Chapter in *The Handbook of Computer Networks*, Hossein Bidgoli (ed.), Wiley, to appear 2007

Network Traffic Management

Thomas M. Chen
Southern Methodist University, Dallas, Texas

OUTLINE:

KEY WORDS: quality of service, traffic control, admission control, policing, packet scheduling, priorities, queueing, buffers, packet discarding, congestion control, random early detection

ABSTRACT

Network resources are shared as needed by a community of users. Without effective traffic controls, networks are vulnerable to possible congestion when the offered traffic exceeds the network capacity, leading to serious deterioration of network performance. This chapter gives an overview of traffic control methods and their underlying principles. The main preventive controls are admission control and access regulation (policing). They seek to avoid congestion by limiting the amount of traffic entering the network. Within the network, packets may have different requirements for reliable or timely delivery. These requirements are recognized by packet scheduling and buffer management algorithms implemented in switches and routers. At the transport layer, protocols such as TCP can adapt their traffic rate to the level of congestion. Finally, this chapter covers the question of measurement and verification of network performance.

INTRODUCTION

Although an analogy between computer networks and automobile highways is simplistic, the view of networks as a type of infrastructure points out the need for traffic control. Highways have a limited capacity which can be exceeded when many people want to travel at the same time. Vehicles begin to slow down and back up in a congested area. The backup spreads if traffic approaches the congested area faster than traffic can leave. Similarly, computer networks are designed to handle a certain amount of traffic with an acceptable level of network performance. Network performance will deteriorate if the offered traffic exceeds the given network capacity.

Packets will suffer long queueing delays at congested nodes and possibly packet loss if buffers overflow.

Traffic management refers to the set of traffic controls within the network that regulate traffic flows for the purpose of maintaining the usability of the network during conditions of congestion. Traffic management has multiple goals. First, it attempts to distinguish different types of traffic and handle each type in the appropriate way. For example, real-time traffic is forwarded with minimal delay while best-effort traffic can afford to wait longer for any unused bandwidth. Second, traffic management responds to the onset of congestion. TCP is an example of a protocol that adapts the rate of TCP sources to avoid serious congestion. Third, traffic management seeks to maintain an acceptable level of network performance under heavy traffic conditions. The primary means of protection are admission control and access regulation (or policing) which limit the rate of traffic entering the network. By restricting the total amount of carried traffic, congestion will be avoided, and acceptable network performance will thus be sustained at the expense of blocking some amount of ingress traffic.

Quality of service

Quality of service (QoS) is closely related to network performance, but QoS metrics are oriented towards a user's end-to-end experience (Gozdecki et al., 2003). Network performance is measured mainly from the network provider's viewpoint. Network performance metrics are defined to provide a meaningful portrait of network behavior to the network provider or network manager. Metrics can include end-to-end or hop-by-hop performance, and may be aggregated over the entire user population. Examples of network performance metrics may be average end-to-end packet delay or fraction of packets lost per hop observed over an interval of time. While

interesting to the network provider, these metrics do not mean that a particular user will see this network performance. Therefore, they are not that meaningful to users concerned with their application's requirements.

In contrast, QoS is the end-to-end network performance seen from the viewpoint of a user's particular connection. QoS parameters are defined to quantify the performance needed and experienced by a user's application. For example, a real-time application might require QoS guarantees such as an end-to-end packet delay of 10 msec and packet loss rate of 10^{-6}. The user would be assured that his or her application would see this level of performance. At the same time, a different user may require a different QoS from the same network. Ultimately, a certain amount of subjectivity is involved. A user will judge the QoS provided by the network according to how well his or her application seems to work.

QoS may not necessarily be quantified in specific terms. "Hard" QoS may refer to specific guarantees, whereas "soft" QoS involves specific parameters but no guarantees. An example of hard QoS is a guarantee that no more than one percent of packets will experience a delay of 100 msec. Hard QoS can be verifiable by measurements of the QoS parameters. Soft QoS may specify target QoS parameters but offers no guarantees that the QoS will be met, hence soft QoS is not verifiable by measurements. Soft QoS is often defined for one service relative to another service. An example is a high priority service that is guaranteed to always have a shorter average packet delay than a low priority service, but there are no absolute guarantees for either high or low priority service.

ATM has taken the approach of hard QoS guarantees. Table 1 lists the ATM definitions of QoS parameters (ATM Forum, 1999). Although guarantees are offered on all QoS parameters, not all guarantees need to be specified explicitly. For example, it is assumed implicitly that the

bit error rate in ATM networks will be very low, so the cell error ratio will be very low and therefore unnecessary to negotiate explicitly. Likewise, misinserted cells are considered to be very rare and negligible under normal operating conditions. The QoS parameters that are explicitly negotiated are mainly the cell loss ratio, cell transfer delay, and cell delay variation.

Table 1. ATM definitions of QoS parameters

Aspect	Parameter	Definition
Accuracy	Cell error ratio	Fraction of cells delivered with bit errors
	Severely errored cell block ratio	Fraction of N-cell blocks delivered with M or more errored cells
	Cell misinsertion rate	Rate of appearance of misdirected cells from a different connection
Dependability	Cell loss ratio	Fraction of cells not delivered
Speed	Cell transfer delay	Maximum delay in delivering cells measured by p-percentile
	Cell delay variation (jitter)	Range between maximum and minimum cell delays, or deviation of cell delivery times from a reference pattern

Similar QoS definitions have been standardized for IP networks (Seitz, 2003). In the late 1990s, the IETF (Internet Engineering Task Force) IP Performance Metrics (IPPM) Working Group documented a set of QoS metrics for IP networks including one-way packet delay, roundtrip packet delay, and one-way packet loss. More recently, the ITU-T (International Telecommunication Union - Telecommunication Standardization Sector) has standardized the set of IP performance metrics listed in Table 2 (ITU-T, 1999).

Table 2. QoS metrics for IP networks

Aspect	Parameter	Definition
Accuracy	IP packet error ratio	Fraction of packets delivered with bit errors
	Spurious IP packet rate	Rate of appearance of misdirected packets at a network egress
Dependability	IP packet loss ratio	Fraction of packets not delivered
Speed	IP packet transfer delay	Mean or maximum delay in delivering packets

	IP packet delay variation (jitter)	Deviation in packet transfer delays from a reference packet transfer delay

Service level agreements

Expectations for QoS may be specified in the form of a service level agreement (SLA) between users and their network provider or between two network providers (Verma, 2004). An SLA formally spells out expectations about the level of service in a way that allows monitoring and verification. SLAs often include terms for billing and financial compensation in case the received service level falls short of expectations. Hence SLAs are legal contracts that must include clearly specified QoS metrics - typically availability, packet delay, jitter, packet loss, and throughput (Park, Baek, and Hong 2001). In common practice, SLAs are viewed more as insurance for unexpected lapses in network performance than as absolute guarantees due to the practical difficulties of measuring and verifying QoS metrics.

SLAs typically share a few common elements: criteria to be used for measuring service levels; a method for reporting outages and service failures; timeframe for the network provider to respond to identified failures; a method for escalating network problems; and procedures for enforcement and compensation for failure to meet SLA goals.

SLAs are typically limited to a single provider network. However, routes through the Internet can cross multiple networks. SLAs can be negotiated between network providers, but the issue becomes how to guarantee the end-to-end QoS to users (Pongpaibool and Kim, 2003).

General principles

Traffic management is a difficult problem because it fundamentally involves a balance between conflicting objectives: statistical sharing versus isolation. At one extreme, statistical

sharing of network resources without reservations is desirable to achieve a high efficiency. But since demand can exhaust resources, it is difficult (if not impossible) to guarantee QoS without reservations. At the other extreme, it is well known that QoS can be guaranteed by reserving resources for each traffic flow. Reservations isolate resources for each traffic flow so that flows do not have to compete for resources.

