A Strategy for Designing Error Detection Schemes for General Data Networks

Jane M. Simmons, *Member*, *IEEE*

***Abstract*— Any real-world network system is subject to a variety of errors. Error detection mechanisms, such as cyclic redundancy checks (CRCs), are typically included along with the data to prevent errored data from being accepted as error-free by the destination. This paper identifies the fundamental issues involved with providing error protection and specifies guidelines for designing protocols that effectively and efficiently handle errors. A five-step methodology is presented that provides insight into: the order in which errors should be considered when designing an error detection scheme; which types of error detection mechanisms are most effective; and which layer should be responsible for detecting a given type of error.**

***Index Terms*— ATM, CRC, Error Detection, Network Layers, TCP/IP**

I. INTRODUCTION

ANY real-world network system is subject to a variety of errors. Error detection mechanisms, such as cyclic redundancy checks (CRCs), are typically included along with the data to prevent errored data from being accepted as error-free by the destination. In this paper, a five-step methodology is presented that provides insight into: the order in which errors should be considered when designing an error detection scheme; which types of error detection mechanisms are most effective; and which network layer should be responsible for detecting a given type of error.

This paper specifically addresses error detection in networking systems designed with a layered architecture. In such systems, a given layer performs its assigned function, without other layers needing to be aware of the processing details, and passes its results to adjacent layers. Failure to detect an error at one layer of a network may lead to the error propagating to other layers. Error detection mechanisms likely need to be implemented at more than one network layer. However, regardless of the number of error detection fields included, undetected errors are inevitably going to occur in any system. The purpose of the error detection scheme is to reduce the rate of undetected errors to a level acceptable to users.

In general, error detection is accomplished by transmitting redundant information along with the data; the receiver checks for inconsistencies between the received data and this redundant information. For example, transmitting the destination address along with the data helps detect misdirected data; transmitting a length field helps detect lost data. There are many possible error detection fields; which fields to include depends on the underlying error characteristics of the network. It is important that there be protection against all likely error types since the effectiveness of an error detection scheme is measured by the error scenario that results in the most undetected errors. Adding a lot of overhead to greatly decrease the undetected error rate of one particular error scenario may not be efficacious if there is another scenario that yields a much higher rate of undetected error.

Obviously, the many data networks currently deployed include error detection schemes. However, as evidenced by early attempts to design the error protection mechanism for Asynchronous Transfer Mode (ATM), there does not appear to be an established systematic approach to designing such schemes. In this paper, we establish guidelines for designing effective and efficient error detection schemes. Our approach looks at three layers of a network, which we arbitrarily call layers N, N-1, and N-2, where layer N is the highest layer of the three. These three layers need not correspond to actual OSI layers. (For a further description of network layering, see [1,2].) Rather, they are a natural way to divide up some of the basic functions common to most networks. We look at the various error detection options at each layer. Throughout this paper, we assume we have control over the error detection scheme of all three layers.

II. DESCRIPTION OF LAYERS

This section describes the basic functions of each of the three layers of interest. To provide a reference point, we also map the three layers to the corresponding layers in ATM and Transmission Control Protocol/Internet Protocol (TCP/IP). (Note that we are not saying that there is a one-to-one mapping of layers between ATM and TCP/IP; they are based on two different protocol stacks.) As depicted in Figs. 1 and 2, the highest of the three layers is layer N and the data unit exchanged between layer N modules is referred to as a message. We assume that layer N operates end-to-end (i.e., this layer operates on the data only at the source and destination nodes). In ATM, the Convergence Sublayer of the ATM Adaptation Layer (AAL) corresponds to layer N, and the layer N data unit is called a frame. In TCP/IP, layer N

Manuscript received August 13, 2003. This work was performed at the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA. This work was funded by fellowships from the National Science Foundation and Department of Defense.

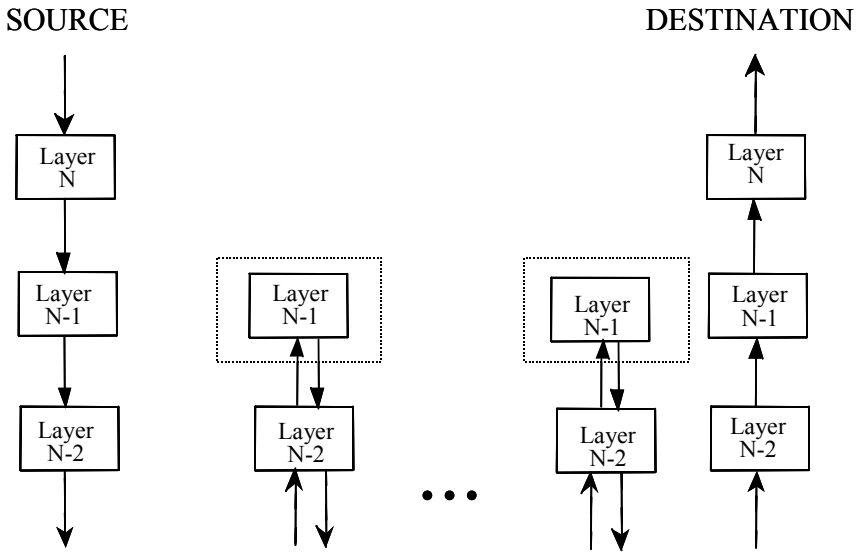

Fig. 1. The layers of interest along the data path. Layer N-1 may or may not operate on the data at the intermediate nodes; i.e., it possibly may fragment the packets further at the intermediate nodes. The message is reconstructed by Layer N-1 only at the destination.

