White Paper

Bringing Comprehensive Quality of Service Capabilities to Next-Generation Networks

By: Dr. Syed Ijlal Ali Shah, TM Product Line Manager for Motorola's C-Port Family of Network Processors

Quality of Service (QoS) can generally be defined as a mechanism for networks to satisfy the varied quality and grade of service required by an application, while at the same time maximizing bandwidth utilization. Traditional circuit switched networks have been optimal for supporting applications that generate traffic at a fixed rate and require low end-to-end delay and jitter. However, applications with varied traffic rates and patterns waste bandwidth on these networks because of their more bursty nature. Traditional data networks generally provide best effort services, which are fine for packet switching, but they do not support high quality transmission of video and voice.

At the edge and access areas of the network, these technologies are converging in devices such as multiservice switches, cable head-ends, voice gateways, and so on, creating a world where bandwidth must be fully utilized (for business to be cost-effective), and where the services provided must enforce the appropriate level of transmission quality for the various types of applications, whether they are Voice over IP (VoIP), streaming video, Web casting, or simple data transfers. Here more than anywhere, implementation of comprehensive QoS management is critical.

Many technologies have been developed to aid in the deployment of QoS across these next-generation networks, such as Asynchronous Transfer Mode (ATM), Internet Protocol Differentiated Services (IP DiffServ), and Multiprotocol Label Switching (MPLS). The objectives of all of these technologies is to provide sufficient QoS differentiation, ease the task of managing and provisioning QoS, while at the same time maximizing network utilization.

To aid in the deployment of these technologies, silicon vendors have been developing hard-wired coprocessors, called traffic management coprocessors (TMCs), that implement the complex algorithms and buffering mechanisms needed to support the QoS technologies. These coprocessors off-load the complexity of QoS implementation from the data path processing performed by a network processor (or NP chipset), allowing network equipment vendors to develop products that provide efficient and differentiated QoS solutions.

This paper focuses on the application requirements for QoS, the underlying principles of QoS management, the mechanics of supporting differentiated end-to-end QoS, and the requirements of TMCs to support QoS effectively in next-generation networks.

Table of Contents

MOTOROLA digital dna

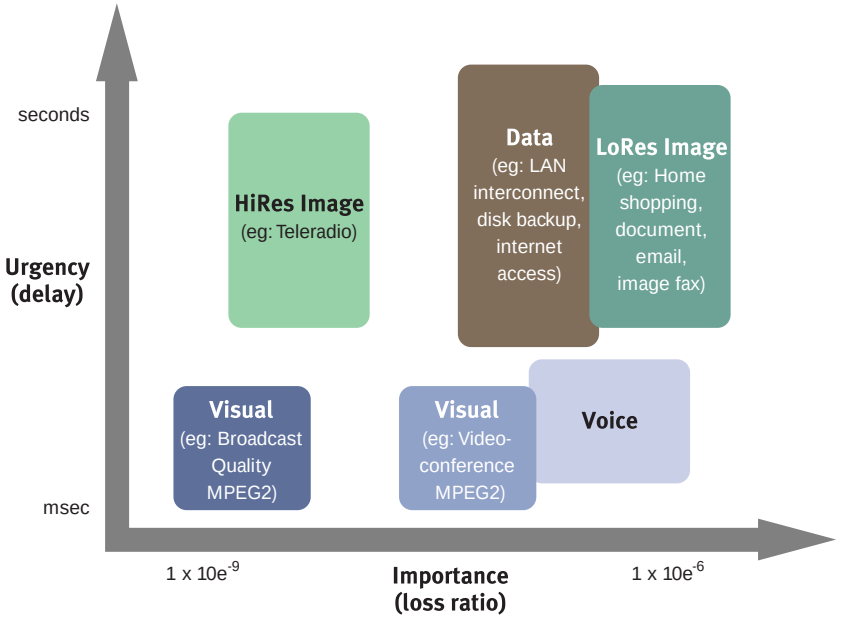
Varied QoS Requirements for Video, Voice, and Data

The applications running on access and edge networks can be divided into three primary groups, namely video, voice and data. These groups can be further sub-divided into interactive and non-interactive applications. The end-to-end QoS requirements, defined by loss of data, delay and delay jitter, of an application in general depends on the basic group it belongs to and whether the application is interactive or non-interactive.

Interactive video has the most stringent QoS requirement, with a data loss ratio requirement of $1x10^{-9}$ cells/packets and a delay requirement of 500 microseconds per switching node. Whereas voice applications in general are more tolerant to loss, but less tolerant to delay with a loss ratio requirement of $1x10^{-6}$ and delay requirement of 500 microseconds per switching node. Studies have also shown voice to be tolerant to delays of as large as 150 microseconds without causing any significant degradation in conversational dynamics. As a contrast, data applications are much more tolerant to loss and delay when compared to voice and video applications [12].

Figure 1 shows the relative position of some of the applications in regards to loss and delay requirements. Not only do the applications have different QoS requirements they also generate traffic that exhibits different characteristics that depend on the nature of the applications and the mode of transaction. For example, voice applications generally transmit at a fixed data rate; however, if packetized and coupled with silence suppression, they become time-varying with bursts of activity followed by idle periods. Other applications such as video, video conferencing, and data transactions (FTP, email, WEB browsing, and so on) also generate data at a variable rate with high activity periods followed by periods of low activity.