Another general principle is that traffic controls can be exercised simultaneously at various levels. At the node level, routers and switches are responsible for packet scheduling and selective packet discarding. Ingress nodes can perform access policing. Nodes may be capable of explicit congestion notification. At the network level, the network makes admission control decisions to accept or block new packet flows. At the higher level, transport or application layer protocols can be adaptive to the congestion level in the network.

Traffic control methods can be classified as preventive or reactive. Reactive methods are activated to ameliorate congestion after it is detected. For example, TCP reacts to congestion by reducing TCP source rates. A general principle is that prevention of congestion is preferred to reacting to congestion. If congestion occurs, packets will be lost, and it takes time to clear the queues in congested nodes. Preventive methods such as explicit congestion notification are preferred because congestion and packet loss may be avoided.

Overview of methods

In practice, networks generally depend on a variety of traffic controls employed at different points in the networks (El-Gendy, Bose, and Shin, 2003). The main preventive traffic control is admission control. Admission control gives the network an opportunity to make a decision whether to accept or reject a new traffic flow before the flow begins. Admission control

can be quite effective in protecting the network from congestion by blocking new traffic flows at the network ingress (Firoiu, Le Boudec, Towsley, and Zhang, 2002). Typically, admission control relies on a signaling protocol that allows applications to explicitly request a QoS or service class. Information about the new traffic flow (e.g., peak rate, average rate, burstiness) must be provided to the network at the same time so that sufficient resources can be allocated. A signaling protocol is not absolutely necessary if the required QoS can be inferred implicitly from knowledge about the traffic. For example, the telephone network uses admission control. Since every telephone call requires the same QoS, the required QoS does not have to be signaled explicitly.

Access or ingress policing is another preventive traffic control that typically accompanies admission control. Since admission control decisions are based on information about the accepted new traffic, it is important that source traffic conforms to the given parameters because excessive traffic could degrade network performance to the point of violating QoS guarantees. The most widely used algorithm for access policing is the leaky bucket. The leaky bucket algorithm is simple to implement with counters, and allows adjustable tolerance for bursty traffic sources. For conforming traffic, the access policing should be invisible and make no difference. For excess traffic, three actions are possible: the excess traffic is admitted; admitted but marked with lower priority; or discarded immediately.

Packet scheduling is an essential traffic control because it recognizes that packets have different forwarding requirements. Packet scheduling should recognize different service priorities and give preferential treatment to packets with more stringent delay requirements.

Buffer management is a complementary problem to packet scheduling. Whereas packet scheduling deals with the order of packets to depart from a buffer, buffer management dictates

how incoming packets should fill up limited buffer space. Buffer management should recognize different loss priorities and selectively discard packets of lower loss priority. In a way, loss priorities may be viewed as priorities related to space, while service priorities are related to time. In addition to simple loss priorities, packets may be discarded in coordination with higher layer protocols. For example, random early detection (RED) is a random packet discarding strategy that takes advantage of the congestion control algorithm in TCP to improve network throughput and stability (Floyd and Jacobson, 1993).

Instead of simply discarding packets during congestion, it is better to avoid congestion entirely if possible. Congestion might be avoided by inferring the state of congestion in the network by watching for long packet delays and packet losses. This does not require the network to provide any congestion information. However, the most effective way to avoid congestion is explicit congestion notification by the network. Protocols such as ATM and frame relay allow switches to mark packets when congestion appears to be imminent. Traffic sources are given enough advance warning to slow down and prevent congestion. This so-called closed loop control works only for some adaptable applications that can adjust their transmission rate dynamically (in contrast, admission control is open loop control because sources are not controlled by feedback after they are accepted).

ADMISSION CONTROL

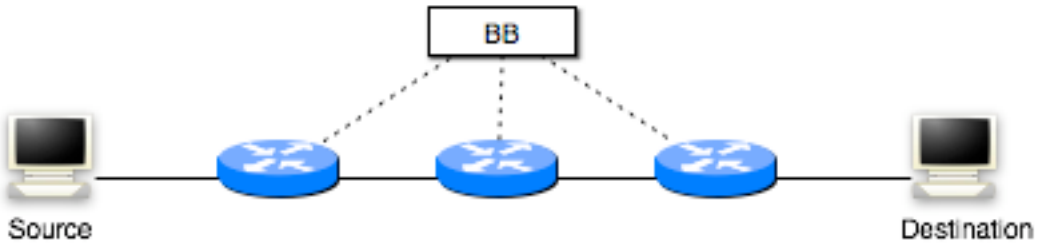
Admission control is one of the most effective means of congestion control. The objective is to prevent congestion by blocking new packet flows before the flow begins. If a new flow is accepted, the network is explicitly or implicitly agreeing to provide the required QoS for

the duration of the flow. This agreement is explicit if the admission control is carried out with a signaling protocol.

Many approaches to admission control can be found in the literature (Firoiu et al., 2002). As shown in Fig. 1, approaches can be classified according to where the acceptance decision is made: hop-by-hop, endpoint or edge router, or centralized bandwidth broker (BB). Hop-by-hop admission control follows the traditional approach of telecommunications networks. A signaling message requesting resources attempts to find a path through the network. Each router or switch along the path has an opportunity to accept and forward the request, or reject and block the request. Although this is a natural approach, the complexity required for routers limits the scalability to large networks.

(a)

(b)



(c)

Fig. 1. (a) Hop-by-hop admission (b) endpoint admission (c) centralized bandwidth broker.

The second approach leaves the admission decision to the source endpoint (host) or an edge (ingress or egress) router (Elek, Karlsson, and Ronngre, 2000). The decision can be based on data collected passively (without any extra effort) or actively by special probe packets. This approach simplifies the role of the core network, resulting in better scalability.

The third approach is also aimed at better scalability. Admission into each network domain is controlled by a centralized bandwidth broker (Zhang, Duan, Gao, and Hou, 2000). The available resources of the domain are monitored by the bandwidth broker, which makes all admission decisions and keeps track of accepted traffic flows. The bandwidth broker concept has two potential advantages. It keeps the core network stateless for scalability. Also, it can make network-wide optimal resource allocation decisions.

In order to make an acceptance or rejection decision, the network must estimate the amount of bandwidth and buffer resources needed for a new traffic flow. The required resources are compared with the available resources. The network also estimates the hypothetical impact of the new traffic flow on the current network performance. There must be assurance that the new flow will not cause the QoS for current traffic flows to fall below acceptable levels.

Approaches for admission control can be alternatively classified by how the decision is made: deterministic, stochastic, or measurement-based (Perros and Elsayed, 1996). Deterministic

approaches use worst-case bounds. In contrast, stochastic approaches assume statistical characteristics of the traffic for more accurate estimation of required resources. However, the accuracy of stochastic approaches depends heavily on the choice of traffic models. Traffic modeling is a difficult problem due to the dynamic nature of network traffic. Measurement-based approaches attempt to avoid the modeling problem by using measurements of the actual traffic instead of assumed traffic models.

Hop-by-Hop Admission Control

As in traditional telecommunications networks, a signaling message is sent along a path to request resources. The signaling message carries information needed for routers and switches along the path to make decisions about whether sufficient resources are available. If a router or switch has sufficient resources, it forwards the request to the next hop. If insufficient resources are available, it drops the request and notifies the source host by a "reject" signaling message.

RSVP (Resource Reservation Protocol) is a well known signaling protocol for IP networks capable of unicast or multicast reservations (Braden, Zhang, Berson, Herzog, and Jamin, 1997). In the basic reservation setup, a source sends an RSVP PATH message which is forwarded by each router, according to its routing table, until it reaches the destination, as shown in Fig. 2. The most important element is a TSpec object that provides information about the source traffic characteristics. Traffic characteristics in the TSpec object are specified in token bucket parameters: token rate, token bucket size, peak rate, minimum policed unit, and maximum packet size.