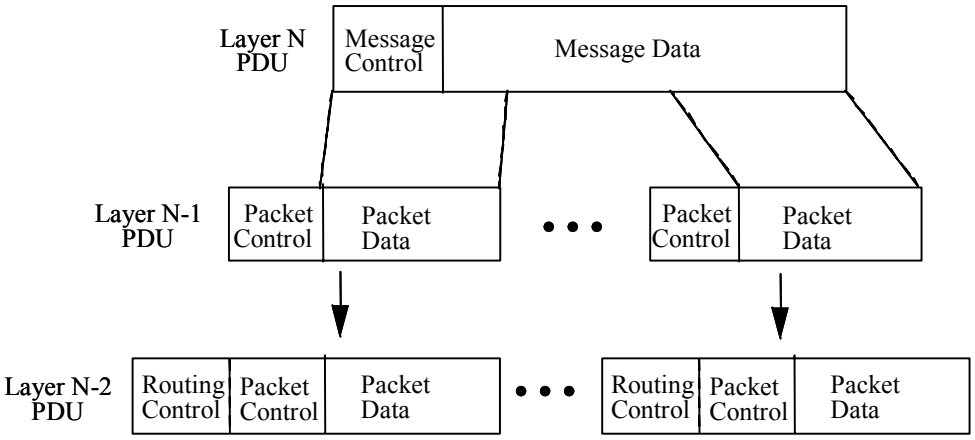

Fig. 2. The data units at each of the three layers (PDU = Protocol Data Unit)

corresponds to the TCP layer, and the layer N data unit is called a segment.

At the source, layer N passes the message down to layer N-1. Layer N-1 is responsible for dividing the message up into smaller units; it typically adds control information to each of these units. We call the layer N-1 data unit a packet. While the initial fragmentation of the message occurs at the source, further division of the packets into smaller packets may be permitted at intermediate nodes along the data path. At the destination, layer N-1 is responsible for merging the packets together to form a message, which is then passed up to layer N. We assume that reconstruction of the message occurs only at the destination. In ATM, the Segmentation and Reassembly Sublayer of the AAL is analogous to layer N-1; the data unit is called a segment. In TCP/IP, the IP layer performs the duties we associate with layer N-1; the IP data unit is called a fragment.

We assume that layer N-2 is responsible for routing the packets to the correct destination. For simplicity, we refer to the layer N-2 data unit as a packet also. However, layer N-2 may add further control information to the layer N-1 packet in order to perform the routing. Layer N-2 is involved with the routing at each node along the data path. The ATM layer performs the routing in ATM systems; the ATM layer data unit is called a cell. In TCP/IP systems, the IP layer performs the routing.

Error detection may take place at any of the three layers, but we will assume that *recovery* from errors is performed end-to-end at layer N and the unit of retransmission is a message (different message control may be included when the message data is retransmitted). Another option would have been to retransmit individual errored packets at layer N-2 or N-1 rather than the entire message. In general, this is expected to be more efficient since fewer overall packets are likely to be retransmitted. However, in typical systems, the

