Quality of Service

By Paul Krzyzanowski
September 29, 2012

Introduction

Four factors affect network performance:

1. Bandwidth. Bandwidth, also known as *bit rate*, refers to a network's ability to move a volume of data over a unit of time. It measures the *throughput* of a network and is typically expressed in bits per second.
2. Latency. Latency, also known as *delay*, is a measure of the time it takes to traverse the network, including going through any intermediate routers. It represents the elapsed time between a sending node sending a packet and receiving node receiving that packet. For example, on my system, a local loopback (sending a packet back to the same machine) gives me a latency of 0.043 ms (milliseconds). The time to send a packet to a remote system on my LAN incurs a latency of 0.18 ms, sending a packet to Rutgers (about 30 km or 19 miles straight line distance) takes 11.6 ms, and sending a packet to IIT in Bangalore, India (about 13400 km or 8300 miles) takes 140 ms.
3. Jitter. Jitter represents the variation in packet latency, and is sometimes called *packet delay variation*. For example, in sending 100 packets to a server at IIT in Bangalore, India, I see delays ranging from a low of 139.6 ms to a high of 190.0 ms, producing a variation of 50.4 ms between the fastest and slowest packet latency. RFC 3393 defines techniques for measuring jitter.
4. Packet loss. Packet loss represents the percentage of packets that do not make it to their destination or those that arrive with errors and are therefore useless. It is usually a result of network congestion due to insufficient bandwidth at some point in the network.

Each of these factors impacts the quality of service that the network presents to applications.

- Low bandwidth impacts our ability to deliver high-bandwidth content such as streaming high-definition video and slows down file transfers. A Blu-ray disk streams H.264 compressed HD video at around 25-25 megabits per second. A 4 MB music file can be downloaded in under a 1 second over a 50 Mbps end-to-end link versus 16 seconds over a 2 Mbps link. The 665.48 MB update of Apple's OS X Mountain Lion v10.8.2 would have been an impractical download in the early 1990s since it would take over 27 hours to download via a 56 kbps modem.
- Latency affects interactive performance of applications. The recommendation for voice over IP (VoIP) service is a latency of under 150 ms (ITU standard G.114). Echo cancelation is required when latency exceeds 25 ms. Average jitter should be under 30 ms and packet loss under 1 percent.
- Jitter is the case of variable latency and we will have to deal with packets arriving at a non-constant rate. As we will see later with the leaky bucket algorithm, we may need to set up a *jitter buffer* to buffer up packets and allow the application to process them at constant rates. Unfortunately, this effectively increases the end-to-end latency. The size of the buffer can also lead to lost packets due to overflow (buffer overruns) and excessive jitter can lead to buffer underruns (empty buffer), both impacting the quality of service.
- Packet loss results in glitches in streaming media (voice or video). For other data transfers, it creates a need to retransmit lost packets to achieve reliable transfers, effectively increasing jitter tremendously (because of the time to detect a lost packet and get a new one) and possibly resulting in out-of-order packet delivery.

Our goal in addressing quality of service is to come up with ways in which we can provide applications with the network data flow performance that they need. These performance requirements place constraints on any or all of bandwidth, delay, jitter, and packet loss. For example, VoIP service does not require particularly high bandwidth but requires low latency (under 150 ms) and low jitter (under 30 ms on average). Streaming video may require high bandwidth but latency and jitter do not matter as much since we can buffer up a fair amount of video and interactive performance does not matter – it is usually not a problem if you are watching content several seconds after it has been sent. For applications such as file transfer or accessing a web page, neither latency nor jitter matter much but high reliability (low loss) is important.

Why is this hard?

