Internet Quality of Service: an Overview

Weibin Zhao, David Olshefski and Henning Schulzrinne
Columbia University
{zwb, dpo1, hgs}@cs.columbia.edu

Abstract

This paper presents an overview of Internet QoS, covering motivation and considerations for adding QoS to Internet, the definition of Internet QoS, traffic and service specifications, IntServ and DiffServ frameworks, data path operations including packet classification, shaping and policing, basic router mechanisms for supporting QoS, including queue management and scheduling, control path mechanisms such as admission control, policy control and bandwidth brokers, the merging of routing and QoS, traffic engineering, constraint-based routing and multiprotocol label switching (MPLS), as well as end host support for QoS. We identify some important design principles and open issues for Internet QoS.

1 Introduction

Quality of Service (QoS) [15, 17] has been one of the principal topics of research and development in packet networks for many years. This paper presents an overview of Internet QoS.

The paper is structured as follows: Section 2 discusses motivation and special considerations for adding QoS to the Internet. Section 3 defines Internet QoS. After outlining two important frameworks, integrated and differentiated service, in Section 4, we present basic data path operations, namely packet classification, shaping, policing, and two important router QoS supporting mechanisms, queue management and scheduling, in Section 5. We show control path mechanisms in Section 6, including admission control, policy control and bandwidth broker. In Section 7, we cover the merging of Internet routing and QoS, addressing traffic engineering, constraint-based routing and multiprotocol label switching (MPLS). In Section 8, we discuss end host support for QoS. We conclude by listing open issues.

2 Motivation

Quality of service (QoS) generally describes the assurance of sufficiently low delay and packet loss for certain types of applications or traffic. The requirements can be given by human factors, e.g., bounds on delay for interactive voice communications, or by business needs, e.g., the need to complete a transaction within a given time horizon.

QoS can be described qualitatively (relative) or quantitatively (absolute). Relative QoS definitions relate the treatment received by a class of packets to some other class of packets, while absolute definitions provide metrics such as delay or loss, either as bounds or as statistical indications. Examples of absolute bounds are statements such as "no more than 5% of the packets will be dropped" or "no packet will experience a delay of more than 100 ms". A set of such statements, along with guarantees about reliability, are often called a Service Level Agreement (SLA). Proportional QoS [13, 14] tries to refine and quantify relative QoS.

As long as the sum of the bandwidths of the ingress links exceeds the minimum capacity of a network, QoS can be offered only in one of two ways: either by predicting the traffic and engineering the network to make violations of the committed QoS sufficiently unlikely or by restricting the total amount of traffic competing for the same resources. In many cases, the network capacity is effectively partitioned by packet prioritization, so that higher-priority traffic is largely unaffected by lower-priority traffic.

QoS guarantees can be made either over an aggregate of communication associations, or for an individual group of packet delineated in time. The latter is often called a "flow". QoS is assured by reserving resources, primarily bandwidth and sometimes buffer space.

Excess traffic can be dropped either at the packet level (policing) or at the flow level (admission control), as discussed later. When network traffic is limited via admission control, packet loss and excessive delay is replaced by flow blocking. The network has to have sufficient capacity to ensure only modest levels of flow blocking. (For example, the telephone network is generally engineered to have less than 1% call blocking.) The permissible level of flow blocking is often also part of an SLA.

Flow-level blocking is appropriate only for applications whose utility function drops to zero at some non-zero bandwidth. For those applications, waiting for available bandwidth is preferable to obtaining bandwidth insufficient for the application. As an example, consider a network with a bottleneck bandwidth of 1 Mb/s. If the network is to be used for voice calls, with a minimum bandwidth of 64 kb/s and a tolerable packet loss of 5%, no more than 16 voice calls can be admitted. If the 17th call is admitted, the quality of all calls drops below the tolerable threshold, so it is preferable to delay one call in that case. We refer to traffic whose utility function drops to zero above zero bandwidth as QoS-sensitive.

It has been argued that data networks have a sufficiently low utilization to make resource reservation for QoS-sensitive traffic unnecessary. However, while the average utilization in a network may be low, there are likely to be times and places where congestion occurs, at least temporarily.

Applications differ in their QoS requirements. Most applications are loss-sensitive; while data applications can recover from packet loss via retransmission, losses above 5% generally lead to very poor effective throughput. Data applications such as file transfer are not generally delay-sensitive, although human patience imposes lower throughput bounds on applications such as web browsing. Continuous media applications such as streaming audio and video generally require a fixed bandwidth, although some applications can adapt to changing network conditions (see Section 8.2).

This diversity of applications makes the current Internet approach of offering the same, "best-effort" service, to all applications inadequate. ISPs also see service differentiation as a way to obtain higher revenue for their bandwidth.

In short, it is likely that at least portions of the Internet will see service differentiation in the near future [9, 33]. Since best-effort service will continue to be dominant, all Internet QoS mechanisms are layered on top of the existing Internet rather than replacing it with a new infrastructure. Internet design principles [10] such as connectionless service, robustness and end-to-end principles should serve as a guidance for any proposed enhancement to current Internet.

