go ahead and allocate the bandwidth across their link without asking. In the example in Figure 4, there is ample bandwidth along the full end-to-end path, resulting in an acknowledgement that is returned to the LBL BB, which can in turn configure the leaf node policer nearest to Van's workstation to accept 10Kbps of Premium traffic from Vern to MIT.

4. Conclusion

The DiffServ architecture represents a promising alternative to achieving scalable, interdomain quality of service. Some work has already begun within ESNet to prototype a Premium service, as was demonstrated at SC'97. However, there is still much work to be done to refine the architecture and evolve good models for bandwidth brokers. Internet2 can play an important role in providing a testbed for experimentation with DiffServ techniques.



EVOLUTION OF END-TO-END QOS: A DESIGN PHILOSOPHY

John Wroclawski, *MIT Lab for Computer Science*

1. Introduction

It is obvious that networked applications depend on networks that work. Building such networks is a task of many parts, ranging from the finest details of routing and traffic scheduling to the down-to-earth (ahem) issue of managing the bulldozers that install wires and fiber. One part of building effective networks, network Quality of Service control, or QoS, allows users or applications to specify, request, and then count on specific levels of performance from the network.

The demand for network QoS control exists for several reasons. New classes of network services may be needed to support new kinds of applications. Networks that can deliver a known level of service with high assurance may be needed for mission-critical uses. Network owners and operators may want to offer different classes of service to maximize revenue — the same yield management strategy that is used so effectively today by airline operators. Finally, specific, known levels of network service may be demanded by accounting or legal policies, independently of whether they provide actual value to the network user.

Many designers of QoS control strategies have heavily emphasized one or the other of these motivators. An important example, the Internet's RSVP and Integrated Services work, was originally driven by the perceived needs of real-time and multimedia applications. More recently, though, Internet researchers have taken a broader focus, looking for mechanisms that gracefully combine support for new applications, higher levels of assurance for traditional applications, and traffic allocation and management capabili-

ties for network operators. These desires, together with some concerns about the scalability of traditional per-flow reservation techniques, have led to a new class of QoS control architectures colloquially known as Differentiated Services, or DiffServ.

DiffServ's name arises from the fact that QoS control is essentially about treating different customers or groups of customers differently. One type of difference, important to many network operators, is the ability to assign relative levels of importance to different customers (e.g. dropping the packets of non-paying customers before the packets of paying customers). Another type of difference, and the focus of this paper, is the ability to offer application users different quantifiable, absolute levels of performance, and different degrees of commitment that they will receive that absolute level.

Like any QoS control approach, a DiffServ design must provide four functions in order to implement these different levels of performance and assurance:

1. Admission control, the ability of the network to refuse customers when demand exceeds capacity.
2. Packet scheduling, a method for treating different customers' data differently as needed.
3. Traffic classification, the ability to sort data streams passing through the net into the sub-streams that receive these different treatments.
4. A set of policies and rules for allocating the network's resources.

In traditional QoS architectures the first three of these functions are repeated at each hop along a data path in the network. Thus each RSVP router or ATM switch in a net is expected to independently admit, classify, and schedule the packets in a single session

or connection request. In contrast, the DiffServ model distributes these functions across a domain, which is typically a single provider's network or similar administratively-consistent region. Inside the domain, traffic has been sorted into broad classes based only on the per-hop scheduling behavior, or PHB, it requires. This sorting, together with other QoS functions such as admission control, happens at the edge of the domain in an edge behavior device or "Profile Meter." It is the combination of sorting and profile metering at the network edge that yields the desired QoS control behavior.

2. End-to-end QoS Control Within a DiffServ Framework

One simple but useful QoS service that can be provided with DiffServ mechanisms is a leased line emulation widely known as "Premium service." The definition of this service is straightforward at the network level:

- Users must ask for some amount of Premium-service bandwidth — the service is admission controlled. (This does not imply "ask dynamically." One might ask to sign a contract to purchase a specified amount of Premium service for a month, for example.)
- If the request is granted, the Premium-service user will receive the requested amount of bandwidth, and will not receive any more than the requested amount.
- The user's traffic will pass through the net with essentially no queuing latency.

Premium service can be implemented easily using a DiffServ approach. Elements in the "middle" of a DiffServ domain simply carry all Premium traffic in a separate, high-priority queue. The profile meter at the edge of each domain allows any Premium traffic packets that are within the requested bandwidth constraint to enter the domain, and discards any traffic that is over the bandwidth limit.

It is easy to construct an end-to-end Premium service by concatenating domains, because the amount of Premium capacity needed from a next-hop domain is simply the sum of all previous-hop Premium services that do not terminate in the local domain. There are no statistical multiplexing or aggregation effects. Thus, DiffServ mechanisms can provide useful end-to-end QoS control.

However, the DiffServ QoS environment is likely to evolve far beyond this simple starting point.

3. A Design Philosophy

Future approaches to QoS should respond to the needs of adaptive applications and to the needs of applications designers. They should offer a much more flexible control mode, allowing for either sender or receiver control of QoS, independent of the application. And they should be constituted of reusable "building blocks" behaviors easily adjusted in order to implement the widest possible variety of services.

3.1 Services for Adaptive Applications

Adaptive applications can adjust to their environments, to offer their users the best level of service possible in any given situation. These applications have the ability to take advantage of any extra network capacity that may be available to improve their performance. In addition to adding robustness, this allows the same application to meet a wide variety of user price/performance needs simply by adjusting the level of available resources. A single adaptive application can cover a broad range of customer requirements.

When used without network QoS control, the user-level performance of adaptive applications can swing through a wide range, from excellent to unacceptable. The goal of QoS control in this case is often to bound the range, allowing the user to purchase some minimum level of service, benefiting from any extra capacity that is avail-

able but preventing the application from swinging into the unacceptable range. Thus, it is desirable for QoS control services for adaptive applications to have a “best effort with floor” nature, rather than the hard clamp nature of the Premium service described earlier.

This type of lower-bound QoS control will be useful with many classes of adaptive applications, including some TCP-based applications, many reliable multicast and group communication applications, and many multimedia human communication applications.

3.2 Application Oriented Service Specifications

Network QoS requests are traditionally specified in low-level terms such as raw bandwidth and latency. These types of parameters make sense to the network designer, but may not be directly useful to the application designer. However, QoS services can be more directly tied to application needs, as in Allocated Capacity [1], described in more detail in Section 4.

Allocated Capacity service profiles express QoS as actual TCP throughput, rather than raw network bandwidth. The application designer is saved the tasks of understanding TCP’s detailed performance behavior and translating application-level requirements into lower-level parameters.

3.3 Sender, Receiver, and Zone control of QoS

Many approaches to network QoS control assume that the sender of the data should also determine the level of service with which it is delivered. This assumption typically derives from implementation simplicity.

The Internet’s RSVP protocol adopted the model of receiver-based control of QoS. This choice was motivated by heterogeneity. The designers of RSVP recognized that, particularly for multicast sessions, only the receiving applications and users had the knowl-

edge to choose the appropriate parameters for RSVP’s resource reservations.

However, neither of these approaches gets to the heart of the matter.

Logically, the choice of QoS should fall to the user who is receiving value from the data transmission and would be prepared to pay for the QoS chosen. This choice is independent of protocol or application. We can see an analogy in the telephone system, where the same application and network supports both normal sender-based and 1-800 receiver-based call payment methods.

A versatile QoS mechanism should allow for either sender or receiver control of QoS, independent of the application.

However, even more flexible models may be of interest. One possibility is a “zone” model that allows for each participant in a session to control (and presumably pay for) the QoS used to transport data through part of the network. For example, the owner of a high-volume Web site might choose to “push” its traffic into the core of the network with high QoS, and leave to each individual browser whether to “pull” the data the rest of the way with high assurance or merely best-effort service.

3.4 Reusable Building Blocks

Traditional QoS frameworks, including ATM and the Internet’s Integrated Services effort, took the approach of designing and specifying a small number of (hopefully) widely applicable network services that could be provided to applications.

Much current thinking in the DiffServ community is along similar lines, i.e., approaching the problem of end-to-end QoS by identifying a small set of (hopefully) widely applicable end-to-end services, and then defining per-hop behaviors and edge behaviors that will implement each of those services.

An alternative approach, currently being studied at MIT, is based on DiffServ’s separa-

tion of per-hop behaviors and edge behaviors. In this approach, called Allocated Capacity Profiles, the overriding purpose of a per-hop behavior is not to support a single service, but instead to act as a versatile building block that can implement the widest possible variety of services when used in conjunction with different edge behaviors.

Under this scheme, rapid evolution of existing QoS services and deployment of new services for new applications require only the design of new edge behaviors, rather than upgrading every network node. The task of the per-hop behavior design in this model is twofold: (1) to effectively and efficiently implement a wide variety of QoS services, and (2) to simplify the job of implementing new behaviors by simplifying the task of designing new edge behaviors.

An interesting aspect of reusable per-hop behaviors of this sort is that they are most flexible when they are most completely and accurately specified. This is because the more precisely the behavior of a PHB can be understood and relied upon, the simpler it is to design an edge behavior that will reliably combine with it to create a specific overall service. This is in contrast with PHBs that are implicitly designed to support one specific service. In such a case, it may be best to specify as little as possible about the PHB, in order to avoid restricting implementation options.

4. Allocated Capacity Profiles — An Example

The goal of Allocated Capacity Profiles [1] is to distribute the available capacity (bandwidth) of a network among different users of TCP-based applications so that

- The actual application performance target desired by a user is described by a capacity profile.
- Different users can have different profiles, so that the available bandwidth of the network can be unevenly allocated among the users.

- Each user that has obtained a capacity profile has a high expectation or assurance of obtaining the performance specified by the profile, even when the network is overloaded.
- When the network is underloaded, users will obtain better performance than that specified in their profiles — the profile specifies a performance floor, not a fixed point. When the network is not fully loaded by users operating within their capacity profiles, bandwidth is shared between best-effort traffic and users operating above the “floor” of their capacity profile.

Allocated Capacity Profiles are implemented using a simple extension of the RED queue management algorithm called RIO (RED with In and Out marking) as the per-hop behavior, and an edge profile meter that marks packets as in or out of profile based on the user’s target throughput level. These two mechanisms, working together, cause individual TCPs to limit their throughput to the target level when the network is full.

The RIO per-hop behavior used to implement this service is a reusable building block. It can also be used to implement other QoS control services, like simple fixed-capacity bulk-bandwidth service. This service is similar to the Premium service described above, but without the additional promise of low latency.

