Abstract

In recent years there has been an emerging interest in using ATM for wireless transmissions. Because ATM is primarily designed for an essentially error-free environment, in the wireless context the sources of errors and their consequences must be thoroughly understood.

While this concern is valid in any network, it takes on a new, more central role in the wireless environment, where error bursts are expected to be a very significant source of degradation. In this article we examine the impact of burst errors on the direct transfer of ATM cells over a wireless channel, taking into account the cell framing format as well as the cell delineation and synchronization mechanism.

We also investigate the consequences of interleaving. We find that fragmenting the data in cells could result in a much higher error rate as seen by the higher layers relative to the bit error rate on the raw channel. We assess the adequacy of HEC-based cell delineation and deduce the most appropriate values of the parameters of the cell delineation algorithm for the wireless channel.

On the Impact of Burst Errors on Wireless ATM

Michele Zorzi, Università di Ferrara
Ramesh R. Rao, University of California at San Diego

Most communication systems are prone to errors due to various physical impairments that occur on the channel. Wireless asynchronous transfer mode (ATM) will be no exception. Even when ATM is supported over relatively noiseless fiber optic channels, provisions are made for the detection, monitoring, and recovery of errors that occur on the physical channel. Because ATM is primarily designed for a rather benign error-free environment, in the wireless context the sources of errors and their consequences must be thoroughly understood.

Error correcting codes have traditionally been developed to overcome or reduce the impact of these errors. It is usually inefficient to attempt to provide a very high degree of protection, and therefore some amount of residual errors pass through undetected. In applications such as voice, these residual errors simply result in speech quality degradation which is designed to be within the range of tolerance. In contrast, applications such as data or lossless image transfers can use retransmission to overcome residual errors. Thus, the residual error process is very relevant to the effective design of higher-layer encoding algorithms for image, video, and other mixed media transmission.

Whenever data is transferred in blocks, standard framing structures are used to define the position of the payload and other control overhead. As a consequence, errors in some of the bit positions may be more detrimental than others. This also leads to the assignment of greater error protection to some bits. Since there is no definite agreement on framing formats for wireless ATM, in this article we assume that cells are directly transmitted on the physical channel, without any mapping between the bitstream and the transmitted signal. Results obtained for the burst error channel in this situation are expected to be representative of the performance of a real system. In addition to framing, physical layer functions must be able to delineate cell boundaries. One especially efficient way of implementing this is to exploit the header error control (HEC) bits to search for and lock onto the boundaries of a cell. ATM standards currently specify a particular algorithm for establishing synchronization and monitoring it on an ongoing basis. This process involves tracking the correctness of successive cells with triggers for declaring loss or establishment of sync. This mechanism is thus affected by channel error events that are even more spread out than the impact of channel errors on framing. One may view this process as a natural extension of the error check mechanisms inherent to individual ATM cells.

In this article we examine the impact of burst errors on the direct transfer of ATM cells, taking into account the cell framing format and the consequences of interleaving techniques. We consider metrics such as cell loss rates, cell error rates, and retransmissions to overcome residual errors. Thus, the residual error process is very relevant to the effective design of higher-layer encoding algorithms for image, video, and other mixed media transmission.

Our analysis explicitly models the residual correlation, not captured by the approach in [3]. In the wireless context, the authors argue that both mechanisms are severely affected by burst errors, and that interleaving must be used in this case. An approach of ad hoc forward error correction (FEC) techniques to supplement HEC (which is considered too weak by the authors) has been proposed. Again, only the iid error case is studied. Finally, the intracell interleaving technique (considered in the following), where bit scrambling is performed within a cell (as opposed to across cells), has been proposed in [5]. Our analysis explicitly models the residual correlation, not captured by the approach in [5].

After describing the channel model in the next section, we address the issues of framing and HEC performance, respectively.

The Channel Model

The error characteristics of the physical channel influence the payload error process significantly. Accurate models for the channel are therefore crucial. Two broad approaches can be used to obtain these models. The first approach is to analyze measured data in order to derive a suitable model. The primary limitation of this approach is the relative paucity of comprehensive measurements. Often, even when measurements are gathered there is a degree of averaging that is done, and thus the recorded data fails to reveal correlation information.
The second alternative is to use parametrically defined analytical models. Such an approach reduces the uncertainty to a smaller set of parameters and makes it possible to study the consequences of the model in an analytically well-defined manner. The drawback is, of course, the need to ensure that the choice of analytical model is reasonable.

One particular analytical model that is useful in studying burst error channels is the Gilbert channel model [6, 7]. In this model, the physical channel is modeled as assuming one of two states (a "good" state, 0, and a "bad" state, 1), each having an associated error probability. The transitions between these two states occur at discrete time instants, so the channel is assumed to stay in a given state for an integer multiple of some time unit, which can be the duration of a bit, or a symbol [8], or even a packet [9]. Let \( y_n = 0 \) if the channel is good during the \( n \)th time unit, and \( y_n = 1 \) otherwise. Then \( y_n \) is a binary Markov process with transition matrix

\[
P = \begin{pmatrix}
P_{00} & P_{01} \\
P_{10} & P_{11}
\end{pmatrix}
\]

The length of a burst (e.g., the number of time units the channel stays in the bad state) is a geometric random variable with mean \( 1/P_{10} \) and similarly for the time the channel is in good state (with mean \( 1/P_{00} \)). The steady-state probability of the channel being in bad state is given by

\[
p_1 = \frac{P_{00}P_{11}}{P_{00}P_{11} + P_{01}P_{10}}
\]

If \( P_e(i) \) is the error probability given that the channel is in state \( i \), we can find the steady-state error rate as [8]

\[
P_e = P_e(0)P_{10} + P_e(1)P_{01} + P_e(1)P_{01}.
\]

According to [10, Ch. 6], we can keep track of events that are associated with transitions by "tagging" the transition diagram of the Markov chain appropriately (i.e., by labeling the edges of the chain flow graph with some transfer functions). Therefore, in addition to the two-state Markov chain that is needed to track the channel status, we can build an analytical structure to keep track of events (in particular, errors).

The key quantity of interest in this context is the error probability of having \( k \) errors in \( n \) symbols, defined as the probability that exactly \( k \) bits are in error in \( 0, \ldots, n-1 \) and \( y_n = j \), given that \( y_0 = i \). This quantity can be computed recursively, as thoroughly described in [11].

