1

Multimedia Over Packet

1.1 TRANSPORTING VOICE, FAX, AND VIDEO OVER A PACKET NETWORK

1.1.1 A Darwinian view of voice transport

1.1.1.1 The circuit switched network

The most common telephone system on the planet today is still analog, especially at the edge of the network. Analog telephony (Figure 1.1) uses the modulation of electric signals along a wire to transport voice.

Although it is a very old technology, analog transmission has many advantages: it is simple and keeps the end-to-end delay of voice transmission very low because the signal propagates along the wire almost at the speed of light.

It is also inexpensive when there are relatively few users talking at the same time and when they are not too far apart. But the most basic analogue technology requires one pair of wires per active conversation, which becomes rapidly unpractical and expensive. The first improvement to the basic 'baseband' analog technology involved multiplexing several conversations on the same wire, using a separate transport frequency for each signal. But even with this hack, analog telephony has many drawbacks:

- Unless you use manual switchboards, analog switches require a lot of electromechanical gear, which is expensive to buy and maintain.
- Parasitic noise adds up at all stages of the transmission because there is no way to differentiate the signal from the noise and the signal cannot be cleaned.

For all these reasons, most countries today use digital technology for their core telephone network and sometimes even at the edge (e.g., ISDN). In most cases the subscriber line remains analogue, but the analogue signal is converted to a digital data stream in the

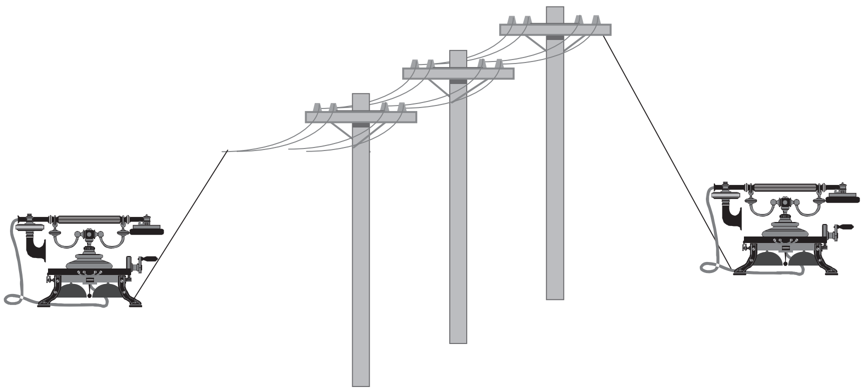

Figure 1.1 Analog telephony, as old as the invention of the telephone, and still in use today at the edge of the network.

first local exchange. Usually, this signal has a bitrate of 64 kbit/s or 56 kbit/s (one sample every 125 µs).

With this digital technology, many voice channels can easily be multiplexed along the same transmission line using a technology called **time division multiplexing (TDM)**. In this technology, the digital data stream which represents a single conversation is divided into blocks (usually an octet), and blocks from several conversations are interleaved in a round robin fashion in the time slots of the transmission line, as shown in Figure 1.2.

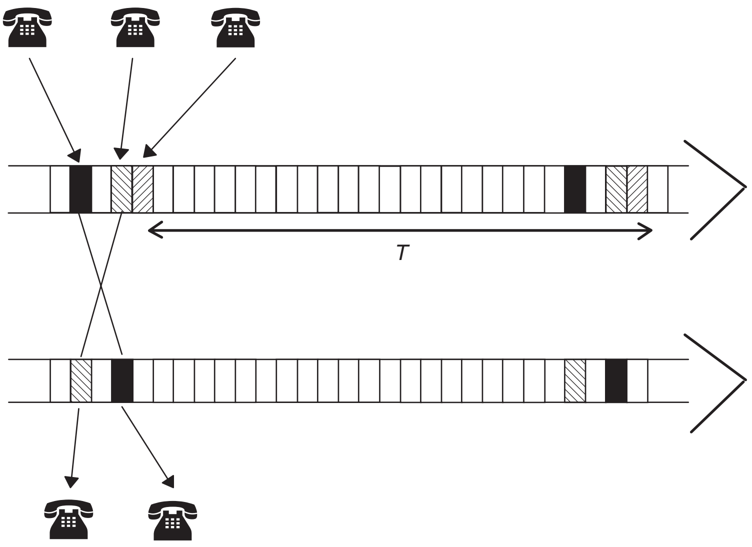

Figure 1.2 TDM switching.

Because of digital technology, the noise that is added in the backbone does not influence the quality of the communication because digital ‘bits’ can be recognized exactly, even in the presence of significant noise. Moreover, digital TDM makes digital switching possible. The switch just needs to copy the contents of one time slot of the incoming transmission line into another time slot in the outgoing transmission line. Therefore, this switching function can be performed by computers.

However, a small delay is now introduced by each switch, because for each conversation a time slot is only available every T µs, and in some cases it may be necessary to wait up to T µs to copy the contents of one time slot into another. Since T equals 125 µs in all digital telephony networks, this is usually negligible and the main delay factor is simply the propagation time.

1.1.1.2 Asynchronous transmission and statistical multiplexing

Unless you really have a point to make, or you’re a politician, you will usually speak only half of the time during a conversation. Since we all need to think a little before we reply, each party usually talks only 35% of the time during an average conversation.

If you could press a button each time you talk, then you would send data over the phone line only when you actually say something, not when you are silent. In fact, most of the techniques used to transform your voice into data (known as codecs) now have the ability to detect silence. With this technique, known as voice activity detection (VAD), instead of transmitting a chunk of data, voice, or silence every 125 µs, as done today on TDM networks, you only transmit data when you need to, asynchronously, as illustrated in Figure 1.3.

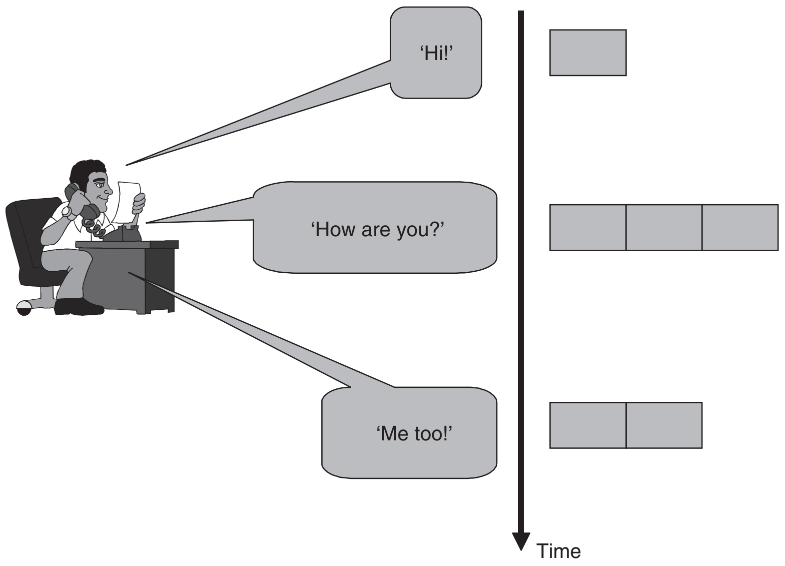

Figure 1.3 Transmitting voice asynchronously.

And when it comes to multiplexing several conversations on a single transmission line, instead of occupying a fraction of bandwidth all the time, ‘your’ bandwidth can be used by someone else while you are silent. This is known as ‘statistical multiplexing’.

The main advantage of **statistical multiplexing** is that it allows the bandwidth to be used more efficiently, especially when there are many conversations multiplexed on the same line (see companion book, *Beyond VoIP protocols* Chapter 5 for more details). But statistical multiplexing, as the name suggests, introduces uncertainty in the network. As just mentioned, in the case of TDM a delay of up to 125 μs could be introduced at each switch; this delay is constant throughout the conversation. The situation is totally different with statistical multiplexing (Figure 1.4): if the transmission line is empty when you need to send a chunk of data, it will go through immediately. If on the other hand the line is full, you may have to wait until there is some spare capacity for you.

This varying delay is caller **jitter**, and needs to be corrected by the receiving side. Otherwise, if the data chunks are played as soon as they are received, the original speech can become unintelligible (see Figure 1.5).

The next generation telephone networks will use statistical multiplexing, and mix voice and data along the same transmission lines. Several technologies are good candidates (e.g., voice over frame relay, voice over ATM, and, of course, voice over IP).

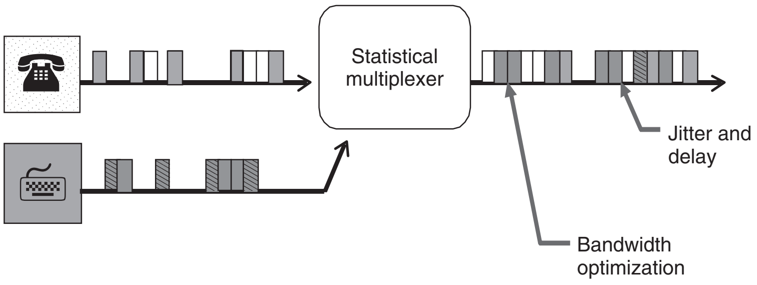

Figure 1.4 Statistical multiplexers optimize the use of bandwidth but introduce jitter.

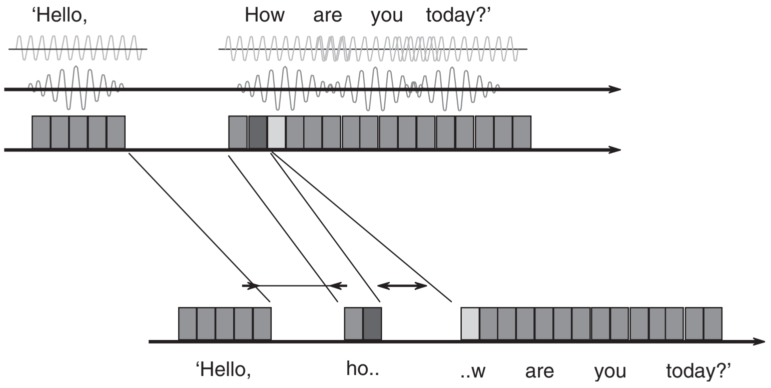

Figure 1.5 Effects of uncompensated jitter.

We believe voice over IP is the most flexible solution, because it does not require setting up virtual channels between the sites that will communicate. VoIP networks scale much better than ATM or frame relay networks, and VoIP also allows communications to be established directly with VoIP endpoints: there is now a variety of **IP-PBXs** (private switches with a VoIP wide-area network interface), or IP phones on the market today that have no ATM or frame relay equivalent.

1.1.2 Voice and video over IP with RTP and RTCP

The Real-time Transport Protocol and Real-time Control Protocol, described in RFC 3550, are the protocols that have been used for the transport of media streams since the first conferencing tools were made available on the Internet. The **visual audio tool (VAT)** used RTP version 0. A description of version 1 is available at ftp://gaia.cs.umass.edu/pub/hgschulz/rtp/draft-ietf-avt-rtp-04.txt

Since then, RTP has evolved into version 2. RTPv2 is not backward compatible with version 1, and therefore all applications should be built to support RTPv2.

1.1.2.1 Why RTP/RTCP?

When a network using statistical multiplexing is used to transmit real-time data such as voice, jitter has to be taken into account by the receiver. Routers are good examples of such statistical multiplexing devices, and therefore voice and video over IP have to face the issue of jitter.

RTP was designed to allow receivers to compensate for jitter and desequencing introduced by IP networks. RTP can be used for any real-time (or more rigorously isochronous) stream of data (e.g., voice and video). RTP defines a means of formatting the payload of IP packets carrying real-time data. It includes:

- Information on the type of data transported (the ‘payload’).
- Timestamps.
- Sequence numbers.

Another protocol, RTCP, is very often used with RTP. RTCP carries some feedback on the quality of the transmission (the amount of jitter, the average packet loss, etc.) and some information on the identity of the participants as well.

RTP and RTCP do not have any influence on the behavior of the IP network and do not control quality of service in any way. The network can drop, delay, or desequence an RTP packet like any other IP packet. RTP must not be mixed up with protocols like **RSVP** (Resource Reservation Protocol). RTP and RTCP simply allow receivers to recover from network jitter and other problems by appropriate buffering and sequencing, and to have more information on the network so that appropriate corrective measures can be adopted (redundancy, lower rate codecs, etc.). However, some routers are actually able to parse

IP packets, discover whether these packets have RTP headers, and give these packets a greater priority, resulting in better QoS even without any external QoS mechanism, such as RSVP for instance. Most Cisco® routers support the `IP RTP PRIORITY` command.

RTP and RTCP are designed to be used on top of any transport protocol that provides framing (i.e., delineates the beginning and end of the information transported), over any network. However, RTP and RTCP are mostly used on top of **UDP (User Datagram Protocol)**.[1] In this case, RTP is traditionally assigned an even UDP port and RTCP the next odd UDP port.[2]

1.1.2.2 RTP

RTP allows the transport of isochronous data across a packet network, which introduces jitter and can desequence the packets. Isochronous data are data that need to be rendered with exactly the same relative timing as when they were captured. Voice is the perfect example of isochronous data; any difference in the timing of the playback will either create holes or truncate some words. Video is also a good example, although tolerances for video are a lot higher; delays will only result in some parts of the screen being updated a little later, which is visible only if there has been a significant change.

RTP is typically used on top of UDP. UDP is the most widely used 'unreliable' transport protocol for IP networks. UDP can only guarantee data integrity by using a checksum, but an application using UDP has to take care of any data recovery task. UDP also provides the notion of a '**port**', which is a number between 0 and 65,535 (present in every packet as part of the destination address) which allows up to 65,536 UDP targets to be distinguished at the same destination IP address. A port is also attached to the source address and allows up to 65,536 sources to be distinguished from the same IP address. For instance, an RTP over UDP stream can be sent from 10.10.10.10:2100 to 10.10.10.20:3200:

Source IP address: 10.10.10.10	Source port: 2100	Destination IP address: 10.10.10.20	Destination port: 3200	RTP Data

When RTP is carried over UDP, it can be carried by multicast IP packets, that is, packets with a multicast destination address (e.g. 224.34.54.23): therefore an RTP stream generated by a single source can reach several destinations; it will be duplicated as necessary by the IP network. (See companion book, *Beyond VoIP Protocols*, Chapter 6. IP multicast routing).

1.1.2.2.1 A few definitions

- **RTP session**: an RTP session is an association of participants who communicate over RTP. Each participant uses at least two transport addresses (e.g., two UDP ports on the

[1] For streaming (e.g., RTSP), since there are is no real-time constraint on transmission delay, it can be used over TCP.

[2] But this is not mandatory, especially when RTP/RTCP ports are conveyed by an out-of-band signaling mechanism.

local machine) for each session: one for the RTP stream, one for the RTCP reports. When a multicast transmission is used all the participants use the same pair of multicast transport addresses. Media streams in the same session should share a common RTCP channel. Note that H.323 or SIP require applications to define explicitly a port for each media channel. So, although most applications comply with the RTP requirements for RTP and RTCP port sharing as well as the use of adjacent ports for RTP and RTCP, an application should *never* make an assumption about the allocation of RTP/RTCP ports, but rather use the explicit information provided by H.323 or SIP, even if it does not follow the RTP RFC guidelines. This is one of the most common bugs still found today in some H.323 or SIP applications.