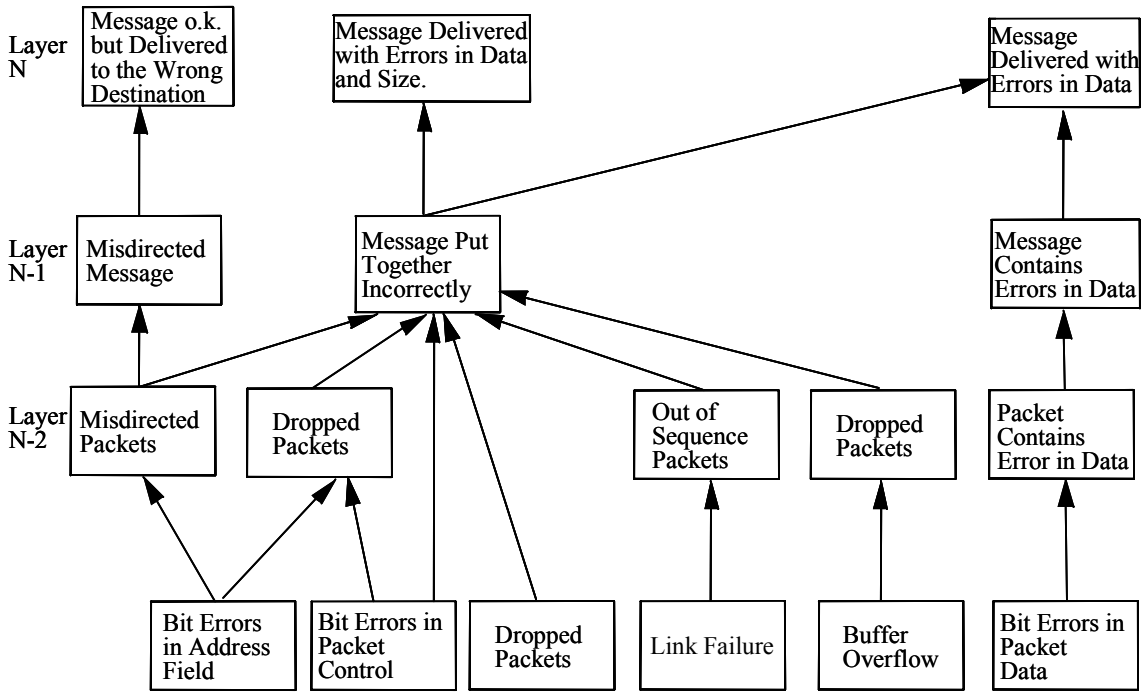

Fig. 3. Diagram of errors at the three layers of interest. The errors shown at a given layer are the errors passed from that layer to the layer above it.

layers that deal with packets do not know what type of data is contained in the packet. For example, if a packet contains video information, it is unlikely that it should be retransmitted if it contains errors. The lower layers of the network, however, would not be aware of the type of content. Normally, the layer that deals with messages, i.e., layer N, is aware of which connections require retransmission of errored data. Thus, we see that end-to-end error recovery at layer N is a reasonable assumption. (Furthermore, this is what is implemented in ATM and TCP.)

Our goal is to have the combination of error detection techniques at all three layers result in a rate of undetected errors at layer N that is below some acceptable threshold. In an actual system, there may be layers above N or layers below N-2 that are also capable of detecting errors. However, we will assume that we cannot rely on layers other than N-2, N-1, and N to detect errors.

Below, we examine the various error scenarios that are relevant at each of the three layers, starting with layer N-2. Figure 3 provides an overview of the errors at the various layers.

A. Error Detection at Layer N-2

We assume that at some layer below N-2, transmissions are subject to random bit errors and burst errors. Also, data can be lost. The four types of errors that are pertinent at layer N-2 and their ramifications are as follows:

1) The packet addressing information has been corrupted due to bit errors or burst errors.

Layer N-2 routes packets; thus, if an error in the packet address is not detected by layer N-2, the packet may arrive at the wrong destination. We refer to this type of error as misdirected data. If layer N-2 detects a corrupt address, it can either attempt to correct the error or it can drop the packet. These options are discussed in Section V.A.

2) The packet data has been corrupted due to bit errors or burst errors.

In order to detect bit errors in the data some sort of redundancy check on the data needs to be added. If the errors are not detected at layer N-2, then layer N-2 passes a packet that has bit errors in it up to layer N-1.

3) A link on which layer N-2 would like to route a packet is down.

If the connection is not aborted, a link failure may cause packets to be lost or delivered out-of-sequence.

4) A buffer at an intermediate node is full.

If a buffer is full, packets are likely dropped. We assume that the node does not send notification to the source or to the destination indicating which packets have been dropped.

B. Error Detection at Layer N-1

Next, we look at the possible errors at layer N-1. The design decisions at layer N-2 affect the frequency of these errors. Recall that the responsibility of layer N-1 is to reconstruct the message at the destination. Undetected errors at layer N-1 result in an errored message being passed up to layer N.

1) Layer N-2 delivers to the destination a packet that does not belong there.

Layer N-1 can add control information to each packet to help detect this error. For instance, it could add an ID field to identify which packets comprise one message.

2) Layer N-2 delivers a duplicate copy of a packet to the destination.

Layer N-1 can add a packet sequence number to help detect a duplicate packet.

3) Layer N-2 fails to deliver a packet to the destination.

Layer N-1 can add a packet sequence number to help detect a lost packet. It could also add an ID field to identify packets belonging to one message, to help detect the case where the "message delimiting" packet is lost and packets from multiple messages are merged together.

4) Layer N-2 delivers a packet out-of-sequence.

Note that this is not necessarily an error event; a datagram delivery service is expected to deliver packets out-of-sequence. Layer N-1 can add a packet sequence number to help detect out-of-sequence packets.

5) Layer N-2 delivers to the destination a packet that has bit errors in the packet data.

In order to detect bit errors in the packet data, some sort of redundancy check on the data needs to be added.

6) Layer N-2 delivers to the destination a packet that has errors in the control information.

For example, there could be an error in a field that aids the destination in reconstructing the message (e.g., an end-of-message flag). Redundancy checks on these types of fields would help detect such errors. Or, there could be errors in the check fields themselves, which may result in other error scenarios going undetected.

