Chapter 2: The Components of Network QoS

Regardless of the size and scope of an IP network, the observed end-to-end quality of service (QoS) is built from the concatenation of edge-to-edge QoS provided by each domain through which the traffic passes. Ultimately, the end-to-end QoS depends on the QoS characteristics of the individual hops along any given route. For example, in Figure 2.1 the QoS experienced by the intra-LAN phone application depends solely on the LAN, whereas the wide area phone application experiences QoS that depends on the LANs at either end, the Internet service providers (ISPs) at either end, and the IP backbone in the middle. A nonspecific PC-to-PC application depends on two LANs and the local ISP providing the LAN-to-LAN interconnect.

Figure 2.1
End-to-end QoS from a concatenation of segments.

Not surprisingly, much of the unpredictable and undifferentiated packet loss and jitter in today’s IP services is due to the manner in which traditional Best Effort routers cope with transient internal congestion. If a particular output port becomes the focal point for two or more inbound aggregate traffic streams, a Best Effort router simply uses first in, first out (FIFO) queuing of packets destined for transmission on the associated outbound link. Queuing introduces latency (delay) and the potential for packet loss if a queue overflows. When traffic patterns are bursty, the queuing-induced latency varies unpredictably from packet to packet—manifesting itself as jitter in the affected traffic streams.

IP networks (enterprise, access, and backbone) are being called upon to carry traffic belonging to a growing variety of customers with diverse requirements—for example, IP Telephony, IP virtual private networks (VPNs), bulk data transfer, and mission-critical e-commerce. Each customer makes unique demands for some level of service predictability, even in the presence of transient congestion due to other traffic traversing the network.

The demand for relative or absolute protection from other traffic on any particular network segment applies equally well to a high-speed LAN, a network based on T1 or E1 private links, a dial-up or ISDN access network, or a high-capacity backbone running at OC-48/STM-16 rates or higher.

This demand leads directly to three technical requirements:

- **Per-hop QoS**—The smallest controllable element in the network is the node (router or switch) joining two or more links. These nodes must be based on an architecture that allows sufficient differentiated queuing and scheduling to be applied at each hop and be able to appropriately utilize the QoS characteristics of inter-node links.

- **Routing and traffic engineering**—Where multiple parallel paths exist through a network, distributing traffic across these paths can reduce the average load and burstiness along any given path. This practice improves the network's apparent service quality because each router is less likely to drop or jitter packets. Mechanisms for discovering and imposing non-shortest-path forwarding are required.

- **Signaling and provisioning**—Controllable per-hop QoS and non-shortest-path forwarding is of little use if its not easily manageable. A practical solution requires some degree of automated
distribution of QoS parameters and/or traffic engineering constraints to all the nodes (routers or switches) in the network. New information is distributed whenever a customer imposes or changes specific end-to-end (or edge-to-edge) QoS requirements.

These requirements are explored in more depth in the rest of this chapter.

2.1 A Hierarchy of Networks

Any network you might care to name is built from a hierarchy of components. Any path from one point to another is usually formed from the concatenation of shorter paths (hops) at the same level. A path at some level $N$ becomes one hop in a path at level $N + 1$. Take the IP layer as the point of reference: It is made up of routers acting as switching points for IP packets and links that carry IP packets between routers. Each link is a single IP hop, yet the link itself might be made up of a number of its own hops and nodes.

The link can be a single Ethernet, a segment of a bridged Ethernet network, an IP tunnel, or an asynchronous transfer mode (ATM) virtual connection. In the case of a bridged Ethernet, one or more Ethernet switches may exist between the two routers. IP tunnels use one IP network to act as a link for another IP network (or sometimes the same IP network when certain types of traffic need to be hidden from sections of the network). An ATM virtual connection (VC) provides an end-to-end service between the ends of the VC, but in reality the VC may pass through many ATM switches along the way.

The IP-level QoS between two points depends on both the routers along the path and the QoS characteristics of each link’s technology. Clearly the inter-router packet transport builds on the QoS capabilities of each link. If the link technology has no controllable QoS, the routers can do little to compensate because they rely on each link to provide predictable inter-router connectivity. However, in the presence of QoS-enabled link technologies, the router’s behavior makes or breaks the availability of IP-level QoS.

Layering is recursive. For example, the QoS characteristics of an ATM VC depend on the predictability of the inter-switch links as much as on the ATM switches themselves. An ATM VC may span multiple ATM switches using Synchronous Optical Network (SONET) or Synchronous Digital Hierarchy (SDH) circuits for inter-switch cell transport. The SONET or SDH circuit itself is made up of one or more hops through various rings and multiplexors. Finally, the SONET or SDH circuits may have been multiplexed onto a single fiber along with totally unrelated circuits using different optical wavelengths—using wavelength-division multiplexing (WDM), an optical fiber multiplication technology that allows lots of virtual fibers to be provisioned within a single physical segment of fiber.

The Internet adds an extra wrinkle on the preceding model because many of the end-to-end paths used are not contained entirely within a single IP network—they are quite likely to span a number of independently administered IP networks (for example, LANs, service providers, and backbone operators as shown in Figure 2.2), each with its own routing policies and QoS characteristics.

Figure 2.2
One level’s edge-to-edge network is another level’s link.

When only Best Effort is required or expected, you don't really need to care about the intermediate
networks along the path, as long as their routing policy allows them to forward traffic. However, to support end-to-end QoS, you need to know more about the network’s dynamic behavior. You do not need to know how each network achieves its QoS goals. It is enough to simply characterize each network in terms of the latency, jitter, and packet loss probabilities that may be imposed on the traffic.

Because one person’s network is another person’s link, the notion of end-to-end QoS must be generalized into one of edge-to-edge QoS. The QoS achieved from one end of a network to another is built from the concatenation of networks with their own edge-to-edge QoS capabilities, and each of these network’s internal paths is built from links that may be networks in their own rights, again characterized by specific edge-to-edge QoS capabilities. The ability to characterize a network’s edge-to-edge QoS behavior depends on the ability to characterize and control both the link and node behaviors at the network level.