Figure 1 Application Loss and Delay Requirements



How ATM, MPLS and IP Diffserv Solve the Problem

In order, for a single technology or protocol, to support voice, video, or data applications, multiple qualities of service have to be offered along with the ability to handle different traffic profiles associated with these applications. Three technologies that have been the focus of studies by the telecommunications industry for the past few years, and that have emerged as key solutions to QoS implementation issues are ATM, IP DiffServ, and MPLS.

ATM QoS Classes

Supporting QoS in the network is a complex task. The complexity of supporting more than a few QoS classes in the network increases beyond realm of practicality [15], [20]. It is because of this reason that applications are classified into groups and these groups are carried across the network over a finite set of QoS classes.

The ATM forum defines six QoS classes [18]. These classes offer different loss, delay and jitter guarantees to satisfy the wide range of QoS requirements of the applications. Applications, such as voice and interactive video that are delay and loss sensitive are supported by the Constant Bit Rate (CBR) service. Applications that are not as loss and delay sensitive are supported by Variable Bit Rate real-time (VBR-rt) and non real-time (VBR-nrt) class.

Typical data applications that generally do not require any fixed guaranteed QoS are supported by the Unspecified Bit Rate (UBR) service. Applications that require minimum bandwidth guarantees but do not require stringent delay and loss performance of either CBR or VBR classes, can use the Available Bit Rate (ABR) and Guaranteed Frame Rate (GFR) classes. In an ATM network, applications inform the network of the desired class of service during connection setup time. Once the connection is setup, class of service identification is implicitly done through the VPI/VCI address.

IP Diffserv

The Internet Engineering Task Force (IETF), realizing the need to support QoS differentiation in packet networks developed the DiffServ architecture [7]. DiffServ defines two types of QoS differentiations: Expedited Forwarding (EF) [9] and Assured Forwarding (AF) [8] and uses the ToS (Type of Service) bits in the IP header to differentiate packets at a DiffServ switching node.

Expedited Forwarding, much like the CBR service in ATM, is designed to provide a very low loss and low delay QoS to applications such as VoIP or interactive video. Assured Forwarding on the other hand is similar to ATM VBR service and is designed to provide low loss and low delay to applications; however it is not as stringent as the EF service. A third QoS, often added to DiffServ, is opportunistic in nature and does not guarantee any QoS. This service is called the Best Effort service and is similar to what is offered by packet switched networks.

MPLS Labels

MPLS is a technique where packets are assigned specific labels as they enter the MPLS network [10]. All subsequent treatment of cells/packets within the MPLS network is based on that label. MPLS was introduced as a way of improving the forwarding speed of routers, but now it has emerged as a crucial technology that facilitates scalability of IP networks. MPLS labels encapsulate IP packets as they enter the network. The MPLS protocol allows traffic engineering and provisioning for the support of QoS classes in the network through signalling protocols such as CR-LDP, RSVP, and so on [11]. In fact, any protocol can be encapsulated by MPLS labels. These labels are then used for routing and service differentiation.

Underlying QoS Principles

ATM, IP Diffserv, and MPLS all work on the same basic underlying QoS principles of resource provisioning and real-time controls that are required to provide comprehensive QoS management across the network.

Network Provisioning

The task of network engineering and dimensioning is to intelligently deploy just enough resources (switches, transmission links, and so on) in the network to satisfy the end-to-end QoS requirements of the applications. Determining how much network resources are sufficient is not an easy task and depends on the type of equipment being used, the QoS being offered, type of applications, average traffic load, and the growth of traffic within the network [16],[13],[14]. These resources, once determined, are then deployed in the network in the form of switches, routers, and transmission links.

However, predicting the future is very difficult in this rapidly changing communications landscape. If information about the behavior of applications, their growth rate, and traffic loads were known for all future times, it is conceivable to engineer and dimension a network where there will never be any congestion. In reality due to the variable behavior of applications, traffic growth rates and emergence of yet unknown services, it is impossible to engineer or dimension a network, without gross over provisioning, that will not get congested at a future time. Gross over provisioning is not a good option as it would result in higher service costs that may drive the overall revenue down.

The trick is to provision just enough and then rely on network-related controls (such as connection admission and routing policies) and then the finer-grained cell/packet level controls. These controls will enable a network to maintain the end-to-end QoS and respond in ways that do not degrade the QoS of connections, or if degradation does occur, it degrades in a graceful manner.

Network-related Controls

Two network-related elements can go a long way toward ensuring that appropriate resources are available on the path: connection admission control and routing. Connection admission control and routing prevent the network from getting congested by simply denying admission if resources are not available, or by searching for a less congested path for the connection. However, they are only useful in controlling congestion during connection setup time — not after the connection has been accepted.

Connection admission control is responsible for calculating the amount of resources a particular connection/call would require and in determining whether those resources are available at each hop of the path. The resources are determined based on the traffic characteristics of the connection and the desired QoS [17],[3]. For example, the resources required by connections having the same peak and average rates but different QoS requirements would be different. A connection is accepted if enough resources are available in the network and denied otherwise.

Routing is responsible for determining the optimum path for the call/connection through the network. Normally, if enough resources are not available on a particular path, then a few other options are tried based on the routing policy before giving up and denying admission to the connection.

