**Chapter 9. Error Correction**

• Forward Error Correction

• Channel Coding

• Retransmission

• Implementation Considerations

Although it is clearly important to be able to conceal the effects of transmission

errors, it is better if those errors can be avoided or corrected. This chapter presentstechniques that the sender can use to help receivers recover from packet loss andother transmission errors.

The techniques used to correct transmission errors fall into two basic categories:forward error correction and retransmission.HTPU80UTPH Forward error correction relies onadditional data added by the sender to a media stream, which receivers can thenuse to correct errors with a certain probability. Retransmission, on the other hand,relies on explicit requests for additional copies of particular packets.

The choice between retransmission and forward error correction depends on theapplication and on the network characteristics. The details and trade-offs of thedifferent approaches are discussed in more detail in this chapter.

**Forward Error Correction**

TForward error correctionT (FEC) algorithms transform a bit stream to make it robustfor transmission. The transformation generates a larger bit stream intended fortransmission across a lossy medium or network. The additional information in thetransformed bit stream allows receivers to exactly reconstruct the original bit

stream in the presence of transmission errors. Forward error correction algorithmsare notably employed in digital broadcasting systems, such as mobile telephony andspace communication systems, and in storage systems, such as compact discs,computer hard disks, and memory. Because the Internet is a lossy medium, andbecause media applications are sensitive to loss, FEC schemes have been proposedand standardized for RTP applications. These schemes offer both exact and

approximate reconstruction of the bit stream, depending on the amount and type ofFEC used, and on the nature of the loss.

When an RTP sender uses FEC, it must decide on the amount of FEC to add, on thebasis of the loss characteristics of the network. One way of doing this is to look atthe RTCP receiver report packets it is getting back, and use the loss fraction

statistics to decide on the amount of redundant data to include with the mediastream.

In theory, by varying the encoding of the media, it is possible to guarantee that acertain fraction of losses can be corrected. In practice, several factors indicate thatFEC can provide only probabilistic repair. Key among those is the fact that addingFEC increases the bandwidth of a stream. This increase in bandwidth limits theamount of FEC that can be added on the basis of the available network capacity, andit may also have adverse effects if loss is caused by congestion. In particular, addingbandwidth to the stream may increase congestion, worsening the loss that the FECwas supposed to correct. This issue is discussed further in the section titled At the

Sender, under HTUImplementation ConsiderationsUTH, later in this chapter, as well as inHTUChapter 10UTH, Congestion Control.

Note that although the amount of FEC can be varied in response to reception qualityreports, there is typically no feedback about individual packet loss events, and noguarantee that all losses are corrected. The aim is to reduce the residual loss rate tosomething acceptable, then to let error concealment take care of any remainingloss.

If FEC is to work properly, the loss rate must be bounded, and losses must occur inparticular patterns. For example, it is clear that an FEC scheme designed to correct5% loss will not correct all losses if 10% of packets are missing. Less obviously, itmight be able to correct 5% loss only if the losses are of nonconsecutive packets.

The key advantage of FEC is that it can scale to very large groups, or groups whereno feedback is possible.HTPU54UTPH The amount of redundant data added depends on theaverage loss rate and on the loss pattern, both of which are independent of thenumber of receivers. The disadvantage is that the amount of FEC added depends onthe average loss rate. A receiver with below-average loss will receive redundantdata, which wastes capacity and must be discarded. One with above-average losswill be unable to correct all the errors and will have to rely on concealment. If theloss rates for different receivers are very heterogeneous, it will not be possible tosatisfy them all with a single FEC stream (layered coding may help; see HTUChapter 10UTH,Congestion Control).

Another disadvantage is that FEC may add delay because repair cannot happen until

the FEC packets arrive. If FEC packets are sent a long time after the data theyprotect, then a receiver may have to choose between playing damaged data quicklyor waiting for the FEC to arrive and potentially increasing the end-to-end delay. Thisis primarily an issue with interactive applications, in which it is important to havelow delay.

Many FEC schemes exist, and several have been adopted as part of the RTP

framework. We will first review some techniques that operate independently of themedia format—parity FEC and Reed–Solomon encoding—before studying thosespecific to particular audio and video formats.

**Parity FEC**

One of the simplest error detection/correction codes is the parity code. The parityoperation can be described mathematically as an exclusive-or (XOR) of the bitstream. The XOR operation is a bitwise logic operation, defined for two inputs in thisway:

The operation may easily be extended to more than two inputs because XOR isassociative:

0 XOR 0 = 0

1 XOR 0 = 1

0 XOR 1 = 1

1 XOR 1 = 0

A XOR B XOR C = (A XOR B) XOR C = A XOR (B XOR C)

Changing a single input to the XOR operation will cause the output to change,

allowing a single parity bit to detect any single error. This capability is of limitedvalue by itself, but when multiple parity bits are included, it becomes possible todetect Tand correctT multiple errors.

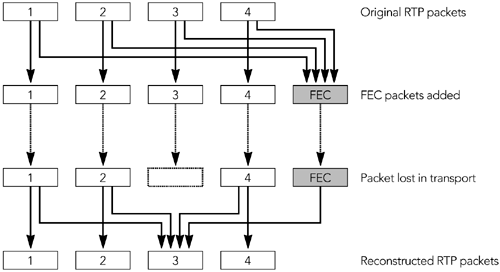
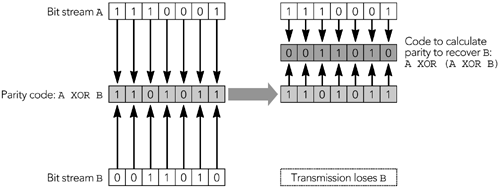
To make parity useful to a system using RTP-over-UDP/IP—in which the dominanterror is packet loss, not bit corruption—it is necessary to send the parity bits in aseparate packet to the data they are protecting. If there are enough parity bits, theycan be used to recover the complete contents of a lost packet. The property thatmakes this possible is that

A XOR B XOR B = A

for any values of TAT and TBT.

If we somehow transmit the three pieces of information TAT, TBT, and TA XOR BT separately,we need only receive two of the three pieces to recover the values of TAT and TBT.

