**Chapter 11. Header Compression**

• Introductory Concepts

• Compressed RTP

• Robust Header Compression

• Considerations for RTP Applications

One area where RTP is often criticized is the size of its headers. Some argue that 12octets of RTP header plus 28 octets of UDP/IPv4 header is excessive for, say, a14-octet audio packet, and that a more efficient reduced header could be usedinstead. This is true to some extent, but the argument neglects the other uses ofRTP in which the complete header is necessary, and it neglects the benefit to thecommunity of having a single open standard for audio/video transport.

Header compression achieves a balance between these two worlds: When applied toa link layer, it can compress the TentireT 40-octet RTP/UDP/IP header into 2 octets,giving greater efficiency than a proprietary protocol over UDP/IP could achieve, withthe benefits of a single standard. This chapter provides an introduction to the

principles of header compression, and a closer look at two compression standards:Compressed RTP (CRTP) and Robust Header Compression (ROHC).

**Introductory Concepts**

The use of header compression has a well-established history in the Internet

community, since the definition of TCP/IP header compression in 1990HTPU4UTPH and itswidespread implementation along with PPP for dial-up links. More recently, the

standard for TCP/IP header compression has been revised and updated,HTPU25UTPH and newstandards for UDP/IP and RTP/UDP/IP header compression have been

developed.HTPU26UTP**H**,HP**T** U27UTP**H**,HP**T** U37UTPH

A typical scenario for header compression comprises a host connected to the

network via a low-speed dial-up modem or wireless link. The host and the first-hoprouter compress packets passing over the low-speed link, improving efficiency onthat link without affecting the rest of the network. In almost all cases, there is aparticular bottleneck link where it is desirable to use the bandwidth more efficiently,and there is no need for compression in the rest of the network. Header

compression works transparently, on a per-link basis, so it is well suited to thisscenario: The compressed link looks like any other IP link, and applications cannotdetect the presence of header compression.

These features—per-link operation and application transparency—mean that

header compression is usually implemented as part of the operating system (oftenas part of a PPP implementation). An application usually does not need to be awareof the presence of header compression, although in some circumstances,

consideration for the behavior of header compression can increase performancesignificantly. These circumstances are discussed in the section titled HTUConsiderationsfor RTP ApplicationsUTH later in this chapter.

**Patterns, Robustness, and Local Implementation**

Compression relies on patterns in the packet headers: Many fields are constant or

change in a predictable manner between consecutive packets belonging to the samepacket stream. If we can recognize these patterns, we can compress those fields toan indication that &quot;the header changed in the expected manner,&quot; rather than

explicitly sending them. Only header fields that change in an unpredictable mannerneed to be transmitted in every header.

An important principle in the design of header compression standards has been

robustness. There are two aspects to this: robustness to packet loss, and

robustness to misidentified streams. Network links—especially wireless links—canlose or corrupt packets, and the header compression scheme must be able to

function in the presence of such damage. The most important requirement is thatdamage to the compressed bit stream not cause undetectable corruption in theuncompressed stream. If a packet is damaged, the following packets will either bedecompressed correctly or discarded. The decompressor should never produce acorrupted packet.

The second robustness issue is that RTP flows are not self-identifying. The UDPheader contains no field indicating that the data being transported is RTP, and thereis no way for the compressor to determine unambiguously that a particular

sequence of UDP packets contains RTP traffic. The compressor needs to be informedexplicitly that a stream contains RTP, or it needs to be capable of making an

educated guess on the basis of observations of packet sequences. Robust

engineering requires that compression must not break anything if mistakenly

applied to a non-RTP flow. A misinformed compressor is not expected to compressother types of UDP flows, but it must not damage them.

Together, the principles of pattern recognition and robustness allow the formulationof a general principle for header compression: The compressor will send occasionalpackets containing full headers, followed by incremental updates indicating whichheader fields changed in the expected manner and containing the &quot;random&quot; fieldsthat cannot be compressed. The full headers provide robustness: Damage to theincremental packets may confuse the decompressor and prevent operation, but thiswill be corrected when the next full header arrives.