**The Effect of Framing**

Let us now focus on a generic framing strategy, where packets are composed of \( N_1 \) header bits and \( N_2 \) payload bits. We do not make any assumption here about how the header bits and payload bits are actually transmitted, since we will consider some possible strategies in the following subsections. In any event, it is clear that the Markov character of the channel allows us to compute the joint probability of having \( k_1 \) header errors and \( k_2 \) payload errors, \( P(k_1, k_2) \), by simply applying the total probability theorem appropriately once the framing format is specified. It is also worth noting that the exact number of errors contained in the header is often of no interest, since a cell whose header is in error is just discarded. On the other hand, detailed information about the error distribution within the payload is very useful in some applications, such as whenever the payload is passed up to the higher layers if a cell has a correct header, irrespective of whether or not the payload itself is correct. In such cases, a quantity of some interest is the joint probability of header success/failure and the number of payload errors.

Some specific formats which can be used to map the packets onto the physical channel will be considered in the following subsections.

**Direct Cell Transfer with No Interleaving**

We start considering the case in which ATM cells are placed on the physical channel as they are. In this case, an \( N_1 \)-bit header is followed by an \( N_2 \)-bit payload, and the joint distribution of \((\beta_1, k_2)\) can be found as

\[
P[\beta_1 = 0, k_2 = k] = \sum_{i=0}^{1} \sum_{j=0}^{N_1} \sum_{z=0}^{k} \pi_i \phi_{in}(i, N_1) \phi_{out}(k, N_2).
\]

**Direct Cell Transfer with Interleaving**

In the previous subsection, packets were assumed to be transmitted on the channel one after the other. However, in many communications systems the error burstiness of the channel is mitigated through the use of interleaving.

Let \( d \) be the interleaving depth and \( N \) the packet length. Then, prior to transmission, the packets are written as rows of a \( d \times N \) matrix, which is read by column to obtain the data stream which is actually sent on the channel. At the receiving end, the dual operation is performed and the original order of the data restored. This operation has the effect of mitigating the effect of the channel memory and breaking error bursts so that error correcting codes can be more effectively used, while requiring some memory and causing some delay. Since we are interested here in the first-order statistics (e.g., average BER or packet loss probability) rather than the joint statistics between successive packets [11], the extension of the above results to the case with interleaving is straightforward. In fact, the effect of interleaving is just to induce a deterministic sampling of the channel, obtained by taking every \( d \)th symbol. This results in another Gilbert channel, with transition probability matrix \( P' \), and the above analysis still applies.

Note that interleaving is usually introduced in order to disperse errors, since this is commonly believed to be a means of improving the overall performance due to the better behavior of error correcting codes. However, there is a price to be paid in terms of increased transmission delay, since the matrix must be filled before transmission can start. There is therefore a trade-off between error rates and delay, and specific quality of service (QoS) requirements may lead to favoring either quantity.

---

1. In this model, aggregation of ATM cells into a larger packet leads to the same result if each of these new packets now comprises \( d \) rows of the interleaving matrix.
2. We will see in the results section that this is not necessarily true.
3. The one just described is usually referred to as a block interleaver. Convolutional interleavers [12] have slightly better delay performance, but basically suffer from similar problems.
whose distribution is given by

Intracellular Interleaving

More sophisticated forms of interleaving can be envisioned (e.g., where bits within a single packet are scrambled). For example, after appending an 8-bit cyclic redundancy check (CRC) to the four control bytes, one could distribute the resulting 5-byte header throughout the packet, thereby reducing correlation between errors on the header bits [5]. This strategy has the advantage of achieving minimum delay, since interleaving is performed within a packet, as opposed to classic interleaving, where one has to collect $d$ packets before being able to transmit (or retrieve) the whole interleaving matrix.

Strategies of this sort can be studied by extending the analysis in the following way. Assume that a packet is transmitted as a group of header bits (possibly just one), followed by a group of payload bits, in turn followed by another group of header bits, and so on. By using the Markov property and applying the total probability theorem as many times as needed, the above analysis can be extended to this case.

As an example, let us consider a strategy where header bits are uniformly spread throughout the cell. For analytical convenience, we assume that the number of payload bits is an integer multiple of the number of header bits, and define $L = N_2/N_1$. Note that this strategy is reasonable for a standard ATM environment where the HEC cannot correct two or more errors, so uniform spreading of the header bits appears to be a reasonable choice. Also, note that for the standard ATM cell format, $L$ is not an integer: the extension of the analysis is conceptually straightforward and will not be addressed here.

Therefore, the ATM cell will look like $N_1$ subcells, each composed of one header bit and $L$ payload bits. As per the above analysis, it is possible to find the joint statistics of the number of errors in subcell $n$ as

$$\theta_j(k_o, l_o) = \sum_{n=0}^{\infty} \phi_{nn}(k_o,1) \phi_{n0}(l_o, L).$$

The transmission of an ATM cell can therefore be seen as the transmission of $N_1$ subcells. Another semi-Markov model can track the cumulative number of errors corresponding to the consecutive transmission of $N_1$ subcells: two states are adequate to track the channel being good or bad, whereas the transitions now imply a number of header and payload errors whose distribution is given by $\theta_j(k_o, L)$ above. Recursive relationships, although a little more complex than those for $\theta_j(k_o, l_o)$ above, since now multiple errors may occur on a single transition, can be established and solved to obtain the joint distribution of header and payload errors, which contains all the information needed for cell performance characterization.

Higher-Order Statistics

In some instances it may be of interest to consider statistics of order higher than one. In fact, as discussed in [11, 13], in the presence of correlation between errors in successive packets the marginal statistics does not allow adequate characterization of the error process and accurate performance prediction. Although the specific issue is not addressed here in detail, the above analysis can be extended to the computation of higher-order statistics, following [11].

Performance Metrics

The analysis outlined makes it possible to compute the joint statistics of the number of errors in the header and in the payload. This information is all that is needed in order to evaluate a number of performance metrics which are commonly used [1]. For the sake of concreteness, we refer in the following to an ATM environment, where packets are ATM cells composed of a 5-byte header with a single-bit error correction capability and a 48-byte payload with no error correction capability. Other possibilities could be considered, where some error control in the payload is provided by the ATM adaptation layer (AAL).

The reference performance measure is the raw error rate on the channel, that is, the average BER one would achieve by just transmitting a bitstream on the channel. This is obviously equal to $e$, as given in Eq. 3.