It is important to note that layer N-1 is capable of correcting some of the errors enumerated above. If layer N-1 detects an extra packet (either a misdirected packet or a duplicate packet) it can simply drop it. Or, if packets arrive out-of-sequence and some unique ordering scheme is included in the packet control information, then layer N-1 can reorder the packets correctly. In both of these cases, if layer N-1 does not fix the error, then it passes an errored message up to layer N. Layer N may be able to detect the error, but it likely will not be able to correct it. Thus, dealing with certain errors at layer N-1 rather than at layer N may decrease the number of messages that are dropped.

C. Error Detection at Layer N

In this section, we consider the possible error scenarios at layer N.

1) Layer N-1 delivers a message to layer N that has the correct number of bits, but contains at least one bit error.

2) Layer N-1 delivers a message to layer N that has the wrong number of bits.

3) Layer N-1 delivers a message to layer N, where the beginning of the message belongs to one message that was actually sent, and the end of the message belongs to another message that was actually sent.

4) Layer N-1 delivers a message to layer N that is correct except that the message was meant for a different destination.

Layer N has a few error detection options to address these four error scenarios:

a) A redundancy check on the whole message would aid in detecting all of the above error scenarios.

b) A message length check would aid in detecting scenario 2 and possibly scenario 3.

c) A message ID in the beginning and end of the message would help detect scenario 3.

d) Including the destination address as control information in the message would help detect scenario 4.

III. Review Of Cyclical Redundancy Checks

Since CRCs are such an important error detection mechanism, we review their properties below. For a more complete discussion, refer to [3].

Assume that we are using an Extended Hamming Code CRC of length L to check the integrity of a data block of length K. Then, as shown in [4], L should be chosen such that:

$$2^{L-1}-L \geq K+1 \tag{1}$$

Such a CRC is guaranteed to detect all single, double, and triple bit errors that occur in the string of bits comprising the data and the CRC. It is shown in [5] that the number of possible four-bit patterns that can cause an undetected error can be upper bounded by:

$$\binom{L+K}{3}\frac{1}{4} \tag{2}$$

If an error occurs such that the string of bits is random, then an L bit CRC fails to detect the error with probability: (see [4])

$$2^{-L} \tag{3}$$

The CRC also can be used to correct single bit errors, but this increases the probability an error will not be detected. Three bit errors may appear to be a single bit error, and the CRC will 'correct' the error to the wrong value. Thus, three bit errors rather than four can result in an undetected error. Also, if a burst error hits such that the data and CRC bits are random, and if the CRC attempts to correct single bit errors, then the probability the CRC will not detect the burst error is shown in [5] to be:

$$(L + K + 1)\, 2^{-L} \tag{4}$$

A second type of CRC is an Extended BCH Code CRC. This type of CRC requires roughly twice the number of check bits as an Extended Hamming Code CRC, but can detect with certainty up to five bit errors [6]. It is shown in [7] that the number of possible four bit patterns that can cause an undetected error can be upper bounded by:

$$\binom{L+K}{4} \Big/ \binom{6}{4} \tag{5}$$

IV. Degree Of Difficulty In Detecting An Error

Before continuing the analysis, we define the notion of one

error being easier to detect than another. Essentially, the more inconsistencies produced by an error, the greater the diversity of options that can be used to detect the error, and thus, the easier the error is to detect.

This is illustrated in the following example. Assume that we have a system where the beginning packet of a message contains a BEGIN flag, and the final packet of a message contains an END flag. Assume that packets are expected to arrive in sequence. Consider the following two scenarios involving an END packet arriving at the wrong destination.

First, assume the stray END packet arrives immediately after another END packet. Thus, the destination receives an END packet without a corresponding BEGIN packet. In this scenario, the misdirection has occurred in such a way as to produce an invalid message. Thus, the error should be easy to detect.

Second, assume the misdirected END packet arrives immediately after a BEGIN packet, and that the actual message meant for this destination was composed of two packets. The message will appear to be valid, and the message length will appear to be correct. Thus, there are fewer options for detecting the error than there are in the first case; i.e., this scenario is harder to detect.

V. Guidelines For Designing Error Detection Schemes

In this section we use the above discussion of the various error scenarios to provide guidelines for designing effective and efficient error detection schemes. Before designing an error detection scheme, it is important to enumerate the various errors that can be expected in the system and estimate the likelihood of their occurrence. It is important that the error detection scheme be robust. Thus, when estimating the frequency of occurrence of the various errors, reasonable worst-case scenarios should be considered.

A. Step 1: Reduce Level of Misdirected Data

The first step should be to deal with errors that involve delivering data to the wrong destination. Of all the errors shown in Fig. 3, misdirected data is the most serious since it is a potential security threat. Preventing data from being sent to the wrong destination is important, as opposed to just detecting the stray data after it has reached the incorrect destination. Of the three layers discussed, only layer N-2 is capable of preventing data from being sent to the wrong destination. A redundancy check on the routing field, such as a CRC, can be included in the layer N-2 packet control information. Each intermediate node along the data path would then check the value of the CRC before allowing the packet to be forwarded.

