

# **PART V**

## **Other Aspects Of Computer Networking**

### **Network Performance, QoS, Security, Management, And Emerging Technologies**

#### **Chapters**

- 28 Network Performance (QoS and DiffServ)**
- 29 Multimedia And IP Telephony (VoIP)**
- 30 Network Security**
- 31 Network Management (SNMP)**
- 32 Trends In Networking Technologies And Uses**

## *Chapter Contents*

- 28.1 Introduction, 471
- 28.2 Measures Of Performance, 471
- 28.3 Latency Or Delay, 472
- 28.4 Throughput, Capacity, And Goodput, 474
- 28.5 Understanding Throughput And Delay, 475
- 28.6 Jitter, 476
- 28.7 The Relationship Between Delay And Throughput, 477
- 28.8 Measuring Delay, Throughput, And Jitter, 478
- 28.9 Passive Measurement, Small Packets, And NetFlow, 480
- 28.10 Quality Of Service (QoS), 481
- 28.11 Fine-Grain And Coarse-Grain QoS, 482
- 28.12 Implementation Of QoS, 484
- 28.13 Internet QoS Technologies, 486
- 28.14 Summary, 487

# 28

## *Network Performance (QoS and DiffServ)*

### 28.1 Introduction

Early chapters consider the fundamental properties of data communications systems, and discuss the relationships among signals, frequencies, bandwidth, channel coding, and data transmission. The chapters explain measures of underlying data transmission systems, discuss data network size, and explain that each networking technology is classified as a PAN, LAN, MAN, or WAN.

This chapter continues the discussion by considering the topic of network performance. The chapter discusses quantitative measures of networks, and explains how protocols and packet forwarding technologies can implement mechanisms that provide priority for some traffic.

### 28.2 Measures Of Performance

Informally, we use the term *speed* to describe network performance, and refer to *low-speed* or *high-speed* networks. However, such definitions are inadequate because network technologies change so rapidly that a network classified as “high speed” can become medium or low speed in as little as three or four years. Thus, in place of qualitative descriptions, scientists and engineers use formal, quantitative measures to specify network performance precisely. After reviewing basic measures, we will explain how they are used to implement tiered services. Although beginners often prefer informal descriptions, quantitative measures are important because they make it possible to compare the exact features of two networks and to build mechanisms that provide higher

priority for some traffic. Figure 28.1 lists the major measures of network performance, and successive sections explain each.

Measure	Description
Latency (delay)	The time required to transfer data across a network
Throughput (capacity)	The amount of data that can be transferred per unit time
Jitter (variability)	The changes in delay that occur and the duration of the changes

**Figure 28.1** Key measures of data network performance.

### 28.3 Latency Or Delay

The first property of networks that can be measured quantitatively is *latency* or *delay*. Latency specifies how long it takes for data to travel across a network from one computer to another; it is measured in fractions of seconds. Delays across the Internet depend on the underlying infrastructure as well as the location of the specific pair of computers that communicate. Although users care about the total delay of a network, engineers need more precise measurements. Thus, engineers usually report both the maximum and average delays, and divide a delay into several constituent parts. Figure 28.2 lists the various types of delay.

Type	Explanation
Propagation Delay	The time required for a signal to travel across a transmission medium
Access Delay	The time needed to obtain access to a transmission medium (e.g., a cable)
Switching Delay	The time required to forward a packet
Queuing Delay	The time a packet spends in the memory of a switch or router waiting to be selected for transmission
Server Delay	The time required for a server to respond to a request and send a response

**Figure 28.2** Various types of delay and an explanation of each.

*Propagation Delay.* Some delay in a network arises because a signal requires a small amount of time to travel across a transmission medium. In general, propagation delays are proportional to the distance spanned. Even with long cable runs, a typical LAN used within a single building has a propagation delay under a millisecond. Although such delays seem irrelevant to a human, a modern computer can execute over one hundred thousand instructions in a millisecond. Thus, a millisecond delay is significant when a set of computers need to coordinate (e.g., in the financial industry, where the exact time a stock order arrives determines whether an order is accepted). A network that uses a GEO satellite has much higher delay — even at the speed of light, it takes hundreds of milliseconds for a bit to travel to the satellite and back to earth.

*Access Delay.* Many networks use shared media. The set of computers that share a medium must contend for access. For example, a Wi-Fi wireless network uses a CSMA/CA approach to medium access. Such delays are known as *access delays*. Access delays depend on the number of stations that contend for access and the amount of traffic each station sends. Access delays remain small and fixed unless the medium is overloaded.

*Switching Delay.* An electronic device in a network (e.g., a Layer 2 switch or router) must compute a next-hop for each packet before transmitting the packet over an output interface. The computation often involves table lookup, which means memory access. In some devices, additional time is needed to send the packet over an internal communication mechanism such as a bus or fabric. The time required to compute a next hop and begin transmission is known as a *switching delay*. Fast CPUs and special-purpose hardware have made switching delays among the least significant delays in a computer network.