A very common index is the cell loss rate (CLR), defined as the probability that the header is in error, that is,

$$\text{CLR} = \sum_{k_2=0}^{N_2} \xi(1, k_2),$$

where we introduced the function $\xi(\beta, k) = \mathbb{P}[h_1 = \beta, k_2 = k]$, $\beta = 0, 1, k_2 = 0, ..., N_2$. We should note that in wireline ATM over fiber cell loss is usually due to buffer overflow at the network nodes, whereas bit errors are so rare that a cell loss due to unrecoverable header errors (and consequent mis-routing) has a much smaller probability and can effectively be neglected. This is not the case in ATM over wireless links, where the poor channel quality has a much bigger impact and is expected to be the dominant factor. Therefore, in this context we identify the event of a cell loss with an unrecoverable header error (i.e., two or more bit errors in the header). Note that in order to compute this probability we do not need the statistics of the payload errors, since the payload plays no role in this case. Also, we compute the probability that a cell is lost in a single transmission attempt. Retransmission strategies, not included in this analysis, could be used to recover some of these errors, effectively reducing the probability that a cell does not reach its destination, at the expense of increased bandwidth and transmission delay.

Other higher-level performance metrics, related to how errors are handled, can be computed as well. For example, one might want to know the BER seen by the application above the AAL (payload error rate). For the purpose of discussion, let us assume that if a cell header is correct, all payload bits are delivered to the higher layer, whereas if the header is in error, $\omega N_2$ bit errors result on average. The value

$^1$ Note that the methodology can easily be extended to other cases as needed (e.g., when header compression is used).
of \( \omega \) depends on the AAL strategy used; for example, if nothing is passed up (the case considered in the numerical results) we have \( \omega = 1 \), whereas if random bits are generated we have \( \omega = 0.5 \) (another possibility, not considered here, would be to declare \( N_2 \) erasures). From the above analysis, the average BER of the stream at the output of the AAL is given by

\[
P_{\text{BER}} = \sum_{k_2 = 0}^{N_2} \left[ \frac{k_2}{N_2} \xi(0,k_2) + \omega \xi(1,k_2) \right].
\]

Similar expressions can be found according to how errors are handled by the AAL.

Finally, another index of interest is the BER of the correctly routed cells, that is, the average number of errors in the payload given that the header is correct (cell error rate, CER), which is easily found to be

\[
C_{\text{ER}} = \frac{\sum_{k = 0}^{N_2} k \xi(0,k)}{N_2}.
\]

The analysis presented here can be applied to other performance indices and other framing schemes as well.

### Numerical Results

Numerical results can be obtained for the performance metrics introduced in the previous subsections. According to the model introduced in the second section, we assume a Gilbert-Elliott model \([6, 7]\) for the channel errors. Results obtained for wireless fading channels \([14]\) have been presented in \([15]\), and are qualitatively similar to the ones shown here.

#### No Interleaving

Let us consider the case of no interleaving first. Figure 1 shows CLR, CER, and PBER vs. the burst length of errors in the channel, \( L_{\text{burst}} \), for an average BER \( \epsilon = 0.001 \) (\( \omega = 1 \) will be used throughout). Most results turn out to be proportional to \( \epsilon \), so other values just result in the same curves being shifted appropriately, and will not be presented explicitly. The raw channel BER is also plotted for reference. It can be seen that the average CLR can be substantially larger (up to one order of magnitude) than the raw BER, especially for short bursts. On the other hand, the CER (i.e., the average BER given a correct header) is generally smaller than the raw BER, since the payload following a correct header is likely to contain fewer errors than in the unconditional case. This effect is almost absent for short bursts, where the channel memory is rapidly lost, whereas it is significant for channels with long memory. Finally, we note that the number of errors passed onto the higher layers is dominated by the cell loss events. This can be seen by noting that

\[
P_{\text{BER}} = \omega \text{CLR} + \text{CER}(1 - \text{CLR}) \approx \omega \text{CLR} + \text{CER},
\]

and that usually \( \text{CLR} \gg \text{CER} \). This leads to the observation that, for the case under consideration, the simple fact of having the data fragmented in cells has a big impact on the error rate seen by the higher layers, which may be substantially larger than the BER on the raw channel. In the case of independent channel errors this behavior can be observed for high values of BER, whereas for very reliable channels the CLR decreases rapidly so that \( \text{CLR} \ll \text{CER} \), and PBER and BER are essentially the same.

#### Interleaving

The effect of classic interleaving is essentially to reduce the burst length. Therefore, the CLR will generally increase (Fig. 1), due to the fact that standard HEC protec-

*Figure 2.* a) Average CLR; b) average CER; c) average PBER; vs. average burst length (in bits) for a Gilbert channel with BER \( \epsilon = 10^{-3} \), packet size \( N = 424 \), and standard HEC. No interleaving, classic interleaving with \( d = 10 \), and intracell interleaving compared.
Turning now to the CER (Fig. 2b), we note that the intracell interleaving scheme guarantees substantially better performance. This is due to the fact that, unlike in classic interleaving, the header bits and payload bits are mixed together, and knowing that the header bits are correct makes the error probability in the payload very small. On the other hand, classic interleaving is again harmful, unless the depth is sufficiently large (resulting in delays and memory requirements) and/or a powerful error correction code is used (which would require modifying the standard AAL and involve considerable decoding complexity).

Finally, the best PBER is achieved in the case with no interleaving, as shown in Fig. 2c. In fact, from the PBER point of view, having clustered error, where if the header is in error the payload is also in error, is the best scenario, since the payload bits of a cell which suffers multiple header errors are lost anyway. In this view, interleaving strategies are actually harmful, since they weaken the correlation.

In all the above results, the case of classic interleaving was assumed to have an interleaving depth equal to $L + 1$, which makes it directly comparable with intracell interleaving (the two schemes have in fact the same CLR, as already noted). However, it is clear from intuition, and supported by the above results, that the interleaving depth should be chosen according to the burst error statistics. For the sake of discussion, consider the case in which the interleaving depth is proportional to the average length of a burst of errors, more precisely,

$$d = \frac{y}{p_0} + 1,$$

where $y$ is the proportional interleaving parameter. As seen in Fig. 3, the CLR vs. burst length for some values of $y$ is practically a constant in the presence of interleaving, showing that proportional interleaving has the desirable effect of making error performance independent of channel burstiness. Note that for $y = 10$ and 100, the performance coincides with the iid case, which is very favorable for the single-error correction capability of the HEC. On the other hand, for smaller values of $y$, where the residual error correlation is such as to prevent the HEC from being effective, the noninterleaved case (i.e., $y = 0$) generally performs better. This clearly shows once again that, with the weak protection guaranteed by the standard HEC, the interleaving depth is to be chosen very large, leading to large (possibly unacceptable) transmission delay.