- **Synchronization source (SSRC)**: identifies the source of an RTP stream, identified by 32 bits in the RTP header. All RTP packets with a common SSRC have a common time and sequencing reference. Each sender needs to have an SSRC; each receiver also needs at least one SSRC as this information is used for **receiver reports** (RRs).
- **Contributing source (CSRC)**: when an RTP stream is the result of a combination put together by an RTP mixer from several contributing streams, the list of the SSRCs of each contributing stream is added in the RTP header of the resulting stream as CSRCs. The resulting stream has its own SSRC. This feature is not used in H.323 or SIP.
- **NTP format**: a standard way to format a timestamp, by writing the number of seconds since 1/1/1900 at 0 h with 32 bits for the integer part and 32 bits for the decimal part (expressed in $\frac{1}{2^{32}}$ s (e.g., $0 \times 800\,000\,00$ is 0.5 s). A compact format also exists with only 16 bits for the integer part and 16 bits for the decimal part. The first 16 digits of the integer part can usually be derived from the current day, the fractional part is simply truncated to the most significant 16 digits.

1.1.2.2.2 *The RTP packet*

All fields up to the CSRC list are always present in an RTP packet (see Figure 1.6). The CSRC list may only be present behind a mixer (a device that mixes RTP streams, as defined in the RTP RFC). In practice, most conferencing bridges that perform the function of a mixer (H.323 calls them ‘multipoint processors’, or MPs) do not populate the CSRC list.

Here is a short explanation of each RTP field:

- Two bits are reserved for the **RTP version**, which is now version 2 (10). Version 0 was used by VAT and version 1 was an earlier IETF draft.
- A **padding bit P** indicates whether the payload has been padded for alignment purposes. If it has been padded (P = 1), then the last octet of the payload field indicates more precisely how many padding octets have been appended to the original payload.
- An **extension bit X** indicates the presence of extensions after the eventual CSRCs of the fixed header. Extensions use the format shown in Figure 1.7.
- The 4-bit **CSRC count** (CC) states how many CSRC identifiers follow the fixed header. There is usually none.