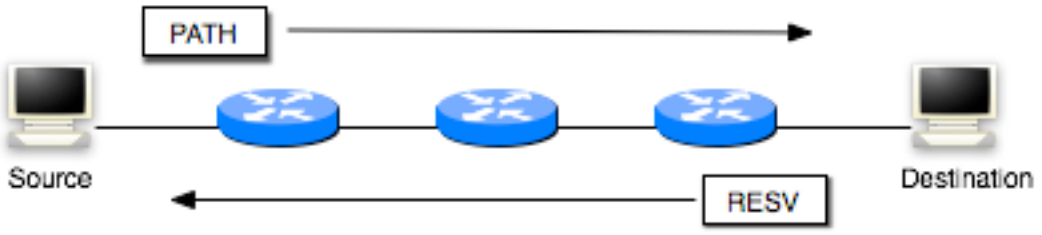


Fig. 2. RSVP messages.

The RSVP RESV message is returned from the destination to the source to request resources along the path traversed by the PATH message. In addition to a TSpec object, the RESV message can contain a controlled load or guaranteed load flow specification. Controlled load and guaranteed services are part of the IETF Integrated Services (IntServ) framework (Braden, Clark, and Shenker, 1994). Controlled load service is intended for real-time applications that can tolerate a limited amount of packet loss and excessive packet delay. It can be viewed as "better than best-effort" and equivalent to QoS seen from an unloaded network. The controlled load flow specification in the RESV message consists of the token bucket parameters as in the TSpec object. Guaranteed service provides "hard" and measurable QoS guarantees in terms of throughput and maximum packet delay. The controlled load flow specification in the RESV message consists of token bucket parameters and an RSpec object which includes a rate and a delay slack (delay variation) term.

RSVP suffers from two serious drawbacks: complexity and scalability (El-Gendy, Bose, and Shin, 2003). Routers must be programmed to understand the signaling protocol and perform packet classification, packet scheduling, and admission control. The scalability problem is caused by the need for traditionally stateless routers to keep state information for each flow. RSVP uses soft state, meaning that reservation states are automatically deleted if not refreshed after some time. However, it is clear that the number of flows in the Internet will only be

increasing over time, necessitating more state information to be maintained by RSVP-capable routers.

Endpoint/Edge Admission Control

Hop-by-hop admission control suffers from two drawbacks: the need for a signaling protocol, and the need for all routers or switches to keep track of flow reservations. An alternative idea is the source host could probe the network to determine if it can accommodate the new flow (Elek et al., 2000). For example, the source host (or ingress edge router) could inject a stream of probe packets at the same rate of the new flow and measure the packet delay and loss rate. This is a direct measurement of whether the new flow can be accommodated or not. The probe test may work better if routers along the path can mark the packets with congestion information (Gibbens and Key, 2001; Ganesh, Key, Polis, and Srikant, 2006).

The probing method has been proposed for voice over IP (VoIP) calls (Mase, 2004). It is claimed that the probing method over-controls call admission, so the method is modified so that call requests failing the probe test can be accepted with some adjustable probability.

Egress admission control places the decision at the egress edge router (Cetinkaya, Kanodia, and Knightly, 2001). The network is viewed as a black box, and the egress router just monitors data flows to infer the internal state of the network. The service capacity between ingress and egress endpoints is estimated. If the new flow combined with the existing flows can be accommodated by the service capacity, the new flow is admitted. The advantage of the method is no need for probe packets.

Bandwidth Brokers

In Diffserv, the core routers are stateless for scalability, and complex traffic controls are pushed to the edge routers. Although ingress or egress routers could take responsibility for admission control, another idea is to centralize admission control in a BB entity (Nichols, Jacobson, and Zhang, 1999). The BB keeps track of allocated and available resources in a Diffserv domain, makes all admission decisions, and configures the packet classifiers and traffic conditioners at the edges of the Diffserv domain to regulate incoming traffic.

The major advantage is scalability because core routers are concerned only with packet forwarding. Another advantage is the network-wide view of the BB. This network-wide view should enable the BB to make optimal resource allocation decisions (e.g., balance traffic across the network). However, there may be a performance issue because the BB keeps all state information and makes all decisions. As a Diffserv domain grows larger, the BB could become a processing and communications bottleneck. It is also a single point of failure. To alleviate this problem, a hierarchy of BBs may be possible.

It is envisioned that bandwidth brokers in adjacent Diffserv domains could negotiate with each other to establish end-to-end services across multiple domains. However, an inter-BB protocol does not currently exist.

Deterministic bounds

The deterministic approach to admission control assumes the worst case traffic scenario which is the peak rate for each traffic flow. A new flow is accepted if the sum of the peak rates for all flows is less than the physical capacity. It is obvious that traffic flows may be well below the peak rates in actuality, leading to considerable underutilization of capacity. However, this approach has appealing advantages. First, admission control is a simple decision. Second, the

peak rate is always known, so the physical capacity will never be exceeded and QoS can be guaranteed with certainty.

Stochastic bounds

For bursty traffic flows with random time-varying rates, it makes sense to allocate bandwidth less than the peak rates and assume some capacity can be saved by statistical multiplexing (Firoiu, et al., 2002). When a large number of flows are multiplexed, it is unlikely that all flows will be bursting at their peak rates simultaneously. Hence, an amount of bandwidth somewhat less than the sum of peak rates is needed.

The savings from statistical multiplexing is usually estimated from a finite-size queueing model. The queue represents the buffer in the switch or router making the admission control decision. The input to the queue is a stochastic process represents the random behavior of traffic sources. A number of traffic models have been proposed in the literature based on studies of historical traffic measurements (Michiel and Laevens, 1997). Common traffic models include the Poisson process, interrupted Poisson process, simple on-off model, Markov modulated processes (such as the Markov modulated Poisson process), and ARMA (autoregressive moving averages) time series (Adas, 1997). Due to the challenges of solving complex queueing systems, another popular model is the stochastic fluid buffer, where the input traffic is viewed as Markov modulated fluid flow instead of discrete packets. In any case, the delay through the queue and probability of buffer overflow are calculated for the hypothetical traffic. If the delay and buffer overflow probability meet the desired QoS, then the new traffic flow is accepted.

A popular alternative approach is to calculate the equivalent capacity for each traffic flow (Kelly, 1996). The appeal of equivalent capacity is a simple admission control decision. The

equivalent capacity for the total traffic is compared with the available physical capacity. The equivalent capacity for a source is derived by imagining the source as input into a finite queue. The equivalent capacity is the service rate resulting in the target buffer overflow probability. There are different expressions for equivalent capacity, depending on the source traffic characteristics. The equivalent capacity for multiple sources is derived similarly, where the aggregate traffic from the sources is the input into a finite queue. Generally, the equivalent capacity for multiple sources will be less than the sum of their individual equivalent capacities, due to the savings from statistical multiplexing.

Measurement-based admission control

The accuracy of statistical admission control depends on correct assumptions for the source traffic models. Measurement-based admission control is motivated by the difficulty of choosing a traffic model for a new packet flow based on limited information. A traffic source may provide descriptive parameters but they may be inaccurate or uncertain. Measurement-based admission control avoids the need to assume a traffic model by characterizing sources by statistics measured from the actual traffic instead.

