CHAPTER 24

Congestion Control and Quality of Service

Congestion control and quality of service are two issues so closely bound together that improving one means improving the other and ignoring one usually means ignoring the other. Most techniques to prevent or eliminate congestion also improve the quality of service in a network.

We have postponed the discussion of these issues until now because these are issues related not to one layer, but to three: the data link layer, the network layer, and the transport layer. We waited until now so that we can discuss these issues once instead of repeating the subject three times. Throughout the chapter, we give examples of congestion control and quality of service at different layers.

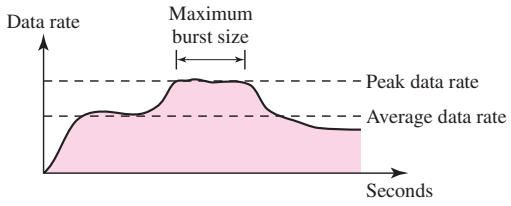
24.1 DATA TRAFFIC

The main focus of congestion control and quality of service is data traffic. In congestion control we try to avoid traffic congestion. In quality of service, we try to create an appropriate environment for the traffic. So, before talking about congestion control and quality of service, we discuss the data traffic itself.

Traffic Descriptor

Traffic descriptors are qualitative values that represent a data flow. Figure 24.1 shows a traffic flow with some of these values.

Figure 24.1 *Traffic descriptors*



Average Data Rate

The **average data rate** is the number of bits sent during a period of time, divided by the number of seconds in that period. We use the following equation:

$$\text{Average data rate} = \frac{\text{amount of data}}{\text{time}}$$

The average data rate is a very useful characteristic of traffic because it indicates the average bandwidth needed by the traffic.

Peak Data Rate

The **peak data rate** defines the maximum data rate of the traffic. In Figure 24.1 it is the maximum *y* axis value. The peak data rate is a very important measurement because it indicates the peak bandwidth that the network needs for traffic to pass through without changing its data flow.

Maximum Burst Size

Although the peak data rate is a critical value for the network, it can usually be ignored if the duration of the peak value is very short. For example, if data are flowing steadily at the rate of 1 Mbps with a sudden peak data rate of 2 Mbps for just 1 ms, the network probably can handle the situation. However, if the peak data rate lasts 60 ms, there may be a problem for the network. The **maximum burst size** normally refers to the maximum length of time the traffic is generated at the peak rate.

Effective Bandwidth

The **effective bandwidth** is the bandwidth that the network needs to allocate for the flow of traffic. The effective bandwidth is a function of three values: average data rate, peak data rate, and maximum burst size. The calculation of this value is very complex.

Traffic Profiles

For our purposes, a data flow can have one of the following traffic profiles: constant bit rate, variable bit rate, or bursty as shown in Figure 24.2.

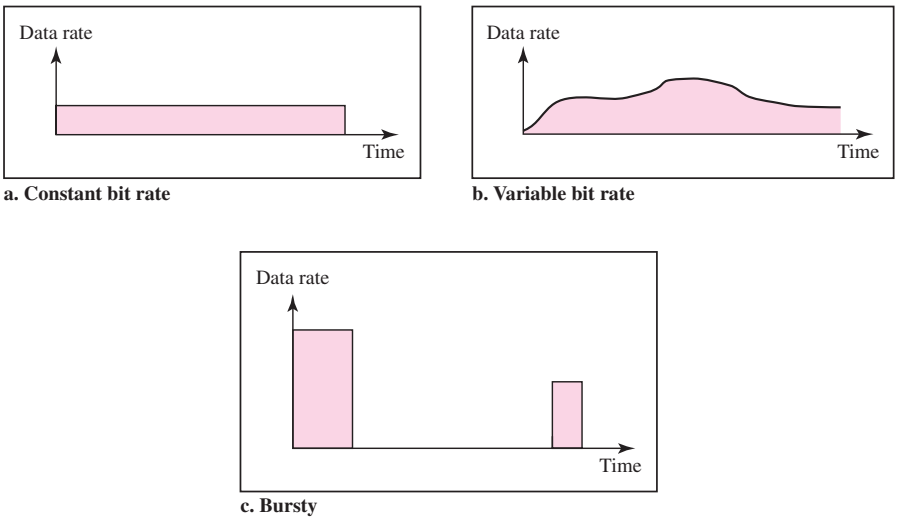
Constant Bit Rate

A **constant-bit-rate (CBR),** or a fixed-rate, traffic model has a data rate that does not change. In this type of flow, the average data rate and the peak data rate are the same. The maximum burst size is not applicable. This type of traffic is very easy for a network to handle since it is predictable. The network knows in advance how much bandwidth to allocate for this type of flow.

Variable Bit Rate

In the **variable-bit-rate (VBR)** category, the rate of the data flow changes in time, with the changes smooth instead of sudden and sharp. In this type of flow, the average data

Figure 24.2 *Three traffic profiles*



rate and the peak data rate are different. The maximum burst size is usually a small value. This type of traffic is more difficult to handle than constant-bit-rate traffic, but it normally does not need to be reshaped, as we will see later.

Bursty

In the **bursty data** category, the data rate changes suddenly in a very short time. It may jump from zero, for example, to 1 Mbps in a few microseconds and vice versa. It may also remain at this value for a while. The average bit rate and the peak bit rate are very different values in this type of flow. The maximum burst size is significant. This is the most difficult type of traffic for a network to handle because the profile is very unpredictable. To handle this type of traffic, the network normally needs to reshape it, using reshaping techniques, as we will see shortly. Bursty traffic is one of the main causes of congestion in a network.

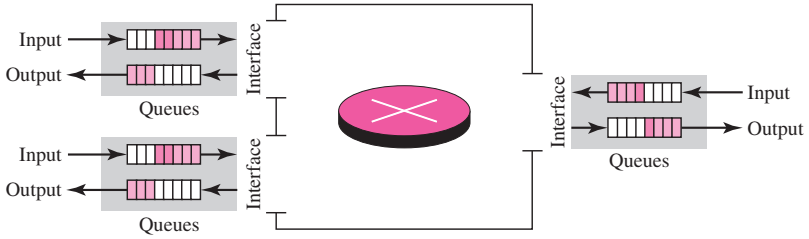
24.2 CONGESTION

An important issue in a packet-switched network is **congestion.** Congestion in a network may occur if the **load** on the network—the number of packets sent to the network—is greater than the *capacity* of the network—the number of packets a network can handle. **Congestion control** refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

We may ask why there is congestion on a network. Congestion happens in any system that involves waiting. For example, congestion happens on a freeway because any abnormality in the flow, such as an accident during rush hour, creates blockage.

Congestion in a network or internetwork occurs because routers and switches have queues—buffers that hold the packets before and after processing. A router, for example, has an input queue and an output queue for each interface. When a packet arrives at the incoming interface, it undergoes three steps before departing, as shown in Figure 24.3.

Figure 24.3 *Queues in a router*



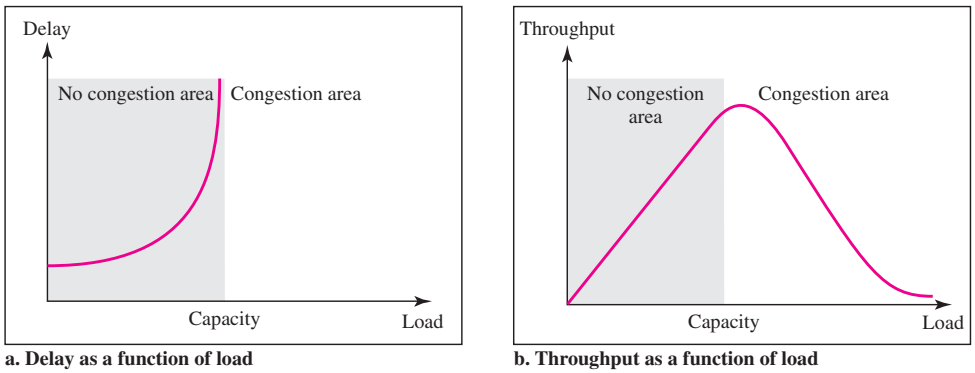
1. The packet is put at the end of the input queue while waiting to be checked.
2. The processing module of the router removes the packet from the input queue once it reaches the front of the queue and uses its routing table and the destination address to find the route.
3. The packet is put in the appropriate output queue and waits its turn to be sent.

We need to be aware of two issues. First, if the rate of packet arrival is higher than the packet processing rate, the input queues become longer and longer. Second, if the packet departure rate is less than the packet processing rate, the output queues become longer and longer.