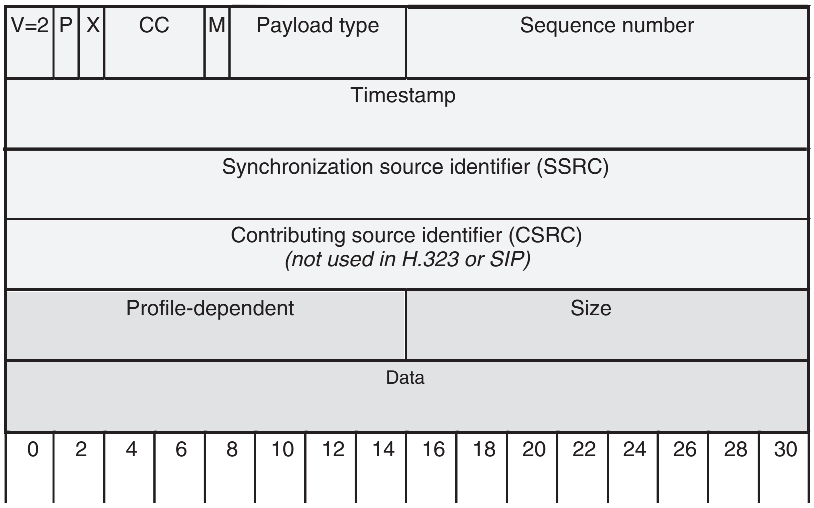

Figure 1.6 RTP packet format.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        defined by profile       |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           header extension                      |
|                               ....                              |
```

Figure 1.7 Optional extension header.

- **Marker (M)**: 1 bit. Its use is defined by the RTP profile. H.225.0 says that for audio codings that support silence suppression, it must be set to 1 in the first packet of each talkspurt after a silence period. This may allow some implementations to dynamically reduce the jitter buffer size without running the risk of cutting important words (e.g., by trimming off some silence packets).
- **Payload type (PT)**: 7 bits. The payload of each RTP packet is the real-time information contained in the packet. Its format is completely free and must be defined by the application or the profile of RTP in use. It enables applications to distinguish a particular format from another without having to analyse the content of the payload. Some common identifiers are listed in Table 1.1; they are used by H.225 and SIP. These are called **static payload types** and are assigned by **IANA (Internet Assigned Numbers Authority)**; a list can be found at http://www.isi.edu/in-notes/iana/assignments/rtp-; parameters PT 96 to 127 are reserved for **dynamic payload types**. Dynamic payload types are defined in the RTP audio-visual (A/V) profile and are not assigned in the IANA list. The dynamic PT meaning is defined only for the duration of the session. The exact meaning of the dynamic payload type is defined through some out-of-band

Table 1.1 Common static payload types

Payload type	Codec	
0	PCM, μ-law	Audio
8	PCM, A-law	
9	G.722	
4	G.723	
15	G.728	
18	G.729	
34	H.263	Video
31	H.261	

mechanism (e.g., though Session Description Protocol parameters for protocols like SIP, H.245 OpenLogicalChannel parameters for H.323, or through some convention or other mechanism defined by the application). The codec associated with a dynamic PT is negotiated by the conference control protocol dynamically. Since RTP itself doesn't define the format of the payload section, each application must define or refer to a profile. In the case of H.323, this work is done in annex B of H.225.

- **A sequence number and timestamp**: the 16-bit sequence number and timestamp start on a random value and are incremented at each RTP packet. The 32-bit timestamp uses a clock frequency that is defined for each payload type (e.g., H.261 payload uses a 90-kHz clock for the RTP timestamp). For narrow-band audio codecs (G.711, G.723.1, G.729, etc.) the RTP clock frequency is set to 8,000 Hz. For video, the RTP timestamp is the tick count of the display time of the first frame encoded in the packet payload. For audio, the RTP timestamp is the tick count when the fist audio sample contained in the payload was sampled. Each RTP packet carries a sequence number and a timestamp. RTP timestamps do not have an absolute meaning (the initial timestamps of an RTP stream can be selected at random); even timestamps of related media (e.g., audio and video) in a single session will be unrelated. In order to map RTP packet timestamps to absolute time, one must use the information held in RTCP sender reports, where RTP timestamps are associated with the absolute NTP time. Depending on the application, timestamps can be used in a number of ways. A video application, for instance, will use it to synchronize audio and data. An audio application will use the sequence number and timestamp to manage a reception buffer. For instance, an application can decide to buffer 20 10-ms G.729 audio frames before commencing playback. Each time a new RTP packet arrives, it is placed in the buffer in the appropriate position depending on its sequence number. It is important to note that the protection against jitter allowed by RTP comes with a price: a greater end-to-end delay in the transmission path. If a packet doesn't arrive on time and is still missing at playback time, the application can decide to copy the last sample of the packet that has just been played and repeat it long enough to catch up with the timestamp of the next received packet, or use some interpolation scheme as defined by the particular audio codec in use. The sequence number is used to detect packet loss.

1.1.2.3 RTCP

RTCP is used to transmit control packets to participants regarding a particular RTP session. These control packets include various statistics, information about the participants (their names, email addresses, etc.), and information on the mapping of participants to individual stream sources. The most useful information found in RTCP packets concerns the quality of transmission in the network. All participants in the sessions send RTCP packets: senders send '*sender* reports' and receivers send '*receiver* reports'.

1.1.2.3.1 Bandwidth limitation

All participants must send RTCP packets. This causes a potential dimensioning problem for large multicast conferences: RTCP traffic should grow linearly with the number of participants. This problem does not exist with RTP streams in audio-only conferences using silence suppression, for instance, since people generally don't speak at the same time (Figure 1.8).

Since the number of participants is known to all participants who listen to RTCP reports, each of them can control the rate at which RTCP reports are sent. This is used to limit the bandwidth used by RTCP to a reasonable amount, usually not more than 5% of the overall session bandwidth (which is defined as the sum of all transmissions from all participants, including the IP/UDP overhead).

This budget has to be shared by all participants. Active senders get one-quarter of it because some of the information they send (e.g., CNAME information used for synchronization) is very important to all receivers and RTCP sender reports need to be very responsive. The remaining part is split between the receivers. The average sending rate is derived by the participant from the size of the RTCP packets that he wants to send and from the number of senders and receivers that appear in the RTCP packets it receives. This is clearly relevant for multicast sessions; in fact, many of the recommendations and features present in the RTP RFC are useless for most VoIP applications, which have a maximum of three participants in most cases. Even for small sessions, the fastest rate at

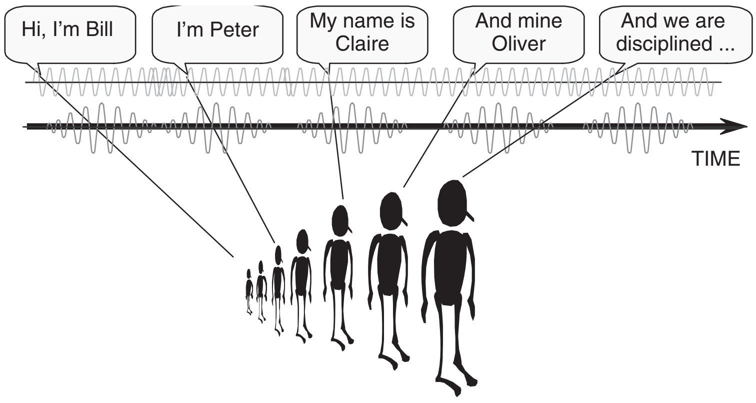

Figure 1.8 Bitrate is self-limiting in audio conferences (at least among polite participants).

which a participant is allowed to send RTCP reports is one every 5 s. The sending rate is randomized by a factor of 0.5 to 1.5 to avoid unwanted synchronization between reports.

Most H.323 and SIP implementations actually use a simplified version of these guidelines, which is not a problem because there is no scaling issue. The RFC recommendations remain applicable for larger conferences, however, such as the conferences using the **H.332** protocol to broadcast information to multiple receivers.

1.1.2.3.1.1 RTCP packet types

There are various types of RTCP messages defined for each type of information:

- **SR**: sender reports contain transmission and reception information for active senders.
- **RR**: receiver reports contain reception information for listeners who are not also active senders.
- **SDES**: source description describes various parameters relating to the source, including the name of the sender (CNAME).
- **BYE**: sent by a participant when he leaves the conference.
- **APP**: functions specific to an application.

Several RTCP messages can be packed in a single transport protocol packet. Each RTCP message contains enough length information to be properly decoded if several of those RTCP messages are packed in a single UDP packet. This packing can be useful to save overhead bandwidth used by the transport protocol header.

1.1.2.3.1.2 Sender reports

Each SR contains three mandatory sections, as shown in Figure 1.9.

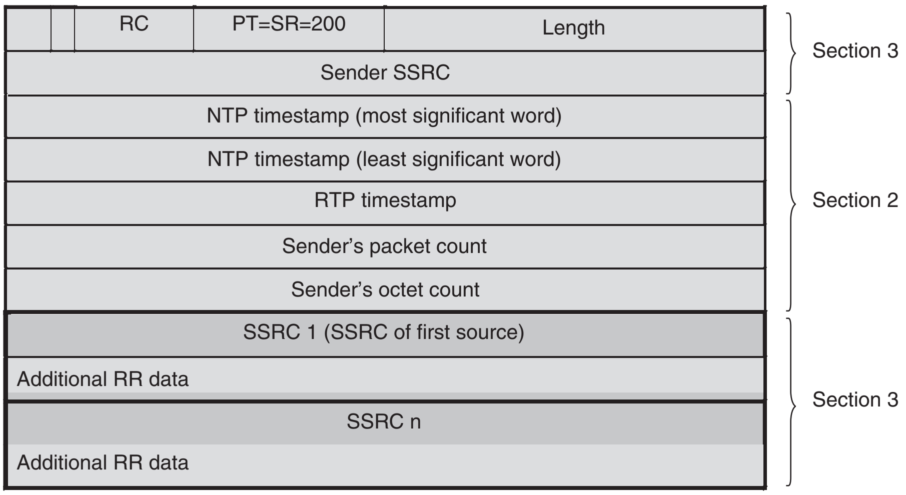

Figure 1.9 RTCP packet format.

The first section contains:

- The 5-bit reception report count (RC), which is the number of report blocks included in this SR.
- The packet type (PT) is 200 for an SR. In order to avoid mixing a regular RTP packet with an SR, RTP packets should avoid payload types 72 and 73 which can be mistaken for SRs and RRs when the marker bit is set. However, normally a UDP port is dedicated to RTCP to eliminate this potential confusion.
- The 16 bit length of this SR including header and padding (the number of 32-bit words minus 1).
- The SSRC of the originator of this SR. This SSRC can also be found in the RTP packets that originate from this host.

The second section contains information on the RTP stream originated by this sender (this SSRC):

- The NTP timestamp of the sending time of this report. A sender can set the high-order bit to 0 if it can't track the absolute NTP time; this NTP measurement only relates to the beginning of this session (which is assumed to last less than 68 years!). If a sender can't track elapsed time at all it may set the timestamp to 0.
- The RTP timestamp, which represents the same time as above, but with the same units and random offset as in the timestamps of RTP packets. Note that this association of an absolute NTP timestamp and the RTP timestamps enables the receiver to compute the absolute timestamp of each received RTP packet and, therefore, to synchronize related media streams (e.g., audio and video) for playback.
- Sender's packet count (32 bits) from the beginning of this session up to this SR. It is reset if the SSRC has to change (this can happen in an H.323 multiparty conference when the active MC assigns terminal numbers).
- Sender's payload octet count (32 bits) since the beginning of this session. This is also reset if the SSRC changes.

The third section contains a set of reception report blocks, one for each source the sender knows about since the last RR or SR. Each has the format shown in Figure 1.10:

- SSRC_n (source identifier)(32 bits): the SSRC of the source about which we are reporting.
- Fraction lost (8 bits): equal to Floor(received packets/expected packets $*$ 256).
- Cumulative number of packets lost (24 bits) since the beginning of reception. Late packets are not counted as lost and duplicate packets count as received packets.
- Extended highest sequence number received (32 bits): the most significant 16 bits contain the number of sequence number cycles, and the last 16 bits contain the highest sequence number received in an RTP data packet from this source (same SSRC).

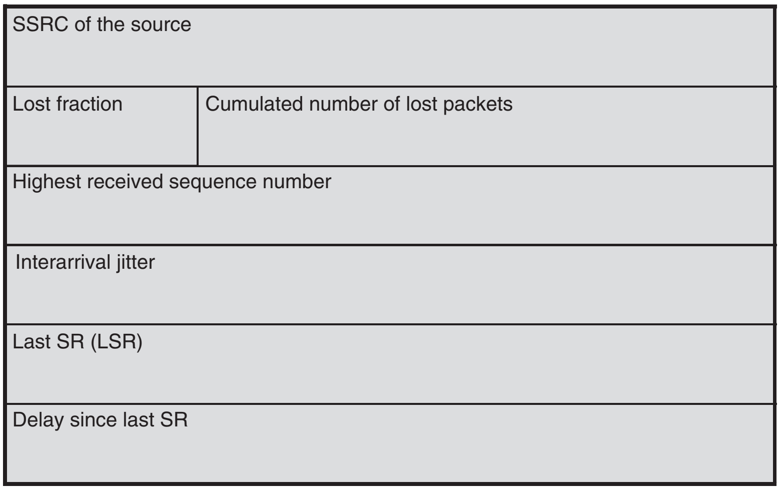

Figure 1.10 Format of a reception report block.

- Interarrival jitter (32 bits): an estimation of the variance in interarrival time between RTP packets, measured in the same units as the RTP timestamp. The calculation is made by comparing the RTP timestamp of arriving packets with the local clock, and averaging the results (as shown in Figure 1.11).
- The last SR timestamp (LSR) (32 bits): the middle 32 bits of the NTP timestamp of the last SR received (this is the compact NTP form).
- The delay since the last SR arrived (DLSR) (32 bits): expressed in compact NTP form (or, more simply, in multiples of 1/65536 s). Together with the last SR timestamp, the sender of this last SR can use it to compute the round trip time.

1.1.2.3.1.3 Receiver reports

A receiver report looks like an SR, except that the PT field is now 201, and the second section (concerning the sender) is absent.

1.1.2.3.1.4 SDES: source description RTCP packet

An SDES packet (Figure 1.12) has a PT of 202 and contains SC (source count) chunks. Each chunk contains an SSRC or a CSRC and a list of information. Each element of this list is coded using the type/length/value format. The following types exist but only CNAME has to be present:

- CNAME (type 1): unique among all participants of the session; is of the form user@host, where host is the IP address or domain name of the host.
- NAME (type 2): common name of the source.
- EMAIL (type 3).

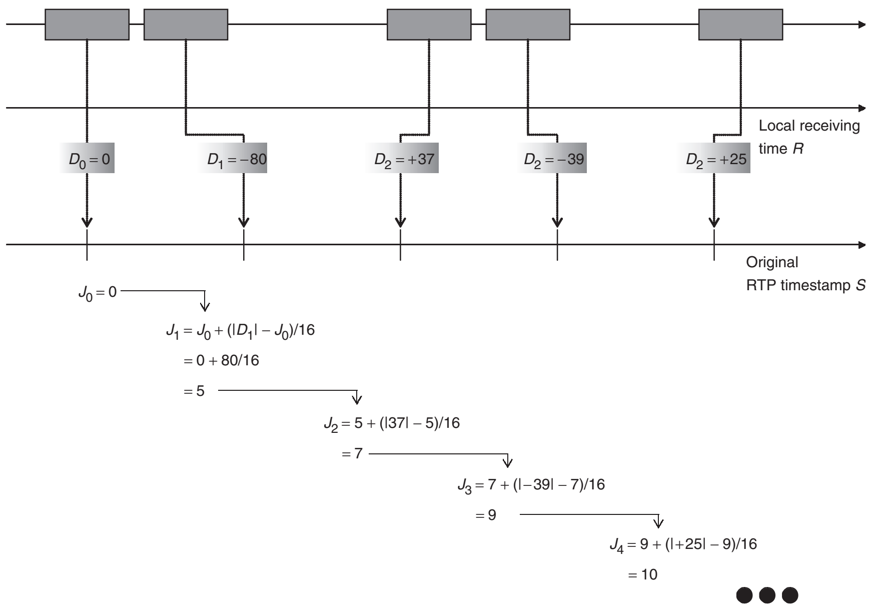
$D_0 = 0$
$D_1 = -80$
$D_2 = +37$
$D_2 = -39$
$D_2 = +25$
Local receiving time R
Original RTP timestamp S
$J_0 = 0$
$J_1 = J_0 + (|D_1| - J_0)/16$
$= 0 + 80/16$
$= 5$
$J_2 = 5 + (|37| - 5)/16$
$= 7$
$J_3 = 7 + (|-39| - 7)/16$
$= 9$
$J_4 = 9 + (|+25| - 9)/16$
$= 10$

Figure 1.11 Jitter evaluation.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|    SC   |  PT=SDES=202  |             length            | header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          SSRC/CSRC_1                          | chunk
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   1
|                           SDES items                          |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          SSRC/CSRC_2                          | chunk
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   2
|                           SDES items                          |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1.12 SDES message format.

- PHONE (type 4).
- LOC (type 5): location.

1.1.2.3.1.5 BYE RTCP packet

The BYE RTCP packet (Figure 1.13) indicates that one or more sources (as indicated by source count SC) are no longer active.

1.1.2.3.1.6 APP: application-defined RTCP packet

This can be used to convey additional proprietary information. The format is shown in Figure 1.14. The PT field is set to 204.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|    SC   |   PT=BYE=203  |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           SSRC/CSRC                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                              ...                              :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|     length    |               reason for leaving            ... (opt)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1.13 BYE message format.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P| subtype |   PT=APP=204  |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           SSRC/CSRC                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          name (ASCII)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   application-dependent data                ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1.14 APP message format.

1.1.2.4 Security

Security can be achieved at the transport level (e.g., using IPSec) or at the RTP level. The RTP RFC presents a way to ensure RTP-level privacy using DES/CBC (data encryption standard, cipher block chaining) encryption. Since **DES**, like many other encryption algorithms, is a block algorithm (for a more detailed description see Section 2.6.2 about H.235), there needs to be some adaptation when the unencrypted payload is not a multiple of 64 bits.

The most straightforward method, padding, is described in RTP (RFC 1889, Section 6.1). When this method is used the padding bit of the RTP header is set, and the last octet of the RTP payload contains the number of padding bits to remove (Figure 1.15). The last octet can be located because the underlying transport protocol must support framing. There are other encryption methods that do not require padding (e.g., ciphertext stealing); some of these alternative methods are described in Chapter 2 (on H.235).