Another perspective is given in Fig. 4, where the CLR is plotted vs. $y$ for some values of the burst length. Once again, the iid case is a constant and represents a best case. The curves for correlated channel exhibit an interesting behavior. Of course, since $y$ is smaller than $p_{00}$, there is no interleaving ($d = 1$ in Eq. 11), and this gives rise to the presence of floor values on the left side of the graph. As $y$ reaches $p_{00}$ some interleaving is used, and the performance correspondingly becomes worse, since such a limited degree of interleaving is not able to adequately decorrelate the channel errors. This behavior is observed up to some value of $y$, where finally the interleaving depth is large enough and the performance improves again, until for large $y$ it approaches the iid case. Note that the value of $y$ which guarantees “almost iid” performance may lead to large memory and delay requirements (especially for slow channels). On the other hand, if the channel is not too fast, the noninterleaved case is only slightly outperformed by the perfectly interleaved one, which suggests that, for some applications, not using interleaving at all might be a better choice.

**A Summary of Results on Framing**

In this section we examined the impact that errors occurring due to physical channel impairments have on ATM cell transfer. For the case under consideration, fragmenting the data in cells has an adverse impact on the error rate seen by the higher layers. In the case of independent channel errors, this behavior is observed for high values of BER, whereas for very reliable channels the CLR decreases rapidly, and PBER and BER are essentially the same. We find that the intracell interleaving scheme guarantees substantially better performance. This is due to the fact that, unlike in classic interleaving, the header bits and the payload bits are mixed together, and knowing that the header bits are correct makes the error probability in the payload of correctly routed cells very small. On the other hand, classic interleaving is harmful unless the depth is sufficiently large or a powerful error correction code is used (which would require modifying the standard AAL and increase decoding complexity). For the case when the interleaving depth is proportional to the average length of a burst of errors we found that, with the weak protection guaranteed by the standard HEC, the proportionality factor, $y$, is to be chosen very large. The best PBER is achieved with no interleaving. From the PBER point of view, having clustered error is the best, since the payload bits of a cell which suffers multiple header errors are lost anyway. In this case, interleaving is harmful, since it weakens the correlation.
Even though it is impossible to make definitive statements about the appropriateness of using interleaving, since this depends on the specific environment (channel statistics) and QoS requirements, the results shown clearly suggest that the use of interleaving should not be taken for granted. In this view, radios where the interleaving block can be turned off or its depth can be selected as needed would be very valuable.

**HEC Cell Delineation**

Let us now consider the process of cell delineation, another physical layer mechanism that is likely to be affected by channel errors. Cell delineation is performed on an ongoing basis to ensure integrity of data even in the fiber optic domain. Its significance can only grow in wireless applications. As with framing, we shall assume here that ATM cells are transmitted directly over a wireless channel that exhibits Markovian errors. Our goal here is to uncover sensitivity of various synchronization metrics to the wireless channel parameters.

Cell delineation, described in detail in [1, p. 68] and [2], is based on the observation that in the absence of errors the header CRC provides a means to verify correct cell alignment. In fact, in the presence of correct sync the CRC matches most of the time, whereas with incorrect alignment it does so only infrequently. The alignment search procedure then starts by checking the CRC on a 40-bit window starting at every bit in the sequence. As soon as a match is detected, the system assumes the correct alignment has been found, and stops checking every bit, instead using a presync mode in which it checks every slot to make sure that the CRC check was in fact due to true alignment and not to a random combination of data which happened to mimic it. After a number of consecutive checks (sufficiently high to make the probability of a false alignment small enough), correct sync is declared, and the syn state is entered. Similarly, to avoid giving up correct synchronization because of CRC failure due to channel errors, the sync state is abandoned only after a number of consecutive failures are observed, and after that the hunt mode (bit-by-bit search) is restarted.

**The Synchronization Machine**

The HEC cell delineation mechanism operates as follows (Fig. 5): when searching, it is in the H state; if in that state the HEC check is unsuccessful, the search window slides one bit forward and another check is performed. If the sync pattern is detected, the system moves to the V1 state (verification or presync state). When in state V1 it verifies the acquired synchronism one slot later: if this is true, it moves to state V1+1 (where, by definition, Vm = L0, in which true lock is assumed); otherwise, it goes back to the H state, slides the window one bit forward, and starts searching again. A sync pattern must be observed in consecutive times before being considered correct.

Once the mechanism has entered state L0, it remains in it as long as it verifies the correct sync. Upon detecting a sync failure, it moves to state L1; in general, detecting a sync failure leads from L1 to L1+1 (where, by definition, Lm = H). On the other hand, detecting a correct sync pattern when in L1 always leads back to L0. This means that the lock is not considered lost until a consecutive failures have been detected.

**Synchronization Alignment**

The state-machine corresponding to the cell delineation algorithm is fully characterized by the set of states described earlier. However, in order to track the performance of the scheme, we must jointly keep track of the synchronization algorithm and the true synchronization (which is of course unknown to the sync algorithm itself).

The chain of states characterizing the algorithm state machine is then to be replicated for all possible values of the misalignment, x, between the true and current synchronization references. More specifically, if a slot starts with bit i, the current estimate of the algorithm has i + x as the first bit of that slot. Since here the synchronization only involves determining the slot boundaries, x can assume any integer value between 0 (true sync) and N - 1, with N the packet size.

**The Channel Model**

We assume that the channel state in a given slot is described by a two-state Markov process (Gilbert channel) with transition matrix

\[
P = \begin{pmatrix}
P(B|B) & P(G|B) \\
P(B|G) & P(G|G)
\end{pmatrix},
\]

where B and G indicate a bad and good channel state, respectively. Note that this model refers to the slot timescale, whereas the model used for the framing study referred to the bit timescale. The relationship between the two processes can be found as in [11]. For convenience, here we choose the packet-level parameters as independent variables.