5. Summary

The DiffServ approach can potentially provide the basis for an efficient, scalable, and service-rich end-to-end application QoS environment. Many of the benefits of the approach appear when PHBs are designed to act as lower-level building blocks, rather than being tuned to a single specific service. The overall DiffServ approach effectively couples technical and economic issues, and offers several inducements for provider deployment. Research on end-to-end differentiated services QoS mechanisms is at an

early stage, and significant further improvements and capabilities will likely be developed if demand exists.

Reference

[1] "Explicit Allocation of Best Effort Packet Delivery Service," Wenjia Fang and David D. Clark. To appear in IEEE Transactions on Networking. Presently available as <<ftp://mercury.lcs.mit.edu/pub/wfang/expected-capacity.ps>>.

A SCALABLE RESOURCE MANAGEMENT FRAMEWORK FOR DIFFERENTIATED SERVICES

Lixia Zhang, *UCLA*

1. Introduction

From a router's viewpoint, QoS support is made of three basic steps: defining packet treatment classes, allocating an adequate amount of resources for each class, and sorting all incoming packets into their corresponding classes. The first step involves standardization, while both the second and third steps require protocols and mechanisms that can scale well with the ever-increasing network sizes and speeds.

Through a joint effort of the research community and industry, the IETF Differentiated Services (DiffServ) working group is reaching agreement on an initial set of definitions for "per-hop behaviors" (PHBs), a set of differentiated packet treatments at each router based on the ToS field value (called "code-point") carried in the IP header [1]. This work addresses both the first and third issues above: it defines the traffic classes as well as provides a simple packet classification mechanism — routers easily sort packets into their corresponding treatment classes by the ToS value, without having to know which flows or what types of applications the packets belong to.

As work on DiffServ progresses, there has been a continued discussion on the second issue, that is whether differentiated services would need any signaling protocols for dynamic resource management. A commonly perceived notion is that manually configured resource allocations should be able to give us a jump start on deploying differentiated services, and that end-to-end performance can be achieved through the concatena-

tion of PHBs [3]. However we also believe that automatic protocol mechanisms will be needed in the near future as the volume and scope of QoS-requiring traffic increases, to effectively and efficiently meet the ultimate goal of end-to-end QoS delivery.

2. A Picture of the Internet Today

The Internet today is made of the interconnection of multiple autonomous networks called autonomous systems, or administrative domains, each under a separate administrative control. Each domain contracts its neighboring domain(s) to deliver its IP traffic; the neighbor domain, in turn, may pass the traffic to its neighbors, so on and so forth until packets reach their destinations. For example, a campus contracts an ISP to deliver its traffic and the ISP delivers the campus' traffic either over its own network if the destinations are connected to the same ISP, or otherwise passes the packets to other ISPs.

Following the administrative-domain-based network topology, as described above, today's Internet routing architecture is a two-level hierarchical design. Each administrative domain, or Autonomous System (AS), is free to choose whatever routing protocol it deems proper to run internally. To assure global connectivity, neighbor domains speak BGP (Border Gateway Protocol) with each other to exchange network reachability information. Reachability information can be aggregated. For example, if nearby networks share common prefixes, then their reachability reports can be merged, so that a remote site will keep only one entry in its forwarding table showing the common prefix. The separation of the Interior Gateway Protocols (IGPs) and the Border Gateway Protocol,

coupled with the ability to aggregate routing advertisements, provides the routing architecture with proven scaling characteristics.

3. A Framework for Scalable Resource Management

Following the structure of the global routing architecture, we propose an analogous hierarchical approach to resource allocation for the global Internet. We suggest that individual administrative domains be the basic control unit for resource management. Bilateral service level agreements (SLAs) are made between neighboring administrative domains regarding the aggregate border-crossing traffic, while each administrative domain individually makes its own decisions on strategies and protocols to use for internal resource management to meet internal clients' QoS needs and to fulfill external commitments.

More specifically, we assume that a resource manager, named the Bandwidth Broker (BB) by Van Jacobson of LBNL, exists in each administrative domain [1]. A BB will be in charge of both the internal affairs and the external relations regarding resource management and traffic control. Internally, a BB keeps track of QoS requests from individual users and applications, and allocates internal resources according to the domain-specific resource usage policies. Those policies specify which users may use how much resource or resource shares (and perhaps also under what specific conditions). The internal resource allocation can be done in a number of ways. For example, for bandwidth-rich domains, perhaps little needs to be done other than closely monitoring the network utilization level and re-provisioning accordingly; for bandwidth-poor domains or those with high variation in link capacities, on the other hand, the BB may make use of some internal signaling protocol, such as RSVP, to reserve bandwidth for individual applications [3].

Externally, a BB will be responsible for setting up and maintaining bilateral service

agreements with the BBs of neighbor domains regarding the QoS handling of its border-crossing data traffic. The BB for a transit domain (i.e. a provider network) must also keep those external service commitments within its internal resources capacity.

To scale with both the number of application flows and the link speed, the SLAs between BBs will be in terms of differentiated traffic classes. A BB collects from internal users/applications the requests for external resources, and based on these requests it makes its SLA arrangement; it may also readjust the SLA according to changing demand and conditions. Individual BBs tell their own border routers how much traffic each border router can export and import for each PHB class; therefore border routers only need to perform QoS control on the granularity of DiffServ traffic classes.

There remain a number of challenges in realizing this proposed two-level resource management. For example, the BB-to-BB communications must be secure, robust, and scalable. To scale well it is desirable that BB-to-BB resource requests be destination-independent, that is, one domain tells its neighbor domain how much bandwidth should be reserved for Premium traffic, without having to list all possible destination domains. We also have yet to understand what is the best way to implement the BB. The BB is a logical entity; actual implementations may take either a centralized or a distributed approach, or a combination of both.

4. Summary

Today's Internet is made of interconnected autonomous networks controlled by administrative domains. Following the global routing architecture we have proposed a two-tier hierarchical framework for global Internet resource management, in which resource allocations between domains will be done for aggregate border-crossing traffic, while allocations within a domain can be done with a number of different possibilities depending

on the user requirements and resource availability. We believe this two-tier hierarchical framework gives us both scalability in providing global-scale QoS support, as well as flexibility in managing resources within each administrative domain.

References

- [1] "Differentiated Services Operational Model and Definitions", Nichols and Blake, editors. INTERNET-DRAFT <draft-nichols-dsopdef-00.txt>, February 1998.
- [2] "A Two-bit Differentiated Services Architecture for the Internet", Nichols, Jacobson, Zhang. Internet Draft <draft-nichols-diff-serv-arch-00.txt>, December 1997.
- [3] "A Framework for Use of RSVP with Diff-serv Networks," Bernet, Yavatkar, Ford, Baker, Zhang, Nichols, Speer. Internet Draft, draft-ietf-diffserv-rsvp-00.txt, June 1998.



NYSERNET QOS AND POLICY ROUTING

Jim Grisham, *NYSERNet*

1. Background

NYSERNet (the New York State Educational and Research Network) is a consortium of 21 higher education institutions within New York State. Of these, currently only Cornell University has a connection to the vBNS (very high performance Backbone Network Service). NYSERNet has a total of seven universities that have submitted HPC (High Performance Connections) proposals to the NSF (National Science Foundation); of these Columbia, NYU and Syracuse University have funding. Rensselaer Polytechnic Institute, SUNY Buffalo and the University of Rochester have approval; however, they are awaiting funding. An additional 14 NYSERNet member institutions are not vBNS awardees. Of the 21 NYSERNet member institutions seven are Internet2 members and potential Abilene members.

There are institutions that are part of NYSERNet that cannot be awarded HPC grants nor become members of Internet2. NYSERNet must somehow provide connectivity to other campuses across the country for these institutions.

1.1 NYSERNet Statewide Topology

NYSERNet is building an OC-12 statewide network from Buffalo to New York City with OC-3 connections for each institution. NYSERNet has requested both backbone and local loop service providers to submit proposals for SONET (Synchronous Optical Network) services in order to implement an ATM (Asynchronous Transfer Mode) infrastructure statewide. In partnership with Newbridge Networks Corporation, NYSERNet is installing ATM-based 36170 switches in Albany, Buffalo, New York City, Rochester and Syracuse. Each institution will have the option of individually select-

ing an edge device that provides the best solution for their campus.

1.2 Edge Connectivity

Each of the institutions will still maintain their existing commodity T3 connection. The biggest issue the campuses face is the routing "fish problem."

The institutions have a few options for their edge device solutions: install an ATM card in the existing router, connect a switch to their existing router, or provide a separate router with an ATM card to their existing router. An engineering task force will be developed within NYSERNet and its member institutions to address many of the issues a campus will face. Another edge device solution that Newbridge offers the institutions is Carrier Scale Internetworking (CSI). CSI is an IP-over-ATM solution for those institutions that desire an IP-based edge solution. CSI is in the early development phases, and will be a major part of the NYSERNet 2000 network.

2. Policy and Quality of Service

Initially NYSERNet will provide QoS to each institution by means of the inherent levels of QoS built into ATM. NYSERNet will offer PVPs (Permanent Virtual Paths), PVCs (Permanent Virtual Circuits) and SVCs (Switched Virtual Circuits) to institutions for their connectivity across the state and to the vBNS.

NYSERNet plans on working in conjunction with the vBNS engineers and with Abilene project engineers to provide the appropriate QoS handoffs into each network. However, NYSERNet will offer all levels of ATM QoS within the state infrastructure to each of the institutions. Policy routing is an issue that NYSERNet must address at both the campus edge and when passing egress traffic to net-

works and gigaPoPs outside the NYSERNet 2000 network.

Most of the policy issues will be implemented on the campus edge. NYSERNet will enforce those policy issues at the New York City gigaPoP as egress traffic is handed off to the vBNS or Abilene.

Within the NYSERNet 2000 network, NYSERNet intends to build an environment free of AUPs (Acceptable Use Policies). Quality of service will be implemented on each campus by an administrator/network engineer and will be enforced within the backbone infrastructure and on the campus edge device as necessary.

Another issue that will be addressed in the future by NYSERNet is oversubscription and bandwidth reservations of the OC-3 connections to the vBNS and Abilene as well as the OC-12 backbone. It is the NYSERNet 2000 Engineering Work Group's initial thought that in the beginning of the NYSERNet 2000 network, QoS and bandwidth reservations may not be much of an issue due to the

amount of bandwidth available and the initial lack of available applications. However, the workgroup believes QoS and bandwidth reservations will be addressed within 12 months, and possibly sooner for the vBNS and Abilene connections.

3. Summary

It is NYSERNet's desire to work with each of the equipment vendors to offer the best and most recent developments in QoS in an evolving process.