3 Internet QoS Definition

We define QoS as providing service differentiation and performance assurance for Internet applications. Service differentiation provides different services to different applications according to their requirements. Performance assurance addresses bandwidth, loss, delay and delay variation (jitter). Bandwidth is the fundamental network resource, as its allocation determines the application's maximum throughput and, in some cases, the bounds on end-to-end delay. Jitter is a secondary quality-of-service metric, since a playout buffer at the receiver can transform it into additional constant delay.

Service differentiation can be per-flow or at an aggregate. A flow is commonly defined by a 5-tuple, namely source IP address, source port number, destination IP address, destination port number, and protocol (UDP, TCP). This fine granularity protects flows from other, possibly misbehaving, applications, but scales poorly in backbone

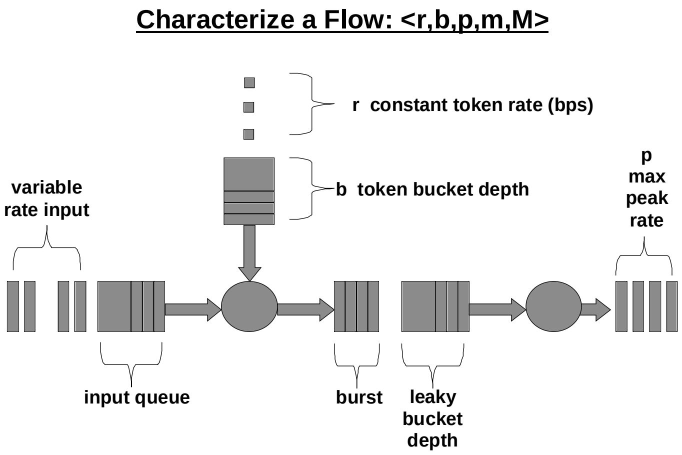

Figure 1: Token bucket

networks where there are possibly tens of thousands of flows.

Thus, a coarser granularity of classification has been proposed, where packets are grouped into several traffic classes, each treated differently. This approach assumes that packets in the same class have similar QoS requirements no matter what flows they belong to. While this aggregate classification scales better and has lower per-packet complexity, its performance guarantees are not as strong as those for the per-flow approach.

Current efforts are focused on aggregate traffic classification. Trying to combine the advantages of both approaches, some research efforts attempt to emulate the behavior and performance of per-flow based mechanism under an per-aggregate-class based framework [32].

In order to provide Internet QoS, we need to describe the properties of flows and aggregates as well as their service requirements. The token bucket is the most commonly used flow specification, for example in the form of the TSpec [30]. The TSpec combines a token bucket with a peak rate p, a minimum policed unit m, and a maximum datagram size M. The parameters M and m are used for packet filtering: a packet whose size is less than m bytes is counted as m bytes and any packet over M bytes is considered out of profile. The token bucket has a bucket depth b, and a bucket rate r, with b specifying the maximum burst size and r specifying the maximum service rate. When a packet of length x is serviced, x bytes are removed from the token bucket. If the bucket is empty, the packet must wait in the queue until the bucket fills up with enough tokens. In implementation, a token bucket is often paired with a leaky bucket. Figure 1 illustrates this. Service requirements can be specified in a variety forms, such as the RSpec [30] which includes a service rate (R) and a delay slack term (S).

The specifications of traffic and its desired service can be given on a per-flow basis or in service level agreement

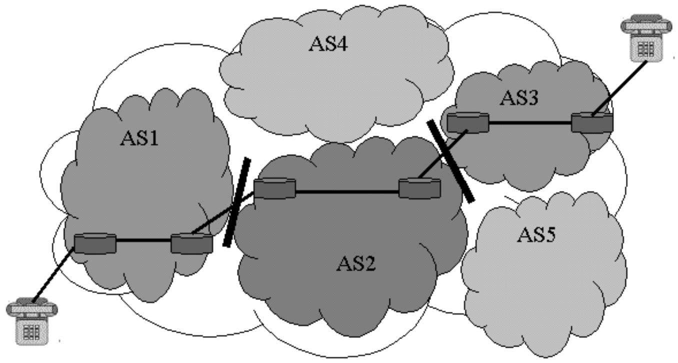

Figure 2: End-to-end QoS

(SLA). An SLA is a service contract between a customer and a service provider. A customer may be an end user (source domain) or an adjacent upstream domain. In addition to the traffic specification, an SLA specifies all the aspects of packet forwarding treatment that a customer should receive from its service provider. Less technical contents may also defined in SLA such as pricing and billing procedures including refunds for unexpected service failures, encryption services provided by the service provider, authentication mechanisms used to verify users, and procedures for re-negotiation or cancellation of the service. To facilitate the establishment of SLA, certain negotiation mechanism is needed. Although an SLA is deemed to be relatively stable, it should be updated to reflect the changes of traffic pattern and its desired service, which requires a re-negotiation of the SLA.

4 Frameworks

How can we achieve end-to-end QoS in the Internet? Since today's Internet interconnects multiple administrative domains (autonomous systems (AS)), it is the concatenation of domain-to-domain data forwarding that provides end-to-end QoS delivery (Figure 2). Although there are variety of choices, two major frameworks, integrated services (IntServ) and Differentiated Services (DiffServ), have emerged as the principal architectures for providing Internet QoS.