2.2 Predictable Per-hop Behavior

The goal in a QoS-enabled environment is to enable predictable service delivery to certain classes or types of traffic regardless of what other traffic is flowing through the network at any given time. An alternative expression of this goal is the process of aiming to create a multiservice IP network solution where traditional bursty traffic may share the same infrastructure (routers, switches, and links) as traffic with more rigorous latency, jitter, bandwidth, and/or packet loss requirements. Regardless of whether you focus on enterprise, access, or backbone networks (or some combination of them all), the end-to-end path followed by a single user’s packets is merely a sequence of links and routers. So, your attention must initially be drawn to the dynamics of a router’s forwarding behavior. Although a traditional router chiefly focuses on where to send packets (making forwarding decisions based on the destination address in each packet and locally held forwarding table information), routers for QoS-enabled IP networks must enable control of when to send packets. You need to look more closely at those elements of a router that affect when packets are actually forwarded.

2.2.1 Transient Congestion, Latency, Jitter, and Loss

Each router is the smallest controllable convergence and divergence point for tens, hundreds, and thousands of unrelated flows of packets. In most data networks, traffic arrives in fluctuating bursts. On regular occasions, the simultaneous arrival of packet bursts from multiple links, which are all destined for the same output link (itself having only finite capacity), leave a router with more packets than it can immediately deliver. For example, traffic converging from multiple 100Mbit per second Ethernet links might easily exceed the capacity of a 155Mbit per second OC-3/STM-1 wide area circuit, or traffic from a number of T3/E3 links may simultaneously require forwarding out along a much smaller T1/E1 link. To cope with such occasions, all routers incorporate internal buffers (queues) within which they store excess packets until they can be sent onwards. Under these circumstances packets attempting to pass through the router experience additional delays. Such a router is said to be suffering from "transient congestion."

The end-to-end latency experienced by a packet is a combination of the transmission delays across each link and the processing delays experienced within each router. The delay contributed by link technologies such as SONET or SDH circuits, "leased line" circuits, or Constant Bit Rate (CBR) ATM virtual circuits is fairly predictable by design. However, the delay contributed by each router’s congestion-induced buffering is not so predictable. It fluctuates with the changing congestion patterns, often varying from one moment to the next even for packets heading to the same destination.
As you recall from Chapter 1, "The Internet Today," this randomly fluctuating component of the end-to-end latency is commonly referred to as "jitter."

Another issue is packet loss. Given that routers have only finite buffering (queuing) capacity, a sustained period of congestion may cause the buffer(s) to reach their capacity. When packets arrive to find buffer space exhausted, packets must be discarded until buffer space becomes available.

Clearly, you have a problem. The traditional router has, effectively, only a single queue for each internal congestion point (for example, in Figure 2.3 an output interface is draining the queue as fast as the interface speed allows) and no mechanism to isolate different classes or types of traffic from the effects of other traffic passing through it. The vagaries of the unrelated traffic passing through the shared queue at each internal congestion point is likely to have a heavy influence on each traffic stream’s latency, jitter, and packet loss. Some types of traffic (for example, TCP connections carrying email) tolerate latency better than they tolerate packet loss, suggesting that long queues are ideal. However, other types of traffic—for example, User Datagram Protocol (UDP) carrying streaming video or audio—prefer that packets be discarded if held too long by the network, suggesting that shorter queues are better.

**Figure 2.3**
First-in, first-out queuing on a Best Effort router.

Consider the scenario in Figure 2.3. Packets arrive from each input port at a maximum rate of \( Y_1 \) through to \( Y_n \) packets per second (pps). The outbound link extracts packets from the queue at \( X \) pps. Take the total input rate as \( Y \), the sum of \( (Y_1 + Y_2 + ... Y_n) \). When \( Y \) is less than \( X \), packets will not need to wait in the queue. However, it is more than likely that \( Y \) can burst well above \( X \); in which case, the queue sees a net growth in size. The number of packets (\( P \)) in the queue after some interval (\( T \)) is expressed as \( P = T \times (Y - X) \). A packet arriving at time \( T \) and finding the queue partially full experiences additional latency of \( X \times P \) seconds (because the packet must wait for the queue to drain at \( X \) pps). If a packet arrives when the queue is full (\( P = L \), the available queue space), the packet has nowhere to go and is dropped. Jitter comes from the fact that the components of \( Y \) are bursty and not correlated.

The preceding description also holds if you express the input and output rates in bits per second and the available queue space in bits (or bytes). If packets had a fixed length, a simple relationship would exist between the two forms of expression. However, in a typical IP environment packets are not of fixed length, adding further variability to the relationship between output link rate, the number of backlogged packets, and the latency experienced by backlogged packets.

---

**Note** - Latency can also be a function of the subnet technology—for example, the backoff scheme of Ethernet. However, backoff on Ethernet simply reveals itself as temporal unpredictability of the "link."

---

### 2.2.2 Classification, Queuing, and Scheduling

So what do you need to improve? The latency, jitter, and packet loss characteristics of any given IP network ultimately boil down to QoS characteristics of links and the dynamics of queue utilization.
and queue management within each router.

If network load exceeds service rate, a single queue at each internal congestion point is no longer sufficient. Instead, you need a queue for each identifiable class of traffic for which independent latency, jitter, and packet loss characteristics are required.

Each of these queues should have its own packet discard policies (for example, different thresholds beyond which packets are randomly or definitely discarded). Of course, the multiple queues per output interface are useless without a mechanism for assigning packets to the correct queues. A classification method is required over and above the router's traditional next-hop forwarding lookup. Finally, the queues must all share the finite capacity of the output link they feed into. This requirement implies the addition of a scheduling mechanism to interleave packets from each queue and, thus, mediate link access in a controllable and predictable manner.

For the purposes of this book, the preceding requirements can be captured as a statement that QoS-enabled networks require routers that can differentially Classify, Queue, and Schedule (CQS) all types of traffic as needed (see Figure 2.4). For the purposes of this book, such routers will be said to have a CQS architecture.

