

# A Service Architecture for ATM: From Applications to Scheduling

Mark W. Garrett, Bellcore

## Abstract

This article derives a rationale for the service architecture of the ATM Forum's Traffic Management 4.0 specification. This model distinguishes a small number of general ways to provide quality of service (QoS) which are appropriate for different classes of applications. We construct the set of ATM service categories by first analyzing the QoS and traffic requirements for a reasonably comprehensive list of applications. The most important application properties and the complexity of the related network mechanisms are used to structure the services. This method has the desirable property that the number of service categories does not expand rapidly with the introduction of new applications. We also discuss packet scheduling as the key component for realizing such a set of services, and report on an experimental realization of a fair queuing scheduler.

The ATM Forum Traffic Management (TM) working group has recently completed its TM 4.0 specification, which includes a revised service architecture for asynchronous transfer mode (ATM) [1]. Five *service categories* have been identified as follows:

- CBR — Constant bit rate
- rt-VBR — Real-time variable bit rate
- nrt-VBR — Non-real-time variable bit rate
- UBR — Unspecified bit rate
- ABR — Available bit rate

The purpose of such an abstraction is to draw a small number of very basic distinctions in the quality of service (QoS) behavior of the network. It provides a conceptual tool to both equipment designers and network service providers by structuring the problem of providing appropriate QoS and traffic management over the vast space of application requirements.

We first provide a brief view of the historical development of this service architecture. The next section identifies a taxonomy of application classes, and the third section analyzes the traffic and QoS requirements of these applications. We then develop the service categories by considering simple and then more complex services appropriate to the applications identified. The next sections discuss the methodology of our service model derivation. In the remaining two sections, we argue that packet (or cell) scheduling is the essential mechanism for attaining meaningfully differentiated QoS behavior, and report on a high-speed scheduler implementation.

### History

This service architecture is similar to the ATM *bearer capability* (BC) concept, and is essentially a newly thought-out version of that idea. The form of the BC information element, which is signaled as part of an ATM connection setup message, dates back to the earliest standards for

broadband integrated services digital network (B-ISDN). The BC service model distinguishes three properties: whether a source is constant or variable rate, whether the source and destination are time-synchronized, and whether the service above the ATM layer is connection-oriented or connectionless.

By comparison, our new model essentially retains the CBR/VBR distinction. The timing requirement in the BC is a very strong property. When it is asserted, the network must provide some explicit means of timing recovery, such as the synchronous residual timestamp mechanism of ATM adaptation layer 1 (AAL-1). In the new model we have a fundamental notion of real-time or non-real-time, but it is limited to determining whether timing parameters such as delay and jitter are specified. Finally, the connection-oriented indication of the BC does not serve a useful purpose because a flow must always set up a connection in ATM. Whether or not a higher-layer protocol is connection-oriented has little effect on the behavior of the ATM network. The BC service model is pervasive in the International Telecommunications Union (ITU) traffic management standards, and provided the original structure for both the AALs and the QoS classes [2].

In order to better accommodate data communications applications, the ATM Forum incorporated a "best effort" indication in the signaling specification of User-Network Interface (UNI) 3.0 [3]. During the early TM discussions which led to the TM 4.0 model (in September 1993), it became clear that best effort should be split into two modes, a "plain best effort" (which became UBR) and a "better best effort" (which became ABR) [4]. One contributing factor was that people from the telecommunications industry were less comfortable with a totally unspecified service than those from the data communications side. It was not clear for several months, however, exactly how these two services would be distinguished. In

early 1994, the genesis of ABR emerged as a consensus that an ATM data-oriented service should offer a very low cell loss rate (in order to keep the Internet Protocol packet loss rate reasonable [5]), and that this would be accomplished through some sort of native ATM feedback flow control. The VBR service underwent a similar split and clarification. In September 1994, Dan Grossman of Motorola presented an argument that VBR, which is characterized by a double leaky bucket (LB) traffic description, should have both a real-time and a non-real-time version, the former of which includes delay and jitter parameters [6].

The rationalization presented in this article for the ATM service model was contributed to the ATM Forum in September 1994 in order to develop consensus that this model was correct, useful and not arbitrary [7]. Although it was presented publicly after the model was effectively in place, the work leading to this result actually predated the best effort split of 1993 (so it was not as much an afterthought as it might seem).

We proceed with the derivation by looking at a set of applications and identifying the important characteristics. We show that the structure in the service categories reflects structure in both the application set and the switch functionality.

### A Top-Down Approach: Start with Applications

An assumption is sometimes made in the networking research community that because the future will be full of new and radically different applications, we cannot learn to design future networks by looking at current communications applications. While the premise of this statement has some truth to it, the conclusion does not. The current applications will continue to exist (largely), and collectively they do represent a very wide range of features and requirements.

Consider the list of application classes in Table 1. These are at a level of abstraction corresponding to what the user is generically doing. This list cannot be claimed as complete, of course, but it is large and representative, and therefore sufficient for our purpose. Note that this list does not include terms like "circuit emulation" or "switched multimegabit data service (SMDS)," because these are services at a lower level of abstraction and themselves carry more basic applications. It also does not distinguish between business video conferencing and residential video telephone; both would be interactive video at this level.

This list is derived (roughly following an ITU taxonomy [8]) by considering video, voice, image, and data in conversational, messaging, distribution, and retrieval modes. Some computer applications have been added which vary in volume of traffic and tolerance to time delay. *Conversational* or *interactive* means there is a person on each end of the connection. *Messaging* means a person talking to a machine. *Retrieval* is a person causing a machine to transfer information. *Distribution* is a machine sending to people or machines who listen passively. The computer communications application classes listed (17–21) are machine-to-machine interactions.