4.1 Integrated Services (IntServ)

IntServ [7, 12] is a per-flow based QoS framework with dynamic resource reservation. Its fundamental philosophy is that routers need to reserve resources in order to provide quantifiable QoS for specific traffic flows. RSVP (Resource Reservation Protocol) [8] serves as a signaling protocol for application to reserve network resources.

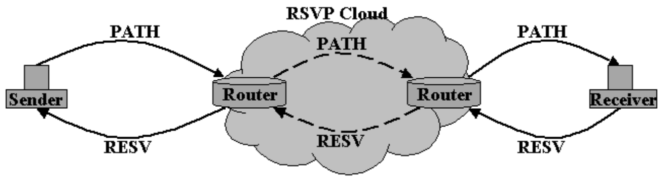

Figure 3: RSVP signaling

RSVP adopts a receiver-initiated reservation style which is designed for a multicast environment and accommodates heterogenous receiver service needs. RSVP works as follows (Fig. 3). The flow source sends a PATH message to the intended flow receiver(s), specifying the characteristic of the traffic. As the PATH message propagates towards the receiver(s), each network router along the way records path characteristics such as available bandwidth. Upon receiving a PATH message, the receiver responds with a RESV message to request resources along the path recorded in the PATH message in reverse order from the sender to the receiver. Intermediate routers can accept or reject the request of the RESV message. If the request is accepted, link bandwidth and buffer space are allocated for the flow, and the flow-specific state information is installed in the routers. Reservations can be shared along branches of the multicast delivery trees.

RSVP takes the *soft state* approach, which regards the flow-specific reservation state at routers as cached information that is installed temporarily and should be periodically refreshed by the end hosts. State that is not refreshed is removed after a timeout period. If the route changes, the refresh messages automatically install the necessary state along the new route. The *soft state* approach helps RSVP to minimize the complexity of connection setup and improves robustness, but it can lead to increased flow setup times and message overhead.

The IntServ architecture adds two service classes to the existing best-effort model, guaranteed service and controlled load service. Guaranteed service [29] provides an upper bound on end-to-end queuing delay. This service model is aimed to support applications with hard real-time requirements. Controlled-load service [35] provides a quality of service similar to best-effort service in an underutilized network, with almost no loss and delay. It is aimed to share the aggregate bandwidth among multiple traffic streams in a controlled way under overload condition.

By using per-flow resource reservation, IntServ can deliver fine-grained QoS guarantees. However, introducing flow-specific state in the routers represents a fundamental change to the current Internet architecture. Particularly in the Internet backbone, where a hundred thousand flows may be present, this may be difficult to manage, as a router may need to maintain a separate queue for each flow.

Although RSVP can be extended to reserve resources for aggregation of flows, many people in the Internet community believe that IntServ framework is more suitable for intra-domain QoS or for specialized applications such as high-bandwidth flows. IntServ also faces the problem that incremental deployment is only possible for controlled-load service, while ubiquitous deployment is required for guaranteed service, making it difficult to be realized across the network.

4.2 Differentiated Services (DiffServ)

To address some of the problems associated with IntServ, differentiated services (DiffServ) has been proposed by the IETF with scalability as the main goal. DiffServ [4, 25] is a per-aggregate-class based service discrimination framework using packet tagging [24]. Packet tagging uses bits in the packet header to mark a packet for preferential treatment. In IPv4, the type-of-service (TOS) byte is used to mark packets. The TOS byte [1] consists of a 3-bit precedence field, a 4-bit field indicating requests for minimum delay, maximum throughput, maximum reliability and minimum cost, and one unused bit. However, these bits were never widely used. DiffServ redefines this byte as the DS field, of which six bits make up the DSCP (Differentiated Service CodePoint) field, and the remaining two bits are unused. The interpretation of the DSCP field is currently being standardized by the IETF.

DiffServ uses DSCP to select the per-hop behavior (PHB) a packet experiences at each node. A PHB is an externally observable packet forwarding treatment which is usually specified in a relative format compared to other PHBs, such as relative weight for sharing bandwidth or relative priority for dropping. The mapping of DSCPs to PHBs at each node is not fixed. Before a packet enters a DiffServ domain, its DSCP field is marked by the end-host or the first-hop router according to the service quality the packet is required and entitled to receive. Within the DiffServ domain, each router only needs to look at DSCP to decide the proper treatment for the packet. No complex classification or per-flow state is needed.

DiffServ has two important design principles, namely pushing complexity to the network boundary and the separation of policy and supporting mechanisms. The network boundary refers to application hosts, leaf (or first-hop) routers, and edge routers. Since a network boundary has relative small number of flows, it can perform operations at a fine granularity, such as complex packet classification and traffic conditioning. In contrast, a network core router may have a larger number of flows, it should perform fast and simple operations. The differentiation of network boundary and core routers is vital for the scalability of DiffServ.

The separation of control policy and supporting mechanisms allows these to evolve independently. DiffServ only defines several per-hop packet forwarding behaviors (PHBs) as the basic building blocks for QoS provisioning, and leaves the control policy as an issue for further work. The control policy can be changed as needed, but the supporting PHBs should be kept relatively stable. The separation of these two components is key for the flexibility of DiffServ. A similar example is Internet routing. It has very simple and stable forwarding operations, while the construction of routing tables is complex and may be performed by a variety of different protocols. (This often reflects a software-hardware split, where PHBs are implemented in hardware, while the control policy is implemented in software.)