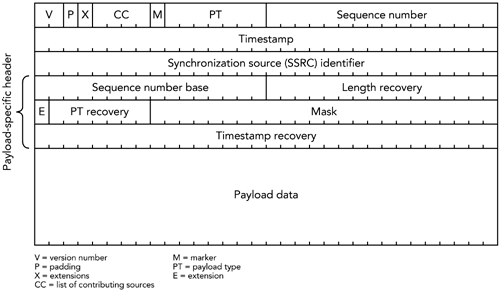
HTUFigure 9.1UTH shows an example in which a group of seven lost bits is recovered viathis process, but it works for bit streams of any length. The process may be directlyapplied to RTP packets, treating an entire packet as a bit stream and calculatingparity packets that are the XOR of original data packets, and that can be used torecover from loss.

**Figure 9.1. Use of Parity between Bit Streams to Recover LostData**

The standard for parity FEC applied to RTP streams is defined by RFC 2733.HTPU32UTPH Theaim of this standard is to define a generic FEC scheme for RTP packets that canoperate with any payload type and that is backward-compatible with receivers thatdo not understand FEC. It does this by calculating FEC packets from the original RTPdata packets; these FEC packets are then sent as a separate RTP stream, which maybe used to repair loss in the original data, as shown in HTUFigure 9.2UTH.

**Figure 9.2. Repair Using Parity FEC (From C. Perkins, O.**

**Hodson, and V. Hardman, &quot;A Survey of Packet Loss RecoveryTechniques for Streaming Media,&quot; IEEE Network Magazine,September/October 1998. © 1998 IEEE.)**

**FORMAT OF PARITY FEC PACKETS**

The format of an FEC packet, shown in HTUFigure 9.3UTH, has three parts to it: the

standard RTP header, a payload-specific FEC header, and the payload data itself.

With the exception of some fields of the RTP header, the FEC packet is generatedfrom the data packets it is protecting. It is the result of applying the parity operationto the data packets.

**Figure 9.3. Format of a Parity FEC Packet**

The fields of the RTP header are used as detailed here:

• The version number, payload type, sequence number, and timestamp are

assigned in the usual manner. The payload type is dynamically assigned,according to the RTP profile in use; the sequence number increases by onefor each FEC packet sent; and the timestamp is set to the value of the RTPmedia clock at the instant the FEC packet is transmitted. (The timestamp isunlikely to be equal to the timestamp of the surrounding RTP packets.) As aresult, the timestamps in FEC packets increase monotonically,

independently of the FEC scheme.

• The SSRC (synchronization source) value is the same as the SSRC of the

original data packets.

• The padding, extension, CC, and marker bits are calculated as the XOR of the

equivalent bits in the original data packets. This allows those fields to berecovered if the original packets are lost.

• The CSRC (contributing source) list and header extension are never present,

independent of the values of the CC field and X bit. If they are present in the

original data packets, they are included as part of the payload section of theFEC packet (after the FEC payload header).

Note that the prohibition of CSRC list and header extension in parity FEC packets

means that it is not always possible to treat FEC streams according to the

standard, payload format–independent, RTP processing rules. In particular, an

FEC stream cannot pass through an RTP mixer (the media data can, but the

mixer will have to generate a new FEC stream for the mixed data)

.

The payload header protects the fields of the original RTP headers that are notprotected in the RTP header of the FEC packet. These are the six fields of the

payload header:

1. T**Sequence number base**T. The minimum sequence number of the original

packets composing this FEC packet.

2. T**Length recovery**T. The XOR of the lengths of the original data packets. The

lengths are calculated as the total length of the payload data, CSRC list,header extension, and padding of the original packets. This calculationallows the FEC procedure to be applied even when the lengths of the mediapackets are not identical.

3. T**Extension (E)**T. An indicator of the presence of additional fields in the FEC

payload header. It is usually set to zero, indicating that no extension ispresent (the ULP format, described later in this chapter, uses the extensionfield to indicate the presence of additional layered FEC).

4. T**Payload type (PT) recovery**T. The XOR of the payload type fields of the

original data packets.

5. T**Mask**T. A bit mask indicating which of the packets following the sequence

number base are included in the parity FEC operation. If bit TiT in the mask isset to 1, the original data packet with sequence number TN + iT is associatedwith this FEC packet, where TNT is the sequence number base. The leastsignificant bit corresponds to TiT = 0, and the most significant to TiT = 23,

allowing for the parity FEC to be calculated over up to 24 packets, which maybe nonconsecutive.

6. **T imestamp recovery**T. The XOR of the timestamps of the original data

packets.

The payload data is derived as the XOR of the CSRC list (if present), header

extension (if present), and payload data of the packets to be protected. If the datapackets are different lengths, the XOR is calculated as if the short packets werepadded out to match the length of the largest (the contents of the padding bits areunimportant, as long as the same values are used each time a particular packet isprocessed; it is probably easiest to use all zero bits).

**USE OF PARITY FEC**

The number of FEC packets and how they are generated depend on the FEC schemeemployed by the sender. The payload format places relatively few restrictions onthe mapping process: Packets from a group of up to 24 consecutive original packetsare input to the parity operation, and each may be used in the generation of multipleFEC packets.

The sequence number base and mask in the payload header are used to indicatewhich packets were used to generate each FEC packet; there is no need for

additional signaling. Accordingly, the packets used in the FEC operation can changeduring an RTP session, perhaps in response to the reception quality informationcontained in RTCP RR packets. The ability of the FEC operation to change gives thesender much flexibility: The sender can adapt the amount of FEC in use according tonetwork conditions and be certain that the receivers will still be able to use the FECfor recovery.

A sender is expected to generate an appropriate number of FEC packets in real time,

as the original data packets are sent. There is no single correct approach for

choosing the amount of FEC to add because the choice depends on the loss

characteristics of the network, and the standard does not mandate a particularscheme. Following are some possible choices:

• The simplest approach is to send one FEC packet for every TnT – 1 data packets,

as shown in HTUFigure 9.4AUTH, allowing recovery provided that there is at mostone loss for every TnT packets. This FEC scheme has low overhead, is easy tocompute, and is easy to adapt (because the fraction of packets that are FECpackets directly corresponds to the loss fraction reported in RTCP RR

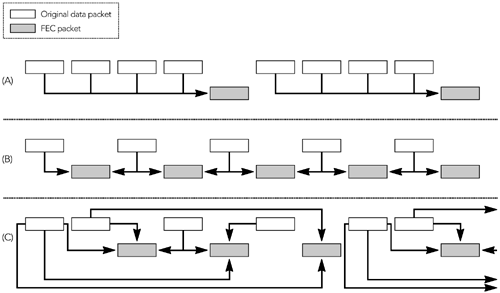
packets).

