Data Transmission in Telecommunication Engineering

Sulema Dougherty

First Edition, 2012

ISBN 978-81-323-3038-7

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Introduction

Data transmission, **digital transmission**, or **digital communications** is the physical transfer of data (a digital bit stream) over a point-to-point or point-to-multipoint communication channel. Examples of such channels are copper wires, optical fibres, wireless communication channels, and storage media. The data is represented as an electromagnetic signal, such as an electrical voltage, radiowave, microwave, or infrared signal.

While analog communications is the transfer of continuously varying information signal, digital communications is the transfer of discrete messages. The messages are either represented by a sequence of pulses by means of a line code (*baseband transmission*), or by a limited set of continuously varying wave forms (*passband transmission*), using a digital modulation method. The passband modulation and corresponding demodulation (also known as detection) is carried out by modem equipment. According to the most common definition of digital signal, both baseband and passband signals representing bit-streams are considered as digital transmission, while an alternative definition only considers the baseband signal as digital, and passband transmission of digital data as a form of digital-to-analog conversion.

Data transmitted may be digital messages originating from a data source, for example a computer or a keyboard. It may also be an analog signal such as a phone call or a video signal, digitized into a bit-stream for example using pulse-code modulation (PCM) or more advanced source coding (analog-to-digital conversion and data compression) schemes. This source coding and decoding is carried out by codec equipment.

Distinction between related subjects

Courses and textbooks in the field of *data transmission* as well as *digital transmission* and *digital communications* have similar content.

Digital transmission or data transmission traditionally belongs to telecommunications and electrical engineering. Basic principles of data transmission may also be covered within the computer science/computer engineering topic of data communications, which also

includes computer networking or computer communication applications and networking protocols, for example routing, switching and process-to-process communication. Although the Transmission control protocol (TCP) involves the term "transmission", TCP and other transport layer protocols are typically *not* discussed in a textbook or course about data transmission, but in computer networking.

The term tele transmission involves the analog as well as digital communication. In most textbooks, the term analog transmission only refers to the transmission of an analog message signal (without digitization) by means of an analog signal, either as a non-modulated baseband signal, or as a passband signal using an analog modulation method such as AM or FM. It may also include analog-over-analog pulse modulatated baseband signals such as pulse-width modulation. In a few books within the computer networking tradition, "analog transmission" also refers to passband transmission of bit-streams using digital modulation methods such as FSK, PSK and ASK. Note that these methods are covered in textbooks named digital transmission or data transmission, for example.

The theoretical aspects of data transmission are covered by information theory and coding theory.

Protocol layers and sub-topics

Courses and textbooks in the field of data transmission typically deal with the following OSI model protocol layers and topics:

- Layer 1, the physical layer:
  - Channel coding including
    - Digital modulation schemes
    - Line coding schemes
    - Forward error correction (FEC) codes
  - Bit synchronization
  - Multiplexing
  - Equalization
  - Channel models
- Layer 2, the data link layer:
  - Channel access schemes, media access control (MAC)
  - Packet mode communication and Frame synchronization
  - Error detection and automatic repeat request (ARQ)
  - Flow control
- Layer 6, the presentation layer:
  - Source coding (digitization and data compression), and information theory.
  - Cryptography (may occur at any layer)

Applications and history

Data (mainly but not exclusively informational) has been sent via non-electronic (e.g. optical, acoustic, mechanical) means since the advent of communication. Analog signal data has been sent electronically since the advent of the telephone. However, the first data electromagnetic transmission applications in modern time were telegraphy (1809) and teletypewriters (1906), which are both digital signals. The fundamental theoretical work in data transmission and information theory by Harry Nyquist, Ralph Hartley, Claude Shannon and others during the early 20th century, was done with these applications in mind.

Data transmission is utilized in computers in computer buses and for communication with peripheral equipment via parallel ports and serial ports such us RS-232 (1969), Firewire (1995) and USB (1996). The principles of data transmission is also utilized in storage media for Error detection and correction since 1951.

Data transmission is utilized in computer networking equipment such as modems (1940), local area networks (LAN) adapters (1964), repeaters, hubs, microwave links, wireless network access points (1997), etc.

In telephone networks, digital communication is utilized for transferring many phone calls over the same copper cable or fiber cable by means of Pulse code modulation (PCM), i.e. sampling and digitization, in combination with Time division multiplexing (TDM) (1962). Telephone exchanges have become digital and software controlled, facilitating many value added services. For example the first AXE telephone exchange was presented in 1976. Since late 1980th, digital communication to the end user has been possible using Integrated Services Digital Network (ISDN) services. Since the end of 1990th, broadband access techniques such as ADSL, Cable modems, fiber-to-the-building (FTTB) and fiber-to-the-home (FTTH) have become wide spread to small offices and homes. The current tendency is to replace traditional telecommunication services by packet mode communication such as IP telephony and IPTV.

Transmitting analog signals digitally allows for greater signal processing capability. The ability to process a communications signal means that errors caused by random processes can be detected and corrected. Digital signals can also be sampled instead of continuously monitored. The multiplexing of multiple digital signals is much simpler to the multiplexing of analog signals.

Because of all these advantages, and because recent advances in wideband communication channels and solid-state electronics have allowed scientists to fully realize these advantages, digital communications has grown quickly. Digital communications is quickly edging out analog communication because of the vast demand to transmit computer data and the ability of digital communications to do so.

The digital revolution has also resulted in many digital telecommunication applications where the principles of data transmission are applied. Examples are second-generation

(1991) and later cellular telephony, video conferencing, digital TV (1998), digital radio (1999), telemetry, etc.

Baseband or passband transmission

The physically transmitted signal may be one of the following:

1. **A baseband signal** ("digital-over-digital" transmission): A sequence of electrical pulses or light pulses produced by means of a line coding scheme such as Manchester coding. This is typically used in serial cables, wired local area networks such as Ethernet, and in optical fiber communication. It results in a pulse amplitude modulated signal, also known as a pulse train.
2. **A passband signal** ("digital-over-analog" transmission): A modulated sine wave signal representing a digital bit-stream. Note that this is in some textbooks considered as analog transmission, but in most books as digital transmission. The signal is produced by means of a digital modulation method such as PSK, QAM or FSK. The modulation and demodulation is carried out by modem equipment. This is used in wireless communication, and over telephone network local-loop and cable-TV networks.

Serial and parallel transmission

In telecommunications, serial transmission is the sequential transmission of signal elements of a group representing a character or other entity of data. Digital serial transmissions are bits sent over a single wire, frequency or optical path sequentially. Because it requires less signal processing and less chances for error than parallel transmission, the transfer rate of each individual path may be faster. This can be used over longer distances as a check digit or parity bit can be sent along it easily.

In telecommunications, parallel transmission is the simultaneous transmission of the signal elements of a character or other entity of data. In digital communications, parallel transmission is the simultaneous transmission of related signal elements over two or more separate paths. Multiple electrical wires are used which can transmit multiple bits simultaneously, which allows for higher data transfer rates than can be achieved with serial transmission. This method is used internally within the computer, for example the internal buses, and sometimes externally for such things as printers, The major issue with this is "skewing" because the wires in parallel data transmission have slightly different properties (not intentionally) so some bits may arrive before others, which may corrupt the message. A parity bit can help to reduce this. However, electrical wire parallel data transmission is therefore less reliable for long distances because corrupt transmissions are far more likely.

Types of communication channels

- Simplex
- Half-duplex
- Full-duplex
- Point-to-point
- Multi-drop:
    - Bus network
    - Ring network
    - Star network
    - Mesh network
    - Wireless network

Asynchronous and synchronous data transmission

Asynchronous transmission uses start and stop bits to signify the beginning bit ASCII character would actually be transmitted using 10 bits e.g.: A "0100 0001" would become "**1** 0100 0001 **0**". The extra one (or zero depending on parity bit) at the start and end of the transmission tells the receiver first that a character is coming and secondly that the character has ended. This method of transmission is used when data is sent intermittently as opposed to in a solid stream. In the previous example the start and stop bits are in bold. The start and stop bits must be of opposite polarity. This allows the receiver to recognize when the second packet of information is being sent.

Synchronous transmission uses no start and stop bits but instead synchronizes transmission speeds at both the receiving and sending end of the transmission using clock signal(s) built into each component. A continual stream of data is then sent between the two nodes. Due to there being no start and stop bits the data transfer rate is quicker although more errors will occur, as the clocks will eventually get out of sync, and the receiving device would have the wrong time that had been agreed in the protocol for sending/receiving data, so some bytes could become corrupted (by losing bits). Ways to get around this problem include re-synchronization of the clocks and use of check digits to ensure the byte is correctly interpreted and received

Chapter 1

Modulation

In electronics, **modulation** is the process of varying one or more properties of a high-frequency periodic waveform, called the *carrier signal*, with respect to a *modulating signal* (which typically contains information to be transmitted). This is done in a similar fashion to a musician modulating a tone (a periodic waveform) from a musical instrument by varying its volume, timing and pitch. The three key parameters of a periodic waveform are its amplitude ("volume"), its phase ("timing") and its frequency ("pitch"), all of which can be modified in accordance with a low frequency signal to obtain the modulated signal. Typically a high-frequency sinusoid waveform is used as carrier signal, but a square wave pulse train may also occur.

In telecommunications, modulation is the process of conveying a message signal, for example a digital bit stream or an analog audio signal, inside another signal that can be physically transmitted. Modulation of a sine waveform is used to transform a baseband message signal into a passband signal, for example low-frequency audio signal into a radio-frequency signal (RF signal). In radio communications, cable TV systems or the public switched telephone network for instance, electrical signals can only be transferred over a limited passband frequency spectrum, with specific (non-zero) lower and upper cutoff frequencies. Modulating a sine-wave carrier makes it possible to keep the frequency content of the transferred signal as close as possible to the centre frequency (typically the carrier frequency) of the passband.

A device that performs modulation is known as a modulator and a device that performs the inverse operation of modulation is known as a demodulator (sometimes *detector* or *demod*). A device that can do both operations is a modem (modulator–demodulator).

Aim

The aim of **digital modulation** is to transfer a digital bit stream over an analog bandpass channel, for example over the public switched telephone network (where a bandpass filter limits the frequency range to between 300 and 3400 Hz), or over a limited radio frequency band.

The aim of **analog modulation** is to transfer an analog baseband (or lowpass) signal, for example an audio signal or TV signal, over an analog bandpass channel, for example a limited radio frequency band or a cable TV network channel.

Analog and digital modulation facilitate frequency division multiplexing (FDM), where several low pass information signals are transferred simultaneously over the same shared physical medium, using separate passband channels.

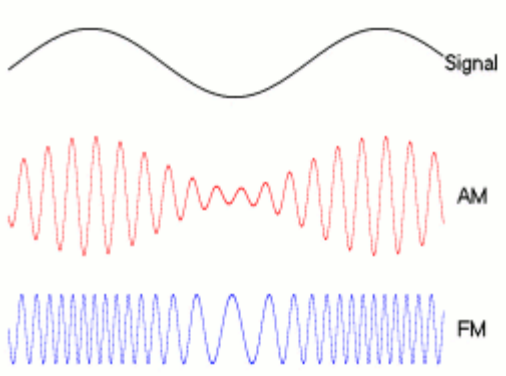
The aim of **digital baseband modulation** methods, also known as line coding, is to transfer a digital bit stream over a baseband channel, typically a non-filtered copper wire such as a serial bus or a wired local area network.

The aim of **pulse modulation** methods is to transfer a narrowband analog signal, for example a phone call over a wideband baseband channel or, in some of the schemes, as a bit stream over another digital transmission system.

In music synthesizers, modulation may be used to synthesise waveforms with a desired overtone spectrum. In this case the carrier frequency is typically in the same order or much lower than the modulating waveform.

Analog modulation methods

In analog modulation, the modulation is applied continuously in response to the analog information signal.



A low-frequency message signal (top) may be carried by an AM or FM radio wave.