Currently, DiffServ provides two service models besides best effort. Premium service [20] is a guaranteed peak rate service, which is optimized for very regular traffic patterns and offers small or no queuing delay. This model can provide absolute QoS assurance. One example of using it is to create “virtual leased lines”, with the purpose of saving the cost of building and maintaining a separate network. Assured service [19] is based on statistical provisioning. It tags packets as *In* or *Out* according to their service profiles. *In* packets are unlikely to be dropped, while *Out* packets are dropped first if needed. This service provides a relative QoS assurance. It can be used to build “Olympic Service” which has gold, silver and bronze service levels.

5 Data Path Mechanisms

Having outlined the frameworks, we will discuss the details of Internet QoS mechanisms along two major axes: data path and control path. Data path mechanisms are the basic building blocks on which Internet QoS is built. They implement the actions that routers need to take on individual packets, in order to enforce different levels of service. Control path mechanisms are concerned with configuration of network nodes with respect to which packets get special treatment what kind of rules are to be applied to the use of resources.

We first discuss the basic data path operations in routers (Fig. 4), including packet classification, marking, metering, policing, and shaping. Then we cover the two basic router mechanisms, queue management and scheduling. They are closely related, but they address rather different performance issues [6]. Queue management controls the length of packet queues by dropping or marking packets when necessary or appropriate, while scheduling determines which packet to send next and is used primarily to manage the allocation of bandwidth among flows.

5.1 Basic Packet Forwarding Operation

As a packet is received, a packet classifier determines which flow or class it belongs to based on the content of some portion of the packet header according to certain specified rules. There are two types of classification:

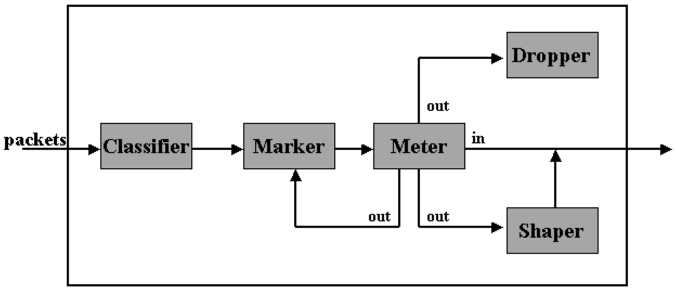

Figure 4: Basic data path operations

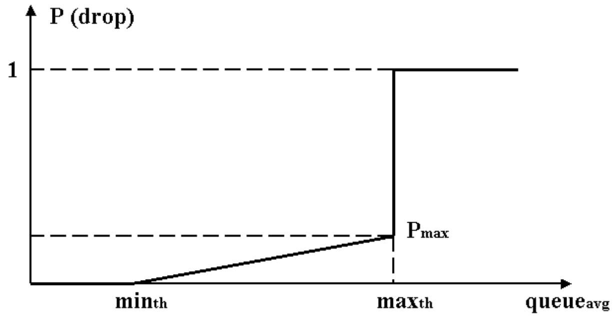

Figure 5: RED queue management algorithm

- General classification performs a transport-level signature-matching based on a tuple in the packet header. It is a processing-intensive operation. This function is needed at any IntServ-capable router. In DiffServ, it is referred as multifield (MF) classification, and it is needed only at network boundary.
- Bit-pattern Classification sorts packet based on only one field in the packet header. It is much simpler and faster than general classification. In DiffServ, it is referred as behavior aggregate (BA) classification which is based only on DS field. It is used at network core routers.

After classification, the packet is passed to a logical instance of a traffic conditioner which may contain a meter, marker, shaper, and dropper. A marker marks certain field in the packet, such as DS field, to label the packet type for differential treatment later. A meter is used to measure the temporal properties of the traffic stream against a traffic profile. It decides that the packet is in profile or out of profile, then it passes the state information to other traffic conditioning elements. Out of profile packets may be dropped, remarked for a different service, or held in a shaper temporarily until they become in profile. In profile packets are put in different service queues for further processing. A shaper is to delay some or all of packets in a packet stream in order to bring the stream into compliance with its traffic profile. It usually has a finite buffer, and packets may be discarded if there is insufficient buffer space to hold the delayed packets. A dropper can be implemented as a special case of a shaper by setting shaper buffer size to zero packets. It just drops out-of-profile packet. The function of a dropper is known as traffic policing.

5.2 Queue Management

One goal of Internet QoS is to control packet loss. It is achieved mainly through queue management. Packets get lost for two reasons: damaged in transit or dropped when network congested [21]. Loss due to damage is rare ($\ll 1\%$), so packet loss is often a signal of network congestion.

To control and avoid network congestion, we need some mechanisms both at network end-points and at intermediate routers. At network end-points, we depend on the TCP protocol which uses adaptive algorithms such as slow start, additive increase and multiplicative decrease. Inside routers, queue management is used. Our goals are to achieve high throughput and low delay. The effectiveness can be measured by network power, which is the ratio of throughput to delay.

The buffer space in the network is designed to absorb short term data bursts rather than be continuously occupied. Limiting the queue size can help to reduce the packet delay bound.