Application Class	Example Applications
(1) Interactive video (2) Interactive audio (3) Interactive text/data (4) Interactive image	Video conferencing, distributed classroom Telephone Banking transaction, credit card verification Multimedia conferencing
(5) Video messaging (6) Audio messaging (7) Text/data messaging (8) Image messaging	Multimedia e-mail Voice mail E-mail, telex, fax High-resolution fax
(9) Video distribution (10) Audio distribution (11) Text distribution (12) Image distribution	Television Radio, audio feed News feed, netnews Weather satellite pictures
(13) Video retrieval (14) Audio retrieval (15) Text/data retrieval (16) Image retrieval	Video on demand Audio library File transfer Library browsing
(17) Aggregate LAN (18) Remote terminal (19) Remote procedure call (20) Distributed file service (21) Computer process swap	LAN interconnection or emulation Telecommuting, telnet Distributed simulation

■ Table 1. List of application classes and some example applications.

Each of these classes corresponds to a set of applications; some well-known examples are indicated in Table 1. Signaling for the ATM network itself is an important source of traffic. In terms of QoS, it is an asynchronous application which requires timely delivery. Signaling would require a QoS perhaps similar to either remote procedure call or distributed file service.

For some applications there may be different technical solutions, such as CBR MPEG vs. VBR motion-JPEG coding for video, that significantly affect the traffic and QoS of the application. These innovations often improve a service by exploiting trade-offs (e.g., statistical multiplexing for loss), which in turn impose different requirements on the ATM service. These differences are not reflected in Table 1, but will come into play in defining the service categories.

### Detailed Discussion of Traffic Description and QoS Requirements

From our list of applications, we can identify a number of important traffic and QoS issues, such as the following: CBR/VBR, degree of burstiness, statistical multiplexing, real-time delay constraints, delay tolerance in non-real-time applications, interactivity, loss tolerance, priority, use of leftover bandwidth, multiresolution coding (e.g., for loss concealment), QoS consistency, and fairness. Keeping this list in mind, we can now analyze the QoS needs of our set of applications.

Audio and video can be transmitted conveniently using CBR because they are traditionally CBR-coded for circuit transport. Even with packet transport and VBR coding, the peak-to-average ratio for audio and video is typically in the range of 3 to 10 [9, 10]. Since this number bounds the statistical multiplexing gain, we could argue that CBR transport is moderately efficient for real-time audio and video applications.

Traditional computer communications applications as well as video/audio/image applications which transport dis-

crete units of information (i.e., messaging and retrieval applications) can be quite bursty and therefore have a lot to gain from statistical multiplexing. Also, such *unit-oriented* applications do not have a natural rate. A first-order description of QoS for such applications is that the transaction is complete when all of the data arrives; the sooner the transfer completes, the better the quality. The user does not care about the timing details of particular cells, only about the transfer seen as a single object. This property identifies a fundamental distinction between such *unit-oriented* applications and *rate-oriented* applications such as voice and video.

Considering this more carefully, however, we can note that there is a minimum transfer time, below which the user will not notice any better quality. For real-time applications, a late cell is a lost cell. For unit-oriented transfers, it is not so easy to describe good delay QoS concisely. For example, some computer communications applications are sensitive to delay because they involve tight interaction between machines. These include remote procedure calls, distributed file service, and memory paging/swap. (Note that these applications can have different sensitivity to delay, even though we may call them all "non-real-time" or "delay-tolerant.") An occasional bad delay is quite tolerable, but excessive average delay will badly affect the machine processes' efficiency. A unit-oriented application can also be interactive. A text-based interaction between people or between a person and a machine is also degraded by excessive average delay, although an occasional long delay is tolerated.

Some applications are semi-interactive, meaning that one side of the conversation treats the communication as interactive and the other does not. For example, any audio/video messaging or retrieval application has a person on one end and a machine on the other. This can be handled by either a real-time rate-oriented transport or a non-real-time unit-oriented transport. An answering machine takes advantage of the existence of the telephone network, and the information is transported as if the application were interactive. One could alternatively record the message locally and send the audio file by electronic mail. In this case the transport would be noninteractive. This choice depends on what network options are conveniently available to the user.

All messaging applications are very tolerant to delay. Distribution applications can be delayed substantially, but the delay variation has to be bounded because a buffer is necessary to re-synchronize the playback. For retrieval applications, a user may be somewhat sensitive to delay beyond some number of seconds; however, this is much longer than the interactive real-time delay constraint.

Computer communications applications are generally sensitive to loss because all data must be received correctly and errors cause retransmission. The cost or inconvenience of retransmission depends on whether the application is also sensitive to delay. When a machine process must wait for retransmitted data, it may be desirable to purchase a more reliable network service. For bulk message forwarding, retransmission over a lossy or delayed channel may be perfectly acceptable and cheaper. Audio and video can be quite tolerant to loss if they are coded correctly. Loss of video frame synchronization information will cause signifi-

*The single most important distinction among applications is that of real-time and rate-oriented vs. non-real-time and unit-oriented.*

cant QoS damage, while loss of some picture element resolution or some speech samples could easily go unnoticed. The current thinking in video coding research is toward layered or multiresolution coding, where high- and low-priority streams are created. Loss of low-priority cells has very limited effect on the overall quality, while the high-priority stream must be well protected against loss [11, 12]. (Note that layered coding is not yet reflected in video standards such as MPEG, which were not originally designed for packetized transport.) The permitted loss tolerance of a video service may depend greatly on the user application. A video conferencing application may require less reliability than entertainment and distribution video, for example, because the users can easily compensate for slight glitches in the transmission.