Common analog modulation techniques are:

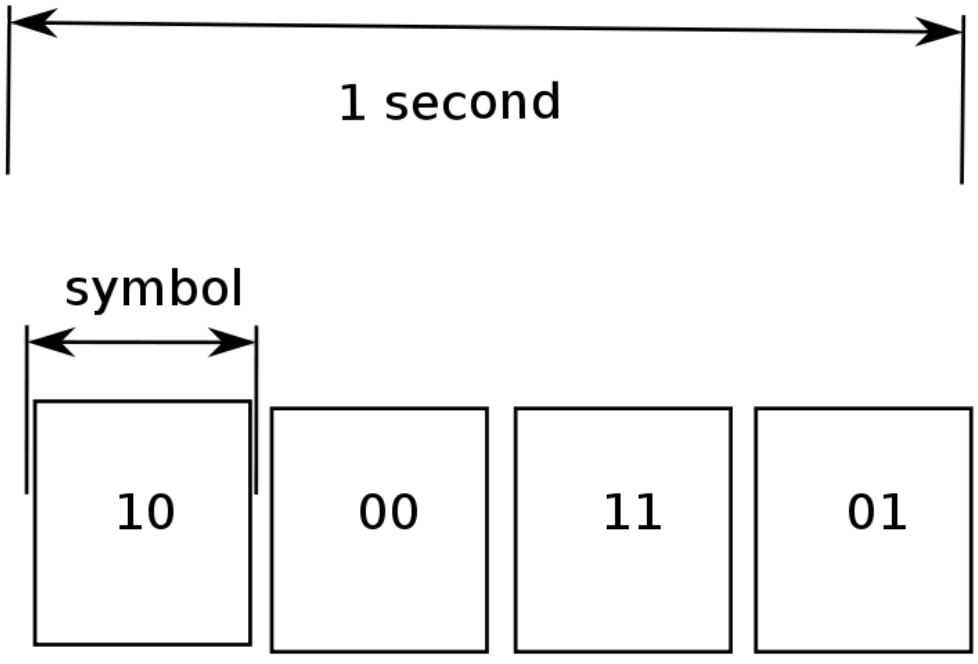
- Amplitude modulation (AM) (here the amplitude of the carrier signal is varied in accordance to the instantaneous amplitude of the modulating signal)
    - Double-sideband modulation (DSB)
        - Double-sideband modulation with carrier (DSB-WC) (used on the AM radio broadcasting band)

        - Double-sideband suppressed-carrier transmission (DSB-SC)
        - Double-sideband reduced carrier transmission (DSB-RC)
    - Single-sideband modulation (SSB, or SSB-AM)
        - SSB with carrier (SSB-WC)
        - SSB suppressed carrier modulation (SSB-SC)
    - Vestigial sideband modulation (VSB, or VSB-AM)
    - Quadrature amplitude modulation (QAM)

- Angle modulation
    - Frequency modulation (FM) (here the frequency of the carrier signal is varied in accordance to the instantaneous amplitude of the modulating signal)
    - Phase modulation (PM) (here the phase shift of the carrier signal is varied in accordance to the instantaneous amplitude of the modulating signal)

Digital modulation methods

In digital modulation, an analog carrier signal is modulated by a digital bit stream. Digital modulation methods can be considered as digital-to-analog conversion, and the corresponding demodulation or detection as analog-to-digital conversion. The changes in the carrier signal are chosen from a finite number of M alternative symbols (the *modulation alphabet*).



Schematic of 4 baud (8 bps) data link.

A simple example: A telephone line is designed for transferring audible sounds, for example tones, and not digital bits (zeros and ones). Computers may however communicate over a telephone line by means of modems, which are representing the digital bits by tones, called symbols. If there are four alternative symbols (corresponding to a musical instrument that can generate four different tones, one at a time), the first symbol may represent the bit sequence 00, the second 01, the third 10 and the fourth 11. If the modem plays a melody consisting of 1000 tones per second, the symbol rate is 1000 symbols/second, or baud. Since each tone (i.e., symbol) represents a message consisting of two digital bits in this example, the bit rate is twice the symbol rate, i.e. 2000 bits per second. This is similar to the technique used by dialup modems as opposed to DSL modems.

According to one definition of digital signal, the modulated signal is a digital signal, and according to another definition, the modulation is a form of digital-to-analog conversion. Most textbooks would consider digital modulation schemes as a form of digital transmission, synonymous to data transmission; very few would consider it as analog transmission.

Fundamental digital modulation methods

The most fundamental digital modulation techniques are based on keying:

- In the case of PSK (phase-shift keying), a finite number of phases are used.
- In the case of FSK (frequency-shift keying), a finite number of frequencies are used.
- In the case of ASK (amplitude-shift keying), a finite number of amplitudes are used.
- In the case of QAM (quadrature amplitude modulation), a finite number of at least two phases, and at least two amplitudes are used.

In QAM, an inphase signal (the I signal, for example a cosine waveform) and a quadrature phase signal (the Q signal, for example a sine wave) are amplitude modulated with a finite number of amplitudes, and summed. It can be seen as a two-channel system, each channel using ASK. The resulting signal is equivalent to a combination of PSK and ASK.

In all of the above methods, each of these phases, frequencies or amplitudes are assigned a unique pattern of binary bits. Usually, each phase, frequency or amplitude encodes an equal number of bits. This number of bits comprises the *symbol* that is represented by the particular phase, frequency or amplitude.

If the alphabet consists of $M = 2^N$ alternative symbols, each symbol represents a message consisting of N bits. If the symbol rate (also known as the baud rate) is f_S symbols/second (or baud), the data rate is Nf_S bit/second.

For example, with an alphabet consisting of 16 alternative symbols, each symbol represents 4 bits. Thus, the data rate is four times the baud rate.

In the case of PSK, ASK or QAM, where the carrier frequency of the modulated signal is constant, the modulation alphabet is often conveniently represented on a constellation diagram, showing the amplitude of the I signal at the x-axis, and the amplitude of the Q signal at the y-axis, for each symbol.

Modulator and detector principles of operation

PSK and ASK, and sometimes also FSK, are often generated and detected using the principle of QAM. The I and Q signals can be combined into a complex-valued signal $I+jQ$ (where j is the imaginary unit). The resulting so called equivalent lowpass signal or equivalent baseband signal is a complex-valued representation of the real-valued modulated physical signal (the so called passband signal or RF signal).

These are the general steps used by the modulator to transmit data:

1. Group the incoming data bits into codewords, one for each symbol that will be transmitted.
2. Map the codewords to attributes, for example amplitudes of the I and Q signals (the equivalent low pass signal), or frequency or phase values.
3. Adapt pulse shaping or some other filtering to limit the bandwidth and form the spectrum of the equivalent low pass signal, typically using digital signal processing.
4. Perform digital-to-analog conversion (DAC) of the I and Q signals (since today all of the above is normally achieved using digital signal processing, DSP).
5. Generate a high-frequency sine wave carrier waveform, and perhaps also a cosine quadrature component. Carry out the modulation, for example by multiplying the sine and cosine wave form with the I and Q signals, resulting in that the equivalent low pass signal is frequency shifted into a modulated passband signal or RF signal. Sometimes this is achieved using DSP technology, for example direct digital synthesis using a waveform table, instead of analog signal processing. In that case the above DAC step should be done after this step.
6. Amplification and analog bandpass filtering to avoid harmonic distortion and periodic spectrum

At the receiver side, the demodulator typically performs:

1. Bandpass filtering.
2. Automatic gain control, AGC (to compensate for attenuation, for example fading).
3. Frequency shifting of the RF signal to the equivalent baseband I and Q signals, or to an intermediate frequency (IF) signal, by multiplying the RF signal with a local oscillator sinewave and cosine wave frequency.

4. Sampling and analog-to-digital conversion (ADC) (Sometimes before or instead of the above point, for example by means of undersampling).
5. Equalization filtering, for example a matched filter, compensation for multipath propagation, time spreading, phase distortion and frequency selective fading, to avoid intersymbol interference and symbol distortion.
6. Detection of the amplitudes of the I and Q signals, or the frequency or phase of the IF signal.
7. Quantization of the amplitudes, frequencies or phases to the nearest allowed symbol values.
8. Mapping of the quantized amplitudes, frequencies or phases to codewords (bit groups).
9. Parallel-to-serial conversion of the codewords into a bit stream.
10. Pass the resultant bit stream on for further processing such as removal of any error-correcting codes.

As is common to all digital communication systems, the design of both the modulator and demodulator must be done simultaneously. Digital modulation schemes are possible because the transmitter-receiver pair have prior knowledge of how data is encoded and represented in the communications system. In all digital communication systems, both the modulator at the transmitter and the demodulator at the receiver are structured so that they perform inverse operations.

Non-coherent modulation methods do not require a receiver reference clock signal that is phase synchronized with the sender carrier wave. In this case, modulation symbols (rather than bits, characters, or data packets) are asynchronously transferred. The opposite is coherent modulation.

List of common digital modulation techniques

The most common digital modulation techniques are:

- Phase-shift keying (PSK):
  - Binary PSK (BPSK), using M=2 symbols
  - Quadrature PSK (QPSK), using M=4 symbols
  - 8PSK, using M=8 symbols
  - 16PSK, using M=16 symbols
  - Differential PSK (DPSK)
  - Differential QPSK (DQPSK)
  - Offset QPSK (OQPSK)
  - π/4–QPSK
- Frequency-shift keying (FSK):
  - Audio frequency-shift keying (AFSK)
  - Multi-frequency shift keying (M-ary FSK or MFSK)
  - Dual-tone multi-frequency (DTMF)
  - Continuous-phase frequency-shift keying (CPFSK)
- Amplitude-shift keying (ASK)

- On-off keying (OOK), the most common ASK form
    - M-ary vestigial sideband modulation, for example 8VSB
- Quadrature amplitude modulation (QAM) - a combination of PSK and ASK:
    - Polar modulation like QAM a combination of PSK and ASK.
- Continuous phase modulation (CPM) methods:
    - Minimum-shift keying (MSK)
    - Gaussian minimum-shift keying (GMSK)
- Orthogonal frequency-division multiplexing (OFDM) modulation:
    - discrete multitone (DMT) - including adaptive modulation and bit-loading.
- Wavelet modulation
- Trellis coded modulation (TCM), also known as trellis modulation
- Spread-spectrum techniques:
    - Direct-sequence spread spectrum (DSSS)
    - Chirp spread spectrum (CSS) according to IEEE 802.15.4a CSS uses pseudo-stochastic coding
    - Frequency-hopping spread spectrum (FHSS) applies a special scheme for channel release

MSK and GMSK are particular cases of continuous phase modulation. Indeed, MSK is a particular case of the sub-family of CPM known as continuous-phase frequency-shift keying (CPFSK) which is defined by a rectangular frequency pulse (i.e. a linearly increasing phase pulse) of one symbol-time duration (total response signaling).

OFDM is based on the idea of frequency-division multiplexing (FDM), but is utilized as a digital modulation scheme. The bit stream is split into several parallel data streams, each transferred over its own sub-carrier using some conventional digital modulation scheme. The modulated sub-carriers are summed to form an OFDM signal. OFDM is considered as a modulation technique rather than a multiplex technique, since it transfers one bit stream over one communication channel using one sequence of so-called OFDM symbols. OFDM can be extended to multi-user channel access method in the orthogonal frequency-division multiple access (OFDMA) and multi-carrier code division multiple access (MC-CDMA) schemes, allowing several users to share the same physical medium by giving different sub-carriers or spreading codes to different users.

Of the two kinds of RF power amplifier, switching amplifiers (Class C amplifiers) cost less and use less battery power than linear amplifiers of the same output power. However, they only work with relatively constant-amplitude-modulation signals such as angle modulation (FSK or PSK) and CDMA, but not with QAM and OFDM. Nevertheless, even though switching amplifiers are completely unsuitable for normal QAM constellations, often the QAM modulation principle are used to drive switching amplifiers with these FM and other waveforms, and sometimes QAM demodulators are used to receive the signals put out by these switching amplifiers.