Traditionally, packets are dropped only when the queue is full. Either arriving packets are dropped (tail drop), the packets that have been in the queue the longest are dropped (drop front) or a randomly chosen packet is discarded from the queue. There are two drawbacks with drop-on-full, namely lock-out and full queues. Lock-out describes the problem that a single connection or a few flows monopolize queue space, preventing other connections from getting room in the queue. The "full queue" problem refers to the tendency of drop-on-full policies to keep queues at or near maximum occupancy for long periods. Lock-out causes unfairness of resource usage while steady-state large queues results a longer delay.

To avoid these two problems, we need active queue management which drops packets before a queue becomes full. It allows routers to control when and how many packets to drop. An important example of such algorithm is Random Early Detection (RED) [16].

RED controls the average queue size using time-based exponential decay, and it marks (or drops) arriving packets probabilistically. The probability of marking increases as the estimated average queue size grows. It uses two thresholds: $\min_{th}$ and $\max_{th}$ (Fig. 5). When the average queue size is less that $\min_{th}$, no packets are marked. This should be the normal mode of operation. When the average queue size is greater that $\max_{th}$, every arriving packet is marked. This mode should occur only under congestion. When the average queue size is between $\min_{th}$ and $\max_{th}$, each arriving packet is marked with a probability $p_a \in [0, \text{maxp}]$, where p_a is a function of the average queue size. This is the congestion avoidance phase. Packets get marked in proportion to the flow's link share. RED has two important properties: It avoids global synchronization of TCP by introducing randomness and it has no bias against bursty traffic.

RIO [11] refines RED with In/Out bits. The idea of RIO is to tag packets as being "In" or "Out" according to their service profiles, and preferentially drop packets that are tagged as being "Out" if congestion occurs. RIO uses two sets of parameters, one for *In* packets, and one for *Out* packets. The probability of marking an *In* packet depends on *avg_in*, the average queue size for *In* packets, while the probability of marking an *Out* packet depends on *avg_total*, the average total queue size for *all* (both *In* and *Out*) packets.

5.3 Scheduling

Packet delay control is an important goal of Internet QoS. Packet delay has three parts: propagation, transmission, and queuing delay. Propagation delay is given by the distance, the medium and the speed of light, roughly 5 μs/km. The per-hop transmission delay is given by the packet size divided by the link bandwidth. The queueing delay is the waiting time that a packet spends in a queue before it is transmitted. This delay is determined mainly by the scheduling policy.

Besides delay control, link sharing is another important goal of scheduling. The aggregate bandwidth of a link can be shared among multiple entities, such as different organizations, multiple protocols (TCP, UDP), or multiple services (FTP, telnet, real-time streams). An overloaded link should be shared in a controlled way, while an idle link can be used in any proportion.

Although providing delay guarantee and rate guarantee, are crucial for scheduling, scheduling needs to be kept simple since it needs to be performed at packet arrival rates. For example, at OC-48 rates, a scheduler only has 100 ns per packet to make a scheduling decision.

Scheduling can be performed on a per-flow basis or a per-traffic-class basis. A combination of these two results in a hierarchical scheduling. There are variety of scheduling disciplines [18, 31].

- First Come First Serve (FCFS) is the simplest scheduling policy. It has no flow or class differentiation, no delay or rate guarantee.
- Priority scheduling provides a separate queue for each priority class. Basically, it is a multiple-queue FCFS scheduling discipline with the higher priority queue being served first. It has a coarse granularity class differentiation. But is has no delay or rate guarantee for individual flows.
- Weighted Fair Queuing (WFQ) is variation of weighted round robin scheduling, where the weights are coupled with reserved link rates. It can provide end-to-end delay guarantee on a per-flow basis. But it cannot provide separate delay and rate guarantee. A resulting problem of this is that a low bandwidth flow will experience high delay. There are many variants of WFQ, most of them can be compared with GPS (Generalized Processor Sharing) [27], which is defined for a fluid model of traffic, and serves as a theoretic reference model.
- Earliest Deadline First (EDF) is a form of dynamic priority scheduling. Each packet is assigned a sending deadline which is the sum of arrival time and delay guarantee. Coupled with traffic shapers, EDF can provide separate delay and rate guarantee.

6 Control Path Mechanisms

In this section, we discuss the control path mechanisms including admission control, policy control, and bandwidth brokers.

6.1 Admission Control

Admission control [23, 22] implements the decision algorithm that a router or host uses to determine whether a new traffic stream can admitted without impacting QoS assurances granted earlier. As each traffic stream needs certain amount of network resources (link bandwidth and router buffer space) for transferring data from source to destination, admission control is used to control the network resource allocation. The goal is to correctly compute the admission region, since an algorithm that unnecessarily denies access to flows that could have been successfully admitted will underutilize network resource; while an algorithm that incorrectly admits too many flows will induce QoS violations.

There are three basic approaches for admission control: deterministic, statistic, and measurement-based. The first two use a priori estimation, while the later one is based on the current measurement of some criteria parameters. The deterministic approach uses a worst-case calculation which disallows any QoS violation. It is acceptable for smooth traffic flows, but it is inefficient for bursty flows and leads to a lower resource utilization. Both statistical and measurement-based approach allow a small probability of occasional QoS violation to achieve a high resource utilization.