A properly engineered circuit switched network coupled with appropriate connection admission control and routing will not get congested. This is because each connection in a circuit network is allocated a fixed amount of bandwidth with no statistical multiplexing inside the network. That is, the incoming rate to a switch/router never exceeds the outgoing rate. This lack of statistical multiplexing manifests itself in contention free transmission of data inside the network. On the other hand, a properly engineered cell/packet network with admission control and routing can still get congested due statistical multiplexing and the varied nature of cell/packet arrivals. A cell/packet network, therefore, has to resort to yet another level of control mechanisms that operate closer to media transfer rates and are commonly referred to as the Cell/Packet Level controls.

Cell/Packet Level Controls

While the connection admission control and routing generally reside in the control/management path of a switch/router, the cell/packet level controls operate at the data link layer and reside in the forwarding path of a switch/router, making their function more time-critical. There are several metrics that can be used to measure QoS experienced by an application, ranging from higher layer call metrics (call blocking, misrouted calls, billing, time to provision and so on) to physical layer metrics (bit errors, burst errors, and so on). However, the QoS in the forwarding path is generally measured in terms of cell/packet loss, mean delay, and delay jitter.

The cell/packet real time controls use different traffic management mechanisms to ensure that the applications receive the desired QoS in terms of loss, delay, and delay jitter. The building blocks of QoS control at the cell/packet level comprise of the following traffic management functions:

- Policing, monitoring and shaping of flows
- Policy-based buffering and Active Queue Management (AQM)
- Policy-based scheduling of flows

Policing, Monitoring and Shaping of Flows

Policing and monitoring of flows is an important component of traffic management and QoS control and is required to guard against flows that do not adhere to the contracted flow specifications. The main function of traffic policing is to determine whether the incoming traffic conforms to the traffic specifications agreed upon between the user and the service provider. If the traffic is non-conforming, appropriate action is taken based on the QoS policy.

One of the main causes of congestion in the network is bursty unpredictable traffic. If traffic on a particular flow or connection could be made to transmit at a uniform and predictable rate, congestion would be less common. Traffic shaping configures the outgoing traffic to the agreed upon traffic profile for the flow.

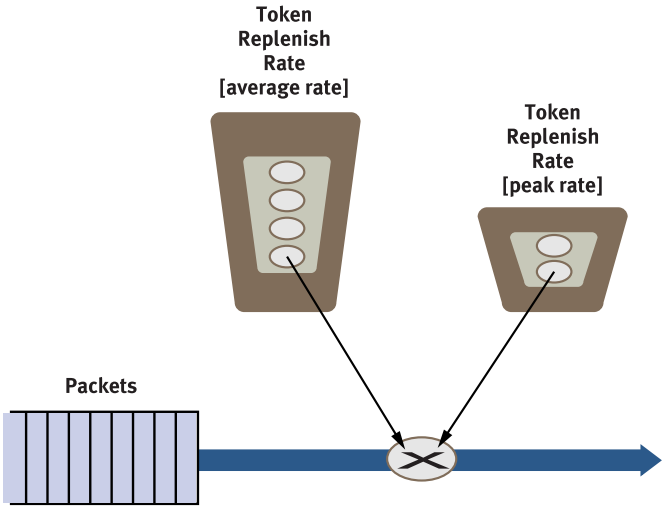
Most of the policing/shaping mechanisms used today are based on leaky bucket mechanism. To understand the leaky bucket mechanism, imagine a bucket with a small hole at the bottom. It does not matter, at what rate the water enters the bucket, it can only leak out at the rate determined by the hole at the bottom of the bucket. Thus a leaky bucket shapes the outflow of water to the rate determined by the hole.

The leaky bucket could be implemented by using a counter. The counter holds tokens where each token may represent a cell/packet or a certain number of bytes. Tokens are added to the counter at fixed intervals of time and decremented as the data flows through. If there are no tokens available or the tokens that are available do not cover the entire length of the data, the cell/packet would be buffered and not allowed to enter the network until sufficient tokens have accumulated. The number of tokens that can accumulate in the counter is generally referred to as the leaky bucket depth.

If the flow is idle, then tokens may accumulate to the extent determined by the QoS policy and the bucket depth. The amount of tokens accumulated represent the burst size that may be admitted into the network. By controlling the depth of the bucket, the network could regulate the permissible burst size. For example, if a flow is to be shaped at a particular Time Division Muliplexing (TDM) type peak rate, then the rate at which the tokens are added to the counter would specify the peak rate and by keeping the bucket depth such that it only allows a single cell/packet's worth of tokens to get accumulated limits the burst size to a single cell or packet.

The network can thus shape a flow to a particular profile by adjusting the rate at which tokens are added to the counter as well as the number of tokens that can be accumulated. Often times it is desirable to shape to the peak as well as the average rate. This would require the use of two buckets. One bucket would regulate the peak rate and other the average rate. A cell/packet would enter the network if there are enough tokens in both the buckets and would be buffered otherwise. This arrangement is commonly referred to as the dual leaky bucket configuration as shown in Figure 2.

Figure 2 Dual Leaky Bucket Configuration



Policing uses the same mechanism as shaping. Instead of buffering cells/packets, the policing agent either discards or tags the packets if enough tokens are not available. For example, in ATM networks, cells would be either tagged or discarded if they do not conform to the specified profile. In DiffServ, different policing configurations could be set up.