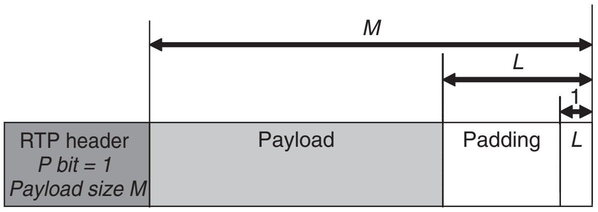

Figure 1.15 RTP payload padding for encryption using block algorithms.

Authentication and negotiation of a common secret is not within the scope RTP. For instance, the negotiation of a common secret can be performed out of band using a **Diffie–Helmann** scheme (see Section 2.6.2.1).

1.2 ENCODING MEDIA STREAMS

1.2.1 Codecs

We have seen already that isochronous (audio, video, etc.) data streams could be carried over RTP. But these analogue signals first need to be transformed into data. This is the purpose of codecs. This section provides a high-level overview of some of the most popular voice and video coding technologies, sufficient in most cases to understand H.323, SIP, or MGCP and to help in the recurring debates about the 'best' codec. The reader wanting more detailed knowledge should read the voice-coding background chapter in the companion book, *Beyond VoIP Protocols*.

1.2.1.1 What is a good codec?

When the International Multimedia Telecommunications Consortium (www.IMTC.org) tried to choose a default low-bitrate codec, sufficient to promote interoperability, they faced a difficult issue because there was not common agreement about what constituted a good codec. The difficulty was so great that other bodies who are also trying to profile VoIP applications are reticent to enter into the debate at all.

Let's look at the criteria that must now be considered when evaluating a voice codec.

1.2.1.1.1 Bandwidth usage

The bitrate of available narrow-band codecs (approximately 300–3400 Hz) today ranges from 1.2 kbit/s to 64 kbit/s. Of course there is a consequence on the quality of restituted voice. This is usually measured by MOS (mean opinion score) marks. MOSs for a particular codec are the average mark given by a panel of auditors listening to several recorded samples (voice samples, music samples, voice with background noise, etc.). These scores range from 1 to 5:

- From 4 to 5 the quality is 'high' (i.e., similar to or better than the experience we have when making an ISDN phone call).
- From 3.5 to 4 is the range of 'toll quality'. This is more or less similar to what is obtained with the G.726 codec (32 kbit/s ADPCM) which is commonly taken as the reference for 'toll quality'. This is what we experience on most phone calls. Mobile phone calls are usually just below the 'toll' quality.
- From 3.0 to 3.5, communication is still good, but voice degradation is easily audible.

- From 2.5 to 3, communication is still possible, but requires much more attention. This is the range of ‘military quality’ voice. In extreme cases the expression ‘synthetic’, or ‘robotic’, voice is used (i.e., when it becomes impossible to recognize the speaker).

There is a trade-off between voice quality and bandwidth used. With current technology toll quality cannot be obtained below 5 kbit/s.

1.2.1.1.2 Silence compression (VAD, DTX, CNG)

During a conversation, we only talk on average 35% of the time. Therefore, silence compression or suppression is an important feature. In a point-to-point call it saves about 50% of the bandwidth, but in decentralized multicast conferences the activity rate of each speaker drops and the savings are even greater. It wouldn’t make sense to undertake a multicast conference where there are more than half a dozen participants without silence suppression.

Silence compression includes three major components:

- **VAD (voice activity detector)**: this is responsible for determining when the user is talking and when he is silent. It should be very responsive (otherwise the first word may get lost and unwanted silence might occur at the end of sentences), without getting triggered by background noise. VAD evaluates the energy and spectrum of incoming samples and activates the media channel if this energy is above a minimum and the spectrum corresponds to voice. Similarly, when the energy falls below a threshold for some time, the media channel is muted. If the VAD module dropped all samples until the mean energy of the incoming samples reaches the threshold, the beginning of the active speech period would be clipped. Therefore, VAD implementations require some lookahead (i.e., they retain in memory a few milliseconds worth of samples to start media channel activation before the active speech period). This usually adds some delay to the overall coding latency, except on some coders where this evaluation is coupled with the coding algorithm itself and does not add to the algorithmic delay. The quality of the implementation is important: good VAD should require minimal lookahead, avoid voice clipping, and have a configurable hangover period (150 ms is usually fine, but some languages, such as Chinese, require different settings).
- **DTX (discontinuous transmission)**: this is the ability of a codec to stop transmitting frames when the VAD has detected a silence period. If the transmission is stopped completely, then it should set the marker bit of the first RTP packet after the silence period. Some advanced codecs will not stop transmission completely, but instead switch to a silence mode in which they use much less bandwidth and send just the bare minimum parameters (intensity, etc.) in order to allow the receiver to regenerate the background noise.
- **CNG (comfort noise generator)**: it seems logical to believe that when the caller isn’t talking, there is just silence on the line and, when the VAD detects a silence period, it should be enough to switch off the loudspeaker completely. In fact, this

approach is completely wrong. Movie producers go to great lengths to recreate the proper background noise for 'silent' sequences. The same applies to phone calls. If the loudspeaker is turned off completely, street traffic and other background noise that could be overheard while the caller was talking would stop abruptly. The called party would get the impression that the line had been dropped and would ask the caller whether he is still on the line. The CNG is here to avoid this and recreate some sort of background noise. With the most primitive codecs that simply stop transmission it will use some random noise with a level deduced from the minimal levels recorded during active speech periods. More advanced codecs such as G.723.1 (annex A) or G.729 (annex B) have options to send enough information to allow the remote decoder to regenerate ambient noise close to the original background noise.

1.2.1.1.3 Intellectual property

End-users don't care about this, but manufacturers have to pay royalties to be allowed to use some codecs in their products. For some hardware products where margins are very low, this can be a major issue. Another common situation is that some manufacturers want to sell some back-end server, while distributing software to clients for free. If the client includes a codec, then again intellectual property becomes a major choice factor.

1.2.1.1.4 Lookahead and frame size

Most low-bitrate codecs compress voice in chunks called frames and need to know a little about the samples immediately following the samples they are currently encoding (this is called lookahead).

There has been a lot of discussion (especially at the IMTC when they tried to choose a low-bitrate codec) over the influence of frame size on the quality of the codec. This is because the minimal delay introduced by a coding/decoding sequence is the frame length plus the lookahead size. This is also called the algorithmic delay. Of course, in reality DSPs do not have infinite power and most of the time a fair estimate is to consider the real delay introduced as twice or three times the frame length plus the overhead (some authors improperly call this the algorithmic delay, although this is just an estimate of DSP power).

So, codecs with a small frame length are indeed better than codecs with a longer frame length regarding delay, when each frame is sent immediately on the network. This is where it becomes tricky, because each RTP packet has an IP header of 20 octets, a UDP header of 8 octets, and an RTP header of 12 octets! For instance, for a codec with a frame length of 30 ms, sending each frame separately on the network would introduce a 10.6-kbit/s overhead—much more than the actual bitrate of most narrow-band codecs!

Therefore, most implementations choose to send multiple frames per packet, and the real frame length is in fact the sum of all frames stacked in a single IP packet. This is limited by echo and interactivity issues (see companion book, Chapter 3 *Beyond VoIP Protocols*). A maximum of 120 ms of encoded voice should be sent in each IP packet.

So, for most implementations, the smaller the frame size, the more frames in an IP packet. This is all there is to it and there is no influence on delay. Overall, it is better to use codecs that have been designed for the longest frame length (limited by the acceptable delay), since this allows even more efficient coding techniques: the longer you observe a phenomenon, the better you can model it!

We can conclude that in most cases frame size is not so important for IP videoconferences when bandwidth is a concern. The exception is high-quality conferences where interactivity is maximized at the expense of the required bitrate.

1.2.1.1.5 Resilience to loss

Packet loss is a fact of life in IP networks and the short latency required by interactive voice and video applications does not allow us to request retransmissions. Since packets carry codec frames, this in turn causes codec frame loss. However, packet loss and frame loss are not directly correlated; many techniques such as FEC (forward error correction) can be used to lower the frame loss rate associated with a given packet loss rate. These techniques spread redundant information over several packets so that frame information can be recovered even if some packets are lost.

However, the use of redundancy to recover packet loss is a very tricky thing. It can lead to unexpected issues and, can even make the problem worse. To understand this, let's look at what some manufacturers could do (and have done!):

- You prepare a demonstration to compare your product and that of a competitor. You let it be known that you can resist a 50% packet loss without any consequence on voice quality.
- You simulate packet loss by losing one packet out of two.
- You put frames N and $N - 1$ into your RTP packet.

Your product can recover all the frames despite one packet out of two being lost. Your competitor is restricted to emitting a few cracks. Bingo! The customer is convinced.

Well, the only problem is that packet loss on the Internet in not so neat. Packet loss occurs in a correlated way and you are much more likely to lose several packets in a row, than exactly one packet out of two. So, this simple RTP redundancy scheme will be close to useless under real conditions and still add a 50% overhead!

The effect of frame erasure on codecs should be considered on a case-by-case basis. If you lose N samples from a G.711 codec (stateless coder) this will just result in a gap of $N * 125\,\mu s$ at the receiving end. If you lose just one frame from a very advanced codec it may be audible for much more than the duration of this frame, because the decoder will need some time to resynchronize with the coder. For a frame of 20 ms or so, this may result in a very audible crack of 150 ms. Codecs such as G.723.1 are designed to cope relatively well with an uncorrelated frame erasure of up to 3%, but beyond this quality drops off very rapidly. The effect of correlated loss is not yet fully evaluated. It is possible to reduce the occurrence of consecutive frame loss by interleaving codec frames across multiple RTP packets: unfortunately, this adds a lot of delay to the transmission and therefore can only be used in streaming media transmissions, not in the context of real-time communications.

Apart from the built-in features of the codec itself, it is possible to reduce the frame loss associated with packet loss by using a number of techniques.

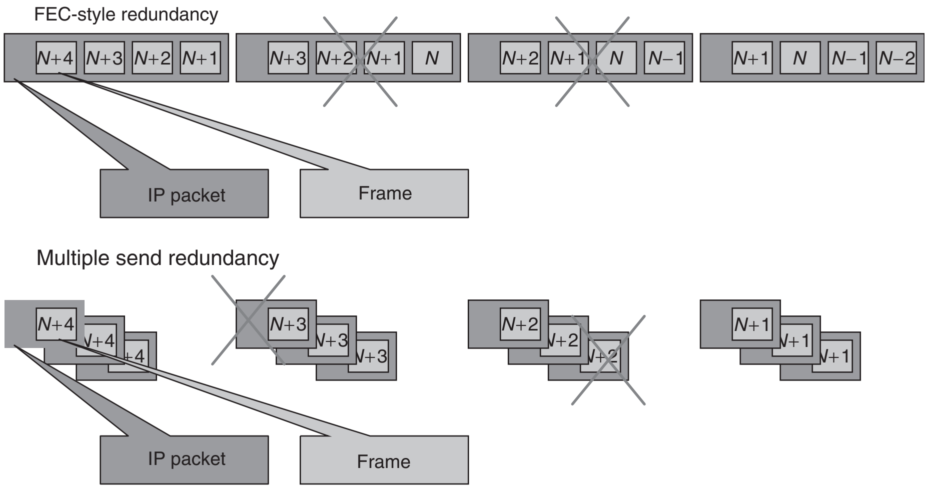

Figure 1.16 Using redundancy to reduce codec frame loss.

FEC-style redundancy (Figure 1.16) can be used to recover from serious packet loss conditions, but it has a significant impact on delay. For instance, if you choose to repeat the same G.723.1 frame in four consecutive IP packets in order to recover from the loss of three consecutive packets, then the decoder needs to maintain a buffer of four IP packets, but this ruins the delay factor. More sophisticated FEC methods use XOR sums instead of simple repetitions, but have the same impact on delay.

It is also possible to send several copies of each frame immediately. But if one packet gets lost, probably all the copies will reach the same congested router at nearly the same time and might get lost as well.

An understanding of the different types of congestion is also important in deciding whether redundancy is useful and which type to use. The network can lose packets because a link is congested or because a router has to route too many small packets per second.

If a link is congested, then any type of redundancy will add to the congestion and increase the overall loss percentage of IP packets. But the frame loss rate of communicating devices that use FEC redundancy will still be reduced.

Some arithmetic proves this. Say we have congestion on a 2-Mbit/s line. It receives 2.2 Mbit/s and the average loss rate is $0.2/2.2 = 9\%$. Part of this is caused by someone using a codec producing a 100 kbit/s stream. The software detects a high loss and decides to use the FEC scheme described above. Now that same application produces a 400 kbit/s stream (the influence of headers is not taken into account for simplicity). The 2-Mbit/s line receives 2.5 Mbit/s and the packet loss rate is increased to 20% for all the users of the link. However, if we assume the congested link never causes the loss of four packets in a row (on average one packet in five is dropped), then the software will recover from *all* loss. However, this would be unacceptable behaviour, because it would be unfair to other users and could destabilize the network. Next-generation IP networks will probably include advanced techniques, such as RED, that will detect the greedy user and drop most of his packets.

If congestion is due to an overrun router (exceeding its packet/s limit), then FEC-style redundancy is not such a bad thing. It increases the size of packets but does not increase the average number of packets that the router has to forward per second. In this case increasing the size of the packets will not add to the congestion. The other type of redundancy (multiple simultaneous sending) will increase the number of packets through the router and would not work.

1.2.1.1.6 Layered coding

