NETWORK FLOW AND CONGESTION CONTROL

Modern computer networks connect geographically dispersed nodes using switches and routers, and transmission lines between them. In this way, bursty and random traffic streams can be statistically multiplexed to make more efficient use of resources. For example, the hub network in Fig. 1 is used to connect N users with N shared links and an $N \times N$ switch. Communication between pairs of users is accomplished by going through the hub. If, instead, all nodes were to be connected using dedicated links, $N(N-1)/2$ links would be required. However, at any point in time communication usually takes place only between a small fraction of the users. Hence, providing full and unshared connectivity between all users would be wasteful of resources. The functionality of computer networks in connecting users is quite similar, in many aspects, to that of highways and local streets in connecting households. In both cases, effective traffic control mechanisms are needed to regulate the flow of traffic and ensure high throughput.

Unlike traditional voice communications, where an active call requires constant bit rate from the network, data communication is bursty in its nature. A typical data session may require very low data rates during periods of inactivity and much higher rates at other times. Consequently, there may be times when incoming traffic to a network exceeds its capacity.

Flow and congestion control are mechanisms used in computer networks for preventing users from overwhelming the network with more data than the network can handle. The simplest way to handle network congestion is to temporarily buffer the excess traffic at a congested switch until it can all be transmitted. Yet, since switch buffers are limited in size, there may be times when sustained excessive demand on parts of the network causes buffers to fill-up, and excess packets can no longer be buffered and must be discarded.

When packets are discarded it is typically left up to higher-layer protocols to recover the lost packets using an appropriate retransmission mechanism. For example, the Transmission Control Protocol (TCP) recovers from such buffer overflows by using a timed acknowledgment mechanism and retransmitting packets for which an acknowledgment does not arrive in time. Consequently, at times of congestion, packets may be retransmitted not only because of buffer overflows but also because of the increased delay that is due to the congestion. In the absence of flow control, this, sometimes unnecessary, retransmission of packets can lead to instability where little if any new traffic can flow through the network. A well-designed flow-control mechanism should keep the traffic levels in the network low enough to prevent buffers from overflowing and maintain relatively low end-to-end delays. Furthermore, in the event of congestion, the flow-control mechanism should allow the network to stabilize. Figure 2 illustrates the benefits of an effective flow-control mechanism.

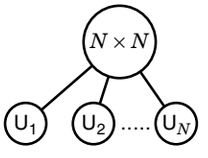

Figure 1. A hub network.

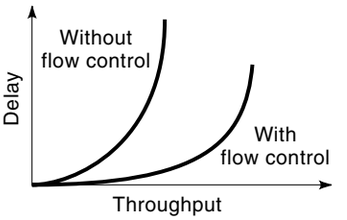

Figure 2. An effective flow-control mechanism can yield both higher throughputs and decreased delays.

In addition to the obvious objectives of limiting delays and buffer overflow, a good flow-control scheme should also treat all sessions fairly. One notion of "fairness" is to treat all sessions in the network equally. However, this notion is not appropriate for networks that attempt to provide Quality-of-Service (QoS) guarantees. In some networks users may be offered service contracts guaranteeing minimum data rates, maximum packet delays, and packet discard rates, as well as other performance measures. In such networks, it is up to the flow-control mechanism to make sure that these guarantees are met. Clearly, in this case, sessions cannot be treated equally and a different notion of fairness, related to the service agreements of the users, must be used. A more detailed discussion of fairness and how flow-control mechanisms attempt to provide fairness will be given in the next section.

There are a number of flow-control mechanisms that are used in practice, all of which attempt to limit delays and buffer overflows in the network by keeping packets waiting outside the network rather than in buffers within the network. The simplest mechanism for preventing congestion in the network is call admission. Here a call may be blocked, or prevented from entering the network, if it is somehow determined that the network lacks sufficient resources to accept the call. Call admission is a passive flow-control mechanism in the sense that once a call is admitted, nothing further is done to regulate traffic. It is therefore appropriate for traffic with very predictable behavior. Typically, call-admission mechanisms are used in circuit-switched networks (e.g., the telephone network); however, with the recent emergence of packet network services offering QoS guarantees, call blocking may also play a role in data networks, in conjunction with additional mechanisms to regulate traffic among active sessions. This article will focus on active flow-control mechanisms that attempt to regulate the traffic flow among active sessions. A comprehensive discussion of flow control in data networks can be found in Refs. 1 and 2.

As described in Ref. 3, one way to classify flow-control mechanisms is based on the layer of ISO/OSI reference model at which the mechanism operates. For example, there are data link, network, and transport layer congestion-control schemes. Typically, a combination of such mechanisms is used. The selection depends upon the severity and duration of congestion. Figure 3 shows how the duration of congestion affects the choice of the method.

In general, the longer the duration, the higher the layer at which control should be exercised. For example, if the congestion is permanent, the installation of additional links is required. If the congestion lasts for the duration of the connec-

J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*. Copyright © 1999 John Wiley & Sons, Inc.