**The Semi-Markov Model**

The above two processes, algorithm state machine and misalignment, must be tracked together with the Markov channel model. We can describe the whole system by means of a semi-Markov model, as defined in [10, ch. 10]. Transitions occur between states, and the time delay associated with a transition is generally a nonnegative random variable. Such a model can be represented through a transition diagram whose transitions are labeled with the z-transforms of the delay distributions (for a detailed description see [10]). In this case, the variable z represents one slot, that is, a transition labeled z involves a k-slot delay. This representation is useful since it allows manipulation of the transition diagram and elimination of states through flow graph reduction by recomputing the transition functions appropriately. If one is interested in visits to a limited number of states, it is possible to reduce the full transition diagram to one with a smaller number of states. The new diagram still represents a semi-Markov process, but is easier to study in general. No useful information is lost in this case if all states relevant to the quantities of interest are retained.

A full description of the system status in this case can be
provided by means of states of the form $(S, x, c)$, where $S \in \{H, V, j = 1, ..., m - 1, L, j = 0, ..., n - 1\}$ gives the state of the synchronization machine, $x$ gives the phase of the current estimate of the start of the slot with respect to the actual start of the slot, and $c \in \{B, G\}$ gives the channel status in the current slot. Note that the value of $c$ refers to the channel state in the actual slot, which for $x = N - 1$ is the slot which is just ending (i.e., $c$ does not depend on the value of $x$).

More specifically, if the system is in state $(S, x, c)$ in slot $t$, this means that:

- The channel state in slot $t$ is $c$.
- The estimated alignment is $x$ bit periods late with respect to the correct alignment.
- The following events happened previously:
  - If $S = H$, a CRC check failure was detected one bit earlier.
  - If $S = V, j = 1, ..., m$ (with $V = L_0$), successful CRC checks have been observed in slots $t = i, ..., t - 1$ but not in slot $t - 1 - j$.
  - If $S = L, j = 1, ..., n - 1$, the system was in sync in slot $t - j - 1$, and $j$ unsuccessful CRC checks have been observed in slots $t - j, ..., t - 1$.

Before we consider the details of the transition structure, some notation needs to be established. We define $p_y$ as the probability that HEC is successfully checked in the presence of incorrect alignment. Due to the randomness of the bitstream, in an ATM setting this can be assumed to happen with probability $2^{-8}$, regardless of the channel state.\(^5\) Therefore, by this independence, the joint statistics of the one-slot channel evolution and the false-lock check is just given by the product of the channel transition matrix, $P_{c}$, and the false-lock probability, $p_{fl}$, that is,

$$P_1 = p_{fl}P_c.$$  \hspace{1cm} (13)

Similarly, we define $P_2 = (1 - p_{fl})P_c$, which describes the joint statistics of the channel evolution and the HEC check failure when in incorrect alignment. It is assumed that successive HEC checks are independent of each other. This is of course true if they occur in different slots (since they use different sets of bits in this case). On the other hand, if HEC check fails when in hunt mode, a new check is performed one bit later (i.e., all bits but one are the same as before). A precise study of the interaction of these two check operations would be too complex and would involve the details of the code used and keeping memory of the bit values used during the previous attempt, a clearly impossible task. However, the complicated interaction among the bits being used for this operation can be expected to decorrelate sequences with different phases. Therefore, assuming independence between them seems reasonable. This independence assumption has been made in other articles addressing similar problems \cite{2}.

If the alignment is in fact correct (i.e., $x = 0$), we define $p_{c}(e)$ as the probability of a successful HEC check given that the channel is in state $c$. If $e = B$, it is likely that many errors occur, and the probability of a correct HEC check is essentially the same as if the alignment was wrong, i.e., $p_{c}(B) = p_{fl} = 2^{-8}$. On the other hand, if $e = G$, it is very likely that the correct alignment will be detected (i.e., $p_{c}(G) = 1$). As in the previous case, we can define matrices that jointly track the success/failure of the HEC check and the channel status. In this case, the event that HEC is successfully/unsuccesfully checked does depend on the channel state. Therefore, the joint probabilities of the one-slot channel evolution and correct HEC check are given by

$$P_{f} = \begin{bmatrix}
P(B|1)P_{f}(B) & P(G|1)P_{f}(B) \\
P(B|G)P_{f}(G) & P(G|G)P_{f}(G)
\end{bmatrix}. $$  \hspace{1cm} (14)

More compactly, we have

$$P_{f} = \text{diag}(p_{f}(B), p_{f}(G)), \quad P_{c} = P_{f}P_{c}, \quad P = P_{c} - P_{f},$$  \hspace{1cm} (15)

where $P_{f}$ corresponds to a miss of the correct alignment.

This matrix notation of Eq. 15 is advantageous for two reasons. First, it allows us to neglect the channel state in our description and to make it more compact. Second, it allows us to readily extend our analysis to channel models with more than two states \cite{16} (as a matter of fact, all analytical expressions given below would remain exactly the same, and consideration of such a new model would only affect the above matrix definitions). Therefore, in what follows the generating functions with which transitions are labeled are to be treated as matrix functions (in particular, this requires careful handling in some cases since, e.g., product is not commutative in general).

Let us focus first on a specific value of $x$, with $x > 0$. This corresponds to the synchronization mechanism being aligned on the $x$th bit of each slot. The set of states $(S, x, j = 1, ..., m - 1, L, j = 0, ..., n - 1)$ adequately describes the possibilities as to the synchronization machine. Each state here actually represents a number of states, corresponding to the channel description. As in \cite{10}, we account for this by using matrices (instead of scalars) to describe the transition structure between vector states. The following describes all possible transitions that originate from states with a given value of $x > 0$. These transitions may lead to states in the same stage $x$ or to the hunt state in the next bit position $(x + 1) \mod N$. For simplicity of notation, we denote the $H$ state of the next stage by $E$ (denoting exit from stage $x$).

- From $H$, a transition may occur to $V_{j}$ with probabilities $P_{f}$ (successful HEC check) and one-slot delay (i.e., another check is performed in the next slot) or to $E$ with probabilities $(1 - p_{fl})I$ (unsuccessful HEC check) and zero delay (i.e., another check is performed in the same slot, although one bit later). The corresponding transition functions are represented by $P_{2}$ and $(1 - p_{fl})I$, respectively. Since the channel state is tracked on the true slot structure and the phase memory is kept through $x$, single-bit delays are neglected here for simplicity of notation. This is exact as long as we introduce a one-slot delay when going from stage $x = N - 1$ back to stage $x = 0$, to take into account the bit-by-bit delay accumulation.