There are two aspects of priority. The first is to differentiate between applications that tolerate loss or delay differently. The second is to determine who gets access to resources (bandwidth or buffers) in real time. Resources may be allocated to one service but, if not used, can be made available to another [13]. The statistics of the applications sharing a VBR service, for example, are interesting in two ways. First, the statistical behavior and some knowledge of the required QoS can be used to produce resource allocation rules that realize statistical multiplexing gain while retaining appropriate QoS. Second, the statistics of the sources will indicate how well the leftover bandwidth not used by that service can be relied upon by other services running at lower priority. An obvious example is that even somewhat delay-sensitive data applications can make good use of bandwidth left over from a VBR video service, because on the rare occasion when the video uses all of its allocation, the data service can easily survive on less for the moment.

Finally, whenever various users are statistically multiplexed there is an issue of fairness. Even when a service is defined symmetrically with respect to two users, it is possible that one user gets better service than another because of the quantity or timing of the traffic submitted. If there is a wide variation in quality, such as the delay in transmitting a message, then the user may be annoyed by a perceived lack of QoS consistency provided by the network. The network can potentially provide some functionality to guarantee (or aid) fair and consistent service provision across the set of users who subscribe to the same service.

### *Service Categories Derived from Application Properties*

*The single most important distinction among applications is that of real-time and rate-oriented vs. non-real-time and unit-oriented. When people are observing a continuous flow of visual or auditory information, a lack of continuity caused by excessive jitter or loss has a significant quality impact. Furthermore, when the application involves people interacting in real time, there is an overall delay constraint of a few hundred milliseconds, beyond which the delay becomes noticeable and annoying. Applications having this delay constraint require very precise handling. Those that do not can be switched more economically. The*

tolerance of delay can be exploited to allow the switch considerable flexibility.

#### *CBR Service Category*

Real-time applications are predominately those which contain video or audio information. These applications also tend not to be very bursty, in the sense that it is reasonably efficient to assign a fixed bandwidth to a single source. For this reason, voice and video communications are commonly carried over fixed-rate circuits with tiny delays incurred in switches. This leads to a definition of the CBR service as a very simple, reliable, guaranteed channel. CBR is (by this definition) intended for real-time applications, and therefore assumes a real-time delay constraint. The allocated rate is defined, and the application is assumed to offer traffic constantly at this rate. The acceptable loss rate, maximum delay, and maximum jitter are also specified.

Although CBR may be the simplest service to define, it may not necessarily be the easiest to implement well. For example, consider a CBR connection spanning several administrative domains. The CBR rate is measured and policed by an LB mechanism<sup>1</sup> at each administrative boundary. After traversing the first network, the measured jitter, or cell delay variation (CDV), will surely increase. The traffic entering the next network is then more bursty, even though the description signaled at connection setup is the same. One approach to this problem is to initially set the bucket size parameter to accommodate the jitter inherent in the source and added by all of the networks in the path. However, this will probably lead to overly pessimistic engineering rules because the entity setting up the connection does not generally know about the whole path. The problem of CDV accumulation, and how it scales as the network gets very large, is perhaps one of the most important and underestimated open problems in ATM networking.<sup>2</sup>

#### *UBR Service Category*

The applications that will not utilize a CBR channel efficiently are those which sporadically send discrete units of information. Most computer communications and messaging applications behave this way and therefore benefit greatly from statistical multiplexing. These applications do not have the real-time delay constraint, but can be moderately sensitive to loss. We can call these applications unit-oriented, as distinguished from rate-oriented, because fundamentally the object is to transfer a fixed quantity of bits rather than to provide a continuous flow.

The most natural, simple service discipline (in terms of implementation) for these applications is first-in first-out (FIFO) queuing, as we find in the Internet and other computer networks. The tolerance to delay implies that large buffers are appropriate in the network because they trade delay for loss. The notion of a simple service for non-real-time bursty traffic (for which a FIFO might be an appropriate implementation) is what we commonly call "best

*We can now add more service categories by finding features of applications which are not completely or efficiently satisfied by the existing categories.*

effort." This has evolved in the ATM Forum TM group to be called unspecified bit rate (UBR) because it is a service without any explicit rate parameter. Note that UBR is exactly the service model of the current Internet, which argues that, although simple and lacking much control, it is quite useful and commercially viable.

The UBR definition therefore includes no notion of rate or GCRA parameters, although it may be useful to negotiate a ceiling rate corresponding to the smallest bottleneck along the path (the peak cell rate parameter is used for this). Similarly, there is no cell-wise delay or jitter requirement, or any explicit loss rate contract. The QoS in this case is determined, not by algorithms operating on each cell, but by engineering the capacity of the overall system to accommodate the overall traffic demands. Although there may be no rate

allocation per virtual circuit (VC), it is probably a good design choice for the UBR service, as a whole, to have an amount of bandwidth assigned to it which cannot be pre-empted by other services.

In the current Internet, the subnetwork itself is assumed not to do any congestion control. An end-to-end protocol such as TCP regulates the flow of traffic. This model is generally appropriate for ATM UBR service as well. In cases such as Internet Protocol (IP) over ATM, where a retransmission protocol operates with a data unit larger than a cell, UBR should include an enhancement such as early packet discard [5] to maintain good performance.

#### *Progression from Simple to Complex*

CBR and UBR, therefore, are two very simple services which can accommodate the vast majority of applications. We can now add more service categories by finding features of applications which are not completely or efficiently satisfied by the existing categories. This approach gives us a systematic process and justification for creating new service categories.