NYSERNet plans to extend a network that provides high-speed packet forwarding with single-hop address resolution based on policy-based route selections. Standards development is a main priority for the NYSERNet 2000 Engineering Work Group. The workgroup will maintain a standards-based network solution that will work seamlessly with existing IP equipment, and still offer experimental and emerging technologies, both hardware and software, to each of the NYSERNet 2000 member institutions.



MINIMUM-COMPLEXITY QOS FOR THE ENTERPRISE

Terry Gray, *University of Washington*

1. Introduction

Like everyone else, the University of Washington seeks to provision the best possible network at the least possible cost and the highest possible reliability. Accordingly, our biggest concerns about implementing any QoS schemes are recurring costs and network reliability.

Dynamic (per-flow or “per call”) authentication, authorization, or reservation of bandwidth appear to have negative implications on recurring costs and network reliability. In particular, we are concerned about making the organization’s most important infrastructure asset — the ability to forward packets — dependent on authentication and authorization servers that are not now needed for per-packet (or per-flow) forwarding decisions. Not only will these servers add more capital costs to building the network, but in a world of finite bandwidth, they won’t necessarily guarantee traffic flows with a reliability sufficient to be sold as “premium” service to our campus customers.

Our challenge, then, is to devise a minimum-complexity scheme that will (a) shape peak demand to fit instantaneous capacity, and (b) generate revenue to keep aggregate capacity ahead of aggregate demand.

A pessimistic view of campus network requirements implies the need for per-flow controls. An optimistic view implies that over-provisioning alone will be sufficient. Our idea is a “HedgeMyBets” middle ground that eschews per-flow admission control, but allows for port subscriptions.

The specific approach we advocate is based on class-based queuing (CBQ) and eligibility control, which can take advantage of the

multiple queues in modern network devices, and does not require per-flow authentication and authorization at the campus level.

2. Enterprise QoS Considerations and Requirements

The central issue in designing a QoS mechanism is the cost of controls vs. the cost of resources vs. the value of certainty. Certainty comes into the equation because we are dealing with statistical usage patterns, complete with unexpected peaks; thus “guaranteed service whenever desired” in a bandwidth-sharing environment is a myth.

A reservation system might be able to guarantee a particular service level whenever the user doesn’t get a busy signal, but even with plenty of “spare” capacity, there is still some chance that a transient spike in demand will exceed the available capacity and thus result in busy signals. From the user’s perspective, the “circuit busy” signals associated with reservation-based QoS schemes are network failures. The only way to avoid the uncertainty of statistical multiplexing is to avoid sharing the bandwidth altogether.

Clearly network planners must balance aggregate demand and supply in order to shape peaks. Demand for network capacity increases as a function of number of users, their desire for better performance, the amount of their usage, and the needs of specific applications. On the other hand, cost, availability, and disappointment can all tend to reduce demand.

In our “QoS tool kit” we have several techniques for moderating demand:

- Traffic authorization (premium access demand control)
- Eligibility control
- Admission control

- Traffic modification (instantaneous demand control)
- Traffic shaping
- Traffic policing
- Traffic adaptation (feedback-based demand control)
- Protocol adaptation
- Application adaptation
- User adaptation (behavior shaping)

These approaches complement the ability to segregate traffic into distinct classes using multiple queues in switches and routers. The supply and demand equation applies to each class of service supported by the network, but is complicated by the possibility of high-priority traffic being downgraded to “best effort” when there is insufficient high-priority capacity. The good news is that modern switches can provide very low latency forwarding to high-priority traffic even in the face of large spikes in the best-effort traffic load.

The enterprise network is made up of three kinds of congestion zones:

1. Building subnets, which, when provisioned with contemporary 10/100 and Gigabit Ethernet switches, present a low probability of congestion
2. Campus backbone, where congestion is somewhat more probable, though perhaps still unlikely if Gigabit Ethernet routers are used
3. Wide-area network, where congestion, at least on some links, is likely and where a variety of demand-control mechanisms may be needed

In addition to the primary campus backbone infrastructure, network administrators may decide that it is cost-effective to deploy one or more special-purpose backbones for specific communication requirements. An example would be a tele-immersion experiment where only a few locations are involved, but extremely high bandwidth is required. When such applications become commonplace, we would expect to move that traffic load to the

(suitably upgraded) primary network infrastructure.

The goal, then, is to provide a campus network infrastructure that offers a low probability of congestion on campus, takes advantage of the multiple classes of service available in current products, allows a variety of policies for setting packet priorities, and permits end-systems to interface with potentially more complex QoS mechanisms at the campus border routers for wide-area services. All this happens without per-packet or per-flow lookups or reservations (within the campus).

3. Strategies

Our strategy for achieving this goal is to deploy a class-of-service infrastructure amenable to several different service policy models, e.g., charging per-port subscription and/or usage fees and/or support for differential queuing based on application need, especially delay sensitivity. Our model also allows for end-systems to signal campus border routers acting as bandwidth brokers (e.g., via RSVP) if necessary to negotiate for wide-area premium service, and for “very long term” reservations (i.e., segregated bandwidth) among enterprise sites for IP telephony, IP videoconferencing, etc.

The specific approach we advocate is based on class-based queuing (CBQ) and eligibility control. In order to take advantage of the multiple queues in modern network devices, we need some criteria for differentiating the priority of different packets. For maximum flexibility, we want a scheme that will allow prioritization based on three different kinds of eligibility: (1) need, (2) desire, and (3) privilege.

Eligibility control based on (global) knowledge of an application’s needs can use the TCP or UDP port numbers in the packet header. Unfortunately, this isn’t always a reliable discriminator because the port numbers might be obscured by IPSEC encryption, or the application might dynamically negotiate the data ports. Even more funda-

mentally, the actual need of an application is dependent on circumstances of use as much as on technical parameters. Differentiating packet priority based on user or application desire assumes that the Type of Service bits in each packet header will be set (hopefully in accordance with the pending IETF DiffServ working group recommendations). Because there is no technical barrier to an end-system marking all packets as deserving high-priority treatment, we might want to combine the need and desire differentiators with some notion of user/port privilege. One way to do this is via physical port subscription levels, which could be reflected in the static configurations of edge switches, and which would correlate to a particular Committed Access Rate (CAR) quota that can be enforced by policing in the core routers.

Noticeable by its absence from this model is any talk of “admission control,” which is essentially network busy signals at application initiation or “call setup” time. Our contention is that admission control is needed if — and only if — eligibility controls prove inadequate.

A pessimistic view of campus network requirements implies the need for such per-flow controls. In contrast, an optimistic view implies that over-provisioning alone will be sufficient. Our idea is a “HedgeMyBets” middle ground that eschews per-flow admission control, but allows for port subscriptions (in case eligibility by port numbers and/or ToS bits proves inadequate).

Wide-area connections at the campus border could include commercial or commodity Internet connections (probably with “premium” service options), branch site connections

that will multiplex different types of services over the same channel, Internet2 connections, and possibly connections to regional consortia and federal mission networks. The nature of the QoS options available on these wide-area links will be reflected back to the campus, at least to the border router.

4. Conclusions

Our discussion leaves many unanswered questions. Will premium commercial Internet offerings necessitate post-hoc usage recharges? What do we do with incoming packets marked for high-priority treatment? Is a physical port subscription model necessary or desirable? And, above all, how much bandwidth and QoS will really be needed?

Nevertheless, adequate enterprise QoS without per-flow lookups or reservations appears plausible. Fast/Gigabit Ethernet infrastructure can reduce the odds of congestion; multiple queues and CAR policing provide additional headroom.

No matter its viability, one should bear in mind that the simplest QoS system will have enormous operational impacts. Most campus networks are not ready for QoS even in its simplest form. Campuses should brace themselves for forklift upgrades of both network infrastructure and end-systems, no matter their approach to QoS. Shared LANs must give way to switched full-duplex links and multiple class-of-service queues, and desktop operating systems must evolve to have lower latency for delay-sensitive network applications, even under this minimalist scheme. Likewise, formulating QoS pricing and policies presents a daunting task for campus administrators. In this environment, complexity is the enemy of success.

CROSS-SECTIONAL BREAKOUT SESSIONS

Thursday, May 21, 1998

To stimulate interdisciplinary dialogue among different types of conference participants, and in order to generate a set of problems and discussion points for the following day's specialist sessions, we convened four "cross-sectional" discussion groups, each with a random mix of workshop participants. Each group was identified by a color.

As might be expected from such a diverse group, the questions and discussions were wide-ranging, addressing every aspect of QoS from policy-making to the specifics of implementation. These afternoon discussions marked the start of a rich cross-fertilization of ideas stemming from a wide range of concerns, and helped to identify key issues for Internet2's QoS effort to address. What follows are the main points made in each of these discussions.

YELLOW GROUP

DiffServ and Multicast

Does DiffServ conflict with multicast? Or, why is multicast hard? Multicast assumes that the sender doesn't know where the packets will end up, so the sender wouldn't have control over, and wouldn't want to take responsibility for, resource usage. But it appears that the differentiated services model is sender-oriented. Does DiffServ require that the sender pay for resource utilization? This would be a problem. On the other hand, if each domain has only one interface to the network, it would be less of a problem. What are the implications of this conflict for Internet2?

DiffServ and Campus Economies

Is the DiffServ model inherently in conflict with typical campus "economies"? What is

a typical campus economy? Some universities are more capitalist, some are more socialist — others are practically feudal! Most are all three in varying mixtures. But certainly QoS and DiffServ are more capitalistic than what most of us are used to. The question of how to allocate resources among people is orthogonal to the technical question of how you manage network resources to create new service. If you wanted to be completely egalitarian, you could give everyone a ration of one kilobit of Premium service! On the other hand, a university that wants to retain more of a socialist regime can spread the costs of higher quality service differently than one which more wholeheartedly embraces capitalism. This is more an administrative question than a technical one. Are we talking about privileged people or privileged applications?

The Bandwidth Broker

How will "bilateral agreements" be structured? Are there models appropriate to QoS? Is there an operational bandwidth broker implementation? Is it testable? We need a prototype BB with a few prototype profiles. Who will build them? Vendors? In Phase Zero, BBs could be humans rather than software. If you use port subscriptions — as Gray and Klingenstein suggest — then what is the role of a broker? Whether we need a broker may depend on the type of application, and its level of resource utilization, and the cloud-by-cloud path taken by the flow.

Bandwidth and Latency

Why should we care about latency at all? Isn't bandwidth what most users are concerned about? Yes, but latency is critical for some things, like VR. It all depends on the application. Also, the two can be related within a single application. We know of at least one application whose bandwidth

needs scale geometrically with linear increases in latency. In order to find out what its own bandwidth needs are, the application may need to know what latency rates are available.