Assume that an Extended Hamming Code CRC is used. If the length of the routing field is A, then, from Equation (1), the length of the CRC, L, should be chosen to satisfy: $2^{L-1} - L \geq A+1$. Let P_R be the probability of random bit errors, and let P_B be the probability that the routing field of a packet will be hit by a burst error. We make the worst case assumption that any undetected error in the routing field will result in a misdirection. For a given L, we can use Equations (2) and (3) to upper bound the probability of misdirected data by: $\binom{L+A}{3}\frac{1}{4} P_R^4 + P_B 2^{-L}$.

(We assume that P_R is small enough that terms due to more than four random bit errors are insignificant.) If this probability of misdirected data is not small enough to satisfy the security concerns of network users, then a longer CRC could be used (if the term due to burst errors is the dominant term) or a more powerful check, such as an Extended BCH Code CRC, could be used.

Rather than dropping a packet when a single bit error is detected, a node can use the CRC to correct the error. Then, at least two bit errors will have to occur before the packet is dropped. Thus, the probability of dropped packets due to random bit errors in the routing field decreases from about $(L+A)P_R$ to about $\binom{L+A}{2}P_R^2$. However, as discussed in Section III, the probability of misdirected data increases to: $\binom{L+A}{3} P_R^3 + (L+A+1)\, P_B\, 2^{-L}$.

Obviously, using the CRC in the correction mode involves a tradeoff. If the increase in misdirected packets is greater than the decrease in lost packets, it does not make sense to use the correction option. Or, if the higher probability of misdirection is unacceptable to network users for security reasons, then the correction option should not be used. Also, it is important to consider whether the decrease in probability of dropped packets is significant. If there are other error scenarios that will result in a probability of dropped packets higher than $(L+A)P_R$, then the decrease will be insignificant, and the correction option should not be used. These design decisions obviously depend on the value of P_R and P_B. Also, more elaborate schemes can be implemented; in ATM, for example, the CRC that checks the routing information is used in the correction mode only under certain conditions [8].

B. Step 2: Add Check Fields to Detect Bit Errors in Data

From the point of view of efficiency, the next logical step is to choose a mechanism to detect the error scenario that is most difficult to detect. The mechanism chosen to detect this error may also help detect other scenarios, but the converse is unlikely to be true.

From the error diagram in Fig. 3, we see that "Bit Errors in Packet Data" is the most difficult error scenario to detect. The only inconsistency produced by this error is that the value of one or more data bits is incorrect; it does not affect the reconstruction of the message. The only method of checking the validity of the data is to add a redundancy check at one of the three layers. The redundancy check must be powerful enough to reduce the level of undetected error due to bit errors in the data below the acceptable threshold, since no other error detection fields will help detect this error. We assume a CRC will be chosen for this purpose. (A checksum is an alternative form of redundancy check. However, the checksum

implemented in TCP, for example, can fail with the occurrence of just two bit errors [7].)

The question remains at which layer or layers should the CRC be added. A CRC at layer N that checks on the integrity of the whole message is capable of detecting most other error scenarios also, whereas a CRC at layer N-1 or N-2 that just checks on the integrity of the packet is not. Thus, this decision likely affects the mechanisms that will be chosen to detect other errors. Of course, it is possible to add a CRC at multiple layers. However, it is most efficient to implement a CRC at just one layer.

Checking on the data at layer N-2 entails including a CRC with each packet and checking on the validity of the packet data at each node along the data path. If we choose to check on the data at layer N-1, then a CRC is added to each packet, but the validity of the data is only checked at the destination. Finally, if we choose the layer N option, then a CRC is added to each message, and the data check is only performed at the destination.

The first issue is whether to check on the validity of the data at each node along the data path, or whether to check it on an end-to-end basis only. The advantage of checking the data on a node-by-node basis is that data that picks up an error along a data link can be dropped at the next node, rather than being sent all the way to the destination. Also, it is possible that some errors may be easier to detect if a check is performed node-by-node. For example, recall that an Extended Hamming Code CRC can detect with certainty up to three bit errors, but may fail to detect four or more bit errors. Thus, if picking up a bit error on each link is expected to be a common occurrence, it would be easier to detect the bit errors if the packets were checked at each node rather than after they have crossed several links and picked up many bit errors. The disadvantage of node-by-node checking is that the CRC has to be calculated at each intermediate node, which may add to the packet delay. In general, the only reason to perform the node-by-node check would be if it is likely an error would be discovered on a link. We will assume that the data links are reliable enough not to warrant node-by-node checking. For a further discussion of the advantages of end-to-end checking as opposed to node-by-node checking, see [9].

