QoS-Sensitive Flows:

Issues in IP Packet Handling

Saleem N. Bhatti and Jon Crowcroft
University College London

Imagine a network technology that routinely loses information, experiences variable and unpredictable delay in data delivery, and makes no distinction between applications with different communication requirements. Next, imagine that the use of this technology is growing exponentially and that, even as networks become increasingly congested, its applications are being extended to global-scale voice and video.

This scenario describes the situation we face in supporting real-time applications over an IP-based infrastructure. Such applications involve data flows that have quality-of-service (QoS) requirements. A QoS-sensitive data flow cannot readily tolerate the effects of packet loss, delay (and delay variation, or jitter), and fluctuations in network throughput.

In this article, we examine the problems in attempting to make the existing IP infrastructure do what it was never designed to do—provide QoS support for integrated services to both real-time and non-real-time applications. We concentrate on the issues and principles concerning router modification for IP packet handling.

IP-based networks were never designed for real-time traffic, yet QoS support in such networks is needed to accommodate both global use and the more demanding applications now emerging. Changes in packet handling, in particular, will help provide QoS support in IP networks.

PACKET HANDLING IN TRADITIONAL IP NETWORKS

IP offers a connectionless datagram service that gives no guarantees concerning data delivery and has no notion of *flows*, in which many datagrams form a sequence meaningful to an application. For example, an audio application can package 40-millisecond audio time slices in individual datagrams. The sequence and timeliness of the datagrams has meaning to the application, but the IP network treats them as individual and unrelated. There is no signaling at the IP level—no way to inform the network that it is about to receive traffic with particular handling requirements, and no way for IP to tell users to back off when there is congestion.

IP routers forward individual datagrams on the basis of single metrics and network destination addresses. The routing is dynamic, not fixed, allowing IP packets in a single flow to change network paths in case of router overloads or failures. The absence of a fixed path for traffic of a single flow means, in practice, that the network cannot guaranteed consistent QoS during a session. Even if the path remains stable, the connectionless datagram service, does not protect the packets of one flow from those of another.

Packets in output queues are serviced in a simple first-come, first-serve (FCFS) order, with the packet at the front of the queue transmitted first.

1089-7801/00/$10.00 ©2000 IEEE

Figure 1 describes how packets are processed in a traditional IP router. This traditional IP forwarding mechanism provides the current best-effort IP service. To modify routing behavior so that it can support QoS, we must establish the key parameters of a real-time packet flow and determine how we might control them.

INTEGRATED SERVICE CONCEPTS AND REQUIREMENTS

The Internet protocol architecture was designed to provide robust and scalable support for applications that require not much more than reliable end-to-end data transfer,[1] for example, FTP and telnet. In 1992, Clark et al.[2] described a way to evolve the original Internet architecture to an integrated services network that could support traditional applications as well as emerging real-time applications. They identified four architectural components: a service level, a service interface, an admission control mechanism, and scheduling mechanisms.

The following is a simple description of the interactions between the components:

- *A service level is defined.* This includes all the service semantics: descriptions of how packets should be treated within the network, how the application should inject traffic into the network, and how the service should be policed. Knowledge of the service semantics must be available within routers and applications.
- *An application invokes a service using the service interface and a signaling protocol.* The invocation includes specific information about the traffic characteristics required for the flow, such as data rate. The network indicates if the service invocation was successful and might also inform the application of any service violation, either by the application's use of the service or from a network failure.
- *Admission control uses the information in the service invocation*, plus knowledge about other service requests it is currently supporting, to determine if it can accept the new request. Admission control, typically implemented in the routers, polices service use to ensure that applications do not use more resources than they have requested.
- *Once a service invocation has been accepted, the network employs router mechanisms for scheduling and queue management* to ensure that the packets within the flow receive the requested service.

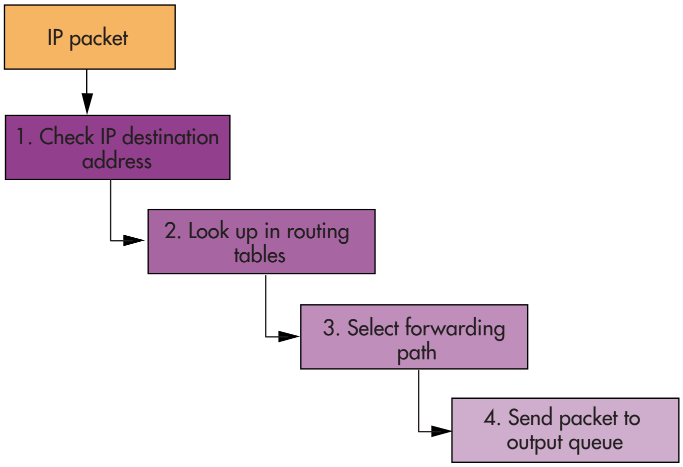
IP packet
1. Check IP destination address
2. Look up in routing tables
3. Select forwarding path
4. Send packet to output queue

Figure 1. Packet processing in a traditional IP router. When a packet arrives at a router, the router (1) looks at the destination address, (2) identifies routing-table entries that may offer a forwarding path for it, (3) selects the best match from the candidate entries using longest-prefix matching for the IP address, and (4) sends the packet to the output queue for the correct outgoing interface.