If the probability that a packet is lost is uniform, this approach works well;however, bursts of consecutive loss cannot be recovered. If bursts of loss arecommon—as in the public Internet—the parity can be calculated acrosswidely spaced packets, rather than over adjacent packets, resulting in morerobust protection. The result is a scheme that works well for streaming buthas a large delay, making it unsuitable for interactive applications.

• A more robust scheme, but one with significantly higher overhead, is to send

an FEC packet between each pair of data packets, as shown in HTUFigure 9.4BUTH.

This approach allows the receiver to correct every single packet loss, andmany double losses. The bandwidth overhead of this approach is high, butthe amount of delay added is relatively small, making it more suitable forinteractive applications.

• Higher-order schemes allow recovery from more consecutive losses. For

example, HTUFigure 9.4CUTH shows a scheme that can recover from loss of up tothree consecutive packets. Because of the need to calculate FEC over

multiple packets, the delay introduced is relatively high, so these schemesare unlikely to be suitable for interactive use. They can be useful in

streaming applications, though.

**Figure 9.4. Some Possible FEC Schemes**

To make parity FEC backward-compatible, it is essential that older receivers do notsee the FEC packets. Thus the packets are usually sent as a separate RTP stream, ona different UDP port but to the same destination address. For example, consider asession in which the original RTP data packets use static payload type 0 (G.711µ-law) and are sent on port 49170, with RTCP on port 49171. The FEC packets couldbe sent on port 49172, with their corresponding RTCP on port 49173. The FECpackets use a dynamic payload type—for example, 122. This scenario could bedescribed in SDP like this:

v=0

o=hamming 2890844526 2890842807 IN IP4 128.16.64.32

s=FEC Seminar

c=IN IP4 10.1.76.48/127

t=0 0

m=audio 49170 RTP/AVP 0 122

a=rtpmap:122 parityfec/8000

a=fmtp:122 49172 IN IP4 10.1.76.48/127

An alternative—described in the section titled HTUAudio Redundancy CodingUTH later in

this chapter—is to transport parity FEC packets as if they were a redundant

encoding of the media.

**RECOVERING FROM LOSS**

At the receiver the FEC packets and the original data packets are received. If no

data packets are lost, the parity FEC can be ignored. In the event of loss, the FECpackets can be combined with the remaining data packets, allowing the receiver torecover lost packets.

There are two stages to the recovery process. First, it is necessary to determinewhich of the original data packets and the FEC packets must be combined in orderto recover a missing packet. After this is done, the second step is to reconstruct thedata.

Any suitable algorithm can be used to determine which packets must be combined.

RFC 2733 gives an example, which operates as shown here:

• When an FEC packet is received, the sequence number base and mask fields

are checked to determine which packets it protects. If all those packets havebeen received, the FEC packet is redundant and is discarded. If some ofthose packets are missing, Tand they have sequence numbers smaller thanthe highest received sequence numberT, recovery is attempted; if recovery issuccessful, the FEC packet is discarded and the recovered packet is storedinto the playout buffer. Otherwise the FEC packet is stored for possible lateruse.

• When a data packet is received, any stored FEC packets are checked to see

whether the new data packet makes recovery possible. If so, after recoverythe FEC packet is discarded and the recovered packet entered into theplayout buffer.

• Recovered packets are treated as if they were received packets, possibly

triggering further recovery attempts.

Eventually, all FEC packets will be used or discarded as redundant, and all

recoverable lost packets will be reconstructed.

The algorithm relies on an ability to determine whether a particular set of datapackets and FEC packets makes it possible to recover from a loss. Making thedetermination requires looking at the set of packets referenced by an FEC packet; ifonly one is missing, it can be recovered. The recovery process is similar to that usedto generate the FEC data. The parity (XOR) operation is conducted on the equivalentfields in the data packets and the FEC packets; the result is the original data packet.

In more detail, this is the recovery process:

1. The SSRC of the recovered packet is set to the SSRC of the other packets.

2. The padding, header extension, CC, and marker bits of the recovered packet

are generated as the XOR of the same fields in the original and FEC packets.

3. The sequence number of the recovered packet is known from the gap in the

original sequence numbers (that is, there is no need to recover it, because itis directly known).

4. The payload type of the recovered packet is generated as the XOR of the

payload type fields in the original packets, and the payload type recoveryfield of the FEC packets. The timestamp is recovered in the same manner.

5. The length of the payload is calculated as the XOR of lengths of the original

packets and the length recovery field of the FEC packets.

6. The CSRC lists (if present), header extension (if present), and payload of the

recovered packet are calculated as the XOR of those fields in the originalpackets, plus the payload of the FEC packets (because the FEC packet nevercontains a CSRC list or header extension itself, and it carries the protectedversion of the original fields as part of its payload).

The result is an exact reconstruction of the missing packet, bitwise identical to theoriginal. There is no partial recovery with the RFC 2733 FEC scheme. If there aresufficient FEC packets, the lost packet can be perfectly recovered; if not, nothingcan be saved.

**Unequal Error Protection**

Although some payload formats must be recovered exactly, there are other formats

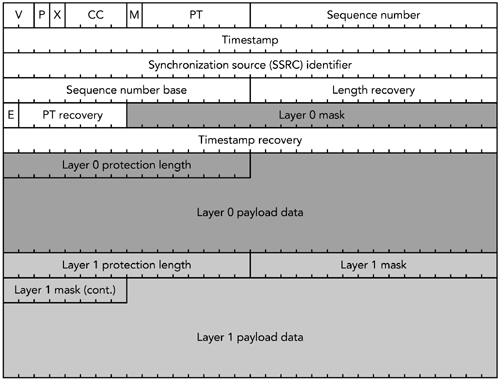
in which some parts of the data are more important than others. In these cases it issometimes possible to get most of the effect while recovering only part of the packet.

For example, some audio codecs have a minimum number of bits that need to berecovered to provide intelligible speech, with additional bits that are not essentialbut improve the audio quality if they can be recovered. A recovery scheme thatrecovers only the minimum data will be lower in quality than one that recovers thecomplete packet, but it may have significantly less overhead.

