The system will then guarantee 5 Mbps all the time, and do its best to provide 10 Mbps when needed, but with no promises.

ABR is the only service category in which the network provides rate feedback to the sender, asking it to slow down when congestion occurs. Assuming that the sender complies with such requests, cell loss for ABR traffic is expected to be low. Traveling ABR is a little like flying standby: if there are seats left over (excess capacity), standby passengers are transported without delay. If there is insufficient capacity, they have to wait (unless some of the minimum bandwidth is available).

Finally, we come to **UBR (Unspecified Bit Rate)**, which makes no promises and gives no feedback about congestion. This category is well suited to sending IP packets, since IP also makes no promises about delivery. All UBR cells are accepted, and if there is capacity left over, they will also be delivered. If congestion occurs, UBR cells will be discarded, with no feedback to the sender and no expectation that the sender slows down.

To continue our standby analogy, with UBR, all standby passengers get to board, but if halfway to the destination the pilot sees that fuel is running low, standby passengers are unceremoniously pushed through the emergency exit. To make UBR attractive, carriers are likely to make it cheaper than the other classes. For applications that have no delivery constraints and want to do their own error control and flow control anyway, UBR is a perfectly reasonably choice. File transfer, email, and USENET news are all potential candidates for UBR service because none of these applications have real-time characteristics.

The properties of the various service categories are summarized in Fig. 5-70.

Service characteristic	CBR	RT-VBR	NRT-VBR	ABR	UBR
Bandwidth guarantee	Yes	Yes	Yes	Optional	No
Suitable for real-time traffic	Yes	Yes	No	No	No
Suitable for bursty traffic	No	No	Yes	Yes	Yes
Feedback about congestion	No	No	No	Yes	No

Fig. 5-70. Characteristics of the ATM service categories.

5.6.5. Quality of Service

Quality of service is an important issue for ATM networks, in part because they are used for real-time traffic, such as audio and video. When a virtual circuit is established, both the transport layer (typically a process in the host machine, the "customer") and the ATM network layer (e.g., a network operator, the "carrier") must agree on a contract defining the service. In the case of a public network, this contract may have legal implications. For example, if the carrier agrees not to lose more than one cell per billion and it loses two cells per billion, the customer's legal staff may get all excited and start running around yelling "breach of contract."

The contract between the customer and the network has three parts:

1. The traffic to be offered.
2. The service agreed upon.
3. The compliance requirements.

It is worth noting that the contract may be different for each direction. For a video-on-demand application, the required bandwidth from the user's remote control to the video server might be 1200 bps. In the other direction it might be 5 Mbps. It should be noted that if the customer and the carrier cannot agree on terms, or the carrier is unable to provide the service desired, the virtual circuit will not be set up.

The first part of the contract is the **traffic descriptor**. It characterizes the load to be offered. The second part of the contract specifies the quality of service desired by the customer and accepted by the carrier. Both the load and the service must be formulated in terms of measurable quantities, so compliance can be objectively determined. Merely saying "moderate load" or "good service" will not do.

To make it possible to have concrete traffic contracts, the ATM standard defines a number of **QoS** (**Quality of Service**) parameters whose values the customer and carrier can negotiate. For each quality of service parameter, the worst case performance for each parameter is specified, and the carrier is required to meet or exceed it. In some cases, the parameter is a minimum; in others it is a maximum. Again here, the quality of service is specified separately for each direction. Some of the more important ones are listed in Fig. 5-71, but not all of them are applicable to all service categories.

The first three parameters specify how fast the user wants to send. **PCR** (**Peak Cell Rate**) is the maximum rate at which the sender is planning to send cells. This parameter may be lower than what the bandwidth of the line permits. If the sender is planning to push out a cell every 4 μsec, its *PCR* is 250,000 cells/sec, even though the actual cell transmission time may be 2.7 μsec.

