LECTURE 9
Principles Underlying Internet QoS

This lecture discusses the main principles underlying Internet QoS. We don't focus much on specific schemes (IntServ and DiffServ), leaving that for the sections at the end.

The material for these notes is drawn partially from:

1. S. Shenker [She95], Fundamental Design Issues for the Future Internet , IEEE Journal on Selected Areas in Communications, Vol. 13, No. 7, September 1995, pp. 1176-1188.

2. D. Clark, S. Shenker , and L. Zhang, Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms, in Proc. SIGCOMM '92, Baltimore, MD, August 1992.

■ 9.1 Motivation: Why QoS?

In the previous three lectures, we studied how the Internet manages two network resources: link bandwidth and queue buffer space. A key feature of how the Internet manages these resources is that it relies heavily on end-system cooperation. The lecture on fair queueing showed how network switches could isolate flows from each other using separate queues and non-FIFO link scheduling.

Armed with link scheduling and queue management methods, we can think about suitable *architectures* for improving on the Internet's best-effort service model. That's what QoS is all about, and research on this topic includes both architecture and specific scheduling algorithms. This lecture will focus on the key architectural approaches, rather than on algorithms (variants of fair queueing suffice for our discussion).

QoS research was motivated by the perceived need for the network to provide more than the best-effort service. The arguments for why the network should provide something better than best-effort are usually made along the following lines:

1. The best-effort model serves all applications, but is not optimal for many applications. In particular, applications that require a minimum rate (e.g., some forms of video), or delay bounds (e.g., telephony), or bounded variation in delays (e.g., live

conferencing, perhaps streaming media) could all benefit from being isolated from other less-demanding applications.

2. Revenue opportunities of ISPs: If ISPs were able to treat certain customers or certain application flows better than others, they might make more money. This argument has become a critical determinant of whether QoS is deployed in any network or not: only schemes that provide clear economic incentives and revenue opportunities are likely to be deployed by network providers.

3. Traffic overload protection: Denial-of-service attacks, flash crowds, and certain bandwidth-hungry applications (e.g., P2P file sharing applications today) all consume large amounts of bandwidth and prevent other traffic from receiving their "share". ISPs are interested in isolating classes of traffic from each other, usually by setting rate controls on traffic classes and/or providing guaranteed bandwidth to certain other classes. QoS mechanisms turn out to help in meeting traffic management and overload protection goals.

■ 9.1.1 What kind of QoS might we want?

From the standpoint of the application provider, the following goals might make sense:

1. Guaranteed bandwidth for certain application flows.

2. Guaranteed delay bounds for certain application flows.

3. Minimizing jitter (the variation in delay).

4. Low or close-to-zero packet loss.

From the standpoint of the network provider, the following goals might make sense:

1. Traffic classes for different customer flows, such as "Gold", "Silver", "Bronze", etc., all priced differently. For example, a provider might sell a customer content provider company "Gold" service at a certain rate (and perhaps delay) for all its port 80 traffic as long as the rate of traffic does not exceed some bound specified in the service level agreement (SLA).

2. SLAs that specify minimum rate guarantees through the ISP network for customer traffic, relative rate shares between traffic classes, maximum rates through a network for traffic classes (for protection), delay priorities, etc.

3. Under some conditions, the network provider might want to control who gets to use the network. If the provider does this, he is said to implement *admission control*. Some QoS architectures fundamentally rely on admission control, but others don't.[1] We will look at this issue later in this lecture.

Although the goal of improving application performance using QoS is laudable, that alone turns out to be poor motivation for anyone to deploy QoS. The only chance QoS has

[1] Recall that the circuit-switched telephone network implements admission control.

of being deployed in practice is if it is strongly tied to an improved revenue stream for ISPs.

It took network researchers close to a decade to fully realize this point. The initial approach to network QoS, called *Integrated Services (IntServ)* did not make significant practical impact because it did not have a clear plan for how ISPs would make money with it. The natural question is why an ISP should deploy complex machinery for a content provider to make money from end clients. The lack of a good answer to this question meant that this architecture did not have the right deployment incentives, even though the IntServ model had well-defined end-to-end service semantics for applications.

In contrast, the *Differentiated Services (DiffServ)* architecture was motivated by ISP interests, and defines *per-hop behaviors* that don't really translate into anything well-defined for applications. However, it allows ISPs to provide service differentiation between customers and their traffic classes.