A critical feature of this integrated-services architecture is *signaling*—talking to the network. Signaling is required in connection-oriented networks, and the signaling part of such networks offers a natural point for communicating particular application requirements. But datagram networks are connectionless, and typically don't require signaling. Any signaling mechanism introduced to IP networks should be compatible with current Internet operation and should not constrain or change the operation of existing applications and services.

SCHEDULING AND QUEUE MANAGEMENT

As application traffic moves from the end systems toward the network's center, packets from different flows are intermixed. Figure 2 (on the next page) shows how the traffic patterns of three flows (voice-over-IP, FTP, and Web traffic) are disrupted when the flows are aggregated. The flows arrive at a router on three different input lines but are destined for the same output line. The router forwards them in the order they arrive, which causes the packets to experience variable delay. This may not be a problem in FTP, but it can be for VoIP. While some delay correction is possible at the end systems (through buffers to smooth the delay variations), the overall effect limits real-time applications on the Internet.

To correct this, we must protect QoS-sensitive flows, giving them priority handling in routers but

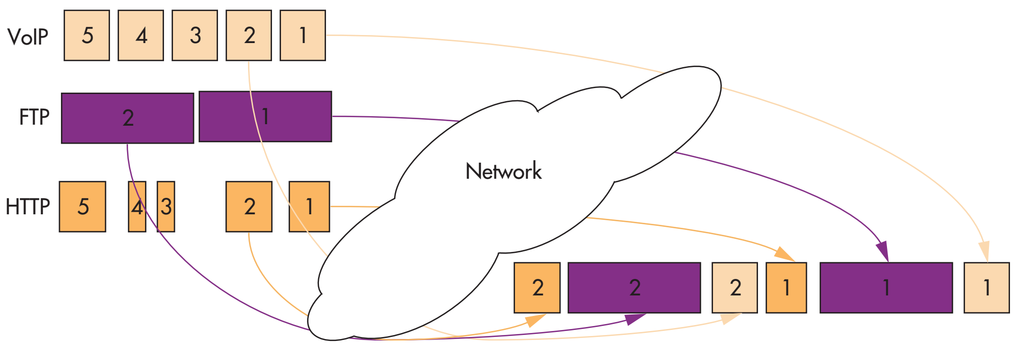

Figure 2. Traffic flow aggregation. When several flows arrive at a router on different input lines, the router forwards them in the order they arrive, without regard to QoS requirements.

still maintaining fairness so that other, nonpriority traffic is also served. This is the scheduling policy.

Figure 3 shows a simple router schematic. The role of the scheduler is primarily to decide in what order service requests (incoming packets) are allowed access to resources (output queues and output lines). Secondarily, it manages the service queues (output buffers). The secondary role reflects the need to manage the server's finite resources in a way consistent with the router's established policy, even in the face of excessive service requests.

Kleinrock's Conservation Law

First come, first serve (FCFS) is a work-conserving scheduling policy. According to such policies, the router is never idle when packets are waiting to be serviced. The Conservation Law, an important theorem developed by Kleinrock,[1] helps in analyzing scheduling disciplines.

Consider N flows arriving at a scheduler, so that the traffic rate of connection n is λ_n, where $1 \leq n \leq N$, and λ is the mean packet rate. If we assume that flow n has a mean service time of μ_n, then its mean utilization of a link is $\lambda_n \mu_n$. Let ρ_n represent this mean link utilization, and let q_n be the mean waiting time caused by the scheduler for the packets of flow n.

According to the Conservation Law, the following expression holds true for work-conserving schedulers:

$$\sum_{n=1}^{N} \rho_n q_n = C$$

where C is a constant. Thus, if we give one flow a lower delay or higher data rate in a work-conserving scheduler, it must be by increasing the delay or reducing the data rate for another flow.

Reference

1. L. Kleinrock, *Queuing Systems, Vol. 2: Computer Applications*, Wiley Interscience, New York, 1975.

Scheduling Disciplines

The scheduler implements policy through an algorithm called a scheduling discipline. In a traditional IP router, the scheduler uses a first-come, first-serve (FCFS) scheduling policy. This policy is simple to implement, and the main concern of most FCFS-scheduled routers is queue management of excessive packet bursts and finite buffer space.

FCFS is a *work-conserving scheduling discipline*, which means that the router is never idle when there are packets waiting to be serviced. In an important theorem, Kleinrock proved that it also means schedulers based on such disciplines cannot give one flow preferential treatment without hurting the others. For more information on this phenomenon, see the sidebar, "Kleinrock's Conservation Law."