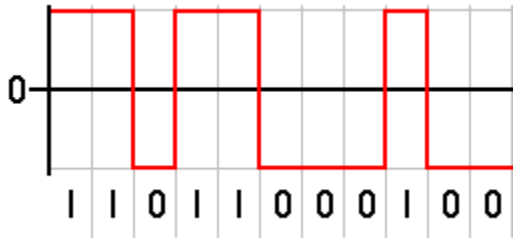
Digital baseband modulation or line coding

The term **digital baseband modulation** (or digital baseband transmission) is synonymous to line codes. These are methods to transfer a digital bit stream over an analog baseband channel (a.k.a. lowpass channel) using a pulse train, i.e. a discrete number of signal levels, by directly modulating the voltage or current on a cable. Common examples are unipolar, non-return-to-zero (NRZ), Manchester and alternate mark inversion (AMI) codings.

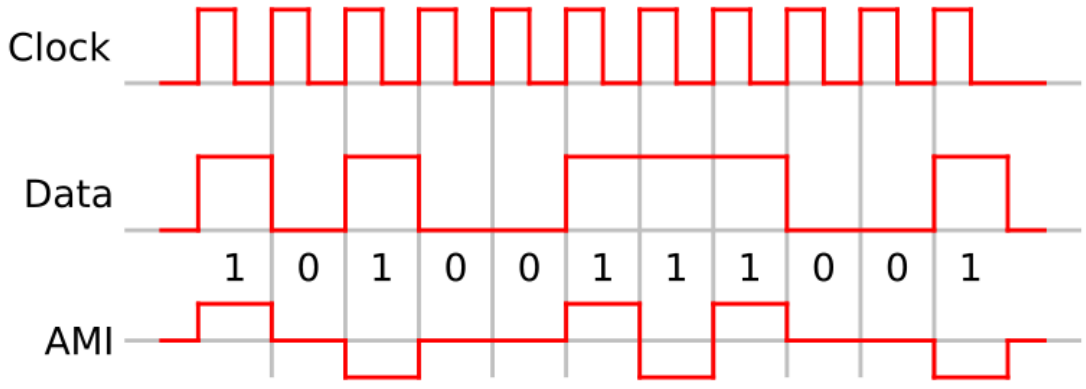
Chapter 2

Line Code & Baseband

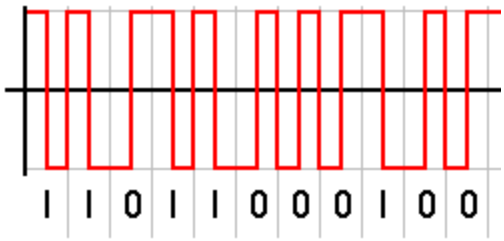
Line Code



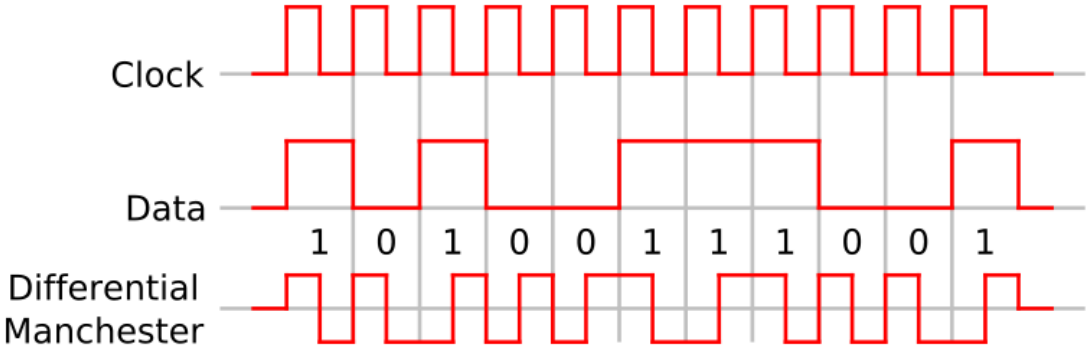
An example of coding a binary signal using rectangular pulse amplitude modulation with polar non-return-to-zero code



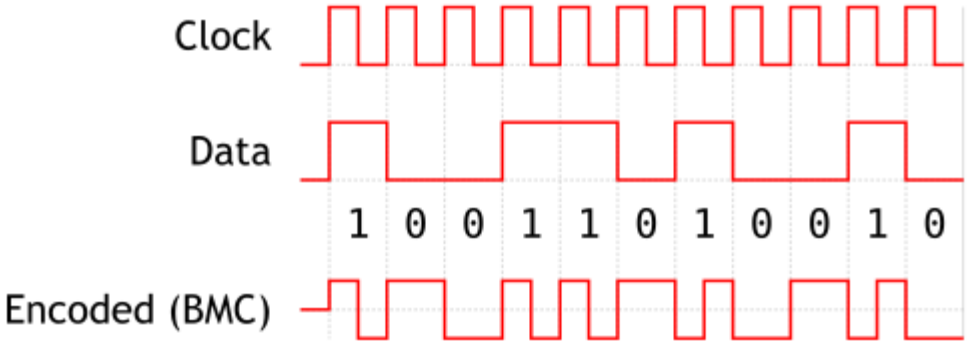
An example of Bipolar encoding, or AMI.



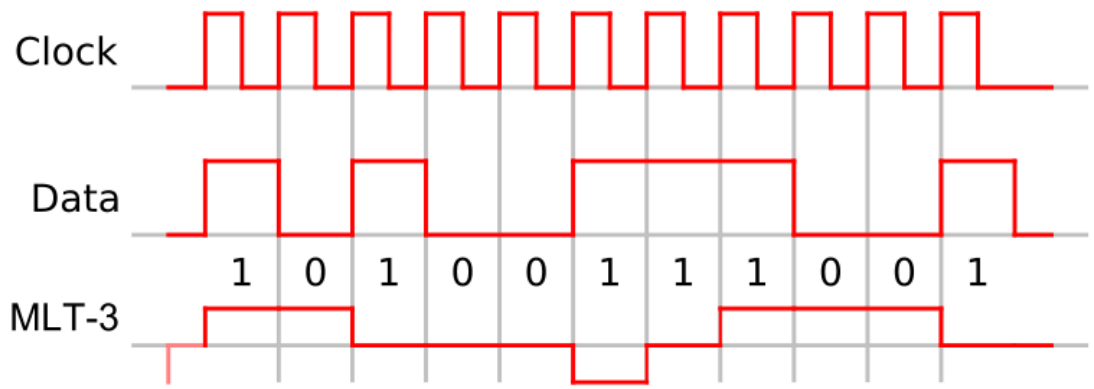
Encoding of 11011000100 in Manchester encoding



An example of Differential Manchester encoding



An example of Biphase mark code



An example of MLT-3 encoding.

In telecommunication, a **line code** (also called **digital baseband transmission** method) is a code chosen for use within a communications system for baseband transmission purposes. Line coding is often used for digital data transport.

Line coding

Line coding consists of representing the digital signal to be transported by an amplitude- and time-discrete signal that is optimally tuned for the specific properties of the physical channel (and of the receiving equipment). The waveform pattern of voltage or current used to represent the 1s and 0s of a digital data on a transmission link is called *line encoding*. The common types of line encoding are unipolar, polar, bipolar and Manchester encoding.

For reliable clock recovery at the receiver, one usually imposes a maximum run length constraint on the generated channel sequence, i.e. the maximum number of consecutive ones or zeros is bounded to a reasonable number. A clock period is recovered by observing transitions in the received sequence, so that a maximum run length guarantees such clock recovery, while sequences without such a constraint could seriously hamper the detection quality.

After line coding, the signal is put through a "physical channel", either a "transmission medium" or "data storage medium". Sometimes the characteristics of two very different-seeming channels are similar enough that the same line code is used for them. The most common physical channels are:

- the line-coded signal can directly be put on a transmission line, in the form of variations of the voltage or current (often using differential signaling).
- the line-coded signal (the "base-band signal") undergoes further pulse shaping (to reduce its frequency bandwidth) and then modulated (to shift its frequency bandwidth) to create the "RF signal" that can be sent through free space.

- the line-coded signal can be used to turn on and off a light in Free Space Optics, most commonly infrared remote control.
- the line-coded signal can be printed on paper to create a bar code.
- the line-coded signal can be converted to a magnetized spots on a hard drive or tape drive.
- the line-coded signal can be converted to a pits on optical disc.

Unfortunately, most long-distance communication channels cannot transport a DC component. The DC component is also called the disparity, the bias, or the DC coefficient. The simplest possible line code, called unipolar because it has an unbounded DC component, gives too many errors on such systems.

Most line codes eliminate the DC component — such codes are called DC balanced, zero-DC, zero-bias or DC equalized etc. There are two ways of eliminating the DC component:

- Use a constant-weight code. In other words, design each transmitted code word such that every code word that contains some positive or negative levels also contains enough of the opposite levels, such that the average level over each code word is zero. For example, Manchester code and Interleaved 2 of 5.
- Use a paired disparity code. In other words, design the receiver such that every code word that averages to a negative level is paired with another code word that averages to a positive level. Design the receiver so that either code word of the pair decodes to the same data bits. Design the transmitter to keep track of the running DC buildup, and always pick the code word that pushes the DC level back towards zero. For example, AMI, 8B10B, 4B3T, etc.

Line coding should make it possible for the receiver to synchronize itself to the phase of the received signal. If the synchronization is not ideal, then the signal to be decoded will not have optimal differences (in amplitude) between the various digits or symbols used in the line code. This will increase the error probability in the received data.

It is also preferred for the line code to have a structure that will enable error detection.

Note that the line-coded signal and a signal produced at a terminal may differ, thus requiring translation.

A line code will typically reflect technical requirements of the transmission medium, such as optical fiber or shielded twisted pair. These requirements are unique for each medium, because each one has different behavior related to interference, distortion, capacitance and loss of amplitude.

Common line codes

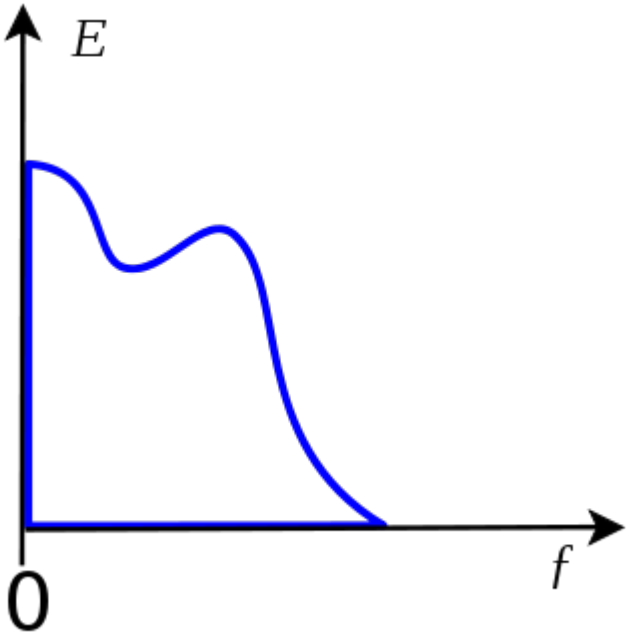
- AMI
- Modified AMI codes: B8ZS, B6ZS, B3ZS, HDB3

- 2B1Q
- 4B5B
- 4B3T
- 6b/8b encoding
- Hamming Code
- 8b/10b encoding
- 64b/66b encoding
- 128b/130b encoding
- Coded mark inversion (CMI)
- Conditioned Diphase
- Eight-to-Fourteen Modulation (EFM) used in Compact Disc
- EFMPlus used in DVD
- RZ — Return-to-zero
- NRZ — Non-return-to-zero
- NRZI — Non-return-to-zero, inverted
- Manchester code (also variants Differential Manchester & Biphase mark code)
- Miller encoding (also known as Delay encoding or Modified Frequency Modulation, and has variant Modified Miller encoding)
- MLT-3 Encoding
- Hybrid Ternary Codes
- Surround by complement (SBC)
- TC-PAM

Optical line codes:

- Carrier-Suppressed Return-to-Zero
- Alternate-Phase Return-to-Zero

Baseband



Spectrum of a **baseband signal**, energy as a function of frequency

In telecommunications and signal processing, ***baseband*** is an adjective that describes signals and systems whose range of frequencies is measured from close to 0 hertz to a cut-off frequency, a maximum bandwidth or highest signal frequency; it is sometimes used as a noun for a band of frequencies starting close to zero. Baseband can often be considered as a synonym to **lowpass** or **non-modulated**, and antonym to passband, bandpass, carrier-modulated or radio frequency (RF) signal.