Network Performance

Congestion control involves two factors that measure the performance of a network: *delay* and *throughput*. Figure 24.4 shows these two performance measures as function of load.

Figure 24.4 *Packet delay and throughput as functions of load*



a. Delay as a function of load

b. Throughput as a function of load

Delay Versus Load

Note that when the load is much less than the capacity of the network, the **delay** is at a minimum. This minimum delay is composed of propagation delay and processing delay, both of which are negligible. However, when the load reaches the network capacity, the delay increases sharply because we now need to add the waiting time in the queues (for all routers in the path) to the total delay. Note that the delay becomes infinite when the load is greater than the capacity. If this is not obvious, consider the size of the queues when almost no packet reaches the destination, or reaches the destination with infinite delay; the queues become longer and longer. Delay has a negative effect on the load and consequently the congestion. When a packet is delayed, the source, not receiving the acknowledgment, retransmits the packet, which makes the delay, and the congestion, worse.

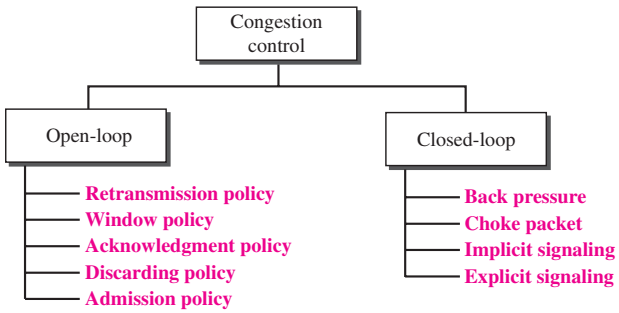
Throughput Versus Load

We defined throughput in Chapter 3 as the number of bits passing through a point in a second. We can extend that definition from bits to packets and from a point to a network. We can define **throughput** in a network as the number of packets passing through the network in a unit of time. Notice that when the load is below the capacity of the network, the throughput increases proportionally with the *load*. We expect the throughput to remain constant after the load reaches the capacity, but instead the throughput declines sharply. The reason is the discarding of packets by the routers. When the load exceeds the capacity, the queues become full and the routers have to discard some packets. Discarding packets does not reduce the number of packets in the network because the sources retransmit the packets, using time-out mechanisms, when the packets do not reach the destinations.

24.3 CONGESTION CONTROL

Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. In general, we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal) as shown in Figure 24.5.

Figure 24.5 *Congestion control categories*



Open-Loop Congestion Control

In **open-loop congestion control,** policies are applied to prevent congestion before it happens. In these mechanisms, congestion control is handled by either the source or the destination. We give a brief list of policies that can prevent congestion.

Retransmission Policy

Retransmission is sometimes unavoidable. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. Retransmission in general may increase congestion in the network. However, a good retransmission policy can prevent congestion. The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion. For example, the retransmission policy used by TCP (explained later) is designed to prevent or alleviate congestion.

Window Policy

The type of window at the sender may also affect congestion. The Selective Repeat window is better than the Go-Back-*N* window for congestion control. In the Go-Back-*N* window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse. The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted.

Acknowledgment Policy

The acknowledgment policy imposed by the receiver may also affect congestion. If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion. Several approaches are used in this case. A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires. A receiver may decide to acknowledge only *N* packets at a time. We need to know that the acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing less load on the network.

Discarding Policy

A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission. For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.

Admission Policy

An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility of future congestion.

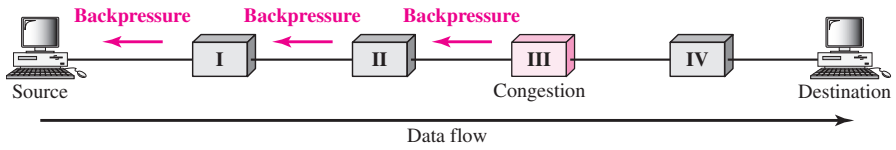
Closed-Loop Congestion Control

Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols. We describe a few of them here.

Backpressure

The technique of *backpressure* refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes. And so on. Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source. The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is coming. Figure 24.6 shows the idea of backpressure.

Figure 24.6 *Backpressure method for alleviating congestion*



Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down. Node II, in turn, may be congested because it is slowing down the output flow of data. If node II is congested, it informs node I to slow down, which in turn may create congestion. If so, node I informs the source of data to slow down. This, in time, alleviates the congestion. Note that the *pressure* on node III is moved backward to the source to remove the congestion.

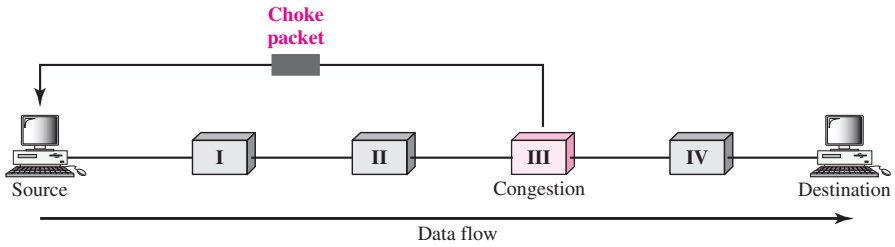
None of the virtual-circuit networks we studied in this book use backpressure. It was, however, implemented in the first virtual-circuit network, X.25. The technique cannot be implemented in a datagram network because in this type of network, a node (router) does not have the slightest knowledge of the upstream router.

Choke Packet

A **choke packet** is a packet sent by a node to the source to inform it of congestion. Note the difference between the backpressure and choke packet methods. In backpressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station. In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly. The intermediate nodes through which the packet has traveled are not warned. We have seen an example of this type of control in ICMP. When a router in the Internet is overwhelmed with IP datagrams, it may discard some of them; but it informs the source

host, using a source quench ICMP message. The warning message goes directly to the source station; the intermediate routers, and does not take any action. Figure 24.7 shows the idea of a choke packet.

Figure 24.7 *Choke packet*



Implicit Signaling

In implicit signaling, there is no communication between the congested node or nodes and the source. The source guesses that there is a congestion somewhere in the network from other symptoms. For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested. The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down. We will see this type of signaling when we discuss TCP congestion control later in the chapter.

Explicit Signaling

The node that experiences congestion can explicitly send a signal to the source or destination. The explicit signaling method, however, is different from the choke packet method. In the choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data. Explicit signaling, as we will see in Frame Relay congestion control, can occur in either the forward or the backward direction.

Backward Signaling A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

Forward Signaling A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

24.4 TWO EXAMPLES

To better understand the concept of congestion control, let us give two examples: one in TCP and the other in Frame Relay.

Congestion Control in TCP

We discussed TCP in Chapter 23. We now show how TCP uses congestion control to avoid congestion or alleviate congestion in the network.

Congestion Window

In Chapter 23, we talked about flow control and tried to discuss solutions when the receiver is overwhelmed with data. We said that the sender window size is determined by the available buffer space in the receiver (*rwnd*). In other words, we assumed that it is only the receiver that can dictate to the sender the size of the sender's window. We totally ignored another entity here—the network. If the network cannot deliver the data as fast as they are created by the sender, it must tell the sender to slow down. In other words, in addition to the receiver, the network is a second entity that determines the size of the sender's window.

Today, the sender's window size is determined not only by the receiver but also by congestion in the network.

The sender has two pieces of information: the receiver-advertised window size and the congestion window size. The actual size of the window is the minimum of these two.

Actual window size = minimum (rwnd, cwnd)

We show shortly how the size of the congestion window (*cwnd*) is determined.

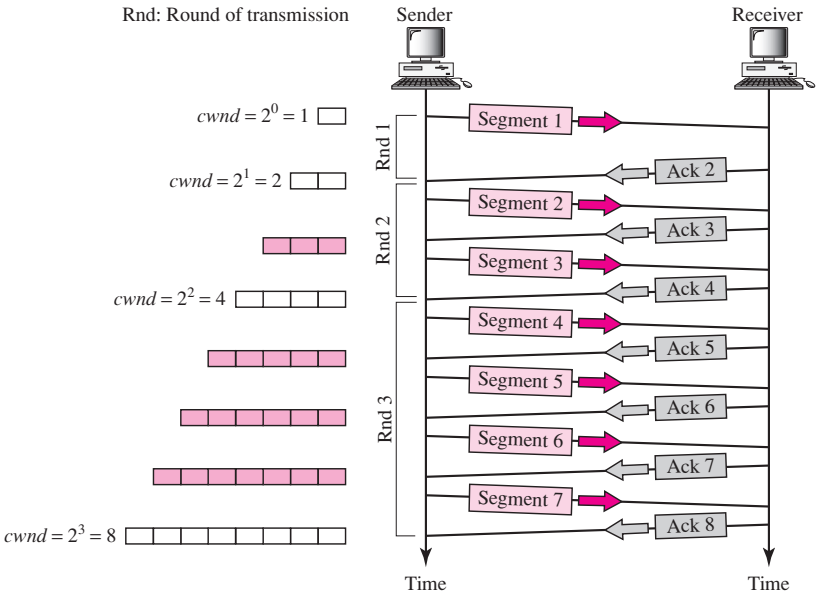
Congestion Policy

TCP's general policy for handling congestion is based on three phases: slow start, congestion avoidance, and congestion detection. In the slow-start phase, the sender starts with a very slow rate of transmission, but increases the rate rapidly to reach a threshold. When the threshold is reached, the data rate is reduced to avoid congestion. Finally if congestion is detected, the sender goes back to the slow-start or congestion avoidance phase based on how the congestion is detected.