The reason that we have problems with quality of service is because we do not have unlimited resources. On the computer, we may have multiple processes all of which may have packets to transmit over a single network interface; they have to take turns. On the network we have only so much bandwidth available. Worse, we will often have a mismatch of bandwidth as we route from a fast network to a slow one. Within the network, the packet will often flow through multiple switches and routers. Each one of these introduces processing delays (typically on the order of 11 microseconds for high-end routers but a lot more for congested lines). Packets arrive at one port, are buffered in memory, have their headers analyzed, and get placed on queue for a specific outbound port on the device. Because of processing delays and the speed at which packets can be sent on an outbound port, packets need to be queued within the router. This requires memory to buffer the packets. If packets are received faster than they can be sent, queues will overflow and packets will get dropped. This situation is known as congestion. We will have to control quality of service by a combination of controlling resource allocation and prioritizing packets.

Let us examine some of the contributors to network degradation.

Contributors to congestion

The two main contributors to congestion within a router or switch are bandwidth mismatch and aggregation. Bandwidth mismatch is the problem of routing from a high-speed network to a low-speed network. If packets are arriving on a 10 Gbps network and need to be routed to a 1 Gbps network, there just isn't enough network capacity: the incoming network has ten times the bandwidth of the outgoing network. The router (or switch) will queue the outgoing packets but the queue will overflow and packets will have to be dropped.
Packet aggregation is a similar problem. Here, instead of having a high-bandwidth incoming link, we have several connections that are all sending packets that are destined to a single outbound link. Three streams of packets coming from three separate 1 Gbps Ethernet links with all packets destined to a single 1 Gbps link will quickly fill up the router's buffer and again result in packet loss. Finally, there is also the potential for congestion because of additional unintended traffic on the network. This traffic can result from faulty hardware, extra retransmission request messages due to high levels of packet loss, or misbehaving (or malicious) applications generating high amounts of network traffic.

Contributors to latency

Latency is accumulated throughout the entire flow of a packet. At the host, there is latency caused by the scheduling of the process that generates the data followed by packetization, the computation necessary to create packets and move them through the layers of the network stack, which typically involves allocating socket buffers, generating TCP or UDP headers (including checksums), creating IP packets, and creating Ethernet MAC packets. After that comes the delay of serialization, which is due to the fact that we can send only one packet out at a time. If there are multiple packets ready to go out, they will be queued behind the packet that is currently being sent onto the network.

Once a packet is on the network, there is the latency caused by propagation; the network carries data at a specific bit rate. Each time a packet hits a router or a switch, it encounters a processing delay on the router since data has to be moved between an input queue and an output queue. In the course of doing that, the packet needs to be inspected (for example, to determine the route). Similar to serialization on the host, the packet may encounter a queuing delay on the router, where it has to wait in a queue until the outgoing interface is available since it may be in use by other packets.

Contributors to jitter

The main reason why packets don't arrive at a constant rate is that computers, routers, and switches can only send one packet out at a time on any single link. Other packets have to be queued. If a host has multiple packets to send, they will be queued. When your application sends a packet, it may sometimes end up first in the queue (short delay) or last in a long queue (long delay). If other packets to the application do not suffer the same delay, the result is jitter: variable latency. This same process is repeated on each switch and router the packet flows through on its way to the destination.

In the past, when ethernet was a true shared medium, and multiple hosts sent a packet out at approximately the same time, the result was a collision and both packets would need to be retransmitted at a later time. With switched ethernet, packets get buffered in the switch but if they are destined for the same port, one will be sent before the other, causing lower verses higher jitter, respectively. The same thing happens in routers. Packets are stored in memory buffers that form queues. The packets then are transmitted from those queues one at a time for each interface. The further back a packet is in the queue the packet ends up, the greater a delay it experiences.

Since protocols such as IP may dynamically reroute a stream of packets, it is possible that one packet may not take the same route as another. It is unlikely that two routes will result in the same end-to-end delay. Hence, the endpoint will experience jitter.