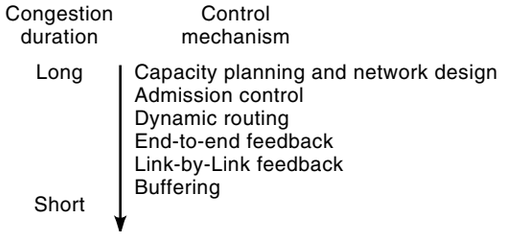

Figure 3. Control mechanisms based on congestion duration.

tion, admission control (e.g., use of busy signal) or dynamic routing (i.e., rerouting of traffic into another less congested path) is more appropriate. If the congestion lasts for several round-trip delays, transport level control with end-to-end feedback is more effective. If the congestion is of a short duration (less than a round-trip delay), link-by-link feedback or sufficient buffering should be used. Since every network can have overloads of all durations, every network needs a combination of control mechanisms at various levels. No single scheme can solve all congestion problems. The rest of this article will focus on mechanisms that deal with congestion that lasts only a few round-trip delays.

ISSUES AND MECHANISMS FOR CONGESTION CONTROL

Buffer Implementation and Management

As discussed earlier, call-admission-control mechanism alone is only appropriate for regulating traffic with steady or predictable bandwidth requirements (e.g., voice), but not effective for dealing with unpredictable bursty traffic (e.g., data). For efficient utilization of network bandwidth, it is often necessary to buffer the traffic when the incoming traffic to a node temporarily exceeds the capacity of its outgoing link. Flow control thus involves buffer management at a node in such a way that the service requirements of connections traversing the node can be satisfied.

In general, packet loss at a node will occur less frequently with a larger buffer than with a smaller one. However, a larger buffer may also lead to larger packet delay. For most data applications, an excessive packet delay will yield the same effect as a packet loss and will trigger a retransmission of the delayed packet. Thus, there is a tradeoff between throughput and delay in regulating network traffic. A key challenge in flow control is to achieve good delay-throughput, among connections with possibly different service requirements competing for network resources.

In addition to buffer size, another issue in buffer management for flow control has to do with the order in which packets of various connections are stored into and transmitted out of the buffer. The simplest way to buffer packets is to implement a single first-in-first-out (FIFO) queueing structure, in which buffered packets of all connections are transmitted on a first-come-first-serve (FCFS) basis. In other words, when a packet arrives at a node and needs to be buffered, it will be put at the end of a queue, regardless of which connection it belongs to. The packet at the front of the queue is always the first to be transmitted. Since the order of packet arrivals determines the order of packet departure, it is difficult to support different service requirements to connections with FIFO queueing. Moreover, since various traffic is mixed into the same queue, some connections can overutilize the buffer space, thereby preventing other connections from using it.

A more sophisticated way is to implement a separate queue for each connection, that is, per-connection queueing, so that buffered packets of different connections are isolated from one another. With per-connection queueing, a scheduling mechanism is used to decide, at any instant, which connection can transmit its packet. Compared with FIFO queueing, per-connection queueing is more expensive to implement, but offers greater flexibility in exercising flow control. For traffic with minimal and similar service requirements, FIFO queueing is usually sufficient. When different classes of traffic with different levels of service requirements are mixed together onto the same link, per-connection queueing may be necessary (4).

In the following sections, the problems of packet scheduling and packet discarding, which are closely related to buffer management for flow control, will be discussed.

Packet Scheduling

In general, a network node can have multiple incoming and outgoing links. Packets can be buffered at either the entrance or the exit interface of a node. The former is called input buffering and the latter output buffering. With input buffering, packets of different connections from the same incoming link of a node will first be buffered before they are transmitted to different outgoing links. To resolve possible contention caused by packets from different incoming links transmitting to the same outgoing link, a scheduling mechanism is required, to determine which packet should be transmitted at any instant. Moreover, when input buffering is implemented using FIFO queueing, packets of slow connections at the front of the queue will block packets of fast connections that follow. This is usually called the head-of-line (HOL) blocking problem. In fact, it is known that under certain traffic assumptions (e.g., a packet from each incoming link is equally likely to be transmitted to any outgoing link of the node), with input FIFO buffering, at most 58% of the maximum possible throughput can be achieved (5).

On the other hand, with output buffering, a packet arriving at a node is immediately transferred to the interface of its destined outgoing link, and is buffered there before it is transmitted. In this case, there is no HOL blocking problem and no scheduling mechanism is needed to resolve transmission contention among packets from different incoming links. However, since packets from all incoming links can potentially go on the same outgoing link at any instant, a node with output buffering needs to transfer packets at a rate that is the aggregate speed of all incoming links. This also means that faster (and, thus, more expensive) switching hardware is usually required for nodes with output buffering than with input buffering.