Alternatively, it is possible to protect the entire packet against some degree of

packet loss but give the most important part of the packet a greater degree ofprotection. In this case the entire packet is recovered with some probability, but theimportant parts have a higher chance of recovery.

Schemes such as these are known as unequal layered protection (ULP) codes. At thetime of this writing, there is no standard for ULP codes applied to RTP. However,there is ongoing work in the IETF to define an extension to the parity FEC codes in

RFC 2733, which will provide this function.HTPU47UTPH This work is incomplete, and the finalstandard may be slightly different from that described here.

The extension provides for layered coding, with each layer protecting a certainportion of the packet. Each layer may have a different length, up to the length of thelongest packet in the group. Layers are arranged so that multiple layers protect thestart of the packet, with later parts being protected by fewer layers. This

arrangement makes it more likely that the start of the packet can be recovered.

The proposed RTP payload format for ULP based on parity FEC is shown in HTUFigure9.5UTH. The start of the payload header is identical to that of RFC 2733, but the

extension bit is set, and additional payload headers follow to describe the layeredFEC operation. The payload data section of the packet contains the protected datafor each layer, in order.

**Figure 9.5. The RTP Payload Format for ULP Based on ParityFEC**

The operation of the ULP-based parity FEC format is similar to that of the standardparity FEC format, except that the FEC for each layer is computed over only part ofthe packet (rather than the entire packet). Each layer must protect the packetsprotected by the lower layers, making the amount of FEC protecting the lower layerscumulative with the number of layers. Each FEC packet can potentially contain datafor all layers, stacked one after the other in the payload section of the packet. TheFEC for the lowest layer appears in all FEC packets; higher layers appear in a subsetof the packets, depending on the FEC operation. There is only one FEC stream,independent of the number of layers of protection.

At the time of this writing, there is a move to revise the RTP payload format for

ULP-based parity FEC described here, so that in addition to providing layered

protection, it also updates the parity FEC format of RFC 2733 to support RTP

mixers better. These changes are not expected to change the layered coding

concepts described, but it is likely that the details of the packet format will

change.

Recovery operates on a per-layer basis, with each layer potentially allowing

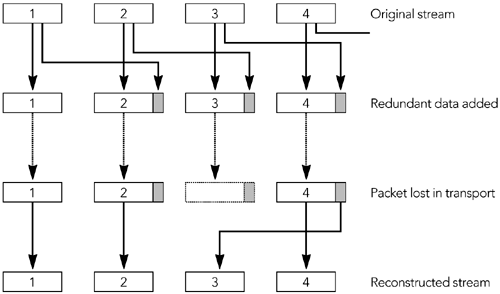
recovery of part of the packet. The algorithm for recovery of each layer is identicalto that of the standard parity FEC format. Each layer is recovered in turn, startingwith the base layer, until all possible recovery operations have been performed.

The use of ULP is not appropriate for all payload formats, because for it to work, thedecoder must be able to process partial packets. When such partial data is useful,ULP can provide a significant gain in quality, with less overhead than complete FECprotection requires.

**Reed–Solomon Codes**

Reed–Solomon codesHTPU98UTPH are an alternative to parity codes that offer protection withless bandwidth overhead, at the expense of additional complexity. In particular,they offer good protection against burst loss, where conventional parity codes areless efficient.

Reed–Solomon encoding involves treating each block of data as the coefficient of apolynomial equation. The equation is evaluated over all possible inputs in a certainnumber base, resulting in the FEC data to be transmitted. Often the procedureoperates per octet, making implementation simpler. A full treatment is outside thescope of this book, but the encoding procedure is actually relatively straightforward,and there are optimized decoding algorithms.

Despite advantages of Reed–Solomon codes compared to parity codes, there is nostandard for their use with RTP. Both equal and unequal FECHTPU48UTPH using

Reed–Solomon codes has generated some interest, and a standard is expected tobe developed in the future.

**Audio Redundancy Coding**

The error correction schemes we have discussed so far are independent of themedia format being used. However, it is also possible to correct errors in a

media-specific way, an approach that can often lead to improved performance.

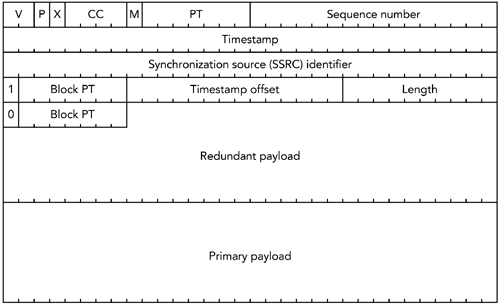
The first media-specific error correction scheme defined for RTP was audio

redundancy coding, specified in RFC 2198.HTPU10UTP**H**,HP**T** U77UTPH The motivation for this codingscheme was interactive voice telecon-ferences, in which it is more important torepair lost packets quickly than it is to repair them exactly. Accordingly, each packetcontains both an original frame of audio data and a redundant copy of a precedingframe, in a more heavily compressed format. The coding scheme is illustrated inHTUFigure 9.6UTH.

**Figure 9.6. Audio Redundancy Coding (From C. Perkins, O.**

**Hodson, and V. Hardman, &quot;A Survey of Packet Loss RecoveryTechniques for Streaming Media,&quot; IEEE Network Magazine,September/October 1998. © 1998 IEEE.)**

When receiving a redundant audio stream, the receiver can use the redundantcopies to fill in any gaps in the original data stream. Because the redundant copy is

typically more heavily compressed than the primary, the repair will not be exact, butit is perceptually better than a gap in the stream.

**FORMAT OF REDUNDANT AUDIO PACKETS**

The redundant audio payload format is shown in HTUFigure 9.7UTH. The RTP header hasthe standard values, and the payload type is a dynamic payload type representingredundant audio.

**Figure 9.7. The RTP Payload Format for Audio RedundancyCoding**

The payload header contains four octets for each redundant encoding of the data,plus a final octet indicating the payload type of the original media. The four-octetpayload header for each redundant encoding contains several fields:

• A single bit indicating whether this is a redundant encoding or the primary

encoding.

• The payload type of the redundant encoding.