The [CF98] paper covers the Diff Serv topic, while the [CSZ92] paper covers the IntServ topic. Note that the methodology for implementing these two service models in the internet today is a bit different from the descriptions in the paper. The papers provide an overview of the key issues.

Sections 9.4 and 9.5 discuss the details of DiffServ and IntServ. Before that, we discuss a few issues in QoS service models.

9.2 Debating the future service model: Best-effort v. reservations

This section is a quick summary of Shenker's paper [She95].

9.2.1 Motivation

- The Internet today offers a single class of "best-effort" service. No assurance of when or if packets will be delivered, no admission control.
- But the Internet isn't just ftp, email, and telnet any more.
- The above applications are *elastic* applications. Q: what does this term mean? (A: they can derive additional "utility" from every marginal increase in bandwidth starting from (close to) 0.)
- Starting to see more and more real-time streaming applications such as voice and video.
- Claim: these apps are not as elastic or adaptive. Delay variations and losses pose severe problems. And they don't *typically* react to congestion.

  This is a questionable assumption, but seems to be true to some extent for real-time Internet telephony.

- **Question: Should we change the underlying Internet architecture to better accomodate these applications?**

- I.e., should the fundamental best-effort Internet architecture be changed to accomodate multiple service classes? And should the architecture be changed to perform *admission control*?
- Goal of paper: To provide a concrete framework in which this debate can rage on.
- At the highest level, the "right" answers depend on the nature of future network applications, the cost of bandwidth, the cost of more complex mechanisms, etc. Hard to quantify! In particular, if all applications can become rate-adaptive, the conclusions of this paper will be questionable.
- Basic idea: multiple service classes and scheduling schemes in the network routers to allocate bandwidth.

■ 9.2.2 What should network architects optimize?

- Throughput? Utilization? Packet drops? Delays?
- Shenker's claim: No! Really want to optimize "user satisfaction," which would in an efficient system be tied to revenue. More precisely, if s_i is the network service (in terms of throughput, drops, etc.) delivered to user i, there is a *utility function* $U_i(s_i)$ which measures the performance of the application and the degree of user satisfaction. The goal of network design is to maximize $V = \sum_i U(s_i)$.
- But what about other optimization criteria? For example, a service provider might choose to optimize revenue.
- Anyway, this formulation makes it apparent that it's worth rethinking the basic service model the architecture provides. Simple examples show that pure FIFO without favor toward particular flows doesn't always optimize V.
- But should we add multiple service classes, or just add more bandwidth?
- Note: while the goal is to optimize V, it won't necessarily optimize each of the $U_i's$. I.e., the nature of this optimization is to take bandwidth away from the elastic applications and satisfy the more sensitive, inelastic applications.

■ 9.2.3 How is service for a flow chosen?

- Option #1: Implicitly supplied
  - Here, network automatically chooses service class based on packet/flow.
  - Advantage: No changes to end-hosts or applications.
  - Disadvantage: New applications cause problems, since routers don't know what to do about them.
  - In general, this approach embeds application information in the network layer, which has deficiencies.
  - Also suffers from stability problems, because of a changing service model, changing unknown to applications.

  - Uses multi-field classification capability to assign service to packet flow.

- Option #2: Explicitly requested
  - Here, applications explicitly ask for particular levels of service.
  - Problem: incentives. Why will some applications volunteer to ask for lower levels of service?
  - Possible solution: pricing.
  - Social implications of usage-based pricing. This will discourage the browsing mentality of users (which information providers find attractive).
  - Key point: even in a single best-effort class, the notion of incentives is important. Addresses the issue of "whether to send data."
  - Usage-based pricing at a higher granularity than an individual user might be more appropriate.
  - In the explicit model, the network service model is known to the application.

- Link-sharing as an implicit model. Works well when dealing with aggregates of flows.

9.2.4 Do we need admission control?

- Some services might need explicit resource reservation, so the network may need to turn away flows, which if admitted, will violate its current quantitative commitments.

- One approach to answering this question: If there are values of population size n and n' such that $V(n) > V(n')$ for $n < n'$, then $V()$ has a local maximum, and it may be better to turn flows away.

- Question amounts to monotonicity of $V = \sum_i U(s_i)$.

- What is the shape of $U(s_i)$ for various classes of applications?
  1. Elastic applications: concave. For this, $V()$ is monotonically increasing with n.
  2. Hard real-time apps: step function. Clearly admission control is important here (e.g., telephone networks).
  3. Delay-adaptive real-time apps: convex first then concave after inflection point.
  4. Rate-adaptive real-time apps: earlier inflection point than above case.