The next sub-section compares performing the data check at layer N-1 versus layer N. In both cases, the check is performed on an end-to-end basis; the only difference is whether the CRC is included per-packet (layer N-1) or per-message (layer N). (Recall that we assume retransmissions are done on a per-message basis. If retransmissions were done on a per-packet basis, then a per-packet redundancy check would allow the system to determine which packets need to be retransmitted.)

1) Per-Packet CRC vs. Per-Message CRC

We want to compare the effectiveness and efficiency of a per-packet CRC to that of a per-message CRC. Assume that we are dealing with fixed length packets and messages, where M is the length of a message and K is the length of a packet (including any control information). Assume M = N K. From Equation (1) we know that the minimum length per-packet CRC that should be used is roughly log K bits; similarly, the minimum length per-message CRC that should be used is roughly log M bits (logs are taken base 2).

Overhead - The bandwidth efficiency of the two schemes can be analyzed in terms of the number of overhead bits added per message, and the number of packets that are retransmitted. Since we have assumed that retransmissions are done on a per-message basis, however, the number of retransmitted packets will be the same regardless of whether the CRC is included at layer N or layer N-1 (assuming the same errors are detected).

Assume that the minimum length CRCs are used in both cases, and assume that N is much smaller than K. Then the percentage overhead represented by the CRC in the two options is:

$$\text{Overhead with the per-packet CRC option} \approx \frac{\log K}{K}$$

$$\text{Overhead with the per-message CRC option} \approx \frac{\log M}{M} \approx \frac{\log K + \log N}{N K} \approx \frac{\log K}{NK}$$

Thus, the per-packet CRC requires roughly N times more overhead than the per-message CRC.

Random Bit Errors - Next, we compare the effectiveness of the two methods in detecting random bit errors. Assume the probability of random bit errors is P_R. Using Equation (2), we can upper bound the probability of four random bit errors causing an undetected error by:

$$\binom{T}{3}\frac{1}{4}\,P_R{}^4 \tag{6}$$

where T equals the length of the data being checked plus the length of the CRC. We assume that P_R is small enough that terms due to more than four bit errors are insignificant compared to this.

If a per-packet CRC is used, then the probability a message will contain undetected random bit errors is upper bounded by:

$$N\binom{K}{3}\frac{1}{4}P_R{}^4 \approx N K^3\ \frac{1}{24}\ P_R{}^4 \tag{7}$$

The factor of N is necessary since the probability of a message containing an undetected error equals the probability that at least one of the packets contains an undetected error.

If a per-message CRC is used, then the probability a message will contain undetected random bit errors is upper bounded by:

$$\binom{M}{3}\frac{1}{4}\ P_R{}^4 \ \approx\ M^3\ \frac{1}{24}\ P_R{}^4 \ \approx\ N^3 K^3\ \frac{1}{24}\ P_R{}^4 \tag{8}$$

Thus, if we are comparing the effectiveness of a per-packet CRC and a per-message CRC where the length of the CRC is roughly log K and log M, respectively, then undetected errors due to random bit errors are roughly N^2 more likely with the per-message CRC.

However, as stated above, a per-message CRC of length log M will likely result in much less overhead than a per-packet CRC of length log K. Thus, for a fair comparison of the

effectiveness of the two methods, we should examine the case where the length of the per-message CRC is increased. If we increase the length of the per-message CRC to roughly 2•log M and use an Extended BCH code rather than an Extended Hamming code, then, using Equation (5), the probability of a message containing undetected random bit errors can be upper bounded by:

$$\binom{M}{4}\frac{1}{15}\,P_R{}^6 \approx N^4 K^4 \frac{1}{360}\,P_R{}^6 \qquad (9)$$

If $N^3 K P_R{}^2 < 15$, which is likely for reasonable values of these parameters, then this probability is smaller than the probability given by Equation (7) for the per-packet CRC option.

We conclude that if random bit errors are a significant problem, and some design specification prohibits us from using a per-message CRC longer than log M, then the per-packet CRC would be the preferred option. Otherwise, as far as random bit errors are concerned, there is not a significant difference between using a per-packet CRC or a per-message CRC.

Random Burst Errors - In this section, we compare the effectiveness of the two types of CRCs in detecting random burst errors. Let P_B represent the probability of a random burst error. It is important that the error detection scheme be robust. Thus, when considering random burst errors, short burst errors are of greater concern than long burst errors. Long burst errors are likely to affect packet control information, and thus cause problems in reconstructing the message. Short burst errors are more likely to affect only the packet data and are thus harder to detect. Below we assume that packets are hit randomly by a short burst error with probability P_B.