Various uses

Baseband bandwidth

A *baseband bandwidth* is equal to the highest frequency of a signal or system, or an upper bound on such frequencies, for example the upper cut-off frequency of a passband filter. By contrast, passband bandwidth is the difference between a highest frequency and a nonzero lowest frequency.

Baseband channel

A *baseband channel* or *lowpass channel* (or *system*, or *network*) is a communication channel that can transfer frequencies that are very near zero. Examples are serial cables and local area networks (LANs), as opposed to passband channels such as radio

frequency channels and passband filtered wires of the analog telephone network. Frequency division multiplexing (FDM) allows an analog telephone wire to carry a baseband telephone call, concurrently as one or several carrier-modulated telephone calls.

Digital baseband transmission

Digital baseband transmission, also known as line coding, aims at transferring a digital bit stream over base-band channel, typically an unfiltered wire, as opposed to passband transmission, also known as *carrier-modulated* transmission. Passband transmission makes communication possible over a bandpass filtered channel, such as the telephone network local-loop or a band-limited wireless channel.

An unfiltered wire is intrinsically a low-pass transmission channel, while a line code is intrinsically a pulse wave signal that occupies a frequency spectrum of infinite bandwidth. According to the Nyquist theorem, error-free detection of the line code requires a channel bandwidth of at least the Nyquist rate, which is half the line code pulse rate.

Baseband transmission in Ethernet

The word "BASE" in Ethernet physical layer standards, for example 10BASE5, 100BASE-T and 1000BASE-SX, implies baseband digital transmission, i.e. that a line code and an unfiltered wire are used.

This is as opposed to 10PASS-TS Ethernet, where "PASS" implies passband transmission. Passband digital transmission requires a digital modulation scheme, often provided by modem equipment. In the 10PASS-TS case the VDSL standard is utilized, which is based on the Discrete multi-tone modulation (DMT) scheme. Other examples of passband network access technologies are wireless networks and cable modems.

Baseband signal

A *baseband signal* or *lowpass signal* is a signal that can include frequencies that are very near zero, by comparison with its highest frequency (for example, a sound waveform can be considered as a baseband signal, whereas a radio signal or any other modulated signal is not).

A signal **"at baseband"** is usually considered to include frequencies from near 0 Hz up to the highest frequency in the signal with significant power.

In general, signals can be described as including a whole range of different frequencies added together. In telecommunications in particular, it is often the case that those parts of the signal which are at low frequencies are 'copied' up to higher frequencies for transmission purposes, since there are few communications media that will pass low frequencies without distortion. Then, the original, low frequency components are referred to as the **baseband** signal. Typically, the new, high-frequency copy is referred to as the

'RF' (radio-frequency) signal. A baseband signal is a low frequency signal which when modulated is transmitted on various channels.

The concept of baseband signals is most often applied to real-valued signals, and systems that handle real-valued signals. Fourier analysis of such signals includes a negative-frequency band, but the negative-frequency information is just a mirror of the positive-frequency information, not new information. For complex-valued signals, on the other hand, the negative frequencies carry new information. In that case, the full two-sided bandwidth is generally quoted, rather than just the half measured from zero; the concept of baseband can be applied by treating the real and imaginary parts of the complex-valued signal as two different real signals.

Equivalent baseband signal

An *equivalent baseband signal* or *equivalent lowpass signal* is – in analog and digital modulation methods with constant carrier frequency (for example ASK, PSK and QAM, but not FSK) – a complex valued representation of the modulated physical signal (the so called passband signal or RF signal). The equivalent baseband signal is $Z(t) = I(t) + jQ(t)$ where $I(t)$ is the inphase signal, $Q(t)$ the quadrature phase signal, and j the imaginary unit. In a digital modulation method, the $I(t)$ and $Q(t)$ signals of each modulation symbol are evident from the constellation diagram. The frequency spectrum of this signal includes negative as well as positive frequencies. The physical passband signal corresponds to

$$I(t)\cos(\omega t) - Q(t)\sin(\omega t) = \mathrm{Re}\{Z(t)e^{j\omega t}\}$$

where ω is the carrier angular frequency in rad/s.

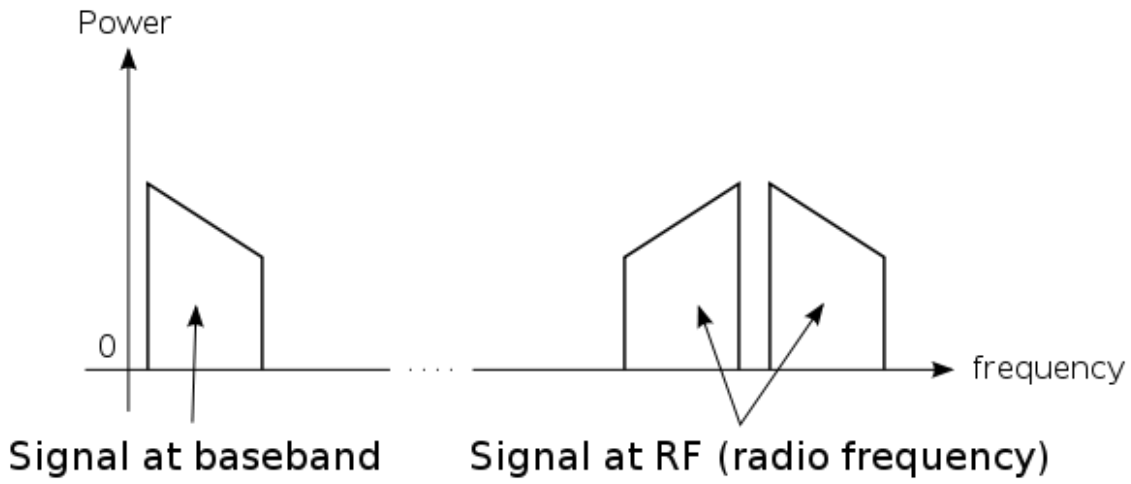
In an *equivalent baseband model* of a communication system, the modulated signal is replaced by a complex valued equivalent baseband signal with carrier frequency of 0 hertz, and the RF channel is replaced by an equivalent baseband channel model where the frequency response is transferred to baseband frequencies.

Modulation

A signal **at baseband** is often used to modulate a higher frequency carrier wave in order that it may be transmitted via radio. Modulation results in shifting the signal up to much higher frequencies (radio frequencies, or RF) than it originally spanned. A key consequence of the usual double-sideband amplitude modulation (AM) is that, usually, the range of frequencies the signal spans (its spectral bandwidth) is doubled. Thus, the RF bandwidth of a signal (measured from the lowest frequency as opposed to 0 Hz) is usually twice its baseband bandwidth. Steps may be taken to reduce this effect, such as single-sideband modulation; the highest frequency of such signals greatly exceeds the baseband bandwidth.

Some signals can be treated as baseband or not, depending on the situation. For example, a switched analog connection in the telephone network has energy below 300 Hz and above 3400 Hz removed by bandpass filtering; since the signal has no energy very close to zero frequency, it may not be considered a baseband signal, but in the telephone systems frequency-division multiplexing hierarchy, it is usually treated as a baseband signal, by comparison with the modulated signals used for long-distance transmission. The 300 Hz lower band edge in this case is treated as "near zero", being a small fraction of the upper band edge.

The figure shows what happens with AM modulation:



Comparison of the equivalent baseband version of a signal and its AM-modulated (double-sideband) RF version, showing the typical doubling of the occupied bandwidth.

The simplest definition is that a signal's baseband bandwidth is its bandwidth before modulation and multiplexing, or after demultiplexing and demodulation.

The composite video signal created by devices such as most newer VCRs, game consoles and DVD players is a commonly used baseband signal.

Chapter 3

Flow Control

In data communications, **flow control** is the process of managing the pacing of data transmission between two nodes to prevent a fast sender from outrunning a slow receiver. It provides a mechanism for the receiver to control the transmission speed, so that the receiving node is not overwhelmed with data from transmitting node. Flow control should be distinguished from congestion control, which is used for controlling the flow of data when congestion has actually occurred . Flow control mechanisms can be classified by whether or not the receiving node sends feedback to the sending node.

Flow control is important because it is possible for a sending computer to transmit information at a faster rate than the destination computer can receive and process them. This can happen if the receiving computers have a heavy traffic load in comparison to the sending computer, or if the receiving computer has less processing power than the sending computer.

Types of flow control

Network congestion
: A prevention mechanism that provides control over the quantity of data transmission that enters a device.

Windowing Flow control
: mechanism used with TCP.

data buffer
: A prevention control mechanism that provides storage to contain data- bursts from other network devices, compensating for the variation of data transmission speeds.

Transmit flow control

Transmit flow control may occur:

- between data "on and off" terminal equipment (DTE) and a switching center, via data circuit-terminating equipment (DCE), the opposite types interconnected straightforwardly,
- or between two devices of the same type (two DTEs, or two DCEs), interconnected by a crossover cable.

The transmission rate may be controlled because of network or DTE requirements. Transmit flow control can occur independently in the two directions of data transfer, thus permitting the transfer rates in one direction to be different from the transfer rates in the other direction. Transmit flow control can be

- either stop-and-go,
- or use a sliding window.

Flow control can be performed

- either by control signal lines in a data communication interface,
- or by reserving in-band control characters to signal flow start and stop (such as the ASCII codes for XON/XOFF).

Hardware flow control

In common RS 232 there are pairs of control lines:

- **RTS flow control**, RTS (Request To Send)/CTS (Clear To Send) and
- **DTR flow control**, DTR (Data Terminal Ready)/DSR (Data Set Ready),

which are usually referred to as **hardware flow control**.

Hardware flow control is typically handled by the DTE or "master end", as it is first raising or asserting its line to command the other side:

- In case of **RTS control flow**, DTE sets its RTS, which signals the opposite end (the slave end such as a DCE) to begin monitoring its data input line. When ready for data, the slave end will raise its complementary line, CTS in this example, which signals the master to start sending data, and for the master to begin monitoring the slave's data output line. If either end needs to stop the data, it lowers its respective "data readyness" line.
- For PC-to-modem and similar links, the case of **DTR flow control**, DTR/DSR are raised for the entire modem session (say a dialup internet call), and RTS/CTS are raised for each block of data.

Software flow control

Oppositely, XON/XOFF is usually referred to as **software flow control**.

Open-loop flow control

The open-loop flow control mechanism is characterized by having no feedback between the receiver and the transmitter. This simple means of control is widely used. The allocation of resources must be a “prior reservation” or “hop-to-hop” type.

The Open-loop flow control has inherent problems with maximizing the utilization of network resources. Resource allocation is made at connection setup using a CAC (Connection Admission Control) and this allocation is made using information that is already “old news” during the lifetime of the connection. Often there is an over-allocation of resources and reserved but unused capacities are wasted. Open-Loop flow control is used by ATM in its CBR, VBR and UBR services.

Closed-loop flow control

The Closed Loop flow control mechanism is characterized by the ability of the network to report pending network congestion back to the transmitter. This information is then used by the transmitter in various ways to adapt its activity to existing network conditions. Closed Loop flow control is used by ABR. Transmit Flow Control described above is a form of Closed-loop flow control.