SCR (**Sustained Cell Rate**) is the expected or required cell rate averaged over a long time interval. For CBR traffic, *SCR* will be equal to *PCR*, but for all the other service categories, it will be substantially lower. The *PCR/SCR* ratio is one measure of the burstiness of the traffic.

MCR (**Minimum Cell Rate**) is the minimum number of cells/sec that the customer considers acceptable. If the carrier is unable to guarantee to provide this much bandwidth it must reject the connection. When ABR service is requested, then the actual bandwidth used must lie between *MCR* and *PCR*, but it may vary

Parameter	Acronym	Meaning
Peak cell rate	PCR	Maximum rate at which cells will be sent
Sustained cell rate	SCR	The long-term average cell rate
Minimum cell rate	MCR	The minimum acceptable cell rate
Cell delay variation tolerance	CDVT	The maximum acceptable cell jitter
Cell loss ratio	CLR	Fraction of cells lost or delivered too late
Cell transfer delay	CTD	How long delivery takes (mean and maximum)
Cell delay variation	CDV	The variance in cell delivery times
Cell error rate	CER	Fraction of cells delivered without error
Severely-errored cell block ratio	SECBR	Fraction of blocks garbled
Cell misinsertion rate	CMR	Fraction of cells delivered to wrong destination

Fig. 5-71. Some of the quality of service parameters.

dynamically during the lifetime of the connection. If the customer and carrier agree to setting MCR to 0, then ABR service becomes similar to UBR service.

CVDT (**Cell Variation Delay Tolerance**) tells how much variation will be present in cell transmission times. It is specified independently for *PCR* and *SCR*. For a perfect source operating at *PCR*, every cell will appear *exactly* 1/*PCR* after the previous one. No cell will ever be early and no cell will ever be late, not even by a picosecond. For a real source operating at *PCR*, some variation will occur in cell transmission times. The question is: How much variation is acceptable? Can a cell be 1 nsec early? How about 30 seconds? *CDVT* controls the amount of variability acceptable using a leaky bucket algorithm to be described shortly.

The next three parameters describe characteristics of the network and are measured at the receiver. All three are negotiable. **CLR** (**Cell Loss Ratio**) is straightforward. It measures the fraction of the transmitted cells that are not delivered at all or are delivered so late as to be useless (e.g., for real-time traffic). **CTD** (**Cell Transfer Delay**) is the average transit time from source to destination. **CDV** (**Cell Delay Variation**) measures how uniformly the cells are delivered.

The model for *CTD* and *CDV* is shown in Fig. 5-72. Here we see the probability of a cell taking time *t* to arrive, as a function of *t*. For a given source, destination, and route through the intermediate switches, some minimum delay always exists due to propagation and switching time. However, not all cells make it in the minimum time; the probability density function usually has a long tail. By choosing a value of *CTD*, the customer and the carrier are, in effect, agreeing, on how late a cell can be delivered and still count as a correctly delivered cell. Normally, *CDV* will be chosen so that, α, the fraction of cells that are rejected for being too late will be on the order of 10^{-10} or less. *CDV* measures the spread in arrival times. For real-time traffic, this parameter is often more important than *CDT*.