Keshav[3] describes four main requirements for scheduling disciplines:

- *Ease of implementation.* Keeping the algorithmic complexity and the state information requirements low makes it is easier to implement in hardware and thus more suitable for high-speed networking.
- *Fairness and protection.* All other things being equal, no flow should receive les than any other flow requesting use of the same resources. The max-min fairness criterion (detailed in the sidebar, "Max-Min Fairness Criterion," p. 52), lets us precisely allocate resources by allowing per-hop local fairness for each flow, which results in global fairness for all flows. The protection requirement states that no flow should suffer due to the misbehavior or characteristics of any other flow.
- *Performance bounds.* To support QoS in networks, it should be possible to specify performance bounds to ensure that the network handles the

flow appropriately. If a scheduler is providing a guaranteed service, it must be able to work with deterministic performance bounds. If relative priorities for flows were being specified, then the scheduler should allow performance bounds to be specified statistically or probabilistically.
- *Admission control.* Where deterministic performance bounds are specified, the network might need to perform admission control before it begins flow transmission.

It is difficult to find a scheduling discipline that fulfills all these requirements. Among the disciplines being deployed, many are based on the *generalized processor sharing* reference model.[4,5] GPS is a work-conserving scheduler that provides max-min fairness and flow protection. It can establish probabilistic/ statistical (relative) as well as deterministic performance bounds. Unfortunately, it cannot be implemented, as the analysis requires that each flow serve only an infinitesimally small amount of data.

For scheduling disciplines that can be implemented, Golestani[6] describes two fairness bounds for assessing the performance of a scheduler that emulates GPS: relative fairness bound (RFB) and absolute fairness bound (AFB). The RFB assesses fairness by comparing the service given to flows served by the same scheduler.

Alternatives to FCFS Scheduling

A number of alternatives to first-come, first-serve scheduling have been developed, some of which use mechanisms that emulate the GPS properties. We describe three of the more widely used ones here. They are mainly rate-based schedulers that control the data rate allocated to a flow.

Priority scheduling. The simplest non-FCFS scheduler is priority scheduling. There is a queue for each flow, and queues are serviced in order of priority (higher numbers before lower numbers). Busy higher priority queues could thus prevent lower priority queues from being serviced, a situation known as *starvation.* This system is easy to implement but not max-min fair, and it must be used with some other mechanism to police traffic into the queues.

Weighted round-robin. The simplest GPS emulation is weighted round-robin (WRR), in which queues are serviced round-robin in proportion to a weight assigned for each flow or queue. Each queue is visited once per round. Normally, at least one packet is transmitted from each nonempty queue.

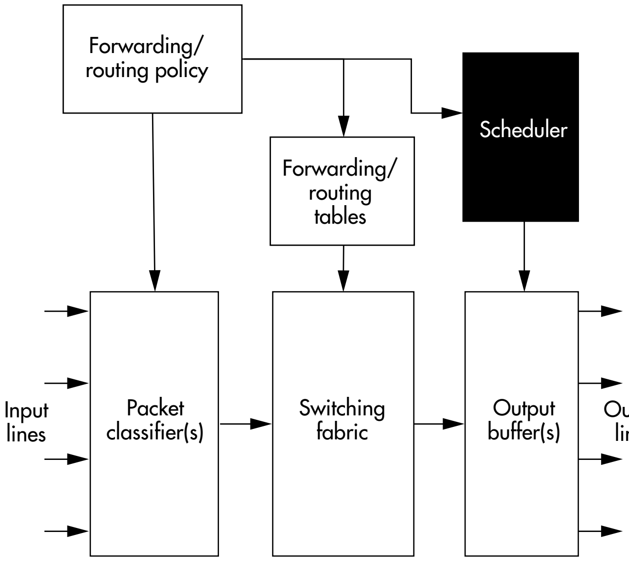

Figure 3. A simple router schematic. The scheduler keeps track of when to transmit the next packet.

The assigned weight is normalized by dividing it by the average packet size for each flow/queue. If this is unknown beforehand, then WRR might be unable to offer the correct service rate for the flow. For long-lived flows, WRR service is fair; however, for short-lived flows, flows with small weights, or many flows, WRR might exhibit unfairness.

Deficit round-robin. DRR bypasses the requirement to know the average packet size for each flow.[7] Each nonempty queue has a deficit counter that begins at zero. As the scheduler visits each nonempty queue, it reads the packet at the head of the queue and tries to serve one quantum of data. If the counter for a queue is zero, then a packet at the head of the queue is served if it is less than or equal to the quantum size. If a packet cannot be served, then the value of the quantum is added to the deficit counter for that queue.

DRR works best for small packet sizes and a small number of flows; it suffers from unfairness behavior similar to that for WRR.