There are several situations in which current codecs are not well suited. For instance, if you want to broadcast the same event to several listeners (H.332 type of conference), some will want high-quality reception (either because they have paid for it or because they have large IP pipes) and others will only be able to receive lower quality. You could send a customized data stream to each listener, but this is not practical for a large audience. The answer is to multicast the data stream to all listeners (for more information on multicast, refer to the multicast chapter of the companion book, *Beyond VoIP Protocols*). Current codecs include complete information in one data stream. If it is multicast, all participants will receive the same amount of data, so you usually have to limit the data rate to the reception capability of the least capable receiver.

Some codecs (most are still at experimental stage) can produce several data streams simultaneously, one with the core information needed for ‘military quality’ reception and the other data streams with more information as needed to rebuild higher fidelity sound or an image. A crude example for video would be to send black and white information on one channel and colour (chrominance) information on another.

Each part of the data stream can be multicast using different group addresses, so that listeners can choose to receive just the core level or the other layers as well. In a pay-for-quality scheme, you would encrypt the higher layers (this way you have the option of receiving a free low-quality preview and later of paying for the broadcast quality image).

Layered codecs are also very useful when it comes to redundancy: the sender can choose to use a redundancy scheme or a quality of service level for the core layer, so that the transmission remains understandable at all times for everyone, but leaves other layers without protection.

H.323v2 was approved with a specific annex on layered video coding (annex B: procedures for layered video codecs).

1.2.1.1.7 Fixed point or floating point

We first need to say a few words about digital signal processors (DSPs). These are processors that have been optimized for operations frequently encountered in signal-processing algorithms. One such operation is $(a * b) + \mathit{previous\ result}$: one multiplication and addition. In a conventional processor, this operation would require multiple processor instructions and would be executed in several clock cycles. A DSP will do it in one instruction and a single clock cycle. Another example is the code book searches frequently used by vocoders. Some conventional processors also have extensions to accelerate signal-processing algorithms (e.g., MMX processors can execute a single instruction simultaneously on several operands as long as they can be contained in a 32-bit register and video algorithms can be accelerated by processing 4 pixels (8 bits each) simultaneously).

There are two types of DSPs: floating point DSPs, which are capable of operating on floating point numbers, and fixed point DSPs. Fixed point DSP operands are represented as a mantissa n and a power p of 2 (e.g., $12345678 * 2^5$), but the DSP can operate on two operands only if the power of 2 is the same for both operands. They are less powerful, but also less expensive, and chosen by many designers for products sold in large quantities. Some codecs have only been specified with fixed point C code. However, many implementations will run on processors or DSPs that are capable of floating point operation, and developers must develop their own version of floating point C code for the algorithm. This often results in interoperability problems between floating point versions.

Therefore, it makes sense for the codec to be specified in floating point C code as well, especially if the code has to run on PCs.

1.2.1.2 Audio codecs

1.2.1.2.1 ITU audio codecs

1.2.1.2.1.1 Choosing a codec at ITU

The choice of a codec at ITU WP3 is typically a very long process. This is not a bureaucracy problem, but rather a problem due to the stringent requirements of ITU experts.

Before a codec is chosen, the ITU evaluates MOS scores and usually requires quality that is equivalent to or better than G.726 (‘toll quality’). The ITU also checks that this quality is constant for men and women, and in several languages. The ability to take into account background noise and recreate it correctly is also evaluated. The ITU pays special attention to the degradation of voice quality in tandem operation (several successive coding/decoding processes), since this a situation that is very likely to happen in international phone calls. Last but not least, if the codec has to be used over a non-reliable medium (a radio link, a frame relay virtual circuit, etc.), the ITU checks that the quality remains acceptable if there is some frame loss.

After checking all these parameters, it frequently occurs that no single proposal passes the test! Therefore, many ITU codecs are combinations of the most advanced technologies found in several different proposals. This leads to state-of-the-art choices, but, as we will see, this is a nightmare for anyone who needs to keep track of intellectual property.

1.2.1.2.1.2 Audio codecs commonly used in VoIP

The companion book, *Beyond VoIP Protocols* provides a detailed view on voice coder technology and discrete time signal processing in general. This section’s purpose is to provide a quick reference to common VoIP coders found in VoIP systems for engineers uninterested in the details and theory of each coder.

(a) G.711 (approved in 1965)

G.711 is the grandfather of digital audio codecs. It is a very simple way of digitizing analogue data by using a semi-logarithmic scale (this is called ‘companded PCM’, and serves to increase the resolution of small signals, while treating large signals in the same way as the human ear does). Two different types of scales are in use, the A-law scale (Europe, international links) and the μ-law scale (USA, Japan). They differ only in the choice of some constants of the logarithmic curve. G.711 is used in ISDN and on most digital telephone backbones in operation today.

A G.711-encoded audio stream is a 64-kbit/s bitstream in which each sample is encoded as an octet; therefore, the frame length is only 125 µs. Of course, all VoIP applications will put more than one sample in every IP packet (about 10 ms typically, or 80 samples).

Most sound cards are able to record directly in G.711 format. However, in some cases it is better to record using CD quality, which samples at 44.1 kHz (one 16-bit sample every 23 µs), especially if echo cancelation algorithms are used, since the full performance of some echo cancelation algorithms cannot be achieved with the quantification noise introduced by G.711.

The typical MOS score of G.711 is usually taken as 4.2; it is used as an anchor for other coder tests.

(b) G.722 (approved in 1988)

Although G.711 achieves very good quality, some of the voice spectrum (above 4 kHz) is still cut. G.722 provides a higher quality digital coding of 7 kHz of audio spectrum at only 48, 56, or 64 kbit/s, using about 10 DSP MIPS. This is an 'embedded' coder, which means that the rate can freely switch between 48, 56, or 64 kbit/s without notifying the decoder.

This coder is very good for all professional conversational voice applications (the algorithmic delay is only 1.5 ms). G.722 is supported by some videoconferencing equipment and some IP phones.

(c) G.722.1

This more recent wideband coder operates at 24 kbit/s or 32 kbit/s. It has been designed by Picturetel, which also sells a 16-kbit/s version (Siren™). The coder encodes frames of 20 ms, with a lookahead of 20 ms. The 16-kbit/s version is supported by Windows® Messenger.

(d) G.723.1 (approved in 1995)

In the early days of VoIP, the VoIP Forum chose the G.723.1 codec as the baseline codec for narrow-band H.323 communications. It is also used by the video cellphones of UMTS 99 (H.324M standard).

(i) Technology

G.723.1 uses a frame length of 30 ms and needs a lookahead of 7.5 ms. It has two modes of operation, one at 6.4 kbit/s and the other at 5.3 kbit/s. The mode of operation can change dynamically at each frame. Both modes of operation are mandatory in any implementation, although many VoIP systems have an incorrect implementation that works on only one of the two modes.

G.723 is not designed for music and does not transmit DTMF tones reliably (they must be transmitted out-of-band). Modem and fax signals cannot be carried by G.723.1.

Table 1.2 Impact of frame-erasure and tandeming quality

	G.723.1, 6.4 kbit/s	32 kbit/s ADPCM
Clear channel, no errors or frame erasure	3.901	3.781
3% frame erasure	3.432	—
Tandeming of two codecs	3.409	3.491

G.723.1 achieves an MOS score of 3.7 in 5.3-kbit/s mode and 3.9 in 6.4-kbit/s mode. Table 1.2 compares the performance of G.723.1 (6.4 kbps) and the ADPCM released by Bell Labs in March 94.

The main effect of frame erasures is to desynchronize the coder and the decoder (they may need many more frames to resynchronize). In practice, networks should always have a frame error rate below 3% (and below 1% ideally).

G.723.1 is specified in both fixed point (where it runs at 16 MIPS on a fixed point DSP) and floating point C code (running on a Pentium 100, it takes 35% to 40% of the power of the processor). The fixed point implementation runs on VoIP gateways, while the floating point version runs on all Windows® PCs.

Beyond VoIP systems, G.723.1 is used in the H.324 recommendation (ITU recommendation for narrow-band videoconferencing on PSTN lines) and will also be used in the new 3G-324M (3GPP, 3GPP2 organizations) standard for 3G wireless multimedia devices.

(ii) Silence compression

G.723.1 supports voice activity detection (VAD), discontinuous transmission (DTX), and comfort noise generation (CNG) (defined in annex A of the recommendation).

Silence is coded in very small, 4-octet frames at a rate of 1.1 kbit/s. If silence information doesn't need to be updated, transmission stops completely.

(iii) Intellectual property

G.723.1 is one of the codecs that resulted from many contributions and, therefore, uses technology patented from several sources. About 18 patents currently apply to G.732.1 (the precise number is hard to keep track of), from eight different companies.

The main licensing consortium, which is made up of AudioCodes, DSP Group, FT/CNET, Université de Sherbrooke, and NTT, oversees the patents. The rights are managed by the DSP Group and SiproLabs (www.sipro.com) for all members of this consortium. Other patents are held by AT&T (1), Lucent (3), British Technology Group (1, formerly held by VoiceCraft), Nokia Mobile Phone (1, formerly held by VoiceCraft). Patent applications have also been made by Siemens, Robert Bosch, and CSELT. The source code is copyrighted by four companies.

There are typically several licensing agreements for this codec (the details hinge, of course, on the company involved), depending on whether the application is for a single user or multiple users, whether it is going to be a paying or free application, and on the volume licensed.

Of course, exact prices have to be negotiated with both patent owners and implementers, but some data can be gathered from conferences and newsgroups, *although they must be* *taken cautiously*. For instance, here are some price indications for acquisition of the intellectual rights of G.723.1:

- A license for a single-user client is said to be worth an initial payment of around \$50,000 plus \$0.8 per unit.
- A license for a server is said to be worth an initial payment of about \$20,000 plus \$5 per port.
- A license for unlimited distribution of a single-user application is said to be worth about \$120,000.

Then, unless you do your own implementation (which is not recommended if you are not an expert!), be prepared to approximately double the previous fees to license a well-optimized implementation.

Here is a quote from a company trying to license these codecs, picked from a mailing list:

> *We have been trying to negotiate licensing arrangements with the patent holders for more than six months. As of today, we have received terms and conditions from six of the holders, and little to no response from the rest. The costs proposed by the first six strongly imply a substantial initial investment, and a per port cost in excess of \$20.00.*
>
> *Our concern, however, extends far beyond the cost. The Internet's success is due to its readily available standards and lack of non-essential rules and constraints. The time requirements and logistics of establishing contact with 12 parties and negotiating licensing are significant barriers to growth in the industry. The legal risks associated with not doing so are an impediment to the rapid evolution of the industry.*

The reality is not quite so bad, as many IPR rights are now managed by Sipro Labs (www.sipro.com). The investments needed to produce the technology of standardized coders such as G.723.1 indeed justify a fee. But the question is how much is reasonable? When patented technology becomes a standard, the temptation is high to use this monopoly situation to maintain high licence prices. This underlies the so-called 'codec wars' that periodically break out in VoIP standard bodies.

(e) G.726 (approved in 1990)

G.726 uses an ADPCM technique to encode a G.711 bitstream in words of 2, 3, or 4 bits, resulting in available bitrates of 16, 24, 32, or 40 kbit/s.

G.726 at 32 kbit/s achieves a MOS score of 4.3 and is often taken as the benchmark for 'toll quality'. It requires about 10 DSP MIPs of processing power (full duplex) or 30% of the processing power of a Pentium 100. This is a low-delay coder: 'frames' are 125 µs long and there is no lookahead. There is also an embedded version known as G.727.

(f) G.728 (approved in 1992–1994)

G.728 uses an LD-CELP (low-delay, code-excited linear prediction) coding technique and achieves MOS scores similar to that obtained by G.726 at 32 kbit/s, but with a bitrate of only 16 kbit/s. Compared with PCM or ADPCM techniques, which are waveform coders (i.e., they ignore the nature of the signal), CELP is a coder optimized for voice (vocoder). These coders specifically model voice sounds and work by comparing the waveform to encode with a set of waveform models (linear predictive code book) and find the best match. Then, only the index of this best match and parameters like voice pitch are transmitted. As a result music does not transmit well on CELP coders, and it is only at 2.4 kbit/s that fax or modem transmission can succeed with G.728 compression. G.728 is used for H.320 videoconferencing and some H.323 videoconferencing systems.

G.728 needs almost all the power of a Pentium 100 and 2 Kb of RAM to implement. It is a low-delay coder (between 625 µs and 2.5 ms).

(g) G.729 (approved in 1995–1996)

(i) Technology

G.729 is very popular for voice over frame relay applications and V.70 voice and data modems. Together with G.723, it has become the most popular voice coder for VoIP, but is still not supported natively on the Windows® platform. It uses a CS-ACELP (conjugate structure, algebraic code-excited linear prediction) coding technique. G.729 is not designed for music and does not transmit DTMF tones reliably (they must be transmitted out-of-band). Modem and fax signals cannot be carried by G.729.