First, we note that a CRC of length L can detect with certainty burst errors of length less than or equal to L bits. Thus, since the per-message CRC is longer than the per-packet CRC, there are more burst errors that will be caught with certainty by the per-message CRC than the per-packet CRC. For burst errors longer than this but shorter than the length of a packet, we have:

a) If we use the per-packet CRC option, the probability a message will have an undetected error due to a burst error is:

$$N P_B 2^{-L} \approx N P_B 2^{-(\log K)} = \frac{N P_B}{K} \qquad (10)$$

b) If we use the per-message CRC option, the probability a message will have an undetected error due to a burst error is:

$$N P_B 2^{-L} \approx N P_B 2^{-(\log M)} = \frac{N P_B}{M} = \frac{P_B}{K} \qquad (11)$$

The per-message CRC is N times more effective in detecting short burst errors. Also, as mentioned above, it is likely that a per-message CRC of length longer than log M would be used. Assume that the length is about 2•log M. Then the probability of undetected burst errors is:

$$N P_B 2^{-L} \approx N P_B 2^{-(2 \log M)} = \frac{N P_B}{M^2} = \frac{P_B}{N K^2} \qquad (12)$$

Thus, with this doubling of the CRC length, the per-message CRC option is N^2K more effective in detecting burst errors than the per-packet CRC option.

Implementation - It is probably easier to implement the per-packet CRC, since the CRC can be calculated as soon as the packet arrives. If a per-message CRC is used, the CRC can be fully calculated only after the whole message has arrived. If partial calculation of the per-message CRC is done as the packets arrive, then these partial results need to be stored. Packets from different connections are likely to be intermixed, so there will need to be storage for all active connections at the destination [10].

From the standpoint of implementation, the drawback of the per-packet CRC lies in its inability to detect several error scenarios. It can only check on the integrity of the packet, not the whole message. For example, a per-packet CRC cannot detect the case of a lost packet. The limited error detection power of a per-packet CRC forces the need for additional error detection fields. These extra fields will have to be checked at the destination, which adds to the complexity of the scheme.

We conclude that if random bit errors are a significant problem and we are forced to use a per-message CRC no longer than length log M, then a per-packet CRC option should be used. Otherwise, a per-message CRC is preferable.

2) Correction of Errored Data

The CRC could also be used to correct single bit errors in the data. The tradeoffs are similar to what was discussed in Step 1, where we considered using the routing field CRC to correct errors. For example, if a per-message CRC is used and the length of the message is M (including an L bit CRC), then operating in the correction mode decreases the probability of dropped messages due to random bit errors in the data from $M•P_R$ to $\binom{M}{2} P_R{}^2$. However, the probability of undetected error due to bit errors or burst errors in the data increases from: $\binom{M}{3}\frac{1}{4} P_R{}^4 + P_B 2^{-L}$

to: $\binom{M}{3} P_R{}^3 + (M + 1) P_B 2^{-L}$.

If using the CRC to correct errors does not significantly reduce the overall level of dropped messages, then the correction option should not be used. Another consideration is whether the resulting increase in undetected errors is tolerable. If it is not, then an alternative would be to use an Extended BCH Code CRC of length 2L in the correction mode. Such a CRC is capable of correcting all single and double bit errors. If the Extended BCH Code CRC is used to correct errors, then the probability of undetected error would be approximately the same as if an Extended Hamming code of length L is used in the detection-only mode. (For more details, see [7].) Finally, if it is not critical to have error-free data (e.g., a video application), then using a CRC to correct errors would probably not be worthwhile.

Thus, using a CRC to correct bit errors is appropriate for only certain connections.

C. Step 3: Consider Additional Correction Options at Layer N-1

The next step is to look at the errors, other than bit errors in the packet data, that can be 'corrected' if dealt with at layer N-1. As stated in Section II.B, stray packets, out-of-sequence packets, and duplicate packets are all scenarios that layer N-1 is capable of correcting: stray and duplicate packets can be dropped, and out-of-order packets can be resequenced. One needs an estimate of the likelihood of these scenarios to decide if the decrease in the number of retransmissions that will be accomplished by correcting these scenarios at layer N-1 justifies the overhead that must be added per packet to perform the corrections. In general, if layer N-1 does not handle these errors, it will pass up an errored message to layer N. Layer N may be able to detect the error, but it likely will not be able to correct the error. For example, if packets are put together in the wrong order, layer N will likely detect the error if it uses a per-message CRC, but it would not be capable of correctly resequencing the data. Thus, the message would be dropped. Also, note that in general, layer N-2 is not capable of correcting errors involving reconstruction of the message since layer N-2 deals with packets on an individual basis.

Including the destination address or a message ID in the packet are ways of detecting stray packets. Once a stray packet is detected by layer N-1, it can simply be dropped. However, if the method of preventing misdirected packets is effective enough, it may not be worthwhile to include extra check fields for the purpose of discarding stray packets.

If it is expected that a large number of packets will arrive at the destination out-of-sequence, then it makes sense to include a mechanism at layer N-1 to properly resequence the packets. Especially for a datagram system, where out-of-sequence packets are expected, it is necessary to include some type of packet numbering scheme. The combination of a numbering scheme and message ID field can also be used to help identify duplicate packets.