Finally, if we rely on the transport layer to provide the illusion of reliable, in-order packet delivery, as TCP/IP does, that layer of software can contribute to jitter in two ways. First, to ensure that packets are delivered to an application in the order that they were sent requires holding back any packets that arrive out of order. These will be delivered only after the earlier packet was received. Any hold-back adds to the end-to-end latency and hence increases jitter. Second, if a packet was lost or corrupt, it will need to be retransmitted by the sender. This introduces the additional delay of determining that a packet is missing (TCP uses a retransmission timer at the source to detect a missing acknowledgement from the receiver after a certain amount of elapsed time) and transmitting that packet again. These delays tend to be far greater than any delays caused by packet dequeuing at routers and can greatly increase the variance in packet delivery time (jitter).

Contributors to packet loss

The primary reason packets get lost is because they get discarded due to a buffer overrun (queue overflow) condition in routers and switches. When a queue is full and there is no memory to hold new packets, they end up being discarded. Additional factors for packet loss include bad transmission lines, where signal degradation or interference can cause packet loss as well as faulty hardware. On wireless networks, interference and collision can be significant contributors to packet loss.

QoS-aware networks

If a network is QoS-aware, that means it allows an application to specify its quality needs (e.g., bandwidth, delay, jitter, loss) and ensure that it reserves sufficient capacity in all switching elements between the source and the destination to meet these needs. To do this, an application needs to initiate a QoS request to the network and get permission, a go/no-go decision, from the network. If the network grants access then the application has assurance from the network that the application's quality of service requirements will be met. This technique where quality requests are issued by applications and granted by networks is called admission control. Within the routers of a network, traffic control is the set of algorithms responsible for classifying, queuing, and scheduling data traffic, often prioritizing one packet over another.

There are three grades, or service models, for networks to handle QoS:

1. No QoS. This is sometimes called a best effort network and is the default behavior for IP routers with no QoS. While routers do the best job they can in getting packets through, there is no preferential treatment of one packet over another and the host is not involved in any way in specifying the quality of service. The traditional approach for achieving desired service quality in best effort packet-switched networks was through over-engineering; ensuring that the network has more than enough capacity.

2. Soft QoS. This is also known as differentiated services. Here, there is no admission control performed by the network and hence no explicit setup of a quality-controlled data stream. Instead, routers identify flows of related packets and give certain flows preferential treatment over others. QoS information may be embedded into packets and routers are configured with rules on how to prioritize one type of packet over another. Because there is no admission control, the network cannot prevent congestion or lengthy queues, so there is the possibility that packets can still be delayed or lost.

3. Hard QoS. This level of service provides a guarantee by the network to deliver the specified quality of service for a particular end-to-end data flow. To achieve this, admission control is required so that each router can commit to reserving sufficient resources for moving the traffic for each flow.

Since failure to deliver a specific quality of service is due to not having unlimited networking resources, the key to ensuring that a network can provide desired levels of service is to allocate network resources appropriately. We refer to a flow as a stream of packets that represents a communication stream between a pair of hosts and goes through a specific set of routers. In TCP/IP and UDP/IP, a flow is typically the set of packets going from one address and port to another address and port, all with the same protocol (e.g., TCP or UDP). To allocate resources, we can take a *router-based* or *host-based* approach.

With a router-based approach, each router is responsible for deciding how to prioritize its traffic. With a host-based approach, each host observes conditions on its network and adjusts its behavior appropriately (for example, slowing down its rate of packet transmission). Data communication can be either *reservation-based* or *feedback-based*. With reservation-based communication, we enforce admission control. Each host asks for a particular grade of service from the network (bandwidth, delay, jitter, and possibly packet loss). This request is forwarded through all routers along the path that the packet needs to traverse to get to its destination. Each router will allocate sufficient buffer space for that flow and prioritize it accordingly to satisfy the service request. If any router cannot grant the desired level of service, the reservation is denied and the host will have to either give up, wait and try again, or try to make a request for a lesser grade of service (e.g., transmitting video at a lower bandwidth). With feedback-based communication, data can be sent onto the network with no *a priori* reservation. However, the host will either get congestion feedback from a router or will detect an increased packet loss rate and adjust its transmission rate.