#### *Real-Time VBR Service Category*

An improvement in efficiency can be made by allowing real-time applications to use VBR coding. Sources can be statistically multiplexed, resulting in more efficient bandwidth usage. Since multiplexing implies some nonzero probability of loss, the applications must be coded appropriately.

We can define a VBR service category as real-time (like CBR), but where the source rate is allowed to vary. The acceptable cell loss rate, delay and jitter are specified. The sustainable cell rate and maximum burst size (SCR/MBS) parameters can be used in addition to the peak rate to describe the variable bandwidth process. An effective way to provide statistical multiplexing while ensuring appropriate QoS for VBR video is to make engineering rules for the amount of bandwidth and buffers that are necessary for a given number of multiplexed sources. To produce these rules would require extensive experience with user-perceived quality as a function of allocated resources and perhaps other properties of the service. This design is more feasible when a dominant type of coder exists, and its statistical behavior can be studied in depth. For this to

<sup>1</sup> In ATM terminology, this is the Generic Cell Rate Algorithm (GCRA) [1].

<sup>2</sup> Jitter accumulation is no less difficult in the Internet, if we hope to introduce resource reservation and guaranteed delay bounds.

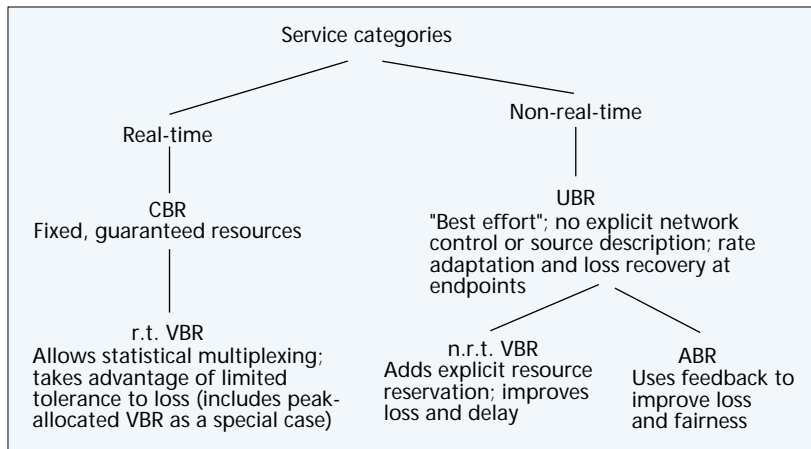


Figure 1. Architecture of ATM service categories. The fundamental dichotomy is between real-time and non-real-time applications. For each case there is a very simple solution and then one or two more complex solutions.

work in ATM, it may be useful to signal a source type parameter to identify the appropriate set of engineering rules for the application. (Note that currently no such parameter exists, and, in general, an adequate consensus regarding the use of rt-VBR has not been attained in the industry.)

There are two flavors of real-time VBR, which are not explicitly separated in the standards. In the first, the traffic rate varies but the QoS (loss and delay) is consistent because sufficient resources are always guaranteed (call this peak-allocated VBR, or PVBR). In the second, the sources are truly statistically multiplexed, so both the traffic presented and the throughput provided to the source vary (call this statistically multiplexed VBR, or SMVBR).

For PVBR, the allocated resources are always sufficient to ensure no congestion loss, assuming the source conforms to a condition such as the GCRA. When the source is not using its peak rate, there is some leftover bandwidth which may be utilized by other services at lower priority. Since the leftover bandwidth is intermittent, it could be used for best effort traffic (either UBR or ABR), which benefits from the extra bandwidth but has no allocation contract for it.

In the SMVBR case, the network allocates less than peak rate for each source. This introduces a nonzero probability of loss and statistically multiplexes the sources against each other. Again, any leftover bandwidth may be utilized through the use of a work-conserving scheduler.

From the viewpoint of the network's QoS guarantee and perhaps resource allocation, PVBR is like CBR. However, from the view of the traffic description, and whether there is any possibility of economizing on leftover bandwidth, PVBR is more like SMVBR. These two forms of PVBR can be distinguished in a sense in ATM. The definition of CBR [1], strictly speaking, commits resources, but does not preclude a source from underutilizing those resources. Therefore, a varying source using CBR is a PVBR service with no attempt to recover unused bandwidth, while a real-time-VBR service with peak and sustainable rates set equal allocates the peak rate, but allows the network to recover unused bandwidth if it can.

All types of video and audio applications can gain bandwidth efficiency through statistical multiplexing. However, given the choice, some applications will continue to use CBR transport because they are unwilling to incur any loss.

#### *nrt-VBR and ABR Service Categories*

For some non-real-time unit-oriented applications, QoS can be improved substantially by adding network functions that improve fairness, loss, and/or delay. As discussed previously, an appropriate delay control for these applications is quite different from one appropriate for real-time applications.

The non-real-time VBR service category improves loss and delay over UBR by providing peak and sustainable rate parameters (and corresponding bucket parameters) and a loss rate parameter. These are used to allocate resources for each nrt-VBR connection. The loss rate allows the service category to be engineered for statistical multiplexing while maintaining acceptable performance.

The main QoS considerations that have led the ATM Forum TM group to its definition of ABR have been minimizing loss and providing fairness. The network functionality that realizes these goals is a rate-based feedback flow control protocol, the details of which can be found in the recently completed ATM Forum traffic management specification [1]. End-to-end flow control algorithms (such as TCP) do not necessarily cause resources in the middle of the network to be shared fairly; nor do they minimize congestion as efficiently as might be possible using information from the congested node within the network. The operation of ABR's feedback flow control is assumed to be independent of the windowing mechanism of TCP. This raises two issues. Most immediately, how do these two protocols behave together? Is TCP performance improved by the ABR flow control? Is it ever worse? What happens when the ATM network is only a small part of the end-to-end TCP path? Some encouraging results were presented to the ATM Forum during the development of ABR, but clearly more work is needed. The second issue opens an interesting topic for research: Supposing that ATM becomes a significant technology underlying TCP/IP internets, can the explicit congestion notification from within the ATM subnet be passed to the end-to-end flow control mechanism, and does this improve performance over having independent mechanisms?