DiffServ and ATM

What about ATM? Is there a proposed or actual interface between ATM and DiffServ? In particular, how is DiffServ related to differentiated QoS at layer 2? How can DiffServ be extended across pure ATM exchange points like the Chicago AADS switch? What is the relationship between Multi-Protocol Label Switching (MPLS) and DiffServ?

Why GigaPoPs?

Are gigaPoPs “white elephants”? Big network service providers appear to be moving away from such points of aggregation, so why are we moving toward them? When gigaPoPs are connected to backbone nets and Acceptable Use Policies are used, routers must decide which backbone can be used for each packet. Sometimes they use source-destination routing to make the determination. Why aggregate traffic on the way to a gigaPoP just to separate it later?

Where to Begin?

Where should campuses and gigaPoP planners begin to devote their efforts? Should we start developing prototype bandwidth brokers? Invest in and deploy two-bit capable equipment? Begin intercampus testing? What is the number of campuses that would be right for a testbed? The number of applications? We are probably committed to something, since the Internet2 effort is almost by definition QoS. It would be best to put our eggs in several baskets. We can't just pick one approach and run — we have to choose two or three and run. We need to allow for various simultaneous investigations. But we also need to talk more about these plans before implementing them.

Encouraging Socially Responsible Applications

Do application developers understand the DiffServ approach, and will it meet their needs? Clark's model, known as Assured service, provides a “floor” service, whereas the current DiffServ model, known as Premium, suggests that we go with a peak or “ceiling” service. Which applications would benefit from which services? Assured service works better for adaptive applications than for those which assume a constant bit rate. This could be a good reason for emphasizing it over Premium service. As adaptive applications are both possible and more socially responsible, we should try to encourage them instead of building a network to accommodate nonadaptive applications. To what extent can DiffServ be augmented by methods that will encourage responsible use of resources? Since one target net might be best effort, while another one might be QoS, does the “fish problem” hold for QoS just as it does for routing?

RED GROUP

Where does QoS Matter?

Where in the Internet2 architecture is QoS most important? Host-to-campus connectivity is probably of no concern. However, as the system complexity gets concentrated in the edge routers, there is the risk that a host with a high speed interface could overwhelm the edge router resources. Are denial of service attacks a bigger problem with QoS? QoS is most important at the interconnect points, and in particular the link between the campus and gigaPoP is likely to be the most congested link of the architecture. Congestion in the backbone could also become a problem as demand grows.

Different Solutions for Different Areas

What QoS solutions are best suited to the Internet2 architecture? There will be different solutions in different parts of the network. For instance, Lixia Zhang described

how per-hop RSVP-style reservations could be used at the campus layer, while the backbone deploys a solution based on aggregation of traffic flows. Another issue is what to do with incoming traffic. Should the receiver always respect the priority level specified by the sender? How do we resolve conflicting requests? In particular, what do we do with flows currently in progress when a higher priority flow requests access? Do we allow preemption?

Premium Service

Does the current Van Jacobson definition for Premium service waste bandwidth? It seems like Premium would make it hard to take advantage of excess capacity. Won't there be billing and accounting problems as a result of the BB not keeping state information on a per flow basis? How much can we deviate from this position to include some per flow information? What is the right mix? Which flows require what level of information?

Chickens and Eggs

The question of what the applications need versus what the network can offer is a kind of chicken-and-egg problem: it is not clear which comes first. Is it sufficient for the application to probe the network for the QoS level offered, which would require an adaptive application? Or is the network to expect a QoS initialization from the application? What level of guarantee will an application be able to expect from the network? Will it be relative or absolute?

GREEN GROUP

Scalability and Measurability

Scalability and measurability are two of the primary architectural goals for Internet2 QoS. The DiffServ approach of marking packets at the edge in order to produce aggregated flows at the core addresses the need for scalability, while the use of profiles to specify end-to-end service levels addresses the goal of measurability. Per-hop behav-

iors and profile meters need to be combined to form services.

Keeping Best Effort From Getting Worse

QoS will probably be used mainly for large data flows and WAN links. Best effort service remains important and should not be starved by the new services. How do we prevent starvation of best-effort traffic? Will there be two separate networks? On the campus, simplicity is paramount. In that context it may be best to use overprovisioning, or, for higher quality services, port-based subscriptions. What is the right level of busy signals? What is a minimal implementation of profile-based DiffServ for Internet2 that gives a reasonable service? Will there be a strict separation of QoS and best-effort traffic at the start? What is the minimum testbed we need to answer all these questions? What is the timeframe for setting up such a testbed?

User Issues

Security and charging are issues. There will also be operating systems issues on hosts. How do we manage user expectations? What is the user's source of policy information? Are the local network managers responsible for providing this information? How do users view QoS and how is it provided to them? How do they invoke it? How do they know that they have it? How is it charged back to them? What service guarantees can the host expect?

How Will the BB Work?

For end-to-end service, what is the role of the bandwidth broker? How can resources be allocated across multiple domains? What is the BB interchange? What information must the BB send to routers? What information is needed by the BB to grant requests? Where does that information come from — users, network managers, adjacent networks? Who sets the ToS bits — the host or the network? Does the BB consider parameters other than bandwidth? Which ones?

Setting Up Connections

There will probably be signaling issues, and the RSVP protocol may or may not play a role. How can we tunnel RSVP through non-DiffServ domains? How do queuing strategies interact? There will be issues of how to allocate QoS in each direction, and whether QoS is initiated by the sender or by the receiver. How does the application specify its requirements to the network? DiffServ's per-hop behaviors furnish basic QoS building blocks, but much work remains in building services from these blocks and figuring out how to offer these services to users. In particular, what is the role of ATM in profile-based DiffServ?

BLUE GROUP

Setting Up a Testbed

What will be the demand for QoS and how should a testbed be designed to meet real-world loads? We won't know the QoS demand until the services exist. Where is the QoS need coming from? Will demand be for a small number of high-bandwidth flows or a large number of low-bandwidth flows? Will it be more a matter of a few people using telecubicles, or of many people using desktop video? What applications need low latency? An application with synchronous video and data streams may need it. When do you need QoS? Whenever best effort doesn't do the job.

How Do We Decide Who Gets What?

For what is QoS being reserved? High-end or low-end applications? How involved is the application in the QoS negotiation? Are DiffServ PHBs like a hall pass in school to go to the bathroom? If so, then the bandwidth broker has to decide who has to go the worst.

How to Test?

How will we know if QoS is working? What management system will we use to test QoS,

other than just looking at application response? The test traffic will have to come from willing subjects who know what they are getting into. For study we will need records of reservations and real flows even if they are aggregated upon entering a cloud. Another testing issue is that if all groups on one campus use Internet2, but only some groups on another use it, we will have asymmetric routes. This can make evaluation of QoS methods hard. Also, how will testing affect production networks? Perhaps we should set up a separate research net for testing to answer these questions. vBNS is using this approach of having separate production and testing networks.

Will QoS Discourage Network Use?

From the campus viewpoint, will QoS be a disincentive for network use? Will it be a pain to use? Setting up connections could become a big hassle. People may not use it if they get charged for each use. This makes it important that best effort coexist peacefully with QoS. You don't need to charge for best effort. Only those that need a better grade of service will be charged. At a finer level of detail, there is the issue of asymmetric data flows. For example, one click on a Web page can generate many megabytes of return traffic. One might only need priority service in the return direction. QoS reservations need to be available for each direction independently. What is most of the traffic going to be on the production network? Best effort or QoS? Probably best effort.

Capacity

How much of the aggregate capacity across a cloud can be provisioned for QoS? What if there are 10,000 flows and 10 percent need QoS? Can that be handled? Experience will tell. What about the problem of multiple sources sending data to a single destination, which then ends up oversubscribed? This is a hard question. One solution is to limit the total QoS traffic from all inputs to less than the smallest output pipe capacity. In this case, even if all inputs send their full QoS

allocation to a single output, it will still not be oversubscribed. This approach entails using clouds with a small number of inputs, which in turn would mean you would probably need more clouds. Does there need to be a bandwidth broker per region of congestion? How will bandwidth broker admission control decisions take into consideration the possibility of route flaps? Do we need to pin routes to make DiffServ work?

Getting Started

DiffServ seems like a promising approach, especially because it can be implemented in a reasonable timeframe, but how confi-

dent are we that it will really work? How can you manage the resources when crossing several network domains? How can you coordinate across clouds? Start with static provisions. Take the word of a cloud that it will provide what it says it will. How can this work with even with one cloud? Overprovision the cloud. What is the granularity of the communications between clouds? Between clouds it is the bandwidth of the pipe separated into the different classes of service. Individual flows are not accounted for. How does bandwidth broker to bandwidth broker signaling guarantee QoS? Set a level for QoS. Try it and see if it works.



SPECIALIST BREAKOUT SESSIONS

Friday, May 22, 1998

For Friday's discussion sessions, the conference participants divided into groups according to area of expertise: Host/Application, Campus, GigaPoP, and Backbone. Each group used the set of questions generated from the previous day's discussions as a starting point and worked to resolve, or at least clarify, the concerns raised. What follows are synopses of these discussions provided by volunteer recorders.

Host/Application

1. We don't need more than a few service types.

For a useful set of applications that we specify, can we also specify a compact set of QoS models to cover it? It seems that we could meet the needs of Internet2 applications with as few as four types of service: traditional best effort; class of service, a.k.a. relative QoS; Assured, à la John Wroclawski; and Premium, à la Van Jacobson.

2. We do need a QoS accounting mechanism.

What economic incentives currently or might in future exist for campus development of QoS applications? What are the cost recovery models? The accounting requirements? Many implementations will probably be driven by AUP or campus policies rather than by strictly economic concerns.

3. We must find a way to spread the word to application developers.

How can we encourage greater awareness and deployment of Internet2 QoS-sensitive applications? QoS-using "applications in the commons," as Ken Klingenstein discussed, could not only provide a large base for testing but could also help to create a larger market for DiffServ. How can we introduce QoS to commercial, off-the-shelf applica-

tions? It would be useful to develop a suite of tools that would make it possible for a network administrator to view the status of QoS resources and adjust them accordingly. What do application developers need from the network in order to be able to build useful applications?

4. A number of questions remain about how the proposed QoS API would work.

What about scheduling? Should it be part of the QoS API, or should it be a separate network service? Studying the Session Description Protocol could provide some insight here. The applications working group should also look at current products to see what support exists for server-based QoS applications, time-dependent applications, and scheduling aggregated flows. How would an API support multicast applications? What feedback mechanisms would exist to ensure that the service requested is the service delivered? What naming standards will or should be applied to profiles? How would an API support use of profiles created on the fly or provided dynamically by a server?