Chapter 4

Error Detection and Correction

In information theory and coding theory with applications in computer science and telecommunication, **error detection and correction** or **error control** are techniques that enable reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data.

The general definitions of the terms are as follows:

- *Error detection* is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.
- *Error correction* is the detection of errors and reconstruction of the original, error-free data.

Error correction may generally be realized in two different ways:

- *Automatic repeat request (ARQ)* (sometimes also referred to as *backward error correction*): This is an error control technique whereby an error detection scheme is combined with requests for retransmission of erroneous data. Every block of data received is checked using the error detection code used, and if the check fails, retransmission of the data is requested – this may be done repeatedly, until the data can be verified.
- *Forward error correction (FEC)*: The sender encodes the data using an *error-correcting code (ECC)* prior to transmission. The additional information (redundancy) added by the code is used by the receiver to recover the original data. In general, the reconstructed data is what is deemed the "most likely" original data.

ARQ and FEC may be combined, such that minor errors are corrected without retransmission, and major errors are corrected via a request for retransmission: this is called *hybrid automatic repeat-request (HARQ)*.

Introduction

The general idea for achieving error detection and correction is to add some redundancy (i.e., some extra data) to a message, which receivers can use to check consistency of the delivered message, and to recover data determined to be erroneous. Error-detection and correction schemes can be either systematic or non-systematic: In a systematic scheme, the transmitter sends the original data, and attaches a fixed number of *check bits* (or *parity data*), which are derived from the data bits by some deterministic algorithm. If only error detection is required, a receiver can simply apply the same algorithm to the received data bits and compare its output with the received check bits; if the values do not match, an error has occurred at some point during the transmission. In a system that uses a non-systematic code, the original message is transformed into an encoded message that has at least as many bits as the original message.

Good error control performance requires the scheme to be selected based on the characteristics of the communication channel. Common channel models include memory-less models where errors occur randomly and with a certain probability, and dynamic models where errors occur primarily in bursts. Consequently, error-detecting and correcting codes can be generally distinguished between *random-error-detecting/correcting* and *burst-error-detecting/correcting*. Some codes can also be suitable for a mixture of random errors and burst errors.

If the channel capacity cannot be determined, or is highly varying, an error-detection scheme may be combined with a system for retransmissions of erroneous data. This is known as automatic repeat request (ARQ), and is most notably used in the Internet. An alternate approach for error control is hybrid automatic repeat request (HARQ), which is a combination of ARQ and error-correction coding.

Error detection schemes

Error detection is most commonly realized using a suitable hash function (or checksum algorithm). A hash function adds a fixed-length *tag* to a message, which enables receivers to verify the delivered message by recomputing the tag and comparing it with the one provided.

There exists a vast variety of different hash function designs. However, some are of particularly widespread use because of either their simplicity or their suitability for detecting certain kinds of errors (e.g., the cyclic redundancy check's performance in detecting burst errors).

Random-error-correcting codes based on minimum distance coding can provide a suitable alternative to hash functions when a strict guarantee on the minimum number of errors to be detected is desired. Repetition codes, described below, are special cases of error-correcting codes: although rather inefficient, they find applications for both error correction and detection due to their simplicity.

Repetition codes

A **repetition code** is a coding scheme that repeats the bits across a channel to achieve error-free communication. Given a stream of data to be transmitted, the data is divided into blocks of bits. Each block is transmitted some predetermined number of times. For example, to send the bit pattern "1011", the four-bit block can be repeated three times, thus producing "1011 1011 1011". However, if this twelve-bit pattern was received as "1010 1011 1011" – where the first block is unlike the other two – it can be determined that an error has occurred.

Repetition codes are not very efficient, and can be susceptible to problems if the error occurs in exactly the same place for each group (e.g., "1010 1010 1010" in the previous example would be detected as correct). The advantage of repetition codes is that they are extremely simple, and are in fact used in some transmissions of numbers stations.

Parity bits

A **parity bit** is a bit that is added to a group of source bits to ensure that the number of set bits (i.e., bits with value 1) in the outcome is even or odd. It is a very simple scheme that can be used to detect single or any other odd number (i.e., three, five, etc.) of errors in the output. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous.

Extensions and variations on the parity bit mechanism are horizontal redundancy checks, vertical redundancy checks, and "double," "dual," or "diagonal" parity (used in RAID-DP).

Checksums

A **checksum** of a message is a modular arithmetic sum of message code words of a fixed word length (e.g., byte values). The sum may be negated by means of a one's-complement prior to transmission to detect errors resulting in all-zero messages.

Checksum schemes include parity bits, check digits, and longitudinal redundancy checks. Some checksum schemes, such as the Luhn algorithm and the Verhoeff algorithm, are specifically designed to detect errors commonly introduced by humans in writing down or remembering identification numbers.

Cyclic redundancy checks (CRCs)

A **cyclic redundancy check (CRC)** is a single-burst-error-detecting cyclic code and non-secure hash function designed to detect accidental changes to digital data in computer networks. It is characterized by specification of a so-called *generator polynomial*, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend, and where the remainder becomes the result.

Cyclic codes have favorable properties in that they are well suited for detecting burst errors. CRCs are particularly easy to implement in hardware, and are therefore commonly used in digital networks and storage devices such as hard disk drives.

Even parity is a special case of a cyclic redundancy check, where the single-bit CRC is generated by the divisor $x+1$.

Cryptographic hash functions

A **cryptographic hash function** can provide strong assurances about data integrity, provided that changes of the data are only accidental (i.e., due to transmission errors). Any modification to the data will likely be detected through a mismatching hash value. Furthermore, given some hash value, it is infeasible to find some input data (other than the one given) that will yield the same hash value. Message authentication codes, also called *keyed* cryptographic hash functions, provide additional protection against intentional modification by an attacker.

Error-correcting codes

Any error-correcting code can be used for error detection. A code with *minimum Hamming distance*, d, can detect up to $d-1$ errors in a code word. Using minimum-distance-based error-correcting codes for error detection can be suitable if a strict limit on the minimum number of errors to be detected is desired.

Codes with minimum Hamming distance $d=2$ are degenerate cases of error-correcting codes, and can be used to detect single errors. The parity bit is an example of a single-error-detecting code.

The Berger code is an early example of a unidirectional error(-correcting) code that can detect any number of errors on an asymmetric channel, provided that only transitions of cleared bits to set bits *or* set bits to cleared bits can occur.

Error correction

Automatic repeat request

Automatic Repeat reQuest (ARQ) is an error control method for data transmission that makes use of error-detection codes, acknowledgment and/or negative acknowledgment messages, and timeouts to achieve reliable data transmission. An *acknowledgment* is a message sent by the receiver to indicate that it has correctly received a data frame.

Usually, when the transmitter does not receive the acknowledgment before the timeout occurs (i.e., within a reasonable amount of time after sending the data frame), it retransmits the frame until it is either correctly received or the error persists beyond a predetermined number of retransmissions.

Three types of ARQ protocols are Stop-and-wait ARQ, Go-Back-N ARQ, and Selective Repeat ARQ.

ARQ is appropriate if the communication channel has varying or unknown capacity, such as is the case on the Internet. However, ARQ requires the availability of a back channel, results in possibly increased latency due to retransmissions, and requires the maintenance of buffers and timers for retransmissions, which in the case of network congestion can put a strain on the server and overall network capacity.

Error-correcting code

An error-correcting code (ECC) or forward error correction (FEC) code is a system of adding redundant data, or *parity data*, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a back-channel is not required in forward error correction, and it is therefore suitable for simplex communication such as broadcasting. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks, and RAM.

Error-correcting codes are usually distinguished between convolutional codes and block codes:

- *Convolutional codes* are processed on a bit-by-bit basis. They are particularly suitable for implementation in hardware, and the Viterbi decoder allows optimal decoding.

- *Block codes* are processed on a block-by-block basis. Early examples of block codes are repetition codes, Hamming codes and multidimensional parity-check codes. They were followed by a number of efficient codes, Reed-Solomon codes being the most notable due to their current widespread use. Turbo codes and low-

density parity-check codes (LDPC) are relatively new constructions that can provide almost optimal efficiency.

Shannon's theorem is an important theorem in forward error correction, and describes the maximum information rate at which reliable communication is possible over a channel that has a certain error probability or signal-to-noise ratio (SNR). This strict upper limit is expressed in terms of the channel capacity. More specifically, the theorem says that there exist codes such that with increasing encoding length the probability of error on a discrete memoryless channel can be made arbitrarily small, provided that the code rate is smaller than the channel capacity. The code rate is defined as the fraction *k/n* of *k* source symbols and *n* encoded symbols.

The actual maximum code rate allowed depends on the error-correcting code used, and may be lower. This is because Shannon's proof was only of existential nature, and did not show how to construct codes which are both optimal and have efficient encoding and decoding algorithms.

Hybrid schemes

Hybrid ARQ is a combination of ARQ and forward error correction. There are two basic approaches:

- Messages are always transmitted with FEC parity data (and error-detection redundancy). A receiver decodes a message using the parity information, and requests retransmission using ARQ only if the parity data was not sufficient for successful decoding (identified through a failed integrity check).
- Messages are transmitted without parity data (only with error-detection information). If a receiver detects an error, it requests FEC information from the transmitter using ARQ, and uses it to reconstruct the original message.

The latter approach is particularly attractive on an erasure channel when using a rateless erasure code.

Applications

Applications that require low latency (such as telephone conversations) cannot use Automatic Repeat reQuest (ARQ); they must use Forward Error Correction (FEC). By the time an ARQ system discovers an error and re-transmits it, the re-sent data will arrive too late to be any good.

Applications where the transmitter immediately forgets the information as soon as it is sent (such as most television cameras) cannot use ARQ; they must use FEC because when an error occurs, the original data is no longer available. (This is also why FEC is used in data storage systems such as RAID and distributed data store).

Applications that use ARQ must have a return channel. Applications that have no return channel cannot use ARQ.

Applications that require extremely low error rates (such as digital money transfers) must use ARQ.

The Internet

In a typical TCP/IP stack, error control is performed at multiple levels:

- Each Ethernet frame carries a CRC-32 checksum. Frames received with incorrect checksums are discarded by the receiver hardware.
- The IPv4 header contains a checksum protecting the contents of the header. Packets with mismatching checksums are dropped within the network or at the receiver.
- The checksum was omitted from the IPv6 header in order to minimize processing costs in network routing and because current link layer technology is assumed to provide sufficient error detection.
- UDP has an optional checksum covering the payload and addressing information from the UDP and IP headers. Packets with incorrect checksums are discarded by the operating system network stack. The checksum is optional under IPv4, only, because the IP layer checksum may already provide the desired level of error protection.
- TCP provides a checksum for protecting the payload and addressing information from the TCP and IP headers. Packets with incorrect checksums are discarded within the network stack, and eventually get retransmitted using ARQ, either explicitly (such as through triple-ack) or implicitly due to a timeout.

Deep-space telecommunications

Development of error-correction codes was tightly coupled with the history of deep-space missions due to the extreme dilution of signal power over interplanetary distances, and the limited power availability aboard space probes. Whereas early missions sent their data uncoded, starting from 1968 digital error correction was implemented in the form of (sub-optimally decoded) convolutional codes and Reed-Muller codes. The Reed-Muller code was well suited to the noise the spacecraft was subject to (approximately matching a bell curve), and was implemented at the Mariner spacecraft for missions between 1969 and 1977.

The Voyager 1 and Voyager 2 missions, which started in 1977, were designed to deliver color imaging amongst scientific information of Jupiter and Saturn. This resulted in increased coding requirements, and thus the spacecraft were supported by (optimally Viterbi-decoded) convolutional codes that could be concatenated with an outer Golay (24,12,8) code. The Voyager 2 probe additionally supported an implementation of a Reed-Solomon code: the concatenated Reed-Solomon-Viterbi (RSV) code allowed for

very powerful error correction, and enabled the spacecraft's extended journey to Uranus and Neptune.