*Queuing Delay.* The store-and-forward paradigm used in packet switching means that a device such as a router collects the bits of a packet, places them in memory, chooses a next-hop, and then waits until the packet can be sent before beginning transmission. Such delays are known as *queuing delays*. In the simplest case, a packet is placed in a FIFO output queue, and the packet only needs to wait until packets that arrived earlier are sent; more complex systems implement a selection algorithm that gives priority to some packets. Queuing delays are variable — the size of a queue depends entirely on the amount of traffic that has arrived recently. Queuing delays account for most delays in the Internet. When queuing delays become large, we say that the network is congested.

*Server Delay.* Although not part of a network per se, servers are essential to most communication. The time required for a server to examine a request and compute and send a response constitutes a significant part of overall delay. Servers queue incoming requests, which means that server delay is variable and depends on the current load. In many cases, a user's perception of Internet delay arises from server delay rather than network delays.

## 28.4 Throughput, Capacity, And Goodput

A second fundamental property of networks that can be measured quantitatively is the *capacity* of a network, which is often expressed as the maximum *throughput* that the network can sustain. Throughput is a measure of the rate at which data can be sent through the network, specified in *bits per second (bps)*. Most data communication networks offer a throughput rate of more than 1 Mbps, and the highest-speed networks operate faster than 1 Gbps. As we have seen, however, special cases arise where a network has throughput less than 1 Kbps.

Because throughput can be measured several ways, one must be careful to specify exactly what has been measured. There are several possibilities:

- Capacity of a single channel
- Aggregate capacity of all channels
- Theoretical capacity of the underlying hardware
- Effective data rate achieved by an application (goodput)

Vendors often advertise the theoretical capacity of their equipment and the throughput achieved under optimal conditions. The hardware capacity is often cited as an approximation of the potential throughput because the capacity gives an upper bound on performance — it is impossible for a user to send data faster than the rate at which the hardware can transfer bits.

Users do not care about the capacity of the underlying hardware — they are only interested in the rate at which data can be transferred. Users typically assess the *effective data rate* that an application achieves by measuring the amount of data transferred per unit time; the term *goodput* is sometimes used to describe the measure. The goodput rate is less than the capacity of the hardware because protocols impose overhead — some network capacity is not available to user data because protocols:

- Send packet headers, trailers, and control information
- Impose a limit on the window size (receive buffer)
- Use protocols to resolve names and addresses
- Use a handshake to initiate and terminate communication
- Reduce the transmission rate when congestion is detected
- Retransmit lost packets

The disadvantage of using goodput as a measure arises because the amount of overhead depends on the protocol stack being used. In addition to Transport, Internet, and Layer 2 protocols, goodput depends on the application protocol. For example, consider using the *File Transfer Protocol (FTP)* to measure goodput across an Ethernet. FTP uses TCP, which uses IP. Furthermore, FTP does not compress data before transmission. Instead, FTP places user data in TCP segments, TCP encapsulates each segment in an IP datagram, and IP encapsulates each datagram in an Ethernet frame.

Thus, each frame has an Ethernet header and CRC field, an IP datagram header, and a TCP header. If a user chooses an alternative file transfer application or an alternative protocol stack is used, the goodput may change. The point is:

*Although it provides a measure of the effective rate at which data can be transferred over a network, the goodput depends on the application.*

## 28.5 Understanding Throughput And Delay

In practice, the terminology that networking professionals use to describe network throughput or network capacity can be confusing. For example, chapters on data communications define the bandwidth of a channel, and explain the relationship between the hardware bandwidth and the maximum data rate. Unfortunately, networking professionals often use the terms *bandwidth* and *speed* as synonyms for throughput. Thus, one might hear someone say that a particular network has a “speed of 1 Gbps.” Alternatively, some advertisements use the phrase “bandwidth of 1 Gbps.” In an attempt to distinguish between the two uses of *bandwidth*, engineers reserve *bandwidth* to mean *analog bandwidth*, and use the term *digital bandwidth* as a synonym for *throughput*. Although such statements are common, they can be confusing because throughput, delay, and bandwidth are separate properties.