Campus

1. We need incremental deployment.

One of our main concerns needs to be finding a workable incremental approach to implementing QoS. It is not clear which real research and educational applications will fall into which traffic classes, and it seems unlikely that figuring this out will be a tidy process. It may be difficult to get it right for any given application. It will be even more difficult to market and sell QoS services on our campuses. How are we going to get end users to comprehend all this and understand that they are getting something for their money? Part of the problem is that most of us have only a loose feel for the traffic patterns on our campuses. We are not really sure what is out there.

2. We should start with CoS.

There are disadvantages to starting out with a simple CoS approach. With CoS it becomes difficult to explain to a user what they are actually getting, since the service is relative. How can we as network managers prove to the customer that they are getting their special treatment? Their service may be degraded, just not as badly as everyone else's. We would really be offering insurance rather than assurance. Marketing CoS to our users is not something the majority of us are looking forward to. If we can't explain it, we can't sell it. Another problem is that if we offer both CoS and Premium as subscription services, they might be indistinguishable to our customers. This would be confusing and difficult to explain. On the other hand, many boxes will be shipping CoS-enabled, so shouldn't we take advantage of this? CoS is something we network engineers could use as a tool behind the scenes, rather than making it into a service to which our customers subscribe. This way, we get to make all the decisions and keep the locus of control central — and we like that. Floor-based RIO services are fine for applications like FTP, and if Premium is available for more demanding applications then we should have what we need.

3. Pricing QoS accurately will be hard.

We need to be careful in assigning price and QoS levels, because all of this can only work if we moderate demand for QoS services. Once again, implementing QoS does not create bandwidth. Pricing will shape behavior, but it's impossible to predict the results exactly. It's certainly unclear how well pricing will go over, and what users will be willing to pay for. The question remains whether it should be the sender or the receiver who pays. It seems to depend on the application in question. Of course customers don't want to be charged for spam that someone else decided to send them; they want to be charged only for what they requested.

4. QoS privilege should be per-user at first.

One of yesterday's questions was: Do we have privileged applications or privileged

users? Assuming just privileged users may be the first step, evolving to a combination of a privileged set of users, using particular privileged applications, at a particular time. On the other hand, it may be easier to let the application determine the privileged use at the beginning, as authentication of users remains a relatively difficult problem for many. QoS by application may require that universities use the same applications.

5. It will be hard to prove to QoS buyers that they are getting what they are paying for.

Assuming we do offer a QoS service to certain end users for particular applications, how then will we know if an application's needs are truly being met? We would like to have a NetView-like service to see whether the profiles are being met and the bits are getting through, but the rest is subjective. Will the application itself know if its QoS needs are being met? Such QoS-aware applications will need to be developed. There is also a need for an easy way for end users to monitor the QoS treatment their applications are receiving. It would be helpful if both the level of network performance and the level of application performance could be made clear to the user, as application performance can be greatly affected by a large number of network-independent factors.

Monitoring and auditing end-to-end performance may mean exchanging trouble tickets across administrative boundaries. Checking end-to-end performance would make sense. Large corporations already check end-to-end performance of their critical applications. What level of granularity will be required for auditing? For Phase Zero we may not need to worry about this, but again we need to know if work on this problem is already underway.

6. We only need authentication at the boundaries.

If pair-wise boundary agreements are negotiated, we only need authentication at the boundaries, and authorization can remain local. Eventually we will need a way to pass

along an end-to-end guarantee, but this is less important at this early stage of the game. The reason for authentication is purely to get packets to the right classifier. Authentication can evolve from none to pair-wise to end-to-end. Eventually, the bandwidth brokers will need to be talking to each other across administrative domains. Has work begun on a protocol for this?

7. Dedicated switched networking needs to be ubiquitous.

To move from relative to absolute service guarantees, switched full-duplex connections to individual stations will be needed — otherwise jitter will be a problem.

8. We would like some clarification on aggregating premium flows.

9. We would like to understand where the working group is headed on a bandwidth broker protocol.

10. For bilateral agreements, we really need a standard template that protects best-effort traffic.

11. Let's start small.

To get started, perhaps we can commission three or so universities on a common backbone using common applications to experiment with just a few of the easier cases. All of this is going to take a while. For now, we need to work on underprovisioning expectations and overprovisioning bandwidth. Again, we need to approach this incrementally, a few limited-function betas at a time. Even after some beta testing, we will not be able to implement this all at once on our campuses. How many of us can afford forklift upgrades? Will QoS overload our routers? Are we willing to deal with new, unstable, and buggy software running our network cores? We need to ensure that when things go wrong there is orderly decay, not catastrophic failure.

GigaPoP

1. We need to learn a lot more about QoS.

Many of the technologies raised at this workshop have poorly-understood but clearly large implications for gigaPoPs. We need to combine prodigious research into these possible implications with modesty in our initial testbed choices.

2. QoS is most important at network boundaries.

But, where are these boundaries in our networks? The structure of “bilateral agreements” was also unclear. Are there example models of how bilateral QoS agreements would work?

3. We need a better understanding of the role of ATM in the DiffServ model.

How does DiffServ (a layer 3 QoS solution) compare to QoS at layer 2 (i.e. IEEE 802.1p, Frame Relay CIR)? What is the relationship between MPLS and DiffServ?

4. We need prototypes of bandwidth brokers.

Do prototype implementations of bandwidth brokers (BBs) currently exist? If not, who will build them? Are the vendors working on BBs? In Phase Zero, BBs could be humans rather than software. With subscriptions à la Gray/Klingenstein, what is the role of a broker?

5. QoS may exacerbate the “fish problem.”

Are gigaPoPs “white elephants”? Big network service providers appear to be moving away from AUPs, so why are we moving toward them? As source-destination routing could go away were it not for the AUPs (restricting traffic over backbone nets, e.g. vBNS), why aggregate traffic on the way to a gigaPoP, just to separate it later? What is this problem vis-a-vis QoS? Because one target net might be best effort while another might be QoS, the fish problem seems to hold for QoS just as for routing.

6. Where should gigaPoP planners begin to devote their efforts?

Some possibilities: (a) develop/experiment with prototype BBs, (b) invest in and deploy DiffServ-capable equipment, (c) begin inter-campus testing.

Backbone

1. What policy issues need to be dealt with at the backbone level?

Most policy issues can be pushed to the edges if the backbone BBs can completely trust the campuses (a big if, of course). Whether QoS appears to the user as socialist or capitalist depends not on the technology chosen to implement it, but on the policies of local network administrators. Because users need to figure out what they are getting, DiffServ must be auditable. Backbones will have to audit their downstream SLAs for compliance.

2. What QoS demand are we designing for?

What load do we expect to see in the backbone? Are we designing QoS for under-loaded or overloaded cases? We want a solution that works at heavy loads, not just light ones. Will we see a large number of small flows, or a small number of large ones? The answer is “yes” — both. Applications folks want per-flow guaranteed services.

Also, QoS can range from “better than thou” service upward to absolute services. Are we talking about implementing only class-based service, only flow-based service, or both? More broadly, our choices are: relative or non-relative (absolute), flow-based or aggregate-based, and which parameters to use — bandwidth, latency, jitter, etc. All these are tied to how well the network is provisioned.

3. The strength of DiffServ depends on the strength of admission control.

DiffServ started out as a class-based service, with only relative guarantees. However, Van Jacobson claims to be able to provide a non-probabilistic service. Can this really be accomplished? The admission control algorithms that he described are exceedingly

conservative, guaranteeing that even if all accepted Premium traffic went to a single egress port, the port would still not be over-committed. Less conservative means more complex, and less complex means more conservative. Jacobson’s recursive definition of “cloud” allows him to be less conservative in allocation, but at the price of increased complexity.

For OC-1 speeds and above, routers must forward at line speed, so all induced delay is in queuing. In Jacobson’s model, delay is handled by having different queues and managing them carefully. But this is still based on PHBs, so there is a cumulative effect. For example, route flaps may cause violations of contracted bounds on jitter or delay, even though the PHB is fine at each hop. Jacobson and others think that by using very conservative policies, they can squeeze the queuing delay out of DiffServ. But in the real Internet there are no guarantees. Getting DiffServ to deliver end-to-end bounds on delay will be very hard. Internet2, however, has a more hierarchical topology than the wider Internet and may present an opportunity to subvert some of the complexity of these routing issues.

4. RSVP is here now, so perhaps we should think of it as the quickest way to get something out.

Users will need a range of QoS service offerings. We also have to look at time frames and work out where we are going to start and how we want our projects to evolve over time. Is DiffServ really as simple to do as has been described? To neglect the per-flow accounting is simplistic. The vBNS is basing its initial QoS offering on RSVP signaling, bandwidth reservations and so on, and is planning to migrate to DiffServ support later. Those with high bandwidth will be few in number but of high enough value that we can afford to do the setup needed for RSVP. Others will be able to use DiffServ as it evolves and, as QoS demand grows, exploit the scalability of DiffServ.

CONCLUSIONS AND NEXT STEPS

As university researchers and educators increasingly turn to demanding networked applications, the need for end-to-end QoS grows increasingly urgent. The goals of this workshop were to bring together a diverse set of professionals to examine the application and network engineering QoS requirements of the Internet2 project and to evaluate a family of approaches based on the evolving differentiated services (DiffServ) architectural framework. Specifically, Internet2 engineering put forth the recommendations of the Internet2 QoS Working Group that the project pursue incremental testbed deployment of key DiffServ architectural components and explore several proposed services that offer applications absolute assurances based on quantifiable service profiles.

This report has attempted to summarize the requirements for Internet2 QoS and present the case for the DiffServ QoS architecture. The pre-workshop white papers “Requirements for Internet2 QoS” and “Differentiated Services for Internet2” have been refined and consolidated into the introductory chapter. The body of the report consists of extended abstracts submitted by the presenters, and synopses of the discussions that occurred during the breakout sessions. This final section attempts to summarize the output of the workshop, draw conclusions about the consensus of the community, and outline future directions and concrete next steps for Internet2 QoS.

APPLICATIONS

In the talks by Lanier and Grossman, we have seen examples of advanced applications that promise to change the way that university students, faculty, and staff work and learn in the future. This promise can only be realized, however, if these applications are able to get the kinds of QoS assur-

ances that they need from the network. Tele-immersion and interactive data mining are examples of applications that have QoS requirements based on hard thresholds of human perceptual sensitivity and which are intolerant of even marginally insufficient network performance. To meet the needs of these most demanding applications, strong forms of QoS are needed that can ensure that applications will receive the (non-relative) network performance levels that they require.