Finally, it is important that header compression be locally implementable. The aim isto develop a header compression scheme that operates over a single link withoutend-to-end support: If saving bandwidth is important, two systems can spendprocessor cycles to compress; otherwise, if processing is too expensive,

uncompressed headers are sent. If two systems decide to compress headers on asingle link, they should be able to do so in a manner that is invisible to all othersystems. As a consequence, header compression should be implemented in thenetwork protocol stack, not in the application. Applications are not expected toimplement header compression themselves; an application designer should

understand header compression and its consequences, but the implementationshould be done as part of the protocol stack of the machines on either side of thelink.

**Standards**

There are two standards for RTP header compression. The original Compressed RTP(CRTP) specification was designed for use with dial-up modems and is a

straightforward extension of TCP header compression to work with RTP/UDP/IP.

However, with the development of third-generation cellular technology, which usesvoice-over-IP as its bearer, it was found that CRTP does not perform well in

environments with high link delay and packet loss, so an alternative

protocol—Robust Header Compression (ROHC)—was developed.

CRTP remains the protocol of choice for dial-up use because of its simplicity and

good performance in that environment. ROHC is considerably more complex butperforms much better in a wireless environment. The following sections discusseach standard in detail.

**Compressed RTP**

The standard for CRTP is specified in RFC 2508.HTPU26UTPH It was designed for operationover low-speed serial links, such as dial-up modems, that exhibit low bit-error rates.

CRTP is a direct out-growth of the Van Jacobson header compression algorithm used

for TCP,HTPU4UTP**H**,HP**T** U25UTPH having similar performance and limitations.

The principal gain of TCP header compression comes from the observation that halfof the octets in the TCP/IP headers are constant between packets. These are sentonce—in a full header packet—and then elided from the following update packets.

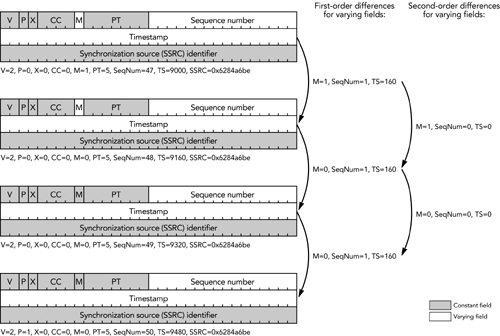
The remaining gain comes from differential coding on the changing fields to reducetheir size, and from eliminating the changing fields entirely for common cases bycalculating the changes from the length of the packet.

RTP header compression uses many of the same techniques, extended by theobservation that although several fields change in every packet, the difference frompacket to packet is often constant and therefore the second-order difference is zero.

The constant second-order difference allows the compressor to suppress the

unvarying fields and the fields that change predictably from packet to packet.

HTUFigure 11.1UTH shows the process in terms of just the RTP header fields. The shadedheader fields are constant between packets—they have a first-order difference ofzero—and do not need to be sent. The unshaded fields are varying fields; they havea nonzero first-order difference. However, their second-order difference is oftenconstant and zero, making the varying fields predictable.

**Figure 11.1. Principles of Header Compression**

In HTUFigure 11.1UTH all the fields except the M bit either are constant or have a zerosecond-order difference. Therefore, if the initial values for the predictable fields areknown, only the change in the M bit needs to be communicated.

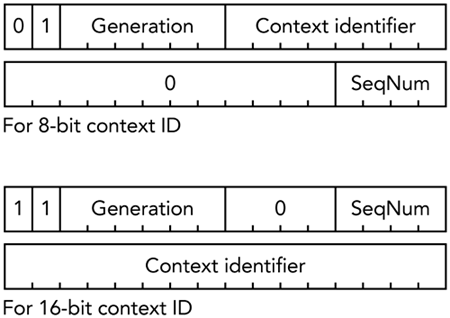
**Operation of CRTP: Initialization and Context**

Compressed RTP starts by sending an initial packet containing full headers, thereby

establishing the same state in the compressor and decompressor. This state is theinitial context of the compressed stream. Subsequent packets contain reducedheaders, which either indicate that the decompressor should use the existing

context to predict the headers, or contain updates to the context that must be usedfor the future packets. Periodically a full header packet can be sent to ensure thatany loss of synchronization between compressor and decompressor is corrected.