The ABR protocol does not use the LB traffic description, and the resource allocation philosophy is therefore closer to that of UBR. It has no signaled cell loss rate, although a network provider could generally advertise a CLR value with its service. No delay or jitter parameters are applicable. The ABR protocol does include an optional minimum cell rate. While this parameter is certainly useful for allocating minimum resources to the connection, and thus improving the delay averaged over, say, a file transfer, it does not necessarily provide a delay bound for each cell carried, as would be the case for a real-time service.

#### *Defining Further Service Categories*

This method of creating service categories is one of expansion by refinement. We create a tree-shaped hierarchy of choices by identifying properties that distinguish one branch from another, as shown in Fig. 1. There are several desirable properties of this method. First, there is progressive growth. An ATM switch can be implemented immediately which accommodates any application (perhaps inefficiently) using CBR and UBR. As the usefulness of VBR and ABR becomes clear, the implementation can be extended. Services are backward-compatible in the sense

that where a complex service is unavailable, an appropriate simple service will exist. This does not guarantee interoperability between switches supporting different levels of complexity, but at least it structures the problem. Also, as service categories are added, it becomes less likely that more will be justified, because the most important functions and requirements are accommodated first.

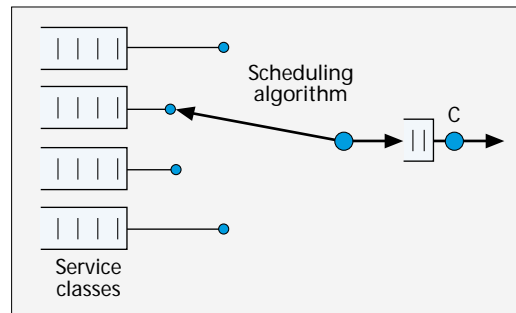
To make the decision whether to create a new category, we can ask whether a newly envisioned function is critical to the applications which use the other functions of the pre-existing category. If it provides a complementary but nonessential function for the category, then it may be appropriate to spin off a new category for it rather than to clutter the definition of the existing category. If the added complexity is minimal, however, it could be incorporated into the existing category, because a new category would not be justified by the distinction. For example, the minimum cell rate parameter was a small enough change to the ABR service that it could simply be added to the definition. Conversely, the TM group felt that extending the ABR mechanism to include real-time QoS would have required defining a new service category, so the delay and jitter parameters were not allowed to be optional within ABR.

### Service Architecture as an Appropriate Abstraction for Network Design

A variety of functional mechanisms are available to switch designers in order to realize the transport of the wide spectrum of applications ATM is intended to support. These include simple and complex scheduling algorithms, priorities, feedback flow control, and other mechanisms which selectively impose delay and loss upon traffic. A switch design combines these mechanisms to realize an appropriate QoS balance for all of the contending sources.

Both the applications and the mechanisms are concrete in the sense that their properties can be well defined, measured, and evaluated. The service model, conversely, is really an abstraction, a tool we use to help match applications to mechanisms. The service category is a concept which suggests that there is a set of important applications with certain common requirements and properties; and that some mechanisms exist which can provide appropriate QoS for these applications. The service category concept provides a way to discuss these common properties without having to identify precisely or enumerate fully the applications and mechanisms we have in mind.

To see the conceptual importance of the service model, consider the problem of designing a network without one. (This discussion reflects a debate that surfaced occasionally in the TM working group before we presented this paper in 1994.) Consider the set of quantitative QoS and traffic parameters which are commonly used in ATM (i.e., PCR, CDVT, SCR, MBS, CTD, CDV, CLR, MCR, etc.). Some might argue that each application can be quantified uniquely with these parameters. The presumption, then, is that the network will somehow know what to do. From the



■ Figure 2. The packet scheduling algorithm determines which service class to serve next. Each FIFO queue represents one service class, which may comprise one or more traffic sources. (Note that service classes in the scheduler may or may not be exactly aligned with the service categories of the service architecture.)

switch designer's point of view, however, the eight-dimensional space of possible characterizations is completely unstructured. At some point in the continuum of values for the parameters the functional mechanism has to change qualitatively, not just quantitatively. It is exactly this type of qualitative relation between applications and network mechanisms which is captured by the distinctions among the several service categories. The service model says which areas of the parameter space are significant in that they are likely to contain meaningful applica-

tions.

### Scheduling Related to Quality of Service

This section describes briefly how scheduling, together with connection admission control (CAC) and resource allocation, can be used to realize a broad range of QoS behaviors, such as those of the ATM service architecture. Admission control and resource allocation operate at the time of connection setup, while scheduling controls the transmission of each packet.<sup>3</sup> These functions provide the following properties:

- Isolation of sources or service classes from cross-traffic
- Sharing of bandwidth by connections within the same class
- Utilization of bandwidth unallocated or allocated to idle sources
- Statistical multiplexing
- Maintaining appropriately low delay and/or loss
- QoS consistency
- Fairness

Scheduling essentially creates a policy of isolation and sharing [13, 14]. Typically employed at the output ports of a switch or router, it determines which of the set of waiting packets will be served next (Fig. 2). Packets within each service class are queued together and (typically) retain their order. So the scheduler is really choosing only among the head-of-line packets of each service class.