However, not all applications are as intolerant as these. A range of other application QoS requirements exists that includes QoS-needy applications that are tolerant and QoS-needy applications that still want to make use of adaptive techniques. There is even a need for relative classes of service that can give precedence to certain flows or flow aggregates without providing absolute assurances.

Discussion in the Host/Application breakout session addressed the question of whether a set of four services proposed under the DiffServ framework would span the space of Internet2 application requirements. The set of services discussed was:

1. Best Effort
2. CoS (Class of Service or “relative” QoS)
3. Assured (from Wroclawski’s presentation)
4. Premium (from Jacobson’s presentation)

The consensus of the group was that this set would, in fact, meet the vast majority of application QoS requirements, and that the greater challenge for the applications community was to focus on the middleware that would enable developers to exploit these newly emerging network services.

A number of participants in the Backbone breakout session felt that Internet2 should

pursue a range of QoS service offerings that would include “hard”, per-flow, hop-by-hop reservations, as well as DiffServ. The concern was that there will continue to exist a small number of high-end users whose traffic has high enough bandwidth requirements and high enough value to justify manually-provisioned or RSVP-signaled reservations, and that these users may not be able to tolerate the unreliability of early experimental DiffServ offerings.

DIFFERENTIATED SERVICES

Jacobson presented an overview of the DiffServ architecture, introduced the abstraction of a bandwidth broker to manage internal network resources and perform admissions control for each administrative domain, and described a proposal for a particular DiffServ service dubbed “Premium”. The DiffServ architectural framework represents the best current thinking on how to engineer highly scalable and interoperable QoS.

DiffServ reduces the state requirements of core routers by careful aggregation of QoS-enabled flows. The aggregates are given a small number of simple differentiated forwarding treatments indicated by bit settings in the packet headers. A broad and flexible range of services is provided by protecting access to the aggregate treatments with per-flow policing at the network periphery and aggregate policing at transit network ingress points.

The differentiated treatments (dubbed “per-hop behaviors” or PHBs) suggest, but do not imply, particular queuing disciplines and consequent services. Proposed examples include default (best effort) forwarding, expedited forwarding (“forward me first”), and drop preference (“drop me last”). Each DiffServ flow is policed and marked at the first trusted router, according to a contracted service profile. Downstream from this leaf router flows are aggregated. All subsequent forwarding and policing is performed on aggregates.

Handling traffic aggregates rather than individual QoS-enabled flows makes it easier to construct end-to-end services by concatenating multiple cloud-to-cloud services. Individual network clouds contract with neighboring clouds to provide differentiated service contracts for different traffic aggregates. Like the per-flow contracts, aggregate contracts are characterized by profiles, which are enforced at cloud-cloud boundaries. This model results in a set of simple bilateral service agreements that mimics current interprovider exchange agreements.

Finally, in order to make appropriate admission control decisions in an end-to-end call setup and to configure local leaf and edge device policers correctly, each cloud has a bandwidth broker. A host signals its local bandwidth broker to initiate a connection and the user is authenticated. The user is subject to local policy-based admission control decisions and resource accounting. Then, on behalf of the requesting user, the local bandwidth broker initiates an end-to-end call setup along the chain of bandwidth brokers representing the clouds to be traversed by the application flow. The bandwidth broker abstraction is centrally important. It allows separately administered network clouds (possibly implemented with very different underlying technologies and with very different policy constraints) to manage their network resources as they see fit.

HOSTS AND OPERATING SYSTEMS

End-to-end QoS does not just mean quality of service from network interface to network interface — it must mean quality of service from eyeball to eyeball. Hence, QoS is not only a networking problem, but is also an operating system problem. Brandt presented an example of ongoing research into middleware that can negotiate with the operating system on behalf of QoS-needy applications in order to obtain the operating system resources (CPU, memory, and buffer space) that the application requires. Operating system resource management is but one piece

of end-host functionality required to make production QoS a reality. End hosts will also need to provide appropriate APIs and middleware for signaling QoS requests and will need client-side support for authentication and billing.

RELATIONSHIP BETWEEN DIFFSERV AND INTSERV

Behind the bandwidth broker abstraction are real network resources that must be reserved during QoS call setup. Zhang discussed the relationship between RSVP and DiffServ, and described several approaches that leverage the significant existing body of work in the RSVP/Integrated Services (IntServ) framework to manage network resources in a DiffServ context. One approach is to use classical RSVP techniques to establish per-flow resource reservations in peripheral networks, to tunnel the RSVP signaling messages through core DiffServ networks, and to map between RSVP flows and DiffServ aggregates at the IntServ/DiffServ boundaries.

A second approach is to use a “lite” version of RSVP signaling within a core DiffServ network to reserve router resources on a hop-by-hop basis through the network from ingress router to egress router. In this approach, individual calls through a DiffServ cloud trigger an admissions control procedure that requires a hop-by-hop evaluation of the availability of the requested network resources, but that does not require core routers to maintain per-flow state.

TYPES OF SERVICE

Premium service offers leased-line emulation at a contracted bandwidth with minimal loss, jitter, and queuing delay. Premium service is accomplished by coloring every Premium packet with an expedited forwarding (EF) PHB, which is equivalent to strict priority queuing at every network node. Furthermore, to implement Premium the call admissions policy must take care never to oversell the EF capacity of any router.

Premium service is appropriate for the most intolerant applications and may have even broader appeal due to the simplicity and elegance of its service contract. However, not all applications are so intolerant or require such hard QoS assurances.

Wroclawski discussed research into DiffServ services that are inherently predictive or that would offer application-oriented assurances. The latter are characterized by high-level parameters like actual TCP throughput, rather than lower-level network transmission parameters like bandwidth, loss, and latency, which application designers may not be able to translate easily into expected end-to-end performance levels.

Additionally, Wroclawski discussed the need for services that would allow adaptive application techniques to continue to play an important role even in the presence of QoS. QoS-needy adaptive applications require a minimum service level, but are able to exploit any extra available network capacity. Consequently, adaptive applications require services that offer a “best effort with floor” assurance. In contrast, Premium service offers both a hard floor and a hard ceiling, thereby denying Premium flows access to any bandwidth in excess of that provided in their contract.

ADMINISTRABILITY

Not surprisingly, the bulk of the cost and complexity of deploying QoS within Internet2 will be borne by the campuses. In their talks, Gray and Klingenstein expressed concern about the administrative complexity of highly dynamic QoS call setups. They presented a proposal for minimizing the complexity of deploying and administering DiffServ on a typical switched LAN campus infrastructure. Their approach seeks to avoid the need for per-call authentication, authorization, and reservation of resources by selling per-port subscription levels to users and approximating a Premium service through a combination of over-provisioning and IEEE 802.1p-style class-based queuing in the LAN

switches. The Campus breakout session discussions confirmed that there will be a diversity of approaches to implementing QoS at the campus level. Some campuses may want to experiment with commercial CoS techniques while waiting for DiffServ QoS to mature.

As has been remarked earlier, QoS is not merely a technical problem, but also has deep economic and policy implications. Gray and Klingenstein raised a number of economic and policy issues that will inevitably need to be addressed by campuses as they strive to migrate toward production QoS service. They also made an explicit and successful effort to begin serious dialogue on these issues among Internet2 campus network administrators.

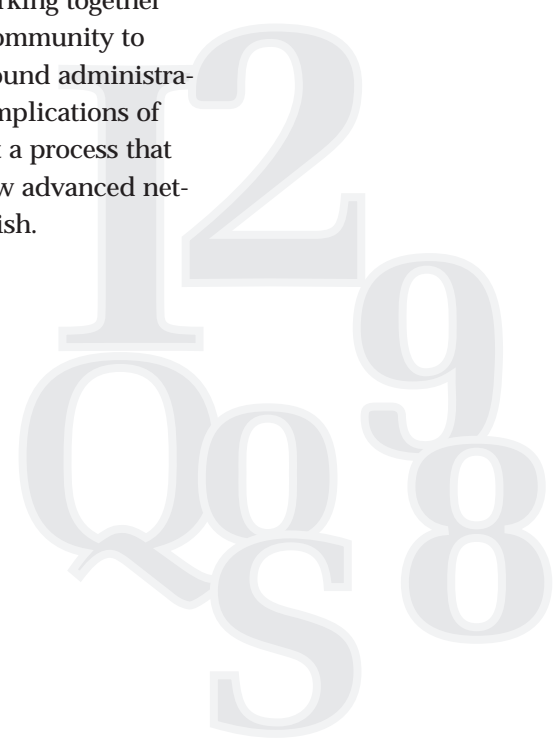
CONCLUSIONS

The First Internet2 Joint Applications/Engineering QoS Workshop represented a remarkable convergence of network engineers, advanced applications developers, CIOs, and members of the networking research community. The attendees examined the requirements for Internet2 QoS and critiqued the evolving differentiated services QoS architecture with respect to those requirements. One result was a growth in shared understanding of the QoS needs of advanced applications and of the technical and administrative ramifications of deploying QoS in a complex heterogeneous environment like Internet2. A second important result was the emergence of a rough consen-

sus around pursuing DiffServ and, in particular, the family of DiffServ services that offer applications absolute per-flow performance assurances.

To accelerate the evolution of viable interdomain QoS, Internet2 is launching the “QBone” — an end-to-end QoS testbed. The QBone will bring a dedicated group of Internet2 and affiliated research and education networks together with select applications and middleware development teams to deploy, evaluate, and refine new IP network services. Initial emphasis will be on deploying key DiffServ functional components and experimenting with one or two proposed DiffServ services.

The QBone will include a set of contiguous networks to implement the correct packet forwarding behaviors, network engineering groups to develop missing architectural components and tools, and advanced applications developers and users to develop or port applications that demonstrate and stress-test new QBone services. It is expected that the QBone will generate a body of invaluable hands-on experience that will inform the standards processes, as well as relevant networking research and corporate R&D efforts. Finally, by helping to evolve scalable and interoperable QoS technologies and by working together with the broader Internet2 community to come to terms with the profound administrative, economic, and policy implications of QoS, the QBone aims to start a process that will open the horizon for new advanced networked applications to flourish.



APPENDIX ONE: USING DIFFSERV PREMIUM SERVICE TO PROVIDE INTERNET2 QOS

Kathleen M. Nichols, *Bay Networks, Inc.*

ABSTRACT

This paper presents a brief analysis of how the IP differentiated services end-to-end service known as “Premium service” [1,2,3] can be used to meet the quality-of-service (QoS) needs of Internet2 [4].

Differentiated services (DiffServ) offers the best means to widespread early deployment of IP QoS and has been designed to offer QoS where the policy decisions and QoS allocations can be made both on a domain-by-domain basis and hierarchically, an excellent match for the needs of Internet2. DiffServ’s Premium service offers a simple, well-defined implementation in the network. The end-to-end QoS it delivers can also be easily measured.