Full header packets comprise the uncompressed original packet, along with twoadditional pieces of information—a context identifier and a sequence number—asshown in HTUFigure 11.2UTH. The context identifier is an 8- or 16-bit field that uniquelyidentifies this particular stream. The sequence number is 4 bits and is used to detectpacket loss on the link. The additional fields replace the IP and UDP length fields inthe original packet, with no loss of data because the length fields are redundant withthe link-layer frame length. The full header packet format is common to severalcompression schemes—including IPv4 and IPv6, TCP, and UDP header

compression—hence the inclusion of the generation field, which is not used with RTPheader compression.

**Figure 11.2. Additional Fields Added to a Full Header Packet**

The link-layer protocol indicates to the transport layer that this is a CRTP (ratherthan IP) Tfull headerT packet. This information allows the transport layer to route thepackets to the CRTP decompressor instead of treating them as normal IP packets.

The means by which the link layer provides this indication depends on the type oflink in use. Operation over PPP links is specified in RFC 2509.HTPU27UTPH

Context identifiers are managed by the compressor, which generates a new context

whenever it sees a new stream that it believes it can compress. On receiving a newcontext identifier, the decompressor allocates storage for the context. Otherwise ituses the context identifier as an index into a table of stored context state. Contextidentifiers are not required to be allocated sequentially; an implementation thatexpects relatively few contexts should use a hash table to reduce the amount ofstorage space needed for the state, and one that expects many contexts shouldsimply use an array with the context identifier as an index.

The context contains the following state information, which is initialized when a fullheader packet is received:

• The full IP, UDP, and RTP headers, possibly including a CSRC list, for the last

packet transferred.

• The first-order difference for the IPv4 ID field, initialized to one when a full

header packet is received. (This information is not needed when

RTP-over-UDP/IPv6 is being compressed, because IPv6 does not have an IDfield.)

• The first-order difference for the RTP timestamp field, initialized to zero

when a full header packet is received.

• The last value of the four-bit CRTP sequence number, used to detect packet

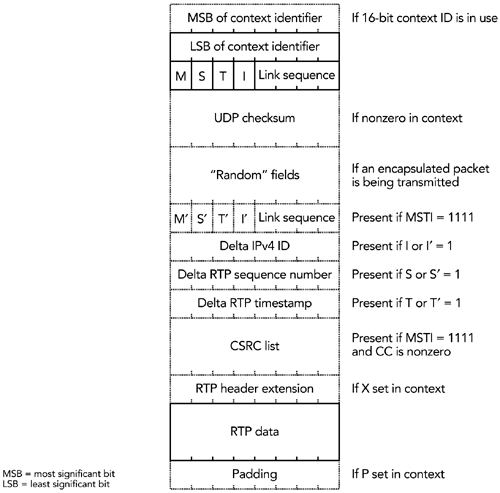
loss on the compressed link.

Given the context information, the receiver can decompress each consecutivepacket it receives. No additional state is needed, although the state stored in thecontext may need to be updated with each packet.

**Operation of CRTP: Compression and Decompression**

After a full header packet has been sent to establish the context, the transition to

TCompressed RTPT packets may occur. Each Compressed RTP packet indicates thatthe decompressor may predict the headers of the next packet on the basis of thestored context. Compressed RTP packets may update that context, allowing forcommon changes in the headers to be communicated without full header packetbeing sent. The format of a Compressed RTP packet is shown in HTUFigure 11.3UTH. Inmost cases, only the fields with solid outline are present, indicating that the nextheader may be directly predicted from the context. Other fields are present asneeded, to update the context, and their presence is either inferred from the contextor signaled directly within the compressed header.