Weighted fair queuing. WFQ, part of Parekh and Gallagher's original work,[4] is a GPS emulation that uses a computation to tag packets in flows with a finish number, indicating approximately when the packets would have finished being served had they been subject to true GPS scheduling. WFQ approximates GPS by using a bit-by-bit round-robin service. A packet of size N bits at the head of a queue

Max-Min Fairness Criterion

In max-min fair share, resources are allocated according to some simple rules. The demands of the resource flows are ordered in increasing demand so that flows with the lowest demands are allocated resources first. This means that resources with small demands are likely to be allocated all that they ask for, as indicated in the example below:

$$m_n = \min\left(x_n, M_n\right) \qquad 1 \le n \le N$$

$$M_n = \frac{C - \sum_{i=1}^{n-1} m_i}{N - n + 1}$$

C : Capacity of resource (maximum resource)

m_n : Actual resource allocation to flow n

x_n : Resource demand by flow n, $x_1 \le x_2 \cdots \le x_N$

M_n : Resource available to flow n

It is possible to assign weights to the resource demands so that some demands receive a greater share of the resources than others. In this case, the rules for max-min fair share are modified as follows:

- Flows are allocated resources in order of increasing weighted demand (demands are normalized by the weight).
- No flow gets more than it needs.
- Flows that have not been allocated as they demand get a weighted share of the available resources.

will have complete service after the scheduler has performed *N* rounds of visiting all other queues. Packets cannot, however, be transmitted one bit at a time from multiple queues, so finish number tags indicate when they would have completed service if they had been served bit by bit. Packets with the smallest finish number are thereby the ones that should be transmitted first.

The WFQ scheduler applies weights to the finish number. WFQ can also be used to attach specific performance bounds to individual flows for data rate by adjusting the flow's weight. A flow with a higher weight gets a lower delay.

The finish time depends on the round number *R*, a counter that tracks how many times all queues have been visited. *F* and *R* are both functions of absolute time *t*. As the number of flows changes, so does the rate of change for *R*. The *R* rate of change decreases as the number of flows increases and increases as the number of flows decreases. This latter property leads to a problem called iterated deletion. A flow is deleted if it has completed (becomes empty). When a flow empties, *R* increases at a faster rate, so the value of *R* approaches the finish number of other packets, already queued, faster than originally evaluated. This might cause some of the other flows to complete also, leading to a further increase in *R*'s rate of change. To keep track of *R*, a WFQ system must recompute the value of *R* whenever a packet arrives or leaves.

Although WFQ provides max-min fairness, protection, and the potential for specific (quantitative) QoS bounds, it is also relatively complex to implement and has a high per-packet processing overhead. Nevertheless, it is used as the basis for many routers offering QoS control (such as some from Cisco and Lucent).

Non-Work-Conserving Disciplines

Work-conserving schedulers are never idle if there are packets waiting. A non-work-conserving scheduler, however, can be idle even if packets are waiting to be serviced, because the scheduler waits for packets to become eligible for transmission. Packet transmission eligibility is determined in different ways. One way is to ensure that it always spends only a fixed time at a router; another is to establish a fixed end-to-end delay for a packet. The main advantage of non-work-conserving disciplines is that they reduce jitter, making downstream traffic more predictable, but the cost is higher overall end-to-end delay. Such systems are also complex to build. Consequently, non-work-conserving disciplines are not used widely and are still considered a research issue.

PACKET CLASSIFICATION AND FLOW MARKING

To handle a packet correctly at the router, a scheduler must be able to determine the flow to which a particular packet belongs. Potential packet information a scheduler could use includes:

- *Existing IP headers.* These are obvious candidates for flow identifiers. Routers make forwarding decisions by looking at the existing IP-header fields, so looking at other IP-header fields presents less computational and implementation overhead than other options.
- *Additional headers.* The IETF could change the packet header format to introduce a specific flow identifier. However, this would require changes to the basic protocol, and, even if additional header extensions were used, they might not be recognized or implemented everywhere.
- *Transport-level headers.* Routers could use port numbers and/or other transport-level identifiers. However, the router would have to first locate the transport-level header, which might