Premium service also offers the following advantages:

- It has a simple, understandable, quantifiable, “strong” service semantic.
- Most of the required components for network elements exist today.
- Implementation is straightforward.
- Sophistication can be added to the resource allocation mechanisms incrementally.

For these reasons, Premium is the ideal roll-out service for Internet2.

1. Introduction

The DiffServ approach to providing IP QoS has been gaining attention in the past year.

One end-to-end service within this framework was originally proposed by Van Jacobson [1,2] and called “Premium service.” A somewhat more general architecture enabling Premium service has also been previously outlined [3]. Also, the Internet Engineering Task Force (IETF) working group on DiffServ is proposing a framework that will encompass Premium service [5,13]. Premium service and Jacobson’s associated allocation architecture [6] are of particular interest to Internet2 because they meet Internet2’s requirements so well.

In Premium service, allocations are expressed in terms of peak rate, as measured by a token bucket with a depth of one or two packets. The smallest unit of allocation is a single “flow” of packets as determined by a match on IP header fields of source address, destination address, ports, and protocol. Packets of the affected flow are marked in their packet headers for Premium service and shaped for conformance to the rate requirements at the edge of the network. This may be done by an edge network node, transparently to the host, or may be done by a host. In the latter case, the usual architecture would call for the first-hop edge network node to monitor that the marked traffic conforms to the allocation the network has recorded for that flow. (“A Two-bit Differentiated Services Architecture for the Internet” [3] describes Premium service in detail.)

The marking that packets carry is used by subsequent network nodes to enqueue the packets for priority service at each output port. The combination of strict priority service, no bursts at the peak rate, and a requirement that the service not be oversubscribed means that packets will experience close to

the minimum possible latency and jitter through the network.

Since bandwidth allocation is by peak rate only, this single number can be simply added up to guide allocation policies. Such simplicity makes early deployment possible and should be quite adequate for early adopting networks, where QoS applications are not expected to be a large percentage of the bottleneck network bandwidth.

Further, the peak rate method allows mixtures of static and dynamic allocation. For example, a completely dynamic QoS might be used within one campus, employing hosts that signal for QoS using the RSVP protocol. When that campus has packet traffic that crosses the boundary onto the Internet2 backbone, priority-marked traffic might be allocated from a static allocation across the boundary and policed to that level. In the same Internet2 internetwork, another campus might use completely static allocations and another might use a mixture of static allocations and requests made directly to the network administrator. The choice of intradomain allocation is up to each domain. The choice of allocation across a boundary depends only on the agreement between the domains on each side of the boundary.

2. Meeting Internet2's Requirements

This section reviews Internet2's QoS requirements and notes how Premium service can meet each one.

2.1 Enabling Advanced Applications

Premium service can deliver latencies that are near network minimum since its packets are marked for priority service at each network node and the amount of packets permitted in that priority queue is strictly limited by the conservative allocation strategy. Maximum jitter within a subscribed service rate is a single packet. The burst and rate limitations also make for small, predictable queues at each network node (on the order of the in-degree of each network node).

Thus, there is no loss due to queue overflow. Premium service, therefore, meets all Internet2's outlined [4] requirements for enabling advanced applications.

Although Premium service has low jitter and per-packet delay, it is entirely conceivable that some important applications will not require this feature. These applications might focus instead on the fact that it is possible to bound the amount of time to transfer a large amount of data, since the peak bandwidth of the transfer is known and guaranteed.

It is worth noting that this service can be provided to very low-rate flows (as in a voice over IP application), as well as high-rate flows.

When there is a large amount of bandwidth available, as is expected in Internet2, Premium service allows for its efficient use. Any unused portion of the "pipe" not in current use — including any portion allocated to a Premium service application — may be used by other applications.

2.2 Multiple and Concatenatable Implementations

To enable Premium service, network nodes must be able to recognize priority marks in packet headers and place those packets in a priority queue for output. Many manufacturers already offer priority queuing and are moving quickly toward the necessary classification functions.

Other differentiated services being discussed either have more subtle features that could give somewhat different behavior in different implementations or are designed for intradomain traffic engineering, rather than for end-to-end services. For example, a proposed "Assured service" [3,7,8] relies on a preferential drop mechanism implemented in every packet buffer in the network in order to queue packets. In other words, rather than marking which packets are sent first, Assured service classifies which packets will be dropped last. Studies [9,10] show that the ratio of the delivered rate to the tar-

get rate in such a service can vary widely depending on several factors, such as round trip time of the connection, percentages of the bottleneck link allocated to Assured traffic, and particular TCP implementations. Assured service does not provide a lower per-packet delay than Premium service, except for any delay benefits that might accrue from using active queue management to keep average buffer sizes low.

Thus, Premium service is the currently defined service most likely to be predictable across a variety of manufacturers' equipment. Its delivered performance can also easily be quantified.

The simplicity of this per-network-node forwarding behavior (what IETF's DiffServ working group calls "per-hop behaviors" or PHBs), priority queuing, coupled with Premium service's "rule" that the bottleneck bandwidth of the network path must not be oversubscribed, means that the results of concatenating the behavior are well understood. The behaviors of other differentiated services under concatenation have not been well explored.

2.3 Scales

Differentiated services have been engineered for scalability, and Premium, as a class of differentiated service, is scalable. DiffServ classifies packets on multiple fields in packet headers and shapes traffic flows at the network's edge where the number of flows that each network node must handle is much smaller than the number of QoS flows in the core of the network. All packets are aggregated into a single category (called a "behavior aggregate") denoted by a one-byte marking in the packet header. Core nodes classify packets only on a small number of possible values in that byte and output queues accordingly. Thus the allocations are also aggregated so that the agreement between a backbone offering Premium QoS and a client network is quite simple, e.g., "Campus A may send priority-marked packets at a rate of X from 8 am to 5 pm every day."

2.4 Administrable

Premium service has a simple allocation strategy: only allocate up to some percentage of the bottleneck link of the network. Further, control of this is also fairly simple. Some trusted network element sets the first-hop network node (or edge device) to classify, mark, and shape a certain stream of packets at a set rate in the case of "unaware" hosts. In the case of "QoS aware" hosts, the edge device is set to police conformance of the already marked packet stream. It is possible that some network administrators may not even feel the need to do this policing of aware hosts, further simplifying administration.

A variety of implementations are possible for the "trusted network element," or Bandwidth Broker [1,2,3,6]. Very simple ones are possible, facilitating early deployment.

Premium's current administrative simplicity is a boon to getting a working QoS testbed off the ground, but it should be noted that more complex allocation strategies, thus more complex administration, may emerge as we gain experience with this method.

2.5 Measurable Service

There are a number of types of differentiated service emerging, but only two of these have some "profile" or specification of the QoS requested, Premium and Assured. The "precedence" or "class" service proposed in [8] assigns relative shares of service at each network node and thus is dependent on the exact setting of the shares as well as the output bandwidth at each node and the additional traffic in the network. This might give services that "feel" less congested to their users and, over time, result in faster data transfers, but it is not apparent how to monitor received QoS vs. requested QoS. Thus, this does not appear to be well suited to Internet2 requirements.

Premium service is the most measurable of all currently defined differentiated services. As discussed previously, Assured service has been shown to deliver target profile rates that vary, with that variance

depending on several network and host features [9,10]. In contrast, the receiver of an end-to-end flow granted Premium service can measure packet rate and jitter to determine compliance. If a two-way Premium service is set up between two hosts, it is possible to measure the round-trip delay, which should both be minimal and have low variance.

2.6 Host Requirements

It is possible to deploy DiffServ in a network using legacy hosts that do not have special operating system features to detect flows marked for the Premium service. Instead, flows can be earmarked for Premium service by some external configuration (see [3]). However, where RSVP is available in hosts, it should be possible to use it to signal for Premium service. (This using an architecture outlined by Bernet et. al. [11].)

Of course, to realize the full benefits of the Premium service, applications must be sending at target rate and within the desired jitter bounds. No network service can improve on the quality of the data stream fed it.

2.7 Early and Incremental Deployment

As mentioned in 2.2, many of the required features are either already in manufactured equipment or should be available very soon. Prototypes of Premium service have already been deployed in the Department of Energy's research network [12].

Since the service can be deployed with very simple and static allocations, many of the hard problems of QoS that have stalled other proposed services can be worked out in parallel with deployment and gaining experience from the simple strategies. Further, a number of variations on allocation and a number of different local policies can coexist in different campuses.

3. Conclusion

Premium service is the most promising profile-based differentiated service (PBDS) to meet the demanding requirements of advanced Internet2 applications. Because it is the most easily-understood PBDS and requires the simplest functionality in network elements, it is also the most promising service for early deployment in an Internet2 testbed.

4. Acknowledgements

Thanks to Ben Teitelbaum and the rest of the Internet2 QoS Working Group.

5. References

- [1] Van Jacobson, "An Architecture for Differentiated Services," Talk at IRTF End-to-end Working Group, <ftp://ftp.ee.lbl.gov/talks/vj-e2e-jul97.pdf>, July 1997.
- [2] Van Jacobson, Presentation at IETF Int-serv Working Group, August 1997.
- [3] K. Nichols, V. Jacobson, L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," Internet Draft <draft-nichols-diff-svc>, <http://www-nrg.ee.lbl.gov/papers/2bitarch.pdf>, November 1997.
- [4] Ben Teitelbaum, "QoS Requirements for Internet2," Internet2 Technical Paper, 1998.
- [5] K. Nichols and S. Blake, "A Differentiated Services Operational {{Definition}}," Internet Draft, <draft-nichols-dsopdef-00.txt>, February 1998.
- [6] Van Jacobson, "Differentiated Services and Bandwidth Brokers," Presentation at Bay Networks, Santa Clara, March 1998.
- [7] Dave Clark, Presentation at IETF Int-serv Working Group, August 1997.
- [8] F. Baker, S. Brim, T. Li, F. Kastenholz, S. Jagannath, and J. Renwick, "IP Precedence in Differentiated Services Using the Assured Service," Internet Draft <draft-diff-serv-precedence-00.txt>, April 1998.
- [9] D.D. Clark, W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," <http://diffserv.lcs.mit.edu/Papers/exp-alloc-ddc-wf.pdf>.
- [10] Wenjia Fang, "The 'Expected Capacity' Framework: Simulation Results," Princeton University Computer Science TR, January 1998.

[11] Y. Bernet et. al., "A Framework for End-to-End QoS Combining RSVP/Intserv and Differentiated Services," draft-bernet-intdiff-00.txt, <ftp://ds.internic.net/internet-drafts/draft-bernet-intdiff-00.txt>, March 1998.