- If U is convex in some region, then $V()$ has a local maximum. Such a U argues for admission control.

- Q: What does this argument miss? I think it does not adequately capture the dissatisfaction of denied flows! Also, it appears to tacitly assume that in general it is better to turn a flow away that terminate an existing flow. This may not be a good assumption in some cases.

- Overprovisioning as an alternative: how much you have to overprovision depends on variance of bandwidth use. Furthermore, while backbone bandwidths will continue to improve dramatically with time, it's unclear that the same trend is true for last-mile access links.
- Need to overprovision even if you have admission control!
- Shenker concludes that admission control is more cost-effective than the amount of overprovisioning you otherwise have to do.
- Bottom line: Paper sets framework for this debate.

■ 9.3 Principles

In my opinion, there are five key principles underlying Internet QoS models. They are:

1. Explicit v. implicit signaling ends up in radically different architectures. The former leads to end-to-end guarantees or assurances; the latter is only per-hop or perhaps per-ISP assurances. The former, however, isn't tied well with today's economic realities, whereas the latter is. To date, no one has successfully figured out how to marry the nice end-to-end properties of explicit signaling schemes with the simpler models and economic/deployment advantages of implicit approaches.
2. Isolation principle: To isolate flows, use some variant of weighted fair queueing. If the traffic feeding any single fair-queueing queue is modulated by a *linear bounded arrival process* (LBAP), then such a scheduling discipline can bound worst-case delays *independent* of what other traffic there is in the network. An LBAP has two parameters, (b, r), and modulates traffic from a source such that within any period of time t, the number of bits from the source (which could be an aggregate of many flows) does not exceed $b + r \cdot t$. That is, b specifies the largest burst size, and r the rate. Then, as long as the network of weighted fair queueing switches assures a rate of at least r to the LBAP-modulated source, the worst case delay is bounded by b/r + a few correction terms that depend on the largest packet size (these are not too important in practice). If the LBAP source gets a higher rate, g, through the network, then the corresponding delay is bounded by b/g plus correction terms.
3. Jitter sharing principle: Flows in a fair queueing network tend to have poor jitter, and in general have lower jitter in a FIFO network. The reason is that fair queueing spreads bursts out in time. Hence, the designer of a network who wishes both guarantees on delays and low jitter has a complex problem: fair queueing can give good bounds on delays, but has lousy jitter properties; FIFO has good jitter properties, but poor delay bounds.

   Researchers have spent years developing highly complex scheduling schemes to have both good delay bounds and good jitter bounds, but these are generally of limited use in practice.
4. Admission control: A good way to think about the need for admission control is using a utility function framework; in that framework, admission control boils down to whether the utility function has a convex portion.

5. Fancy QoS v. proper provisioning: In the end, if a network has persistent overload, the only correct approach is to provision the network better. No amount of fancy QoS can increase capacity; all it can do is to shift resources around between flows differently from straight best-effort FIFO, so it's inevitable that some flows are going to be unhappy. It's true that delays can be apportioned differently than in FIFO with a fancier scheduling scheme, but that's not a cure for a persistently overloaded network.

The following two sections weren't covered in Fall 2005.

9.4 DiffServ Details

This section is based on notes of Balakrishnan's lecture scribed by Matt Lepinski and Peter Portante, who took 6.829 in Fall 2001.

The goal of DiffServ is to allow ISPs to make more money by offering service classes which are better than best effort. The idea behind Diff Serv is similar to airline ticket pricing, where airlines charge more to first and business class passengers who in return receive preferential treatment over those who pay the standard fare. Unlike IntServ, which attempts to provide applications with end-to-end service guarantees, with DiffServ, ISPs provide customers only with certain *per hop behaviors* and make no claims about end-to-end performance.

However, it is reasonable for customers to think that preferential treatment by their local ISP will result in better overall performance because internet bottlenecks frequently occur at the edges. Additionally, DiffServ has the following advantages over Int Serv:

- DiffServ is easier to implement than IntServ because per hop behaviors can be provided without cooperation from upstream or downstream ISPs.
- Since DiffServ guarantees are relative (i.e. Service Class 1 receives better service than Service Class 2) DiffServ can provide this differentiation for any number of flows. This means that no admission control is needed.
- DiffServ Service Level Agreements (SLA) fit nicely into the current ISP business model.