The problem of packet scheduling is further complicated by the fact that different connections may have different bandwidth or service requirements. Thus, a scheduling mechanism is needed to selectively expedite or transmit the buffered packets of various connections. For example, if each con-

nection should share the bandwidth of an outgoing link equally, a node can transmit buffered packets of each connection in a round-robin fashion. Similarly, when a particular connection has a minimum bandwidth requirement of R packets/second, then a node should schedule the transmissions so that, on the average, at least one packet of that connection will be transmitted every $1/R$ s. Furthermore, a node may desire to delay the transmission of the packets of some connections to avoid or relieve congestion further along the paths used by those connections, when such congestion information is available at the node. Simple FIFO queueing at a node often cannot support these and various scheduling mechanisms, and more expensive per-connection queueing is necessary when the scheduling constraints are stringent. More information on packet scheduling can be found in Refs. 6 and 7.

Packet Discarding

Since there is only a finite amount of available buffer space at a node, packets will be discarded if congestion persists. When packets of a connection get discarded, whether they will be retransmitted depends on the service requirements of that connection. For example, when the connection is a file transfer application, where each packet carries essential information, discarded packets need to be retransmitted by the source. The retransmission is usually performed if the receipt of a packet has not been acknowledged after a time-out period. In the case of a TCP connection, the destination returns to the source an acknowledgment packet corresponding to each data packet received, and the retransmission time-out is determined dynamically (8).

On the other hand, for real-time traffic such as voice or video, discarded packets are usually not retransmitted because delayed information is useless in such cases. A common approach to flow control for real-time traffic is to assign different priority levels to packets so that packets of highest priority will be discarded least often, if at all. Before a connection is established, the network may exercise a call-admission mechanism to ensure that the transmission of these highest priority packets can be maintained above a certain rate in order to support a minimum acceptable level of quality of service. This approach is also applicable to data traffic, where each discarded packet needs to be retransmitted. In this case, a network may offer several different classes of service with different priority levels in terms of packet discarding. When a connection is established, it can negotiate with the network to which service it wants to subscribe and, subsequently, during periods of congestion its packets will be discarded, based on their priority level.

It is sometimes desirable to discard packets even when buffer space is still available, particularly if FIFO queueing is used. This is because a connection overutilizing the buffer space in a FIFO queue will cause packets of other connections sharing the FIFO queue to be discarded. Thus, packets should be discarded if they belong to connections that utilize more than their fair share of buffer space, or if they may cause packets of higher priority to be discarded. Furthermore, if packets are to be discarded further down the path, because of congestion there, they should be discarded as early as possible, to avoid wasting additional network resources unnecessarily (9,10).

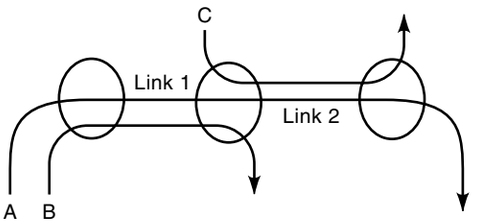

Figure 4. Fair bandwidth allocation.

Fair Allocation of Bandwidth

In addition to limiting delay and buffer overflow, fairness in network use is another objective of flow control. It is difficult to define a simple notion of fairness when different connections with different service requirements are present in the network. Here only a particular notion of fairness on bandwidth allocation will be discussed.

First consider a simple network with two links and three connections, as shown in Fig. 4. For the present, assume that each link has equal capacity, supporting 1 unit/s of traffic. If the fairness criterion is to allocate an equal rate to each connection, then each connection should get a rate of $\frac{1}{2}$ unit/s and the total network throughput in this case would be $\frac{3}{2}$ units/s. Note, however, that the maximum network throughput is 2 units/s, which can be achieved by shutting off connection A and allowing connections B and C each to transmit 1 unit/s of traffic. This example shows that fairness and throughput are two independent (and sometimes conflicting) objectives of flow control.

Now suppose the capacity of link 1 is changed to $\frac{1}{2}$ unit/s. In this case, connections A and B can share the bandwidth of link 1 equally, resulting in a throughput of $\frac{1}{4}$ unit/s for each. However, it will be a waste of bandwidth in link 2 if connection C is allocated with less than $\frac{3}{4}$ unit/s of bandwidth; it will be unfair if the bandwidth allocated to connection C is more, since it would further restrict the bandwidth allocated to connection A. This example motivates the notion of max–min fairness, which refers to maximizing bandwidth utilization for connections with the minimum bandwidth allocation.

More formally, it can be said that a set of connections has a max–min fair bandwidth allocation if the bandwidth allocated to any connection C′ cannot be increased without further decreasing the bandwidth allocated to another connection whose bandwidth is already smaller than C′. For example, in Fig. 4 with the capacity of link 1 being $\frac{1}{2}$ unit/s, one cannot increase the bandwidth allocated to connection C above $\frac{3}{4}$ unit/s without making the bandwidth of connection A smaller than $\frac{1}{4}$ unit/s.

Max–min fairness can also be defined in terms of the notion of a bottleneck link. With respect to some bandwidth allocation, a particular link L is a bottleneck link for a connection C′, which traverses L if the bandwidth of L is fully utilized and if the bandwidth allocated to C′ is no less than the bandwidth allocated to any other connection traversing L. Then a max–min fair bandwidth allocation can be shown to be equivalent to the condition that each connection has a bottleneck link, with respect to that allocation.