G.729 produces 80-bit frames encoding 10 ms of speech at a rate of 8 kbit/s. It needs a lookahead of 5 ms. It achieves MOS scores around 4.0. There are two versions:

- G.729 (approved in December 1996) requires about 20 MIPS for coding and 3 MIPS for decoding.
- G.729A (approved in November 1995): annex A is a reduced complexity version of the original G.729. It requires about 10.5 MIPS for coding and 2 MIPS for decoding (about 30% less than G.723.1).

(ii) Silence compression

Annexes A and B of G.729 define VAD, CNG, and DTX schemes for G.729. The frames sent to update background noise description are 15 bits long and are only sent if the description of the background noise changes.

(iii) Licences

Both G.729 and G.729A are the result of about 20 patents belonging to six companies: AT&T, France Telecom, Lucent, Université de Sherbrooke (USH, Canada), NTT, and VoiceCraft. NTT, France Telecom, and USH have formed a licensing consortium managed by SiproLabs, but not all patents (notably AT&T) are covered by this consortium. The source code is copyrighted by five companies.

As with G.723, there are several ways of getting a several licence for this codec, but the prices of the G.729 IPR pool managed by SiproLabs (www.sipro.com) are in the public domain.

1.2.1.2.2 ETSI SMG audio codecs

The ETSI SMG11 (European Telecommunications Standardization Institute Special Mobile Group) standardized the speech codecs given in Table 2.3. In addition the new AMR coder has been standardized for use in UMTS, but is not used yet in VoIP systems.

Table 1.3 Performance of GSM coders in various environmental conditions

Codec	MOS in clean conditions	Vehicle noise	Street noise
GSM FR	3.71	3.83	3.92
GSM HR	3.85	3.45	3.56
GSM EFR	4.43	4.25	4.18
Reference with no coding	4.61	4.42	4.35

Source: TR 06.85 v2.0.0 (1998). Reproduced by Permission of the European Telecommunications Standards Institute – ETSI.

1.2.1.2.2.1 GSM full rate (1987)

GSM full rate, also called GSM 06.10, is perhaps the most famous codec in use today and runs daily in millions of GSM cellular phones. It provides good quality and operates well in the presence of background noise (see Table 1.3). It uses an RPE-LTP technique to encode voice in frames of 20 ms at a rate of 13 kbit/s. It needs no lookahead. GSM-FR achieves MOS scores slightly below toll quality.

GSM-FR is not extremely complex and requires only about 4.5 MIPS and less than 1 Kb of RAM.

The GSM full-rate patent is held by Philips and the licence is free for mobile phone applications.

1.2.1.2.2.2 GSM half-rate (1994)

Also called GSM 06.20, this coder aims at using less bandwidth while preserving the same or slightly lower speech quality as GSM-FR. This codec uses VSELP and encodes speech at a rate of 5.6 kbit/s. The frames are 20 ms long and there is a lookahead of 4.4 ms. The GSM-HR algorithm requires approximately 30 MIPS and 4 Kb of RAM. This coder has not been very successful, due to its high sensitivity to background noise. The patent is also held by Philips; ATT patents on CELP and NTT patents on LSP may also apply.

1.2.1.2.2.3 GSM enhanced full rate (1995)

This high-quality coder exceeds the G.726 ‘wireline reference’ in clear channel conditions and in background noise. It is also called GSM 06.60. It was selected as the base coder for the PCS 1900 cellular phone service in the US and was standardized by TIA in 1996. This codec uses a CD-ACELP technique and encodes 20-ms frames at a rate of 12.2 kbit/s. Optional VAD/DTX functions with comfort noise generation have been defined and there is also an example implementation for error concealment.

AT&T patents for CELP and NTT patents for LSP may apply.

1.2.1.2.3 Other proprietary codecs

1.2.1.2.3.1 Lucent/Elemedia SX7003P

The SX7003P is another popular codec. Although used in Lucent hardware it is licensed to other manufacturers as well. This codec has a frame size of 15 ms, which contains 4 control octets and 14 data octets. Silence frames have 2 octets of data.

In many VoIP implementations, two frames are packed in each IP packet (overhead of 40 bytes), leading to an IP bitrate of 20.3 kbit/s during voice activity periods and only 13.6 kbit/s during silence periods.

1.2.1.2.3.2 RT24 (Voxware)

The RT24 is one of the ultra-low-bitrate coders. Unfortunately, it is spoiled by the IP overhead. It has a bitrate of 2,400 bit/s and achieves an MOS of 3.2. It has a frame size of 22.5 ms (54 bits) which results in a measured IP-level bitrate of 16.6/9.5/7.1/6 kbit/s with 1/2/3/4 frames per IP packet.

1.2.1.2.4 Future coders

Both the AMR and AMR-WB (G.722.2) coders (described in detail in the companion book, *Beyond VoIP Protocols*) will probably be implemented in VoIP systems. Their ability to dynamically reduce the bitrate to adapt to the conditions of the transmission channel is not as useful in VoIP as it is over radio links. Over radio links, there are bit errors, but AMR makes it possible to add redundancy information dynamically to the media stream without requiring more bandwidth when network conditions degrade. Because of this, the AMR coder can offer better voice quality than any of the current narrow-band coders, over a much wider range of transmission network quality.

Over IP transmission links, there are only frame erasure errors, because each IP packet (containing one or more coder frame) is protected by a CRC code.[3] Redundancy can only be added by repeating each frame in multiple packets (forward error correction and interleaving). This has a significant, often unacceptable impact on end-to-end delay. Therefore, AMR will mainly be used to avoid any transcoding (Tandeming) when communicating with UMTS and CDMA2000 systems, thereby improving end-to-end voice quality.

Usage of the AMR and AMR-WB coders will be closely associated with the deployment of UMTS and CDMA2000 3G systems, and in the future LTE (Long Term Evolution) systems. Both coders will not only require more DSP processing power, they will require more powerful DSPs to achieve the densities of current G.723.1/G.729 systems.

1.2.1.3 ITU video codecs

1.2.1.3.1 Representation of colours

The representation of colours is derived from the fact that any colour can be generated from three primaries. From an artist's point of view, the three primaries are red, yellow, and blue. These colours are called subtractive primaries, because any colour can be generated from a white beam passed through a sequence of red, yellow, and blue filters. When an artist puts a layer of yellow paint of a sheet of paper, this layer acts as a filter

[3] Some VoIP networks have disabled the UDP checksum mechanisms in order to be more tolerant to bit ends. This could open the way to a more efficient use of the capabilities of AMR.

that allows most of the yellow component of the white light to be reflected, but filters out most other colours.

Video monitors use instead additive primaries: red, green, and blue. By mixing three beams of red, blue, and green light with various intensities, it is possible to generate any colour. Therefore, any colour can be represented by its barycentric coordinates (representing the intensity of each primary colour, not necessarily positive as illustrated in Figure 1.17!) in a triangle with a primary colour at each edge: this is the RGB (red–green–blue) format. The weight of each colour usually ranges from 0 to 255 in the RGB format: each pixel is described using 8 bits for each colour weight, which leads to 24 bits per pixel.

Another common representation is to use luminance (brightness represented by Y) and chrominance (hue represented by U and V, or Cr and Cb). Several conventions exist for this conversion (JFIF for JPEG, CCIR 601 for H.261, and MPEG). Figure 1.18 shows how JFIF converts from an RGB format to a YUV format.

Y, U, and V cover the range from 0 to 255 (U and V are often shifted to take values between -128 and $+127$). For CCIR this range is from 16 to 235.

Experiments have shown that the human eye is more sensitive to the luminance information. Because of this U and V values can be sampled at reduced frequency without inducing significant loss in the quality of the image. Typically, U and V are only sampled for a group of 4 pixels. Coding an image in this way leads to a 2:1 compression (i.e., instead of 24 bits per pixel, we now have 8 bits for Y and $(8 + 8)/4$ pixels for U and V).

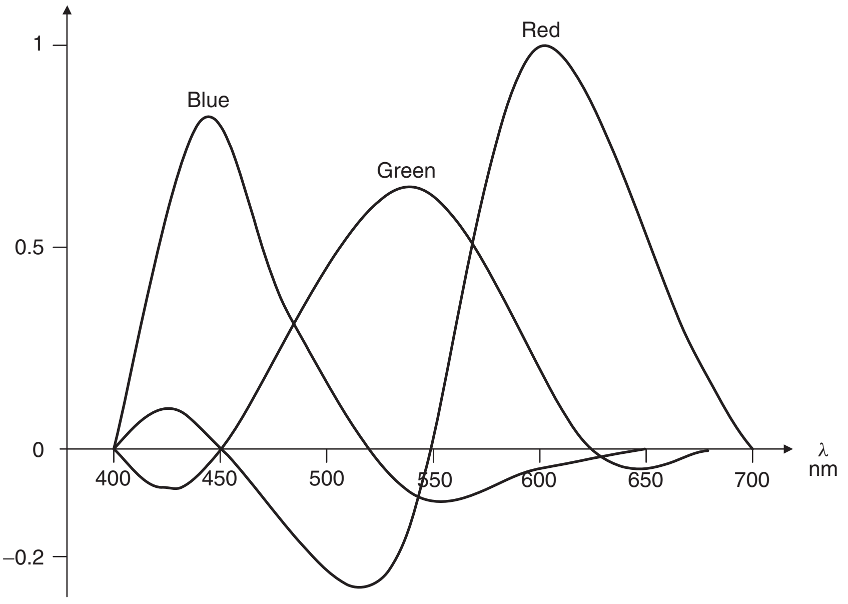

Figure 1.17 Red–green–blue components of visible colours in the 400–700-nm wavelength range.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{pmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Figure 1.18 RGB to YUV conversion (JFIF).

1.2.1.3.2 Image formats

Several image formats are commonly used by video codecs. CIF (common intermediary format) defines a 352∗288 image. This size has been chosen because it can be sampled relatively easily from both the 525- and 625-line video formats and approaches the popular 4/3 length/width ratio.

In addition to the resolution of CIF being below that for TV quality, it is still relatively difficult to transmit over low-bandwidth lines, even with efficient coding schemes such as H.261 and H.263. For this reason two other formats with lower resolutions have been defined. At half the resolution in both dimensions, quarter CIF (QCIF) is for 176∗144 images, and SQCIF is only 128∗96.

For professional video application, CIF is clearly insufficient, and images need to be coded using 4CIF (704∗576) or 16CIF (1,408∗1,152) resolution (see Table 1.4).

1.2.1.3.3 H.261

H.261 is a video codec used in H.320 videoconferencing to encode the image over several 64-kbit/s ISDN connections, but in video over IP applications the bitstream is encoded in a single RTP logical channel. The H.261 codec is intended for compressed bitrates between 40 kbit/s and 2 Mbit/s. The source image is normally 30 (29.97) frames per second, but the bitrate can be reduced by transmitting only 1 frame out of 2, 3, or 4. The image formats shown in Table 1.5 can be encoded by H.261. The 4CIF and 16CIF formats are not supported by H.261.

The H.261 coding process involves several steps. After initial *YUV* coding of the original image using CCIR parameters, as described above, the image is divided in 8∗8 luminance pixels blocks for the luminance plane. The same surface is coded with only

Table 1.4 Uncompressed bitrate for various video formats

	Luminance		Uncompressed bitrate (Mbit/s)			
			10 frames/s		30 frames/s	
Picture format	Pixels	Lines	Grey	Colour	Grey	Colour
SQCIF	128	96	1.0	1.5	3.0	4.4
QCIF	176	144	2.0	3.0	6.1	9.1
CIF	352	288	8.1	12.2	24.3	36.5
4CIF	704	576	32.4	48.7	97.3	146.0
16CIF	1,408	1,152	129.8	194.6	389.3	583.9

Grey images are obtained by transmitting only the Y luminance component. Colour images are obtained by also transmitting the U, V chrominance components sampled at half the resolution.

Table 1.5 Video image sizes supported by H.261

SQCIF	128∗96	Optional
QCIF	176∗144	Required
CIF	352∗288	Optional

4∗4 chrominance pixels for each chrominance plane. Four luminance blocks are grouped with two chrominance blocks (one for U, one for V) in a structure called a **macroblock**.

Each macroblock can be coded using the ‘intra’ method or the ‘inter’ method (Figure 1.19). The intra method codes by means of a local compression method (just using information relative to macroblocks that have already been encoded in the same image), while the inter method codes adjacent frames relatively in time. The coding method can be defined for a macroblock, for a ‘group of blocks’ (GOB’s), or for a full frame. In general, video coders use the same method within a frame; hence the name intraframe (I-frame) or interframe (P-frame) frequently used when discussing video applications. The inter method is much more efficient, but leads to error accumulation; therefore, it is necessary to send intraframes intermittently.

Intraframes (I-frames) use a coding similar to the one used by JPEG, which involves DCT (discrete cosine transform), quantization, run length encoding, and entropy encoding (Figure 1.20).

For interframes (P-frames), the algorithm follows these steps:

- Motion detection: comparison of the image to be coded with the last coded image trying to find those parts of the image that have moved. This results in a representation of the difference between the motion-compensated image and the real one.

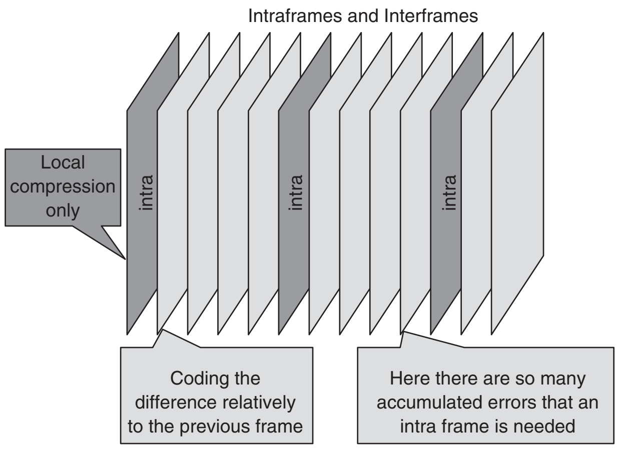
Intraframes and Interframes
Local compression only
intra
intra
intra
Coding the difference relatively to the previous frame
Here there are so many accumulated errors that an intra frame is needed

Figure 1.19 Intra and inter coding methods.

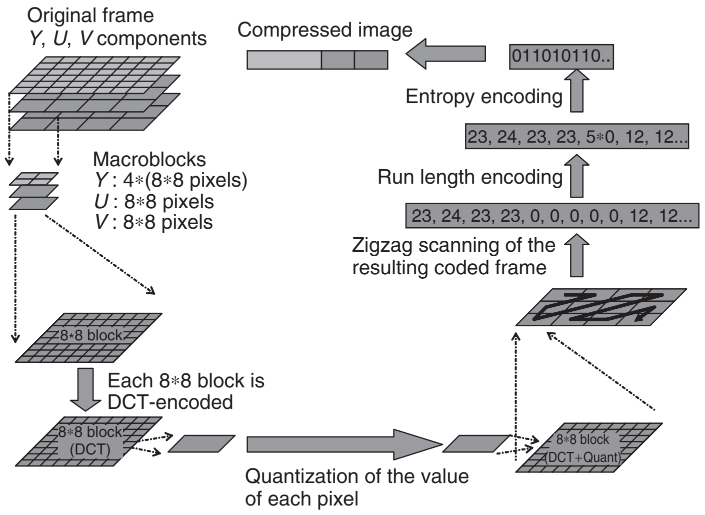

Figure 1.20 JPEG-style encoding used for intra frames.

- Coding of the difference image using DCT transform and run length encoding.
- Entropy encoding to further reduce the image size.

1.2.1.3.3.1 Motion detection

The second stage of the H.261 P-frame coding process uses the fact that most images in a video sequence are strongly related. If the camera angle changes, many pixels will simply shift from one image to another. If an object moves in the scene, most of the pixels representing the object in a frame can be copied from the preceding frame with a shift. For each macroblock of the image to be encoded, the algorithm tries to discover whether it is a translated macroblock of the previous image. The search is done in the vicinity of ± 15 pixels and only considers luminance. The difference between the original macroblock of the $n+1$ frame and each translated block of the n frame in the search area is the absolute value of pixel-to-pixel luminance difference throughout the block. The translation vector of the best match is considered the motion compensation vector for that macroblock (Figure 1.21). The difference between the translated macroblock and the original block is called the motion compensation macroblock.

If the image has changed completely (e.g., a new sequence in a movie), interframe coding is not optimal. Further reason, the H.261 coding process must decide at each frame which coding is better for the macroblock: intra or interframe. The decision function is based on the energy and variance of the original macroblock and the motion-compensated macroblock.

1.2.1.3.3.2 DCT transform

The pixel values of the image difference that we obtained at the previous step vary slowly within a macroblock. Let’s take such a macroblock and repeat it in two dimensions so that

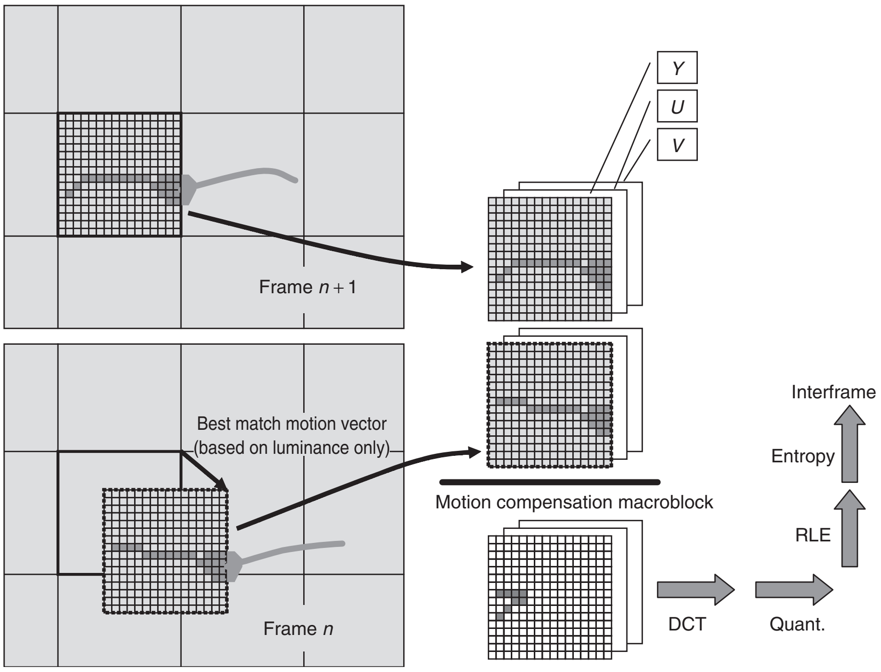

Figure 1.21 Motion prediction and compensation of residual error.

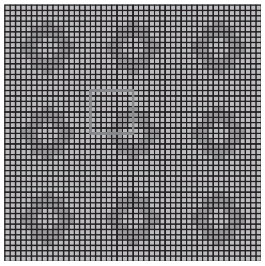

Figure 1.22 Construction of the periodic function used for the DCT transform from the reference macroblock.

we obtain a periodic function (Figure 1.22). Such a function can be reproduced efficiently using just a few coefficients from its Fourier transform.

This transformation is called a bidimensional DCT. The formula used by H.261 to calculate the DCT of an 8∗8 block is

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j)\cos\left((2i+1)u\frac{\pi}{16}\right)\cos\left((2j+1)v\frac{\pi}{16}\right) \quad (1.1)$$