**Figure 2.4**
Per-hop Classify, Queue, and Schedule enables independent queuing and scheduling.

Later in this book, you look at various methods available for classifying traffic, comparing their relative complexities and the inherent granularity with which each scheme isolates different classes of traffic within an aggregate stream of packets. You also evaluate queuing schemes—the most important part of which is the queue's packet-dropping policy. These policies can range from simply dropping the most recently arrived packet when a queue reaches a hard limit (for example, it runs out of space) to making preemptive randomized drop decisions on the most recently arrived packet (based on how close the queue is to filling up and/or certain attributes carried within the packet itself). Finally, you consider the temporal effects of different scheduling algorithms on a network's capability to isolate different traffic classes from each other.

**2.2.3 Link-Level QoS**

Sometimes a router's scheduler must do more than simply interleave traffic at the IP packet level. The scheduler's capability to smoothly interleave traffic belonging to different queues depends on how quickly the outbound link can transmit each packet. For high-speed links (such as 155Mbit per second SONET or SDH circuits) a 1,500-byte IP packet takes less than 80 microseconds to transmit. This allows the scheduler to divide the link's bandwidth into slots up to 80 microseconds long—a very reasonable number, which drops to 20 microseconds on 622Mbit per second (OC-12 or STM-4) circuits. However, at the edges of the Internet many links are operating at 1Mbit per second or slower—in the 56 to 128Kbit per second range for Integrated Services Digital Network (ISDN) in North America and Europe and down to 28.8Kbit per second in the case of many dial-up modem connections.

A 1,500-byte IP packet takes around 94 milliseconds to transmit over a 128Kbit per second link, blocking the link completely during this time. Regardless of whether jitter-sensitive traffic has been classified into a different queue, those packets experience a 94-millisecond jitter when the scheduler pulls the 1,500-byte packet from another queue. Clearly, this poses some problems if QoS-sensitive
applications are to be supported on the far side of typical low-speed access links.

The basic solution is to perform additional segmentation of the IP packets at the link level in a manner transparent to the IP layer itself. The CQS architecture is then applied at the link level by queuing segments rather than whole packets, thus allowing the scheduler to interleave on segment boundaries (see Figure 2.5). By choosing the smaller segment size appropriately, such an approach enables jitter-sensitive IP traffic to avoid being backlogged behind long IP packets. (However, nothing is gained for free—segmentation decreases overall transmission efficiency because each segment carries its own header to allow later re-combination of segments.)

Figure 2.5
Segmentation before scheduling improves interleaving on low speed links.

Although ATM was originally designed for high-speed links, its design reflects a similar concern with minimizing the interval over which traffic on a given class could hold the link. The ATM cell is short by design, and each ATM switch is an example of a CQS architecture. Arriving cells are queued for transmission according to the contents of their virtual path identifier (VPI) and virtual channel identifier (VCI) header fields. Taken together the VPI/VCI identify the VC to which the cell belongs, encoding both path information (where should the cell go next) and service-class information. Good ATM switches have queues for each traffic class on a per-port basis and have schedulers feeding cells out each port in accordance with the bandwidth guarantees given to each class.

2.2.4 Analogies

A real-world example of the CQS process is available from the airline industry. Airport check-in areas typically utilize a form of CQS architecture to provide different levels of service to different classes of passengers. The congestion point is represented by a set of check-in agents who are processing passengers as quickly as possible and at a moderately consistent rate (ignoring for a moment the variability of processing time caused by difficult passengers!). The link speed of this congestion point is represented by the aggregate passenger processing rate of the check-in agents. (The airline can add and remove agents to vary this speed.)

The arrival of passengers for check-in is a bursty process, typically peaking during the hour before a flight's scheduled departure time. Most of us are very familiar with the queues that build up during the sudden arrival of a group of passengers. If you've arrived along with many other passengers, your wait for check-in can be quite long. If you've arrived during a lull in activity, you may be checked-in quite soon after entering the check-in area.

The airlines typically like to provide expedited check-in service to their premium customers (for example, first-class passengers, or those in the higher frequent flyer club status levels). To do so, separate lines (queues) are established prior to the check-in agents. The classification of passengers into the appropriate queue can take a number of forms. Sometimes the airlines leave it to the passengers themselves to pick the appropriate queue; at other times an airline representative performs a perfunctory ticket check and directs people to the queue appropriate to their ticket or frequent-flyer class. It is worth noting here that classification doesn't need to take into account every piece of available information, only the relevant information. For example, although the passenger's identity (name) is important information during the check-in process, someone's name is largely irrelevant to the queue assignment at check-in.
The act of pulling a passenger from one of the queues represents a scheduling decision. Typically, check-in agents are dedicated to each queue (class of passenger), providing a minimum rate of service to that queue regardless of any blockages affecting other queues. To achieve efficient usage of agents, when a high-priority queue empties, the associated agents usually begin (temporarily) processing passengers from the lower-priority queues. By appropriate distribution of check-in agents, the premium class of customers experience faster (shorter lines and, thus, lower latency) and more predictable (lower jitter) check-in service than those in lower classes.

A related real-world example of queuing and scheduling can be seen in the designs of major highways and freeways. Consider exit ramps, which are a form of output buffering for cars. Exit ramps feeding onto smaller roads typically terminate at a controlled intersection. The lights controlling the flow of traffic from the off-ramp into the local road act as a coarse scheduler. When cars begin exiting the highway faster than they are being fed onto the local road, the exit ramp itself begins to fill up. During the morning and evening peak traffic hours, many cars may be consistently arriving that the exit ramp overflows, causing traffic chaos on the main highway itself. Fortunately for drivers, cars are not "dropped" when the exit ramp overflows (although drivers may choose to continue on and search for another exit).

Finally, an example of classification and scheduling can be found at the toll booths that are sometimes placed across major highways. Typically a multilane highway fans out to many more toll lanes, and a self-classification process ensues as cars approach the tollbooths and pick their preferred lane. Particular lanes may be set aside for trucks, or priority lanes restricted to cars holding special electronic passes—motorists are advised of the appropriate self-classification rules prior to arriving at the tollbooths. (This example does not have an equivalent to controlled scheduling because each lane processes cars independently of other lanes.)