Controlling congestion

Our primary goal is to find ways of avoiding congestion, which is caused by more incoming traffic than we can route onto an outgoing network interface. There are three approaches for dealing with congestion: admission control deals with the input of packets and disallowing the input of packets for specific flows and new flows; choke packets provide a feedback mechanism to slow down transmission; and queue management is a set of techniques for prioritizing traffic that gets sent out of a router.

A choke packet is a special packet that is sent back to the originator when a router determines congestion on a link. It tells the originator that a port is congested and requests that the host not send any more packets to that destination. This is strictly voluntary and the host may choose to ignore this. However, IP handles congestion by simply dropping packets once there is no more room in a router's buffer, so the host is also being warned to expect packet loss if it continues transmitting data. RFC 3168 describes the addition of Explicit Congestion Notification (ECN) to IP.

Controlling jitter

When multiple packets arrive at a router, possibly at the same time on different ports, and are all targeted to the same port for output, they get placed on a queue. A lucky packet will be first in the queue and an unlucky packet will be last and hence most-delayed since it has to wait for all earlier packets to get sent out first. The variability in one's position in the queue causes jitter.

Since only one packet can be sent out on any given port at a time, we are bound to have packets queued up for transmission if packets arrive faster than they are routed. However, instead of using a single queue, we can set up a set of output queues, one per class of service. Every packet will be assigned to a specific queue based on some data in the packet header. The router will schedule transmission of packets from high-priority queues prior to transmitting from low-priority queues.

The actual packet header data that is used for determining packet classes can vary and can usually be configured when setting up the router. Typical items to determine the priority of a packet include 802.1p priority bits and the MPLS EXP field at layer 2 of the OSI stack; IP ToS bits (DiffServ – we will cover this a little later), protocol type (e.g., TCP, UDP), source and destination addresses at layer 3; and port numbers at layer 4. Any combinations of these can be used.

Now we need a scheduling algorithm to determine how to select a packet from a queue. The absolutely simplest scheduler is to use a single queue and extract packets in a FIFO (first in, first out) manner. That does not allow us to distinguish flows. A round robin scheduler will cycle through the queues, taking a packet from one queue, then the next queue, and so on. This will give each queue, and hence each class of service, a fair share. That is not what we want either since certain flows are deemed more important than others. A priority scheduler will prioritize queues by level of importance and take a packet from the highest priority queue that has one. However, if there are always packets in high priority queues, we get starvation in the form of head of line blocking, where packets are holding up other packets. A more sophisticated approach is to give each queue a priority as well as a minimum percentage of the link speed (the outgoing port bandwidth). This will ensure that even low priority queues will get scheduled for output even if they do less frequently than high priority queues. This technique is called weighted fair queuing (WFQ).

In some cases, routers try to detect flows automatically and maintain statistics on each flow (e.g., the route, average packet size, and average packet arrival frequency). This makes it easier for a router to allocate resources, cache routes, and prioritize queues. There is no external signaling information that is used to create this data, and this collection of data about a flow is known as a soft state.

Once a router has determined a flow, it can control the packet rate of a specific flow by dropping or delaying packets related to that flow. It can also make decisions on which packets to drop in cases of congestion. Administrative policies can guide the prioritization of these decisions. For example, one may select that TCP packets get dropped over UDP packets since lost TCP packets will be retransmitted, or perhaps UDP packets for some non-essential services be discarded to ensure a higher level of service for other flows.

Routers typically offer administrators two options for weighted fair queuing: flow-based and class-based. Flow-based weighted fair queuing relies on a router's discovery of flows. Flows that are deemed to be interactive, low-bandwidth flows are given high priority. Non-interactive, high-bandwidth flows get their priority evenly divided among the flows of that type. Class-based weighted fair queuing allows an administrator to define classes based on network protocols, input interfaces, and specific source and/or destination addresses and ports. Each class can be assigned a bandwidth, queue limit, and weight (priority). This allows an administrator to do things such as prioritize ssh traffic over ftp traffic or traffic to one machine over another.