The notion of max–min fairness needs to be modified if each connection requires a minimum guaranteed data rate. One possible way is first to define the excess capacity of each

link L to be the bandwidth of L minus the aggregate guaranteed rates of all the connections that traverse L. Then a set of connections has a max–min fair bandwidth allocation if the excess capacity of each link is shared in a max–min fair manner (according to the notion defined earlier). More information on fair queueing algorithms and their performance can be found in Refs. 11–14.

Window Flow Control

The oldest and most common flow-control mechanism used in networks is window flow control. Window flow control has been used since the inception of packet-switched data networks and it appears in X.25, SNA, and TCP/IP networks (1,2). Window flow control regulates the rate with which sessions can insert packets into the network with a simple acknowledgment mechanism. Within a given session, the destination sends an acknowledgment to the source for every packet that it receives. With a window size of W, the source is limited to having W outstanding packets for which an acknowledgment has not been received. Hence, the window scheme limits the number of packets that a given session can have inside the network to the window size, W. These packets can be either in buffers throughout the network or propagating on transmission lines. This strategy is typically implemented using a sliding transmission window, where the start of the window is equal to the oldest packet for which an acknowledgment has not yet been received. Only packets from within the window can be transmitted, and the window is advanced as acknowledgments for earlier packets are received. An example with $W = 4$ is shown in Fig. 5. One reason that this strategy is very popular is its similarity to window-based retransmission mechanisms (e.g., Go Back N or SRP), which are used for error control in data networks, making it easy to implement in conjunction with the error-control scheme.

While window flow control, in effect, limits the number of packets that a given session can have in the network, it also indirectly regulates the rate of the session. Suppose that the round-trip delay for transmitting a packet and receiving its acknowledgment is D seconds. Then with a window of size W, a session can at most transmit $r = W/D$ packets per second. This is because, after sending a full window of packets, the sender must wait for the acknowledgment of the first packet before it can send any new packets. As delays in the network increase (i.e., D increases), the maximum session rate r is forced to decrease, producing the desired effect of slowing transmissions down at time of congestion. As congestion is alleviated, D is decreased, allowing sessions to increase their transmission rates.

One problem with window flow control is that window flow control cannot be used for sessions that require guaranteed data rates, such as real-time traffic, because as delays through the network vary, the rate of the session is forced to vary as well. Another problem is in the choice of a window size. On the one hand, one would like to keep window sizes small, in order to limit the number of packets in the network and prevent congestion. However, one would also want to allow sessions the ability to transmit at the maximum rate, at times when there is no congestion in the network. Consider a network where the transmission time for a packet is X. In order to allow unimpeded transmission, the window size W must be greater than D/X. That is, the window size must be large enough to allow a session to transmit packets continuously, while waiting for acknowledgments to return. Clearly, when W is greater than D/X, flow control is not active (i.e., the session can transmit at the maximum rate of $1/X$ packets per second) and when W is smaller than D/X, flow control is active and the session transmits at a rate of $W/D < 1/X$ packets per second. The problem is in choosing a window size that both allows sessions unimpeded transmission when there is no congestion, and also prevents congestion from building up in the network.

When there is no congestion in the network, the primary source of delay is propagation delay. Since propagation delay would be present, regardless of congestion, the window size should be big enough to allow unimpeded transmission when propagation is the only source of delay in the network. Hence if the propagation delay is equal to D_p then the window size should be at least equal to D_p/X, allowing transmission at a rate of $1/X$ packets per second, when the only source of delay is due to propagation. This is particularly needed in high-speed networks, where propagation delays can be relatively large. However, this can lead to the use of very large windows. Consider, for example, transmission over a satellite, where the round-trip propagation and signal processing delays can be on the order of a second. Suppose that the transmission rate is 10^6 bits/s and that the packet size is 1000 bits. In order to allow sessions to transmit at the full rate of 10^6, the window size must be at least 1000 packets. Hence, as many as 1000 packets per second can be in the network for each session. With so many packets in flight simultaneously, attempting to control congestion in the network becomes very difficult. First, the window mechanism becomes somewhat ineffective, because delays that are due to congestion are likely to be relatively small, compared with the propagation delay. Recalling that when flow control becomes active the allowable session rate is $r = W/D$, and since the overall increase in delay due to congestion is small, as compared with the overall delay, the result is only small decrease in the session rate. Also, with very large windows, sufficient buffering must be present throughout the network, to prevent buffer overflows

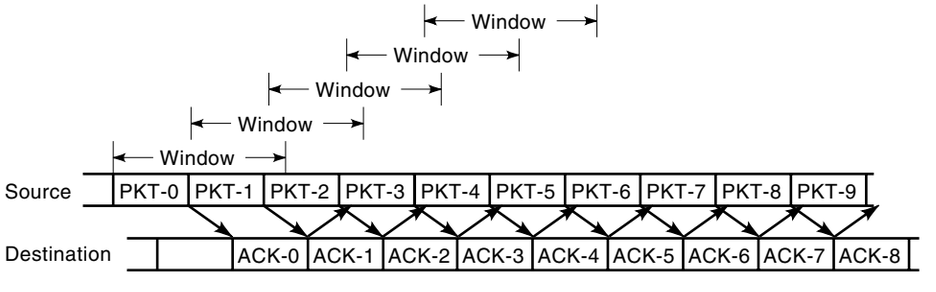

Figure 5. Sliding window mechanism with $W = 4$.

in the event that congestion sets in. Clearly, a mechanism is needed to dynamically alter the window size allocated to a session, based on estimated traffic conditions in the network. In this way, when the network is not congested the window size can be increased, to allow unimpeded transmission, but, as congestion begins to set in, the window size can be reduced to yield a more effective control of the allowed session rate. An example of a dynamic window adjustment mechanism is given in Ref. 15.