As you will see in the following chapters, CQS router architectures may be implemented in a number of permutations, each with its own specific consequences for the QoS characteristics of the IP network as a whole. The fundamental task of each router hop now becomes

- To know where to send the packet (conventional forwarding)
- To know when to send it (the additional QoS requirement)
- To complete the preceding tasks independently of other traffic sharing the router

### 2.3 Predictable Edge-to-Edge Behavior

As noted earlier, any end-to-end service is constructed from both the concatenation and layering of edge-to-edge and per-hop behaviors. Network operators, focusing on the edge-to-edge capabilities of the networks under their control, have a range of possible per-hop behaviors to mix and match together. Over the years a number of solution spaces have emerged, each one reflecting a different set of assumptions and compromises with respect to the CQS and routing capabilities of routers within the network.

The first and most important observation is that network designers face a trade-off between the number of traffic classes carried by their networks and the number of traffic classes that their router's CQS architectures can handle. A number of solutions are based on distributed edge-and-core
architectures, where the cores are fast routers with limited CQS capabilities and the edges are slower but with more advanced CQS capabilities.

A second observation is that the Internet’s existing shortest-path routing algorithms are not necessarily optimal for different classes of traffic across an arbitrary mesh of routers and links. A single metric may not be appropriate for all traffic traversing a particular section of the network. In addition, the destination-based forwarding paradigm itself makes it difficult to force subsets of available traffic into following alternative, non-shortest paths across any given network topology.

2.3.1 Edge-and-Core Models

Whether in hardware or software, the design of a good CQS architecture is generally nontrivial. In many software implementations, tight processing budgets make classification, queue management, and scheduling difficult to introduce without affecting the overall peak performance of the box. Hardware implementations have only just started to become commonplace—and until recently the development of a CQS implementation for IP locked into hardware was too commercially risky.

The edge-and-core model allows core routers to leverage hardware implementations (for speed), while leaving complex (but slower) processing to software-based edge routers. The edge routers might be able to classify and independently queue hundreds or thousands of traffic classes, whereas the core routers are assumed to be limited to a handful of queues.

Limited numbers of queues in core routers leads to a new requirement that edge routers be able to smooth out the burstiness of traffic entering the network. In the preceding discussion of per-hop QoS control, individual traffic classes were permitted to be completely unpredictable on the assumption that you could accurately isolate and reschedule them at every potential congestion point. However, although a smart edge/dumb core model may have the requisite isolation granularity at the edges, it does not in the core. Multiple traffic classes will find themselves aggregated into shared queues within the core routers. The potential for unpredictable mutual interference is high unless the network imposes some level of predictability before the traffic reaches the core routers. The solution is for edge routers to manipulate the temporal characteristics of individual traffic classes (and, hence, the aggregate of those traffic classes) before they enter the core. The Internet Engineering Task Force (IETF) Differentiated Services model is one example, and it will be discussed in Chapter 4, "Edge-to-Edge Network Models."

Shaping and Policing

The primary focus of a CQS architecture is the protection of traffic in each queue from the burstiness of traffic in another queue. On a per-hop basis, it is clear that, given appropriate isolation of all QoS-sensitive traffic into distinct queues, a scheduler needs to guarantee only a certain worst-case servicing interval (or minimum bandwidth). If spare capacity is available, you might expect "good" scheduler behavior to allocate that capacity to any queues having packets waiting to be forwarded. However, this practice is not always desirable from a network-wide perspective.

Simply emptying a queue as fast as the line rate allows (in the absence of traffic in other queues) can increase the burstiness perceived by routers further downstream. As a result, a serious problem can develop if the downstream routers do not differentiate traffic with as much granularity as the local router does. In addition, service providers may wish to cap the maximum rate that a customer can send packets through the network. If the customer frequently gets significantly better bandwidth than
the guaranteed minimum (perhaps because the network is new and/or under loaded), a perception issue surfaces: The customer begins to associate the typical performance with what he or she is paying for. If the spare capacity ever shrinks, the customer will receive edge-to-edge performance closer to the guaranteed minimum. However, the customer simply perceives the service to have degraded and is likely to complain loudly. Managing customer expectations is an important part of running a business, and in this case preemptive rate capping is one of the technology-based tools that may be employed.

Placing an upper bound on the maximum bandwidth (or minimum inter-packet interval) available to a traffic class is known as "traffic shaping." A shaping scheduler is configured to provide both a minimum service interval (the time between pulling packets from the same queue) and a maximum service interval (to guarantee the latency bound or minimum bandwidth). Packets arriving with a shorter inter-packet interval than allowed by the scheduler are queued until transmission—smoothing out the original burstiness. Figure 2.6 shows a scheduler that never samples the top queue more frequently than once every T seconds—no matter how closely bunched up the packets arrive, they are transmitted with at least T seconds between them. (A simple form of shaping scheduler is sometimes referred to as a "leaky bucket," because no matter how fast packets arrive they can only "leak out" at a fixed rate.)

**Figure 2.6**
Shaping requires a minimum scheduler time on certain queues.

A real-world example of shaping can be seen in some freeway entrance ramps. In California, for example, some entrance ramps are equipped with stop/go lights that alternate every few seconds. The net effect is to allow cars onto the freeway with a known minimum time interval between them—regardless of how bursty the car arrival times may have been to the entrance ramp itself. This input traffic shaping improves the capability of cars already on the freeway to interleave with the new traffic.

Shaping is not a simple function to introduce into a Best Effort router because this function presumes the existence of an appropriate CQS architecture. Although not quite so elegant, an alternative solution has been to introduce a packet-dropping behavior that is sensitive to excess burstiness of a traffic class. When too many packets arrive in too short an interval, packets are simply dropped. This process is known as policing.