The CCSDS currently recommends usage of error correction codes with performance similar to the Voyager 2 RSV code as a minimum. Concatenated codes are increasingly falling out of favor with space missions, and are replaced by more powerful codes such as Turbo codes or LDPC codes.

The different kinds of deep space and orbital missions that are conducted suggest that trying to find a "one size fits all" error correction system will be an ongoing problem for some time to come. For missions close to earth the nature of the channel noise is different from that of a spacecraft on an interplanetary mission experiences. Additionally, as a spacecraft increases its distance from earth, the problem of correcting for noise gets larger.

Satellite broadcasting (DVB)

The demand for satellite transponder bandwidth continues to grow, fueled by the desire to deliver television (including new channels and High Definition TV) and IP data. Transponder availability and bandwidth constraints have limited this growth, because transponder capacity is determined by the selected modulation scheme and Forward error correction (FEC) rate.

Overview

- QPSK coupled with traditional Reed Solomon and Viterbi codes have been used for nearly 20 years for the delivery of digital satellite TV.
- Higher order modulation schemes such as 8PSK, 16QAM and 32QAM have enabled the satellite industry to increase transponder efficiency by several orders of magnitude.
- This increase in the information rate in a transponder comes at the expense of an increase in the carrier power to meet the threshold requirement for existing antennas.
- Tests conducted using the latest chipsets demonstrate that the performance achieved by using Turbo Codes may be even lower than the 0.8 dB figure assumed in early designs.

Data storage

Error detection and correction codes are often used to improve the reliability of data storage media.

A "parity track" was present on the first magnetic tape data storage in 1951. The "Optimal Rectangular Code" used in group code recording tapes not only detects but also corrects single-bit errors.

Some file formats, particularly archive formats, include a checksum (most often CRC32) to detect corruption and truncation and can employ redundancy and/or parity files to recover portions of corrupted data.

Reed Solomon codes are used in compact discs to correct errors caused by scratches.

Modern hard drives use CRC codes to detect and Reed-Solomon codes to correct minor errors in sector reads, and to recover data from sectors that have "gone bad" and store that data in the spare sectors.

RAID systems use a variety of error correction techniques, to correct errors when a hard drive completely fails.

Error-correcting memory

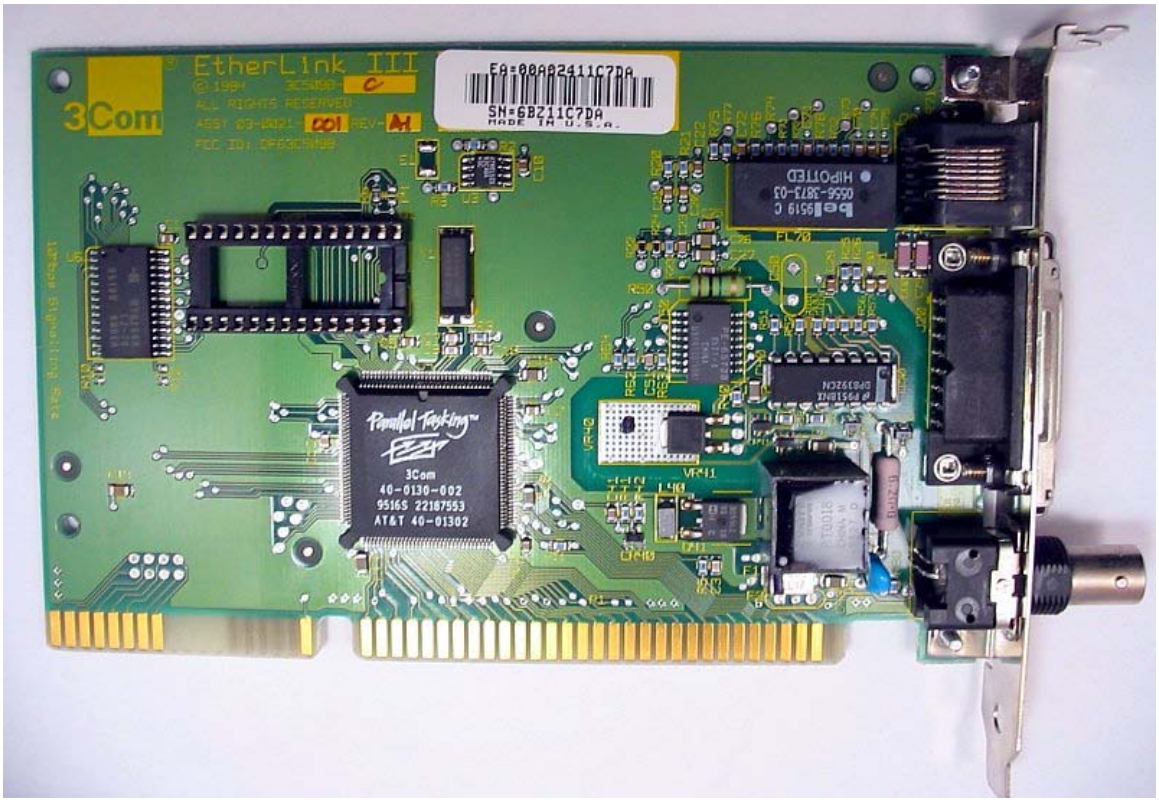
DRAM memory may provide increased protection against soft errors by relying on error correcting codes. Such error-correcting memory, known as *ECC* or *EDAC-protected* memory, is particularly desirable for high fault-tolerant applications, such as servers, as well as deep-space applications due to increased radiation.

Error-correcting memory controllers traditionally use Hamming codes, although some use triple modular redundancy.

Interleaving allows distributing the effect of a single cosmic ray potentially upsetting multiple physically neighboring bits across multiple words by associating neighboring bits to different words. As long as a single event upset (SEU) does not exceed the error threshold (e.g., a single error) in any particular word between accesses, it can be corrected (e.g., by a single-bit error correcting code), and the illusion of an error-free memory system may be maintained.

Chapter 5

Computer Networking



Network cards such as this one can transmit and receive data at high rates over various types of network cables. This card is a 'Combo' card which supports three cabling standards.

Computer networking or **Data communications** (**Datacom**) is the engineering discipline concerned with the communication between computer systems or devices. A computer network is any set of computers or devices connected to each other with the ability to exchange data. Computer networking is sometimes considered a sub-discipline of telecommunications, computer science, information technology and/or computer

engineering since it relies heavily upon the theoretical and practical application of these scientific and engineering disciplines. The three types of networks are: the Internet, the intranet, and the extranet. Examples of different network methods are:

- Local area network (LAN), which is usually a small network constrained to a small geographic area. An example of a LAN would be a computer network within a building.
- Metropolitan area network (MAN), which is used for medium size area. examples for a city or a state.
- Wide area network (WAN) that is usually a larger network that covers a large geographic area.
- Wireless LANs and WANs (WLAN & WWAN) are the wireless equivalent of the LAN and WAN.

All networks are interconnected to allow communication with a variety of different kinds of media, including twisted-pair copper wire cable, coaxial cable, optical fiber, power lines and various wireless technologies. The devices can be separated by a few meters (e.g. via Bluetooth) or nearly unlimited distances (e.g. via the interconnections of the Internet). Networking, routers, routing protocols, and networking over the public Internet have their specifications defined in documents called RFCs.

Views of networks

Users and network administrators typically have different views of their networks. Users can share printers and some servers from a workgroup, which usually means they are in the same geographic location and are on the same LAN, whereas a Network Administrator is responsible to keep that network up and running. A community of interest has less of a connection of being in a local area, and should be thought of as a set of arbitrarily located users who share a set of servers , and possibly also communicate via peer-to-peer technologies.

Network administrators can see networks from both physical and logical perspectives. The physical perspective involves geographic locations, physical cabling, and the network elements (e.g., routers, bridges and application layer gateways that interconnect the physical media. Logical networks, called, in the TCP/IP architecture, subnets, map onto one or more physical media. For example, a common practice in a campus of buildings is to make a set of LAN cables in each building appear to be a common subnet, using virtual LAN (VLAN) technology.

Both users and administrators will be aware, to varying extents, of the trust and scope characteristics of a network. Again using TCP/IP architectural terminology, an intranet is a community of interest under private administration usually by an enterprise, and is only accessible by authorized users (e.g. employees). Intranets do not have to be connected to the Internet, but generally have a limited connection. An extranet is an extension of an intranet that allows secure communications to users outside of the intranet (e.g. business partners, customers).

Unofficially, the Internet is the set of users, enterprises, and content providers that are interconnected by Internet Service Providers (ISP). From an engineering viewpoint, the Internet is the set of subnets, and aggregates of subnets, which share the registered IP address space and exchange information about the reachability of those IP addresses using the Border Gateway Protocol. Typically, the human-readable names of servers are translated to IP addresses, transparently to users, via the directory function of the Domain Name System (DNS).

Over the Internet, there can be business-to-business (B2B), business-to-consumer (B2C) and consumer-to-consumer (C2C) communications. Especially when money or sensitive information is exchanged, the communications are apt to be **secured** by some form of communications security mechanism. Intranets and extranets can be securely superimposed onto the Internet, without any access by general Internet users, using secure Virtual Private Network (VPN) technology.

History of computer networks

Before the advent of computer networks that were based upon some type of telecommunications system, communication between calculation machines and early computers was performed by human users by carrying instructions between them. Many of the social behaviors seen in today's Internet were demonstrably present in the nineteenth century and arguably in even earlier networks using visual signals.

- In September 1940 George Stibitz used a teletype machine to send instructions for a problem set from his Model at Dartmouth College to his Complex Number Calculator in New York and received results back by the same means. Linking output systems like teletypes to computers was an interest at the Advanced Research Projects Agency (ARPA) when, in 1962, J.C.R. Licklider was hired and developed a working group he called the "Intergalactic Network", a precursor to the ARPANET.
- In 1964, researchers at Dartmouth developed the Dartmouth Time Sharing System for distributed users of large computer systems. The same year, at Massachusetts Institute of Technology, a research group supported by General Electric and Bell Labs used a computer to route and manage telephone connections.
- Throughout the 1960s Leonard Kleinrock, Paul Baran and Donald Davies independently conceptualized and developed network systems which used packets that could be used in a network between computer systems.
- 1965 Thomas Merrill and Lawrence G. Roberts created the first wide area network (WAN).
- The first widely used telephone switch that used true computer control was introduced by Western Electric in 1965.
- In 1969 the University of California at Los Angeles, the Stanford Research Institute, University of California at Santa Barbara, and the University of Utah were connected as the beginning of the ARPANET network using 50 kbit/s circuits.

- Commercial services using X.25 were deployed in 1972, and later used as an underlying infrastructure for expanding TCP/IP networks.

Today, computer networks are the core of modern communication. All modern aspects of the Public Switched Telephone Network (PSTN) are computer-controlled, and telephony increasingly runs over the Internet Protocol, although not necessarily the public Internet. The scope of communication has increased significantly in the past decade, and this boom in communications would not have been possible without the progressively advancing computer network. Computer networks, and the technologies needed to connect and communicate through and between them, continue to drive computer hardware, software, and peripherals industries. This expansion is mirrored by growth in the numbers and types of users of networks from the researcher to the home user.

Networking methods

One way to categorize computer networks is by their geographic scope, although many real-world networks interconnect Local Area Networks (LAN) via Wide Area Networks (WAN) and wireless wide area networks (WWAN). These three (broad) types are:

Local area network (LAN)

A local area network is a network that spans a relatively small space and provides services to a small number of people.

A peer-to-peer or client-server method of networking may be used. A peer-to-peer network is where each client shares their resources with other workstations in the network. Examples of peer-to-peer networks are: Small office networks where resource use is minimal and a home network. A client-server network is where every client is connected to the server and each other. Client-server networks use servers in different capacities. These can be classified into two types:

1. Single-service servers
2. Print servers

The server performs one task such as file server, while other servers can not only perform in the capacity of file servers and print servers, but also can conduct calculations and use them to provide information to clients (Web/Intranet Server). Computers may be connected in many different ways, including Ethernet cables, Wireless networks, or other types of wires such as power lines or phone lines.