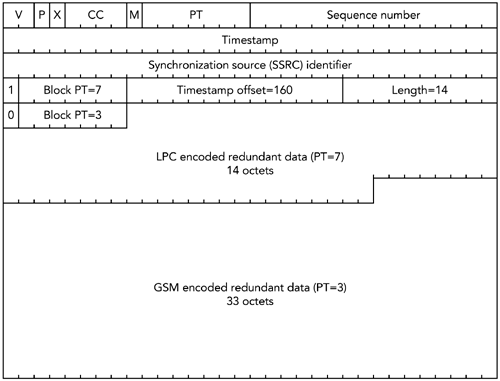
• The length of the redundant encoding in octets, stored as a 10-bit unsigned

integer.

• A timestamp offset, stored as a 14-bit unsigned integer. This value is

subtracted from the timestamp of the packet, to indicate the original playouttime of the redundant data.

The final payload header is a single octet, consisting of one bit to indicate that thisis the last header, and the seven-bit payload type of the primary data. The payload

header is followed immediately by the data blocks, stored in the same order as theheaders. There is no padding or other delimiter between the data blocks, and theyare typically not 32-bit aligned (although they are octet aligned).

For example, if the primary encoding is GSM sent with one frame—20

milliseconds—per packet, and the redundant encoding is a low-rate LPC codec sentwith one packet delay, a complete redundant audio packet would be as shown inHTUFigure 9.8UTH. Note that the timestamp offset is 160 because 160 ticks of an 8kHzclock represent a 20-millisecond offset (8,000 ticks per second x 0.020 seconds =160 ticks).

**Figure 9.8. A Sample Redundant Audio Packet**

The format allows the redundant copy to be delayed more than one packet, as ameans of countering burst loss at the expense of additional delay. For example, ifbursts of two consecutive packet losses are common, the redundant copy may besent two packets after the original.

The choice of redundant encoding used should reflect the bandwidth requirementsof those encodings. The redundant encoding is expected to use significantly lessbandwidth than the primary encoding—the exception being the case in which theprimary has a very low bandwidth and a high processing requirement, in which casea copy of the primary may be used as the redundancy. The redundant encodingshouldn&apos;t have a higher bandwidth than the primary.

It is also possible to send multiple redundant data blocks in each packet, allowingeach packet to repair multiple loss events. The use of multiple levels of redundancyis rarely necessary because in practice you can often achieve similar protection withlower overhead by delaying the redundancy. If multiple levels of redundancy areused, however, the bandwidth required by each level is expected to be significantlyless than that of the preceding level.

The redundant audio format is signaled in SDP as in the following example:

m=audio 1234 RTP/AVP 121 0 5

a=rtpmap:121 red/8000/1

a=fmtp:121 0/5

In this case the redundant audio uses dynamic payload type 121, with the primaryand secondary encoding being payload type 0 (PCM µ-law) and 5 (DVI).

It is also possible to use dynamic payload types as the primary or secondary

encoding—for example:

m=audio 1234 RTP/AVP 121 0 122

a=rtpmap:121 red/8000/1

a=fmtp:121 0/122

a=rtpmap:122 g729/8000/1

in which the primary is PCM µ-law and the secondary is G.729 using dynamic

payload type 122.

Note that the payload types of the primary and secondary encoding appear in boththe Tm=T and Ta=fmtp:T lines of the SDP fragment. Thus the receiver must be preparedto receive both redundant and nonredundant audio using these codecs, both ofwhich are necessary because the first and last packets sent in a talk spurt may benonredundant.

Implementations of redundant audio are not consistent in the way they handle thefirst and last packets in a talk spurt. The first packet cannot be sent with a secondaryencoding, because there is no preceding data: Some implementations send it usingthe primary payload format, and others use the redundant audio format, with thesecondary encoding having zero length. Likewise, it is difficult to send a redundantcopy of the last packet because there is nothing with which to piggyback it: Mostimplementations have no way of recovering the last packet, but it may be possibleto send a nonredundant packet with just the secondary encoding.

**LIMITATIONS OF REDUNDANT AUDIO**

Although redundant audio encoding can provide exact repair—if the redundant copy

is identical to the primary—it is more likely for the redundant encoding to have alower bandwidth, and hence lower quality, and to provide only approximate repair.

The payload format for redundant audio also does not preserve the complete RTPheaders for each of the redundant encodings. In particular, the RTP marker bit andCSRC list are not preserved. Loss of the marker bit does not cause undue problems,

because even if the marker bit were transmitted with the redundant information,there would still be the possibility of its loss, so applications would still have to bewritten with this in mind. Likewise, because the CSRC list in an audio stream isexpected to change relatively infrequently, it is recommended that applicationsrequiring this information assume that the CSRC data in the RTP header may beapplied to the reconstructed redundant data.

**USE OF REDUNDANT AUDIO**

The redundant audio payload format was designed primarily for audio

teleconferencing. To some extent it performs that job very well; however, advancesin codec technology since the format was defined mean that the overhead of thepayload format is perhaps too high now.

For example, the original paper proposing redundant audio suggested the use ofPCM-encoded audio—160 octets per frame—as the primary, with LPC encoding asthe secondary. In this case, the five octets of payload header constitute an

acceptable overhead. However, if the primary is G.729 with ten octets per frame,the overhead of the payload header may be considered unacceptable.

In addition to audio teleconferencing, in which adoption of redundant audio hasbeen somewhat limited, redundant audio is used in two scenarios: with parity FECand with DTMF tones.

The parity FEC format described previously requires the FEC data to be sent

separately from the original data packets. A common way of doing this is to send theFEC as an additional RTP stream on a different port; however, an alternative is totreat it as a redundant encoding of the media and piggyback it onto the originalmedia using the redundant audio format. This approach reduces the overhead of theFEC, but it means that the receivers have to understand the redundant audio format,reducing the backward compatibility.

The RTP payload format for DTMF tones and other telephone eventsHTPU34UTPH suggests theuse of redundant encodings because these tones need to be delivered reliably (for

example, telephone voice menu systems in which selection is made via DTMF touchtones would be even more annoying if the tones were not reliably recognized).

Note that some platforms do not allow UDP checksums to be disabled, and others

allow it as a global setting but not on a per-stream basis. In IPv6-based

implementations, the UDP checksum is mandatory and must not be disabled

(although the UDP Lite proposal may be used).

Encoding multiple redundant copies of each tone makes it possible to achieve veryhigh levels of reliability for the tones, even in the presence of packet loss.

**Channel Coding**

Forward error correction, which relies on the addition of information to the mediastream to provide protection against packet loss, is one form of channel coding. Themedia stream can be matched to the loss characteristics of a particular networkpath in other ways as well, some of which are discussed in the following sections.