**Figure 11.3. A Compressed RTP Packet**

The compressor observes the characteristics of the stream and omits fields that areconstant or that change in a predictable fashion. The compression algorithm willwork on any stream of packets in which the bits in the position of the IPv4 ID, RTPsequence number, and RTP timestamp are predictable. In general, the compressorhas no way of knowing whether a particular stream really is RTP; it must look forpatterns in the headers and, if they are present, start compression. If the stream isnot RTP, it is unlikely that the patterns will be present, so it will not be compressed(of course, if the stream is non-RTP but has the appropriate pattern, that the streamcan be compressed). The compressor is expected to keep state to track whichstreams are compressible and which are not, to avoid wasting compression effort.

On receiving a Compressed RTP packet, the decompressor reconstructs the headers.

You may reconstruct the IPv4 header by taking the header previously stored in thecontext, inferring the values of the header checksum and total length fields from thelink-layer headers. If the I or I´ bit in the Compressed RTP packet is set, the IPv4 IDis incremented by the delta IPv4 ID field in the Compressed RTP packet, and the

stored first-order difference in IPv4 ID in the context is updated. Otherwise the IPv4ID is incremented by the first-order difference stored in the context.

If multiple IPv4 headers are present in the context—for example, because of

IP-in-IP tunneling—their IPv4 ID fields are recovered from the Compressed RTPpacket, where they are stored in order as the &quot;random&quot; fields. If IPv6 is being used,there are no packet ID or header checksum fields, so all fields are constant exceptthe payload length, which may be inferred from the link-layer length.

You may reconstruct the UDP header by taking the header previously stored in thecontext and inferring the length field from the link-layer headers. If the checksumstored in the context is zero, it is assumed that the UDP checksum is not used.

Otherwise the Compressed RTP packet will contain the new value of the checksum.

You may reconstruct the RTP header by taking the header previously stored in thecontext, modified as described here:

• If all of M, S, T, and I are set to one, the packet contains a CC field and CSRC

list, along with M´, S´, T´, and I´ fields. In this case, the M´, S´, T´, and I´fields are used in predicting the marker bit, sequence number, timestamp,and IPv4 ID; and the CC field and CSRC list are updated on the basis of theCompressed RTP packet. Otherwise, the CC field and CSRC list are

reconstructed on the basis of the previous packet.

• The M bit is replaced by the M (or M´) bit in the Compressed RTP packet.

• If the S or S´ bit in the Compressed RTP packet is set, the RTP sequence

number is incremented by the delta RTP sequence field in the CompressedRTP packet. Otherwise the RTP sequence number is incremented by one.

• If the T or T´ bit in the Compressed RTP packet is set, the RTP timestamp is

incremented by the delta RTP timestamp field in the Compressed RTP packet,and the stored first-order difference in timestamp in the context is updated.

Otherwise the RTP timestamp is incremented by the stored first-order

difference from the context.

• If the X bit is set in the context, the header extension is recovered from the

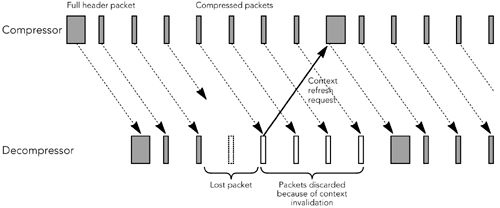
Compressed RTP packet.

• If the P bit is set in the context, the padding is recovered from the

Compressed RTP packet.

The context is updated with the newly received headers, as well as any updates tofirst-order difference in IPv4 ID and RTP timestamp, and the link-layer sequencenumber. The reconstructed headers, along with the payload data, are then passedto the IP stack for processing in the usual manner.

Typically only the context identifier, link sequence number, M, S, T, and I fields arepresent, giving a 2-octet header (3 octets if 16-bit context identifiers are used). Thiscompares well with the 40 octets of the uncompressed headers.

**Effects of Packet Loss**

An RTP receiver detects packet loss by a discontinuity in the RTP sequence number;

the RTP timestamp will also jump. When packet loss occurs upstream of the

compressor, new delta values are sent to the decompressor to communicate thesediscontinuities. Although the compression efficiency is reduced, the packet streamis communicated accurately across the link.