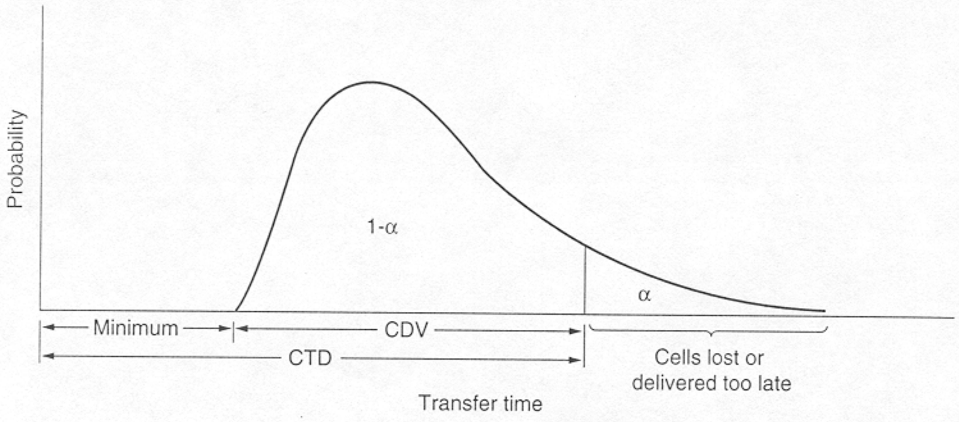

Fig. 5-72. The probability density function for cell arrival times.

The last three QoS parameters specify characteristics of the network. They are generally not negotiable. **CER** (**Cell Error Ratio**) is the fraction of cells that are delivered with one or more bits wrong. **SECBR** (**Severely-Errored Cell Block Ratio**) is the fraction of *N*-cell blocks of which *M* or more cells contain an error. Finally, **CMR** (**Cell Misinsertion Rate**) is the number of cells/sec that are delivered to the wrong destination on account of an undetected error in the header.

The third part of the traffic contract tells what constitutes obeying the rules. If the customer sends one cell too early, does this void the contract? If the carrier fails to meet one of its quality targets for a period of 1 msec, can the customer sue? Effectively, this part of the contract is negotiated between the parties and says how strictly the first two parts will be enforced.

The ATM and Internet quality of service models differ somewhat, which impacts their respective implementations. The ATM model is based strictly on connections, whereas the Internet model uses datagrams plus flows (e.g., RSVP). A comparison of these two models is given in (Crowcroft et al., 1995).

5.6.6. Traffic Shaping and Policing

The mechanism for using and enforcing the quality of service parameters is based (in part) on a specific algorithm, the **Generic Cell Rate Algorithm** (**GCRA**), and is illustrated in Fig. 5-73. It works by checking every cell to see if it conforms to the parameters for its virtual circuit.

GCRA has two parameters. These specify the maximum allowed arrival rate (*PCR*) and the amount of variation herein that is tolerable (*CDVT*). The