**Partial Checksum**

Most packet loss in the public Internet is caused by congestion in the network.

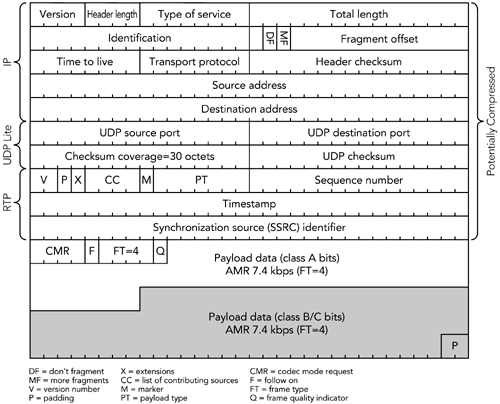
However, as noted in HTUChapter 2UTH, Voice and Video Communication over PacketNetworks, in some classes of network—for example, wireless—noncongestive lossand packet corruption are common. Although discarding packets with corrupted bitsis appropriate in many cases, some RTP payload formats can make use of corrupteddata (for example, the AMR audio codecsHTPU41UTPH). You can make use of partially corruptRTP packets either by disabling the UDP checksum (if IPv4 is used) or by using atransport with a partial checksum.

When using RTP with a standard UDP/IPv4 stack, it is possible to disable the UDPchecksum entirely (for example, using Tsysctlnet.inet.udp.checksum=0T on UNIXmachines supporting TsysctlT, or using the TUDP\_NOCHECKSUMT socket option withWinsock2). Disabling the UDP checksum has the advantage that packets with

corrupted payload data are delivered to the application, allowing some part of thedata to be salvaged. The disadvantage is that the packet header may be corrupted,resulting in packets being misdirected or otherwise made unusable.

A better approach is to use a transport with a partial checksum, such as UDP Lite.HTPU53UTPH

This is a work in progress that extends UDP to allow the checksum to cover only partof the packet, rather than all or none of it. For example, the checksum could coverjust the RTP/UDP/IP headers, or the headers and the first part of the payload. Witha partial checksum, the transport can discard packets in which the headers—or

other important parts of the payload—are corrupted, yet pass those that have errorsonly in the unimportant parts of the payload.

The first RTP payload format to make significant use of partial checksum was theAMR audio codec.HTPU41UTPH This is the codec selected for many third-generation cellular

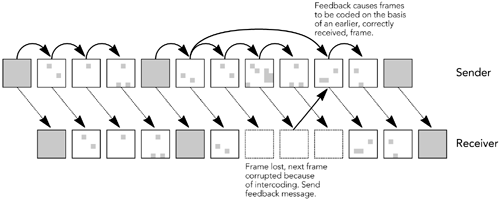
telephony systems, and hence the designers of its RTP payload format placed highpriority on robustness to bit errors. Each frame of the codec bit stream is split intoclass A bits, which are vital for decoding, and class B and C bits, which improvequality if they are received, but are not vital. One or more frames of AMR output areplaced into each RTP packet, with the option of using a partial checksum that coversthe RTP/UDP/IP headers and class A bits, while the other bits are left unprotected.

This lack of protection allows an application to ignore errors in the class B and classC bits, rather than discarding the packets. In HTUFigure 9.9UTH, for example, the shaded

bits are not protected by a checksum. This approach appears to offer little

advantage, because there are relatively few unprotected bits, but when headercompression (see HTUChapter 11UTH) is used, the IP/UDP/RTP headers and checksum arereduced to only four octets, increasing the gain due to the partial checksum.

**Figure 9.9. An Example of the Use of Partial Checksums in theAMR Payload Format**

The AMR payload format also supports interleaving and redundant transmission, forincreased robustness. The result is a very robust format that copes well with the bitcorruption that is common in cellular networks.

Partial checksums are not a general-purpose tool, because they don&apos;t improveperformance in networks in which packet loss is due to congestion. As wirelessnetworks become more common, however, it is expected that future payload

formats will also make use of partial checksums.

**Reference Picture Selection**

Many payload formats rely on interframe coding, in which it is not possible to decodea frame without using data sent in previous frames. Interfame coding is most oftenused in video codecs, in which motion vectors allow panning of the image, or motionof parts of the image, to occur without resending the parts of the preceding framethat have moved. Interframe coding is vital to achieving good compression

efficiency, but it amplifies the effects of packet loss (clearly, if a frame depends onthe packet that is lost, that frame cannot be decoded).

One solution to making interframe encodings more robust to packet loss is

reference picture selection, as used in some variants of H.263 and MPEG-4. This isanother form of channel coding, in which if a frame on which others are predicted islost, future frames are recoded on the basis of another frame that was received (seeHTUFigure 9.10UTH). This process saves significant bandwidth compared to sending thenext frame with no interframe compression (only intraframe compression).

**Figure 9.10. Reference Picture Selection**

To change the reference picture, it is necessary for the receiver to report individualpacket losses to the sender. Mechanisms for feedback are discussed in the nextsection in the context of retransmission; the same techniques can be used forreference picture selection with minor modification. Work on a standard for the use

of reference picture selection in RTP is ongoing, as part of the retransmission profilediscussed next.

**Retransmission**

Losses may also be recovered if the receivers send feedback to the sender, askingit to retransmit packets lost in transit. Retransmission is a natural approach to errorcorrection, and it works well in some scenarios. It is, however, not without problemsthat can limit its applicability. Retransmission is not a part of standard RTP; however,an RTP profile is under developmentHTPU44UTPH that provides an RTCP-based framework forretransmission requests and other immediate feedback.

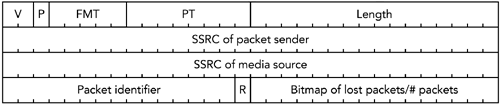
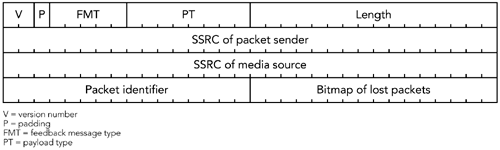
**RTCP as a Framework for Retransmission**

Because RTP includes a feedback channel—RTCP—for reception reports and otherdata, it is natural to use that channel for retransmission requests too. Two steps arerequired: Packet formats need to be defined for retransmission requests, and thetiming rules must be modified to allow immediate feedback.