6.2 Policy Control

Policy [28] specifies the regulation of access to network resources and services based on administrative criteria. Policies control which users, applications, or hosts should have

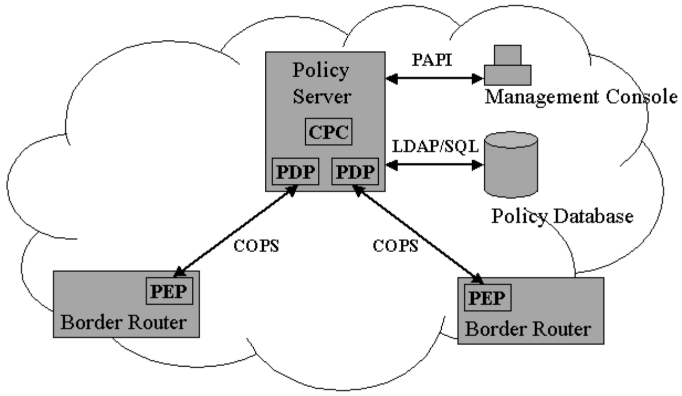

Figure 6: Policy architecture

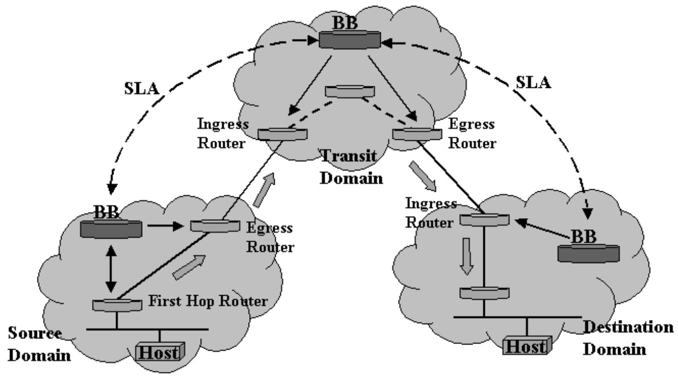

Figure 7: Bandwidth broker

access to which resources and services and under what conditions. Instead of configuring individual network devices, ISPs and corporate administrators would like to regulate the network through policy infrastructure, which provide supports for allowing administrative intentions to be translated into differential packet treatment of traffic flows.

Figure 6 depicts a typical policy architecture. Each domain may contain one or more policy servers whose function is to make policy and configuration decisions for network elements. The policy server has access to a policy database (possibly through LDAP or SQL) as well as authorization and accounting databases. Each policy entry specifies a rule of "if certain condition happens, then take certain action". A human network operator working at a management console would use a GUI management application which interfaces to the policy server through a set of Policy API (PAPI). This allows the operator to update and monitor policy changes in the policy database.

The policy server consists of a central policy controller (CPC) and a set of policy decision points (PDP). PDP's are responsible for determining which actions are applicable to which packets. The CPC is to ensure global consistency between decisions made by the PDP's. The enforcement and execution of policy actions are done by policy enforcement points (PEP). PEP's are typically colocated with packet-forward components, such as border routers. PDP's interact with PEP's via Common Open Policy Service (COPS) [5]. PDP's push configuration information down to the PEP's as well as respond to queries from the PEP's.

6.3 Bandwidth Brokers

A bandwidth broker (BB) [26, 34] is a logical resource management entity that allocates intra-domain resources and arranges inter-domain agreements. A bandwidth broker for each domain can be configured with organizational policies. and controls the operations of edge routers. In the view of policy framework, a bandwidth broker includes the function of PDP and policy database, while edge routers serve as PEPs.

In its inter-domain role, a bandwidth broker negotiates with its neighbor domains, sets up bilateral agreement with each of them, and sends the appropriate configuration parameters to the domain's edge routers (Fig. 7). Bilateral agreement means that a bandwidth broke only needs to coordinate with its adjacent domains. End-to-end QoS is provided by the concatenation of these bilateral agreements across domains, together with adequate intra-domain resource allocation.

Within a domain, a bandwidth broker performs resource allocation through admission control. The choice of the intra-domain algorithm is independent of the inter-domain negotiation.

The architecture of a bandwidth broker bears some similarity to current Internet routing, in which BGP4 serves as the standard inter-domain router protocol, many choices are available for intra-domain routing, and the concatenation of AS-to-AS (Autonomous Systems) forwarding provides end-to-end data delivery.

7 Routing and QoS

Up to now, we address Internet routing and QoS as two separate issues, However, there is evidence that merging routing with QoS can result in better performance. In this section, we briefly review the efforts in this area, covering traffic engineering, constraint-based routing [36], and multiprotocol label switching (MPLS) [2].

7.1 Traffic Engineering

All QoS schemes try to provide differentiated services under overload condition. They differ little from best-effort service if the load is light. There are two reasons for network overloading or congestion: usage demand exceeding the available network resource, or uneven distribution of traffic. In the first case, we can either increase the network capacity or limit usage by QoS mechanisms. In the second case, load distribution and balancing may help. Traffic engineering arranges traffic flows so that congestion caused by uneven network utilization can be avoided.