Slow Start: Exponential Increase One of the algorithms used in TCP congestion control is called **slow start.** This algorithm is based on the idea that the size of the congestion window (*cwnd*) starts with one maximum segment size (MSS). The MSS is determined during connection establishment by using an option of the same name. The size of the window increases one MSS each time an acknowledgment is received. As the name implies, the window starts slowly, but grows exponentially. To show the idea, let us look at Figure 24.8. Note that we have used three simplifications to make the discussion more understandable. We have used segment numbers instead of byte numbers (as though each segment contains only 1 byte). We have assumed that *rwnd* is much higher than *cwnd,* so that the sender window size always equals *cwnd*. We have assumed that each segment is acknowledged individually.

The sender starts with *cwnd* = 1 MSS. This means that the sender can send only one segment. After receipt of the acknowledgment for segment 1, the size of the congestion window is increased by 1, which means that *cwnd* is now 2. Now two more segments can be sent. When each acknowledgment is received, the size of the window is increased by 1 MSS. When all seven segments are acknowledged, *cwnd* = 8.

Figure 24.8 *Slow start, exponential increase*



If we look at the size of *cwnd* in terms of rounds (acknowledgment of the whole window of segments), we find that the rate is exponential as shown below:

Start	→	$cwnd = 1$
After round 1	→	$cwnd = 2^1 = 2$
After round 2	→	$cwnd = 2^2 = 4$
After round 3	→	$cwnd = 2^3 = 8$

We need to mention that if there is delayed ACKs, the increase in the size of the window is less than power of 2.

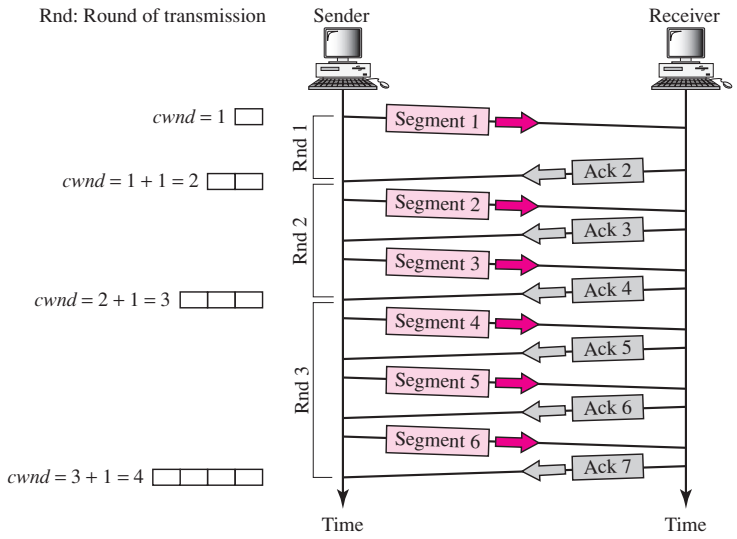
Slow start cannot continue indefinitely. There must be a threshold to stop this phase. The sender keeps track of a variable named *ssthresh* (slow-start threshold). When the size of window in bytes reaches this threshold, slow start stops and the next phase starts. In most implementations the value of *ssthresh* is 65,535 bytes.

In the slow-start algorithm, the size of the congestion window increases exponentially until it reaches a threshold.

Congestion Avoidance: Additive Increase If we start with the slow-start algorithm, the size of the congestion window increases exponentially. To avoid congestion before it happens, one must slow down this exponential growth. TCP defines another algorithm called **congestion avoidance,** which undergoes an **additive increase** instead of an exponential one. When the size of the congestion window reaches the slow-start threshold, the slow-start phase stops and the additive phase begins. In this algorithm, each time the whole window of segments is acknowledged (one round), the size of the

congestion window is increased by 1. To show the idea, we apply this algorithm to the same scenario as slow start, although we will see that the congestion avoidance algorithm usually starts when the size of the window is much greater than 1. Figure 24.9 shows the idea.

Figure 24.9 *Congestion avoidance, additive increase*



In this case, after the sender has received acknowledgments for a complete window size of segments, the size of the window is increased by one segment.

If we look at the size of *cwnd* in terms of rounds, we find that the rate is additive as shown below:

Start	→	***cwnd* = 1**
After round 1	→	***cwnd* = 1 + 1 = 2**
After round 2	→	***cwnd* = 2 + 1 = 3**
After round 3	→	***cwnd* = 3 + 1 = 4**

In the congestion avoidance algorithm, the size of the congestion window increases additively until congestion is detected.

Congestion Detection: Multiplicative Decrease If congestion occurs, the congestion window size must be decreased. The only way the sender can guess that congestion has occurred is by the need to retransmit a segment. However, retransmission can occur in one of two cases: when a timer times out or when three ACKs are received. In both cases, the size of the threshold is dropped to one-half, a **multiplicative decrease.** Most TCP implementations have two reactions:

1. If a time-out occurs, there is a stronger possibility of congestion; a segment has probably been dropped in the network, and there is no news about the sent segments.

In this case TCP reacts strongly:

a. It sets the value of the threshold to one-half of the current window size.
b. It sets *cwnd* to the size of one segment.
c. It starts the slow-start phase again.

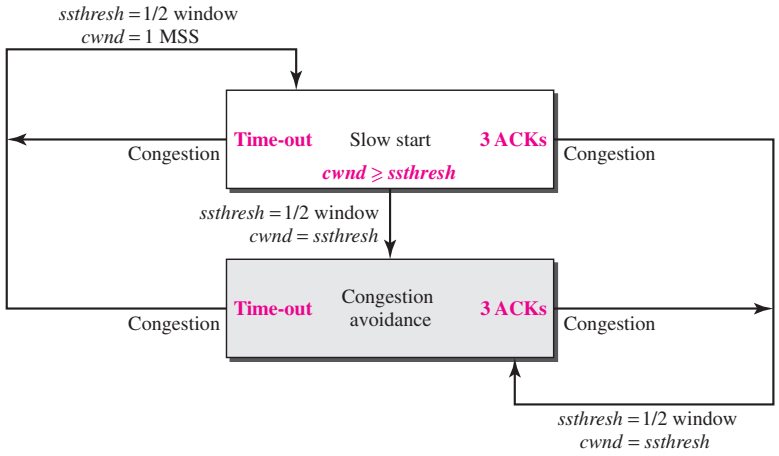
2. If three ACKs are received, there is a weaker possibility of congestion; a segment may have been dropped, but some segments after that may have arrived safely since three ACKs are received. This is called fast transmission and fast recovery. In this case, TCP has a weaker reaction:
   a. It sets the value of the threshold to one-half of the current window size.
   b. It sets *cwnd* to the value of the threshold (some implementations add three segment sizes to the threshold).
   c. It starts the congestion avoidance phase.

An implementations reacts to congestion detection in one of the following ways:

- If detection is by time-out, a new *slow-start* phase starts.
- If detection is by three ACKs, a new *congestion avoidance* phase starts.

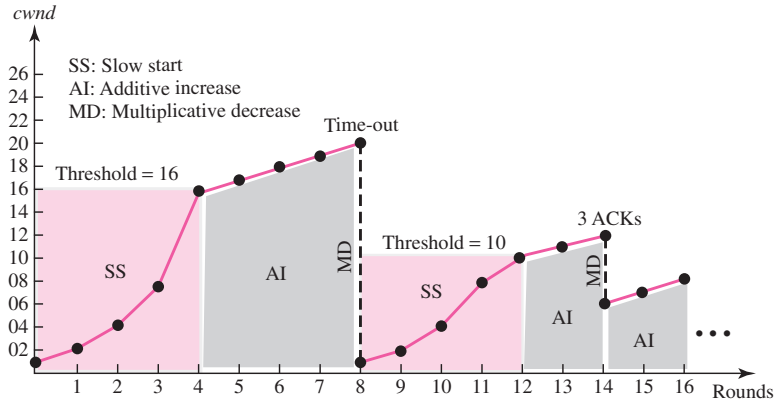
Summary In Figure 24.10, we summarize the congestion policy of TCP and the relationships between the three phases.