One such configuration is the “two rate three color” marking defined by DiffServ. This configuration uses dual leaky buckets to police average rate and the peak rate of the flow. The packet profile is marked as yellow if it exceeds the average rate, it is marked as red if it exceeds the peak rate, and it is marked as green in all other cases. Red and yellow markings define the severity of action taken by the network on that particular packet. Generally, red would mean discard immediately and yellow would be mean discard only if the network is congested. The GCRA algorithm for ATM networks is also based on the leaky bucket mechanism.

Cells or packets that are not discarded enter the switch/router and are buffered if the outgoing link is busy. Buffering is a limited resource in switches/routers and has to be shared among connections in a manner that augments the QoS policy offered by the network. Active queue management (AQM) is a set of different buffer management policies and frameworks that define ways in which buffers could be shared by different connections.

Active Queue Management

So how should buffers be managed for effective QoS management? There is no single answer to this question because it depends on the traffic profile of the applications, the qualities of service offered, and the traffic engineering policies. Buffering temporarily holds traffic when the input rate exceeds the outgoing rate. Since routers and switches have a finite amount of buffering available, cells/packets can be discarded due to buffer overflows if the input rate exceeds the outgoing rate for long periods of time.

An efficient AQM mechanism tries to prevent and minimize cell/packet discards by sharing the buffer space between different connections and selectively discarding cells/packets during traffic overload conditions. Generally, AQM mechanisms are designed with the following objectives in mind:

- Prevent a source/sources from getting an unfair share of the buffer resources
- Prevent the switch/router from getting into a congested state
- If congestion does happen, discard or tag cells/packets/descriptors in fair proportion dictated by the discard policy

There several different mechanisms and flavors of AQM. Some of the more popular ones are listed below:

- Discard Thresholds
- Random Early Detection (RED) and Weighted RED (WRED)
- Early Packet Discard (EPD) and Partial Packet Discard (PPD).
- Buffer Sharing (taking advantage of statistical gain)

Discard Thresholds

Discard threshold is a simple thresholding scheme that starts discarding cells/packets if the queue fills up beyond a predefined level or threshold. Discard thresholds are primarily used to prevent switch/router buffer resources from being unfairly consumed by a few aggressive connections, leaving the rest of the connections with little or no buffer space.

Discard thresholds can also be used to selectively discard traffic belonging to the same flow/connection in event of congestion. For example, ATM connections (CBR, VBR or UBR) may have a cell loss priority (CLP) bit set indicating a discard preference during congestion. As soon as the queue occupancy goes beyond the pre-defined discard threshold value, cells marked with CLP bit would be discarded.

IP DiffServ recommends multiple discard thresholds per queue, where each discard threshold indicates an increasing level of congestion in the switch [8]. Packets in the DiffServ architecture are marked based on the degree of non-conformance and would be discarded as the queue grows past the threshold associate with that color.

Random Early Detection (RED) and Weighted RED (WRED)

Simple discards of the type described above do not work very well for TCP/IP traffic, and in some cases may cause more congestion and reduced overall throughput.

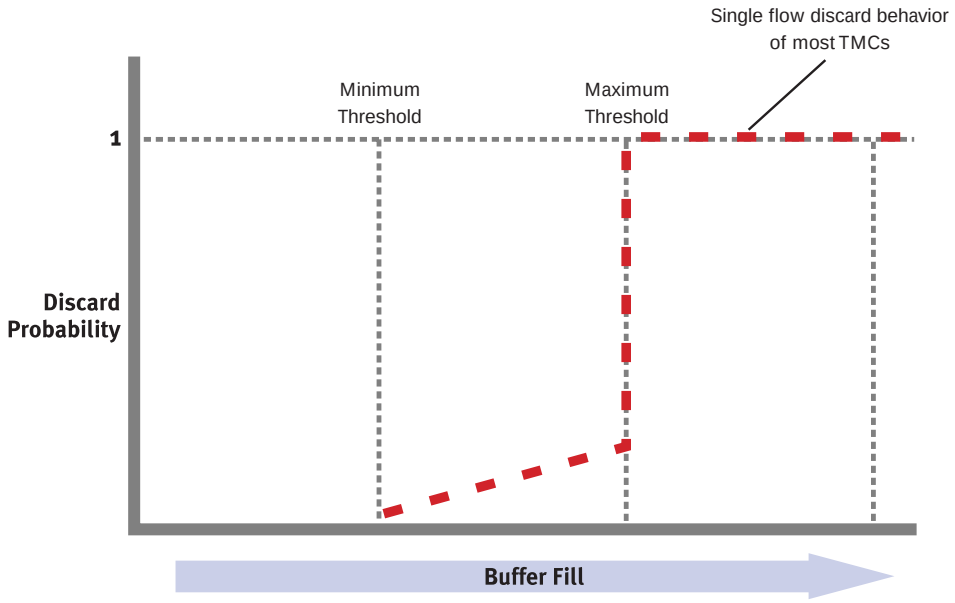
RED, on the other hand, is designed for TCP/IP and has been shown to provide better network throughput as opposed to simple thresholds [2]. RED was designed with the following objectives in mind:

- Detect the beginning of congestion
- Allow bursty traffic by absorbing transient congestion
- Prevent sustained congestion
- Avoid synchronization of TCP connections by randomly choosing connections/ flows to mark or discard

RED achieves these objectives by water marking the average queue length with a minimum and a maximum threshold. The average queue length is calculated at every packet enqueue or dequeue time, or both, to get an accurate value of the average queue length.