Several measurement-based methods have been proposed (Shiomoto, Yamanaka, and Takahashi, 1999). The various methods all depend on measurements of the actual traffic made during a recent time window but differ in their usage of the measurements. For example, the measurements can be used to derive a marginal probability distribution for the source rates. The marginal probability distribution is then used to estimate the buffer overflow probability for a hypothetical finite queue. As a second example, the mean and variance of the source rates are measured. The effective capacity of the sources is derived from the mean and variance. As a

third example, a frequency analysis of the source rates reveals the low frequency and high frequency components. Queueing is considered to accommodate high frequency variations, and admission control is based on the low frequency components.

ACCESS CONTROL

Access control or policing refers to regulation of ingress traffic at the user-network interface. There are reasons for regulating traffic at the network edge and not internally within the network. At the network boundary, excessive traffic can be blocked before consuming any network resources. Moreover, it is natural to verify conformance of the traffic as close to the source as possible before the traffic shape is effected by other packet flows. Finally, individual packet flows at the network edge are slower than multiplexed flows in the core network.

Traffic contracts

When an admission control decision is made, that decision is based on traffic descriptors and QoS parameters provided by traffic sources. If the network admits a packet flow, it might be said that the user and network have agreed upon a traffic contract. Unlike SLAs, traffic contracts are a misnomer because they are not legal agreements. A traffic contract is an implicit understanding between the network and sources that both sides are cooperating in admission control. Sources are obligated to conform to the traffic descriptors that they provided for the admission decision. Excessive traffic will consume more network resources and deteriorate the QoS seen by all users. In accepting a new flow, the network is obligated to provide and sustain the requested QoS.

Traffic classification

Packets must be classified to determine their QoS requirements or which flow they belong to (Ji and Carchia, 2001). In the Intserv framework, reservations are made for individual flows. Packets belonging to the same flow can be identified by these IP header fields: source IP address, destination IP address, source port number, destination port number, and protocol.

Packet classification is difficult due to at least two reasons. First, routers must process terabytes of packets per second (Chao, 2002). Thus, packet classification must be done very quickly and efficiently (Yu, Katz, and Lakshman, 2005). Second, there may be an enormous number of flows going through a router. Classification of a packet to a particular flow essentially involves a table search, but the table could be enormous. Packet classification is an ongoing research problem that is central to router design (Van Lunteren and Engbersen, 2003; Wang et al., 2004; Baboescu and Varghese, 2005).

In the Diffserv framework, packets must be classified at the ingress to the Diffserv network such that an appropriate Diffserv codepoint (DSCP) can be assigned (Carpenter and Nichols, 2002). The DSCP is encoded in the first six bits of the Type of Service (ToS) field in the IPv4 packet header. The DSCP identifies the per hop behavior (PHB) to be applied to the packet.

Traffic regulation

The rate of traffic entering the network can be regulated as a means of enforcing the traffic contract. Since admission control decisions are based on traffic parameters given for a packet flow, it is important that flows do not exceed their stated parameters because excessive traffic would deteriorate the network performance seen by all users.

The goal of traffic regulation is to differentiate between conforming and non-conforming packets (according to the traffic contract). Conforming packets should be allowed immediate entrance into the network as if the traffic policer was transparent. Non-conforming packets may be discarded immediately or allowed entrance with some kind of packet marking. The former is preferred if the network load is heavy, then it would be better to discard the non-conforming packets to avoid any wasted use of network resources. The idea behind packet marking is that the purpose of the network is to carry packets. If the network load is light, it does no harm to carry non-conforming traffic. If congestion is encountered anywhere, the marked packets can be discarded first.

In the Diffserv framework, packets may also be classified into flows which are subject to policing and marking according to a traffic conditioning agreement (TCA). The TCA includes traffic characteristics (such as token bucket parameters) and performance metrics (delay, throughput) as actions required for dropping out of profile packets (Longsong, Tianji, and Lo, 2000). Out of profile packets are non-conforming packets, and either dropped or marked with a lower priority level.

Traffic shaping

Traffic shaping can be done at the source prior to entrance into the network or within the network. Traffic shaping at the source is a means of self regulation in order to ensure conformance to the traffic contract. Conformance is desirable to minimize the amount of traffic discarded at the network ingress.

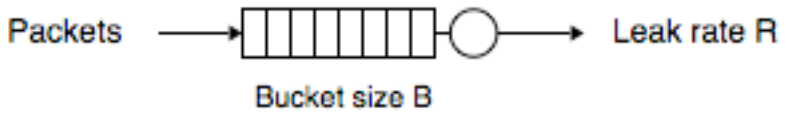
Traffic shaping within the network is not concerned with a traffic contract. The idea is that smooth traffic will cause smaller queues and hence incur shorter queueing delays and less

delay jitter (Rexford, Bonomi, Greenberg, Wong, 1997). Since traffic shaping is done by buffering, some queueing delay is added by the traffic shaper, but then smooth traffic should flow more quickly along the rest of the path. Overall, network performance is improved by keep traffic smooth (Elwalid and Mitra, 1997).

Leaky bucket algorithm

The leaky bucket algorithm is widely used for access control. There are different versions. The basic version consists of a buffer (bucket) of size *B* and a leak rate *R*, as shown in Fig. 3(a). Packets are conforming if they can join the buffer without overflowing. The contents of the buffer are emptied at the leak rate *R*. A larger bucket size allows greater variations from the rate *R*. A burst at higher rate can still be conforming as long as the burst is not long enough to overflow the bucket. Hence the maximum burst length is directly related to the bucket size.

One variation is the token bucket. Tokens at some generation rate *R* can collect in a bucket size of *B*, as shown in Fig. 3(b). A packet is conforming if it can find and consume a token waiting in the bucket. Again, a larger bucket allows longer bursts to be conforming.



(a)

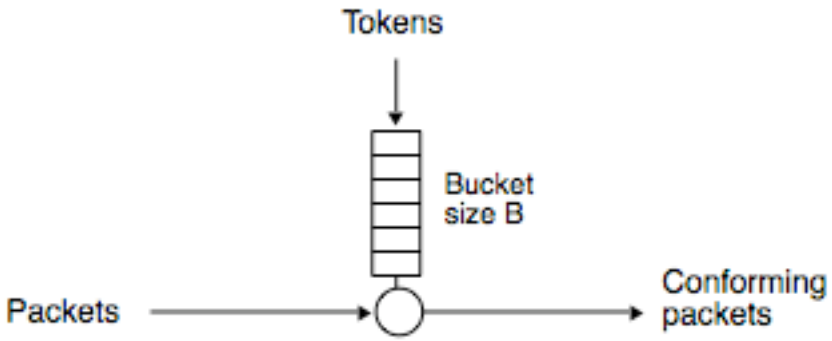


Fig. 3. (a) Leaky bucket policer (b) token bucket.

The leaky bucket algorithm can be easily implemented by a counter. Another advantage is the simple use of dual leaky buckets in tandem to regulate both average rate and peak rate. For example, the first leaky bucket can regulate the peak rate, and the second leaky bucket can regulate the average rate.

PACKET SCHEDULING

Packet scheduling addresses the different needs of buffered packets contending for the same link bandwidth (El-Gendy, et al., 2003). Packet scheduling should take into account the QoS requirements of each packet flow. A packet flow may require a minimum throughput, maximum end-to-end delay, or maximum packet delay jitter.

A general principle of packet scheduling is that the average waiting time for all packets does not change under different schedules. In other words, giving higher priority to transmit one packet earlier comes at the expense of making other packets wait longer. Benefits to one packet flow penalizes other packets. Hence, fairness is another issue when packet schedules are considered (Dua and Bambos, 2005).

Priorities

A simple and intuitive packet schedule gives preferential treatment according to static priorities. Each packet belongs to one of priorities levels 1 to *N*, where priority 1 is always transmitted first, priority 2 is transmitted when there is no priority 1 packets, and so on. The priority is static in the sense that a packet's priority never changes during its lifetime.