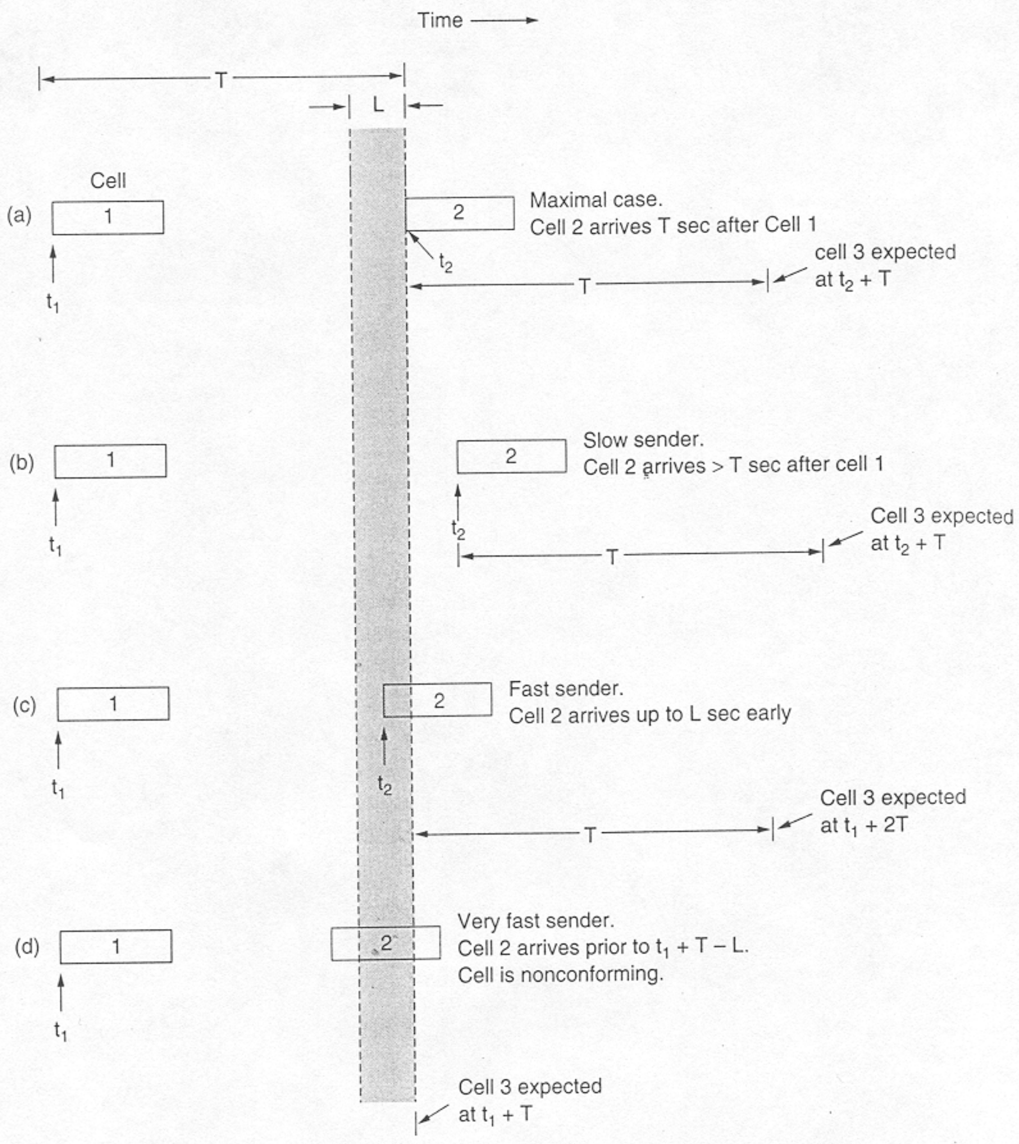

Fig. 5-73. The generic cell rate algorithm.

reciprocal of *PCR*, $T = 1/PCR$, is the minimum cell interarrival time, as shown in Fig. 5-73(a). If the customer agrees not to send more than 100,000 cells/sec, then $T = 10$ μsec. In the maximal case, one cell arrives promptly every 10 μsec. To avoid tiny numbers, we will work in microseconds, but since all the parameters are real numbers, the unit of time does not matter.

A sender is always permitted to space consecutive cells more widely than T, as shown in Fig. 5-73(b). Any cell arriving more than T μsec after the previous one is conforming.

The problem arises with senders that tend to jump the gun, as in Fig. 5-73(c) and (d). If a cell arrives a little early (at or later than $t_1 + T - L$), it is conforming, but the next cell is still expected at $t_1 + 2T$, (not at $t_2 + T$), to prevent the sender from transmitting every cell L μsec early, and thus increasing the peak cell rate.

If a cell arrives more than L μsec early, it is declared as nonconforming. The treatment of nonconforming cells is up to the carrier. Some carriers may simply discard them; others may keep them, but set the *CLP* bit, to mark them as low priority to allow switches to drop nonconforming cells first in the event of congestion. The use of the *CLP* bit may also be different for the different service categories of Fig. 5-69.