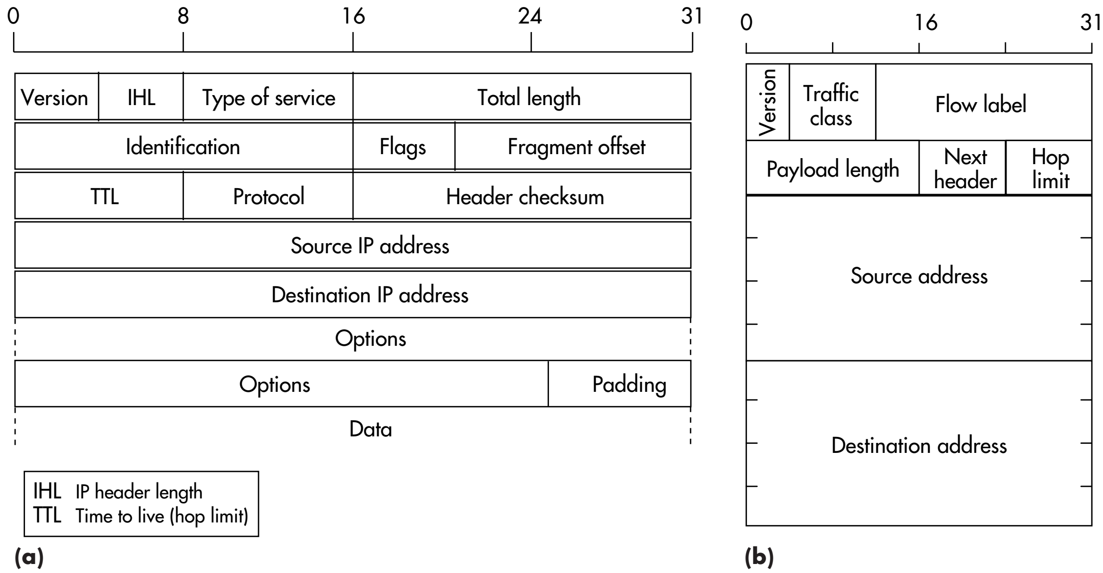

Figure 4. Packet headers for (a) IPv4 and (b) IPv6. There are various fields, such as the IP address and protocol number in IPv4, that can identify a flow. IPv6 has an explicit flow-label, but no agreement has been reached on its exact usage.

be behind optional IP-level headers, resulting in additional overhead. More importantly, transport-level information by itself may not be sufficiently unique to identify a flow.

- *Application-level headers.* This information, in the packet payload, can identify flows; however, adding application-specific information to packet payloads requires the routers to know about the many application-level protocols. The routers would also need to be upgraded when applications change or when new applications are introduced into a network.

Figure 4 shows the packet headers for IPv4 and IPv6. Various combinations of header fields (such as IP address and protocol number) can serve as flow identifiers.

The type of service (ToS) byte in IPv4 could also provide a limited form of packet marking. In fact, the IETF DiffServ working group is redefining the use of the IPv4 ToS byte (along with the IPv6 traffic-class byte) to include the ability to re-mark the value of this byte as a packet travels through the network, thereby dynamically changing how subsequent routers handle that packet. IPv6 has an explicit flow-label, but its use is not yet fully defined.

There are two key points for flow marking and packet classification:

- Any per-packet processing above the normal forwarding functions is overhead and will slow down a packet's progress through the router.
- For classification information to be available for flows as they are created and destroyed, the router must hold and maintain state information for the flow. This, too, is overhead, requiring extra memory and processing.

Packet classification can involve various IP and transport-level header-field values. Processing these fields and identifying flows can incur a large processing overhead at the router. Over the past few years much work has gone into developing high-speed packet classification mechanisms via new data structures and algorithms (see, for example, Srinivasan et al.[8] and Gupta and McKeown[9]).

DiffServ Model

The IETF DiffServ working group (http://www.ietf.org/html.charters/DiffServ-charter.html) is looking at a more scalable model for identifying flows, based on traffic aggregation rather than individual per-application instance flows. In the DiffServ model, packets are classified as belonging to a flow if they have the same marking in their ToS byte (IPv4) or traffic-class byte (IPv6), as shown in Figure 4. The DiffServ work specifies new, common syntax and semantics for these differentiated-services (DS) bytes.

Different DS-byte values—DS code-points—distinguish one flow from another. The meanings of some code-points are fixed, although work is in progress to set up dynamic bindings using the

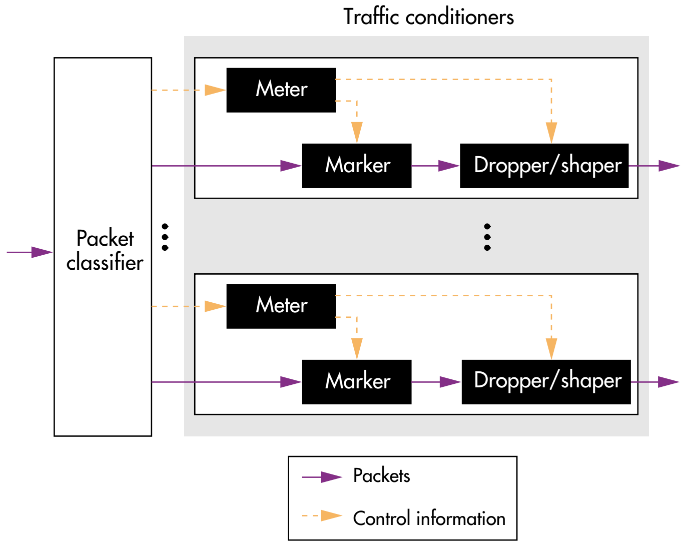

Figure 5. DiffServ packet processing scheme. Meters monitor code-point traffic, while markers, droppers, and shapers implement the service-level policy for each code-point.

resource reservation setup protocol (RSVP). Backbone providers and ISPs link to each other's networks with well-defined service-level agreements specifying the handling requirements for each code-point. Packet classification, marking, and policing based on the code-point and perhaps other header fields is performed at network edges, and in the core of the network only the code-point determines packet handling requirements, reducing the processing required for each packet at the network core.