9.4.1 DiffServ Implementation

DiffServ policy is implemented within a Differentiated Services Region. A DS Region is a group of routers which all use the same Diff Serv policy (typically all the routers belonging to a single ISP). Each DS Region has a set of traffic classes (6 is common in practice) and a policy for associating a traffic aggregate with a traffic class. A DiffServ Router contains the following pieces:

1. A Classifier which looks at each incoming packet and determines which traffic class it belongs in.
2. Multiple Queues, one for each traffic class.

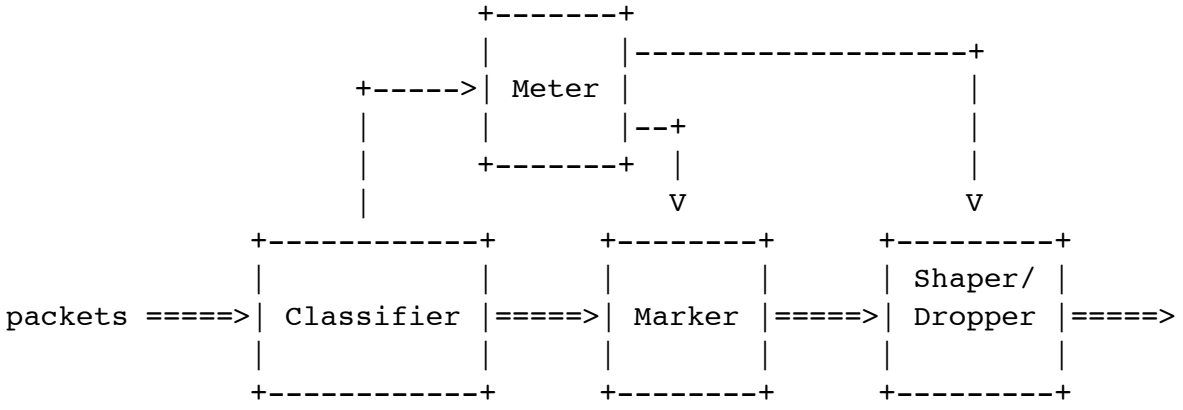
3. A scheduler (using some scheduling algorithm like weighted fair queueing) that decides when each queue gets to send packets.

4. A Meter which determines the the rate of a particular traffic aggregate. This is important because most SLAs specify that a traffic aggregate gets a certain class of service provided that the traffic aggregate doesn't exceed some load. If the rate for a particular traffic aggregate is too high (it is out of profile) then the excess packets either need to be dropped or the traffic needs to be shaped. (Shaping traffic involves holding excess packets in a buffer and letting them trickle out at the rate specified in the SLA).

5. A Dropper/Shaper to deal with traffic which the meter determines to be out of profile

Unfortunately, correctly classifying and metering each incoming packet takes valuable cycles and so it is generally not feasible to do this work at every router within an ISP. Typically, edge routers have extra cycles available for these kind of tasks but high-speed core routers do not. The solution to this problem is to use dynamic packet state (similar to the mechanism used in the Core Stateless Fair Queueing paper). Edge routers do classification and metering and then have a *Marker* which sticks the result of the classification into the IP header (the portion of the IP header which is used is referred to as the DiffServ Code Point (DSCP)). The entire process of classifying, metering, marking and shaping/dropping packets in the edge router is known as *traffic conditioning*.

The picture at the edge looks like this:



■ 9.4.2 Dealing with Excess Traffic

As noted earlier, the Meter in the edge router measures the rate of a traffic aggregate and determines if traffic is out of profile. One solution is to simply drop every packet that is out of profile at the edge router. However, traffic is very bursty and since not all sources are continually sending at their maximum rate, it is possible that the system is able to handle the excess packets. Therefore, dropping at the edge router is often undesirable. The solution then is to let all of the packets through at the edge, but to mark those packets

which are out of profile so that they can be dropped later if the network does not have the capacity to handle the extra traffic.

We then want some mechanism for a bottleneck core router to give preferential treatment to those packets which are marked as being in profile. The easiest solution would be to maintain separate queues for in profile and out of profile packets. This is a bad idea because often a particular flow contains both in profile and out of profile packets and so maintaining separate queues would lead to a large amount of reordering. The solution is to run RED on each queue in the router but to have RED treat each queue as two virtual queues. This algorithm is known as RIO (RED IN/OUT).

RIO maintains a variable q_{ave-in} which is the average number of in profile packets in the queue. RIO then uses this variable along with the RED parameters MIN_{in} and MAX_{in} to determine when to drop in profile packets. Additionally, RIO maintains a variable q_{tot} which is the average size of the queue (including both in profile and out of profile packets). RIO uses this variable along with the RED parameters MIN_{out} and MAX_{out} ($MIN_{out} < MIN_{in}$ and $MAX_{out} < MAX_{in}$) to determine when to drop out of profile packets.