Figure 24.10 *TCP congestion policy summary*



We give an example in Figure 24.11. We assume that the maximum window size is 32 segments. The threshold is set to 16 segments (one-half of the maximum window size). In the *slow-start* phase the window size starts from 1 and grows exponentially until it reaches the threshold. After it reaches the threshold, the *congestion avoidance (additive increase)* procedure allows the window size to increase linearly until a time-out occurs or the maximum window size is reached. In Figure 24.11, the time-out occurs when the window size is 20. At this moment, the *multiplicative decrease* procedure takes

Figure 24.11 *Congestion example*



over and reduces the threshold to one-half of the previous window size. The previous window size was 20 when the time-out happened so the new threshold is now 10.

TCP moves to slow start again and starts with a window size of 1, and TCP moves to additive increase when the new threshold is reached. When the window size is 12, a three-ACKs event happens. The multiplicative decrease procedure takes over again. The threshold is set to 6 and TCP goes to the additive increase phase this time. It remains in this phase until another time-out or another three ACKs happen.

Congestion Control in Frame Relay

Congestion in a Frame Relay network decreases throughput and increases delay. A high throughput and low delay are the main goals of the Frame Relay protocol. Frame Relay does not have flow control. In addition, Frame Relay allows the user to transmit bursty data. This means that a Frame Relay network has the potential to be really congested with traffic, thus requiring congestion control.

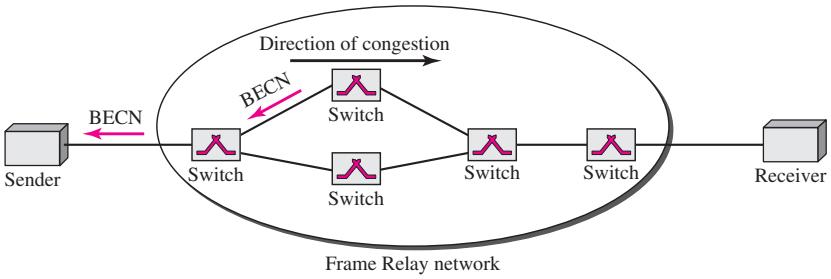
Congestion Avoidance

For congestion avoidance, the Frame Relay protocol uses 2 bits in the frame to explicitly warn the source and the destination of the presence of congestion.

BECN The **backward explicit congestion notification (BECN)** bit warns the sender of congestion in the network. One might ask how this is accomplished since the frames are traveling away from the sender. In fact, there are two methods: The switch can use response frames from the receiver (full-duplex mode), or else the switch can use a predefined connection (DLCI = 1023) to send special frames for this specific purpose. The sender can respond to this warning by simply reducing the data rate. Figure 24.12 shows the use of BECN.

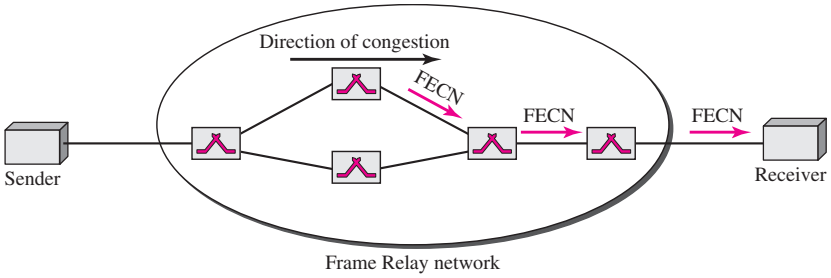
FECN The **forward explicit congestion notification (FECN)** bit is used to warn the receiver of congestion in the network. It might appear that the receiver cannot do anything to relieve the congestion. However, the Frame Relay protocol assumes that the sender and receiver are communicating with each other and are using some type of flow control at a higher level. For example, if there is an acknowledgment mechanism at this

Figure 24.12 *BECN*



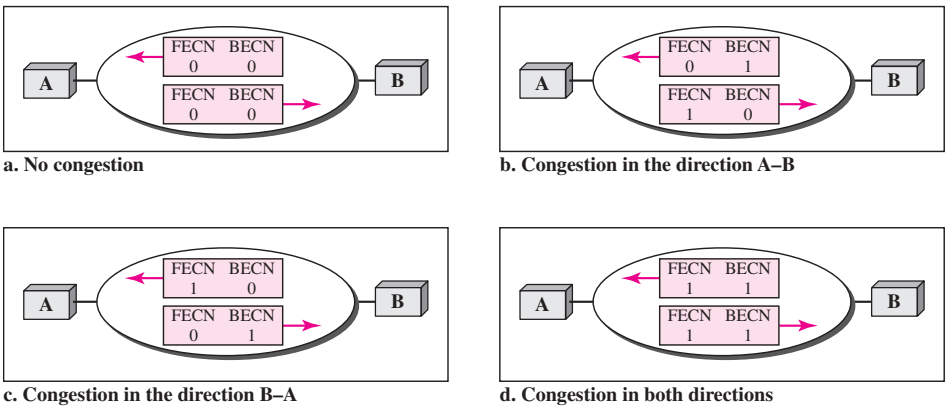
higher level, the receiver can delay the acknowledgment, thus forcing the sender to slow down. Figure 24.13 shows the use of FECN.

Figure 24.13 *FECN*



When two endpoints are communicating using a Frame Relay network, four situations may occur with regard to congestion. Figure 24.14 shows these four situations and the values of FECN and BECN.

Figure 24.14 *Four cases of congestion*



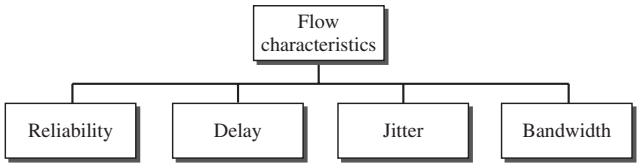
24.5 QUALITY OF SERVICE

Quality of service (QoS) is an internetworking issue that has been discussed more than defined. We can informally define quality of service as something a flow seeks to attain.

Flow Characteristics

Traditionally, four types of characteristics are attributed to a flow: reliability, delay, jitter, and bandwidth, as shown in Figure 24.15.

Figure 24.15 *Flow characteristics*



Reliability

Reliability is a characteristic that a flow needs. Lack of reliability means losing a packet or acknowledgment, which entails retransmission. However, the sensitivity of application programs to reliability is not the same. For example, it is more important that electronic mail, file transfer, and Internet access have reliable transmissions than telephony or audio conferencing.

Delay

Source-to-destination **delay** is another flow characteristic. Again applications can tolerate delay in different degrees. In this case, telephony, audio conferencing, video conferencing, and remote log-in need minimum delay, while delay in file transfer or e-mail is less important.

Jitter

Jitter is the variation in delay for packets belonging to the same flow. For example, if four packets depart at times 0, 1, 2, 3 and arrive at 20, 21, 22, 23, all have the same delay, 20 units of time. On the other hand, if the above four packets arrive at 21, 23, 21, and 28, they will have different delays: 21, 22, 19, and 24.

For applications such as audio and video, the first case is completely acceptable; the second case is not. For these applications, it does not matter if the packets arrive with a short or long delay as long as the delay is the same for all packets. For this application, the second case is not acceptable.

Jitter is defined as the variation in the packet delay. High jitter means the difference between delays is large; low jitter means the variation is small.

In Chapter 29, we show how multimedia communication deals with jitter. If the jitter is high, some action is needed in order to use the received data.

Bandwidth

Different applications need different bandwidths. In video conferencing we need to send millions of bits per second to refresh a color screen while the total number of bits in an e-mail may not reach even a million.

Flow Classes

Based on the flow characteristics, we can classify flows into groups, with each group having similar levels of characteristics. This categorization is not formal or universal; some protocols such as ATM have defined classes, as we will see later.

24.6 TECHNIQUES TO IMPROVE QoS

In Section 24.5 we tried to define QoS in terms of its characteristics. In this section, we discuss some techniques that can be used to improve the quality of service. We briefly discuss four common methods: scheduling, traffic shaping, admission control, and resource reservation.