Figure 5 shows the main functions of the DiffServ packet processing scheme. Here, meters (which can be token bucket specifications) monitor traffic with each code-point. The markers, droppers, and shapers implement the service-level policy for each code-point. After the packets have left the traffic conditioners, they must still be processed as in Figure 1 to determine their forwarding path.

DiffServ does not use per-flow admission control. Instead, applications must be aware that available capacity for a given DS code-point may fluctuate depending on users and applications. Applications, therefore, might need to adapt—for example, by being responsive to congestion signals (see the section, "Congestion Control").

DiffServ makes a trade-off: Protection and fairness for individual per-application instance flows is not necessarily guaranteed, but DiffServ scales better and is likely to be easier to implement than IntServ/RSVP. If fine-grained, per-flow classification is done at the network edges, then as long as service-level agreements are adhered to, there should be some reasonable approximation to fairness and protection across a DiffServ-capable network.

TRAFFIC DESCRIPTIONS AND ADMISSION CONTROL

Scheduling mechanisms can give fairness and protection, but developers have been striving to produce mechanisms that are easy to implement. The two other requirements for scheduling disciplines—performance bounds and admission control—offer a way to specify, respectively, a user's intended flow requirements and a network's test to see if the performance bounds can be met.

Important flow parameters often have end-to-end significance. These include data rate, delay, jitter, ordered delivery, loss, and error rates. End-to-end delay is hard to control on an IP-based infrastructure, because end-to-end paths cannot be fixed. However, it is possible to find the end-to-end delay for a given network path and report it to the upper layers.

Error detection or correction tends to be application-specific. For example, file transfer cannot tolerate any bit errors, but packet audio—depending on its encoding—can tolerate a few. Coping with errors at the network level requires knowledge about the application at the network level, which is undesirable for the reasons mentioned earlier. Ordered delivery is also hard to provide at the network level unless there is state for per-flow packet sequence at the network level and sequence numbers at the IP level. Moreover, packets can instead be reordered at the receiver using transport- or application-level sequence numbering.

Part of the IntServ work includes a widely accepted traffic description of a token bucket, illustrated in Figure 6. A bucket of "credits," or tokens, provides opportunities for transmission. The bucket of size b (bytes) fills with tokens at a rate r (Bps), and data packets consume these tokens as they are transmitted. However, there is also a peak rate, p (Bps), and bursts of data packets can be sent at this rate so long as enough tokens are available. At any time t, the source should not have sent more data than $rt + b$. A meter can use this property to monitor the flow, and routers can police a flow by ensuring that packets do not violate this expression.

To ensure that the network always has sufficient capacity to service all its flows, a flow might need to "ask" permission to use specific resources. In telephone networks, we ask the network for permission to attempt a connection, and the network says yes (or no) by sending a dial tone (or not). Unlike a phone network, which offers only one service with a specific, fixed-resource requirement, an integrated services network could provide resources for a diverse range of services. Using a description of the flow's traffic (and perhaps other information), a network can control a flow's admission to the network.

Admission control is still a research area. Indeed, a body of research is developing on network calculus, which provides mathematical tools to determine characteristics of network performance and assist in admission control, capacity planning, and performance management. (See the Network Calculus Project pages at the Institute for Computer Communications and Applications, http://icawww.epfl.ch/.)

IP uses RSVP to advise the network of a flow's requirements. RSVP also provides a means of signaling admission control decisions back to the user.[10] Currently, RSVP is specified mainly for the IETF IntServ working group, but it has some severe scaling problems. One of the main problems is that RSVP provides a flow granularity to the level of interaction between individual applications. For this mechanism to succeed, backbone routers must hold per-flow state. RSVP uses the destination IP-address, port number, and protocol number to identify a flow. The thought of backbone routers trying to hold state for millions of voice and video calls, as well as forwarding packets and keeping routes updated, borders on overwhelming.

CONGESTION CONTROL

Congestion is the excessive delay or loss of data due to excess traffic. For real-time applications, we consider a maximum one-way delay to be 150 ms. Routers can run out of buffer space and be forced to drop packets. Even if all sources obey their respective token bucket specifications, several of them can peak at once, and the burst of data could cause congestion.

In TCP, the loss or delay of an ACKnowledgment packet indicates congestion and causes the TCP flow to slow its transmission rate. Random Early Detection is a queue management system that uses this TCP behavior to prevent congestion. As the queues (buffers) at a router start to fill, the probability that RED will drop a packet grows. Eventually RED drops a packet at random from a queue, before the queue is actually full. When a packet is dropped, the TCP flow the packet belongs to backs off. This packet drop exploits TCP's response to a lost packet to prevent congestion before it occurs.