where $C(0) = 1/\sqrt{2}$ and $C(x \neq 0) = 1$. The DCT is a ‘frequency’ representation of the original image. The coefficient in the upper left corner is the mean pixel value of the image. Values in higher row positions represent higher vertical frequencies and values in higher column positions represent higher horizontal frequencies.

The DCT is very interesting because most high-frequency coefficients are usually near 0. At the decoder and, the inverse of the DCT is obtained with:

$$f(i,j) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\left((2i+1)u\frac{\pi}{16}\right)\cos\left((2j+1)v\frac{\pi}{16}\right) \qquad (1.2)$$

1.2.1.3.3.3 Quantization

So far the representation of the image that we have is still exact. We could obtain the original image by reversing the DCT and repeatedly adding the resulting block to the shifted block of the previous frame.

Quantization is the lossy stage in H.261; it consists in expressing each frequency domain $F(u,v)$ value in coarser units, so that the absolute value to be coded decreases and the number of zeros increases. This is done using standard quantization functions: one is used for the constant component (DC) coefficient and another is selected for a macroblock. Depending on the amount of loss that can be tolerated, the coder can choose fine or very coarse functions.

1.2.1.3.3.4 Zigzag scanning and entropy coding

Once the DCT coefficients are quantized, they are rearranged in a chain with the DC coefficient first and then they follow the sequence shown in Figure 1.23. This concentrates most nonzero values at the beginning of the chain. Because there are long series of consecutive zeros, the chain is then run length-encoded. This uses an escape code for

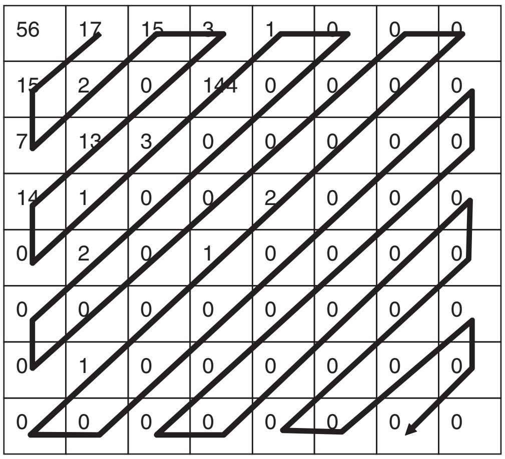

Figure 1.23 Zigzag scanning.

the most frequently occurring sequences of zeros followed by a nonzero coefficient and variable escape codes for other less frequently occurring combinations.

This chain can be further compressed using entropy coding (similar to Huffmann coding), which creates smaller code words for frequently occurring symbols.

Huffmann coding first sorts the values to be encoded according to frequency of appearance, then constructs a tree by aggregating the two least frequent values in a branch, then repeating the process with the two values/branches that have the smallest occurrence values (counting the occurrence of a branch as the sum of the occurrences of its leaf nodes). Once the tree is complete, a '1' is assigned to each left side of any two branches and a '0' to each right side. Any value can be identified by its position in the tree as described by the sequence of digits encountered when progressing from the root of the tree to the value.

The output of the H.261 encoder consists of entropy-encoded DCT values. This bitstream can be easily decoded once the decoder has received the Huffmann tree. In the case of H.261, the tree calculation is not done in real time; the recommendation itself provides codes for the most frequently occurring combinations.

1.2.1.3.3.5 Output format

The H.261 bitstream is organized in GOBs (a group of blocks) of 33 macroblocks (each encoding 16∗16 luminance pixels and 8∗8 *U* and *V* pixels). A PAL CIF image has 12 GOBS, and a PAL QCIF image has 3 GOBS. A CIF picture cannot be larger than 256 kbits, and a QCIF picture cannot be larger than 64 kbits.

The output bitstream will consist of alternating inter-coded macroblocks and intra-coded macroblocks. The receiver can force the use of intra coding to recover from cumulative or transmission errors. Otherwise, a macroblock should be updated in intra mode at least once every 132 transmissions to compensate for error accumulation.

1.2.1.3.3.6 Conclusion on H.261 video streams

The description of H.261 found in the previous sections is not complete, but it is enough to allow a network expert to understand the nature of video traffic. The most important conclusion is that video traffic using H.261-style coding (this is also valid for H.263 and MPEG) is extremely bursty. A typical network load profile is represented in Figure 1.24. For instance, Microsoft Netmeeting sends an intraframe every 15 seconds. A videoconferencing MCU will send an intraframe for all macroblocks each time the speaker, and therefore the image, changes ('videoFastUpdate'). In other circumstances some implementations will not send all intra macroblocks simultaneously, in order to avoid the occurrence of large traffic peaks in the network.

It is also important to remember that H.261 only specifies a decoder. In fact, a very bad implementation could choose to use only intraframes if it was not capable of doing motion vector searches for interframes and still be H.261-compliant. This explains why not all video boards and not all videoconferencing software are equal, despite claiming they are using H.261 or H.263. A network engineer should always try to measure the actual bandwidth used by these devices.

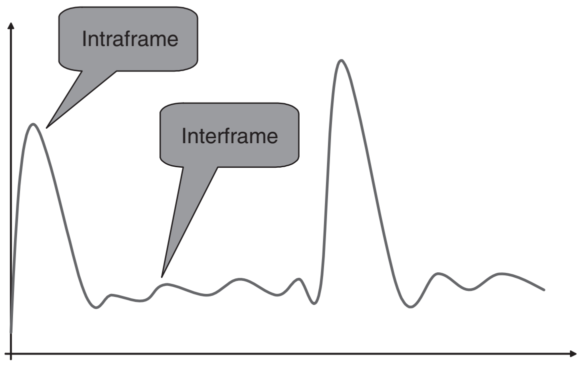

Figure 1.24 Video traffic can be very bursty due to intraframes.

1.2.1.3.4 H.263

Table 1.6 lists the image formats that can be encoded with H.263. H.263 was designed for low-bitrate communication, as low as 20 kbits/s. The coding algorithm of H.263 is similar to that used by H.261, but involves some changes to improve performance and error recovery. H.263 is more recent, more flexible, and about 50% more bitrate-effective than H.261 for the same level of quality. It will replace H.261 in most applications. The main differences between H.261 and H.263 are:

- Half-pixel precision is used by H.263 for motion compensation, whereas H.261 used full-pixel precision and a loop filter. This accounts for much of the improved efficiency.
- Some parts of the hierarchical structure of the data stream are now optional, so the codec can be configured for a lower data rate or better error recovery.
- There are now four negotiable options included to improve performance: unrestricted motion vectors, syntax-based arithmetic coding, advance prediction, and forward and backward frame prediction (similar to MPEG) called P-B frames. Backward frames are added to allow motion vectors to refer not only to past frames, but also to future frames (e.g., when a partly hidden object becomes visible in a future frame).
- H.263 supports five resolutions. In addition to QCIF and CIF, which were supported by H.261, there is SQCIF, 4CIF, and 16CIF. SQCIF is approximately half the resolution of QCIF. 4CIF and 16CIF are 4 and 16 times the resolution of CIF, respectively.

Table 1.6 Image sizes supported by H.263

SQCIF	128∗96	Required
QCIF	176∗144	Required
CIF	352∗288	Optional
4CIF	704∗576	Optional
16CIF	1,408∗1,152	Optional

Support of 4CIF and 16CIF means the codec can now compete with other higher bitrate video-coding standards, such as the MPEG standards.

With these improvements, H.263 is a good challenger to MPEG-1 and MPEG-2 for low to medium resolutions and bitrates. They have comparable features (such as B frames in MPEG and P-B frames in H.263) which are just as good for moderate movements. H.263 even has some options not found in MPEG, like motion vectors outside the picture and syntax-based arithmetic coding. MPEG has more flexibility, but flexibility means overhead. For videoconferencing applications, with little movement and a strong bandwidth constraint, H.263 is a very good choice.

1.2.1.3.5 H.264

H.264 is the latest ITU-standardized video coder and the latest video compression profile for MPEG-4 (part 10). Its production required more than 7 years of work. H.264, or **advanced video coding (AVC)**, requires only one-half to one-third of the video bandwidth necessary for an equivalent MPEG-2 channel when using all the possible optimizations of H.264 (Figure 1.25). Broadcast quality video becomes possible at a rate of 1.5 Mbit/s. This is likely to trigger an accelerated development of on-demand video over IP, in the same way that the ‘MP3’ format made musical applications popular on the Internet. With the traditional rate of 3.75 Mbit/s for MPEG2 movies, delivering video over ADSL is restricted only to the shortest copper lines and densely populated areas. Below 2 Mbps it is possible to add video streaming to many more ADSL lines. It also makes it easier to

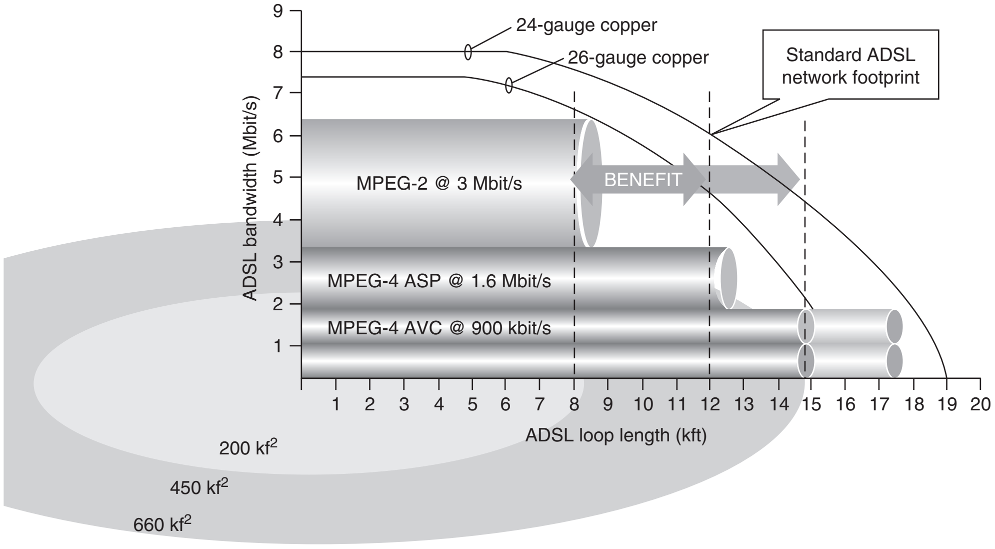

Figure 1.25 H.264/AVC encoding extends the reach of video over ADSL. Reproduced with permission from Envivio, Inc.

provide video content over wireless links (H.264/AVC is one of the standard video coders of 3GPPv6).

With H.263, it is possible to have a business quality videoconference at about 386 kbit/s. With H.264, an equivalent conference can be achieved at about 192 kbit/s. The downside of H.264 is that it requires much more CPU power for compression than H.263. In 2005 the first implementations were able to run only on high end PCs, but now any computer, TV decoder and most video cameras support H.264. In addition, some of the optimizations introduced by H.264 (e.g., the ability to encode interframes referring to future frames) can only be used in non-real-time mode.

In line with the other MPEG standards, H.264 only describes the format of the encoded bitstream and gives no indication of the algorithms that should be used to generate the encoded data. Prediction, DCT, quantization, and entropy encoding are not fundamentally different from the previous standard, but they have been enhanced.

Each frame is processed in 16∗16-pixel macroblocks, each one being encoded in intra or inter mode. In intra mode, the encoded macroblock contains interpolation data using previously encoded macroblocks of the same frame. In inter mode, the encoded macroblock contains motion compensation information based on previous or future frames (up to two previous or subsequent frames).[4] One of the improvements of H.264 over its predecessors is that it allows intra or inter mode to be selected, not at each image, but in groups within the image called ‘slices’.

In inter mode, the ability to refer to frames not immediately adjacent to the frame currently encoded is one of the major optimizations of H.264 compared with the previous generation of video coders. The difference between the predicted macroblock and the macroblock to encode is then computed, block-transformed, quantized, and the reordered coefficients are then entropy-encoded. The bitstream is formed from entropy-encoded coefficients, the quantizer step size, and the information required to recreate the predicted macroblock (motion-compensated vector, etc.).

In intra mode, prediction data describing a block can be built for either 4∗4 or 16∗16 luminance macroblocks, and for the corresponding 8∗8 chrominance macroblock. Prediction block data are built from already-encoded pixels (light gray bands in Figure 1.26), using one of eight extrapolation modes, each based on a characteristic extrapolation direction angle (Figure 1.26 shows mode 4, diagonal to the right of the P-frame, for a 4∗4 luminance macroblock). The mode resulting in the smallest sum of absolute errors compared with the original macroblock is selected.

Both in intra and inter modes, H.264 uses a new ‘deblocking’ filter that considerably reduces the differences between macroblocks in the reconstructed image, which were clearly visible with coders of the previous generation. This filter operates on the reconstructed image just before the differences between the reconstructed macroblocks and the original image are encoded: it smoothes the reconstructed image by reducing the differences across adjacent macroblocks, thereby eliminating in the next phase the brutal changes in difference compensation values between the original image and these adjacent macroblocks.

[4] In the baseline profile, which is more suitable for interactive videoconferencing, only P-frames and I-frames are supported (no backward prediction).

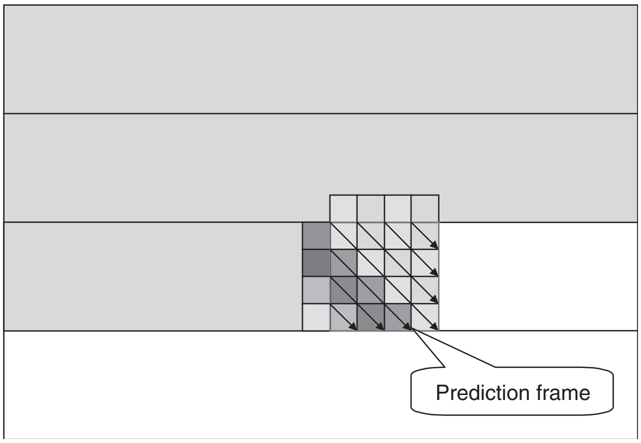

Figure 1.26 Prediction in H.264's intra mode (mode 4).

1.2.2 DTMF

Strictly speaking, DTMF tones that are generated by a touchtone telephone when you press a key are part of the media stream. They are just another sound transmitted by the telephone. In the circuit-switched network this sound is digitized by the G.711 codec as part of the media stream and played back at the receiving end of the line. This does not cause any problem because G.711 does not assume that the signal is voice.

But, some narrow-band codecs that achieve much higher compression rates do use the fact that the signal is voice. Others do not assume the signal is voice, but distort it in such a way that the pure frequencies composing the DTMF tone cannot be correctly recognized when the signal is regenerated. DTMF will not get through these codecs.

Whenever a communication involves an IVR system, it is very important to be able to reliably transmit DTMF tones. In most cases the IVR system just asks a question and waits for a DTMF response. It just cares about which key has been pressed, the exact duration and timing of the tone is not so important. In other cases the IVR system will need more accuracy in the timing (e.g., when the system reads a list and asks you to press the star key when you hear something of interest).

In order to interwork properly with these IVR systems, it was necessary to develop special procedures to handle DTMF:

- H.323 generally uses the signaling channel (H.245 UserInputIndication) to convey DTMF tones (in fully decoded form). This method is sufficient in most cases and works with application servers that need to implement switching functions (e.g., contact centers), without accessing the media stream. Alternatively, since H.323v4 it is also possible to use RFC 2833, which transmits the DTMF tone in fully decoded form, but over the RTP channel. RFC 2833 mandates implementations to be able to handle this telephony event channel as a separate channel (i.e., it should not necessarily be sent to the destination address of other media streams). Unfortunately, most current implementations cannot do this, thereby preventing the service provider from being able to implement application servers in the network. The use of RFC 2833 should be discouraged unless the implementation can correctly send the DTMF information to the application server.

- SIP mainly uses two methods: a signaling method, based on the INFO message or the NOTIFY message (see Chapter 3 for details), which is still not well standardized, or RFC 2833/4733. The same comments apply to RFC 2833. Most implementations do not allow the sending of DTMF information to the application server. De facto it is very difficult in current SIP networks to implement a standards-based application server that is not accessing the media stream.
- MGCP uses a sophisticated out-of-band mechanism that allows the transmission of most telephony events to the call agent in the signaling stream, and implements filtering and accumulation capabilities as well. The mechanism uses the request notification (RQNT) and notify (NTFY) messages. See Chapter 5 for more details.

1.2.3 Fax

1.2.3.1 A short primer on Group 3 fax technology

The purpose of facsimile transmission is to transmit one or several pages of a document across the telephone network. The first fax systems used Group 1 or Group 2 technologies, which scanned the document line by line and converted each line in black or white pixels. The data were then transmitted without compression over the phone line at the rate of 3 lines per second for Group 1 and 6 lines per second for Group 3.