Similarly, if packet loss occurs on the compressed link, the loss is detected by thelink-layer sequence number in the Compressed RTP packets. The decompressorthen sends a message back to the compressor, indicating that it should send a fullheader packet to repair the state. If the UDP checksum is present, the

decompressor may also attempt to apply the deltas stored in the context twice, tosee whether doing so generates a correct packet. The twice algorithm is possibleonly if the UDP checksum is included; otherwise the decompressor has no way ofknowing whether the recovered packet is correct.

The need to request a full header packet to recover the context when loss occursmakes CRTP highly susceptible to packet loss on the compressed link. In particular,if the link round-trip time is high, it is possible that many packets are being receivedwhile the context recovery request is being delivered. As shown in HTUFigure 11.4UTH, theresult is a loss multiplier effect, in which a single loss affects multiple packets untila full header packet is delivered. Thus, packet loss can seriously affect the

performance of CRTP.

**Figure 11.4. CRTP Operation When Packet Loss Occurs**

The other limiting factor of CRTP is packet reordering. If packets are reordered priorto the compressed link, the compressor is required to send packets containing bothsequence number and timestamp updates to compensate. These packets are

relatively large—commonly between two and four octets—and they have a

significant effect on the compression ratio, at least doubling the size of the

compressed headers.

Basic CRTP assumes that there is no reordering on the compressed link. Work isunder way on an extension to make CRTP more robust in the face of packet loss andreordering.HTPU43UTPH It is expected that these enhancements will make CRTP suitable forenvironments with low to moderate amounts of loss or reordering over the

compressed link. The Robust Header Compression scheme described in the nextsection is designed for more extreme environments.

**Robust Header Compression**

As noted earlier, CRTP does not work well over links with loss and long round-trip

times, such as many cellular radio links. Each lost packet causes several subsequentpackets to be lost because the context is out of sync during at least one link

round-trip time.

In addition to reducing the quality of the media stream, the loss of multiple packetswastes bandwidth because some packets that have been sent are simply discarded,and because a full header packet must be sent to refresh the context. Robust

Header Compression (ROHC)HTPU37UTPH was designed to solve these problems, providingcompression suitable for use with third-generation cellular systems. ROHC getssignificantly better performance than CRTP over such links, at the expense of

additional complexity of implementation.

Observation of the changes in header fields within a media stream shows that theyfall into three categories:

1. Some fields are static, or mostly static. Examples include the RTP SSRC, UDP

ports, and IP addresses. These fields can be sent once when the connectionis established, and either they never change or they change very

infrequently.

2. Some fields change in a predictable manner with each packet sent, except

for occasional sudden changes. Examples include the RTP timestamp andsequence number, and (often) the IPv4 ID field. During periods when thesefields are predictable, there is usually a constant relation between them.

When sudden changes occur, often only a single field changes unpredictably.

3. Some fields are unpredictable, having essentially random values, and have

to be communicated as is, with no compression. The main example is theUDP checksum.

ROHC operates by establishing mapping functions between the RTP sequence

number and the other predictable fields, then reliably transferring the RTP sequencenumber and the unpredictable header fields. These mapping functions form part of

the compression context, along with the values of static fields that are

communicated at startup or when those fields change.

The main differences between ROHC and CRTP come from the way they handle thesecond category: fields that usually change in a predictable manner. In CRTP, thevalue of the field is implicit and the packet contains an indication that it changed inthe predictable fashion. In ROHC, the value of a single key field—the RTP sequencenumber—is explicitly included in all packets, and an implicit mapping function isused to derive the other fields.

**Operation of ROHC: States and Modes**

ROHC has three states of operation, depending on how much context has beentransferred:

1. The system starts in initialization and refresh state, much like the full header

mode of CRTP. This state conveys the necessary information to set up thecontext, enabling the system to enter first- or second-order compressionstate.