[12] Van Jacobson, "Moving the Internet Beyond Best-effort," Talk at DOE LSN Research Workshop, <ftp://ftp.ee.lbl.gov/talks/vj-doeqos.pdf>, January 5, 1998.

[13] K. Nichols and S. Blake, Definition of the Differentiated Services Field (DS Byte) in the IPv4 and IPv6 Headers, Internet Draft <draft-ietf-diffserv-00.txt>, May 1998.

Author's Address

Kathleen Nichols
Bay Networks, Inc.
Bay Architecture Lab
4401 Great America Parkway, SC1-04
Santa Clara, CA 95052
Phone: (408) 495-3252
Email: knichols@baynetworks.com

APPENDIX TWO: QoS WORKSHOP ATTENDEES

Osama Aboul-Magd
Advisor, Network Performance
Nortel

William Adamson
Assistant Director for Product Development
University of Michigan at Ann Arbor

Guy Almes
Chief Engineer
Internet2

George Badger
Associate Vice Chancellor
University of Illinois at Urbana-Champaign

Tom Barron
Manager, Network Design Group
University of Minnesota—Twin Cities

Jeffery Bauer
Florida State University

Micah Beck
Assistant Professor
University of Tennessee at Knoxville

Jon Bennett
Assistant Dept. Manager
GTE Internetworking

Javad Boroumand
National Science Foundation (NSF)

Heather Boyles
Chief of Staff
Internet2

Scott Bradner
Technical Consultant
Harvard University

Scott Brim
*Director of Internetworking
Strategic & Technical Planning*
Newbridge Networks

Wayne Bullock
Manager, Network Services
Florida Atlantic University

Lynn Cannon
Assistant Director for Technology
Washington State University

Ben Colley
Manager, Network Operations, MOREnet
University of Missouri at Columbia

Mike Contino
Manager of Backbone Services
Pennsylvania State University

John Coulter
Network Engineer
Bell Canada

Chip Cox
Principle for Research and Development
Vanderbilt University

Gary Crane
Executive Director
NYSERNet, Inc.

Larry Dunn
*Technology Development Manager
Advanced Internet Initiatives*
Cisco Systems

Richard Edell
Graduate Student Researcher
University of California at Berkeley

Khamsa Enaya
National Aeronautics & Space Administration

Paul Ferguson
Consulting Engineer
Cisco Systems

Dale Finkelson
Network Engineer
University of Nebraska

Melyssa Fratkin
Meeting Coordinator
Internet2

Ken Freeman
NREN Project Engineer
National Aeronautics & Space Administration

Mark Fullmer
Computer Specialist
Ohio State University—Main Campus

Ruediger Geib
Deutsche Telekom

Mario Gerla
Professor
University of California at Los Angeles

Ramesh Govindan
University of Southern California

Terry Gray
Director, Networks & Distributed Computing
University of Washington

Jim Grisham
Network Engineer
NYSERNet, Inc.

Michael Grobe
Manager, Distributed Computing Support
University of Kansas

Robert Grossman
Director, Laboratory for Advanced Computing
University of Illinois at Chicago

Matt Grover
Sr. Network Engineer
University of Florida

Roch Guerin
IBM Corporation

John Hall
Director, Computing Systems Services
Florida Atlantic University

Ted Hanss
Director, Applications Development
Internet2

Warren Hardy
Sr Research Engineer
Ericsson CyberLab

Susan Hares
Director, Merit Gated Consortium
Merit Network, Inc.

Jeff Harrison
Manager Network Engineering Services
University of Alaska at Fairbanks

Charles Hedrick
Director, Advanced Computing Applications
Rutgers University at New Brunswick

Rodger Hess
Assistant Director, Advanced Technology & Architecture
University of California at Davis

Sanjay Hiranandani
Network Engineering Specialist
Cornell University

Mike Hite
Manager, Network Application Services
Pennsylvania State University

Russ Hobby
Director, IT: Advanced Networked & Scientific Applications
University of California at Davis

Neal Hodges II
Technical Service Manager
South Dakota School of Mines and Technology

Michael Hrybyk
General Manager
BCnet

Cristian Huitema
Bellcore

Ron Hutchins
Director of Engineering
Georgia Institute of Technology

Julio Ibarra
Associate Director, Telecommunications
Florida International University

Van Jacobson
Information Computing Sciences
LBNL/ESnet

Raj Jain
Ohio State University—Main Campus

Michael James
Manager, Internet Technology Strategy
IBM Corporation

Sugih Jamin
University of Michigan at Ann Arbor

William Jepson
Director of Computing, UCLA Department of Architecture
University of California at Los Angeles

C. Stuart Johnson
Design Engineer
Packet Engines

Marjory Johnson
Senior Scientist
RIACS/NASA

Kevin Jones
National Aeronautics & Space Administration

John Kalbach, Jr.
System Engineer II
Pennsylvania State University

Sunil Kalindini
Internet Engineer
Advanced Network & Services

Charley Kline
Network Architect
University of Illinois at Urbana-Champaign

Kenneth Klingenstein
Director, Computing & Network Services
University of Colorado at Boulder

Peter Kozdon
Manager
Siemens

Jaron Lanier
Advanced Network & Services

Joseph Lappa
Carnegie Mellon University

Caren Litvanyi
Network Engineer
University of Colorado at Boulder

E. Paul Love, Jr.
Internet2

Landy Manderson
Lead Software/Network Specialist
University of Alabama at Birmingham

Willis Marti
Senior Lecturer
Texas A&M University

Dave Marvit
Consultant for Internet Strategy
Fujitsu

Stan McClellan
Assistant Professor
University of Alabama at Birmingham

Don McGregor
Programmer
Naval Postgraduate School

Don McWilliam
Coordinator, Network Operations Centre
BCnet

Tom Meehan
Senior Product Manager
Bay Networks

Nabil Nabhani
Network Consultant
University of California—San Francisco

Vishy Narayan
National Aeronautics & Space Administration

Kathleen Nichols
Senior Principal Engineer
Bay Networks

Ann O'Beay
Director of Corporate Relations
Internet2

Ken Peirce
Senior Project Engineer
3Com Corporation

Carol Politi
Director, Business Development
Torrent Networking Technologies

J. Mark Pullen
Associate Professor, Computer Science
George Mason University

Eddie Rabinovitch
Senior Network Design Consultant
3Com Corporation

Mike Rabne
Software Engineer
Case Western Reserve University

Rayadurgam Ravikanth
Senior Research Engineer
Nokia Research Center

Y.V. Reddy
Director, Concurrent Engineering Research Center
West Virginia University

Bradley Reese
Director, Information Services
George Washington University

Tad Reynales
NERO Project Manager
Oregon State University

Bob Riddle
Technologist
Internet2

Matt Rockwell
Engineering Manager
3Com Corporation

Bill Russell
Senior Networking Engineer
New York University

George Sadowsky
Director, Academic Computing Facility
New York University

Donald Salvin
Network Engineering Manager
University of Pittsburgh

Andrew Schmidt
Product Manager
Ameritech

Shawn P. Sexton
Director of Network Engineering
University of Notre Dame

John Sikora
Technical Manager—Broadband and Advanced Networking
AT&T

Jerry Sobieski
Network Engineer
Internet2

Chuck Song
Advisory Engineer
MCI

Gordon Springer
Associate Professor, Computer Engineering & Computer Science
University of Missouri at Columbia

Vijay Srinivasan
IBM Corporation

Bill St. Arnaud
Director, Network Projects
CANARIE, Inc.

Ben Stein
Sr. Systems Architect
DePaul University

Benjamin Teitelbaum
Internet Engineer
Internet2

Bill Terrell
Senior Software Engineer
Bay Networks

Douglas E. Van Houweling
President & CEO
Internet2

Piet Van Mieghem
Alcatel Telecom

Robert P. Vietzke
*Manager of Video Communications, Communication
Services*
University of Connecticut

Xin Wang
Columbia University

Zheng Wang
Lucent Technologies

Alan Whinery
Senior Network Engineer
University of Hawaii at Manoa

Craig White
Communications Specialist
University of Alabama

Greg Wood
Communications Director
Internet2

John Wroclawski
Research Scientist
Massachusetts Institute of Technology

Eric Yang
MTS
SBC Communications

Hui Zhang
Carnegie Mellon University

Lixia Zhang
University of California at Los Angeles

Michael Zyda
Professor of Computer Science
Naval Postgraduate School

ACRONYMS

AAA: Authentication, Authorization, and Accounting
API: Application Programming Interface
AS: Autonomous System
ATM: Asynchronous Transfer Mode
AUP: Acceptable Use Policy
BB: Bandwidth Broker
BGP: Border Gateway Protocol
CAD: Computer Aided Design
CAR: Committed Access Rate
CBQ: Class-Based Queuing
CIR: Committed Information Rate
CoS: Class of Service
CSI: Carrier Scale Internetworking
EF: Expedited Forwarding
FR: Frame Relay
FTP: File Transfer Protocol
gigaPoP: gigabit Point of Presence
HPC: High Performance Connections
HTTP: HyperText Transfer Protocol
IEEE: Institute of Electrical and
Electronics Engineers
IETF: Internet Engineering Task Force
IGMP: Internet Group Multicast Protocol
IGP: Interior Gateway Protocol
IP: Internet Protocol
IPPM: Internet Protocol Provider Metrics
IPSEC: Internet Protocol SECurity protocol
ISP: Internet Service Provider
LAN: Local Area Network
MBone: Multicast backBone

MPEG: Motion Picture Experts Group
MPLS: Multi-Protocol Label Switching
OC-*n*: Optical Carrier *n*
PBDS: Profile-Based Differentiated Service
PHB: Per-Hop Behavior
PVC: Permanent Virtual Circuit
PVP: Permanent Virtual Path
QBone: QoS backBone
QoS: Quality of Service
RED: Random Early Detection
RIO: RED with In/Out of profile
RJE: Remote Job Entry
RSVP: resource ReSerVation Protocol
SLA: Service Level Agreement
SONET: Synchronous Optical NETwork
SRL: Soft Real-time resource Library
SVC: Switched Virtual Circuit
TCP: Transmission Control Protocol
ToS: Type of Service
UCAID: University Corporation for
Advanced Internet Development
vBNS: very high performance Backbone
Network Service
VPDS: Virtual Private Data Service
WAN: Wide Area Network
WDM: Wave-Division Multiplexing

See also BABEL:

A Glossary of Computer Oriented
Abbreviations and Acronyms at <[http://
www.cis.columbia.edu/glossary.html](http://www.cis.columbia.edu/glossary.html)>.