The ITU-T G.hn standard is an example of a technology that provides high-speed (up to 1 Gbit/s) local area networking over existing home wiring (power lines, phone lines and coaxial cables).

Wide area network (WAN)

A wide area network is a network where a wide variety of resources are deployed across a large domestic area or internationally. An example of this is a multinational business that uses a WAN to interconnect their offices in different countries. The largest and best example of a WAN is the Internet, which is a network composed of many smaller networks. The Internet is considered the largest network in the world. The PSTN (Public Switched Telephone Network) also is an extremely large network that is converging to use Internet technologies, although not necessarily through the public Internet.

A Wide Area Network involves communication through the use of a wide range of different technologies. These technologies include Point-to-Point WANs such as Point-to-Point Protocol (PPP) and High-Level Data Link Control (HDLC), Frame Relay, ATM (Asynchronous Transfer Mode) and Sonet (Synchronous Optical Network). The difference between the WAN technologies is based on the switching capabilities they perform and the speed at which sending and receiving bits of information (data) occur.

Wireless networks (WLAN, WWAN)

A wireless network is basically the same as a LAN or a WAN but there are no wires between hosts and servers. The data is transferred over sets of radio transceivers. These types of networks are beneficial when it is too costly or inconvenient to run the necessary cables.

The most common IEEE 802.11 WLANs cover, depending on antennas, ranges from hundreds of meters to a few kilometers. For larger areas, either communications satellites of various types, cellular radio, or wireless local loop (IEEE 802.16) all have advantages and disadvantages. Depending on the type of mobility needed, the relevant standards may come from the IETF or the ITU.

Network topology

The network topology defines the way in which computers, printers, and other devices are connected, physically and logically. A network topology describes the layout of the wire and devices as well as the paths used by data transmissions.

Network topology has two types:

- Physical
- Logical

Commonly used topologies include:

- Bus
- Star
- Tree (hierarchical)

- Linear
- Ring
- Mesh
    - partially connected
    - fully connected (sometimes known as *fully redundant*)

The network topologies mentioned above are only a general representation of the kinds of topologies used in computer network and are considered basic topologies

Chapter 6

Communications Protocol

A **communications protocol** is a formal description of digital message formats and the rules for exchanging those messages in or between computing systems and in telecommunications. Protocols may include signaling, authentication and error detection and correction capabilities. A protocol describes the syntax, semantics, and synchronization of communication and may be implemented in hardware or software, or both.

Introduction

While there is no generally accepted formal definition of "protocol" in computer science, an informal definition, based on the previous, could be "a description of a set of procedures to be followed when communicating". In computer science the word *algorithm* is a synonym for the word *procedure' so a 'protocol is to communications what an algorithm is to mathematics.*

Communicating systems use well-defined formats for exchanging messages. Each message has an exact meaning intended to provoke a defined response of the receiver. A protocol therefore describes the *syntax*, *semantics*, and *synchronization* of communication. A *programming language* describes the same for computations, so there is a close analogy between protocols and programming languages: *protocols are to communications what programming languages are to computations*. (A less technical reader might appreciate this similar analogy: *protocols are to communications what grammar is to writing*.)

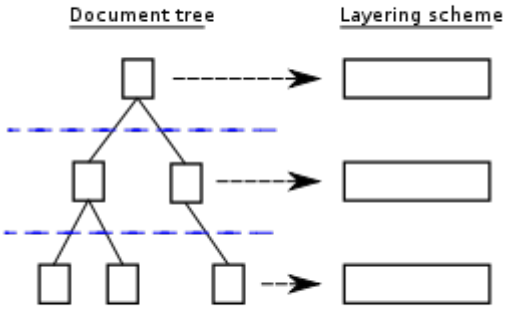


Figure 1. Using a layering scheme to structure a document tree.

Diplomatic documents build on each other, thus creating document-trees. The way the sub-documents making up a document-tree are written has an impact on the complexity of the tree. By imposing a development model on the documents, overall readability can be improved and complexity can be reduced.

An effective model to this end is the *layering scheme* or *model*. In a layering scheme the documents making up the tree are thought to belong to classes, called *layers*. The distance of a sub-document to its root-document is called its level. The level of a sub-document determines the class it belongs to. The sub-documents belonging to a class all provide similar functionality and, when form follows function, have similar form.

The communications protocols in use on the Internet are designed to function in very complex and diverse settings, so they tend to be very complex. Unreliable transmission links add to this by making even basic requirements of protocols harder to achieve.

To ease design, communications protocols are also structured using a layering scheme as a basis. Instead of using a single universal protocol to handle all transmission tasks, a set of cooperating protocols fitting the layering scheme is used.

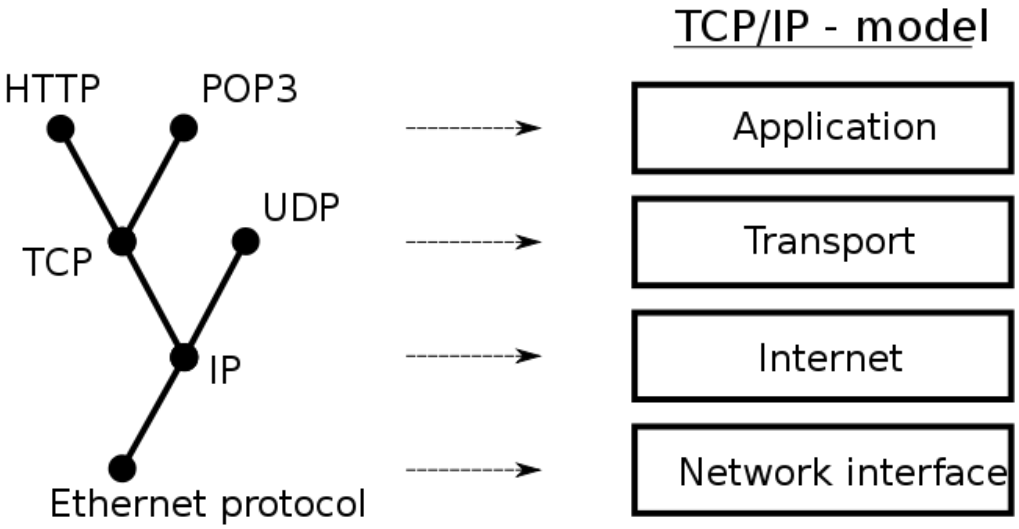


Figure 2. The TCP/IP model or Internet layering scheme and its relation to some common protocols.

The layering scheme in use on the Internet is called the TCP/IP model. The actual protocols are collectively called the Internet protocol suite. The group responsible for this design is called the Internet Engineering Task Force (*IETF*).

Obviously the number of layers of a layering scheme and the way the layers are defined can have a drastic impact on the protocols involved. This is where the analogies come into play for the TCP/IP model, because the designers of TCP/IP employed the same techniques used to conquer the complexity of programming language *compilers* (design by analogy) in the *implementation* of its protocols and its layering scheme.

Like diplomatic protocols, communications protocols have to be agreed upon by the parties involved. To reach agreement a protocol is developed into a *technical standard*. International standards are developed by the International Organization for Standardization (*ISO*).

Communicating systems

The information exchanged between devices on a network or other communications medium is governed by rules (conventions) that can be set out in a technical specification called a communication protocol standard. The nature of the communication, the actual data exchanged and any state-dependent behaviors are defined by the specification. This approach is often taken for protocols in use by telecommunications.
In digital computing systems, the rules can be expressed by algorithms and datastructures, raising the opportunity of hardware independency. Expressing the

algorithms in a portable programming language, makes the protocol software operating system independent. The protocols in use by an operating system itself, lend themselves to be described this way and are usually, just like the rest of the operating system, distributed in binary or source form.

Operating systems are usually conceived of as consisting of a set of cooperating processes that manipulate a shared store (on the system itself) to communicate with each other. This communication is governed by well understood protocols and is only a small part of what a process is supposed to accomplish (managing system resources like cpu's, memory, timers, I/O devices etc, and providing controlled access to the resources), so these protocols can be embedded in the process code itself as small additional code fragments.

In contrast, communicating systems have to communicate with each other using shared transmission media, because there is no common memory. Unlike a memory store operation, a transmission does not need to be reliable and can involve different hardware and operating systems on different systems. This complicates matters up to a point that some kind of structuring is necessary to conquer the complexity of networking protocols, especially, when used on the Internet. The communicating systems can make use of different operating systems, as long as they agree to use the same kind of structuring and the same protocols for their communications.
To implement a networking protocol, the protocol software modules are to be interfaced with a framework assumed to be implemented on the machine's operating system. This framework implements the networking functionality of the operating system. Obviously, the framework needs to be as simple as it can be, to allow for an easier incorporation into the operating systems. The best known frameworks are the TCP/IP model and the OSI model.

At the time the Internet was formed, layering had proven to be a successful design approach for both compiler and operating system design and given the similarities between programming languages and communication protocols, it was intuitively felt that layering should be applied to the protocols as well. This gave rise to the concept of layered protocols which nowadays forms the basis of protocol design.

Systems do not use a single protocol to handle a transmission. Instead they use a set of cooperating protocols, sometimes called a protocol family or protocol suite. Some of the best known protocol suites include: IPX/SPX, X.25, AX.25, AppleTalk and TCP/IP. To cooperate the protocols have to communicate with each other, so there is an unnamed 'protocol' to do this. A technique used by this 'protocol' is called encapsulation, which makes it possible to pass messages from layer to layer in the framework.

The protocols can be arranged on functionality in groups, for instance there is a group of transport protocols. The functionalities are mapped on the layers, each layer solving a distinct class of problems relating to, for instance: application-, transport-, internet- and network interface-functions. To transmit a message, a protocol has to be selected from each layer, so some sort of multiplexing/demultiplexing takes place. The selection of the

next protocol, also part of the aforementioned 'protocol' is accomplished by extending the message with a protocolselector for each layer.

There's a myriad of protocols, but they all only differ in the details. For this reason the TCP/IP protocol suite can be studied to get the overall protocol picture. The Internet Protocol (IP) and the Transmission Control Protocol (TCP) are the most important of these, and the term Internet Protocol Suite, or TCP/IP, refers to a collection of its most used protocols. Most of the communication protocols in use on the Internet are described in the Request for Comments (RFC) documents of the Internet Engineering Task Force (IETF). RFC1122, in particular, documents the suite itself.

Basic requirements of protocols

The data representing the messages is to be sent and received on communicating systems to establish communications. Protocols should therefore specify rules governing the transmission. In general, much of the following should be addressed:

- *Data formats for data exchange*. In digital message bitstrings are exchanged. The bitstrings are divided in fields and each field carries information relevant to the protocol. Conceptually the bitstring is divided into two parts called the *header area* and the *data area*. The actual message is stored in the data area, so the header area contains the fields with more relevance to the protocol. The transmissions are limited in size, because the number of transmission errors is proportional to the size of the bitstrings being sent. Bitstrings longer than the maximum transfer unit (*MTU*) are divided in pieces of appropriate size. Each piece has almost the same header area contents, because only some fields are dependent on the contents of the data area (notably CRC fields, containing checksums that are calculated from the data area contents).

- *Address formats for data exchange*. The addresses are used to identify both the sender and the intended receiver(s). The addresses are stored in the header area of the bitstrings, allowing the receivers to determine whether the bitstrings are intended for themselves and should be processed or (when not to be processed) should be discarded. A connection between a sender and a receiver can be identified using an address pair *(sender address, receiver address)*. Usually some address values have special meanings. An all-*1*s address could be taken to mean all stations on the network, so sending to this address would result in a broadcast on the local network. Likewise, an all-'0's address could be taken to mean the sending station itself (as a synonym of the actual address). Stations have addresses unique to the local net, so usually the address is conceptually divided in two parts: a network address and the station address. The network address uniquely identifies the network on the *internetwork* (a network of networks). The rules describing the meanings of the address value are collectively called an *addressing scheme*.