Policing can be implemented without queues or schedulers, although it typically needs some form of classification to differentiate between the policing rules imposed on different traffic classes. In its simplest form, each traffic class has an associated counter. The counter is incremented regularly every T seconds and decremented whenever a packet (belonging to the counter's class) is forwarded. If a packet arrives to be transmitted when the counter is zero, the packet is dropped instead. When no packets are being transmitted, the counter increments up to a fixed limit L. The net effect is that a packet stream arriving with an average inter-packet interval of T seconds (or greater) passes through untouched. However, if a burst of more than L packets arrive in less than T seconds, the counter reaches zero and extra packets are dropped. The value of L affects the burst tolerance of the policing function, and T sets the rate below which traffic is safe. This practice is a severe, yet effective, way to modify the burstiness of traffic downstream from the policing router!

The utility of policing is based on the assumption that most bursty traffic originates from applications using adaptive end-to-end transport protocols such as TCP. Packet loss is assumed to indicate
transient congestion, and TCP reacts by slowing down the rate at which it injects packets into the network. Policing allows the network operator to fake the existence of transient congestion for a particular traffic class before it actually begins to occur further along the packet’s path. Even if the traffic class is not using an adaptive end-to-end transport protocol, policing protects the rest of the network by continuing to drop packets that exceed the allowed parameters.

Both shaping and policing are extremely useful tools for network designers who face a trade-off between the number of traffic classes carried by their networks and the number of traffic classes their network’s CQS architectures can handle. The basic issue is that individual traffic classes can be permitted to be unpredictable only if you can accurately isolate them at every potential congestion point. If you lack that isolation capability, you must attempt to impose some level of predictability prior to the potential congestion point. In the smart edge/dumb core model, the solution is for each edge router to preemptively shape and/or police the individual traffic classes before they enter the core to impose some overall order, smoothness, and predictability within each traffic class (and, hence, the aggregate of those traffic classes). Shaping may also be useful on the egress from a network in situations where the next network’s aggressive policing would be otherwise detrimental.

**Marking and Reordering**

Although shaping can be a sophisticated solution to smoothing out bursty traffic, simple policing is a blunt instrument. A number of variations have been introduced to soften the effect of edge-router policing. A policing node may choose to only "mark" packets (rather than discarding them immediately) if they exceed a burstiness threshold. Routers further along the path recognize these marked packets as having a lower priority than unmarked packets. If transient congestion begins to fill the queues in a downstream core router, its queue management algorithm can begin dropping marked packets before it begins dropping unmarked packets.

As an additional refinement, the original policing node may implement a staggered set of burst thresholds—if a packet burst exceeds the lower threshold, subsequent packets are marked and transmitted; if the burst continues and exceeds a higher threshold, packets are dropped. Alternatively, the policing node might implement multiple levels of "allowed" average packet arrival rates—a lower rate below which packets are forwarded unmarked, an intermediate range of rates within which packets are marked and forwarded, and an upper threshold above which packets are dropped. The impact on the core of the network is "softer" than would be achieved by a simple policing because many of the packets in the burst will have been marked instead of dropped. The advantage of such a scheme is that, in the absence of other network congestion in the core, this particular traffic class can utilize more of the available bandwidth.

Many algorithms can be invented to provide multiple marking levels and threshold calculations. However, network designers who plan on using edge marking of traffic also need to carefully choose their core routers. The main point of concern is potential reordering of marked packets relative to unmarked packets within a traffic class. This situation can happen if the core router uses two separate queues to differentiate between marked and unmarked packets in the same traffic class (see Figure 2.7). Because marked packets are of "lower priority," an implementation might choose to effect this relative priority by assigning more scheduler bandwidth to the queue of unmarked packets than for the queue of marked packets.

As a consequence a marked packet arriving before an unmarked packet in the same traffic class may
find itself scheduled for transmission after the unmarked packet (or vice-versa). Assuming the marked packet makes it all the way to the other end, the receiving application perceives the traffic to contain out-of-order packets.

Although the IP specifications do not preclude packets being reordered by the network, this practice should be avoided because most end-to-end protocols do not handle this case efficiently. In networks where marking is intended to increase a packet’s drop probability, the solution is not too difficult. Let the core router initially ignore the policing marker when classifying packets into queues, ensuring all packets in a traffic class are placed in one queue regardless of drop priority. Then modify the packet drop threshold for that queue on the basis of whether the packet is marked or not. The core router’s packet-dropping algorithm, thus, activates more aggressively for marked packets, achieving the desired edge-to-edge behavior.

**Figure 2.7**
Separate queues for marked and unmarked packets can lead to reordering.

### 2.3.2 Edge-to-Edge Routing

No particular restrictions dictate how routers and links are interconnected to form an IP network. As discussed in Chapter 1, the Internet’s shortest-path routing mechanisms are based on the assumption that a network’s topology is rarely static and must be tracked dynamically. In any realistic network, each router may have more than one output interface over which it could send a packet—the role of routing protocols is to establish a single interface over which a packet should be sent. To make the calculations tractable, the choice of appropriate interface has largely been driven by algorithms using only a single metric to define the shortest path.

However, two general concerns have been raised with this approach when it comes to supporting QoS. First is the argument that a single metric may not be appropriate for all traffic traversing a particular section of the network. Second, the destination-based forwarding paradigm itself makes it difficult to force subsets of available traffic into following alternative, non-shortest paths across any given network topology.

### QoS-Based Routing

QoS-based routing protocols attempt to take multiple metrics into account when building the network's forwarding tables. These protocols have been studied for years and often begin with an assumption that the network is built from conventional Best Effort IP routers. Starting from this assumption, single-metric routing is seen to have a number of limitations when attempting to meet the mixed QoS demands of a multiservice environment.

A metric can be considered a type of cost with each link (hop) having a cost associated with it. The routing protocols attempt to find paths with minimal total cost summed over all the links to possible destinations. However, this cost cannot represent the interests and needs of all traffic types. Should it represent the link’s latency, its available bandwidth, its packet loss probability, or perhaps the actual expense of sending packets over the link? Pick one. You will end up with some traffic finding the choice appropriate, whereas for other traffic the choice is wasteful of resources.