**PACKET FORMATS**

The profile for retransmission-based feedback defines two additional RTCP packettypes, representing positive and negative acknowledgments. The most commontype is expected to be negative acknowledgments, reporting that a particular set ofpackets was lost. A positive acknowledgment reports that packets were correctlyreceived.

The format of a negative acknowledgment (NACK) is shown in HTUFigure 9.11UTH. TheNACK contains a packet identifier representing a lost packet, and a bitmap showingwhich of the following 16 packets were lost, with a value of 1 indicating loss. Thesender should not assume that a receiver has received a packet just because thecorresponding position in the bit mask is set to zero; all it knows is that the receiverhas not reported the packet lost at this time. On receiving a NACK, the sender isexpected to retransmit the packets marked as missing, although it is under noobligation to do so.

**Figure 9.11. Format of an RTCP Feedback NegativeAcknowledgment**

The format of a positive acknowledgment (ACK) is shown in HTUFigure 9.12UTH. The ACKcontains a packet identifier representing a correctly received packet, and either abitmap or a count of the following packets. If the R bit is set to 1, the final field is acount of the number of correctly received packets following the packet identifier. Ifthe R bit is set to zero, the final field is a bitmap showing which of the following 15packets were also received. The two options allow both long runs of ACKs with fewlosses (R = 1) and occasional ACKs interspersed with loss (R = 0) to be signaledefficiently.

**Figure 9.12. Format of an RTCP Feedback PositiveAcknowledgment**

The choice between ACK and NACK depends on the repair algorithm in use, and onthe desired semantics. An ACK signals that some packets were received; the sendermay assume others were lost. On the other hand, a NACK signals loss of somepackets but provides no information about the rest (for example, a receiver maysend a NACK when an important packet is lost but silently ignore the loss of

unimportant data).

Feedback packets are sent as part of a compound RTCP packet, in the same way asall other RTCP packets. They are placed last in the compound packet, after theSR/RR and SDES items. (See HTUChapter 5UTH, RTP Control Protocol, for a review of RTCPpacket formats.)

**TIMING RULES**

The standard definition of RTCP has strict timing rules, which specify when a packetcan be sent and limit the bandwidth consumption of RTCP. The retransmissionprofile modifies these rules to allow feedback packets to be sent earlier than normal,at the expense of delaying the following packet. The result is a short-term violationof the bandwidth limit, although the longer-term RTCP transmission rate remainsthe same. The modified timing rules can be summarized as follows:

• When no feedback messages need to be sent, RTCP packets are sent

according to the standard timing rules, except that the 5-second minimuminterval between RTCP reports is not enforced (the reduced minimum

discussed in the section titled Reporting Interval in HTUChapter 5UTH, RTP ControlProtocol, should be used instead).

• If a receiver wants to send feedback before the regularly scheduled RTCP

transmission time, it should wait for a short, random dither interval andcheck whether it has already seen a corresponding feedback message fromanother receiver. If so, it must refrain from sending and follow the regularRTCP schedule. If the receiver does not see a similar feedback message fromany other receiver, and if it has not sent feedback during this reportinginterval, it may send the feedback message as part of a compound RTCPpacket.

• If feedback is sent, the next scheduled RTCP packet transmission time is

reconsidered on the basis of twice the standard interval. The receiver maynot send any more feedback until the reconsidered packet has been sent(that is, it may send a feedback packet once for each regular RTCP report).

The dither interval is chosen on the basis of the group size and the RTCP bandwidth.

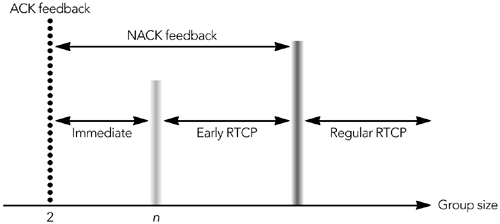
If the session has only two participants, the dither interval is set to zero; otherwise,it is set to half of the round-trip time between sender and receiver, multiplied by thenumber of members (if the round-trip time is unknown, it is set to half of the RTCPreporting interval).

The algorithm for choosing the dither interval allows each receiver to send feedbackalmost immediately for small sessions. As the number of receivers increases, therate at which each can send retransmission requests is reduced, but the chance thatanother receiver will see the same loss and send the same feedback increases.

**MODES OF OPERATION**

The RTP retransmission profile allows feedback to be sent at a higher rate thanstandard RTCP, but it still imposes some limitations on allowable send times.

Depending on the group size, bandwidth available, data rate, packet loss probability,

and desired reporting granularity, an application will operate in one of three

modes—immediate, early, and regular—which are illustrated in HTUFigure 9.13UTH.

**Figure 9.13. Modes of Feedback**

In Timmediate feedback modeT, there is sufficient bandwidth to send feedback foreach event of interest. In Tearly feedback modeT, there is not enough bandwidth toprovide feedback on all events, and the receiver has to report on a subset of thepossible events. Performance is best in immediate mode. As an application movesinto early feedback mode, it begins to rely on statistical sampling of the loss andgives only approximate feedback to the sender. The boundary between immediateand early modes, indicated by group size TnT in HTUFigure 9.13UTH, varies depending on thedata rate, group size, and fraction of senders.

In both immediate and early modes, only NACK packets are allowed. If the sessionhas only two participants, ACK mode can be used. In ACK mode, positive

acknowledgments of each event are sent, providing more detailed feedback to thesender (for example, ACK mode might allow a video application to acknowledgeeach complete frame, enabling reference picture selection to operate efficiently).

Again, the bandwidth limitations of the retransmission profile must be respected.

**Applicability**

The main factor that limits the applicability of retransmission is feedback delay. Ittakes at least one round-trip time for the retransmission request to reach the senderand for the retransmitted packet to reach the receiver. This delay can affect

interactive applications because the time taken for a retransmission may exceed theacceptable delay bounds. For streaming, and other applications in which the delaybounds are less strict, retransmission can be effective.HTPU69UTPH

Retransmission allows a receiver to request repair of only those packets that are lost,and allows it to accept loss of some packets. The result can be very efficient repair,given the right circumstances. But retransmission becomes inefficient in certaincases, such as these:

• Each retransmission request uses some bandwidth. When the loss rate is low,

the bandwidth used by the requests is low, but as losses become more

common, the amount of bandwidth consumed by requests increases.