In fact, throughput is a measure of capacity, not speed. To understand the relationship, imagine a network to be a road between two locations and packets traveling across the network to be cars traveling down the road. The throughput determines how many cars can enter the road each second, and the propagation delay determines how long it takes a single car to travel the road from one town to another. For example, a road that can accept one car every five seconds has a throughput of *0.2* cars per second. If a car requires *30* seconds to traverse the entire road, the road has a propagation delay of *30* seconds. Now consider what happens if a second lane is opened on the road (i.e., the capacity doubles). It will be possible for two cars to enter every five seconds, so the throughput has doubled to *0.4* cars per second. Of course, the *30* second delay will remain unchanged because each car must still traverse the entire distance. Thus, when thinking about measures of networks, remember that:

*Propagation delay specifies the time a single bit remains in transit in a network. Throughput, which specifies how many bits can enter the network per unit time, measures network capacity.*

Networking professionals have an interesting aphorism:

*You can always buy more throughput, but you cannot buy lower delay.*

The analogy to a road helps explain the aphorism: adding more lanes to a road will increase the number of cars that can enter the road per unit of time, but will not decrease the total time required to traverse the road. Networks follow the same pattern: adding more parallel transmission paths will increase the throughput of the network, but the propagation delay, which depends on the distance spanned, will not decrease.

## 28.6 Jitter

A third measure of networks is becoming important as networks are used for the transmission of real-time voice and video. The measure, which is known as a network's *jitter*, assesses the variance in delay. Two networks can have the same average delay, but different values of jitter. In particular, if all packets that traverse a given network have exactly the same delay,  $D$ , the network has no jitter. However, if packets alternate between a delay of  $D + \epsilon$  and  $D - \epsilon$ , the network has the same average delay, but has a nonzero jitter.

To understand why jitter is important, consider sending voice over a network. On the sending side, the analog signal is sampled and digitized and an eight-bit digital value is emitted every 125  $\mu$ seconds. The samples are collected into packets, which are then transferred across the network. At the receiving side, the digital values are extracted and converted back to analog output. If the network has zero jitter (i.e., each packet takes exactly the same time to transit the network), the audio output will exactly match the original input; otherwise, the output will be flawed. There are two general approaches to handling jitter:

- Design an isochronous network with no jitter
- Use a protocol that compensates for jitter

A traditional telephone system uses the first approach: the phone system implements an *isochronous network* which guarantees the delay along all paths is the same. Thus, if digitized data from a phone call is transmitted over two paths, the hardware is configured so that both paths have exactly the same delay.

Transmission of voice or video over the Internet takes the second approach: although the underlying network may have substantial jitter, voice and video applications rely on *real-time protocols* to compensate for jitter<sup>†</sup>. Because using real-time protocols is much less expensive than building an isochronous network, phone companies are relaxing the strict requirements for isochrony. Of course, a protocol cannot compensate for arbitrary jitter — if the variance in delay becomes excessive, output will be affected. Thus, even when using the second approach, service providers attempt to minimize jitter in their networks.

---

<sup>†</sup>The next chapter discusses the transmission of real-time data over the Internet.

## 28.7 The Relationship Between Delay And Throughput

In theory, the delay and throughput of a network are independent. In practice, however, they can be related. To understand why, think of the road analogy discussed above. If cars enter the road at even time intervals, cars traveling along the road at uniform speed are spaced at uniform intervals. If a car slows down for any reason (e.g., at an intersection), others behind it will slow down as well, causing temporary traffic congestion. Cars that enter the road when congestion is occurring will experience longer delays than cars traveling on an uncongested road. A similar situation occurs in networks. If a router has a queue of packets waiting when a new packet arrives, the new packet will be placed on the tail of the queue, and will need to wait while the router forwards the previous packets. If congestion occurs, the packets will experience longer delays than data entering an idle network.

### 28.7.1 Utilization As An Estimate Of Delay

Computer scientists have studied the relationship between delay and congestion, and have found that in many cases, the expected delay can be estimated from the current percentage of the network capacity being used. If  $D_0$  denotes the delay when a network is idle, and  $U$  is a value between 0 and 1 that denotes the current *utilization*, the effective delay,  $D$ , is given by a simple formula:

$$D = \frac{D_0}{(1 - U)} \quad (28.1)$$

When a network is completely idle,  $U$  is zero, and the effective delay is  $D_0$ . When a network operates at 1/2 its capacity, the effective delay doubles. As traffic approaches the network capacity (i.e., as  $U$  becomes close to 1), the delay approaches infinity. Although the formula only provides an estimate of effective delay, we can conclude:

*Throughput and delay are not completely independent. As traffic in a computer network increases, delays increase; a network that operates at close to 100% of its throughput capacity experiences severe delay.*

In practice, network managers understand that extremely high utilization can produce disastrous delay. Thus, most managers work to keep utilization low, and measure the traffic on each network constantly. When the average or peak utilization begins to climb above a preset threshold, the manager increases the capacity of the network. For example, if utilization becomes high on a 100 Mbps Ethernet, the manager might choose to replace it with a Gigabit Ethernet. Alternatively, the manager might choose to divide a network in two, placing half the hosts on one network and half the hosts on the other (such division is extremely easy with a VLAN switch).

How high should the utilization threshold be? There is no simple answer; many managers choose a conservative value. For example, one major ISP that runs a large backbone network keeps utilization on all its digital circuits under 50%. Others set thresholds at 80% to save money. In any case, managers generally agree that a network should not be operated above 90% of capacity.