If the average queue length is below the minimum threshold, packets flowing into the queue are neither marked nor discarded. As the average queue length extends beyond the minimum threshold level, packets are marked or discarded with increasing probability until the maximum threshold is reached. Once the average queue length extends beyond the maximum threshold all packets are discarded. Figure 3 shows the discard behavior of RED as the queue length increases.

Figure 3 Discard Behavior of Random Early Detection



RED differs from simple thresholding in two ways. First, it compares the minimum and the maximum thresholds to the average queue length rather than instantaneous queue length. And secondly, it randomly chooses packets to discard or mark rather than the “all or nothing” mode of simple thresholds. The average queue length is chosen to allow the switch/router to absorb congestion caused by a temporary upsurge of traffic. If, however, congestion persists, RED would start randomly dropping or marking packets in the hope that sources whose packets are dropped or marked will throttle themselves, and thus reduce congestion. The frequency of discard or marking packets increases as the average queue length approaches the maximum threshold. Unlike the simple thresholding scheme, probabilistic discards or markings in RED also prevent global synchronization of TCP sources as they try to adjust to congestion in the switch.

RED operates on an aggregation of flows and does not differentiate between different flows within the aggregated group. To provide further differentiation within the group a variant of RED called Weighted RED (WRED) is used. WRED, unlike RED, uses a weighted probability value to mark or discard packets. The weights are based on the QoS policy associated with a particular flow or a group of flows. For example, DiffServ recommends up to four sub-classes within the Assured Forwarding class. The recommendation also suggests that these sub-classes be treated differently in terms of how their packets are marked and discarded. Using WRED, different weights could be assigned to each class, thereby resulting in different discard behavior for each class.

Early and Partial Packet Discard

The ATM Forum Traffic Management (TM) 4.0 specifies an intelligent packet discard function when TCP/IP packets are carried over ATM as an optional congestion recovery procedure. ATM uses AAL-5 adaptation layer to segment and reassemble packets to and from cells. The objective of intelligent frame discard is to maximize the number of complete packets transferred through the network.

Two schemes that are commonly used by the industry for intelligent frame discard are:

- **Early Packet Discard (EPD)** — Occurs when the switch/router gets congested. EPD discards every cell from an AAL-5 Protocol Data Unit (PDU) and prevents cells from entering the buffer, while at the same time reserving buffer space for cells from packets already admitted to the buffer.
- **Partial Packet Discard (PPD)** — Discards all remaining cells of an AAL-5 PDU once it has discarded a cell (except the first or the last) belonging to the same PDU.

Once the buffer level exceeds the EPD threshold, all cells from entire frames are discarded. Only cells from packets that have already been admitted into the buffer are allowed to enter the buffer. The PPD mechanism discards all remaining cells of a PDU and prevents them from entering the buffer if cells from the same PDU had been discarded earlier.

Buffer Sharing

In the long run, the ingress and egress traffic of a switch/router interface should be equal. However, for brief periods of time, it is possible for the ingress traffic at one or more interfaces to exceed the egress bandwidth. Increase of traffic at one interface often results in reduction of traffic at another interface due to the "negative co-relation of traffic". To take advantage of this negative correlation of traffic, sharing of buffer space among interfaces, classes and even flows is desirable for optimum utilization of the finite buffer space available in the switches and routers.

Schemes of allowing complete sharing or partial sharing could be employed to take advantage of this negative correlation. Partial sharing reserves some buffer space for each flow, class, or interface while sharing the rest with others. Complete sharing does not reserve any buffer space and allows flows, connections and interfaces to occupy all of the buffer space available. Complete sharing may result in an optimum utilization of the available buffer space, but also has the drawback of completely shutting out some of the flows, classes, or interfaces.

Implementing AQM

The AQM schemes described can be applied separately or together to improve the network performance and offer service differentiation. By mapping applications and services to different discard thresholds and RED parameters, for example, service providers can control and extract multiple levels of data loss, delay and jitter performance from the network. The choice of AQM policy along with the respective parameter values (discard threshold values and so on) depends on the QoS offered and the profile of the dominant applications. AQM policies can also be used in concert with one another. For example, WRED can be used in conjunction with simple discard to protect conforming flows from non-responsive malicious flows within the same class.

Once the cells/packets enter the switch/router and are not discarded by either policing or AQM, they have to be transmitted on the outgoing link. This act of transmission is called scheduling and the logic that performs this action is called the scheduler.

Scheduling

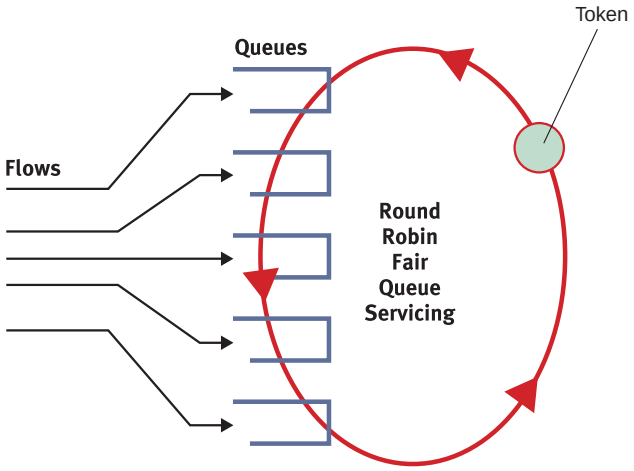
A scheduler modulates the outflow of cells/packets to suit the QoS policy that the network provider is supporting. There are several scheduling algorithms that can be used to moderate the outflow of cell/packet, the simplest being a FIFO.