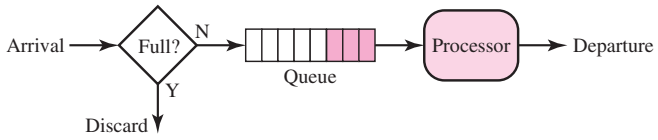
Scheduling

Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Several scheduling techniques are designed to improve the quality of service. We discuss three of them here: FIFO queuing, priority queuing, and weighted fair queuing.

FIFO Queuing

In **first-in, first-out (FIFO) queuing,** packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded. A FIFO queue is familiar to those who have had to wait for a bus at a bus stop. Figure 24.16 shows a conceptual view of a FIFO queue.

Figure 24.16 *FIFO queue*

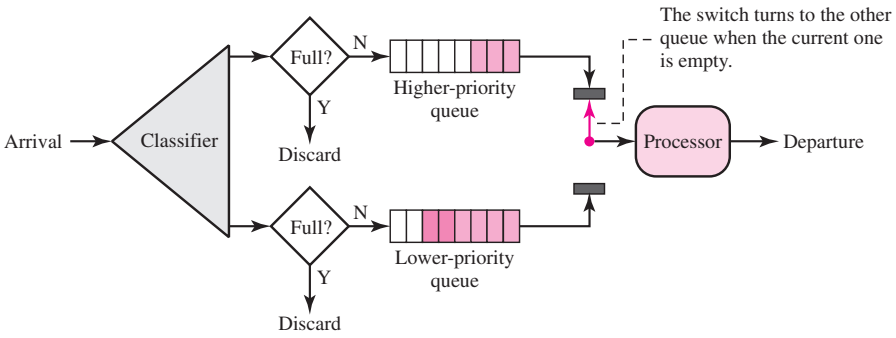


Priority Queuing

In **priority queuing,** packets are first assigned to a priority class. Each priority class has its own queue. The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last. Note that the system does not stop serving

a queue until it is empty. Figure 24.17 shows priority queuing with two priority levels (for simplicity).

Figure 24.17 *Priority queuing*



A priority queue can provide better QoS than the FIFO queue because higher-priority traffic, such as multimedia, can reach the destination with less delay. However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called *starvation*.

Weighted Fair Queuing

A better scheduling method is **weighted fair queuing.** In this technique, the packets are still assigned to different classes and admitted to different queues. The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight. The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight. For example, if the weights are 3, 2, and 1, three packets are processed from the first queue, two from the second queue, and one from the third queue. If the system does not impose priority on the classes, all weights can be equal. In this way, we have fair queuing with priority. Figure 24.18 shows the technique with three classes.

Traffic Shaping

Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic: leaky bucket and token bucket.

Leaky Bucket

If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called **leaky bucket** can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate. Figure 24.19 shows a leaky bucket and its effects.

Figure 24.18 *Weighted fair queuing*

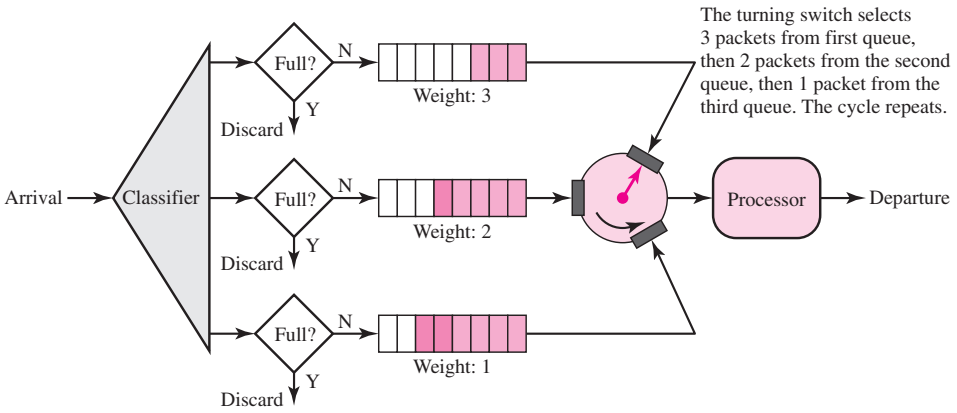
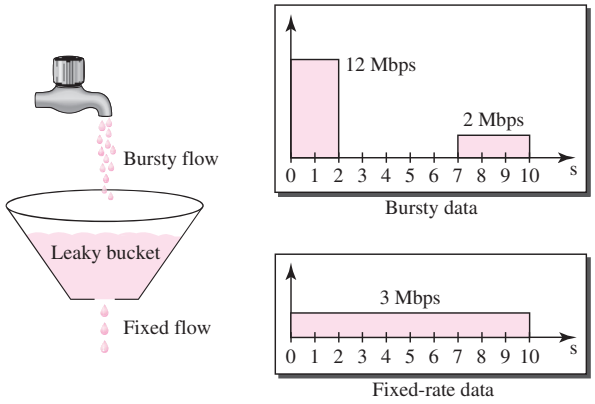


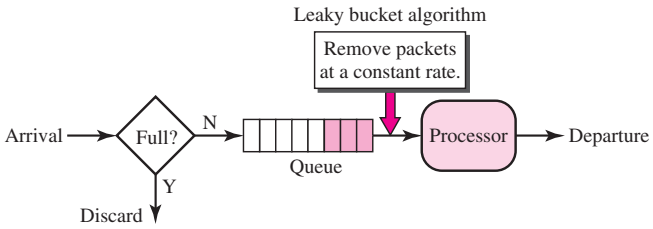
Figure 24.19 *Leaky bucket*



In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure 24.19 the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in 10 s. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 s. Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion. As an analogy, consider the freeway during rush hour (bursty traffic). If, instead, commuters could stagger their working hours, congestion on our freeways could be avoided.

A simple leaky bucket implementation is shown in Figure 24.20. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM

Figure 24.20 *Leaky bucket implementation*



networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to *n* at the tick of the clock.
2. If *n* is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until *n* is smaller than the packet size.
3. Reset the counter and go to step 1.

A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

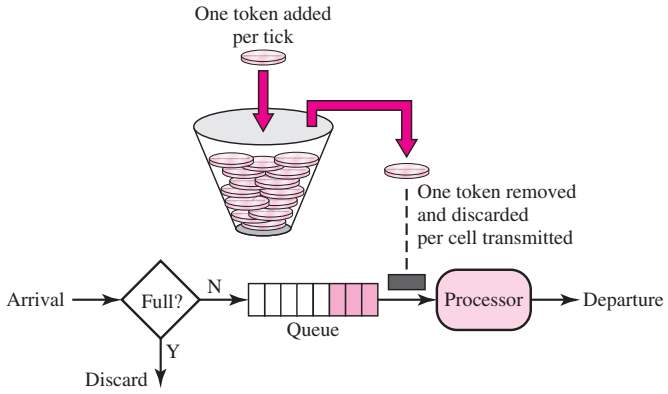
Token Bucket

The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the **token bucket** algorithm allows idle hosts to accumulate credit for the future in the form of tokens. For each tick of the clock, the system sends *n* tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if *n* is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens. Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick. In other words, the host can send bursty data as long as the bucket is not empty. Figure 24.21 shows the idea.

The token bucket can easily be implemented with a counter. The token is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.

The token bucket allows bursty traffic at a regulated maximum rate.

Figure 24.21 *Token bucket*



Combining Token Bucket and Leaky Bucket

The two techniques can be combined to credit an idle host and at the same time regulate the traffic. The leaky bucket is applied after the token bucket; the rate of the leaky bucket needs to be higher than the rate of tokens dropped in the bucket.

Resource Reservation

A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on. The quality of service is improved if these resources are reserved beforehand. We discuss in this section one QoS model called Integrated Services, which depends heavily on resource reservation to improve the quality of service.

Admission Control

Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications. Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

24.7 INTEGRATED SERVICES

Based on the topics in Sections 24.5 and 24.6, two models have been designed to provide quality of service in the Internet: Integrated Services and Differentiated Services. Both models emphasize the use of quality of service at the network layer (IP), although the model can also be used in other layers such as the data link. We discuss Integrated Services in this section and Differentiated Service in Section 24.8.

As we learned in Chapter 20, IP was originally designed for *best-effort* delivery. This means that every user receives the same level of services. This type of delivery

does not guarantee the minimum of a service, such as bandwidth, to applications such as real-time audio and video. If such an application accidentally gets extra bandwidth, it may be detrimental to other applications, resulting in congestion.