### 28.7.2 Delay-Throughput Product

Once a network's delay and throughput are known, it is possible to compute another interesting quantity, the *delay-throughput product*<sup>†</sup>. To understand the meaning of the delay-throughput product, think of the road analogy: when cars are entering a road at a fixed rate of  $T$  cars per second and it takes a car  $D$  seconds to traverse the road, then  $T \times D$  additional cars will enter the road by the time the first car has made a complete trip. Thus, a total of  $T \times D$  cars can be on the road. In terms of networks, the number of bits traveling through a network at any time is given by:

$$\text{Bits present in a network} = D \times T \quad (28.2)$$

where  $D$  is the delay measured in seconds, and  $T$  is the throughput measured in bits per second. To summarize:

*The product of delay and throughput measures the volume of data that can be present on the network. A network with throughput  $T$  and delay  $D$  can have a total of  $T \times D$  bits in transit at any time.*

The delay-throughput product is important for any network with especially long delay or large throughput because it affects transmission — a sending application can transmit a large volume of data before the destination receives the first bit.

## 28.8 Measuring Delay, Throughput, And Jitter

The techniques used to measure throughput and jitter are relatively straightforward. To assess throughput, a sender transfers a large volume of data. A receiver records the time from the start of data arriving until all data has arrived, and calculates the throughput as the amount of data sent per unit of time. The technique for measuring jitter is known as a *packet train*: a sender emits a series of packets with a small, fixed delay between packets. Typically, packets in the train are sent back-to-back. A receiver records the time at which each packet arrives, and uses the sequence of times to compute the differences in delay.

Unlike measurements of throughput or jitter, a precise measurement of the delay on a path from host  $A$  to host  $B$  requires that the two hosts have synchronized clocks.

---

<sup>†</sup>When used as a measure of the underlying hardware, the delay-throughput product is often called the *delay-bandwidth product*.

Furthermore, to measure delay over a short distance (e.g., a LAN), the clocks must be extremely accurate. Instead of using synchronized clocks, many network measurement tools choose an easier approach: measure the round-trip time and divide by two. For example, *ping* can be used.

Measuring network performance can be surprisingly difficult for four reasons:

- Routes can be asymmetric
- Conditions change rapidly
- Measurement can affect performance
- Traffic is bursty

The first point explains why it may not be possible to use round-trip times to approximate delay. Asymmetric routing means that the delay along a path from *B* to *A* can differ substantially from the delay along a path from *A* to *B*. Thus, one-half of the round-trip time may not give an accurate measure.

The second point explains why an accurate measure of network performance can be difficult to obtain: conditions change rapidly. For example, consider a shared network. If only one host is sending data, the host will enjoy low delay, high throughput, and low jitter. As other hosts begin to use the network, utilization increases, which will increase delay and jitter and decrease throughput. Furthermore, because conditions change rapidly, delays can vary widely in as little as a second. Thus, even if measurements are taken every ten seconds, a measurement can miss a major shift in performance.

The third point suggests that sending test traffic to measure a network can affect network performance. On the PlanetLab research testbed, for example, so many researchers used ping to measure performance that ping traffic completely dominated other traffic. The situation became so severe that administrators established a policy to discourage the use of ping.

The fourth point is fundamental: data networks exhibit *bursty* behavior, which means that traffic is uneven. If we consider the traffic sent by a given host computer, the pattern of burstiness is obvious: most hosts remain silent until a user runs an application that communicates over the Internet. When a user enters a URL in a web browser, the browser fetches all parts of the page, and then stops communicating until the user requests another page. Similarly, if a user downloads email, the host computer communicates with an email system, downloads a copy of the user's mailbox, and then waits for the user.

Interestingly, aggregate data traffic is also bursty. One might expect that the burstiness is a local phenomenon and when traffic from millions of Internet users is combined, the result will be a smooth pattern of use. After all, users do not all read email at exactly the same time; so, while one user is downloading, another user might be reading email that was downloaded previously. Indeed, measurements of the voice

telephone network show that telephone traffic from millions of users results in a smooth aggregate. When the traffic from a million Internet users is combined, however, the result is not a smooth aggregate. Instead, the aggregate is bursty in the sense that the total traffic has peaks and low points. In fact, statisticians say that data traffic is *self similar*, which means that the traffic is analogous to a *fractal*, where the same statistics profile is evident at any granularity. Thus, if an enterprise examines a LAN, traffic from local hosts will appear bursty. If an intermediate ISP measures traffic from one thousand users or a large ISP measures traffic from ten million users, the traffic will have large absolute quantities, but will exhibit the same overall statistical pattern as the traffic on a LAN.