Because this took over 3 minutes for an A4 document (1,145 lines of 1,728 bits) even in the best case, Group 3 technology was introduced. Group 3 faxes use a more efficient image-coding mechanism known as modified Huffmann coding (MH). MH coding uses the fact that each line is composed of large sequences of white pixels and large sequences of black pixels. Instead of sending data for each pixel MH coding just sends a short code for the sequence. Now the transmission time depends on the document, but is usually much shorter than 3 min: no wonder Group 3 faxes today rule the fax market.

With the advent of ISDN, Group 4 faxes have been introduced. The main difference from Group 3 is that ISDN can transmit raw data, so Group 4 technology need not care about the many hacks that are needed to carry data over an analog line. However, Group 4 has not succeeded in gaining a significant market, and the probability of having a Group 4 fax talking to another Group 4 fax is so low that this case has not so far been considered in the ITU's SG16 which is in charge of H.323.

1.2.3.1.1 Transmitting a line (Group 3)

Most faxes are physically linked to a printer. Because of compression, it is now possible to transmit a single line very quickly if the line is simple, so quickly that the receiving fax may not have enough time to print it. Of course the fax could buffer it in memory, but most faxes are very simple devices with very little memory. Therefore Group 3 supports

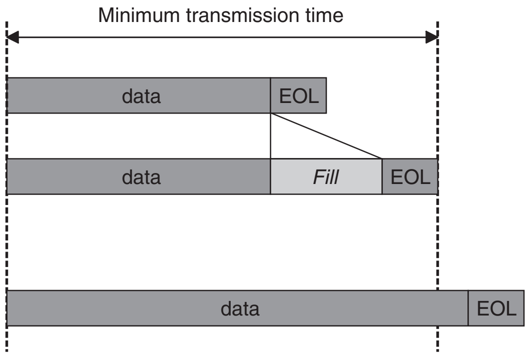

Figure 1.27 Fax line transmission.

a minimum transmission time, as represented in Figure 1.27. If a line does not contain enough compressed data to take more than the MTT to be transmitted, a filling sequence of zeros will be added before the end of line sequence.

1.2.3.1.2 Transmitting a page

As Figure 1.28 shows, the transmission of a page is quite simple. Each line is transmitted in sequence, separated by an EOL, and the whole page is terminated by six consecutive EOLs, which means the fax has to return to command mode (RTC).

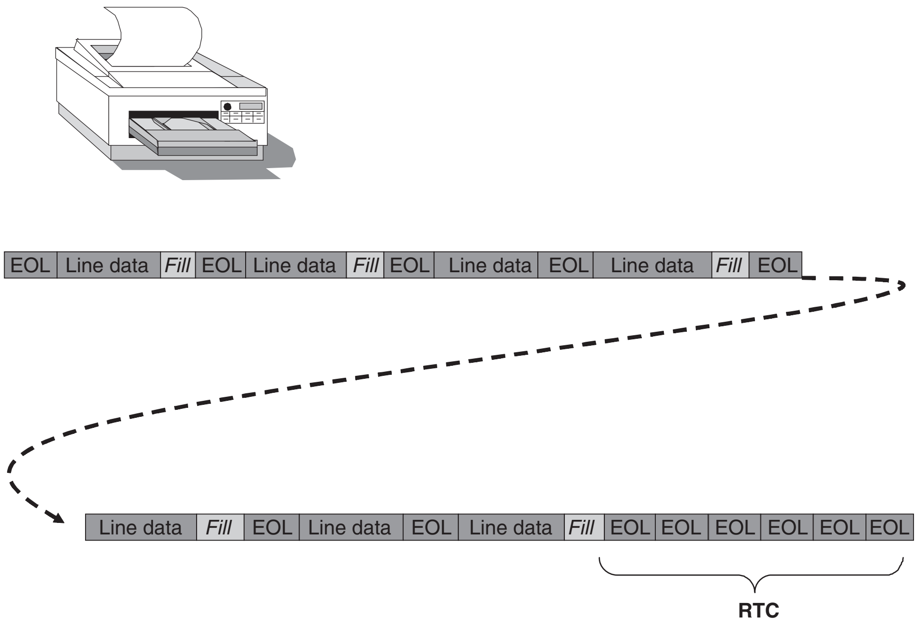

Figure 1.28 Fax page transmission.

1.2.3.1.3 Complete fax transmission

The calling fax dials the destination number, then sends a special sequence called CNG (CalliNG tone), which consists of a repetition of 1,100-Hz tones sent for 0.5 seconds separated by 3 seconds of silence (Figure 1.29). Faxes manufactured before 1993 may not send this tone.

When an incoming connection arrives at the receiving fax, it first sends a special 2,100-Hz tone called CED for 3 seconds. After a short pause, the receiving fax (1) begins to send commands using V.21 modulation (quite slow at 300 bit/s), (2) to transmit synchronizing flags for 1 second (called a preamble), (3) may transmit some non-standardized data

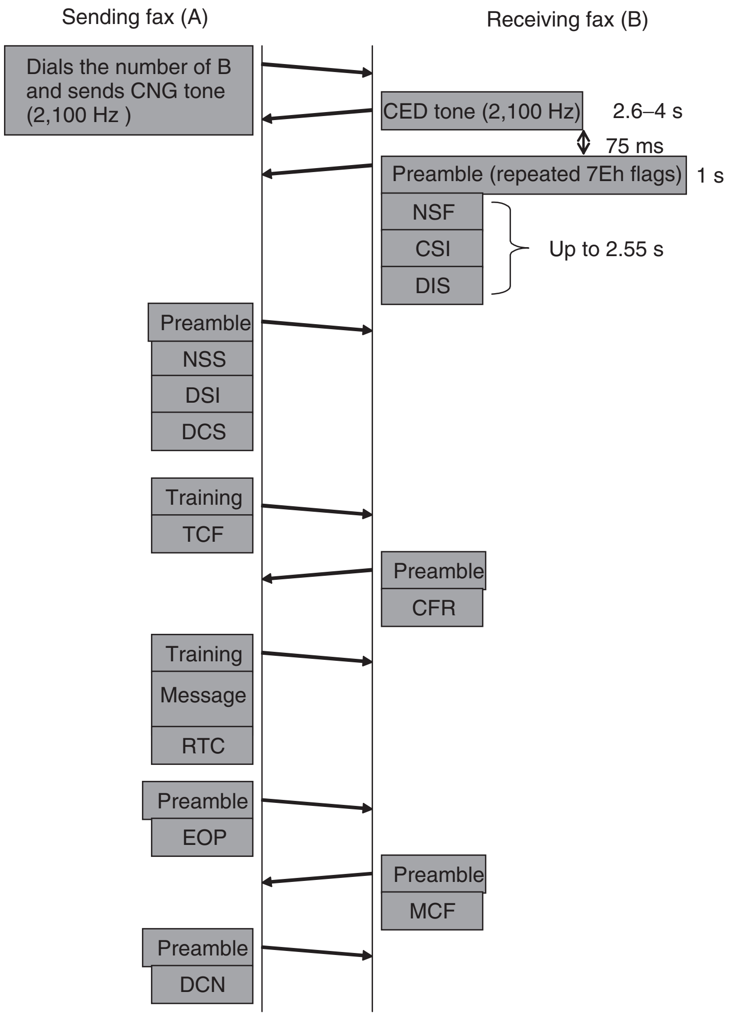

Figure 1.29 Overview of a fax transmission.

(NSF) and its local identity (CSI), and (4) must transmit its capabilities (DIS, or digital identification signal). Each of these data elements is an HDLC frame that consists of:

- A starting flag (7Eh).
- An address field (always set to FFh).
- A command field which is set to C8h for a final frame and C0h otherwise.
- A fax control field (FCF): 02h for CSI, 01h for DIS, etc.
- A variable length fax information field (FIF).
- A checksum (FCS, or frame check sequence).

Transmitting NSF, CSI, and DIS may take up to 2.5 seconds.

The sending fax selects a mode of transmission (DCS) and replies by sending its own capabilities and its identity (TSI). As soon as the receiving fax is ready the sending fax begins the actual transmission phase and will use a faster modulation scheme, such as V.27 (4,800 bits/s) or V.29 (9,600 bits/s). This requires a training phase which is used by the receiving side to compensate for phase distortions and other issues. At the end of the training phase the sending fax sends zeros for 1.5 seconds (called a training check, or TCF). If the called fax receives this sequence correctly it considers the training phase successful and sends a CFR (ConFirmation to Receive) command to let the transmitting fax know it has succeeded. After another training sequence, the sending fax transmits the actual page data as formatted above. This takes approximately 30 seconds in V.29 mode and 1 minute in V.27 mode.

When this is finished, the modem can send an MPS (multi-page signaling) message to send another page or an EOP (end of procedure message) when it has transmitted the last page. The receiving fax acknowledges it with an MCF (Message ConFirmation) which means that the image data have been correctly received, and the sending fax sends a disconnection message DCN (DisCoNnect).

1.2.3.1.4 Detection of fax for VoIP gateways

It is important to reliably detect faxes on VoIP gateways, since fax modulation is not reliably transmitted across low-bitrate voice codecs. On the originating gateway, the T.30 calling tone can be detected, but it is an optional signal. Therefore, detecting CNG is not a reliable way to detect a fax signal. This can be resolved at the terminating gateway by detecting the V.21 preamble flag sequence which follows the called station identification tone (CED), when the CED is present. The CED itself cannot be used because it is also used by modems (V.25 ANS modem tone).

As soon as the signal is detected as a fax, the gateway should stop using regular audio encoding and switch to T.38 encoding.

1.2.3.1.5 Error conditions

If the training is not successful, the receiving fax can send an FTT command to ask for another try at a lower speed.

If an error is present in a line, the receiving fax will find it by counting how many pixels are present in the decoded line. If there are not exactly 1,728 (A4 format), the line is ignored or copied from the previous line, depending on manufacturer preference.

A fax can request the retransmission of a command at any time by sending a CRP command.

1.2.3.2 Fax transmission over IP (T.38 and T.37)

1.2.3.2.1 Store-and-forward fax and the challenge of real-time fax

Sending faxes over the Internet is not something new. Many companies have been offering this service, called store-and-forward fax, for some time. The idea behind store-and-forward fax is quite simple. When computer *A* receives a fax, the fax data are represented as a set of bitmaps. This set of bitmaps is a file that can be transmitted to another computer (*B*) closer to the destination. Once this computer has received the file, it just needs to dial the receiving fax machine and emulate a fax machine to send the bitmap.

This technique is also used for bulk faxing, in which the original document is faxed once to a computer, and the computer is then provided with a list of fax numbers and sends a copy of this fax to each of them. Store-and-forward fax transmission is now standardized at ITU in recommendation T.37. However, since this book focuses on real-time applications we choose to put the emphasis on the real-time standard, T.38.

The problem with store-and-forward fax technology is that many faxes report back on the transmission of the document. Usually, they keep the result code in memory and then print it on demand. Many people tend to rely on these transmission reports: for fax-to-fax transmission this is confirmation that the fax has been correctly received, with a timestamp and the identity of the receiving fax machine.

When using store-and-forward, this report is only a confirmation that the fax has been sent, because the receiving machine is in fact computer *A*.

When it receives the file containing the document, computer *B* will dial the number indicated. But, the fax could be busy or, even worse, it could be a wrong number. So any company providing a store-and-forward service needs to report back to the sender, via email or fax. When receiving a negative acknowledgement the sender needs to know whether it is a problem with the receiving fax or the provider. This leads to potential conflicts and increases the cost of providing the service.

It is much easier for a service provider to be completely transparent in the transmission. In other words, the success report that is received by the originating fax machine should appear as a success report from the distant fax machine: such a service is real-time fax.

Real-time fax is much more complex than store-and-forward fax. There are many timers in the T.30 protocol. Once computer *A* has picked up the line, computer *B* has only a limited time budget to dial the other fax machine and get an answer. During the call, when *A*'s fax machine has sent a command, it expects a reply within 3 seconds. So, during this limited time budget *A* must send the command to *B* over the Internet, *B* must send it to the receiving fax machine, receive the reply, and forward it to *A*.

Fortunately, the ITU had a human operator in mind when setting the value of these timers; so, all are expressed in seconds. Moreover, as we saw in the preceding

Section 1.2.3.1.5 there are many ways of recovering from error conditions, which can be used to spoof the sending fax and get it to wait a little more if needed. These techniques are quite difficult to implement reliably with all brands of faxes. However, some manufacturers have built up a lot of experience and have announced they could transmit real-time faxes over IP networks with a round trip latency of up to 2 seconds!

Lately, many carriers have been tempted to do IP trunking without telling their customers. VoIP gateways make this quite simple. But, without support for real-time fax, whenever a subscriber tries to send a fax that is routed through this IP trunk, it will fail miserably. Of course, it is possible to tell subscribers not to use their faxes or even to dial a special prefix for faxes; but, this significantly complicates their lives. Real-time fax is the only appropriate answer to these issues: all VoIP gateways should be able to dynamically recognize a fax call and switch to T.38 transport mode.

1.2.3.2.2 T.38

1.2.3.2.2.1 IFP

T.38 is the approach of the ITU’s SG16 to the problem of real-time fax. Its title is ‘Procedures for real-time Group 3 facsimile communication between terminals using IP networks’. This recommendation is limited to Group 3 only and describes real-time fax transmission using VoIP gateways over an IP network, between faxes and computers connected on the Internet, or even between computers (the latter may not seem useful, but in some cases the receiving computer will be identified by an H.323 alias or even a phone number, and you may not know this is a computer). Usage of the T.38 protocol within the framework of H.323 is defined in H.323 annex D and was included in H.323v3.

T.38 uses a special transport protocol called IFP. IFP packets can be carried over TCP or UDP. Most gateways support UDP, but TCP transport has also been made mandatory in H.323v4. UDP transport includes a forward error correction mechanism.

IFP packets contain a type field and a data field (Figure 1.30) both encoded using ASN.1 syntax. The type field can have three values:

- T30_INDICATOR: the value of this indicator gives information on received CED and CNG tones, V.21 preambles, and V.27, V.29, and V.17 modulation training.
- T30_DATA: the value of this indicator tells us over which transport (V.21, V.17, or V.29) the data part of the message has been received.
- DISCONNECT: used to disconnect the session normally or after a failure, the value describes the error code.

The DATA part of the IFP message contains T.30 control messages as well as the image data. This DATA element is organized in fields that contain a field type and field data. Examples of these fields are:

- Type HDLC data: the data part of the field contains one, or part of an HDLC data frame, not including the checksum (FCS). This is coded as an ASN-1 octet.
- Type FCS OK: indicates that an HDLC frame is finished and the FCS has been checked. There are still other HDLC frames after an FCS OK.

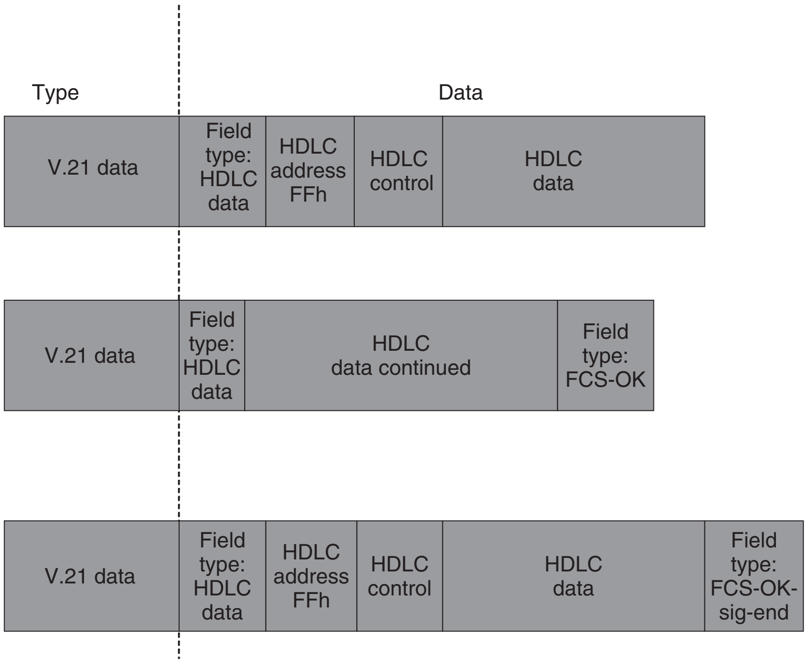

Figure 1.30 IFP packet formats for the first, middle, and last HDLC frames.

- Type FCS OK-sig-end: same as FCS OK, except that this is the last HDLC frame.
- T4-non-ECM: the data part contains the actual image data including filling and RTC.

1.2.3.2.2.2 IFP over TCP or UDP

IFP messages can be carried as TCP payload or can be encapsulated in UDP, as shown in Figure 1.31.

An additional redundancy mechanism has been defined on top of UDP in order to make the delivery of IFP packets more reliable. As shown in Figure 1.31, the payload part contains one or more IFP messages, and the sequence number that appears in the header is the sequence number of the first IFP message in the payload, which is also called the primary message. The first message sent by a gateway should have a sequence number of 0. After this primary message other messages are inserted for error/loss recovery purposes; two modes can be used for this: redundancy mode and FEC (forward error correction) mode.

The control header indicates whether the secondary messages are redundancy messages (bit 3 set to 0) or FEC messages (bit 3 set to 1).

(a) Redundancy mode

In redundancy mode, copies of previous IFP messages are simply inserted after the primary message (Figure 1.32). The number of copies is the number of frames minus one. By

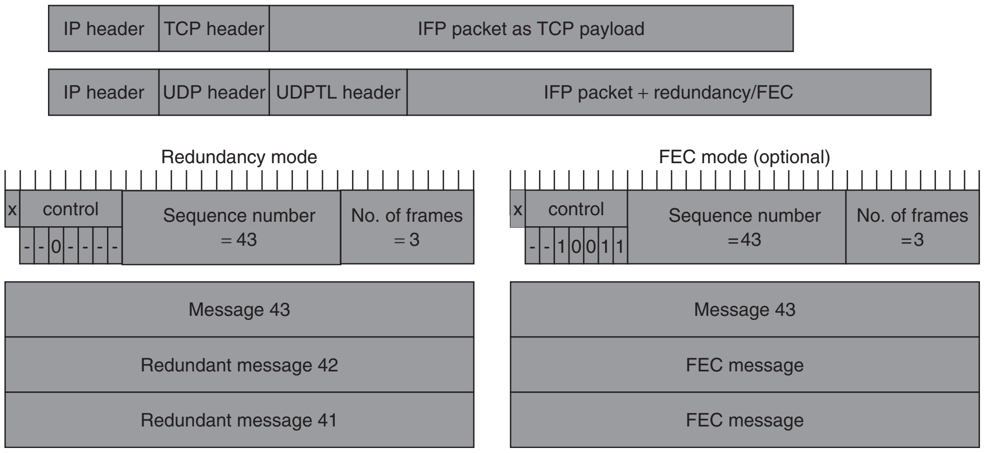

Figure 1.31 IFP transport methods and error correction modes.

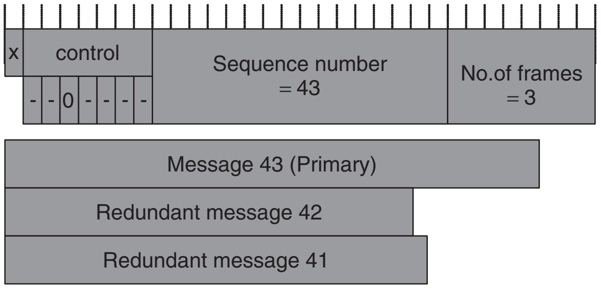

Figure 1.32 Redundancy mode.

adding n copies to the message, the transmission is protected against loss of up to n consecutive packets.

A gateway is not required to transmit redundancy packets, and receiving gateways that do not support them may simply ignore the presence of redundancy packets.

(b) FEC mode

FEC mode is more complex. Each FEC message is the result of a bit-per-bit exclusive-OR performed on n primary IFP messages. Before performing the OR, shorter messages are right-padded with zeros, so the resulting FEC message is as long as the longest n primary message. The value of n is indicated in the four last digits of the control field, '3' in Figure 1.33.

When several FEC messages are added, as in the left part of our example, the primary messages used for each FEC message are interleaved. When n FEC messages are added, transmission is protected against the loss of n consecutive UDP packets.

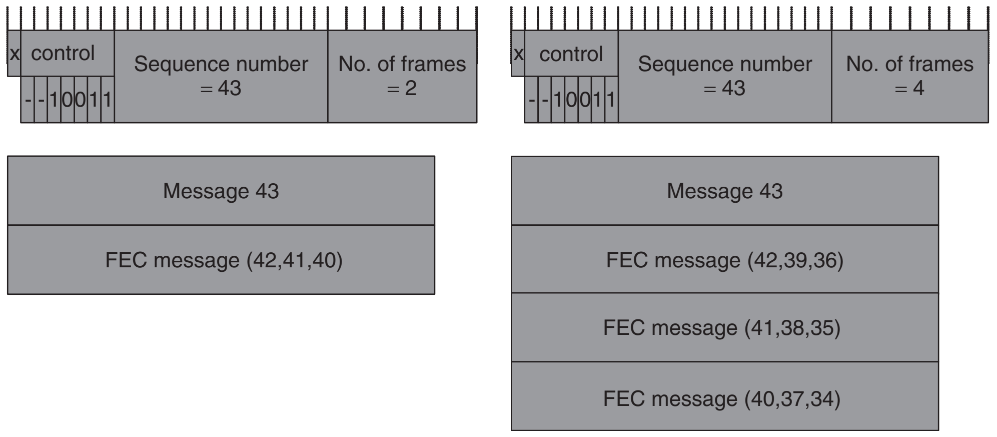

Figure 1.33 FEC mode.

1.2.3.2.2.3 T.38, H.323, and SIP

The use of T.38 by SIP is explained in T.38 annex D ‘SIP/SDP call establishment procedures’. The use of T.38 with the H.323 protocol is described in H.323 annex D and T.38 annex B. H.323 annex D mandates the use of IFP transport over TCP, but transport over UDP is still allowed as an optional mode. In reality, all vendors seem to use IFP over UDP. These capabilities (T38-TCP and T38-UDP) have been added in the DataApplicationCapability of DataProtocolCapability of H.245. IFP is transmitted over two logical channels (sender to receiver and vice versa).