For example, consider a network where latency is the metric. Certainly the shortest path now suits applications with tight real-time requirements. But they are not alone. The network is most likely also
being used by traditional, bursty data applications that care significantly less about latency. The traffic from these other applications also follows the minimum latency shortest paths, adding to the load on the Best Effort routers along the path. An unfortunate side effect is that the bursty traffic consumes the same buffer space being used by the real-time traffic, increasing the jitter and average latencies experienced by all traffic through the routers. This approach also affects the accuracy of the latency costs that the routing protocols use to determine the shortest paths.

QoS-based routing creates multiple shortest-paths trees, covering the same actual topology of routers and links with each tree using different combinations of parameters as link metrics. The goal is to minimize unnecessary coexistence within routers of traffic with widely different QoS requirements. Packets with strict latency requirements are then forwarded by using the tree built with latency as a metric. Packet’s with non-real-time requirements might have a different tree built (for example, to minimize the financial cost of the path). Several practical issues exist with implementation of QoS-based routing:

- Each router needs to have multiple forwarding tables (or their functional equivalent) on which to perform each packet’s destination-based next-hop lookup, one for each type of shortest-path tree. Additional fields in the packet header are used to select one of the possible next hops associated with the packet’s destination address. This situation complicates the design of the next-hop lookup engine.

- An increase in routing protocol overhead occurs because the router’s CPU must support an instance of each protocol for each unique shortest-path tree. This requirement causes an increase in the time it takes for a network of such routers to converge after a transient in the network topology (for example, when links come or go or their costs somehow change). The convergence time increases further if the routing protocol is being asked to calculate trees based on multiple metrics simultaneously.

- Metrics such as latency or available bandwidth are highly dependent on the actual traffic flowing across the network. A shortest-path tree built with statically configured latency values could become outdated when traffic begins to flow across the network. The alternative, of updating each link’s cost with regular real-time measurements, poses a real control-theory nightmare—every cost update would result in a recalculation of the associated shortest-path tree, leading to continual processing load on all the routers.

Interestingly, the development of routers with CQS architectures somewhat reduces the need for QoS-based routing. For example, consider the example that uses latency as a cost metric. Now consider that every router has at least two queues per output interface—one for latency sensitive traffic and the other for all remaining traffic. All traffic is routed along lowest-latency paths. Assuming the routers appropriately classify traffic into the two queues, the service received by latency-sensitive traffic is independent of the burstiness of all other traffic types. Arguably then, any conventional, single-metric IP routing protocol, when coupled with routers based on a CQS architecture, can support multiple levels of service differentiation. The main caveat here is that sufficient capacity exists along the single tree to provide adequate service to all participants.

**Explicit Path Control**

The internal topologies of many networks are such that multiple paths can be found between most points. A major limitation of conventional IP forwarding is that single-metric, shortest-path trees use...
only one of the possible paths toward any given destination. Because lightly loaded alternative paths are not utilized, routers that exist on the shortest-path trees for many different network destinations can be subjected to high-average load—they become *hot spots*, potentially limiting the capability of the network to provide adequate service differentiation even if the router itself has a CQS architecture.

As the average load on a hot-spot router rises, the probability of random packet losses and jitter increases. Although this observation is most evident for networks containing regular Best Effort routers, it also holds true (albeit to a lesser degree) when the network consists of multiple queue routers. To combat this problem, a network operator has two alternatives:

- Upgrade the routers and links to operate faster
- Utilize additional packet-forwarding mechanisms that allow the traffic to be split across alternative paths (some of which may be just as short as the "official" shortest path and others that may be longer according to the prevailing metric)

When the network itself is built from cheap, low- to middle-bandwidth technology, the former approach may be entirely suitable. This description is most likely going to apply to enterprise environments where traffic growth has outpaced the deployed technology and a successor technology is easily deployable (for example, a 10Mbit per second Ethernet environment, where the upgrade to 100Mbit per second or 1Gbit per second Ethernet solutions are available).

Simply upgrading equipment and/or links may not be an option when your network is already pushing the limits of available technology. High-performance IP backbones have this problem—their routers are usually pushed hard to support OC-12 and OC-48 rate interfaces, and to buy or provision such circuits across today's traditional carrier infrastructures presents a serious problem. In addition, although prices are dropping for OC-12 and OC-48 circuits, they remain an expensive resource.

A preferable alternative is to build the equivalent aggregate capacity through parallelism—an IP topology rich in routers and lower-speed links across which the aggregate load can be distributed. Overriding shortest-path routing to more optimally utilize the underlying infrastructure of routers and links is often referred to as traffic engineering. Figure 2.8 shows a simplistic example.

**Figure 2.8**
Overriding a shortest-path route to balance the load.

Access networks A1 and A2 both source traffic to destination D, which is reachable through Access network A3. A3 has two attachment points to the IP backbone, through R6 and R5. Conventional IP forwarding would cause packets from both A1 and A2 to converge to the same forwarding path at interior/core router R3, and be forwarded to R6 (because the path is shorter than R3[→]R4[→]R5). A good way to reduce the average load on R6 is to force some portion of the load (for example, the packets coming from network A1) to follow the R3[→]R4[→]R5 path instead. A network operator may also want to override shortest-path forwarding for policy reasons (for example, the external link between R6 and A3 may have been funded solely by A2 and A3, and therefore A1's traffic must not be allowed to traverse it).

Traffic engineering through explicit path control is an important part of any solution to providing QoS, although the main impact is on the overall efficiency of the network itself, rather than directly
impinging on the end-users. This approach also raises an interesting routing question—having discarded the information being provided by the existing IP routing protocols, network operators need to supply an external source of information to control the traffic-engineered routing within their networks.

Explicit path control can be achieved in a variety of ways, either avoiding or permuting every router's conventional destination-based forwarding decision. The methods available at the IP level include

- Strict and loose source routing options
- Forwarding tables with lookup on the destination address and other fields in the IP packet header
- IP tunneling
- Multiprotocol Label Switching (MPLS)

In theory, an IP packet can have optional header fields added that specify (either explicitly or approximately) the sequence of routers through which the packet must pass on its way to the destination. However, most routers do not efficiently process packets carrying such optional header fields (the peak performance "fast path" through a router is typically optimized for packets having no additional headers). Packets with optional headers are processed in a parallel "slow path"—making this a poor choice if consistent QoS control is desired.