We can summarize:

*Unlike voice telephone traffic, data traffic is bursty. Data traffic is said to be self-similar because aggregates of data traffic exhibit the same pattern of burstiness.*

## 28.9 Passive Measurement, Small Packets, And NetFlow

Network managers who measure networks distinguish between two forms of measurement:

- Active
- Passive

We have discussed the disadvantage of *active* measurement techniques: by injecting traffic into a network, the measurement traffic can change the performance of the network. The alternative is *passive* measurement that monitors a network and counts packets, but does not inject additional traffic. For example, an ISP can count the bytes that are transferred over a link in a given amount of time to produce an estimate of the link utilization. That is, the ISP arranges a passive monitor station that observes a network over an interval of time and accumulates a total of the bytes in all packets.

Interestingly, an ISP may choose to measure the number of packets sent as well as the number of data bytes. To understand why, observe that because link utilization is measured as a percentage of capacity and capacity is measured in bits per second, an ISP needs to measure the total data bits sent per unit time. However, the capacity of switches and routers is measured in packets per second. That is, because a router or switch performs next-hop forwarding once per packet, the computational effort expended is proportional to the number of packets processed rather than the number of bits in a packet. If a stream of data arrives at 1 Gbps, a switch or router performs less work if the stream is divided into a few large packets than it does if the stream is divided into many small packets. Networking equipment vendors understand the concept, and some

vendors make performance claims about the data rate rather than the packet rate (i.e., they measure the performance of their products using large packets).

We can summarize:

*To assess link utilization, an ISP measures the total data transferred over a link per unit time; to assess the impact on a router or switch, an ISP measures the number of packets transferred per unit time.*

One of the most widely used passive measurement techniques was originally created by Cisco and is now an IETF standard: *NetFlow*. A router that implements NetFlow statistically samples packets according to parameters established by the network administrator (e.g., samples one of every one-thousand packets). Information is extracted from the header of each sampled packet, the information is summarized, and the summary is sent to a network management system where it is processed (often, data is saved on disk for later analysis). Typically, NetFlow extracts source and destination IP addresses, the datagram type, and protocol port numbers. To insure that it is passive, a router running NetFlow must send the NetFlow summaries over a special management port rather than route them across one of the networks that handles user data.

## 28.10 Quality Of Service (QoS)

The counterpart of network measurement is *network provisioning*: designing a network to provide a specific level of service. The remainder of the chapter considers mechanisms that can be used to implement service guarantees. Broadly, the topic is known as *Quality of Service (QoS)*.

To understand QoS, consider the contract between a service provider and a customer. The simplest contracts define a service by specifying the data rate that the provider guarantees. For example, a provider that offers a DSL connection to the Internet might guarantee a data rate of 2.2 Mbps. More complex contracts define *tiered services*, where the level of service received depends on the amount paid. For example, a provider might choose a *priority* approach that guarantees packets from a customer who subscribes to the platinum level of service will have priority over packets from customers who subscribe to a silver level of service.

Large corporate customers often demand more stringent *service guarantees*. The financial industry typically creates service contracts with bounds on the delay between specific locations. For example, a brokerage firm might need a service contract that specifies packets must be transferred from the company's main office to the New York Stock Exchange in less than 10 milliseconds; a company that backs up their entire data center each night might need a service contract that guarantees a throughput of not less than 1 Gbps on the TCP connections used for backup.

## 28.11 Fine-Grain And Coarse-Grain QoS

How can a provider specify QoS guarantees, and what technologies does a provider use to enforce QoS? Figure 28.3 lists the two general approaches that have been proposed for service specification. As the figure indicates, the approaches differ in their granularity and whether a provider or a customer selects parameters.

Approach	Description
<b>Fine-Grain</b>	<b>A provider allows a customer to state specific QoS requirements for a given instance of communication; a customer makes a request each time a flow is created (e.g., for each TCP connection)</b>
<b>Coarse-Grain</b>	<b>A provider specifies a few broad classes of service that are each suitable for one type of traffic; a customer must fit all traffic into the classes</b>

**Figure 28.3** Two approaches that have been proposed for specification of QoS services.

### 28.11.1 Fine-Grain QoS And Flows

Much of the early work on QoS arose from telephone companies. The designers assumed a connection-oriented data network modeled after the telephone system: when a customer needed to communicate with a remote site (e.g., a web server), the customers would create a connection. Furthermore, the designers assumed a customer would issue QoS requirements for each connection, and a provider would compute a charge according to the distance spanned and QoS used.

The phone companies incorporated many QoS features in the design of *Asynchronous Transmission Mode (ATM)*. Although ATM did not survive and providers do not generally charge for each connection, some of the terminology that ATM created for fine-grain QoS still persists with minor modifications. Instead of specifying the QoS on a connection, we now use the term *flow*. A flow generally refers to transport-layer communication such as a TCP connection, a set of UDP packets traveling between a pair of applications, or a VoIP telephone call. Figure 28.4 lists four main categories of service that were present in ATM, and explains how they relate to flows.