In order to be able to adjust the window size in response to congestion, a mechanism must exist to provide feedback to the source nodes, regarding the status of congestion in the network. There are many ways in which this can be done, and finding the best such method is an area of active research. One approach, for example, would require nodes in the network, upon experiencing congestion, to send special packets (sometimes called choke packets) to the source nodes, notifying them of the congestion. In response to these choke packets, the source nodes would reduce their window size. Other mechanisms attempt to measure congestion in the network, by observing the delay experienced by packets and reducing the window size as delay increases. Yet another mechanism used by the Transmission Control Protocol (TCP) reduces the window size, in response to lost packets (packets for which an acknowledgment was not received). This is done based on the assumption that lost packets are due to buffer overflows and are a result of congestion. The flow-control mechanism used by TCP, and some of the problems associated with it, will be discussed in more detail in the next section.

One problem with using end-to-end windows for flow control is that, when congestion sets in on some link in the network, the node preceding that link will have to buffer a large number of packets. Consider, for example, a session operating over a multi-hop network and suppose that congestion sets in at one of the links along its path. With a window size of W, as many as W packets can be sent by the session into the network without receiving an acknowledgment. When a link becomes congested, all W packets associated with that session will arrive at the congested link and have to be buffered at the node preceding that link. With many simultaneous sessions, this can lead to significant buffering requirements at every node. An alternative, known as link-by-link window flow control, establishes for a session windows along every link between the source and the destination. These windows can be tailored to the specific link, so that a long delay link (e.g., satellite link) would have a large window, while a short delay link would have a smaller window. These link-by-link windows can be much smaller than the end-to-end windows and, as a result, the amount of buffering at each node can be significantly reduced. In effect, link-by-link windows distribute the buffering in the network evenly among all of the nodes rather than require the congested nodes to handle all of the packets. Of course, link-by-link windows are not always possible, for example, in networks that use datagram routing, sessions do not use a fixed path between the source and destination; hence, setting up windows on a link-by-link basis is not possible.

Rate Flow Control

One problem with window flow control is that in very-high-speed networks, where propagation delays are relatively large, very large windows are required, making window flow control ineffective. Another problem is that window flow control cannot be used for sessions that require guaranteed data rates, such as real-time traffic, because as delays through the network vary, the rate of the session is forced to vary as well. An alternative mechanism, which is more appropriate for high-speed networks and real-time traffic, is based on explicitly controlling the rate at which users are allowed to transmit. For a session that requires an average data rate of r packets per second, a strict implementation of a rate-control scheme would allow the session to transmit exactly one packet every $1/r$ s. Such implementation would amount to time-division-multiplexing (TDM), which is appropriate for constant rate traffic, but inefficient for bursty data traffic. Data sessions typically do not demand a constant transmission rate but are rather bursty, so that, at times, little if any transmission is required, and at other times, much higher rates are required. A more appropriate mechanism for supporting a bursty data session with an average rate of r packets per second is to allow the transmission of B packets every B/r seconds. In this way, bursts of up to B packets can be accommodated.

A common method for accomplishing this form of flow control is the leaky bucket method, shown in Fig. 6. In this scheme, a session of rate r has a "bucket" of permits for its use. The bucket, is constantly fed new permits at a rate of 1 every $1/r$ s and it can hold at most B permits. In order for a packet to enter the network, it must first obtain a permit from the bucket. If the bucket has no more permits, it must wait until such a permit becomes available. It is easy to see that, in this way, up to B packets can burst into the network all at once. An important parameter in the design of a leaky bucket rate control scheme is the bucket size B. Clearly, a small bucket size would result in strict rate control scheme and would be ineffective for bursty traffic. However, too large a bucket would be ineffective in controlling congestion. Again, as with the dynamic adjustment of window size in the window flow-control scheme, it is also sometimes desirable to dynamically alter the bucket size and rates given to a session based on traffic conditions in the network.

FLOW CONTROL IN PRACTICE

TCP Flow Control

The transmission control protocol (TCP) is the most commonly used transport layer protocol in today's internet. Virtu-

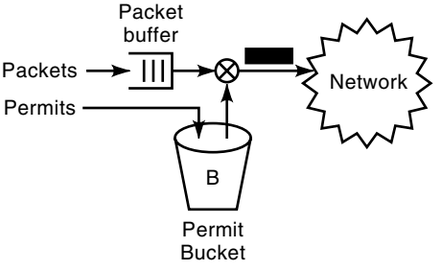

Figure 6. Leaky bucket flow control. Permits arrive at the bucket one every $1/r$ s and a packet must obtain a permit before entering the network.

ally all session-based traffic in the internet uses TCP. Among other things, TCP is responsible for flow control. There are a number of different TCP implementations (16–18), the details of which vary slightly from one another. The details of a particular standard are not emphasized here, but rather the general concepts that guide TCP flow control (19) are described.