Fair Queuing Algorithm

The concept of Fair Queueing is a key underlying principle of scheduling. The basic use of Fair Queueing is to distribute the link bandwidth between different connections in a fair manner while preventing greedy non-responsive connections from taking an unfair share of the bandwidth [5]. In contrast, a simple FIFO mechanism allocates bandwidth to connections in proportion to the traffic they generate, and as such does not prevent greedy non-conforming connections from taking an unfair share of the link bandwidth.

Fair Queueing requires that each connection is allocated its own queue and that the queues are serviced in a round robin fashion. Empty queues are skipped, so given a certain number of active connections, the bandwidth is equally divided between those connections, providing they all are sending the same sized cells/packets. This scheme, however, has some serious limitations. First it ignores packet lengths, therefore, connections that send larger packets would get a larger share of bandwidth than those that send smaller packets. Secondly, it is not cognizant of packet arrival times. A packet that arrived into an empty queue, just after the queue was examined would have to wait until the other queues have been examined before given the chance to transmit

Figure 4 Per Connection Fair Queuing



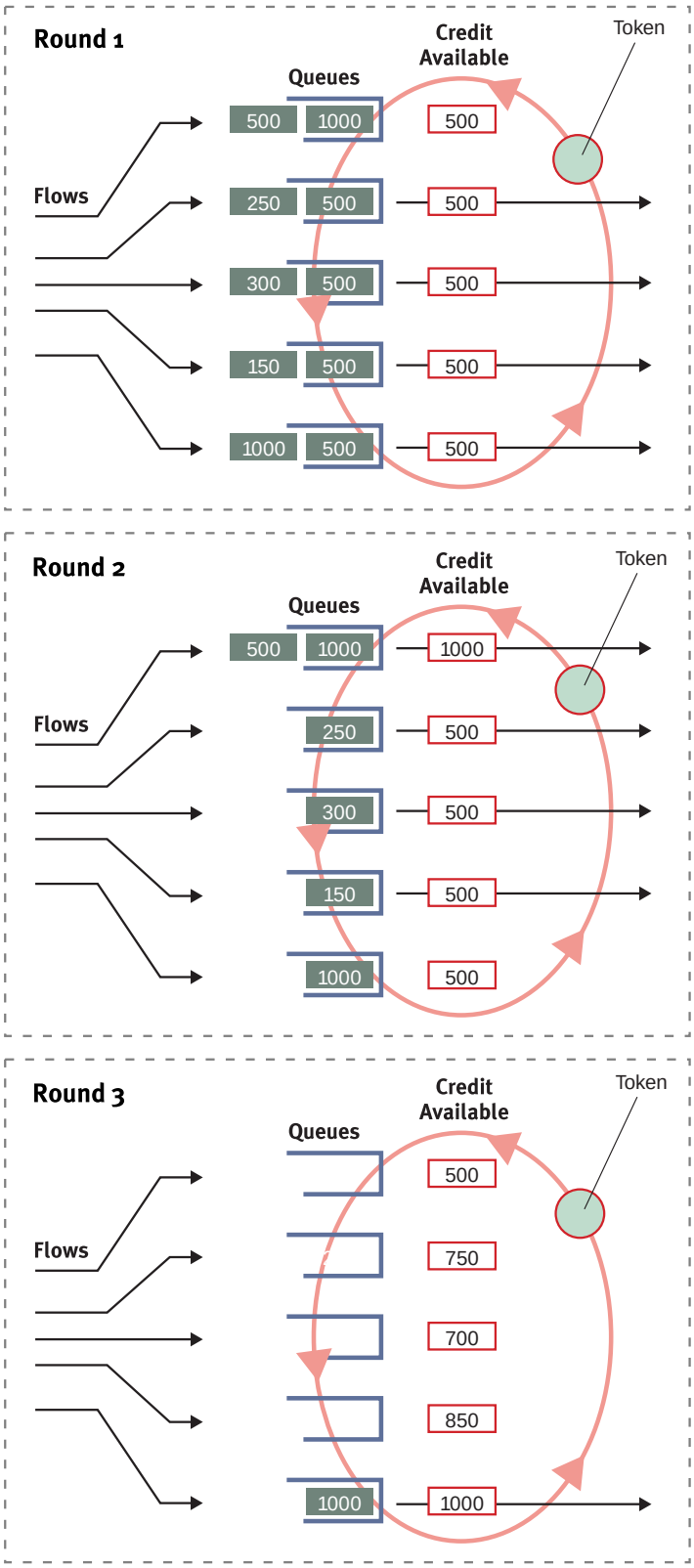
Because of the limitations of Fair Queuing, other prominent scheduling algorithms were invented that improve upon the basic idea of isolation and fair bandwidth allocation but overcome the drawbacks of Fair Queueing.