The advantage of static priorities is straightforward implementation. Different priority packets may be queued in separate buffers, as shown in Fig. 4. The priority buffers may be implemented as physically separate buffers or a single physical buffer separated into multiple virtual buffers.

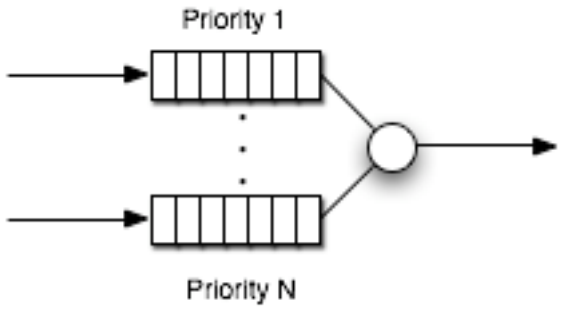


Fig. 4. Priority buffers.

A well-known disadvantage of static priorities is the possibility of starvation of low priorities. The high priority packets can consume all of the bandwidth and prevent low priority packets from receiving service. A possible solution to the starvation problem is dynamic priorities. For example, the priority of a packet can increase with its waiting time. As a packet is waiting, it will eventually receive service, ensuring some fairness. However, dynamic priorities are much more difficult to implement than simple priorities because the waiting times must be tracked for each packet.

Round robin

Round robin is another simple and intuitive scheduling algorithm. Each packet flow is queued in a separate buffer, and the head of line 1 is transmitted, then head of line 2, and so on, eventually back to line 1, as shown in Fig. 5. Empty queues are skipped over immediately to avoid wasting bandwidth.

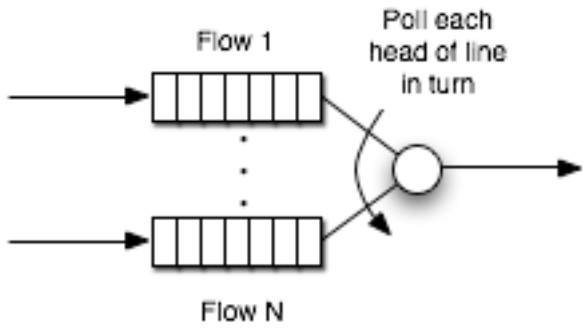


Fig. 5. Round robin scheduling.

Round robin is intuitively fair because each packet flow takes equal turns during a round or cycle. However, it may not be entirely fair. First, packets may be unequal in size. If the packets of one flow are long, that flow will consume more bandwidth compared to flows with shorter packets. Second, certain flows may naturally require more bandwidth than other flows (e.g., video versus voice).

A simple modification is weighted round robin. The packet flows are assigned weights $w_1, w_2,.., w_N$. A larger weight means that more packets are transmitted from that buffer in each turn. For example, if $w_1 = 2w_2$, then twice as many packets are transmitted from queue 1 than

queue 2 in each turn. Sophisticated schemes can adapt the weights per flow based on the traffic behaviors (Wang, Shen, and Shin, 2001).

Fair queueing

Fair queueing is closely related to round robin. Recall that a potential cause of unfairness in round robin scheduling is unequal packet sizes. A logical modification is round robin on the basis of bits instead of packets. Fair queueing might be considered as an approximation to bit-by-bit round robin. It is obvious that bit-by-bit fair queueing can not be implemented in actuality because packets must be transmitted in their entirety, not piecemeal as separate bits.

In fair queueing, the hypothetical finishing time of each packet under bit-by-bit round robin is calculated and associated with that packet (Demers, Keshav, and Shenker, 1989). The next packet chosen to transmit is always the packet waiting with the earliest hypothetical finishing time.

Along the same idea, weighted fair queueing (WFQ) is a packet-by-packet approximation to bit-by-bit weighted round robin. It is so-called fair in the sense that the i-th packet flow is guaranteed a fraction of bandwidth equal to at least $w_i/(w_1 + \ldots + w_N)$.

As a consequence of the guaranteed bandwidth, it may be expected that if a traffic source is rate limited, then the delay through a queueing system with WFQ would be bounded (Parekh and Gallager, 1993). Suppose each traffic flow is regulated by a token bucket with bucket size B_i and token rate R_i. Furthermore, assume that WFQ is used with relative weights $w_i = R_i$ for the flows. The maximum delay experienced by the i-th flow through the node will be bounded by $D_i \leq B_i/R_i$. The bounded delay is an important advantage of WFQ because it means that admission control can guarantee a maximum end-to-end packet delay.

Deadline scheduling

A widely studied class of scheduling algorithms are based on deadlines (Conway, Maxwell, and Miller, 2003). When a packet arrives at the buffer, it is assigned a deadline d which is the time when it should be transmitted. For example, a real-time packet might have a deadline that is a short time after its arrival time a, whereas a nonreal-time packet might have a deadline much farther in the future. The finishing time f of a packet is the sum $f = a + w + x$, where w is the waiting time before starting service, and x is its service time (i.e., transmission time). If a packet is transmitted by finishing time f, the packet's lateness is $l = f - d$. A packet with positive lateness has not met its deadline, whereas a packet with negative lateness is considered early. Typically, a penalty is associated with lateness but not earliness.

A number of optimal schedules are known, depending on the criterion to optimize. The shortest processing time (SPT) discipline takes packets in order of increasing service time. That is, the next packet to transmit is always the packet waiting with the smallest service time. The SPT schedule is known to minimize the average waiting time and minimize the average lateness (averaged over all packets).

In certain cases, the most late packet is more important than the average lateness. Another well known optimal schedule is the earliest due date (EDD) or earliest deadline first (EDF) discipline. The next packet to transmit is always the packet waiting with the earliest deadline. The EDD schedule is known to minimize the maximum lateness. Suppose the packet that is most late under any schedule has lateness l_{max}. EDD is optimal in the sense that no other schedule will realize a smaller l_{max} than the EDD schedule.

BUFFER MANAGEMENT

Buffer management deals with how packets are given preferential access to limited buffer space. If packets are allowed into a full buffer, then packets already waiting in the buffer must be discarded to make space available. Hence buffer management is also the problem of selective packet discarding (Labrador and Banerjee, 1999).

Loss priorities

In theory, packets may be marked with an explicit loss priority. ATM and frame relay are examples of protocols that include a bit in the packet header to indicate loss priority. The ATM cell header has a cell loss priority (CLP) bit, and cells with CLP=1 are dropped first at congested switches. The frame relay header has a discard eligibility (DE) bit. Frames marked with DE=1 are discarded before DE=0 frames.

Although the TOS field in the IP packet header has a bit to maximize reliability (or presumably minimize the likelihood of dropping), the TOS field has ambiguous interpretations and has been preempted by the Diffserv codepoint. Thus, IP does not allow explicit loss priorities. However, strategies for discarding IP packets are an active area of research because packet loss effects TCP performance. These strategies are usually called active queue management (AQM) (Ryu, Rump, and Qiao, 2003). The most important example, random early detection (RED), is discussed later.

Pushout schemes

The simplest packet discarding strategy is tail drop. When a buffer is full, arriving packets are dropped. This strategy has a potential “lock out” problem where a subset of flows

can monopolize the buffer space and prevent other flows from access. In fact, higher rate flows will gain more buffer space at the expense of lower rate flows. Another drawback is that buffer congestion can continue for a long time when the traffic is TCP. TCP slows down when a dropped packet is detected by a retransmission time-out. A packet at the queue tail was transmitted later than a packet at the head of the queue. Therefore, a packet dropped from the tail will time-out at a later time than a packet dropped from the head. TCP sources will continue to transmit for a longer time before slowing down.