Most early ATM switches and Internet routers used the first-in first-out (FIFO) service discipline for scheduling. FIFO is a simple solution, and is quite appropriate where the traffic sources have equal priority and roughly equivalent QoS requirements. A method to provide partial isolation for different classes of applications is through priority scheduling. A two-priority system can effectively separate real-time video sources from bursty computer sources which tolerate delay. However, when there are two statistically multiplexed classes with different QoS requirements — that is, each can overload its allocation but neither can tolerate arbitrary interference from the other — an isolation mechanism beyond priority is necessary. This is certainly the case for ATM. Because the two VBR service categories can be engineered for an acceptable level of loss, there must be some way to allocate bandwidth and buffer resources without statically prioritizing one over the

<sup>3</sup> We use the term "packet" generically as an atomic unit of data. In an ATM network the scheduling algorithm is applied at the ATM cell level.

other.

A number of schedulers which are all variations on fair queuing [15] have the effect of providing access to a share of bandwidth, as if each service class in Fig. 2 had its own server at its given rate. The fact that other classes are sharing the link does increase the delay variation somewhat over this ideal. Because these schedulers are work-conserving, they have the property that when at least one class is backlogged, the server will not be idle. This ensures that any bandwidth left unused by an idle class will not be wasted. A delay bound for real-time services is ensured by allocating enough bandwidth and limiting the buffer size for the service class. Loss rates are controlled by limiting the number of connections that can join a class. To provide a meaningful bound on loss may require more knowledge of the types of traffic sources joining a class than that provided by the double LB parameters of the ATM traffic descriptor.

### Fair Queuing Algorithms

The fair queuing service discipline was originally an attempt to ensure fair access to bandwidth for a set of similar users [15]. The number of users was assumed to vary dynamically without any explicit limit. This version also used equal shares for each source. A generalization known as weighted fair queuing (WFQ) [14, 16] allows an arbitrary balance of the users' access to bandwidth. In contrast to FIFO queuing, where a source can increase its share of service by presenting more demand, fair queuing is designed to serve users according to their weights, or shares, and any overactive sources are penalized by packet loss or delay. Thus, fair queuing is essentially a mechanism to assign bandwidth to connections or sets of connections, while allowing any leftover capacity to be shared by the active users. Fair queuing is itself a generalization of the round-robin service discipline [17, 18]. Round-robin can also be weighted, but always checks the service classes in a fixed order, and requires small-integer ratios of the weights. If a class is idle it moves as quickly as possible to the next active class. It is therefore (like WFQ) work-conserving, and particular transmission slots are not reserved. Fair queuing gains some performance over round-robin by avoiding the fixed service order.

WFQ has a significant implementation problem in the form of high computational complexity. Another variation, known as VirtualClock, was invented at about the same time [19]. This can be easily implemented, but does not always achieve the delay and fairness properties of WFQ. These properties were proved by Parekh and Gallager for WFQ when used in conjunction with the LB packet admission policy [20, 21]. Our experiment uses an approximation to WFQ called self-clocked fair queuing (SCFQ), which is implementable and yet retains most of the desirable properties of WFQ [22, 23, 24].

### Details of WFQ

A compact expression and realization for the WFQ scheme was presented in Abhay Parekh's doctoral thesis [16]. A hypothetical fluid-flow queuing (FFQ) system with the

To provide a meaningful bound on loss may require more knowledge of the types of traffic sources joining a class than that provided by the double Leaky Bucket parameters of the ATM traffic descriptor.

same server speed and an identical pattern of traffic arrivals is considered, and a *virtual finishing time* or service tag is computed. Packets in the real system are then served in the order of these tag values. The service tag  $F_k^i$  is determined as

$$F_k^i = \frac{L_k^i}{r_k} + \max\left(F_k^{i-1}, v(a_k^i)\right), \quad (1)$$

with

$L_k^i$  = length of the  $i$ th packet of class  $k$   
 $r_k$  = relative weight assigned to class  $k$   
 $a_k^i$  = arrival time of the  $i$ th packet of class  $k$

$v(t)$  = virtual time function of the corresponding FFQ system  
 $F_k^0 = 0$  for all  $k$

In order to calculate the packet service tags, we need to know  $v(t)$  for the corresponding FFQ system. Now,  $v(t)$  is a piecewise linear function which describes the cumulative service acquired by a backlogged source of unit weight [16]. The complexity of computation of  $v(t)$  depends on the frequency of its breakpoints, which occur whenever there is a change in the set of concurrently active classes in the fluid model. That is, whenever a class changes from busy to idle, or the reverse, the algorithm requires some computation.

It is possible for a large number of breakpoints to occur during a small period of time. This property makes the implementation of WFQ practically impossible — which leads us to consider approximations.

### Self-Clocked Fair Queuing (SCFQ)

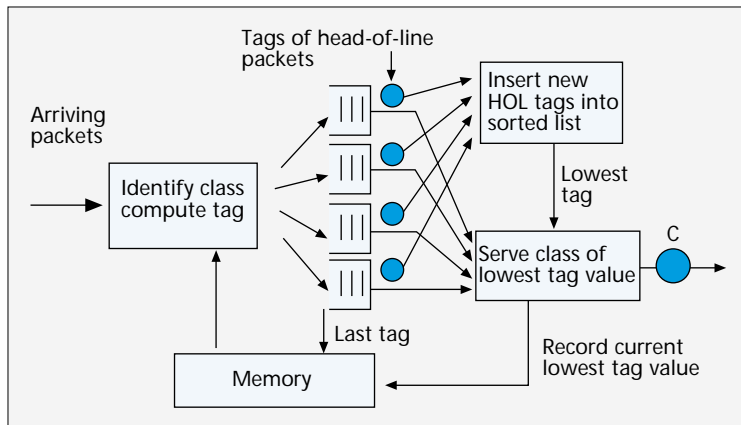
This scheme<sup>4</sup> adopts a different notion of virtual time. Instead of linking virtual time to the work progress in the FFQ system or to an outside clock (as in VirtualClock), it uses a measure of progress in the system itself; hence its name. For SCFQ the service tag is computed as

$$F_k^i = \frac{L_k^i}{r_k} + \max\left(F_k^{i-1}, \tilde{v}(a_k^i)\right), \quad (2)$$

where  $\tilde{v}(t)$  = service tag of the packet in service at time  $t$ .