- From $V_{j}, i = 1, ..., m - 1$, a transition may occur to $V_{j+1}$ with transition functions $P_{2}$ and $(1 - p_{fl})I$, respectively. Note that if an HEC check fails while in verification state, another check is performed just one bit later (i.e., in the same slot, so the delay in our model is zero). Also, recall that $V_{i} = L_{0}$.

- From $L_{j}, j = 0, ..., n - 2$, a transition may occur to $L_{0}$ or $L_{j+1}$ with transition functions $P_{2}$ and $(1 - p_{fl})I$, respectively. (Again, when sync is lost, the next bit position is immediately checked, which corresponds to zero delay in our model.)

From the above description, we can reduce the corresponding flow graph. At first, we retain the states $H, L_{0}$, and $E$. The feedforward structure of the $V_{j}$s can be reduced to two transfer functions, one leading from $H$ to $L_{0}$ and the other leading from $H$ to $E$:

$$\Psi_{HE}(z) = P_{m}z^{m},$$ \hspace{1cm} (16)

$$\Psi_{HE}(z) = (1 - p_{fl}) \sum_{i=0}^{m-1} P_{fl}^{i}z^{i}. $$ \hspace{1cm} (17)

\(^5\) Note that the machine will compute the CRC checksum on four bytes and compare it with the following byte. In the presence of random data (incorrect alignment), the probability that the two match is one in 256 \cite[1, p. 65].
On the other hand, the structure from $L_0$ to $E$ can be shown to have the following transfer function (including the feedback transitions to $L_0$) [10]:

$$\Phi_{XL}(z) = (1 - p_E) [1 - \Phi_{XL}(z)] \frac{1}{1 - \Lambda},$$

where

$$\Phi_{XL}(z) = \sum_{j=0}^{\infty} \frac{[\Gamma_p]^{j+1}}{j!} z^{-j},$$

Finally, the total transfer function associated with the $x$th stage (called a detour in [2]), is given by

$$\Phi(z) = \Phi_{HE}(z) + \Phi_{HL}(z) \Phi_{XL}(z)$$

for all $x \neq 0$.

The description for stage $x = 0$ is very similar to the above, except for the fact that different matrices are to be used. Also, as already mentioned, we must take into account that a transition from stage $x = N - 1$ to stage $x = 0$ involves an additional delay of one slot. This can easily be incorporated by having $E$ for stage $N - 1$ equal to $H'$, from which state $H$ of stage 0 is reached through a further transition with transition function equal to $P_{2x}$ (one-slot channel evolution).

In particular, for states in stage $x = 0$, the following describes all possible transitions:

- From $H$, a transition may occur to $V_1$ or $E$ with transition functions $P_{2x}$ and $(1 - P_{2x})$, respectively.
- From $V_1$, if $i = 1, ..., n - 1$, a transition may occur to $V_{i+1}$ or $E$ with transition functions $P_{2x}$ and $(1 - P_{2x})$, respectively. Note that if an HEC check fails while in verification state, the next position $x = 1$ is checked in the same slot. Also, recall that $P_m = 0$.
- From $L_{ij}$, a transition may occur to $L_0$ or $L_{i+1}$ with transition functions $P_{2x}$ (a successful HEC check brings the system back to lock) and $P_{2x}$, respectively.
- From $L_{i-1}$, a transition may occur to $L_0$ or $E$ with transition functions $P_{2x}$ and $(1 - P_{2x})$, respectively.

**Performance Metrics**

From the Markov characterization of the process, it is possible (at least in principle) to compute any statistical quantity tracked by the model. In this section we will give some details about a few metrics of interest. Others can be computed similarly by appropriately manipulating the chain.

The above description shows how to concatenate stages to form the Markovian transition diagram for the whole process. It is possible to further reduce it in light of the fact that we have the following two elements (we put $x$ as a subscript to status) to obtain the true dimensionality of the state space of interest and to obtain the desired performance metrics, as detailed in the following.

Let $\Psi(z)$ be the transfer matrix of the expanded process just described, which has a total of $2(m + n) + 1$ states. Then the transition matrix of the embedded Markov process is given by

$$M = \Psi(z),$$

and the vector of the average delays associated to state visits is

$$\Delta = D_{1},$$

where

$$D = \lim_{z \to 1} \frac{\Psi(z)}{z},$$

and $1$ is a vector of all ones.

Analogously, let $R$ be a matrix which counts the average reward associated with each transition of the process. The vector of the average reward associated with state visits can be found as

$$\rho = R_{1} = 1.$$

If $\pi$ is the steady-state distribution of the matrix $M$, the steady-state average reward which is earned by the process during one slot is found as

$$\mathbb{E} = \sum_{i} \pi_{i} \pi_{i},$$

Application of the above formula through appropriate definition of $\rho$ allows the computation of a number of metrics of interest.

**The Probability of Being in Sync** — In our model, each transition entering a state in the set $\mathcal{S} = \{L_0, c\}, c = 0, 1, \ldots, n - 1$, $c = B, G$ corresponds to a slot in which the system is rightly assumed to be correctly synchronized. We can then find the steady-state probability that the process is locked to the true synchronization pattern, $P_{sync}$, by defining the $i$th column of $R$ as equal to 1 if $i \in \mathcal{S}$, and zero otherwise.

**The Probability of Correct Packet Reception** — Let us assume that a packet is accepted only when its HEC is correctly detected. Also, let $p_r(c)$ be the probability that the packet payload is successfully decoded when the channel state is $c$. Then all transitions leading to the two states $(L_0, 0, c)$ are to be tagged with $p_r(c)$, if $c'$ is the channel state of the origin of the transition. Again, from the theory of renewal reward processes, the average packet success rate, $P_{succ}$, can be found, and its complement gives the average packet loss rate (due to either payload or header errors).

**The Average Error Rate While in Sync** — Another quantity of interest may be the packet error rate while in sync — the fraction of packets which are lost (due to header or payload errors) while true synchronization is maintained. This rate can be found from the ratio of the average above two quantities. In fact, we have

$$\rho_{ela} = P[error | sync] = 1 - P[succ | sync] = 1 - P_{sync},$$

since in this case $P[succ, sync] = P[succ]$ (i.e., successes can only occur when sync is available).

---

6 Note in fact that sync is gained when entering a state with $L_0$ and is lost when exiting a state with $L_{m-1}$.

7 Note, in fact, that for a transition originating from a state with $c$, $p_r(c')$ is the average number of successful packets associated with that transition.
The Probability of Losing Sync — This is the probability that in a given slot sync is lost. Transitions corresponding to loss of sync are from (\(L_0, 0, G\)) and (\(L_{n-1}, 0, B\)) to states with \(H_1\). By setting the corresponding entries in \(R\) to one and all others to zero, we obtain \(P_s\).