Integrated Services, sometimes called **IntServ,** is a *flow-based* QoS model, which means that a user needs to create a flow, a kind of virtual circuit, from the source to the destination and inform all routers of the resource requirement.

Integrated Services is a *flow-based* QoS model designed for IP.

Signaling

The reader may remember that IP is a connectionless, datagram, packet-switching protocol. How can we implement a flow-based model over a connectionless protocol? The solution is a signaling protocol to run over IP that provides the signaling mechanism for making a reservation. This protocol is called **Resource Reservation Protocol (RSVP)** and will be discussed shortly.

Flow Specification

When a source makes a reservation, it needs to define a flow specification. A flow specification has two parts: Rspec (resource specification) and Tspec (traffic specification). Rspec defines the resource that the flow needs to reserve (buffer, bandwidth, etc.). Tspec defines the traffic characterization of the flow.

Admission

After a router receives the flow specification from an application, it decides to admit or deny the service. The decision is based on the previous commitments of the router and the current availability of the resource.

Service Classes

Two classes of services have been defined for Integrated Services: guaranteed service and controlled-load service.

Guaranteed Service Class

This type of service is designed for real-time traffic that needs a guaranteed minimum end-to-end delay. The end-to-end delay is the sum of the delays in the routers, the propagation delay in the media, and the setup mechanism. Only the first, the sum of the delays in the routers, can be guaranteed by the router. This type of service guarantees that the packets will arrive within a certain delivery time and are not discarded if flow traffic stays within the boundary of Tspec. We can say that guaranteed services are quantitative services, in which the amount of end-to-end delay and the data rate must be defined by the application.

Controlled-Load Service Class

This type of service is designed for applications that can accept some delays, but are sensitive to an overloaded network and to the danger of losing packets. Good examples of these types of applications are file transfer, e-mail, and Internet access. The controlled-load service is a qualitative type of service in that the application requests the possibility of low-loss or no-loss packets.

RSVP

In the Integrated Services model, an application program needs resource reservation. As we learned in the discussion of the IntServ model, the resource reservation is for a *flow*. This means that if we want to use IntServ at the IP level, we need to create a flow, a kind of virtual-circuit network, out of the IP, which was originally designed as a datagram packet-switched network. A virtual-circuit network needs a signaling system to set up the virtual circuit before data traffic can start. The Resource Reservation Protocol (RSVP) is a signaling protocol to help IP create a flow and consequently make a resource reservation. Before discussing RSVP, we need to mention that it is an independent protocol separate from the Integrated Services model. It may be used in other models in the future.

Multicast Trees

RSVP is different from some other signaling systems we have seen before in that it is a signaling system designed for multicasting. However, RSVP can be also used for unicasting because unicasting is just a special case of multicasting with only one member in the multicast group. The reason for this design is to enable RSVP to provide resource reservations for all kinds of traffic including multimedia which often uses multicasting.

Receiver-Based Reservation

In RSVP, the receivers, not the sender, make the reservation. This strategy matches the other multicasting protocols. For example, in multicast routing protocols, the receivers, not the sender, make a decision to join or leave a multicast group.

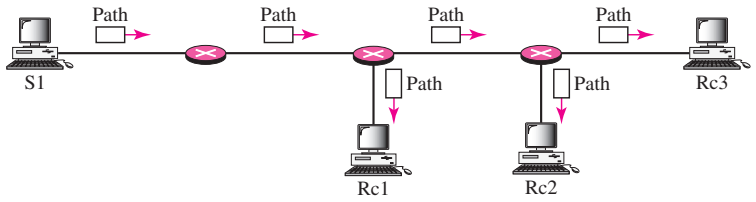
RSVP Messages

RSVP has several types of messages. However, for our purposes, we discuss only two of them: **Path** and **Resv.**

Path Messages Recall that the receivers in a flow make the reservation in RSVP. However, the receivers do not know the path traveled by packets before the reservation is made. The path is needed for the reservation. To solve the problem, RSVP uses *Path* messages. A Path message travels from the sender and reaches all receivers in the multicast path. On the way, a Path message stores the necessary information for the receivers. A Path message is sent in a multicast environment; a new message is created when the path diverges. Figure 24.22 shows path messages.

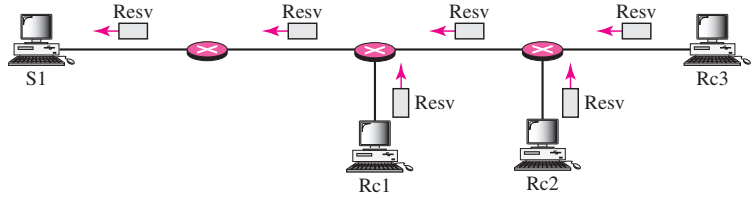
Resv Messages After a receiver has received a Path message, it sends a *Resv* message. The Resv message travels toward the sender (upstream) and makes a resource reservation on the routers that support RSVP. If a router does not support RSVP on the path, it routes

Figure 24.22 *Path messages*



the packet based on the best-effort delivery methods we discussed before. Figure 24.23 shows the Resv messages.

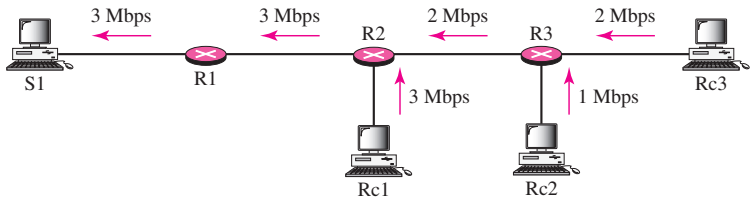
Figure 24.23 *Resv messages*



Reservation Merging

In RSVP, the resources are not reserved for each receiver in a flow; the reservation is merged. In Figure 24.24, Rc3 requests a 2-Mbps bandwidth while Rc2 requests a 1-Mbps bandwidth. Router R3, which needs to make a bandwidth reservation, merges the two requests. The reservation is made for 2 Mbps, the larger of the two, because a 2-Mbps input reservation can handle both requests. The same situation is true for R2. The reader may ask why Rc2 and Rc3, both belonging to one single flow, request different amounts of bandwidth. The answer is that, in a multimedia environment, different receivers may handle different grades of quality. For example, Rc2 may be able to receive video only at 1 Mbps (lower quality), while Rc3 may be able to receive video at 2 Mbps (higher quality).

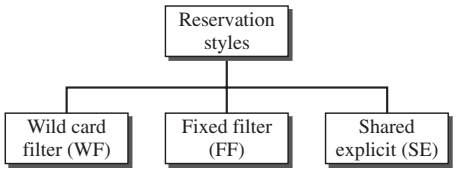
Figure 24.24 *Reservation merging*



Reservation Styles

When there is more than one flow, the router needs to make a reservation to accommodate all of them. RSVP defines three types of reservation styles, as shown in Figure 24.25.

Figure 24.25 *Reservation styles*



Wild Card Filter Style In this style, the router creates a single reservation for all senders. The reservation is based on the largest request. This type of style is used when the flows from different senders do not occur at the same time.

Fixed Filter Style In this style, the router creates a distinct reservation for each flow. This means that if there are *n* flows, *n* different reservations are made. This type of style is used when there is a high probability that flows from different senders will occur at the same time.

Shared Explicit Style In this style, the router creates a single reservation which can be shared by a set of flows.

Soft State

The reservation information (state) stored in every node for a flow needs to be refreshed periodically. This is referred to as a *soft state* as compared to the *hard state* used in other virtual-circuit protocols such as ATM or Frame Relay, where the information about the flow is maintained until it is erased. The default interval for refreshing is currently 30 s.

Problems with Integrated Services

There are at least two problems with Integrated Services that may prevent its full implementation in the Internet: scalability and service-type limitation.

Scalability

The Integrated Services model requires that each router keep information for each flow. As the Internet is growing every day, this is a serious problem.

Service-Type Limitation

The Integrated Services model provides only two types of services, guaranteed and control-load. Those opposing this model argue that applications may need more than these two types of services.

24.8 DIFFERENTIATED SERVICES

Differentiated Services (DS or **Diffserv)** was introduced by the IETF (Internet Engineering Task Force) to handle the shortcomings of Integrated Services. Two fundamental changes were made:

1. The main processing was moved from the core of the network to the edge of the network. This solves the scalability problem. The routers do not have to store information about flows. The applications, or hosts, define the type of service they need each time they send a packet.
2. The per-flow service is changed to per-class service. The router routes the packet based on the class of service defined in the packet, not the flow. This solves the service-type limitation problem. We can define different types of classes based on the needs of applications.

Differentiated Services is a class-based QoS model designed for IP.