Whenever the server becomes idle and no packets are queued, the algorithm may be re-initialized by setting to zero both  $\tilde{v}(t) = 0$  and the last tag for each class. The value of  $\tilde{v}(t)$  is a measure of virtual time in SCFQ as  $v(t)$  is in WFQ. However, there does not appear to be a simple relationship between the two constructions of virtual time, so the mathematical proofs of properties for each algorithm are quite different [20–23]. The advantage of this scheme is that the computation of the virtual time is negligible because it is simply extracted from each packet as it goes into service.

<sup>4</sup> This version of FQ was apparently invented by four people independently: Chuck Davin (then at MIT, now at PSI), ca. 1989 [25]; Fred Baker (then at Vitalink, now at Cisco), ca. 1990; Jamal Golestani (then at Bell-core, now at AT&T), ca. 1993 [22]; and Jim Roberts (CNET, France), ca. 1994 [26].



■ Figure 3. Functional diagram of a fair queuing scheduler in an ATM switch output port controller.

## Fair Queuing Implementation

We have constructed an experimental research prototype for cell scheduling using the output port controller (OPC) of the Sunshine Switch, which was a previous project at Bell Communications Research (Bellcore) [27]. This board is capable of running fair queuing algorithms in real time with link rates of 150 Mb/s. It has two inputs and one output with STS-3c interfaces, so the input traffic is capable of overloading the output during traffic bursts. (This is necessary, of course, to have traffic queued within the scheduler.) The board uses an i960 processor and a large cell store. The scheduling algorithm is realized in software, and controls the order in which cells are transmitted. The SCFQ algorithm has been programmed into the OPC using C code which was compiled into assembly and then further optimized.

The block diagram of Fig. 3 sketches the components of the SCFQ realization. The functions that must be executed for each receive/transmit cycle are as follows:

- Compute the tag,  $F_k^i$ , for the received packet.
- Store the packet in a FIFO queue belonging to class  $k$ .
- Identify any new head-of-line (HOL) packet for each class queue and insert it into a sorted list of HOL tags.
- Transmit the packet corresponding to the lowest tag in the sorted list.
- Store the lowest tag value (it is used for the computation of tags of subsequently arriving packets).

The only major complexity is the maintenance of a sorted list of tags from the head-of-line packets of each class. Using a software algorithm would be prohibitively complex at broadband speeds (e.g., 3  $\mu$ s for an ATM cell at 150 Mb/s). To solve this problem, we use a VLSI chip [28, 29], which can sort up to 256 items at a rate one order of magnitude faster than our requirement.

## Results

This implementation had a measured input bandwidth of about 147 Mb/s. Although not able to keep up with two full-speed inputs, it is sufficient to allow experimentation with FQ algorithms. A demonstration has been set up with two video sources running on workstations at about 4–5 Mb/s. A steady cross-traffic generator is applied to the second input to provide a constant overload on the 150 Mb/s output. A switch output with only FIFO queuing would normally drop cells from all three sources in proportion to their offered load. The fair queuing controller, however, is able to protect the video sources from the overload by allocating to them sufficient bandwidth and isolating their service from the interference caused by the overloading source.

Given this experimental realization, we can argue that such a fair queuing mechanism is easily realizable for real switches. There are processors on the market now with ten times the speed of the i960 (we're using an old version), and a commercial implementation would be reduced to faster hardware devices rather than using a general programmable platform. Between these two factors, we conclude that it should be possible to implement such a device even at 622 Mb/s without difficulty.

## Conclusions

The purpose of the ATM service architecture discussion, as it was presented to the ATM Forum in late 1994, was to develop some consensus on both the model itself and how service categories should generally be defined and distinguished. The qualitative distinctions made by service category definitions are an important supplement to the quantitative traffic/QoS parameter values. We argue that the set of service categories can be derived logically from application requirements. In particular, they distinguish real-time from non-real-time services, and then provide simple and more complex solutions for each case. The added mechanisms in the more complex categories are justified by providing some benefit or economy appropriate to a significant subset of the applications considered.

The fair queuing scheduling algorithms, and SCFQ in particular, provide an extremely useful mechanism for realizing differentiated QoS behavior in ATM switches. An experimental research prototype has been constructed to demonstrate the practicality of such sophisticated scheduling mechanisms.

## Acknowledgments

Many thanks to Takeshi Saito of Toshiba Corp. for his excellent work on the high-speed software implementation of fair queuing while he was visiting Bellcore in 1995. Bill Marcus designed and built the OPC hardware used for this experimental research prototype. His support for this project was invaluable and greatly appreciated. Grenville Armitage generously allowed us to share his IP-over-ATM testbed for the fair queuing experiments. We had a number of useful and interesting discussions on the theory of fair queuing with S. Jamal Golestani, and with Deh-phone Hsing on the ATM traffic management standards. Finally, thanks to Carol Patterson and Stu Personick for initiating my interest in an applications-driven approach to ATM traffic management.