D. Step 4: Add Check Fields to Detect Remaining Errors

The next step is to deal with any remaining error scenarios that have not already been sufficiently handled by the error detection fields chosen in the above steps. Any of the other remaining errors involve problems in reconstructing a message. (The one exception is "Misdirected Message" which we discuss in the next section.) We can deal with these remaining scenarios at either layer N or N-1. In general, we do not have the option of detecting these scenarios at layer N-2, since this layer does not view a packet in terms of belonging to a message.

Error detection at layer N-1 usually involves adding control information to each packet. The effectiveness of this form of error detection greatly depends on the specific circumstances of the error scenario. For example, consider adding a sequence number to each packet to help detect lost packets. Assume packets are expected to travel in sequence. If the message consists of ten packets, and it is the second packet in the message that is lost, then all eight packets after that in the message would have to have an error in the sequence number field in order for the lost packet to go undetected. However, if it is the last packet in the message that is lost, then the sequence number does not help at all in detecting the error. Thus, the effectiveness of adding error detection fields to each packet may be highly variable. In the best case, there must be a bit error in several packets before the error could go undetected. In the worst case, no bit errors are required for the error to go undetected, which likely necessitates the addition of other error detecting fields.

Error detection at layer N usually involves adding a CRC that checks on the integrity of the whole message. A CRC of a given length L is consistent in its ability to detect errors. Errors involving the reconstruction of a message usually result in the data appearing to be completely random compared to what was originally sent or result in a random set of L bits being interpreted as the CRC. In both cases, we model the CRC as failing to detect the error with probability 2^{-L}.

Based on these observations, it is likely that error detection at layer N should be used. When examining error detection schemes, it is important to consider reasonable worst-case scenarios. As was stated in the introduction, it is much better to design a scheme that provides sufficient protection against all (or almost all) possible error scenarios than to design a scheme that provides enormous protection against some errors and little protection against others.

E. Step 5: Consider Single-Packet Messages

One test of the effectiveness of an error detection scheme is the rate of undetected error for average sized messages. However, considering just average sized messages may not provide a good measure of the overall effectiveness of the scheme. It is easier to detect packet control errors in multi-packet messages than in single-packet messages. Thus, more error protection may be needed to guard against undetected errors in single packet messages.

Consider the case of one packet of a multi-packet message having an error in its control information. It is likely that an inconsistency will occur when the destination attempts to join this packet together with the other packets in the message. Or, if a packet is misdirected, the fact that it is misdirected is more likely to be detected if the 'new' destination attempts to fit this stray packet together with packets that do belong at that destination. Essentially, errors in the control information of a packet may be detected by the other packets in the message.

Single-packet messages are more vulnerable to packet control errors. For example, if a single packet message is misdirected, a per-message CRC will likely not detect the error since it does not check the routing information added by layer N-2. In order to protect single-packet messages against these types of errors, it is necessary to add information to the message that can be verified based on the single packet. For example, including the destination address in the message would help detect the case of a single-packet message being misdirected.

VI. Conclusion

We summarize the five steps for designing an error detection scheme:

Step 1: Reduce Level of Misdirected Data. This usually takes the form of adding a redundancy check on the packet address field.

Step 2: Add Check Fields to Detect Bit Errors in the Data. In most circumstances, adding a CRC to check on the whole message is the most effective way of detecting errors in the data.

Step 3: Consider Additional Correction Options at Layer N-1. The most important consideration is whether to deal with resequencing out-of-sequence packets.

Step 4: Add Check Fields to Detect Remaining Errors. Using a CRC at layer N to detect errors rather than adding many fields at layer N-1 will likely provide more robust error detection.

Step 5: Consider Single Packet Messages. Packet control errors are more difficult to detect in single-packet messages; thus more error detection may be needed to handle this special case.

References

[1] A. Tanenbaum, *Computer Networks*, Second Edition, Prentice-Hall, Inc., Englewood Cliffs, 1988.

[2] A. Meijer and P. Peeters, *Computer Network Architectures*, Computer Science Press, 1982.

[3] S. Vanstone, and P. van Oorschoot, *An Introduction to Error Correcting Codes with Applications*, Kluwer Academic Publishers, Boston, 1989.

[4] D. Bertsekas and R. Gallager, *Data Networks*, Second Edition, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1992.

[5] J. M. Simmons, and R. Gallager, "Design of error detection scheme for Class C service in ATM," *IEEE/ACM Transactions on Networking*, vol.2, no.1, pp.80-88, Feb. 1994.

[6] W. Peterson, *Error Correcting Codes*, MIT Press and John Wiley & Sons, Inc, 1961.

[7] J. M. Simmons, *End-to-End Reliable Communication in Data Networks*, PhD Thesis, MIT, August, 1993.

[8] CCITT Recommendation I.432, "B-ISDN User-Network Interface - Physical Layer Specification", Matsuyama, Dec. 1990.

[9] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” *ACM Trans. On Computer Sys.*, vol. 2, no. 4, Nov. 1984.

[10] S. Dravida, Y. Cheng, and V. R. Saksena, "Error performance of broadband ISDN networks," submitted to *IEEE Transactions on Communications*, 1991.