TCP controls the flow of traffic in a session, using end-to-end windows. The key behind TCP flow control is the window size allocated for a given connection. For each connection, TCP determines a maximum allowable window size, W_{max}. The value of W_{max} is typically a function of the particular TCP implementation. Most TCP implementations use a value of W_{max} that is somewhere between 4 kbytes and 16 kbytes (20). Upon connection setup, the value of W_{max} is determined, based on the version of TCP used by the end stations.

Once the maximum window size is determined, the communication can begin. However, in order to prevent a new connection from overwhelming the system, communication does not begin with the maximum window size. Rather, communication starts with a window size of $W = 1$ packet, typically around 512 bytes, and the window size is gradually increased, in what is known as a *slow-start* phase. During the slow-start phase, the window size is increased by one packet for every acknowledgment that returns from the destination. Therefore, the window size is doubled with every successful transmission of a complete window. The slow-start phase continues until the window size reaches half of the maximum window size, at which point the communication turns into what is known as the *congestion-avoidance* phase. During the congestion-avoidance phase, the window size is increased by one packet for every successful transmission of a full window. Hence, during the congestion-avoidance phase, the window size is increased much more slowly than during slow start. The window size continues to increase in this way, until it reaches its maximum value of W_{max}.

In the above discussion we described how TCP sets its initial window size. In addition, TCP adjusts the window sizes in response to congestion in the network. TCP assumes that any packets lost in the network (e.g., packets that are not acknowledged in time) are due to buffer overflows resulting from congestion. In response, upon detecting a lost packet, TCP reduces the window size. Most TCP implementations (20) reduce the window size to one packet, at which point the window size is increased gradually back to the maximum value, in accordance with the slow-start and congestion-avoidance algorithms described above.

While TCP has been used effectively for many years, there are many shortcoming to its flow-control mechanism that make it ineffective for networks of the future. First, as discussed for general window flow control, it is not an effective mechanism for supporting sessions that require guaranteed data rates, such as real-time traffic. Second, as network transmission speeds increase, the window size needed to maintain unimpeded transmission has to be very large, especially over long delay links. With most versions of TCP having a maximum allowable window of around 16 kbytes, this is much too small for future high-speed networks (21,22). Furthermore, TCP's response to lost packets, as if they are due to congestion, may be appropriate for networks that experience very little loss due to transmission errors. However, for wireless or satellite networks, lost packets are likely to be due to transmission errors and, hence, the TCP responses of closing the window is not appropriate, and results in significant performance degradation (23). Finally, the TCP slow-start mechanism, which gives a session a small window and gradually increases the window size with time, prevents TCP from taking full advantage of the high transmission capacity offered by networks of the future.

Flow Control in ATM Networks

Asynchronous transfer mode (ATM) is a network technology developed to carry integrated traffic including data, voice, images, and video. ATM carries all traffic on a stream of fixed-size packets (cells), each comprising 5 bytes of header information and a 48 byte information field (payload). The reason for choosing a fixed-size packet is to ensure that the switching and multiplexing function could be carried out quickly and easily. ATM is a connection-oriented technology, in the sense that, before two systems on the network can communicate, they should inform all intermediate switches about their service requirements and traffic parameters. This is similar to the telephone networks, where a fixed path is set up from the calling party to the receiving party. In ATM networks, each connection is called a virtual circuit or virtual channel (VC), because it also allows the capacity of each link to be shared by connections using that link on a demand basis, rather than by fixed allocations. The connections allow the network to guarantee the quality of service (QoS), by limiting the number of VCs. Typically, a user declares key service requirements at the time of connection setup, declares the traffic parameters, and may agree to control these parameters dynamically as demanded by the network.

The available bit rate (ABR) service is one of the services in ATM developed to support data traffic. In other ATM services, network resources are allocated during connection establishment and sources are not controllable by feedback after a connection is established. ABR service, on the other hand, performs an end-to-end rate-based flow control, by requiring data sources to adapt their rates to their proper share of the available bandwidth by obtaining feedback information from the network.

The feedback information is carried in resource management (RM) cells of each connection. These RM cells are generated by the source between blocks of data cells and returned by the receiver in the backward direction. The feedback control operates in two modes—explicit binary or explicit rate indication. The explicit binary indication mode assumes that a congested network node will mark a specific field (equivalent to a binary bit) in the header of any passing data cell. The receiver monitors the fields of the data cells it receives and sets the congestion fields in the backward RM cells appropriately. The data sources can then increase, decrease, or stay at their current rates based on the congestion information contained in the backward RM cells received. The explicit rate indication mode assumes that network nodes are capable of computing the proper share of available bandwidth for each source and writing this amount into a specific field in passing RM cells. The source will then adjust its rates to no more than the amount indicated in the RM cells.