Abbreviation	Expansion	Meaning
CBR	Constant Bit Rate	Data enters the flow at a fixed rate, such as data from a digitized voice call entering at exactly 64 Kbps
VBR	Variable Bit Rate	Data enters the flow at a variable rate within specified statistical bounds
ABR	Available Bit Rate	The flow agrees to use whatever data rate is available at a given time
UBR	Unspecified Bit Rate	No bit rate is specified for the flow; the application is satisfied with best-effort service

**Figure 28.4** Four main categories of QoS service.

As the figure indicates, CBR service is appropriate for a flow that transfers data at a fixed rate, with digitized voice being the canonical example. VBR service is appropriate for a flow that uses a variable-rate encoding. For example, some video codecs send differential encodings, where the amount of data sent for a frame is proportional to the difference between the previous frame and the current frame. In such cases, a customer can specify the average data rate that is expected as well as a maximum data rate and the length of time the maximum rate will occur. VBR asks users to specify:

- Sustained Bit Rate (SBR)
- Peak Bit Rate (PBR)
- Sustained Burst Size (SBS)
- Peak Burst Size (PBS)

ABR service implies sharing: a customer is willing to pay for any amount of service that is available. If other customers send data, the amount available will be lower (and presumably the provider will charge less). Finally, UBR service means the customer does not want to pay higher fees and is satisfied with best-effort service.

When Internet QoS was first considered, telephone companies argued that fine-grain services would be needed before the quality of voice telephone calls over a packet network would be acceptable. Consequently, in addition to the work on ATM, the research community began to explore fine-grain QoS on the Internet. The research was known as *Integrated Services (IntServ)*.

### 28.11.2 Coarse-Grain QoS And Classes Of Service

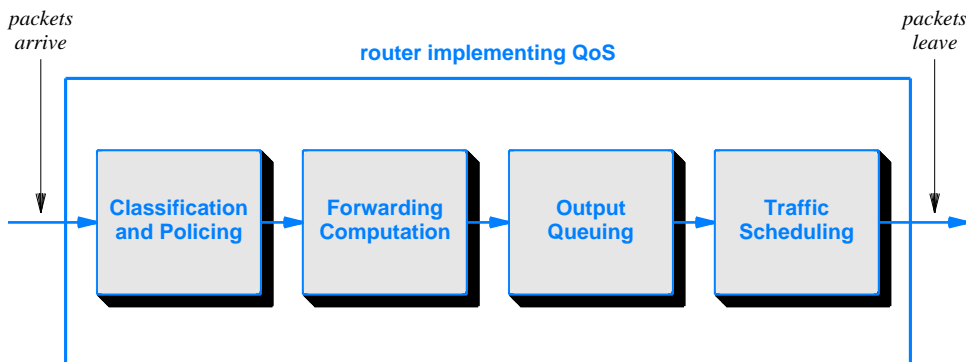
The alternative to fine-grain QoS is a coarse-grain approach in which traffic is divided into *classes* and QoS parameters are assigned to the class rather than individual flows. To understand the motivation for the coarse-grain approach, it is necessary to consider an implementation of QoS on a core router. The connections to routers can each operate at 10 Gbps, which means packets arrive at an extremely high rate; special hardware is needed to perform forwarding because conventional processors are too slow. Furthermore, because it carries traffic among major ISPs, a core router can handle millions of simultaneous flows. QoS requires many additional resources. The router must maintain state for millions of flows, and must perform a complex computation for each packet. Memory accesses slow down processing. In addition, a router must allocate resources when a flow begins, and deallocate them when a flow ends.

After many years of research on Integrated Services and the creation of several protocols, the research community and the IETF concluded that a fine-grain approach was generally both impractical and unnecessary. On one hand, an average user will not have sufficient understanding of QoS to choose parameters. After all, what throughput requirements would one specify for a connection to a typical web site? On the other hand, core routers have insufficient processing power to implement per-flow QoS. Thus, most work on QoS concentrates on defining a few broad classes of service rather than trying to provide end-to-end QoS for each individual flow. We can summarize:

*Despite many years of research and standards work, the fine-grain approach to QoS has been relegated to a few special cases.*

### 28.12 Implementation Of QoS

Figure 28.5 illustrates the four steps a switch or router uses to implement QoS.