Deficit Round Robin Algorithm

The Deficit Round Robin (DRR) algorithm [19] is a variant of the Fair Queueing algorithm. It removes the drawbacks of Fair Queueing algorithm by using a deficit counter. Each queue in the DRR architecture is allocated a fixed number of credits. Unused credits accumulate in the deficit counter after every round. If a queue was backlogged and there are enough credits the packet would be serviced, otherwise it would have to buffered until enough credits build-up.

Consider the Deficit Round Robin (DRR) queueing scheme shown in Figure 5. Each of the four queues have been allocated of 500 bytes of credit. If the queue is backlogged (that is, it has packets to send) then in each round it can send up to 500 bytes. If the number of bytes in the deficit counter are equal to or more than the length of the packet, the packet would be transmitted, otherwise it would have to wait until enough credits build up. In the example, the packet from queue one would have to wait a couple of rounds before it could be transmitted. In the DRR architecture, the bandwidth allocated to a connection is proportional to the ratio of the connection's weight to the sum of the weights of the active connections.

Figure 5 Deficit Round Robin Queuing Scheme



Deficit Round Robin does allocate bandwidth fairly among connections, but it does not address the latency problem. A connection that becomes active, just after the scheduler has checked it would have to wait for all the other connections to transmit before it can will be serviced. Frame Based Deficit Round Robin (FBDRR), improves upon DRR and reduces the DRR worst case latency. FBDRR introduces another variable called the quantum, in addition to the weight defined in DRR. In FBDRR, the scheduler only allows a quantum's worth of data to be sent before moving on to the next queue. The main difference between FBDRR and DRR is that the weight may be equal to several quantum. It should be noted that the quantum has to be at least equal to the maximum packet size.

The latency experienced by a connection before its cells/packets are transmitted in an FBDRR scheduler is given as:

$$Latency_{FBDRR} = \sum_{i=0,\, i \in B}^{N} \frac{quantum_i}{BW}$$

From the above equation, it is clear that the latency in servicing a connection is equal to the time it would take to service a quantum's worth of data of all active connections, regardless of the weights assigned to those connections. For example, if there are 1000 connections, each with a 64KB quantum, then with a link bandwidth of 2.4×10^9 the latency would be equal to 213 microseconds. This latency may not be acceptable for real-time applications. To further improve on the latency and allow more flexibility in assigning data transfer rates to connections, the Weighted Fair Queueing (WFQ) algorithm was designed [6].

Weighted Fair Queueing

The WFQ scheme is based on the Generalized Processor Sharing (GPS) framework [6]. In the GPS framework, backlogged sessions are serviced simultaneously in proportion to their service shares. Using the GPS framework it is possible to bound the end-to-end delay of a connection through the network irrespective of the cell/packet arrival rate of other connections. GPS systems are also fair in distributing the bandwidth among different connections. However, the GPS system cannot be realized in the real world since it assumes that traffic is infinitely divisible and that multiple traffic streams can receive service simultaneously.

In real world systems, traffic is packetized and only a single stream can be serviced at any given time. Packet approximations to the GPS system are called Packetized WFQ. In the packetized version of the GPS system, packets are serviced by the scheduler in increasing order of start-times or the finish-times of packets. These start and finish times are calculated as follows:

$$S^f_i = max\{v(t), F(p^f_i)\}$$

$$F(p^f_i) = S^f_i + \frac{PacketLenght}{BandwidthShare}$$

Where V(t) is the virtual time. Different implementations of WFQ define V(t) differently. WFQ algorithms are work conserving algorithms that are fair in distributing link bandwidth among connections. Any unused bandwidth is distributed according to the weights of the active connections.

Several improvements to the basic concept of WFQ have been reported [21], [1], [4]. The WFQ algorithms distribute bandwidth in relation to the weights assigned to different connections, however, they do not guarantee TDM type bandwidth allocation for connections that require minimum bandwidth guarantees. Changing the virtual time to actual time allows WFQ algorithms to become non-work conserving and hence able to guarantee TDM type minimum bandwidth. A combination of WFQ algorithms that provide sharing of bandwidth on the weights assigned to a connection and minimum bandwidth guarantees is essential to support of VBR and DiffServ AF services.

Requirements of Traffic Management Coprocessors to Support Comprehensive QoS

The objectives of implementing QoS controls in the network are two fold:

- Maximize network utilization while meeting the end-to-end QoS requirements of a diverse set of applications and services.
- Provision and reprovision new services to meet the changing needs of applications and customers in a time effective manner.

To meet these goals, a traffic management coprocessor (TMC) should have the following attributes:

- **Multiprotocol** — The TMC should be flexible enough to support multiple technologies and protocols to provide the equipment vendors the flexibility to implement any standard or non-standard protocol. This attribute will also provide them with the added advantage of upgrading the line card interface to another protocol in the future without changing the underlying hardware.
- **Micro-level QoS Controls** — The TMC should be able to implement cell/packet level controls not only at the class level but also at the flow/connection level. This attribute allows the equipment vendor to preserve the QoS of each individual flow/connection even within a service class, and as such the equipment vendor can honor Service Level Agreements (SLA) on a per connection/flow level as opposed to a class level. Micro-level controls at the cell/packet level include large number of queues to buffer data from connections/flows, as well as the ability to police, shape, apply AQM to, and schedule at the connection/flow level.
- **Flexible Scheduling Hierarchy** — The TMC will typically use schedulers to aggregate traffic, but the greater the degree of aggregation (levels of a scheduling hierarchy) offered by the TMC, the greater flexibility is provided in deploying new and differentiable services. A scheduling hierarchy should be able to be configured up to the maximum level. That is, it should be possible to either collapse the scheduling hierarchy or expand it to suite the service/class requirement. Furthermore, each scheduler within the scheduling

hierarchy should be flexible enough to support the QoS classes and their attributes. As an example, the scheduling hierarchy should be able to support minimum and fair share bandwidth guarantees all through the scheduling hierarchy. Priority should be supported at all levels of the hierarchy as well to allow delay sensitive traffic to flow through without getting delayed unnecessarily.