A third drawback is that simple tail drop does not recognize that packets have different loss priorities. If an arriving packet has high loss priority, it should be able to access the buffer space by discarding a packet of lower priority already in the buffer. The packet of lower priority is said to be pushed out (Roy and Panwar, 2002). There are mostly three push out strategies: dropping low priority packets near the head first, near the tail first, or randomly from anywhere in the buffer. For TCP traffic, dropping from the head of the queue works better (Lakshman, Neidhardt, and Ott, 1996). In general though, optimal pushout schemes are not known (Sharma and Viniotis, 1999).

Random early detection

The simple tail drop strategy causes the undesirable phenomenon of global synchronization of TCP flows. TCP sources search for the proper sending rate by essentially increasing the rate until a dropped packet is detected by a retransmission time-out. The details of the TCP congestion avoidance algorithm are described later. When a buffer overflows, it takes some time for TCP sources to detect a packet loss. In the meantime, the buffer continues to overflow, and packets will be discarded from multiple TCP flows. Thus, many TCP sources will

detect packet loss and slow down at the same time. Even if the TCP sources were initially out of phase, they will become synchronized. The synchronization phenomenon causes underutilization and large queues. Underutilization occurs when all sources slow down (in "slow start"), then large queues are accumulated when all sources simultaneously increase their transmission rates.

RED eliminates the synchronization phenomenon and thereby achieves better utilization and smaller queue (Floyd and Jacobson, 1993). A great advantage is that RED can be implemented as a buffer management strategy without any change to the existing TCP protocol. Instead of waiting for the buffer to overflow, RED drops a packet randomly with the goal of losing packets from multiple TCP flows at different times. Therefore, the TCP flows slow down and ramp up at different times. Since they are unsynchronized, their aggregate rate is smoother than before. From a queueing theory viewpoint, smooth traffic achieves the best utilization and shortest queue.

RED keeps track of the average queue length, calculated as an exponentially weighted moving average of the instantaneous queue length. The average queue length is compared to a minimum threshold, T_{min}, and a maximum threshold, T_{max}. The packet drop probability is calculated with the help of an intermediate drop probability q shown in Fig. 6. Below T_{min}, no packets are dropped because the short queue means no congestion. Above T_{max}, all packets are discarded because the large queue is a sign of serious congestion. Between T_{min} and T_{max}, the intermediate drop probability q increases linearly to a maximum probability P. The actual drop probability p_{drop} is calculated as

$$p_{drop} = \frac{q}{1 - q \cdot count}$$

where *count* is the number of undropped packets since the last dropped packet. In this way, RED attempts to drop packets in proportion to the rates of connections.

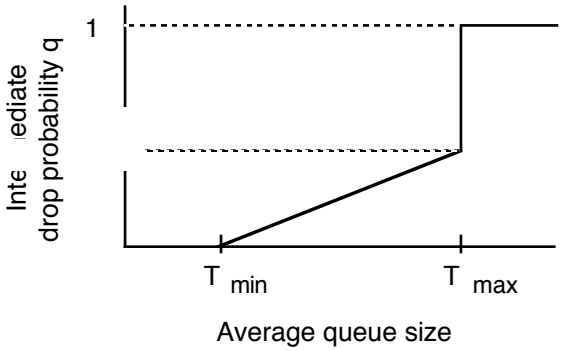


Fig. 6. RED probabilities of discarding.

Since its introduction, RED has found wide commercial acceptance. Under certain traffic conditions, RED performs effectively if its parameters are properly tuned. However, it can be difficult to determine the proper parameters. Moreover, several problems can arise depending on the traffic, such as unfairness, lowered throughput, and increased delay. Consequently, numerous variations have been proposed, including flow RED (FRED), weighted RED (WRED), stabilized RED (SRED) (Ott, Lakshman, and Wong, 1999), BLUE (Feng, Shin, kandlur, and Saha, 2002), PURPLE (Pletka, Waldvogel, and Mannal, 2003), adaptive virtual queue (AVQ) (Kunniyur and Srikant, 2004), and predictive AQM (PAQM) (Gao, He, and Hou, 2002), just to name a few. RED continues to be an active subject of research (Zhu, Wang, Aweya, Ouellette, and Montuno, 2002; Bohacek, Shah, Arce, and Davis, 2004).

CONGESTION CONTROL

Congestion control is a very broad term that is sometimes used interchangeably with traffic control. Here congestion control refers to actions taken by hosts to prevent serious congestion. Congestion control can be classified as reactive or predictive (proactive). Reactive congestion control ameliorates congestion after it occurs. Generally, hosts observe the symptoms

of congestion (packet loss and long packet delays) and then slow down. Predictive congestion control attempts to avoid congestion altogether by slowing down before the onset of congestion. Avoidance of congestion is obviously preferable, but predictive congestion control usually depends on explicit congestion information provided from the network.

Reactive congestion control

Reactive congestion control takes action when congestion is detected. A prime example of this approach is the congestion avoidance algorithm in TCP. TCP takes responsibility for congestion control in the Internet because IP is too simple by design to handle congestion control. Strictly speaking, TCP does not actually avoid congestion. TCP pushes the network gently to the brink of congestion and then backs off. This is necessary because a TCP source does not know the appropriate transmission rate and must search for it.

A TCP source limits itself by means of a congestion window. The congestion window is adjusted dynamically to the level of network congestion by an AIMD (additive increase multiplicative decrease) algorithm (Jacobson, 1988). The onset of congestion is detected by a retransmission time-out. As shown in Fig. 7, a parameter *ssthresh* (slow start threshold) is set to half of the congestion window when congestion was encountered. The congestion window is closed to one TCP segment and the congestion control algorithm goes into slow start. In the slow start phase, the congestion window can increase at an exponential rate as long as acknowledgements for packets are returned promptly. When the congestion window size reaches *ssthresh*, the congestion control algorithm enters the congestion avoidance algorithm. In this phase, the congestion window increases by one segment for every roundtrip time until congestion is encountered again. The idea is that linear increase during congestion avoidance is a

cautious approach to the previous congestion point. When congestion is encountered, the *ssthresh* parameter is set to half of the congestion window, and the congestion control algorithm repeats again.

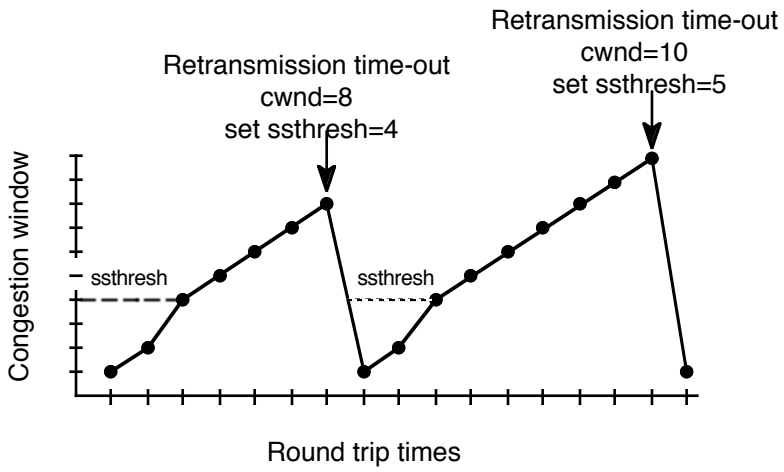


Fig. 7. TCP congestion control algorithm.

Predictive congestion avoidance

The congestion avoidance algorithm in TCP is forced to probe for the appropriate transmission rate by pushing the network to the brink of congestion because the IP network is incapable of providing congestion information. A capability for explicit congestion notification is preferable as a method of "early warning." Protocols such as ATM and frame relay include explicit forward congestion indication (EFCI) and backward/forward explicit congestion notification (BECN/FECN), respectively.