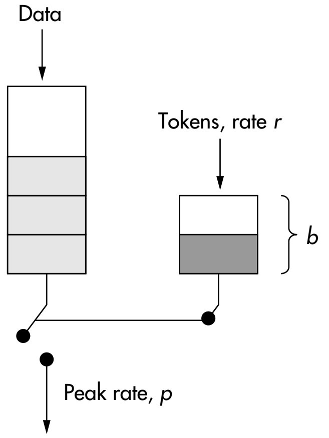

Figure 6. The token bucket. The token bucket specification describes the traffic pattern from a source but can also meter function in routers.

What about real-time applications that do not use TCP? Work is in progress to make real-time applications TCP-like (see http://www.psc.edu/networking/tcp_friendly.html), but they rarely have the kind of timely feedback an ACKnowledgment system offers. An alternative is explicit congestion notification (ECN) in IP-packet headers. The scheme would see routers set one- or two-bit flags in IP-packet headers when those packets pass through a congested part of the network. Of course, there is still the problem that the direction of such a signal is wrong (received by the destination and not by the source), but it is hoped that applications will adapt their signaling to report congestion to the source.

Another problem for applications that use ECN is that all packets passing through the congested part of the network—not just the misbehaving flows—will see the ECN signal. To improve on this behavior, some per-flow state would be required, though this might be undesirable.

OTHER ISSUES

At least two other major issues concern support for real-time applications and integrated services:

- modifications to routing for QoS and for multiparty communication, and
- the extra functionality required at the transport and application levels.

IP- and QoS-Related RFC Documents

Automatic flow-control and congestion control
M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," RFC 2581, Apr. 1999; http://www.ietf.org/rfc/rfc2581.txt.

Explicit congestion notification in IP headers
K. Ramakrishnan and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP," RFC 2481, Jan. 1999; http://www.ietf.org/rfc/rfc2481.txt.

IETF DiffServ Working Group
M. Carlson et al., "An Architecture for Differentiated Services," RFC 2475, Dec. 1998; http://www.ietf.org/rfc/rfc2475.txt.

Internet protocol architecture
B. Carpenter, "Architectural Principles of the Internet," RFC 1958, June 1996; http://www.ietf.org/rfc/rfc1958.txt.

Open Shortest Path First
J. Moy, "OSPF Version 2," RFC 2328, Apr. 1998; http://www.ietf.org/rfc/rfc2328.txt.

QoS-based routing
E. Crawley et al., "A Framework for QoS-based Routing in the Internet," RFC 2386, Aug. 1998; http://www.ietf.org/rfc/rfc2386.txt.

Real-time transport protocol
H. Schulzrinne et al., " RTP: A Transport Protocol for Real-Time Applications," RFC 1889, Jan. 1996; http://www.ietf.org/rfc/rfc1889.txt.

Resource reservation protocol
R. Braden, ed., L. Zhang et al., "Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification," RFC 2205, Sept. 1997; http://www.ietf.org/rfc/rfc2205.txt.

RSVP and the IETF Intserv Working Group
J. Wroclawski, "The Use of RSVP with IETF Integrated Services," RFC 2210, Sept. 1997; http://www.ietf.org/rfc/rfc2210.txt.

RSVP and scaling problems
A. Mankin, ed., F. Baker et al., "Resource ReSerVation Protocol (RSVP) — Version 1 Applicability Statement: Some Guidelines on Deployment," RFC 2208, Sept. 1997; http://www.ietf.org/rfc/rfc2208.txt.

Routing information protocol
G. Malkin, "RIP Version 2," RFC 2453, Nov. 1998; http://www.ietf.org/rfc/rfc2453.txt.

QoS and Multicast Routing

There is some interest in using multiple routing metrics such as delay and throughput, rather than just the single, unitless (policy-specific) number used on most networks today. Multiple metrics will cause routing protocols to become more complex, and routing updates become bigger, requiring more processing. Also, recall that at the level of backbone interconnection, the individual networks—autonomous-system networks—do not exchange routing metrics but exchange "reachability" information. Providing QoS-based routing across the wide area, therefore, becomes difficult to architect. Crawley et al.[11] provide an overview of QoS-based routing for IP-based networks.

The efficient transmission of multicast information to allow multipoint communication has yet to be widely accepted in the commercial sector. Currently, an experimental multicast backbone—the Mbone—is all that exists for Internet-wide multicast. See Diot et al. for a survey of multicast routing protocols.[12]

Integrating multicast provisioning with QoS routing is still a research issue. Most multipoint access solutions rely on mechanisms other than multicast to provide multipoint connectivity across the wide area, and such mechanisms are often not sufficiently scalable.

Transport- and Application-Level Signaling

To support real-time applications, the transport and application levels must provide new functionality. The real-time transport protocol (RTP) is the standard for real-time data transmission on an IP-based network. RTP provides no QoS capability but implements specific framing for real-time media, which can be treated as if it were a set of header fields extensions (such as sequence numbers and time stamps) to the user datagram protocol (UDP). RTP is allied with the real-time control protocol (RTCP), which provides feedback about the flow's performance (such as loss information and end-to-end delay). It does this via special report packets called sender reports—generated by a flow's source—and receiver reports—generated by the flow's destination host or hosts.