- **Buffer Management** — The TMC should support the ability to flexibly allocate buffers, which allows sharing between different traffic types and flows/ connections. This enables better congestion management and allows the network provider to take advantage of negative correlation of traffic at different ports.
- **Fully Configurable QoS** — The ability to differentiate the QoS solution is a competitive advantage for the service provider. The TMC should be configurable beyond the simple setting of weights on the schedulers, or setting of thresholds. The TMC should present choices at all functional levels (policing, shaping, AQM and scheduling) so that service providers can customize their own deployment of QoS.
- **Host Processing Bandwidth** — In most cases the configuration/ reconfiguration, connection setup and tear down and so on would be done by the host, there should be sufficient bandwidth available between the Traffic Management co-processor and the host to not get congested.
- **Programming QoS Efficiently and Quickly** — QoS management is complex to start with, add to it all the hardware intricacies and the QoS solution becomes very complex. It is highly desirable from a network/service provider point of view to reduce this complexity by providing high level APIs that ease provisioning, dynamic bandwidth, and QoS modification in real time, as well as preserving the investment in QoS management software by being backward compatible.

References

[1] J. C. R. Bennett and H. Zhang, WF2Q: Worst-case Fair Weighted Fair Queueing", INFOCOM 1996, Mar, 1996.

[2] S. Floyd and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transaction on Networking*, 1(4), August 1993.

[3] E. Gelenbe, X. Mang, and R. Onvural, "Bandwidth Allocation and Call Admission Control in High-Speed Networks", IEEE Comm. Magazine, Vol. 35, No. 5, May 1997, pp. 121-127.

[4] P. Goyal, H. M. Vin and H. Chen, "Start-time Fair Queueing: A scheduling algorithm for integrated services", ACM-SIGCOM 96, pages 157--168, Palo Alto, CA, August 1996

[5] J. Nagel, "*Congestion Control in IP/TCP Internetworks; RFC-896*", RFC 896, DDN Network Information Center, January 1984.

[6] A. Parekh and R. Gallager, "*A generalized processor sharing approach to flow control in Integrated Services Networks: Single Node Case*", IEEE/ACM Transaction on Networking, June 1993.

[7] RFC 2475, "An architecture for Differentiated Services", *IETF RFC 2475*, December 1998.

[8] RFC 2597, "An Assured Forwarding PHB", *IETF RFC 2597*, July 2001.

[9] RFC 2598, "An Expedited Forwarding PHB", *IETF RFC 2598*, Sept. 2001.

[10] RFC 3031, "Multiprotocol Lable Switching Architecture", *IETF RFC 3031*, January 2001.

[11] RFC 3033, "Signalling for the Internet Protocol", IETF RFC 3033, January 2001.

[12] S. I. A. Shah, P. Ashton, and M. Wernik, "Constant Bit Rate Services on ATM," *Special Issue on Traffic Management of International Journal of Communications*, 1994.

[13] S. I. A. Shah and A. Girard, "Multi-service Network Design: A decomposition approach," *IEEE Globecom 98*, Sydney, Australia, Nov. 1998.

[14] S. I. A. Shah and H. T. Mouftah, "Scalable QoS Network Design," *Conf. Record IEEE ICC*, June 2000, New Orleans, USA.

[15] S. I. A. Shah, J. Yan and M. Beshai, "Designing for Uncertainties of ATM Traffic", *Conf. Record ITU Asia Telecom 97*, Singapore, 1997.

[16] S. I. A. Shah, J. Yan and M. Beshai, "Designing for Uncertainties of ATM traffic", *Proc. Telecom Asia*, ITU, 1997.

[17] S. I. A. Shah and T. Yang, "ATM Resource Allocation Algorithms: A comparison," *IEEE Globecom 95*, Nov. 1998, Singapore.

[18] "Traffic Management Specifications", Version 4.1, AF-TM-0121.000, The ATM Forum, March 1999.

[19] George Varghese, "Efficient fair queueing using deficit round robin", ACM-SIGCOMM 1995, vol. 25, pages 231--243.

[20] J. Yan, S. Shah and M. Beshai, "Service Specific networks within a shared network", *IEEE Symposium on planning and design of broadband networks*, Canada, 1996.

[21] Z. L. Zhang, D. Towsley and J. Kurose, "Statistical analysis of generalized processor sharing scheduling discipline", IEEE Journal on Selected Area in Communications, 13(6): 1071-1080, August 1995.21

For more information about the Q-5 Traffic Management Coprocessor, please contact your local Motorola sales representative or call (800) 521-6274. You can also visit Motorola's Smart Networks Web site at:

www.motorola.com/smartnetworks

MOTOROLA

QOSM-WP/D
Rev 0 October, 2001