• If the group is large and many receivers see the same loss, they may all

request retransmissions at once. Many requests use a lot of bandwidth, andthe implosion of requests may overwhelm the sender.

• If the group is large and each receiver sees a different loss, the sender will

have to retransmit most packets even though each receiver lost only a smallfraction of the packets.

Retransmission works best when groups are small and the loss rate is relatively low.

When the number of receivers, or the loss rate, increases, requesting

retransmission of lost packets rapidly becomes inefficient. Eventually, a cutoff isreached beyond which the use of forward error correction is more effective.

For example, Handley has observedHTPU122UTPH multicast groups in which TmostT packets arelost by at least one receiver. The result could be a retransmission request for almostevery packet, which would require tremendous overhead. If forward error

correction is used, each FEC packet repairs multiple losses, and the amount of repairdata that has to be sent is much lower.

The retransmitted packet does not have to be identical to the original. This flexibilityallows retransmission to be used in cases when it might otherwise be inefficient,because the sender can respond to requests by sending an FEC packet, rather thananother copy of the original.HTPU85UTPH The fact that the retransmitted and original packetsdo not have to be identical may also allow a single retransmission to repair manylosses.

**Implementation Considerations**

If error correction is used, an RTP implementation can be made significantly morerobust to the adverse effects of IP networks. These techniques come at a price,though: The implementation becomes somewhat more complex, with the receiverneeding a more sophisticated playout buffer algorithm, and the sender needinglogic to decide how much recovery data to include and when to discard that data.

**At a Receiver**

Use of these error correction techniques requires that the application have a moresophisticated playout buffer and channel-coding framework than it might otherwiseneed. In particular, it needs to incorporate FEC and/or retransmission delay into itsplayout point calculation, and it needs to allow for the presence of repair data inplayout buffers.

When calculating the playout point for the media, a receiver has to allow sufficienttime for the recovery data to arrive. This may mean delaying the playout of

audio/video beyond its natural time, depending on the time needed to receive therecovery data, and the desired playout point of the media.

For example, an interactive voice telephony application might want to operate witha short jitter buffer and a playout delay of only one or two packets&apos; worth of audio.

If the sender uses a parity FEC scheme such as that shown in HTUFigure 9.2UTH, in whichan FEC packet is sent after every four data packets, the FEC data will be uselessbecause it will arrive after the application has played out the original data it wasprotecting.

How does an application know when recovery data is going to arrive? In some casesthe configuration of the repair is fixed and can be signaled in advance, allowing thereceiver to size its playout buffers. Either the signaling can be implicit (for example,RFC 2198 redundancy in which the sender can insert zero-length redundant datainto the first few packets of an audio stream, allowing the receiver to know that realredundancy data will follow in later packets), or it can be explicit as part of sessionsetup (for example, included in the SDP during a SIP invitation).

Unfortunately, advance signaling is not always possible, because the repair schemecan change dynamically, or because the repair time cannot be known in advance(for example, when retransmission is used, the receiver has to measure the

round-trip time to the sender). In such cases it is the responsibility of the receiverto adapt to make the best use of any repair data it receives, by either delayingmedia playout or discarding repair data that arrives late. Generally the receivermust make such adaptation without the help of the sender, relying instead on itsown knowledge of the application scenario.

A receiver will need to buffer arriving repair data, along with the original media

packets. How this is done depends on the form of repair: Some schemes are weaklycoupled with the original media, and a generic channel-coding layer can be used;others are tightly coupled to the media and must be integrated with the codec.

Examples of weak coupling include parity FEC and retransmission in which repairscan be made by a general-purpose layer, with no knowledge of the contents of thepackets. The reason is that the repair operates on the RTP packets, rather than onthe media data itself.

In other cases the repair operation is tightly coupled with the media codec. Forexample, the AMR payload formatHTPU41UTPH includes support for partial checksums andredundant transmission. Unlike the audio redundancy defined in RFC 2198, thisform of redundant transmission has no separate header and is specific to AMR: Eachpacket contains multiple frames, overlapping in time with the following packet. Inthis case the AMR packetization code must be aware of the overlap, and it mustensure that the frames are correctly added to the playout buffer (and that

duplicates are discarded). Another example is the reference picture selection

available in MPEG-4 and some modes of H.263, in which the channel coding

depends on the shared state between encoder and decoder.

**At the Sender**

When error correction is in use, the sender is also required to buffer media datalonger than it normally would. The amount of buffering depends on the correctiontechnique in use: An FEC scheme requires the sender to hold on to enough data togenerate the FEC packets; a retransmission scheme requires the sender to hold onto the data until it is sure that the receivers will no longer request retransmission.

The sender has an advantage over the receiver when it comes to buffering becauseit knows the details of the repair scheme used and can size its buffers appropriately.

This is obviously the case when FEC is being used, but it is also true if retransmissionis in use (because RTCP allows the sender to calculate the round-trip time to eachreceiver).

The sender must also be aware of how its media stream is affecting the network.

Most techniques discussed in this chapter add additional information to a mediastream, which can then be used to repair loss. This approach necessarily increasesthe data rate of the stream. If the loss were due to network congestion—which is thecommon case in the public Internet—then this increase in data rate could lead to aworsening of the congestion, and could actually increase the packet loss rate. Toavoid these problems, error correction has to be tied to congestion control, which isthe subject of HTUChapter 10UTH.

**Summary**

In this chapter we have discussed various ways in which errors due to packet losscan be corrected. The schemes in use today include various types of forward errorcorrection and channel coding, as well as retransmission of lost packets.

When used correctly, error correction provides a significant benefit to the perceivedquality of a media stream, and it can make the difference between a system beingusable or not. If used incorrectly, however, it can lead to a worsening of the

problems it was intended to solve, and it can cause significant network problems.

The issue of congestion control—adapting the amount of data sent to match thenetwork capacity, as discussed in more detail in HTUChapter 10UTH, Congestion

Control—forms an essential counterpoint to the use of error correction.

One thing that should be clear from this chapter is that error correction usuallyworks by adding some redundancy to a media stream, which can be used to repairlost data. This mode of operation is somewhat at odds with the goal of media

compression, which seeks to remove redundancy from the stream. There is a

trade-off to be made between compression and error tolerance: At some stage,extra effort spent compressing a media stream is counterproductive, and it is betterto use the inherent redundancy for error resilience. Of course, the point at whichthat line is passed depends on the network, the codec, and the application.