Bandwidth management

There are two approaches to managing bandwidth. One is traffic shaping, whose goal is to regulate the average rate of data transmission for each flow. This is often a problem of controlling jitter and works by queueing packets during surges and releasing them later. An example is routing bursty traffic from a high bandwidth link onto a low bandwidth link. The other approach is traffic policing. Here, the goal is to monitor network traffic per flow and discard traffic that exceeds allowable bandwidth.

The classic approach to traffic shaping is the leaky bucket algorithm. The visualization is a bucket with a hole at the bottom. It is filled with water at a varying rate but water leaks out of the hole at a constant rate. Incidentally, this is the basis for hourglasses and some types of water clocks. The bucket is the packet buffer (packet queue). If a packet comes in and there is no more room, the packet is discarded. This is called buffer overrun and the visualization is that you're adding water to a bucket that is already full, so it just spills out over the rim. If a packet comes in and there is nothing to transmit because the buffer (bucket) is empty, we have a buffer underrun. As long as we don't experience buffer overrun (our buffer is sufficiently large) or buffer underrun (we don't have long periods with no incoming data), the leaky bucket will convert an uneven flow of packets into an even one. Jitter is eliminated (or greatly reduced) at the expense of additional latency (packets sitting in the buffer).

A similar-sounding counterpart to the leaky bucket algorithm is the token bucket. Here, instead of holding the data itself, the bucket holds tokens that are generated at a fixed rate. To transmit a packet, the bucket must have a token. That token is removed from the bucket and discarded as the packet is transmitted. The token bucket serves to save up permissions to allow us to send large bursts later. The bucket sizs determines the maximum allowable burstiness.

Here's how the algorithm works. The "bucket" containing tokens is really just a counter. The count represents the number of tokens and each token represents the permission to send one byte (tokens don't really exist; we're just interested in counting them). If our desired *average* bandwidth is *r* bytes per second, we will add a token every 1/_r_ seconds. If the token count is greater than the size of the bucket then discard the token. When a packet of *n* bytes arrives, we look at the number of tokens in the bucket. If the number of tokens is greater than or equal to *n* then remove *n* tokens and transmit the packet. If, however, there isn't a sufficient amount of tokens (less than *n*), we have two options. If we want to perform *traffic shaping*, we queue the packet until enough tokens arrive to allow us to send it. If we want to perform *traffic policing*, we simply drop the packet.

Both algorithms regulate bandwidth. The leaky bucket algorithm focuses on traffic shaping and turns a bursty stream into a smooth (jitter-free) stream. The token bucket algorithm focuses on guaranteeing the delivery of average bandwidth with no concern as to whether the generated traffic stream is bursty or not.

Quality of service in IP

The Internet Protocol (IP) was not designed with any Quality of Service controls in mind. IP normally cannot take advantage of any QoS controls that may be offered by an underlying network since it was designed to work with any packet-switched network and make no assumptions beyond that the underlying network offers unreliable datagram packet delivery. Over time, some QoS mechanisms were created as add-ons to the core protocols of IP. There are four core issues that affect the quality of service in IP networks:

1. Bandwidth mismatch and aggregation. As always, routing from a high-bandwidth link to a low-bandwidth one or aggregating traffic from multiple links can lead to packet congestion.

2. Inefficient packet transmission. If we want to send a one-byte packet via TCP (e.g., a single character), we incur an overhead of 58 additional bytes for the various headers (20 bytes for the TCP header, 20 bytes for the IP header, 18 bytes for an ethernet MAC header). This is not a QoS issue but does impact network bandwidth, packet serialization from the host, and packet queuing and scheduling on routers. Big packets are more efficient to transmit than small ones.

3. Unreliable delivery. IP was designed to run on unreliable datagram networks. A packet is not guaranteed to arrive at its destination. TCP was created as a transport-layer protocol to provide software-based reliability through retransmissions. If a sender fails to receive an acknowledgement to a sent TCP packet within a certain window of time, that packet is considered to be lost and is retransmitted. This provides reliability but leads to high jitter.