DS Field

In Diffserv, each packet contains a field called the DS field. The value of this field is set at the boundary of the network by the host or the first router designated as the boundary router. IETF proposes to replace the existing TOS (type of service) field in IPv4 or the class field in IPv6 by the DS field, as shown in Figure 24.26.

Figure 24.26 *DS field*

The DS field contains two subfields: DSCP and CU. The DSCP (Differentiated Services Code Point) is a 6-bit subfield that defines the **per-hop behavior (PHB).** The 2-bit CU (currently unused) subfield is not currently used.

The Diffserv capable node (router) uses the DSCP 6 bits as an index to a table defining the packet-handling mechanism for the current packet being processed.

Per-Hop Behavior

The Diffserv model defines per-hop behaviors (PHBs) for each node that receives a packet. So far three PHBs are defined: DE PHB, EF PHB, and AF PHB.

DE PHB The DE PHB (default PHB) is the same as best-effort delivery, which is compatible with TOS.

EF PHB The EF PHB (expedited forwarding PHB) provides the following services:

- Low loss

- Low latency
- Ensured bandwidth

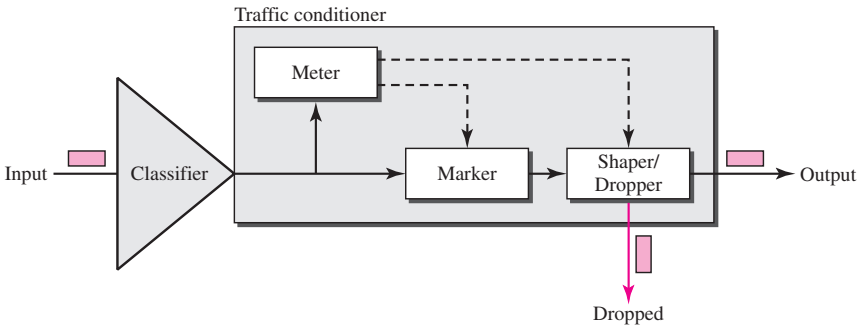
This is the same as having a virtual connection between the source and destination.

AF PHB The AF PHB (assured forwarding PHB) delivers the packet with a high assurance as long as the class traffic does not exceed the traffic profile of the node. The users of the network need to be aware that some packets may be discarded.

Traffic Conditioner

To implement Diffserv, the DS node uses traffic conditioners such as meters, markers, shapers, and droppers, as shown in Figure 24.27.

Figure 24.27 *Traffic conditioner*



Meters The meter checks to see if the incoming flow matches the negotiated traffic profile. The meter also sends this result to other components. The meter can use several tools such as a token bucket to check the profile.

Marker A marker can remark a packet that is using best-effort delivery (DSCP: 000000) or down-mark a packet based on information received from the meter. Down-marking (lowering the class of the flow) occurs if the flow does not match the profile. A marker does not up-mark (promote the class) a packet.

Shaper A shaper uses the information received from the meter to reshape the traffic if it is not compliant with the negotiated profile.

Dropper A dropper, which works as a shaper with no buffer, discards packets if the flow severely violates the negotiated profile.

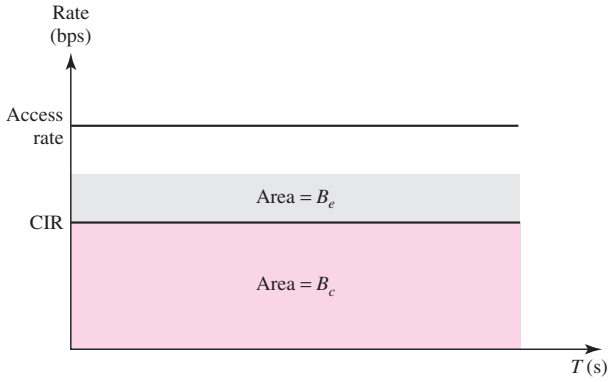
24.9 QoS IN SWITCHED NETWORKS

We discussed the proposed models for QoS in the IP protocols. Let us now discuss QoS as used in two switched networks: Frame Relay and ATM. These two networks are virtual-circuit networks that need a signaling protocol such as RSVP.

QoS in Frame Relay

Four different attributes to control traffic have been devised in Frame Relay: access rate, committed burst size B_c, committed information rate (CIR), and excess burst size B_e. These are set during the negotiation between the user and the network. For PVC connections, they are negotiated once; for SVC connections, they are negotiated for each connection during connection setup. Figure 24.28 shows the relationships between these four measurements.

Figure 24.28 *Relationship between traffic control attributes*



Access Rate

For every connection, an **access rate** (in bits per second) is defined. The access rate actually depends on the bandwidth of the channel connecting the user to the network. The user can never exceed this rate. For example, if the user is connected to a Frame Relay network by a T-1 line, the access rate is 1.544 Mbps and can never be exceeded.

Committed Burst Size

For every connection, Frame Relay defines a **committed burst size B_c.** This is the maximum number of bits in a predefined time that the network is committed to transfer without discarding any frame or setting the DE bit. For example, if a B_c of 400 kbits for a period of 4 s is granted, the user can send up to 400 kbits during a 4-s interval without worrying about any frame loss. Note that this is not a rate defined for each second. It is a cumulative measurement. The user can send 300 kbits during the first second, no data during the second and the third seconds, and finally 100 kbits during the fourth second.

Committed Information Rate

The **committed information rate (CIR)** is similar in concept to committed burst size except that it defines an average rate in bits per second. If the user follows this rate continuously, the network is committed to deliver the frames. However, because it is an average measurement, a user may send data at a higher rate than the CIR at times or at

a lower rate other times. As long as the average for the predefined period is met, the frames will be delivered.

The cumulative number of bits sent during the predefined period cannot exceed B_c. Note that the CIR is not an independent measurement; it can be calculated by using the following formula:

$$\text{CIR} = \frac{B_c}{T} \text{ bps}$$

For example, if the B_c is 5 kbits in a period of 5 s, the CIR is 5000/5, or 1 kbps.

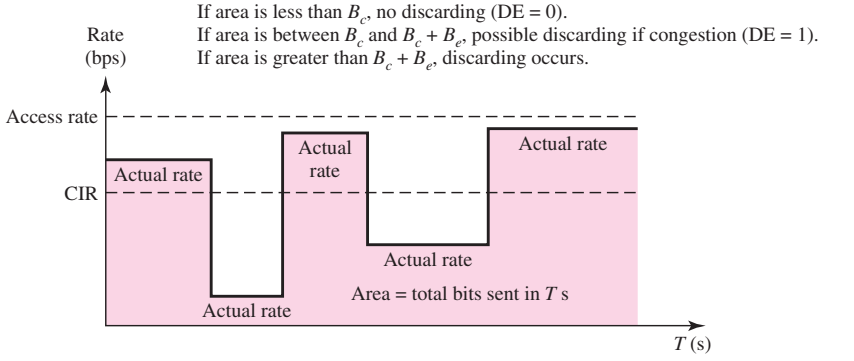
Excess Burst Size

For every connection, Frame Relay defines an **excess burst size B_e.** This is the maximum number of bits in excess of B_c that a user can send during a predefined time. The network is committed to transfer these bits if there is no congestion. Note that there is less commitment here than in the case of B_c. The network is committing itself conditionally.

User Rate

Figure 24.29 shows how a user can send bursty data. If the user never exceeds B_c, the network is committed to transmit the frames without discarding any. If the user exceeds B_c by less than B_e (that is, the total number of bits is less than $B_c + B_e$), the network is committed to transfer all the frames if there is no congestion. If there is congestion, some frames will be discarded. The first switch that receives the frames from the user has a counter and sets the DE bit for the frames that exceed B_c. The rest of the switches will discard these frames if there is congestion. Note that a user who needs to send data faster may exceed the B_c level. As long as the level is not above $B_c + B_e$, there is a chance that the frames will reach the destination without being discarded. Remember, however, that the moment the user exceeds the $B_c + B_e$ level, all the frames sent after that are discarded by the first switch.

Figure 24.29 *User rate in relation to B_c and $B_c + B_e$*



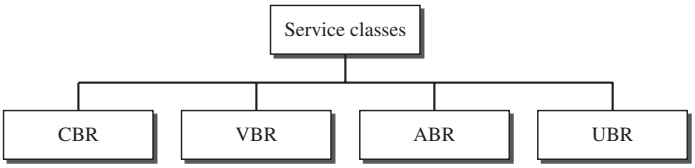
QoS in ATM

The QoS in ATM is based on the class, user-related attributes, and network-related attributes.