## References

- [1] The ATM Forum, *ATM Traffic Management Specification*, Version 4.0, Upper Saddle River, NJ: Prentice Hall, expected 1996.
- [2] ITU-T Rec. I.356, "B-ISDN ATM Layer Cell Transfer Performance," ITU Study Group 13, Geneva, Switzerland, Feb. 1995.
- [3] The ATM Forum, *ATM User-Network Interface Specification*, Version 3.0, Englewood Cliffs, NJ: Prentice Hall, 1993.
- [4] N. Giroux, "Categorization of the ATM Layer QoS Classes and Structure of the Traffic Management Work," Contribution ATM Forum/93-0837, Sept. 1993.
- [5] A. Romanow and S. Floyd, "Dynamics of TCP Traffic over ATM Networks," *IEEE JSAC*, vol. 13, no. 4, May 1995, pp. 633–41.
- [6] D. Grossman, "Definition of VBR Service," Contribution ATM Forum/94-0816, Sept. 1994.
- [7] M. W. Garrett, "ATM Service Architecture: From Applications to Scheduling," Contribution ATM Forum/94-0846, Sept. 1994.
- [8] ITU-T Rec. I.211 [Rev. 1], "B-ISDN Service Aspects," ITU Study Group 13, Geneva, Switzerland, Nov. 1993.
- [9] M. W. Garrett and W. Willinger, "Analysis, Modeling and Generation of

- Self-Similar VBR Video Traffic," *Proc. ACM SigComm*, London, U.K., Aug. 1994.
- [10] M. W. Garrett, "Contributions Toward Real-Time Services on Packet-Switched Networks," Ph.D. dissertation CU/CTR/TR 340-93-20, Columbia University, New York, NY, May 1993 (esp. Ch. 4, "Statistical Analysis of a Long Trace of Variable Bit Rate Coded Video").
- [11] M. Vetterli and K. M. Uz, "Multiresolution Coding Techniques for Digital Video: A Review," *Multidimensional Sys. and Signal Proc.*, Special Issue on Multidimensional Processing of Video Signals, no. 3, 1992, pp. 161-87.
- [12] M. W. Garrett and M. Vetterli, "Joint Source/Channel Coding of Statistically Multiplexed Real Time Services on Packet Networks," *IEEE/ACM Trans. Networking*, vol. 1, no. 1, Feb. 1993, pp. 71-80.
- [13] S. Floyd and V. Jacobson, "Link-Sharing and Resource Management Models for Packet Networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 4, Aug. 1995.
- [14] D. D. Clark, S. Shenker, and L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," *Proc. ACM SIGComm Symp.*, Baltimore MD, Aug. 1992, pp. 14-26.
- [15] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair-Queuing Algorithm," *Proc. ACM SIGComm Symp.*, Austin, TX, Sept. 1989, pp. 1-12.
- [16] A. K. J. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," Ph.D. dissertation, MIT, Cambridge, MA, Feb. 1992.
- [17] E. Hahne, "Round Robin Scheduling for Fair Flow Control," Ph.D. dissertation, MIT, Cambridge, MA, Dec. 1986.
- [18] L. Kleinrock, *Queueing Systems, Volume 1: Theory*, New York: Wiley-Interscience, 1975.
- [19] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," *Proc. ACM SIGComm Symp.*, Philadelphia, PA, Sept. 1990, pp. 19-29.
- [20] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, June 1993, pp. 344-57.
- [21] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case," *IEEE/ACM Trans. Networking*, vol. 2, no. 2, Apr. 1994, pp. 137-50.
- [22] S. J. Golestani, "A Self-Clocked Fair Queuing Scheme for Broadband Applications," *Proc. IEEE Infocom*, May 1994.
- [23] S. J. Golestani, "Network Delay Analysis of a Class of Fair Queuing Algorithms," *IEEE JSAC*, vol. 13, no. 6, pp. 1057-70, Aug. 1995.
- [24] J. C. R. Bennett and H. Zhang, "WF2Q: Worst-Case Fair Weighted Fair Queuing," *Proc. IEEE Infocom*, Apr. 1996.
- [25] J. R. Davin and A. T. Heybey, "A Simulation Study of Fair Queuing and Policy Enforcement," *ACM Comp. Comm. Rev.*, vol. 20, no. 5, Oct. 1990.
- [26] J. W. Roberts, "Virtual Spacing for Flexible Traffic Control," *Int'l. J. Commun. Systems*, vol. 7, 1994, pp. 307-18.
- [27] J. N. Giacobelli *et al.*, "Sunshine: A High-Performance Self-Routing Broadband Packet Switch Architecture," *IEEE JSAC*, vol. 9, no. 8, Oct. 1991.
- [28] H. J. Chao, "A Novel Architecture for Queue Management in the ATM Network," *IEEE JSAC*, vol. 9, no. 7, Sept. 1991, pp. 1110-18.
- [29] H. J. Chao and N. Uzun, "A VLSI Sequencer Chip for ATM Traffic Enforcer and Queue Manager," *IEEE JSAC*, vol. 27, no. 11, Nov. 1992, pp. 1634-43.

### Biography

MARK W. GARRETT [SM] joined Bell Communications Research (Bellcore) in 1984. He has been an active member of the ATM Forum Traffic Management working group for three years. His research has included work on LAN protocols, packet video, traffic analysis, packet scheduling, and quality of service. He received his Ph.D. in electrical engineering from Columbia University in 1993. Dr. Garrett is currently serving as guest editor for an upcoming *JSAC* issue on real-time video in multimedia networks.