7.2 Constraint-based Routing

Current Internet routing is mainly based on network topology. It tries to transfer each packet along the shortest path from the source to the destination. Constraint-based routing is an extension to the basic topology-based routing. It routes packets based on multiple constraints. The constrains include network topology (shortest path), network resource availability information (mainly link available bandwidth), flow QoS requirements, and policy constraints. Constraint-based routing can help to provide better performance and improve network utilization. But it is much more complex, may consume more network resource and may lead to potential routing instability.

7.3 MPLS

MPLS offers an alternative to IP-level QOS. An MPLS packet has a header that is sandwiched between the link layer header and the network layer header. The MPLS header contains a 20-bit label, a three-bit class of service (COS) field, a three-bit label stack indicator, and an eight-bit time to live (TTL) field. When a packet enters an MPLS domain, it is assigned an MPLS label which specifies the path the packet is to take while inside the MPLS domain. Throughout the interior of the MPLS domain, each MPLS router switches the packet to the outgoing interface based only on its MPLS label. At the same time, the packet gets marked with a new label prior to transmission. The COS field is used to choose the correct service queue on the outgoing interface. At the egress to the MPLS domain, the MPLS header is removed and the packet is sent on its way using normal IP routing.

MPLS is a label-based message forwarding mechanism. By using labels, it can set up explicit routes within an MPLS domain. A packet's forwarding path is completely determined by its MPLS label. If a packet crosses all MPLS domains, an end-to-end explicit path can be established for the packet. Label also serves as a faster and efficient method for packet classification and forwarding.

MPLS also also to route multiple network layer protocols within the same network and can be used as an efficient tunneling mechanism to implement traffic engineering.

For example, the switching tables must be pushed down to the MPLS routers from a central controller, similar to a policy server. Configuring these tables can be quite complex, which leads to scalability problems.

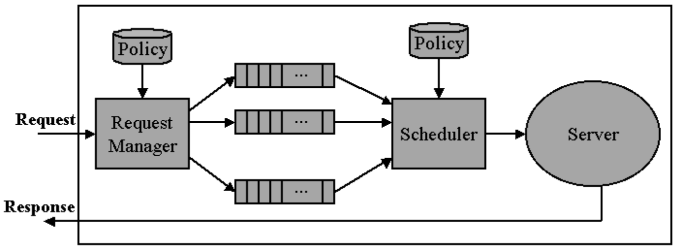

Figure 8: Server QoS

8 End Host Support for QoS

All QoS mechanisms discussed so far are operate within routers. However, network QoS by itself is not sufficient to deliver end-to-end QoS. End host support for QoS, including server QoS and application adaptation, also play an important role.

8.1 Server QoS

Empirical evidence suggests that overloaded servers can have significant impact on user perceived response time. Furthermore, FIFO scheduling done by servers can undermine any QoS improvements made by network since a busy server can indiscriminately drop high priority network packets. There is an increasing need for server QoS mechanisms to provide overload protection and to enable tiered (or differentiated) service support.

Figure 8 gives a simple scheme to enable server QoS [3]. In this scheme, a request manager is used to intercept all requests. It classifies the requests and places the requests in the appropriate queue. To ensure that server is not overloaded, admission control is used by request manager. Request classification and admission control can be based on a variety of policies. After the requests are put on the appropriate queue, a scheduling process determines the order in which the requests are to be served, and feeds the chosen requests to the server. Similar to the scheduling in network routers, different policies can be adopted, such as strict priority, weighted fair queuing, or earliest deadline first.

8.2 Application Adaptation

Instead of indicating requirements to the network via resource reservation mechanisms, applications can adapt their rate to the changing delays, packet loss and available bandwidth in the network.

Playout delay compensation allow applications to function with the lowest overall delay even as the network delay changes. A playout buffer is a queue for arriving packets emptied at the playback rate. It converts a variable-delay packet network into a fixed delay, and it has to be large enough to compensate for the delay jitter.

For loss adaptation, several techniques have been explored, such as redundant transmission, interleaving, and forward error correction.

Bandwidth adaptation depends on the media. For audio, applications can change the audio codec, while video applications can adjust the frame rate, image size and quality.

9 Conclusions

We have surveyed the principal components of the current Internet QoS architecture. Scalability is a fundamental requirement for any Internet QoS scheme. It is an issue that impacts both data path and control path, including flow and queue management, network device configuration, accounting and billing, authorization, monitoring, and policy enforcement. Aggregation is one common solution to scaling problems, but it comes at the price of looser guarantees and coarser monitoring and control.

By separating of control policy from data forwarding mechanisms, we can have a set of relative stable mechanisms on top of which Internet QoS can be built. At the same time, we can easily adjust or add any control policy whenever needed with a simple re-configuration or remapping from policy to mechanisms. All supporting mechanisms can be kept unchanged.

Since the Internet comprises multiple administrative domains, inter-domain QoS and intra-domain QoS can be designed separately. At the inter-domain level, pure bilateral agreement based on traffic aggregate is used. Within each domain, variety of different choices can be adopted. The concatenation of domain-to-domain data forwarding provides end-to-end QoS delivery.

Although important technical progress has been made, much work needs to be done before QoS mechanisms will be widely deployed. In general, pricing and billing are the primary issues that need to be resolved. Once services are billed for, customers will need to be able to monitor that the purchased service is meeting specifications.