**Figure 28.5** The four key steps used to implement QoS.

*Classification And Policing.* When a packet arrives, a router *classifies* the packet by assigning the packet a flow identifier. For a fine-grain system, the identifier specifies an individual connection; for a coarse-grain system, the identifier specifies a traffic class. Once an identifier has been assigned, the router performs *policing*, which means that the router verifies that the packet does not violate parameters for the flow. In particular, if a customer sends data faster than the maximum rate for which the customer is paying, a policer begins to discard packets. One technique used for policing is *Random Early Discard (RED)* in which packets on a given flow are dropped probabilistically. A queue is established for the flow, and the current size of the queue is used to determine the probability of drop. When the queue is less than half full, the probability is set to zero. When the queue is completely full, the probability is set to one. At queue sizes in between, the probability is linearly proportional to the number of packets in the queue. Using RED helps avoid a cyclic problem caused by *tail drop* in which all incoming packets are discarded once a queue fills, many TCP sessions each back off and begin slow start, traffic increases until the queue fills again, and the cycle repeats.

*Forwarding Computation.* When computing a next-hop, a router or switch can use the flow identifier. In some cases, the flow identifier determines the path to be followed (e.g., all voice traffic is sent out port 54 to a voice switch). In other cases, the flow identifier is ignored and the destination address in each packet is used to select a next hop. The exact details of forwarding depend on the purpose of a particular switch or router and the manager's QoS policies.

*Output Queuing.* Most implementations of QoS create a set of queues for each output port. Once the forwarding computation selects an output port for the packet, the output queueing mechanism uses the flow identifier to place the packet in one of the queues associated with the port. A coarse-grain system typically uses one queue per class. Thus, if a manager establishes eight QoS classes, each output port will have eight queues. A fine-grain system often has one queue per connection, with queues arranged in a hierarchy. For example, one network processor chip provides 256,000 queues arranged in a multi-level hierarchy.

*Traffic Scheduling.* A *traffic scheduler* implements the QoS policies by selecting a packet to send whenever a port is idle. For example, a manager might specify that three customers each receive 25% of the capacity and all other customers share the remaining capacity. To implement such a policy, a traffic scheduler might use four queues and a *round-robin* approach to select packets. Thus, if all customers are sending data, the three designated customers will each receive one quarter of the capacity, as specified.

More sophisticated packet selection algorithms can be used to implement complex proportional sharing. Complexity arises because a traffic scheduler must maintain long-term policies even though packets arrive in bursts. Thus, a traffic scheduler must adapt to situations where a given queue temporarily exceeds its allotted data rate provided the long-term average meets the specified bounds. Similarly, a traffic scheduler must adapt to a situation where one or more queues is temporarily empty by dividing the unused capacity among other queues.

Many traffic scheduling algorithms have been proposed and analyzed. It is not possible to create a practical algorithm that achieves perfection; each is a compromise between fairness and computational overhead. Figure 28.6 lists traffic management algorithms that have been proposed and studied.

Algorithm	Description
Leaky Bucket	Allows a queue to send packets at a fixed rate by incrementing a packet counter periodically and using the counter to control transmission
Token Bucket	Allows a queue to send data at a fixed rate by incrementing a byte counter periodically and using the counter to control transmission
Weighted Round Robin	Selects packets from a set of queues according to a set of weights that divide the capacity into fixed percentages, assuming a uniform packet size
Deficit Round Robin	A variant of the round-robin approach that accounts for bytes sent rather than packets transferred, and allows a temporary deficit caused by a large packet

**Figure 28.6** Example traffic scheduling algorithms.

## 28.13 Internet QoS Technologies

The IETF has designed a series of technologies and protocols related to QoS. Three significant efforts are:

- RSVP and COPS
- DiffServ
- MPLS

*RSVP and COPS.* As it explored IntServ, the IETF developed two protocols to provide QoS: the *Resource ReSerVation Protocol (RSVP)* and the *Common Open Policy Services (COPS)* protocol. RSVP is a fine-grained version of QoS. Thus, RSVP is needed for each TCP or UDP session. To use RSVP, an application sends a request that specifies the desired QoS. Each router along the path from the source to the destination reserves the requested resources and passes the request to the next router. Eventually, the destination host must agree to the request. When every hop along the path has agreed to honor the request, a flow identifier is generated and returned. Traffic can

then be sent along the reserved path. COPS is a companion protocol for RSVP used to specify and enforce policies. A router that implements policing uses COPS to communicate with a policy server and obtain information about the flow parameters. Because it is designed to provide fine-grain, per-flow QoS, RSVP is seldom used.

*DiffServ.* Once it abandoned IntServ and fine-grain QoS, the IETF created *Differentiated Services (DiffServ)* to define a coarse-grain QoS mechanism. The DiffServ effort produced a definition of how classes can be specified and how the *TYPE OF SERVICE* field in an IPv4 or IPv6 header can be used to specify the class of a datagram. Although various ISPs have experimented with DiffServ, the technology does not enjoy widespread acceptance.