Time in Sync and Time to Regain Sync — Following the analysis in [17], it is possible to compute the probability that the set \(\mathcal{E}\) is entered at time 0 and left for the first time at time \(t\). This gives the probability mass function of the time in sync, which is of the matrix-geometric type (i.e., the dependence on \(t\) is in the exponent of some matrix), whose moments are readily computed.

On the other hand, the time to regain sync cannot be obtained by directly applying the analysis in [17], due to the semi-Markov character of the chain. In principle, the whole probability mass function can still be computed according to the analysis of [10, Ch. 10], although this involves cumbersome recursions which may not be very practical.

In both cases, it is very easy to find the average values of these two quantities, given by

\[
E[T_{\text{sync}}] = \frac{P_{\text{sync}}}{P_s}, \quad E[T_{\text{nost}}] = \frac{1 - P_{\text{sync}}}{P_s},
\]

which can be derived from the fact that the system alternates sync and nonsync periods.

Error-Free Time — Similar results hold for error-free time (i.e., the length of a sequence of correct packets) if instead of the set \(\mathcal{E}\) we just use the two states \((L_0, 0, c)\), along with the appropriate packet success probabilities. In particular, if we assume that packets are always correctly received in good state and always fail in bad state, every successfully received packet corresponds to entering state \((L_0, 0, G)\). The number of consecutively successful packets is then a geometric random variable, whose parameter is directly found from the transition probability from \((L_0, 0, G)\) to itself. Since this probability is given by \(p(G)p_{\text{GC}}\) (i.e., HEC is successful and channel state remains good), we have

\[
P[\text{consecutive successes}] = (1 - p(G)p_{\text{GC}})^{-1} P(G)p_{\text{GC}}^n - 1
\]

and

\[
E[T_{\text{free}}] = \frac{1}{1 - p(G)p_{\text{GC}}}.
\]

Numerical Results

All the results are given for an average packet error rate of 0.01, which can be taken as a typical value on wireless channels, and for a packet size of \(N = 424\) bits, which corresponds to an ATM cell. The other parameters are channel burstiness (measured in average number of consecutive erroneous packets), acquisition threshold, \(m\), and sync loss threshold, \(n\).

Figure 6 reports results for \(P_{\text{sync}}\) and \(P_{\text{succ}}\) vs. the acquisition threshold for \(n = 7\) — the value suggested by the International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) for wireline ATM — and for four values of the burst length. It is interesting to see that both quantities are maximized for \(m = 2\). This indicates that while immediate synchronization is not desirable (since it leads to false lock situations which hurt the performance), choosing \(m > 2\) is also harmful, since an exceedingly conservative choice leads to many wasted slots before sync is declared (i.e., before packets can be accepted). Also, as seen from Fig. 6b, different sensitivity to \(m\) is induced by different degrees of burstiness (this effect will be examined in more detail later). For different values of \(n\), \(m = 2\) still remains the best choice. Results for \(P_{\text{el}}\) and \(P_{\text{sg}}\) are not reported here, since these quantities are basically independent of \(m\) (they correspond to events that occur after sync is acquired).

Figure 7 reports \(P_{\text{sync}}\), \(P_{\text{succ}}\), \(P_{\text{el}}\), and \(P_{\text{sg}}\) vs. the sync loss threshold, \(n\). The acquisition threshold is set at the best value, \(m = 2\). The quantities \(P_{\text{sync}}\) and \(P_{\text{succ}}\) obviously increase with \(n\), since with larger \(n\) it is harder to lose sync. The crossover among curves indicates that for small \(n\) long bursts are better, due to the fact that runs of good slots are also longer. On the other hand, for large \(n\) the opposite is true, since in this case shorter bursts may not lead to sync loss. Similar considerations hold for \(P_{\text{succ}}\). The probability of packet errors while in sync, \(P_{\text{el}}\) (Fig. 7c), is essentially zero for \(n = 1\), since in this case the channel becomes bad sync is lost, and thereafter.

8 The only exception is the case with \(n = 1\), in which \(m = 1\) is best. However, this seems of limited interest, since it basically corresponds to no synchronization algorithm.

9 The only case when this is not true is for \(n = 1\), which results in a high probability of false lock. In this case, smaller \(n\) helps the system leave false lock states quickly. On the other hand, for \(m = 2\) or larger, the false lock events are greatly reduced, and this effect is eliminated.
Figure 7. a) Probability of being in correct sync, $P_{\text{sync}}$, b) probability of correct packet reception, $P_{\text{success}}$, c) probability of packet error while in sync $P_{\text{e|sync}}$, d) probability losing sync, $P_{\text{e|los}}$, vs. sync loss threshold $n$. Average packet error rate 0.01, acquisition threshold $m = 2$, packet size $N = 424$, various values of the average burst length (in packets).

Therefore while it is in sync almost no errors are made,$^{10}$ $P_{\text{e|los}}$ increases with $n$, since for large $n$ sync is retained even in the presence of a number of errors. On the other hand, $P_{\text{e|sync}}$ decreases as burstiness increases since in this case it is less likely to have isolated errors, and most errors will be followed by many more, leading to sync loss. The probability of losing sync is clearly a decreasing function of $n$ (Fig. 7d). Results obtained for different values of the average packet error rate, $e$ (assumed equal to 0.01 in the above results), show that the quantities $1 - P_{\text{sync}}, 1 - P_{\text{success}}, P_{\text{e|sync}},$ and $P_{\text{e|los}}$ scale almost proportionally with $e$; that is, for $e = 0.001$ those quantities would be reduced by about a factor of 10 when compared to the case $e = 0.01$.

Figure 8 shows results for the average time in sync, $E[T_{\text{sync}}]$, and to regain sync, $E[T_{\text{regain}}]$. The average time in sync is obviously insensitive to the acquisition parameter, $m$. On the other hand, as shown in Fig. 8a, it is an increasing function of $n$, whose slope depends on the burstiness. In particular, the increase is faster for short bursts, as is to be expected, since an increase of $n$ by a fixed amount has more impact in this case. Reduced burstiness results in larger time in sync, since short bursts do not lead to sync loss (the opposite may be true if $n$, here equal to 7, is reduced). The average time to regain sync is insensitive to $n$,$^{11}$ whereas its dependence on $m$ is studied in Fig. 8b. Interestingly, $m = 2$ is best for this quantity as well, because for $m = 1$ false lock events dominate, whereas for $m > 2$ it just takes longer to acquire ($E[T_{\text{regain}}]$ shows an almost linear increase with $m$ for $m \geq 2$).