A slightly more feasible method is for the forwarding table to be constructed with regard not only for where the packet is going but also for where it has come. In this manner, it becomes possible to return different next-hop information for the same destination address just by taking the source address into account. However, this approach works only for a very constrained set of topologies and traffic-engineering scenarios. It is also expensive in terms of memory space in the forwarding tables.

**Traffic Engineering with IP Tunnels**

IP-IP tunneling forces the desired traffic patterns through the use of logical links. An IP packet is tunneled by placing it into the payload of another IP packet (the tunneling packet, as shown in Figure 2.9), which is then transmitted toward the desired tunnel endpoint. When the tunneling packet reaches its destination, the tunnel endpoint extracts the original IP packet and forwards it as though it had arrived over a regular interface.

**Figure 2.9**
Packet encapsulation for IP-IP tunneling.

Taking the network example in Figure 2.8, R1 would be configured to encapsulate traffic for D inside tunneling packets addressed to R5, and R2 would be configured to encapsulate traffic for D inside tunneling packets addressed to R6. Figure 2.10 shows the effective topology resulting from this arrangement.

**Figure 2.10**
Traffic engineering with IP-IP tunneling.
Several problems exist with this solution.

- Routers do not necessarily perform tunneling encapsulation and decapsulation in their "fast path"—this can be a major performance hit at the tunnel endpoints.

- The tunneling encapsulation (shown in Figure 2.9) adds overhead to each packet, reducing the Maximum Transmission Unit (MTU) that can be supported by the virtual link represented by the tunnel if fragmentation within the tunnel is to be avoided.

- The effective traffic engineering is very coarse—an IP-IP tunnel only allows control over the tunneled packet's final destination (the egress routers R5 or R6 in this example). The tunneling packet takes the shortest path across the backbone to R5 or R6 as appropriate.

### Traffic Engineering with Label-Switched Paths

**Multiprotocol Label Switching (MPLS)** is discussed in some detail later in this book, but it is worth noting here that the primary role of MPLS for service providers is traffic engineering. MPLS is a connection-oriented form of IP networking—packets have labels added and are forwarded along preconstructed label-switched paths (LSPs) by routers modified to switch MPLS frames (label-switching routers, LSRs).

LSPs can mimic the IP-IP tunnels in Figure 2.10—one LSP between R1 and R5, and another LSP between R2 and R6. R1 would be configured to label all traffic for D with the label corresponding to the first hop of the LSP from R1 to R5. R2 would be configured to label all traffic for D with the label corresponding to the first hop of the LSP from R2 to R6.

The effective topology in Figure 2.11 between the edge LSRs is identical to that in Figure 2.10, but the solutions are different. First, the overhead per packet is reduced (an MPLS header is 4 bytes, compared to 20+ bytes for a complete encapsulating IP header). Second, the packet's actual hop-by-hop path within the backbone is under the control of the network operator when the LSP is established.

**Figure 2.11**
Traffic engineering with explicitly routed label-switched paths.

LSP and ATM VC are similar in many ways. Backbone operators who use ATM to transport their wide area IP traffic already utilize explicitly routed permanent virtual connections (PVCs) between the edges of their ATM networks and rely on the edge routers to map the correct traffic onto the appropriate PVCs. For many service providers, the move to MPLS is simply a generalization of ATM but with variable-length packets instead of fixed-length cells.

### 2.4 Signaling

Assuming that you can provide differentiated queuing and scheduling on a per-hop basis and have the appropriately controllable underlying link layers, the question becomes one of establishing and modifying the network's actual behavior. This matter requires coordination of the actual (rather than theoretical) behaviors along each path. A generic term for this process is **signaling**—the act of
informing each hop along a path (or paths) how to recognize traffic for which a special processing behavior is required and the type of special processing required.

Signaling can be achieved in a number of ways with varying degrees of timeliness, flexibility, and human intervention (not all of which are conventionally considered signaling per se). At one extreme sits dynamic edge-to-edge signaling, where the network is informed each and every time a new class of traffic requires specific support.

The network itself responds on demand by internally establishing additional information (or modifying existing information) at each hop to achieve the requested edge-to-edge behavior. Examples of on-demand signaling include ATM’s User Network Interface (UNI) and Network Node Interface (NNI) signaling protocols and the IETF’s Resource Reservation Protocol (RSVP).

New network technologies frequently either do not have fully dynamic signaling protocols defined or have not matured to the point where reliable implementations of their signaling protocols exist. Under such circumstances the networks are usually provisioned for new services—often entailing human intervention to configure (or reconfigure) the controllers of the links and nodes along the affected paths. Provisioning is a form of signaling, even though the response time is usually orders of magnitude slower than dynamic signaling.

Because the number of links and nodes in a network can be quite large, many vendors are developing centralized controllers or servers where configuration or provisioning actually occurs. These controllers then automatically distribute the appropriate rules to the links and nodes in the network on behalf of the human operator. Designs that are more advanced allow these controllers to automatically react to changing network conditions in accordance with general policies that may be imposed by the human operator. Although such centralized schemes also constitute a dynamic mechanism, they differ from edge-to-edge signaling in that it is not user controlled.

The Internet has used a form of dynamic signaling for years—its routing protocols. Although most of us have been conditioned to think of signaling and routing as distinct activities, protocols such as Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP) are the Internet's mechanisms for signaling topology changes. These mechanisms ensure the construction of up-to-date forwarding tables that reflect the best set of shortest paths across the network and adapt dynamically to changing topological conditions. Therefore, these mechanisms (that is, OSPF and BGP) qualify as signaling protocols. However, their focus is internal to the network itself, and their actions are generally not explicitly triggered by some user's request. Furthermore, their actions are designed primarily to effect the construction of paths, not the allocation of resources or priority processing for specific traffic along those paths.

Typically, signaling in the IP context is thought of as the additional actions required to establish a particular edge-to-edge QoS over and above the default Best Effort QoS. As previously noted this process can involve dynamic or provisioned behaviors (or some combination). In all cases, the process of establishing a desired edge-to-edge QoS requires careful balancing of existing per-hop resources and network-wide paths.