9.4.3 In Practice

In practice, DiffServ implementations use six traffic classes. These classes include one Expedited Forwarding (EF) class, Four Assured Forwarding Classes and one Best Effort class. Standard implementations have the following properties:

- No reordering of packets within a traffic class.
- No standard for bandwidth allocation between classes.
- For each class, there are three drop precedences.
- Expedited Forwarding has the property that whenever a packet is in this queue it gets sent out immediately. Note: This requires that edge routers drop all out of profile EF packets (since core routers never drop EF packets, you can't just mark EF packets out of profile and send them on).

9.4.4 Paper contributions

The authors of [CF98] provide two notable contributions. The first is the movement of packet classification to the edge routers using DSCP fields in the IP header. The second is a set of policies on how to deal with excess traffic, in particular the RIO scheme.

9.5 IntServ Details

This section is based on notes of Balakrishnan's lecture scribed by Matt Lepinski and Peter Portante, who took 6.829 in Fall 2001.

In contrast to DiffServ, IntServ takes an end-to-end approach to the task of providing a QoS beyond Best Effort to application flows through absolute or statistical guarantees. Note that most of the complexity of IntServ results from handling multicast. Additional difficulties arise attempting to perform effective admission control in the face of "controlled load".

9.5.1 End-to-End QoS classes

There are three end-to-end QoS classes:

- Best Effort Service

  As the default class of service provided by the internet, this service makes no guarantees for delivery rate, capacity or even successful delivery of data transmitted through the network.

- Guaranteed Service

  ([RFC 2212] provides the in-depth description of this service)

  This class of service provides firm (mathematically provable) bounds on end-to-end datagram queueing delays. This class of service makes it possible to provide a service that guarantees both delay and bandwidth. [RFC 2212]

- Controlled Load Service

  ([RFC 2211] provides the in-depth description of this service)

  This class of service provides the client data flow with a quality of service closely approximating the QoS that same flow would receive from an unloaded network element, but uses capacity (admission) control to assure that this service is received even when the network element is overloaded. [RFC 2211]

9.5.2 End-to-End QoS Mechanisms

We have covered aspects of the internet's existing end-to-end Best Effort service in previous lectures. Here we describe how Guaranteed and Controlled Load services achieve their provided service levels.

These last two services share three aspects of their design. They contain a signaling protocol used to request service, an admission control scheme to help ensure proper service, and scheduling algorithms in routers which provide the basis for the service.

- Signaling protocol

  A signaling protocol is used to request one of the last two classes of service. The signaling mechanism provides the traffic characteristics (or specification) and a reservation for the resulting traffic specification provided.

  The Resource ReSerVation Protocol (RSVP, [RFC 2205]) is a signalling protocol which takes a traffic specification (TSpec) as input, and returns a PATH message confirming the resource reservation as output. Receivers initiate the reservation request which allows the protocol to accommodate both unicast and multicast traffic flows.

  A TSpec consists of the following information:

  - Maximum Transmission Rate
  - Average Transmission Rate
  - Maximum Burst
  - Minimum Packet Size

  - Maximum Packet Size

  As a TSpec flows through the network, each router provisions resources based on a flow's TSpec, adjusting the TSpec based on what it decides it can handle before forwarding it on to the next router.

  Once the TSpec arrives at the sender, a PATH message is returned to the receiver on the reverse path confirming the reservation. For Guaranteed Service, the PATH message also contains an RSpec, which indicates the level of service reserved for this flow.

  Since each router along the path implicitly keeps a record of a flow's reservation, to avoid keeping track of too much state, routers throw out this information after a certain period of time. This means the state of reservation has to be periodically refreshed.

  There are two aspects to signaling which complicate the design, support for multicast and route changes (since neither service appears to pin routes down).

  Below is a summary of the characteristics of the RSVP signaling protocol taken directly from [RFC 2205]:

  - Makes resource reservations for both unicast and many-to- many multicast applications, adapting dynamically to changing group membership as well as to changing routes.
  - Is simplex, i.e., it makes reservations for unidirectional data flows.
  - Is receiver-oriented, i.e., the receiver of a data flow initiates and maintains the resource reservation used for that flow.
  - Maintains "soft" state in routers and hosts, providing graceful support for dynamic membership changes and automatic adaptation to routing changes.
  - Is not a routing protocol but depends upon present and future routing protocols.
  - Transports and maintains traffic control and policy control parameters that are opaque to RSVP.
  - Provides several reservation models or "styles" (defined below) to fit a variety of applications.
  - Provides transparent operation through routers that do not support it.
  - Supports both IPv4 and IPv6.