Explicit congestion notification allows routers and switchers to give early warning to sources before congestion occurs. This advance warning can give sufficient time for traffic sources to reduce their rates in time to avoid congestion. This is a proactive approach to avoiding congestion instead of a reactive approach to ameliorate congestion. It has been shown that

explicit congestion notification improves the performance of TCP with active queue management (Yan, Gao, and Ozbay, 2005).

Explicit congestion notification in the forward direction refers to marking packets as they pass through a router or switch. Packets in the forward direction are delivered to the destination, and the destination must have a separate means of communicating congestion information to the source. The overall delay is one roundtrip time. For efficiency, explicit congestion notification in the backward direction goes directly to the source. However, it is harder to implement because a router or switch must find and mark a packet going in the backward direction.

The current version of IP does not include a capability for explicit congestion notification but an IETF proposal calls for using the last two bits from the TOS field in the IP packet header (Ramakrishnan and Floyd, 1999). One bit for ECN capable transport (ECT) would allow hosts to signal whether they are capable of making use of ECN. The second bit for congestion experienced (CE) would allow routers or switches to signal a state of congestion to hosts. An open research issue is how routers should choose packets to mark (Leung and Muppala, 2001).

QOS MONITORING

Observability is a prerequisite for controllability. Measurements of QoS can verify the effectiveness of traffic management and provide evidence to validate SLAs. In addition, QoS monitoring can identify network performance problems and help to tune traffic controls (Jiang, Tham, and Ko, 2000).

Network layer performance monitoring

Since the Internet has not historically supported QoS, there has not been a compelling need for sophisticated performance monitoring at the IP layer. By far, most methods and tools at the IP layer are based on *ping* (ICMP echo and echo response messages) or *traceroute* (which exploits the time-to-live field in the IP packet header).

A few large-scale performance monitoring projects use ping and traceroute to make regular measurements between multiple selected points in the Internet (Chen and Hu, 2002). The basic idea is that performance measured on these selected routes will reflect the overall performance of the entire Internet if the monitoring points are numerous and geographically distributed. Repeated pings are an easy way to obtain a sample distribution function of round trip time and an estimate of packet loss ratio (reflected by the fraction of unreturned pings). Although round trip times measured by ping are important, especially for adaptive protocols such as TCP, ping is unable to measure one-way packet delays without additional means for clock synchronization among hosts such as GPS. Another drawback to ping is the low priority or blocking given to pings by some networks, because pings are invasive and possibly involved in some types of security attacks.

Transport/application layer performance monitoring

Another monitoring method is TCP bulk transfer between selected sites. One site acts as the source and another as the sink for TCP bulk transfer. The packets are recorded by *tcpdump* and analyzed for packet delay and loss.

Measurements at the transport layer is appealing because they are closer to the application's perspective. Moreover, they do not depend on ICMP or traceroute, which are not entirely reliable. At the transport layer, the idea is to run a program emulating a TCP session to

send test traffic through the Internet. The performance of the TCP connection can be measured in terms of delay, loss, and throughput.

For example, *TReno* (traceroute Reno) is an emulation of TCP to measure throughput or bulk transfer capability. It combines traceroute and an idealized version of the flow control algorithm in Reno TCP. Other examples are *ttcp* (throughput TCP) and *netperf*. Ttcp is a client/server benchmarking program to measure throughput and retransmissions. Netperf is a network performance benchmarking program for measuring bulk data transfer and request/response performance using TCP or UDP. One host runs netperf and another runs netserver. Tests can be a TCP stream performance test, UDP bulk transfer, or TCP request/response test.

Similarly, network performance can be measured at the application layer. For example, NetIQ's commercial Chariot or Qcheck software agents are installed and configured at hosts with scripts to emulate various applications and collect application-level performance (throughput, delay) measurements. Application layer measurements have the advantage of measuring performance that is reflective of what applications are seeing. On the other hand, special software is needed at selected sites and it is an active monitoring method that adds traffic into the network.

CONCLUSIONS

As seen in this chapter, traffic management is a challenging problem covering an impressive breadth and depth. The breadth is due to various traffic controls that should work together. Moreover, many details are involved in engineering each control to work effectively.

Traffic management continues to be an active area of study, not because solutions are not possible but because many solutions are possible. The "optimal" solution continues to be elusive because many factors (fairness, effectiveness, efficiency, practicality) usually need to be considered, allowing for different opinions of optimality.

Even as next-generation network provide more bandwidth, traffic management will still be an important problem. Next-generation networks will support a greater variety of applications. Different applications require different treatment by the network. The intelligence to recognize different traffic and their requirements implies the need for sophisticated traffic management.

GLOSSARY

Access policing: regulation of ingress traffic at the user-network interface.

Active queue management (AQM): intelligent buffer management strategies enabling traffic sources to respond to congestion before buffers overflow.

Admission control: an open loop control method to accept or block traffic flows before flows start.

Differentiated services (Diffserv): an IETF framework supporting QoS classes by means of Diffserv code point marking, per hop behaviors, and stateless core routers.

Explicit congestion notification: a proactive congestion avoidance method involving routers or switches conveying congestion information to hosts.

Integrated services (Intserv): an IETF framework supporting QoS based on RSVP reservations per flow.

Leaky bucket: an algorithm for policing a specific traffic rate while allowing a limited variation around that rate.

Quality of service (QoS): the end-to-end network performance defined from the perspective of a specific user's connection.

Random early detection (RED): an active queue management approach for preventing synchronization of TCP connections.

Resource Reservation Protocol (RSVP): an IETF standardized signaling protocol for IP networks.

Service level agreement (SLA): a contract between users and network provider or between two network providers specifying expectations about the level of service in a way that allows monitoring and verification.

Signaling protocol: rules for exchanging messages to request reservations of network resources.

Traffic contract: an implicit agreement between users and the network to meet QoS requirements as long as ingress traffic conforms to specified traffic descriptors.

Traffic shaping: smoothing the traffic rate to improve network performance.

REFERENCES

Adas, A. (1997). Traffic models in broadband networks. IEEE Commun. Mag., 35, 82-89.

ATM Forum (1999). Traffic Management Specification Version 4.1.

Baboescu, F., and Varghese, G. (2005). Scalable packet classification. IEEE/ACM Trans. on Networking, 13, 2-14.

Bohacek, S., Shah, K., Arce, G., and Davis, M. (2004). Signal processing challenges in active queue management. IEEE Signal Proc. Mag., 21, 69-79.

Braden, R., Zhang, L., Berson, S., Herzog, S., and Jamin, S. (1997). Resource reservation protocol (RSVP) – version 1 functional specification. IETF RFC 2205.

Braden, R., Clark, D., and Shenker, S. (1994). Integrated services in the Internet architecture: an overview. IETF RFC 1633.

Carpenter, B., and Nichols, K. (2002). Differentiated services in the Internet. Proc. of the IEEE, 90, 1479-1494.

Cetinkaya, C., Kanodia, V., and Knightly, E. (2001). Scalable services via egress admission control. IEEE Trans. on Multimedia, 3, 69-81.

Chao, J. (2002). Next generation routers. Proc. of the IEEE, 90, 1518-1558.

Chen, T., and Hu, L. (2002). Internet performance monitoring. Proc. of the IEEE, 90, 1592-1603.

Conway, R., Maxwell, W., and Miller, L. (2003). Theory of Scheduling. Dover Publications.

Demers, A., Keshav, S., and Shenker, S. (1989). Analysis and simulation of a fair queueing algorithm. In proc. of ACM SIGCOMM 89.

Dua, A., and Bambos, N. (2005). On the fairness delay trade-off in wireless packet scheduling. In proc. of IEEE Globecom 2005.