While the standards for ABR service specify the general behavior of the source and the receiver, the specific mechanism that governs when each network node should set the congestion field, or how it should compute the explicit rate, is

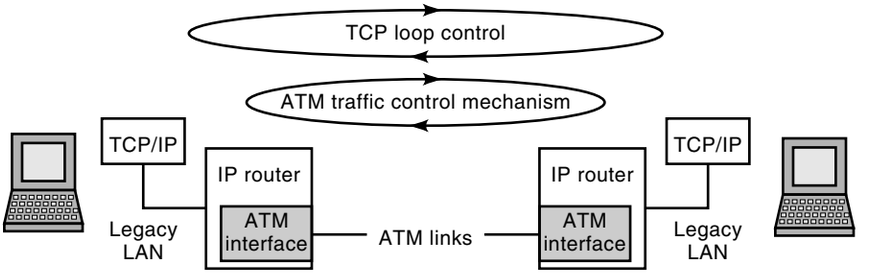

Figure 7. A typical network scenario where TCP traffic flows over an ATM network.

left to the discretion of network equipment vendors. Design objectives of such a mechanism include maximal utilization of network bandwidth, fairness in network use, and low cost, in terms of algorithm complexity and buffer space. For a good historic account of the development of the ABR service in ATM, see Ref. 24. Detailed descriptions of various approaches and mechanisms for flow control in ATM networks can also be found in Refs. 24–28.

ADVANCED ISSUES

Because most Internet applications are currently supported using TCP, a lot of research activities in the networking community have been focused on improving TCP performance. It has been realized that, in a high-latency network environment, the window flow-control mechanism of TCP may not be very effective, because it relies on packet loss to signal congestion, instead of avoiding congestion and buffer overflow (29). For bulky data connections, the arrival time of the last packet of data is of primary concern to the users, whereas delays of individual packets are not important. However, for some interactive applications such as Telnet, the user is sensitive to the delay of individual packets. For such low-bandwidth delay-sensitive TCP traffic, unnecessary packet drops and packet retransmissions will lead to significant delays perceived by the users.

It is suggested in some recent work (30) that the performance of TCP can be significantly improved if intermediate routers can detect incipient congestion and explicitly inform the TCP source to throttle its data rate before any packet loss occurs. This explicit congestion notification (ECN) mechanism would require modifications of existing TCP protocols. For example, a new ECN field can be implemented in the packet header, and will be used by an IP router, which monitors the queue size and, during congestion, marks the ECN field of an acknowledgment packet. The TCP source will then slow down after receiving the acknowledgment and seeing the ECN field being marked.

An additional motivation for using ECN mechanisms in TCP/IP networks concerns the possibility of TCP/IP traffic traversing networks that have their own congestion-control mechanisms (e.g., ABR service in ATM). Figure 7 shows a typical network scenario where TCP traffic is generated from a source connected to a LAN (e.g., Ethernet) aggregated through an edge router to an ATM network. Congestion at the edge router occurs when the bandwidth available in the ATM network cannot support the aggregated traffic generated from the LAN. Existing implementations of TCP only rely on packet drop as an indication of congestion, to throttle the source rates. By incorporating ECN mechanisms in TCP protocols, TCP sources can be informed of congestion at network edges and will reduce their rates before any packet loss occurs. The use of such ECN mechanisms to inform TCP sources of congestion would be independent of the congestion control mechanisms within the ATM networks.

Instead of incorporating ECN mechanisms in TCP, which requires modifications of TCP, it is proposed in Ref. 31, that congestion can be controlled by withholding at network edges the returned acknowledgments to the TCP sources. Such a mechanism has the effect of translating the available bandwidth in the ATM network to an appropriately timed sequence of acknowledgments. A key advantage of this mechanism is that it does not require any changes in the TCP end-system software.

There are other research efforts in improving TCP performance in a wireless network environment. The main problem here is that the noise in wireless transmission medium often can lead to corrupted TCP packets. Such corrupted packets are usually discarded at the destination, and the flow-control mechanism of TCP will mistakenly treat such packet loss as an indication of congestion and will throttle the source rate. Several solutions have been proposed to address this problem, most of which require modifying TCP protocols by decoupling the flow-control loops in the wireless medium from the wireline network. The readers are referred to Ref. 32 for detailed description of these mechanisms.

BIBLIOGRAPHY

1. D. P. Bertsekas and R. Gallager, *Data Networks,* Englewood Cliffs, NJ: Prentice-Hall, 1987.
2. M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis,* Reading, MA: Addison-Wesley, 1987.
3. R. Jain, Myths about congestion management in high speed networks, *Internetwork.: Res. Exp.,* **3** (3): 101–113, 1992.
4. N. McKeown, P. Varaiya, and J. Walrand, Scheduling calls in an input-queued switch, *Electron. Lett.,* **29** (25): 2174–2175, 1993.
5. M. J. Karol, M. G. Hluchyj, and S. P. Morgan, Input versus output queueing in a space-division packet switch, *IEEE Trans. Commun.,* **35**: 1347–1356, 1987.
6. J. Rexford et al., Scalable architectures for integrated traffic shaping and link scheduling in high-speed ATM switches, *IEEE J. Sel. Areas Commun.,* **15** (5): 938–950, 1997.
7. Hui Zhang, Service disciplines for guaranteed performance service in packet-switching networks, *Proc. IEEE,* **83**: 1374–1396, 1995.
8. A. Romanow and S. Floyd, Dynamics of TCP traffic over ATM networks, *IEEE J. Sel. Areas Commun.,* **13** (4): 633–641, 1995.