  The use of RSVP under Guaranteed and Controlled Load services is specified in [RFC 2210, 2211 and 2212].

- Admission control

  In order to provide a given service level it is necessary to only admit flows up to the point where the service could no longer be provided, e.g. the telephone system. Guaranteed and Controlled Load services use admission control in slightly different ways.

- Scheduling algorithms

  Guaranteed and Controlled Load services require scheduling algorithms in network routers in order to implement their QoS. Variants on Weighted Fair Queueing are used with Guaranteed Service and FIFO variants are used under Controlled Load.

9.5.3 Guaranteed Service

Guaranteed Service provides an assured level of bandwidth to a conforming flow, with a bounded delay and no queueing losses. This is accomplished by combining parameters from individual network elements (routers, subnets, etc.) to produce the guarantee of service.

A conforming flow adheres to a token bucket behavior. Given this description of a flow, a network element computes various parameters describing how the element will handle the flow's data. By combining these individual parameters, it is possible to compute the maximum delay a piece of data will experience on the flow's path (assuming no network element failures or routing changes during the life of the flow). [RFC 2212]

A flow can be described as a token bucket flow with parameters (r, b), where r is the average rate and b is the maximum burst size, if for any time interval, T, the number of bytes sent during T is $\leq (r * T + b)$.

Signaling

The RSVP protocol is modified to add an RSpec along with the TSpec when a receiver initiates its request. A RSpec consists of simply a rate term, R, and a slack term, S, where the slack term is the difference between the requested rate and the reserved rate.

The returned PATH message then contains a modified RSpec describing the exact level of service provided to the requesting flow.

Admission Control

Admission control is straightforward for this service. When a new flow requests service, it is only added if the flow does not cause the combined set of flows in service to exceed the network's capacity.

$$\sum_i alloc + req \leq \text{available G.S. bandwidth}$$

It should be noted that under bursty traffic, the network might end up underutilized. This is a potential drawback to this service.

Scheduling

The routers participating in a guaranteed service offering use a variant on Weighted Fair Queueing for scheduling flows.

Since the flow's traffic conforms to a token bucket behavior, then if WQ: $R_i >= r_i$, there is a maximum bound on end-to-end delay:

$$\text{Max queue } delay < \frac{b}{r_i}$$

Summary

In summary, there are two costs to guaranteed service:

- Potential under-utilization of the network caused by rejecting flows.
- Higher per-flow jitter caused by WFQ spreading out bursts.

9.5.4 Controlled Load

This service class tries to remove the costs associated with Guaranteed Service. It relaxes the admission control policy to address network under-utilization issues, and uses FIFO queueing in the routers to address high jitter.

The end-to-end behavior provided to an application by a series of network elements providing controlled-load service tightly approximates the behavior visible to applications receiving best-effort service *under unloaded conditions* from the same series of network elements. [RFC 2211]

Signaling

Uses the receiver initiated RSVP protocol, but does not rely on RSpec like data as in the Guaranteed Service. Clients provide a TSpec estimating the data traffic they generate, just like they would for Guaranteed Service. A PATH message is returned, but if the clients traffic should exceed its previously specified TSpec, the client's flow may encounter delayed or lost packets.

Admission control

Unlike Guaranteed Service, where admission control is based on whether or not a flow, considering its TSpec, can be strictly accommodated by the network, here network elements are allowed to admit flows with some level of over-commitment. A network element could admit a flow even though the sum of all existing flows' requested TSpecs is greater than the available network capacity, if it notices that flows in general are falling short of their TSpec.

So a network element use the following algorithm to determine if a flow is admitted:

$$\sum_i TSpec_{avg} < \text{avail C.L. bandwidth}$$

The network element sums all known flows' average TSpec, which is based on individual observation, instead of taking it from the RSVP setup message. This would allow network elements to accommodate fluctuations in flow traffic rates while maximizing the number of allowed flows.

Scheduling

In order to alleviate the effects on bursty traffic by WFQ scheduling, routers use FIFO scheduling variants. A FIFO will receive a burst and transmit a burst without spreading out the packets as WFQ thus reducing jitter.