4. Unpredictable packet delivery. IP offers no controls on bandwidth, delay, or jitter. Moreover, packets may take different routes to their destination, resulting in changing service levels.

Dealing with inefficient packets

Some software generates lots of small packets (e.g., a lot of single-byte transmissions), which creates a glut of packets that need to be routed and can cause head of line blocking in routers, excess congestion, and excess jitter. Nagle's algorithm is a tweak to TCP that holds off on transmitting a TCP packet if an acknowledgement for a previous packet did not yet arrive (subject to a time-out, of course). Any new data written by the application is added onto the existing packet instead of being placed into new packets. Most systems incorporate Nagel's algorithm into their TCP stack.

Other approaches to make packets more "efficient" are to compress headers (see RFC 3843) and to compress the packet payload (RFC 2393). The problem with these approaches is that both sides have to participate and, in the case of header compression, all involved routers have to be aware of this. Also, there's a tradeoff betwween the CPU time spent on compression and decompression versus the netwrok time spent on moving the packets. In practice, these approaches are rarely, if ever, used.

Differentiated Services (soft QoS)

Differentiated services, often called DiffServ, is a technique to allow a host to identify a class of service for each packet. A router will then be able to use this data to make decisions on scheduling or dropping the packet. This is a soft QoS approach since it provides parameters for classifying packets but does not restrict the host from sending the packets as an admission-based protocol would.

An IP packet contains a one-byte field (bits 9 through 16 in the header) that used to be known as ToS, for Type of Service. This field is obsolete and the first six bits have been repurposed for a Differentiated Services Field (see RFC 2474), which can be used by an application to identify the service class of the packet. The number in this field is called the *Differentiated Services Code Point* (DSCP) value. A range of values have been defined in a number

of other RFCs (see here) to identify a number per-hop behaviors for varying types of traffic. The categories include default, expedited forwarding (low delay, low loss, low jitter, suitable for voice and video), voice admit (dedicated for voice traffic), assured forwarding (high assurance of delivery), and class selector (where the top three bits define a priority value).

DiffServ is used extensively in configuring class-based WFQ scheduling on routers but it is advisory in nature. Being a soft QoS mechanism, it cannot prevent congestion or ensure low jitter. While DiffServ is used extensively within ISPs, particularly in differentiating voice over IP (VoIP) telephony, video, and normal data services, its use on Internet across ISPs is limited due to lack of peering agreements. There is little or no assurance that ISPs will use the same DS field values for the same types of service.

Integrated Services (hard QoS)

While DiffServ provided a soft QoS approach, Integrated Services (IntServ, RFC 1633) provides a mechanism for end-to-end reservation of service levels and routing resources. IntServ comprises two parts: a Traffic Specification (TPSEC), which defines the token bucket size, specifying the bit rate and buffer size, and a Request Specification (RPSEC), which specifies the level of assurance. A level of assurance is one of three classes of service: best effort (no reservation needed), controlled load (a form of soft QoS where data rates are allowed to increase and packet loss may occur), and a guaranteed level that placed tight bounds on bandwidth and end-to-end delay. The underlying mechanism used to implement IntServ is the Resource Reservation Protocol (RSVP).

The Resource Reservation Protocol, RSVP (RFC 2205), requires a host to specify a desired quality of service for each flow of packets. The router reserves necessary resources and sets up its packet scheduler to handle this traffic. The request is then forwarded to other routers in the route so that all routers in the path will be prepared to handle the traffic.

ATM Networks

One of the pitfalls of IP networking is that it was designed with no mechanisms for an application to specify how the traffic that it generates is to be scheduled over the network. This causes problems in continuous media applications such as video and voice. In these applications, excessive delays in packet delivery can produce unacceptable results. IP version 6 attempts to alleviate this somewhat by allowing applications to tag packets with a priority level, but this does not translate directly to bits per second, jitter constraints, or delay requirements. A different approach to networking emerged in the late 1980s and was adopted as an international standard. This form is known as ATM, or Asynchronous Transfer Mode, networking. Its goal is to merge voice and data networking. The former is characterized by a low, but constant bandwidth. The later tends to be bursty in bandwidth requirements (0 one minute, 100 Mbps the next). Circuit switching is too costly for data networking since it is a waste of resources to allocate a fixed-bandwidth circuit to bursty traffic. IP-style packet switching, on the other hand, is not well-suited for the constant bandwidth requirements for voice telephony.