Elek, V., Karlsson, G., and Ronngre, R. (2000). Admission control based on end-to-end measurements. In proc. of IEEE INFOCOM'00.

El-Gendy, A., Bose, A., and Shin, K. (2003). Evolution of the Internet QoS and support for soft real-time applications. Proc. of the IEEE, 91, 1086-1104.

Elwalid, A., and Mitra, D. (1997). Traffic shaping at a network node: theory, optimum design, admission control. In proc. of IEEE INFOCOM'97, 444-454.

Feng, W-C., Shin, K., Kandlur, D., and Saha, D. (2002). The BLUE active queue management algorithms. IEEE/ACM Trans. on Networking, 10, 513-528.

Firoiu, V., Le Boudec, J-Y., Towsley, D., and Zhang, Z-L. (2002). Theories and models for Internet quality of service. Proc. of the IEEE, 90, 1565-1591.

Floyd, S., and Jacobson, V. (1993). Random early detection gateways for congestion avoidance. IEEE/ACM Trans. on Networking, 1, 397-413.

Ganesh, A., Key, P., Polis, D., and Srikant, R. (2006). Congestion notification and probing mechanisms for endpoint admission control. IEEE/ACM Trans. on Networking, 14, 568-578.

Gao, Y., He, G., and Hou, J. (2002). On exploiting traffic predictability in active queue management. In proc. of IEEE INFOCOM'02, 1415-1424.

Gibbens, R., and Key, P. (2001). Distributed control and resource marking using best-effort routers. IEEE Network, 15, 54-59.

Gozdecki, J., Jajszczyk, A., and Stankiewicz, R. (2003). Quality of service terminology in IP networks. IEEE Commun. Mag., 41, 153-159.

ITU-T (1999). Rec. I.380: Internet Protocol Data Communication Service – IP Packet Transfer and Availability Performance Parameters.

Jacobson, V. (1988). Congestion avoidance and control. In proc. of ACM SIGCOMM'88, 314-329.

Jiang, Y., Tham, C-K., and Ko, C-C. (2000). Providing quality of service monitoring: challenges and approaches. In proc. of IEEE NOMS 2000, 115-128.

Kelly, F. (1996). Notes on effective bandwidths. In Stochastic Networks: Theory and Applications (pp. 141–168). Oxford: Oxford University Press.

Kunniyur, S., and Srikant, R. (2004). An adaptive virtual queue (AVQ) algorithm for active queue management. IEEE/ACM Trans. on Networking, 12, 286-299.

Labrador, M. and Banerjee, S. (1999). Packet dropping policies for ATM and IP networks. IEEE Commun. Surveys, 2, 2-14.

Lakshman, T., Neidhardt, A., and Ott, T. (1996). The drop from front strategy in TCP and in TCP over ATM. In proc. of IEEE INFOCOM'96.

Leung, K., and Muppala, J. (2001). Effect of different marking strategies on explicit congestion notification (ECN) performance. In proc. of IEEE ICC 2001, 1812-1816.

Longsong, L., Tianji, J., and Lo, J. (2000). A generic traffic conditioning model for differentiated services. In proc. of ICC 2000, 1305-1309.

Mase, K. (2004). Toward scalable admission control for VoIP networks. IEEE Commun. Mag., 42, 42-47.

Michiel, H., and Laevens, K. (1997). Teletraffic engineering in a broadband era. Proc. of the IEEE, 85, 2007-2033.

Nichols, K, Jacobson, V., and Zhang, L. (1999). A two-bit differentiated services architecture for the Internet. IETF RFC 2638.

Ott, T., Lakshman, T., and Wong, L. (1999). SRED: stabilized RED. In proc. of IEEE INFOCOM'99, 1346-1355.

Parekh, A., and Gallager, R. (1993). A generalized processor sharing approach to flow control in integrated services networks: the single node case. IEEE/ACM Trans. on Networking, 1, 344-357.

Park, L-T., Baek, J-W., and Hong, J. (2001). Management of service level agreements for multimedia Internet service using a utility model. IEEE Commun. Mag., 39, 100-106.

Perros, H., and Elsayed, K. (1996). Call admission control schemes: a review. IEEE Commun. Mag., 34, 82-91.

Pletka, R., Waldvogel, M., and Mannal, S. (2003). PURPLE: predictive active queue management utilizing congestion information. In proc. of IEEE LCN'03, 21-30.

Pongpaibool, P., and Kim, H. (2003). Guaranteed service level agreements across multiple ISP networks. In proc. of 4th Int. Workshop on Design of Reliable Commun. Networks (DRCN 2003), 325-332.

Ramakrishnan, K., and Floyd, S. (1999). A proposal to add explicit congestion notification (ECN) to IP. IETF RFC 2481.

Rexford, J., Bonomi, F., Greenberg, A., and Wong, A. (1997). Scalable architectures for integrated traffic shaping and link scheduling in high speed ATM switches. IEEE J. on Sel. Areas in Commun., 15, 938-950.

Roy, R., and Panwar, S. (2002). Efficient buffer sharing in shared memory ATM systems with space priority traffic. IEEE Commun. Letters, 6, 162-164.

Ryu, S., Rump, C., and Qiao, C. (2003). Advances in Internet congestion control. IEEE Commun. and Surveys, 5, 28-39.

Seitz, N. (2003). ITU-T QoS standards for IP-based networks. IEEE Commun. Mag., 41, 82-89.

Sharma, S., and Viniotis, Y. (1999). Optimal buffer management policies for shared-buffer ATM switches. IEEE/ACM Trans. on Networking, 7, 575-587.

Shiomoto, K., Yamanaka, N., and Takahashi, T. (1999). Overview of measurement-based connection admission control schemes. IEEE Commun. Surveys, 2, 2-13.

Van Lunteren, J., and Engbersen, T. (2003). Fast and scalable packet classification. IEEE J. on Se. Areas in Commun., 21, 560-571.

Verma, D. (2004). Service level agreements on IP networks. Proc. of the IEEE, 92, 1382-1388.

Wang, H., Shen, C., and Shin, K. (2001). Adaptive-weighted packet scheduling for premium servce. In proc. of IEEE ICC 2001, 1846-1850.

Wang, P-C., Chan, C-T., Hu, S-C., Lee, C-L., and Tseng, W-C. (2004). High-speed packet classification for differentiated services in next-generation networks, IEEE Trans. on Multimedia, 6, 925-935.

Yan, P., Gao, Y., and Ozbay, H. (2005). A variable structure control approach to active queue management for TCP with ECN. IEEE Trans. on Control Systems Technol., 13, 203-215.

Yu, F., Katz, R., and Lakshman, T. (2005). Efficient multimatch packet classification and lookup with TCAM. IEEE Micro, 25, 50-59.

Zhang, Z-L., Duan, Z., Gao, L., Hou, Y. (2000). Decoupling QoS control from core routers: a novel bandwidth broker architecture for scalable support of guaranteed services. In proc. of ACM SIGCOMM'00.

Zhu, C., Wang, O., Aweya, J., Ouellette, M., and Montuno, D. (2002). A comparison of active queue management algorithms using the OPNET modeler. IEEE Commun. Mag., 40, 158-167.

FURTHER READING

The literature on traffic management is quite extensive due to its broad scope and allowance for different viewpoints. A number of useful books cover the topic of traffic management in depth, such as:

Chao, J., and Guo, X. (2002). Quality of Service Control in High Speed Networks. NY: Wiley & Sons.

Jha, S., and Hassan, M. (2002). Engineering Internet QoS, Norwood, MA: Artech House.

Wang, Z. (2001). Internet QoS: Architectures and Mechanisms for Quality of Service. San Francisco, CA: Morgan Kaufmann.