Classes

The ATM Forum defines four service classes: CBR, VBR, ABR, and UBR (see Figure 24.30).

Figure 24.30 *Service classes*



CBR The **constant-bit-rate (CBR)** class is designed for customers who need real-time audio or video services. The service is similar to that provided by a dedicated line such as a T line.

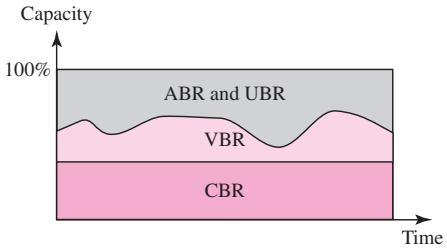
VBR The **variable-bit-rate (VBR)** class is divided into two subclasses: real-time (VBR-RT) and non-real-time (VBR-NRT). VBR-RT is designed for those users who need real-time services (such as voice and video transmission) and use compression techniques to create a variable bit rate. VBR-NRT is designed for those users who do not need real-time services but use compression techniques to create a variable bit rate.

ABR The **available-bit-rate (ABR)** class delivers cells at a minimum rate. If more network capacity is available, this minimum rate can be exceeded. ABR is particularly suitable for applications that are bursty.

UBR The **unspecified-bit-rate (UBR)** class is a best-effort delivery service that does not guarantee anything.

Figure 24.31 shows the relationship of different classes to the total capacity of the network.

Figure 24.31 *Relationship of service classes to the total capacity of the network*



User-Related Attributes

ATM defines two sets of attributes. User-related attributes are those attributes that define how fast the user wants to send data. These are negotiated at the time of contract between a user and a network. The following are some user-related attributes:

SCR The *sustained cell rate* (SCR) is the average cell rate over a long time interval. The actual cell rate may be lower or higher than this value, but the average should be equal to or less than the SCR.

PCR The *peak cell rate* (PCR) defines the sender's maximum cell rate. The user's cell rate can sometimes reach this peak, as long as the SCR is maintained.

MCR The *minimum cell rate* (MCR) defines the minimum cell rate acceptable to the sender. For example, if the MCR is 50,000, the network must guarantee that the sender can send at least 50,000 cells per second.

CVDT The *cell variation delay tolerance* (CVDT) is a measure of the variation in cell transmission times. For example, if the CVDT is 5 ns, this means that the difference between the minimum and the maximum delays in delivering the cells should not exceed 5 ns.

Network-Related Attributes

The network-related attributes are those that define characteristics of the network. The following are some network-related attributes:

CLR The *cell loss ratio* (CLR) defines the fraction of cells lost (or delivered so late that they are considered lost) during transmission. For example, if the sender sends 100 cells and one of them is lost, the CLR is

$$\text{CLR} = \frac{1}{100} = 10^{-2}$$

CTD The *cell transfer delay* (CTD) is the average time needed for a cell to travel from source to destination. The maximum CTD and the minimum CTD are also considered attributes.

CDV The *cell delay variation* (CDV) is the difference between the CTD maximum and the CTD minimum.

CER The *cell error ratio* (CER) defines the fraction of the cells delivered in error.

24.10 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [. . .] refer to the reference list at the end of the text.

Books

Congestion control and QoS are discussed in Sections 5.3 and 5.5 of [Tan03] and in Chapter 3 of [Sta98]. Easy reading about QoS can be found in [FH98]. The full discussion of QoS is covered in [Bla00].

24.11 KEY TERMS

access rate
additive increase
available bit rate (ABR)
average data rate
backward explicit congestion notification (BECN)
bursty data
choke packet
closed-loop congestion control
committed burst size B_c
committed information rate (CIR)
congestion
congestion avoidance
congestion control
constant bit rate (CBR)
delay
Differentiated Services (DS or Diffserv)
effective bandwidth
excess burst size B_e
first-in, first-out (FIFO) queuing
forward explicit congestion notification (FECN)
Integrated Services (IntServ)
jitter
leaky bucket
load
maximum burst size
multiplicative decrease
open-loop congestion control
Path message
peak data rate
per-hop behavior (PHB)
priority queuing
quality of service (QoS)
reliability
Resource Reservation Protocol (RSVP)
Resv message
slow start
throughput
token bucket
traffic shaping
unspecified bit rate (UBR)
variable bit rate (VBR)
weighted fair queuing

24.12 SUMMARY

- The average data rate, peak data rate, maximum burst size, and effective band width are qualitative values that describe a data flow.
- A data flow can have a constant bit rate, a variable bit rate, or traffic that is bursty.
- Congestion control refers to the mechanisms and techniques to control congestion and keep the load below capacity.
- Delay and throughput measure the performance of a network.
- Open-loop congestion control prevents congestion; closed-loop congestion control removes congestion.

- TCP avoids congestion through the use of two strategies: the combination of slow start and additive increase, and multiplicative decrease.
- Frame Relay avoids congestion through the use of two strategies: backward explicit congestion notification (BECN) and forward explicit congestion notification (FECN).
- A flow can be characterized by its reliability, delay, jitter, and bandwidth.
- Scheduling, traffic shaping, resource reservation, and admission control are techniques to improve quality of service (QoS).
- FIFO queuing, priority queuing, and weighted fair queuing are scheduling techniques.
- Leaky bucket and token bucket are traffic shaping techniques.
- Integrated Services is a flow-based QoS model designed for IP.
- The Resource Reservation Protocol (RSVP) is a signaling protocol that helps IP create a flow and makes a resource reservation.
- Differential Services is a class-based QoS model designed for IP.
- Access rate, committed burst size, committed information rate, and excess burst size are attributes to control traffic in Frame Relay.
- Quality of service in ATM is based on service classes, user-related attributes, and network-related attributes.

24.13 PRACTICE SET

Review Questions

1. How are congestion control and quality of service related?
2. What is a traffic descriptor?
3. What is the relationship between the average data rate and the peak data rate?
4. What is the definition of bursty data?
5. What is the difference between open-loop congestion control and closed-loop congestion control?
6. Name the policies that can prevent congestion.
7. Name the mechanisms that can alleviate congestion.
8. What determines the sender window size in TCP?
9. How does Frame Relay control congestion?
10. What attributes can be used to describe a flow of data?
11. What are four general techniques to improve quality of service?
12. What is traffic shaping? Name two methods to shape traffic.
13. What is the major difference between Integrated Services and Differentiated Services?
14. How is Resource Reservation Protocol related to Integrated Services?
15. What attributes are used for traffic control in Frame Relay?
16. In regard to quality of service, how do user-related attributes differ from network-related attributes in ATM?

Exercises

17. The address field of a Frame Relay frame is 1011000000010111. Is there any congestion in the forward direction? Is there any congestion in the backward direction?
18. A frame goes from A to B. There is congestion in both directions. Is the FECN bit set? Is the BECN bit set?
19. In a leaky bucket used to control liquid flow, how many gallons of liquid are left in the bucket if the output rate is 5 gal/min, there is an input burst of 100 gal/min for 12 s, and there is no input for 48 s?
20. An output interface in a switch is designed using the leaky bucket algorithm to send 8000 bytes/s (tick). If the following frames are received in sequence, show the frames that are sent during each second.

    Frames 1, 2, 3, 4: 4000 bytes each

    Frames 5, 6, 7: 3200 bytes each

    Frames 8, 9: 400 bytes each

    Frames 10, 11, 12: 2000 bytes each
21. A user is connected to a Frame Relay network through a T-1 line. The granted CIR is 1 Mbps with a B_c of 5 million bits/5 s and B_e of 1 million bits/5 s.
    a. What is the access rate?
    b. Can the user send data at 1.6 Mbps?
    c. Can the user send data at 1 Mbps all the time? Is it guaranteed that frames are never discarded in this case?
    d. Can the user send data at 1.2 Mbps all the time? Is it guaranteed that frames are never discarded in this case? If the answer is no, is it guaranteed that frames are discarded only if there is congestion?
    e. Repeat the question in part (d) for a constant rate of 1.4 Mbps.
    f. What is the maximum data rate the user can use all the time without worrying about the frames being discarded?
    g. If the user wants to take a risk, what is the maximum data rate that can be used with no chance of discarding if there is no congestion?
22. In Exercise 21 the user sends data at 1.4 Mbps for 2 s and nothing for the next 3 s. Is there a danger of discarded data if there is no congestion? Is there a danger of discarded data if there is congestion?
23. In ATM, if each cell takes 10 μs to reach the destination, what is the CTD?
24. An ATM network has lost 5 cells out of 10,000 and 2 are in error. What is the CLR? What is the CER?