9. S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, *IEEE/ACM Trans. Network.,* **1** (4): 347–413, 1993.
10. H. Li et al., On TCP performance in ATM networks with per-VC early packet discard mechanisms, *Comput. Commun.,* **19** (13): 1065–1076, 1996.
11. A. Demers, S. Keshav, and S. Shenker, Analysis and simulation of a fair queueing algorithm, *Proc. ACM SIGCOMM,* **19** (4): 1–12, 1989.
12. J. C. R. Bennett and Hui Zhang, Hierarchical packet fair queueing algorithms, *IEEE/ACM Trans. Network.,* **5** (5): 875–889, 1997.
13. A. K. Parekh and R. G. Gallager, A generalized processor sharing approach to flow control in integrated services networks: The single-node case, *IEEE/ACM Trans. Network.,* **1** (3): 344–357, 1993.
14. S. J. Golestani, Network delay analysis of a class of fair queueing algorithms, *IEEE J. Sel. Areas Commun.,* **13** (6): 1057–1070, 1995.
15. D. Mitra and J. B. Seery, Dynamic adaptive windows for high-speed data networks with multiple paths and propagation delays, *Comput. Networks ISDN Syst.,* **25** (6): 663–679, 1993.
16. V. Jacobson, Berkeley TCP evolution from 4.3-Tahoe to 4.3 Reno, *Proc. Internet Eng. Task Force, 18th,* Vancouver, 1990.
17. L. S. Brakmo and L. Peterson, TCP Vegas: End-to-end congestion avoidance on a global internet, *IEEE J. Sel. Areas Commun.,* **13** (8): 1465–1480, 1995.
18. L. S. Brakmo, S. W. O'Malley, and L. Peterson, TCP Vegas: New techniques for congestion detection and avoidance, *Comput. Commun. Rev.,* **24** (4): 24–35, 1994.
19. V. Jacobson, Congestion avoidance and control, *Proc. ACM SIGCOMM,* **18**: 314–329, 1988.
20. W. R. Stevens, *TCP/IP Illustrated Vol. 1: The Protocols,* Reading, MA: Addison-Wesley, 1994.
21. V. Jacobson, R. Braden, and D. Dorman, TCP extensions for high performance, *Internet Eng. Task Force,* 1992, RFC-1323.
22. R. C. Durst, G. J. Miller, and E. J. Travis, TCP extensions for space communications, *Wireless Networks,* **3** (5): 389–403, 1997.
23. T. V. Lakshman and U. Madhow, The performance of TCP/IP for networks with high bandwidth-delay products and random loss, *IEEE/ACM Trans. Network.,* **5** (3): 336–350, 1997.
24. Ohsaki et al., Rate-based congestion control for ATM networks, *Comput. Commun. Rev.,* **25** (2): 60–72, 1995.
25. F. Bonomi and K. W. Fendick, The rate-based flow control framework for the available bit rate ATM service, *IEEE Network,* **9** (2): 25–39, 1995.
26. R. Jain, S. Kalyanaraman, and R. Viswanathan, The OSU scheme for congestion avoidance in ATM networks: Lessons learned and extensions, *Perform. Eval.,* **31** (1–2): 67–88, 1997.
27. P. Narvaez and K.-Y. Siu, Optimal feedback control for ABR service in ATM, *Int. Conf. Network Protocols,* pp. 32–41. Atlanta, GA, 1997.
28. K.-Y. Siu and H.-Y. Tzeng, Intelligent congestion control for ABR service in ATM networks, *Comput. Commun. Rev.,* **24** (5): 81–106, 1994.
29. K. Fall and S. Floyd, Simulation-based comparisons of Tahoe, Reno, and SACK TCP, *Comput. Commun. Rev.,* **26** (3): 5–21, 1996.
30. S. Floyd, TCP and explicit congestion notification, *Comput. Commun. Rev.,* **24** (5): 8–23, 1994.
31. P. Narvaez and K.-Y. Siu, An acknowledgment bucket scheme to regulate TCP flow over ATM, *Proc. IEEE Globecom,* **3**: 1838–1844, 1997.
32. H. Balakrishnan et al., A comparison of mechanisms for improving TCP performance over wireless links, *IEEE/ACM Trans. Network,* **5** (6): 756–769, 1997.

EYTAN MODIANO
MIT Lincoln Laboratory

KAI-YEUNG SIU
MIT d'Arbeloff Laboratory for Information Systems and Technology

NETWORK INTERCONNECTION.

See INTERNETWORKING.