2. First-order compression state allows the system to efficiently communicate

irregularities in the media stream—changes in the context—while still

keeping much of the compression efficiency. In this state, only a compressedrepresentation of the RTP sequence number, along with the context

identifier, and a reduced representation of the changed fields are conveyed.

3. Second-order state is the highest compression level, when the entire header

is predictable from the RTP sequence number and stored context. Only acompressed representation of the RTP sequence number and a (possiblyimplicit) context identifier are included in the packet, giving a header thatcan be as small as one octet.

If any unpredictable fields are present, such as the UDP checksum, then both first-and second-order compression schemes communicate those fields unchanged. Asexpected, the result is a significant reduction in the compression efficiency. Forsimplicity, this description omits further mention of these fields, although they willalways be conveyed.

The compressor starts in initialization and refresh state, sending full headers to thedecompressor. It will move to either first- or second-order state, sending

compressed headers, after it is reasonably sure that the decompressor has correctlyreceived enough information to set up the context.

The system can operate in one of three modes: unidirectional, bidirectional

optimistic, and bidirectional reliable. Depending on the mode chosen, the

compressor will transition from the initialization and refresh state to either first- orsecond-order compression state, according to a timeout or an acknowledgment:

It&apos;s important to keep the difference between ROHC states and modes clear. The

state determines the type of information sent in each packet: full headers, partial

updates, or fully compressed. The mode determines how and when feedback is

sent from the decompressor: (1) never, (2) when there is a problem, or (3)

always.

1. T**Unidirectional mode**T. No feedback is possible, and the compressor

transitions to first- or second-order state after a predetermined number ofpackets has been sent.

2. T**Bidirectional optimistic mode**T. The compressor transitions to the first- or

second-order state after a predetermined number of packets has been sent,much as in unidirectional mode, or when an acknowledgment is received.

3. T**Bidirectional reliable mode**T. The compressor transitions to the first- or

second-order state on receipt of an acknowledgment.

The choice of unidirectional or bidirectional feedback depends on the characteristicsof the link between compressor and decompressor. Some network links may notsupport a (convenient) back channel for feedback messages, forcing unidirectionaloperation. In most cases, though, one of the bidirectional modes can be used,allowing the receiver to communicate its state to the sender.

The compressor starts by assuming unidirectional operation. The decompressor willchoose to send feedback if the link supports it, depending on the loss patterns of thelink. Receipt of feedback messages informs the compressor that bidirectional

operation is desired. The choice between optimistic and reliable mode is made bythe decompressor and depends on the capacity of the back channel and the losscharacteristics of the link. Reliable mode sends more feedback but is more tolerantof loss.

Typically the system transitions from initialization and refresh state to the

second-order state after context has been established. It then remains in

second-order state until loss occurs, or until a context update is needed because ofchanges in the stream characteristics.

If loss occurs, the system&apos;s behavior depends on the mode of operation. If one ofthe bidirectional modes was chosen, the decompressor will send feedback causingthe compressor to enter the first-order state and send updates to repair the context.

This process corresponds to the sending of a context refresh message in CRTP,causing the compressor to generate a full header packet. If unidirectional mode isused, the compressor will periodically transition to lower states, to refresh thecontext at the decompressor.

The compressor will also transition to the first-order state when it is necessary toconvey a change in the mapping for one of the predictable fields, or an update toone of the static fields. This process corresponds to the sending of a compressedpacket containing an updated delta field or a full header packet in CRTP. Dependingon the mode of operation, the change to first-order state may cause the

decompressor to send feedback indicating that it has correctly received the newcontext.

**Operation of ROHC: Robustness and Compression Efficiency**

If the compressed link is reliable, ROHC and CRTP have similar compression

efficiency, although ROHC is somewhat more complex. For this reason, dial-upmodem links do not typically use ROHC, because CRTP is less complex and yieldscomparable performance.

When there is packet loss on the compressed link, the performance of ROHC showsbecause of its flexibility in sending partial context updates, and its robust encodingof compressed values. The capability to send partial context updates allows ROHC toupdate the context in cases in which CRTP would have to send a full header packet.