*MPLS.* Chapter 19 describes *MultiProtocol Label Switching (MPLS)* as a communication-oriented communication mechanism built on top of IP. To use MPLS, a manager configures forwarding paths through a set of MPLS-capable routers. At one end of a path, each datagram is encapsulated in an MPLS header and injected into the MPLS path; at the other end, each datagram is extracted, the MPLS header is removed, and the datagram is forwarded to its destination. In many cases, a traffic scheduling policy is assigned to an MPLS path, which means that when a datagram is inserted in a particular path, QoS parameters are set for the datagram. Thus, an ISP might set up an MPLS path for voice data that is separate from the MPLS path used for other data.

## 28.14 Summary

The two primary measures of network performance are delay, the time required to send a bit from one computer to another, and throughput, the number of bits per second that can be transmitted across the network. Although throughput is commonly called speed, throughput is a measure of network capacity. The delay-throughput product measures the amount of data that can be in transit at a given instant. Delay and throughput are not independent — as throughput approaches 100% of capacity, delays increase rapidly.

Jitter, a measure of variance in delay, is becoming important in data networks. Low jitter can be achieved with an isochronous network or with a protocol that handles the transmission of real-time audio and video; the Internet uses the protocol approach.

Measuring network performance can be difficult. Asymmetric routes mean synchronized clocks are needed to measure delay; bursty traffic means performance can change rapidly. Because additional traffic from measurement can alter network conditions, many managers prefer passive measurement technologies, such as NetFlow.

Both fine-grain and coarse-grain QoS have been studied; fine-grain efforts have generally been abandoned. ATM defined categories of service, and the acronyms are still used: Constant, Variable, Available, and Unspecified Bit Rate (CBR, VBR, ABR, and UBR).

To implement QoS, a switch or router classifies and polices incoming data, forwards and places each packet on an output queue, and uses a traffic scheduler to select a packet to send when an output port becomes free. Several traffic scheduling algorithms have been proposed and analyzed; each is a tradeoff between optimal fairness and computational overhead.

The IETF defined RSVP and COPS as part of the IntServ effort; when emphasis shifted away from fine-grain QoS, the IETF defined DiffServ. The IETF also defined MPLS as a traffic engineering technology. QoS parameters can be associated with each MPLS tunnel, meaning that once a datagram has been classified, its MPLS association defines its QoS parameters.

## EXERCISES

- 28.1 List and describe the three primary measures of network performance.
- 28.2 Give five types of delay along with an explanation of each.
- 28.3 Would you expect access delays to be longer on a LAN or on a WAN? Queuing delays? Why?
- 28.4 How can throughput be measured?
- 28.5 What name is used for the form of throughput that is the most meaningful to a user?
- 28.6 Give examples of processing that make goodput less than the channel capacity.
- 28.7 Provide an explanation of delay and throughput in terms of bits being transmitted.
- 28.8 Which of delay or throughput provides the most fundamental limit on performance? Why?
- 28.9 Use *ping* to measure network latency to local and distant sites. What is the minimum and maximum Internet delay you can find?
- 28.10 If one pings IP address 127.0.0.1, the latency is extremely low. Explain.
- 28.11 Download a copy of the program *ttcp* and use it to measure throughput on a local Ethernet. What is the goodput? Estimate the link utilization achieved.
- 28.12 Compare the throughput of a 100 Mbps network and a 1 Gbps network.
- 28.13 What is jitter, and what are the two approaches used to overcome jitter?
- 28.14 Professionals sometimes refer to a “knee” in the delay curve. To understand what they mean, plot the effective delay for values of utilization between 0 and 0.95. Can you find a value of utilization for which the curve appears to increase sharply?
- 28.15 How much data can be “in flight” between a sending ground station, a satellite, and a receiving station? To find out, compute the delay-throughput product for a GEO satellite network that operates at 3 Mbps. Assume that the satellite orbits at 20,000 miles above the earth, and that radio transmissions propagate at the speed of light.
- 28.16 Why is measurement of network performance difficult?
- 28.17 How does data traffic differ from voice traffic?

- 28.18** Explain why ISPs count the number of packets received per unit time instead of merely the number of bytes received per unit time.
- 28.19** What are the two types of QoS?
- 28.20** Estimate the computational power needed to implement fine-grain QoS in the core of the Internet: assume a 10 Gbps link delivering 1000 byte packets and  $N$  arithmetic operations per packet, and calculate the number of operations a processor needs to perform per second.
- 28.21** List the four main categories of QoS that were derived from ATM, and give the meaning of each.
- 28.22** Consider a web browser. What type of QoS would be appropriate for a typical flow where the browser downloads a web page? Why?
- 28.23** If two users create a chat session over the Internet, what category of QoS will they be using?
- 28.24** What four parameters are used to characterize a VBR flow?
- 28.25** Explain the four steps used to implement QoS.
- 28.26** If your ISP uses leaky bucket to schedule packet transmission, will your throughput be higher with large packets or small packets? Explain.
- 28.27** What is DiffServ?
- 28.28** How does MPLS forwarding differ from conventional IP forwarding?