While adaptive applications have been built to respond to changes in network performance, integrating a mechanism to adapt to price changes over time adds yet another dimension.

Incorporating wireless transmission into the Internet adds additional QoS issues, such as mobile setup, limited bandwidth, and hand over.

References

[1] P. Almquist. Type of service in the internet protocol suite. Request for Comments (Proposed Standard) 1349, Internet Engineering Task Force, July 1992.

[2] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over MPLS. Request for Comments (Informational) 2702, Internet Engineering Task Force, September 1999.

[3] Nina Bhatti and Rich Friedrich. Web server support for tiered services. *IEEE Network*, 13(5):64–71, September/October 1999.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. Request for Comments (Informational) 2475, Internet Engineering Task Force, December 1998.

[5] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry. The COPS (common open policy service) protocol. Request for Comments (Proposed Standard) 2748, Internet Engineering Task Force, January 2000.

[6] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the internet. Request for Comments (Informational) 2309, Internet Engineering Task Force, April 1998.

[7] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. Request for Comments (Informational) 1633, Internet Engineering Task Force, June 1994.

[8] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation protocol (RSVP) – version 1 functional specification. Request for Comments (Proposed Standard) 2205, Internet Engineering Task Force, September 1997.

[9] Lee Breslau and Scott Shenker. Best-effort versus reservations: A simple comparative analysis. *ACM Computer Communication Review*, 28(4):3–16, September 1998.

[10] David D. Clark. The design philosophy of the DARPA internet protocols. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 106–114, Stanford, California, August 1988. ACM. also in *Computer Communication Review* 18 (4), Aug. 1988.

[11] David D. Clark and Wenjia Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[12] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: architecture and mechanism. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 14–26, Baltimore, Maryland, August 1992. ACM. *Computer Communication Review*, Volume 22, Number 4.

[13] Constantinos Dovrolis and Parameswaran Ramanathan. A case for relative differentiated services and the proportional differentiation model. *IEEE Network*, 13(5):26–34, September/October 1999.

[14] Constantinos Dovrolis, Dimitrios Stiliadis, and Parameswaran Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. *ACM Computer Communication Review*, 29(4):109–120, October 1999.

[15] P. Ferguson and G. Huston. *Quality of Service: Delivering QoS in the Internet and the Corporate Network*. Wiley Computer Books, New York, NY, 1998.

[16] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[17] Ian Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, California, 1998.

[18] R. Guerin and V. Peris. Quality-of-service in packet networks: Basic mechanisms and directions. *Computer Networks*, 31(3):169–189, February 1999.

[19] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. Request for Comments (Proposed Standard) 2597, Internet Engineering Task Force, June 1999.

[20] V. Jacobson, K. Nichols, and K. Poduri. An expedited forwarding PHB. Request for Comments (Proposed Standard) 2598, Internet Engineering Task Force, June 1999.

[21] Van Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, August 1988. Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988.

[22] Sugih Jamin, Peter B. Danzig, Scott J. Shenker, and Lixia Zhang. A measurement-based admission control algorithm for integrated services packet networks. *IEEE/ACM Transactions on Networking*, 5(1):56–70, February 1997.

[23] Edward W. Knightly and Ness B. Shroff. Admission control for statistical qos: Theory and practice. *IEEE Network*, 13(2):20–29, March/April 1999.

[24] Martin May, Jean Bolot, Alain Jean-Marie, and Christophe Diot. Simple performance models of differentiated services schemes for the internet. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, March 1999.

[25] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. Request for Comments (Proposed Standard) 2474, Internet Engineering Task Force, December 1998.

[26] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. Request for Comments (Informational) 2638, Internet Engineering Task Force, July 1999.

[27] Abhay Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Mass., February 1992.

[28] Raju Rajan, Dinesh Verma, Sanjay Kamat, Eyal Felstaine, and Shai Herzog. A policy framework for integrated and differentiated services in the internet. *IEEE Network*, 13(5):36–41, September/October 1999.

[29] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. Request for Comments (Proposed Standard) 2212, Internet Engineering Task Force, September 1997.

[30] S. Shenker and J. Wroclawski. General characterization parameters for integrated service network elements. Request for Comments (Proposed Standard) 2215, Internet Engineering Task Force, September 1997.

[31] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. *ACM Computer Communication Review*, 28(4):118–130, September 1998.

[32] Ion Stoica and Hui Zhang. Providing guaranteed service without per flow management. *ACM Computer Communication Review*, 29(4):81–94, October 1999.

[33] Benjamin Teitelbaum, Susan Hares, Larry Dunn, Robetr Neilson, Raytheon Vishy Narayan, and Francis Reichmeyer. Internet2 qbone: Building a testbed for differentiated services. *IEEE Network*, 13(5):8–16, September/October 1999.

[34] A. Terzis, L. Wang, J. Ogawa, and L. Zhang. A two-tier resource management model for the internet. In *Proceedings of Global Internet*, December 1999.

[35] J. Wroclawski. Specification of the controlled-load network element service. Request for Comments (Proposed Standard) 2211, Internet Engineering Task Force, September 1997.

[36] Xipeng Xiao and Lionel M. Ni. Internet qos: A big picture. *IEEE Network*, 13(2):8–18, March/April 1999.