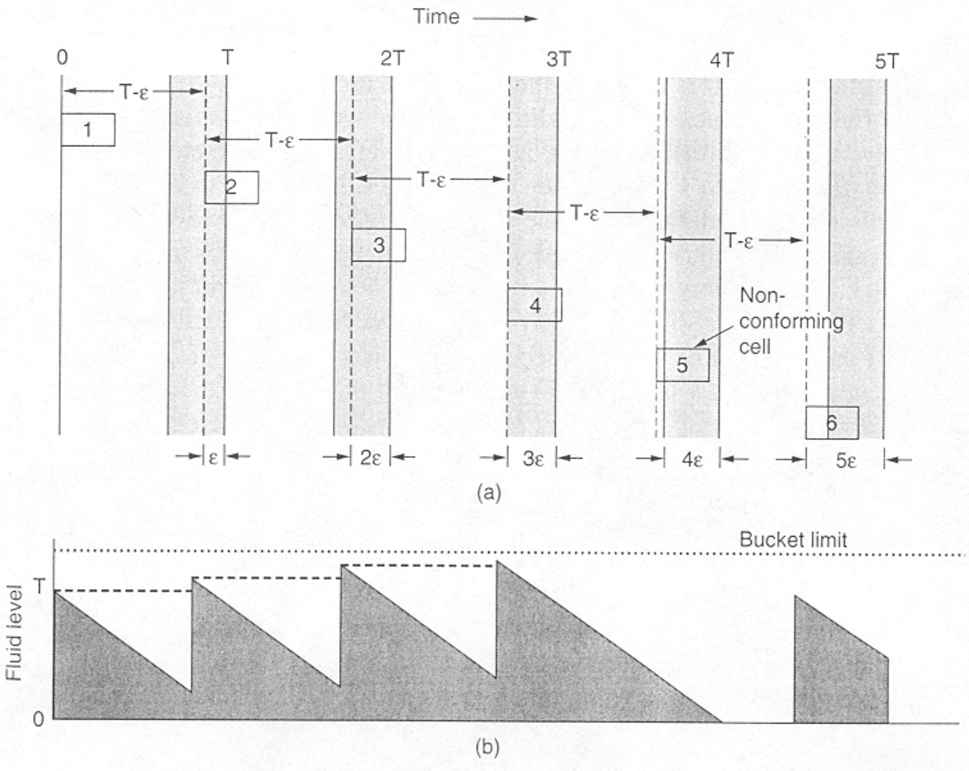

Fig. 5-74. (a) A sender trying to cheat. (b) The same cell arrival pattern, but now viewed in terms of a leaky bucket.

Now let us consider what happens if a sender tries to cheat a little bit, as shown in Fig. 5-74(a). Instead of waiting until time T to send cell 2, the sender transmits it a wee bit early, at $T - \varepsilon$, where, say, $\varepsilon = 0.3L$. This cell is accepted without problems.

Now the sender transmits cell 3, again at $T - \varepsilon$ after the previous cell, that is, at $T - 2\varepsilon$. Again it is accepted. However, every successive cell inches closer and closer to the fatal $T - L$ boundary. In this case, cell 5 arrives at $T - 4\varepsilon$ $(T - 1.2L)$ which is too early, so cell 5 is declared nonconforming and is discarded by the network interface.

When viewed in these terms, the GCRA algorithm is called a **virtual scheduling algorithm**. However, viewed differently, it is equivalent to a leaky bucket algorithm, as depicted in Fig. 5-74(b). Imagine that each conforming cell that arrives pours T units of fluid into a leaky bucket. The bucket leaks fluid at a rate of 1 unit/μsec, so that after T μsec it is all gone. If cells arrive precisely every T μsec, each arriving cell will find the bucket (just) emptied, and will refill it with T units of fluid. Thus the fluid level is raised to T when a cell arrives and is reduced linearly until it gets to zero. This situation is illustrated in Fig. 5-74(b) between 0 and T.

Since fluid drains out linearly in time, at a time t after a cell arrives, the amount of its fluid left is $T - t$. At the time cell 2 arrives, at $T - \varepsilon$, there are still ε units of fluid in the bucket. The addition of the new cell raises this value to $T + \varepsilon$. Similarly, at the time cell 3 arrives, 2ε units are left in the bucket so the new cell raises the fluid level to $T + 2\varepsilon$. When cell 4 arrives, it is raised to $T + 3\varepsilon$.