At the application level, there is a need for application-specific signaling. For example, there is currently much interest in providing VoIP with signaling capabilities to implement such features as call forwarding and busy signals. These are application-level protocols in IP networks but appear as network-level signaling in traditional telco networks.[13] For coordination of multiparty scenarios, such as teleconferencing, distributed floor-control mechanisms might be required.

CONCLUSIONS

A traditional IP-based network needs many modifications to support real-time applications and integrated services. This article focuses on only one area of QoS provisioning: modifying routers for packet handling such that networks can provide performance better than the current best-effort service.

Routers must recognize that many packets may be related and treat packets belonging to real-time flows with priority. This involves marking such packets so that they can be recognized, classifying the packets based on the markings so that they can be given the correct treatment, and allowing the scheduling mechanisms in the routers to transmit the packets in a timely fashion. The process requires a large overhead, involving mechanisms to perform the packet marking, classification, and scheduling. Therefore, before introducing QoS mechanisms into an IP-based network, routers may need to undergo hardware as well as software upgrades.

There are still many research issues concerning the development and deployment of technology for a QoS-enabled IP infrastructure. Many of these are technical issues concerned with the kind of things discussed in this article. However, an extremely important issue (not only for researchers but in general)—the resolution of which will be vital in deploying QoS mechanisms on a wider scale—is this: How do we charge for a QoS-enabled Internet? ■

REFERENCES

1. D.D. Clark, “The Design Philosophy of the DARPA Internet Protocols,” *Proc. ACM SIGComm 88*, ACM Press, New York, 1988, pp. 106-114.
2. D.D. Clark, S. Shenker, and L. Zhang, “Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism,” *Proc. ACM SIGComm 92*, ACM Press, New York, 1992, pp. 14-26.
3. S. Keshav, *An Engineering Approach to Computer Networking*, Addison Wesley Longman, Reading, Mass., 1997.
4. A.K. Parekh and R.G. Gallagher, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case,” *IEEE/ACM Trans. on Networking*, Vol. 1, No. 3, 1993, pp. 344-357.
5. A.K. Parekh and R.G. Gallagher, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case,” *IEEE/ACM Trans. on Networking*, Vol. 2, No. 2, 1994, pp. 137-150.
6. S.J. Golestani, “A Self-Clocked Fair Queuing Scheme for Broadband Application,” *Proc. IEEE Infocom 94*, IEEE Computer Society, Los Alamitos, Calif., 1994, pp. 636-646.
7. M. Shreedhar and G. Varghese, “Efficient Fair Queuing using Deficit Round Robin,” *ACM Computer Comm. Review*, Vol. 25, No. 4, 1995, pp. 231-242.
8. V. Srinivasan, S. Suri, and G. Varghese, “Packet Classification using Tuple-Space Search,” *Proc. SIGComm 99*, ACM Press, New York, pp. 135-146.
9. P. Gupta and N. McKeown, “Packet Classification on Multiple Fields,” *Proc. SIGComm 99*, ACM Press New York, pp. 147-160.
10. C. Metz, “RSVP: General-Purpose Signaling for IP,” *IEEE Internet Computing*, Vol. 3, No. 3, May/June 1999, pp. 95-99.
11. E. Crawley et al., “A Framework for QoS-based Routing in the Internet,” RFC 2386, Aug. 1998; http://www.ietf.org/rfc/rfc2386.txt.
12. C. Diot, W. Dabbous, and J. Crowcroft, “Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms,” *IEEE J. Selected Areas in Comm.*, Vol. 15, No. 3, Apr. 1997, pp. 277-290.
13. J. Crowcroft, S.N. Bhatti, and C. Perkins, “What is the Place for User-Network Signaling in the 21st Century?” *Proc. UKTS2000: 16th IEE UK Teletraffic Symp.*, Nortel Networks, Harlow, England, 2000.

Saleem N. Bhatti is a lecturer in data communications and networking in the Dept. of Computer Science, University College London. He has a BEng (Hons) in electrical and electronic engineering, an MSc in data communication networks and distributed systems, and a PhD in computer science, all from UCL. His work has involved aspects of multiservice networking, teleworking, multicast, network and systems management, network security, and most recently, the consideration of quality-of-service adaptability support for Internet applications. He is a member of the IEEE and ACM.

Jon Crowcroft is a professor of networked systems in the Department of Computer Science, University College London. He has a bachelor’s in physics from Trinity College, Cambridge University; and an MSc and PhD, both in computing, from UCL. He is a member of the ACM, a senior member of the IEEE, and a Fellow of the British Computer Society, the IEE, and the Royal Academy of Engineering. He is a member of the IAB and is on the editorial team for the *ACM/IEEE Trans. on Networks and Computer Communications.*

Readers can contact the authors at Dept. of Computer Science, University College London, Gower Street, London WC1E 6BT, UK; e-mail: {s.bhatti, j.crowcroft}@cs.ucl.ac.uk.