ROHC can also reduce the size of a context update when there is loss on the link.

Both of these capabilities give improved performance, compared to CRTP.

The combination of robust encoding of compressed values and sequence

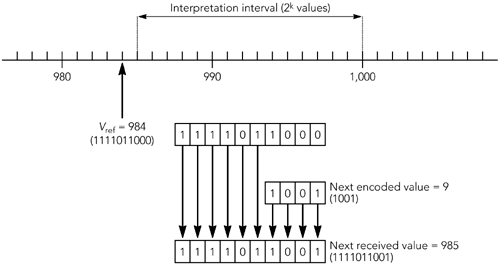
number–driven operation is also a key factor. As noted earlier, the ROHC contextcontains a mapping between the RTP sequence number and the other predictableheader fields. Second-order compressed packets convey the sequence numberusing a window-based least-significant bit (W-LSB) encoding, and the other fieldsare derived from this. It is largely the use of W-LSB encoding that gives ROHC itsrobustness to packet loss.

Standard LSB encoding transmits the TkT least-significant bits of the field value,instead of the complete field. On receiving these TkT bits, and given the previouslytransmitted value TVTBrefB, the decompressor can derive the original value of the field,provided that it is within a range known as the Tinterpretation intervalT.

The interpretation interval is the 2PkP values surrounding TVTBrefB, offset by a parameterTpT so that it covers the range TVTBrefB – TpT to TVTBrefB + 2PkP – 1 – TpT. The parameter TpT is chosenon the basis of the characteristics of the field being transported and conveyed to thedecompressor during initialization, forming part of the context. Possible choicesinclude the following:

• If the field value is expected to increase, TpT = –1.

• If the field value is expected to increase or stay the same, TpT = 0.

• If the field value is expected to vary slightly from a fixed value, TpT = 2P(k–1)P +

1.

• If the field value is expected to undergo small negative changes and large

positive changes—for example, the RTP timestamp of a video stream usingB-frames—then TpT = 2P(k–2)P – 1.

As an example, consider the transport of sequence numbers, in which the last

transmitted value TVTBrefB = 984 and TkT = 4 least-significant bits are sent as the encodedform. Assume also that TpT = –1, giving an interpretation interval ranging between985 and 1,000, as shown in HTUFigure 11.5UTH. The next value sent is 985 (in binary:1111011001), which is encoded as 9 (in binary: 1001, the 4 least-significant bits ofthe original value). On receiving the encoded value, the decompressor takes TVTBrefBand replaces the TkT least-signifi-cant bits with those received, restoring the originalvalue.

**Figure 11.5. An Example of LSB Encoding**

LSB encoding will work provided that the encoded value is within the interpretationinterval. If a single packet were lost in the preceding example and the next valuereceived by the decoder were 10 (in binary: 1010), then the restored value would be986, which is correct. If more than 2PkP packets were lost, however, the decoderwould have no way of knowing the correct value to decode.

The window-based variant of LSB encoding, W-LSB, maintains the interpretationinterval as a sliding window, advancing when the compressor is reasonably surethat the decompressor has received a particular value. Confidence that the windowcan advance is obtained by various means: In bidirectional optimistic mode, thedecompressor sends acknowledgments; in bidirectional optimistic mode, the

window advances after a period of time, unless the decompressor sends a negative

acknowledgment; and in unidirectional mode, the window simply advances after aperiod of time.

The advantage of W-LSB encoding is that loss of a small number of packets withinthe window will not cause the decompressor to lose synchronization. This

robustness allows an ROHC decompressor to continue operation without requestingfeedback in cases when a CRTP decompressor would fail and need a context update.

The result is that ROHC is much less susceptible to the loss multiplier effect thanCRTP: A single packet loss on the link will cause a single loss at the output of a ROHC

decompressor, whereas a CRTP decompressor must often to wait for a contextupdate before it can continue decompression.