If this goes on indefinitely, some cell is going to raise the level to above the bucket capacity and thus be rejected. To see which one it is, let us now compute what the bucket capacity is. We want the leaky bucket algorithm to give the same result as Fig. 5-74(a), so we want overflow to occur when a cell arrives L μsec early. If the fluid left requires L μsec to drain out, the amount of fluid must be L, since the drain rate is 1 unit/μsec. Thus we want the bucket capacity to be $L + T$ so that any cell arriving more than L μsec early will be rejected due to bucket overflow. In Fig. 5-74(b), when cell 5 arrives, the addition of T units to the 4ε units of fluid already present raises the bucket level to $T + 4\varepsilon$. Since we are using $\varepsilon = 0.3L$ in this example, the bucket would be raised to $T + 1.2L$ by the addition of cell 5, so the cell is rejected, no new fluid is added, and the bucket eventually empties.

For a given T, if we set L very small, the capacity of the bucket will be hardly more than T, so all cells will have to be sent with a very uniform spacing. However, if we now raise L to a value much greater than T, the bucket can hold multiple cells because $T + L \gg T$. This means that the sender can transmit a burst of cells back-to-back at the peak rate and have them still accepted.

We can easily compute the number of conforming cells, N, that can be transmitted back-to-back at the peak cell rate ($PCR = 1/T$). During a burst of N cells, the total amount of fluid added to the bucket is NT because each cell adds T. However, during the maximum burst, fluid drains out of the bucket at a rate of 1 unit per time interval. Let us call the cell transmission time δ time units. Note that $\delta \leq T$ because it is entirely possible for a sender on a 155.52 Mbps line to agree to send no more than 100,000 cells/sec, in which case $\delta = 2.73$ μsec and $T = 10$ μsec. During the burst of N cells, the amount of drainage is $(N-1)\delta$ because drainage does not start until the first cell has been entirely transmitted.

From these observations, we see that the net increase in fluid in the bucket during the maximum burst is $NT - (N-1)\delta$. The bucket capacity is $T + L$. Equating these two quantities we get

$$NT - (N-1)\delta = T + L$$

Solving this equation for N we get

$$N = 1 + \frac{L}{T - \delta}$$

However, if this number is not an integer, it must be rounded downward to an integer to prevent bucket overflow. For example, with $PCR = 100{,}000$ cells/sec, $\delta = 2.73$ μsec, and $L = 50$ μsec, seven cells may be sent back-to-back at the 155.52 Mbps rate without filling the bucket. An eighth cell would be nonconforming.

The GCRA is normally specified by giving the parameters T and L. T is just the reciprocal of PCR; L is $CDVT$. The GCRA is also used to make sure the mean cell rate does not exceed SCR for any substantial period.

In this example we assumed that cells arrive uniformly. In reality, they do not. Nevertheless, the leaky bucket algorithm can also be used here, too. At every cell arrival, a check is made to see if there is room in the bucket for an additional T units of fluid. If there is, the cell is conforming; otherwise it is not.

In addition to providing a rule about which cells are conforming and which ones are not, the GCRA also shapes the traffic to remove some of the burstiness. The smaller $CDVT$ is, the greater the smoothing effect, but the greater the chance that cells will be discarded as nonconforming. Some implementations combine the GCRA leaky bucket with a token bucket, to provide additional smoothing.

5.6.7. Congestion Control

Even with traffic shaping, ATM networks do not automatically meet the performance requirements set forth in the traffic contract. For example, congestion at intermediate switches is always a potential problem, especially when over 350,000 cells/sec are pouring in on each line, and a switch can have 100 lines. Consequently, a great deal of thought has gone into the subject of performance and congestion in ATM networks. In this section, we will discuss some of the approaches used. For additional information, see (Eckberg, 1992; Eckberg et al., 1991; Hong and Suda, 1991; Jain, 1995; and Newman, 1994).

ATM networks must deal with both long-term congestion, caused by more traffic coming in than the system can handle, and short-term congestion, caused