ATM attacks the problem by using fixed-size packets over virtual circuits. A sender establishes a connection, specifying the bandwidth requirements and traffic type. Traffic type may be constant bandwidth rate (CBR), variable bandwidth with bounded delay (VBR), or available bandwidth (ABR) . If the connection is accepted (admission control), a route is determined, routing information is stored in the switches within the network, and each router (ATM switch) that is involved in routing the traffic commits to allocating the necessary resources for the level of service that was requested. All traffic is carried in fixed-size cells. Fixed size cells provide for predictable scheduling (a large packet is not going to hold up smaller ones behind it) and rapid switching.

An ATM cell is 53 bytes: 48 bytes for data and 5 bytes for the header. Because of the relatively small packet size, a data stream will generally have to deal with a large number of them. For example, a saturated 622 Mbps link will carry 12 million ATM cells per second. If a device were to receive each of these cells, that would translate to 12 million interrupts per second! To avoid congesting a computer with millions of interrupts per second (one interrupt for each incoming packet), ATM hardware supports the splitting of larger chunks of data into multiple ATM cells and assembling incoming ATM cells into larger packets. This is called an ATM adaptation layer (AAL). A few adaptation layers exist for handling different traffic types (e.g. AAL 1 for constant bit rate traffic, AAL 2 for variable bit rate traffic, etc.). Perhaps the most popular for data networking is AAL 5. Outbound data is fragmented into multiple ATM cells with a bit set in a field in the last cell to indicate an end of packet. The destination, accepting the packet, simply assembles the cells coming in on that circuit until it gets a cell with an end of packet bit set. At that point, it can deliver the full data packet up to the upper layers of the system software. Compatibility with IP (useful for legacy applications) can be achieved by running the IP protocol over the ATM layer, segmenting and reassembling each IP packet into ATM cells.

While ATM solves a number of problems present in IP/Ethernet networks, its switches and interface cards never reached the price and mass market acceptence of Ethernet. Meanwhile, IP networks kept getting faster and the over-engineered network, along with soft QoS controls, delayed the need for precise cell-level scheduling.

References

- Karie Gonia, Latency and QoS for Voice over IP, SANS Institute InfoSec Reading Room, version 2.4b Option 1, © 2004 SANS Institute.

- Michael Patterson, ToS, DSCP and NetFlow.... what the DiffServ?. NetFlow, July 2009.

- Cisco, Understanding Delay in Packet Voice Networks, July 31, 2008.

- Cisco, CCNP Self-Study: Understanding and Implementing Quality of Service in Cisco Multilayer Switched Networks. Cisco Press, May 13, 2004.

- Cisco, Class-Based Weighted Fair Queueing. Cisco IOS Software Releases 12.0 T.

- Tim Szigeti, Christina Hattingh, Quality of Service Design Overview, Cisco Press. December 17, 2004.

- K. Ramakrishnan, S. Floyd, D. Black, RFC 3168, The Addition of Explicit Congestion Notification (ECN) to IP. September 2001

- Chris Bryant, Flow-Based Weighted Fair Queueing. A Cisco Router Tutorial, thebryantadvantage.com.

Patrik Carlsson, Doru Constantinescu, Adrian Popescu, Markus Fiedler and Arne A. Nilsson, Delay Performance in IP Routers. Dept. of Telecommunication Systems School of Engineering Blekinge Institute of Technology 371 79 Karlskrona, Sweden,

- Sudakshina Kundu, Fundamentals of Computer Networks. Second Edition, Prentice Hall of India Learning, 2008