**Considerations for RTP Applications**

RTP header compression—whether by CRTP or ROHC—is transparent to the

application. When header compression is in use, the compressed link becomes amore efficient conduit for RTP packets, but aside from increased performance, anapplication should not be able to tell that compression is being used.

Nevertheless, there are ways in which an application can aid the operation of thecompressor. The main idea is regularity: An application that sends packets withregular timestamp increments, and with a constant payload type, will produce astream of RTP packets that compresses well, whereas variation in the payload typeor interpacket interval will reduce the compression efficiency. Common causes ofvariation in the interpacket timing include silence suppression with audio codecs,reverse predicted video frames, and interleaving:

• Audio codecs that suppress packets during silent periods affect header

compression in two ways: They cause the marker bit to be set, and theycause a jump in the RTP timestamp. These changes cause CRTP to send apacket containing an updated timestamp delta; ROHC will send a first-orderpacket containing the marker bit and a new timestamp mapping. Both

approaches add at least one octet to the size of the compressed header.

Despite this reduction in header compression efficiency, silence suppressionalmost always results in a net saving in bandwidth because some packets arenot sent.

• Reverse predicted video frames—for example, MPEG B-frames—have a

timestamp less than that of the previous frame.HTPU12UTPH As a result, CRTP sendsmultiple timestamp updates, seriously reducing compression efficiency. Theeffects on ROHC are less severe, although some reduction in compressionefficiency occurs there also.

• Interleaving is often implemented within the RTP payload headers, with the

format designed so that the RTP timestamp increment is constant. In thiscase, interleaving does not affect header compression, and it may even be

beneficial. For example, CRTP has a loss multiplier effect when operating onhigh-delay links, which is less of an issue for inter-leaved streams than fornoninterleaved streams. In some cases, though, interleaving can result inpackets that have RTP timestamps with nonconstant offsets. Thus,

interleaving will reduce the compression efficiency and is best avoided.

The use of UDP checksums also affects compression efficiency. When enabled, theUDP checksum must be communicated along with each packet. This adds two octetsto the compressed header, which, because fully compressed RTP/UDP/IP headersare two octets for CRTP and one octet for ROHC, is a significant increase.

The implication is that an application intended to improve compression efficiencyshould disable the checksum, but this may not necessarily be appropriate. Disablingthe checksum improves the compression ratio, but it may make the stream

susceptible to undetected packet corruption (depending on the link layer; somelinks include a checksum that makes the UDP checksum redundant). An applicationdesigner must decide whether the potential for receiving corrupt packets outweighsthe gain due to improved compression. On a wired network it is often safe to turn offthe checksum because bit errors are rare. However, wireless networks often have arelatively high bit-error rate, and applications that may be used over a wireless linkmight want to enable the checksum.

The final factor that may affect the operation of header compression is generation ofthe IPv4 ID field. Some systems increment the IPv4 ID by one for each packet sent,allowing for efficient compression. Others use a pseudorandom sequence of IPv4 IDvalues, making the sequence unpredictable in an attempt to avoid certain securityproblems. The use of unpredictable IPv4 ID values significantly reduces the

compression efficiency because it is necessary to convey the two-octet IPv4 ID inevery packet, rather than allowing it to be predicted. It is recommended that IPimplementations increment the IPv4 ID by one when sending RTP packets, althoughit is recognized that the IP layer will not typically know the contents of the packets(an implementation might provide a call to inform the system that the IPv4 IDshould increment uniformly for a particular socket).

**Summary**

The use of RTP header compression—whether CRTP or ROHC—can significantlyimprove performance when RTP is operating over low-speed network links. Whenthe payload is small—say, low-rate audio, with packets several tens of octets inlength—the efficiency to be gained from CRTP with 2-octet compressed headers,compared to 40-octet uncompressed headers, is significant. The use of ROHC canachieve even greater gains in some environments, at the expense of additionalcomplexity.

Header compression is beginning to be widely deployed. ROHC is an essential partof third-generation cellular telephony systems, which use RTP as their voice bearerchannel. CRTP is widely implemented in routers and is beginning to be deployed inend hosts.