The effect of the channel burstiness is studied in Fig. 9, where the four probabilities $P_{\text{sync}}, P_{\text{success}}, P_{\text{e|sync}},$ and $P_{\text{e|los}}$ are plotted vs. channel burstiness for some values of $m$ and $n$. It is interesting to see that $P_{\text{sync}}$ is an increasing function of the burstiness for $n = 1$, whereas for $n = 2$ this is no longer true, because isolated errors (likely for short bursts) will not lead to sync loss in this case (Fig. 9a). Similar considerations hold for $P_{\text{success}}$ (Fig. 9b). As already observed, increased burstiness reduces $P_{\text{e|sync}}$ (Fig. 9c). Again, $P_{\text{e|los}}$ is insensitive to the values of $m$. As before, an exception is the case of $m = 1$, where $E[T_{\text{regain}}]$ is an increasing function of $n$. Again, this is explained by noticing that false lock events are in this case fairly frequent, and therefore a larger value of $n$ leads to longer times to exit false lock situations. As noted before, this effect becomes negligible as soon as $m = 2$ or larger.

---

$^{10}$ This probability would be exactly zero if we had zero false lock probability and one-to-one correspondence between good channel status and HEC and payload success.

$^{11}$ As before, an exception is the case of $m = 1$, where $E[T_{\text{regain}}]$ is an increasing function of $n$. Again, this is explained by noticing that false lock events are in this case fairly frequent, and therefore a larger value of $n$ leads to longer times to exit false lock situations. As noted before, this effect becomes negligible as soon as $m = 2$ or larger.
Figure 8. a) Average time in sync, \( E[T_{\text{sync}}] \) vs. sync loss threshold, \( n \) for \( m = 2 \); b) average time to regain sync, \( E[T_{\text{reset}}] \) vs. acquisition threshold, \( m \), for \( n = 7 \). Average packet error rate 0.01, packet size \( N = 424 \), various values of the average burst length (in packets).

Figure 9. a) Probability of being in correct sync, \( P_{\text{sync}} \); b) probability of correct packet reception, \( P_{\text{corr}} \); c) probability of packet error while in sync, \( P_{\text{err}} \); d) probability of losing sync, \( P_{\text{lost}} \) vs. average burst length (in packets). Average packet error rate 0.01, packet size \( N = 424 \), various values of the acquisition threshold, \( m \), and of the sync loss threshold, \( n \).
m (as observed in Fig. 7c, $P_{err} = 0$ for $n = 1$). Finally, the behavior of $P_T$ parallels that of $P_{sync}$ with monotonic shape for $n = 1$ but not for $n = 2$ (Fig. 9d).

A Summary of Results on Cell Delineation

From the results we obtained, we can conclude that best performance is achieved for small $m$ (usually equal to 2). For such a value, a large value of $n$ slightly improves $P_{sync}$ and $P_{sync}$ and leads to a considerable improvement of $P_T$ and $E[T_{sync}]$ while substantially degrading $P_{err}$.

It appears that applications that can recover from packet errors (and therefore do not suffer much because of increased $P_{err}$) should choose a larger $n$, whereas applications where more frequent sync losses can be tolerated (while guaranteeing a higher probability of correct packet reception while in sync) should choose small values of $n$. In any event, it is clear that what was suggested for wireline ATM is far from optimal here, especially regarding the acquisition threshold, $m$. In the presence of channel burstiness and high error rate, the application should aggressively lock to whatever appears to be a reasonably correct synchronization, in order to avoid wasting too many slots in the presync state. It also appears reasonable that such an aggressive acquisition strategy be balanced by an aggressive release of the acquired sync, in order to avoid excessive dwelling times in false lock. Except for the case $n = 1$, the former recommendation (aggressive acquisition) seems to be more important, whereas the latter (aggressive release) leads to more modest performance improvements, also depending on channel error burstiness.

Conclusions

In this article we examine the impact of errors occurring due to physical channel impairments on ATM cell transfer. The new element here is the explicit recognition that, depending on the coding, framing, and cell delineation mechanisms used, some bits are more important than others and should be better protected. Conversely, some errors are more detrimental than others. Hence, tracking the exact consequences of a set of bit errors is crucial and at the same time more involved than counting bit errors, but can more accurately reveal their effect on higher layers. Based on the analysis and results presented, we are able to quantitatively assess the performance of wireless ATM connections when standard framing formats and HEC mechanisms are used for synchronization, evaluate the adequacy of such mechanisms, and make specific recommendations about the most appropriate values of the thresholds to be used.

A key conclusion of this study is that the use of interleaving is not always helpful, and that the availability of radios in which it is not hardwired, so its depth can be selected according to channel conditions and application requirements, or it can even be excluded, would be a very valuable asset.

References


Biographies

MICHELE ZORZI [SM'98] (zorzi@ing.unife.it) received a Laurea Degree and a Ph.D. in electrical engineering from the University of Padova, Italy, in 1990 and 1994, respectively. During academic year 1992-1993, he was on leave at the University of California, San Diego (UCSD), attending graduate courses and doing research on multiple access in mobile radio networks. In 1993 he joined the faculty of the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy. After spending three years with the Center for Wireless Communications at UCSD, in 1998 he joined the School of Engineering of the University of Ferrara, Italy, where he is currently an associate professor. His present research interests include performance evaluation in mobile communication systems, random access in mobile radio networks, and energy constrained communications protocols. He currently serves on the Editorial Boards of IEEE Personal Communications and the ACM/SIGMOBILE Journal of Wireless Networks. He is also guest editor for special issues of IEEE Personal Communications (Energy Management in Personal Communications Systems) and the IEEE Journal on Selected Areas in Communications (Multimedia Network Radios).

RAMESH R. RAO [SM'90] (rrao@ucsd.edu) received his Honours Bachelor's degree in electrical and electronics engineering from the University of Madras in 1980. He did his graduate work at the University of Maryland, College Park, receiving an M.S. degree in 1982 and a Ph.D. degree in 1984. Since then, he has been on the faculty of the Department of Electrical and Computer Engineering at UCSD. His research interests include architectures, protocols, and performance analysis of computer and communication networks.