When a signaling request states a particular QoS goal, there are a number of variables to consider. In theory, both the path and the resources along the path are open to modification. For a given path, the signaling protocol should determine whether resources (for example, queuing space or share of link bandwidth) are available along that path. If the first path checked does not support the desired QoS,
an ideal signaling protocol would find another path and try again. As an example, ATM’s Private Network Node Interface (PNNI) signaling tries different paths until it finds one that can support the requested edge-to-edge QoS. Trying alternative paths presumes that the network has the capability to force traffic along the path that is discovered to be capable of supporting the desired QoS.

In conventional connectionless IP, however, traffic must follow the shortest-path trees established by the routing protocols (using whatever metric is specified by the network operator). As a consequence the IETF developed its RSVP signaling mechanism to simply follow (and adapt to) whatever routing exists in the network without attempting to discover alternative, non-shortest-path routes that might better support the requested QoS. Inherently a trade-off, RSVP avoids any reengineering of existing IP networks, doesn’t reinvent or replicate the actions of the existing IP routing protocols, and can be introduced as a simple hardware or software upgrade to existing routers. However, if resources are exhausted along a particular shortest path, no simple way exists for RSVP to force traffic along a longer, but perhaps more lightly loaded path.

Another issue with signaling is the amount of additional state information the routers must carry. State information is anything that the router needs to characterize the special traffic—for example, IP header information on which to classify the packets—and to process the special traffic—for example, associated queue(s), packet-drop parameters, and scheduler priorities or weights. The routing and forwarding tables already held by each router represent topological state information—adding signaling for QoS only increases the number of tables consuming valuable memory in routers.

Any realistic QoS solution for IP networking must cope with the often conflicting demands that it be easy to implement, not send the amount of state information skyrocketing, optimize the use of network resources, adapt dynamically to routing changes, and work in a world where routing and signaling are decoupled.

2.5 Policies, Authentication, and Billing

If I offer you a first-class seat on the plane for the price of an economy-class seat, you would take it, right? At worst you might wonder what the catch is, but in the end nobody refuses better service if it costs no more than worse service—whether the comfort of an airplane's seating, the speed of package delivery from a shipping service, or Internet access from a local ISP is being discussed. Of course, a practical problem immediately arises if you're not being charged a premium price for the premium service—everybody else wants it, too.

Any networking technology that offers differentiation of service levels must also address the need to differentiate each user's right to use particular service levels. If everyone has a right to use the best service level at the same time, the resources would either run out, or the network would have to be engineered to cope. In general the network's resources are limited at various service levels, and so the task is one of allowing or disallowing particular users access to service levels based on their right to use. (If the network were engineered to handle everyone asking for premium service, without any differential impact on the cost of running the network, what would be the point of offering lesser service levels?) This right to use can be established in a number of ways—for example, payment of fees (financial cost) or administrative assignment (ranking of the user's importance). A commercial service provider would be inclined to utilize a fee basis—you get the service you pay for. A corporate enterprise network may determine service allocations based at least in part on the status of each user (or the user's department) within the company.
The whole issue of establishing and monitoring a user’s right to use certain service levels opens up a can of worms that the Internet industry is only beginning to address. First are questions of policy (identifying the service classes that particular users are entitled to negotiate). Second is the problem of authentication (proving that the entity currently using the network is the claimed user, either during right-to-use negotiations or subsequent traffic transmission). Third is the question of billing (extracting the fee from the correct user) if fees are used to establish the right to use. Billing is even of interest to enterprise networks, where it may be used to provide additional granularity of usage control beyond the corporate status of a user or the user’s department.

All three issues are also tightly coupled to the network’s signaling because the network’s signaling system must establish the requested edge-to-edge service levels and associate them with traffic coming from the user. If the users are utilizing dynamic, edge-to-edge signaling to negotiate their right to use, the signaling protocol itself must be tightly coupled with the policy, authentication, and billing mechanisms.

Human nature being what it is, the network must be capable of authenticating any user’s request for, and use of, particular service levels. Users must not be billed for services they don’t request, lest the operator finds itself in a court of law or being lambasted in the media (perhaps a worse fate for a service provider trying to garner the trust of the market place!). Of course, users must also be accurately billed for the service levels they do request. If the operator’s fee structure is based in some part on the actual amount of usage, the consumption of services must also be tracked and authenticated.

If dynamic, edge-to-edge user-signaling protocols (such as RSVP) are to be used in fee-for-use environments, these protocols clearly need to incorporate sufficiently strong user authentication fields. (An operator might attempt to deduce a user’s identity from physical attachment points on the network, but in an age of dial-up IP access and mobile nodes this approach is rarely effective.) In the absence of such capabilities, the user and service provider are forced to rely on more traditional or manual channels to negotiate service levels (the fax, phone, or postal service). Alternatively, the service provider can simply hope users don’t go around impersonating each other when ordering service levels.

Enterprise environments are typically more structured and controlled, and in these environments authentication based solely on the node’s topological position might be quite feasible. However, if the enterprise network includes mobile nodes or any likelihood that users will move around the network’s topology, it will need to consider the same issues faced by a commercial service provider.

Two problems develop if the service provider decides to incorporate a usage-based component in the right-to-use fee. First, no clear industry consensus has emerged on what constitutes a realistic metric for use—is it simple packet counts, burstiness, peak or mean bandwidths, or some complex measurement of delivered latency and jitter?

Second, after you decide on a metric that you think the customers will understand, you face the problem of accurately measuring it in your network and reliably associating your measurements to particular users. Real-time measurement of traffic patterns is a major problem because it requires significant processing capabilities and needs to be undertaken for each and every instance of a distinct, user-defined traffic class.
This book cannot hope to cover the emerging solutions to policy management, user authentication, and billing models. However, you will be left with an understanding of the roles played by these important components of a total IP QoS solution and have the ability to assess whatever the industry offers.

© Copyright Pearson Education. All rights reserved.