- *Address mapping*. Sometimes protocols need to map addresses of one scheme on addresses of another scheme. For instance to translate a logical IP address specified by the application to a hardware address. This is referred to as *address mapping*. The mapping is implied in hierarchical address schemes where only a part of the address is used for the map address. In other cases the mapping needs to be described using tables.

- *Routing*. When systems are not directly connected, intermediary systems along the *route* to the intended receiver(s) need to forward messages (instead of discarding them) on behalf of the sender. Determining the route the message should take is called *routing*. On the Internet, the networks are connected using routers (gateways). This way of connecting networks is called *internetworking*. To determine the next router on the path to the destination, all systems consult locally stored tables consisting of *(destination network address, delivery address)* - entries and a special entry consisting of *(a 'catch-all' address, default router address)*. The delivery address is either the address of a router assumed to be closer to the destination and the hardware interface to be used to reach it, or the address of a hardware interface on the system directly connecting a network. The default router is used when no other entry matches the intended destination network.

- *Detection of transmission errors* is necessary, because no network is error-free. Bits of the bitstring become corrupted or lost. Usually, CRCs of the data area are added to the end of packets, making it possible for the receiver to notice many (nearly all) differences caused by errors, whilst recalculating the CRCs of the received packet and comparing them with the CRCs given by the sender. The receiver rejects the packets on CRC differences and arranges somehow for retransmission.

- *Acknowledgements* of correct reception of packets by the receiver are usually used to prevent the sender from retransmitting the packets. Some protocols, notably datagram protocols like the Internet Protocol (IP), do not acknowledge.

- *Loss of information - timeouts and retries*. Sometimes packets are lost on the network or suffer from long delays. To cope with this, a sender expects an acknowledgement of correct reception from the receiver within a certain amount of time. On timeouts, the packet is retransmitted. In case of a broken link the retransmission has no effect, so the number of retransmissions is limited. Exceeding the retry limit is considered an error.

- *Direction of information flow* needs to be addressed if transmissions can only occur in one direction at a time (*half-duplex* links). To gain control of the link a sender must wait until the line becomes idle and then send a message indicating its wish to do so. The receiver responds by acknowledging and waits for the transmissions to come. The sender only begins transmitting after the

acknowledgement. Arrangements have to be made to accommodate the case when two parties want to gain control at the same time.

- *Sequence control*. We have seen that long bitstrings are divided in pieces, that are send on the network individually. The pieces may get 'lost' on the network or arrive out of sequence, because the pieces can take different routes to their destination. Sometimes pieces are needlessly retransmitted, due to network congestion, resulting in duplicate pieces. By sequencing the pieces at the sender, the receiver can determine what was lost or duplicated and ask for retransmissions. Also the order in which the pieces are to be processed can be determined.

- *Flow control* is needed when the sender transmits faster than the receiver can process the transmissions or when the network becomes congested. Sometimes, arrangements can be made to slow down the sender, but in many cases this is outside the control of the protocol.

Getting the data across is only part of the problem. The data received has to be evaluated in the context of the progress of the conversation, so a protocol has to specify rules describing the context and explaining whether the (form of the) data fits this context or not. These kind of rules are said to express the *syntax* of the communications. Other rules determine whether the data is meaningful for the context in which the exchange takes place. These kind of rules are said to express the *semantics* of the communications.

Both intuitive descriptions as well as more formal specifications in the form of finite state machine models are used to describe the expected interactions of the protocol. Formal ways for describing the syntax of the communications are Abstract Syntax Notation One (a ISO standard) or Augmented Backus-Naur form (a IETF standard).

An example protocol: the ethernet protocol

To get a feel for protocols and what needs to be described, the *ethernet protocol* is explained in some detail.

Ethernet was invented at Xerox PARC in the early 1970s. It has gained widespread use on LANs and became standardized as IEEE standard 802.3. To connect to a LAN, a computer has to be equipped with an ethernet network interface card. The combination of computer and interface card is called a *station*. Using wiring, interface cards are all connected to a *hub*.
Conceptually, all stations share a single communication channel called a *shared bus*. Transmissions on this channel are received by all stations at (nearly) the same time. The hardware provides no indication to the sender about whether the transmission was delivered and is therefore called a *best-effort delivery* mechanism. A carrier wave is used to transmit the data in packets, referred to as *ethernet frames*. Each station is constantly tuned in on the channel, so each transmission is noticed. In order to determine whether the channel is free, the carrier wave can be sensed by the hardware; if not present the

channel is free for transmission.
Stations wanting to transmit, wait for the channel to become free and then start transmitting one single frame and then stop for a small amount of time before transmitting a next frame to allow others to transmit. The transmissions take time to reach all the stations due to the limited travelling speed of the signal (approximately 70% of the speed of light), so sometimes two stations start transmitting very shortly after each other, creating what is called *collisions*, because the second station notices too late that the channel is not free. The result is a scrambled signal, so the frames need to be retransmitted. During the transmissions, the hardware monitors for interference and if detected stops transmitting. To avoid further collisions, each station should wait for a different amount of time before starting the retransmission. The amount of time to wait is picked randomly from a range using an exponential backoff policy, which ensures that further collisions become increasingly improbable by increasing the random range exponentially (2^n) on each retry.
Ethernet cards are assigned unique 48 bit numbers, known as *ethernet addresses*, which are used to address the stations. To broadcast a message, a special *broadcast* address (all *1*s) is used. Ethernet establishes link level connections, which can be defined using both the destination and sources addresses. The frame format is as follows: a preamble (8 octets), destination address (6 octets), source address (6 octets), frame type (2 octets), frame data (46-1500 octets), a CRC (4 octets). The preamble is used to synchronize the receiving interface. It consists of 64 bits of alternating *0*s and *1*s. This bit pattern is relatively easy to detect and synchronize to, and therefore suitable as a start marker of the transmission. The frame type is a 16 bit integer identifying the type of data in the frame data. The *cyclic redundancy check* (CRC) is derived from the frame data and used by the receiver to verify the integrity of the transmitted data.
On reception of a transmission, the receiver uses the destination address to determine whether the transmission is relevant to the station or should be ignored. Both broadcast address and a destination address equal to the address of the receiving interface card are to be handled. The frame type is used by the operating system on the receiving station to select the appropriate protocol module (i.e. the ethernet protocol module). Ethernet frames are said to be *self-identifying*, because of the frame type. Self-identifying frames make it possible to intermix multiple protocols on the same physical network and allow a single computer to use multiple protocols together.

Protocols and programming languages

The following two analogies between communications and computations are used throughout this text:

- Protocols are to communications what algorithms are to computations.
- Protocols are to communications what programming languages are to computations.

The analogies have important consequences for both the design and the development of protocols.

The word protocol has different meanings in the two analogies: *1*. a variant of (or something like) an algorithm. *2*. a variant of (or something like) a programming language. Arguably, it would make more sense to speak of a protocolling language (compare algorithmic language) when using the word protocol in its second meaning, but this is not a standard practice in a networking context. To further clarify on this, one has to consider the fact that algorithms, programs and protocols are just different ways of describing expected behaviour of interacting objects. Assuming the use of *pseudo-code* in the case of both algorithms and protocols, all of the representations can be referred to as *applications* of *programming languages*. In other words protocols are applications of protocolling languages.

A familiar example of a protocolling language is the HTML language used to describe webpages which are the actual webprotocols.

In programming languages the association of *identifiers* to a *value* is termed a *definition*. Program text is structured using *block* constructs and definitions can be local to a block. The localized association of an identifier to a value established by a definition is termed a *binding* and the region of program text in which a binding is effective is known as its *scope*. The computational state is kept using two components: the *environment*, used as a record of identifier bindings, and the *store*, which is used as a record of the effects of assignments.

In communications, message values are transferred using transmission media. By analogy, the equivalent of a store would be a collection of transmission media, instead of a collection of memory locations. A valid assignment in a protocol (as an analog of programming language) could be *Ethernet:='message'* , meaning a message is to be broadcast on the local ethernet.
On a transmission medium there can be many receivers. For instance a mac-address identifies an ether network card on the transmission medium (the 'ether'). In our imaginary protocol, the assignment *ethernet[mac-address]:=message value* could therefore make sense.

By extending the assignment statement of an existing programming language with the semantics described, a protocolling language could easily be imagined.

Operating systems provide reliable communication and synchronization facilities for communicating objects confined to the same system by means of *system libraries*. A programmer using a general purpose programming language (like C or ADA) can use the routines in the libraries to implement a protocol, instead of using a dedicated protocolling language.

Language layering

Early compilers translated high-level language sources to machine code. Instead of translating directly into machine code, some compilers translate to a machine independent intermediate code in order to enhance portability of the compiler and

minimize design efforts. Often at the expense of execution speed. The intermediate language defines a *virtual machine* that can execute all programs written in the intermediate language (a machine is defined by its language and vice versa). The intermediate code instructions are translated into equivalent machine code sequences by a *code generator* to create executable code. It is also possible to skip the generation of machine code by actually implementing the virtual machine in machine code. This virtual machine implementation is called an *interpreter*, because it reads in the intermediate code instructions one by one and after each read executes the equivalent machine code sequences (the interpretation) of the read intermediate instruction directly.

Translation of a programming language into intermediate code and intermediate code into machine code to generate an executable is an example of layering of languages. Because languages define machines, this can be viewed as a layering of virtual machines as well. Using machines (or mechanisms) to build more complex machines, which in their turn are used for even more complex machines etcetera, results in what is referred to as a *multi-level machine*.
A modern computer system is usually a six level machine with the following levels (also called layers) present: the digital logic level, the microprogramming level, the conventional machine level, the operating system level, the assembly language level and the problem-oriented programming language level.
The Internet can be viewed as a layered machine as well: a (best-effort) hardware delivery mechanism is used to build a connectionless packet delivery system on top of which a reliable transport system is build, which is used to build an application. The delivery system is defined by the IP protocol and the transport system by the TCP protocol.

Universal protocols

Despite their numbers, networking protocols show little variety, because all networking protocols use the same underlying principles and concepts, in the same way. So, the use of a general purpose programming language would yield a large number of applications only differing in the details. A suitably defined (dedicated) protocolling language would therefore have little syntax, perhaps just enough to specify some parameters or optional modes of operation, because its virtual machine would have incorporated all possible principles and concepts making the virtual machine itself a *universal* protocol. The protocolling language would have some syntax and a lot of semantics describing this universal protocol and would therefore in effect be a protocol, hardly differing from this universal networking protocol. In this (networking) context a protocol is a language.

The notion of a universal networking protocol provides a rationale for standardization of networking protocols; assuming the existence of a universal networking protocol, development of protocol standards using a consensus model (the agreement of a group of experts) might be a viable way to coordinate protocol design efforts.

Networking protocols operate in very heterogeneous environments consisting of very different network technologies and a (possibly) very rich set of applications, so a single

universal protocol would be very hard to design and implement correctly. Instead, the IETF decided to reduce complexity by assuming a relatively simple network architecture allowing decomposition of the single universal networking protocol into two generic protocols, TCP and IP, and two classes of specific protocols, one dealing with the low-level network details and one dealing with the high-level details of common network applications (remote login, file transfer, email and web browsing). ISO choose a similar but more general path, allowing other network architectures, to standardize protocols.

Protocol design

Communicating systems operate in parallel. The programming tools and techniques for dealing with parallel processes are collectively called *concurrent programming*. Concurrent programming only deals with the synchronization of communication. The syntax and semantics of the communication governed by a low-level protocol usually have modest complexity, so they can be coded with relative ease. High-level protocols with relatively large complexity could however merit the implementation of language interpreters. An example of the latter case is the HTML language.
Concurrent programming has traditionally been a topic in operating systems theorie texts. Formal verification seems indispensable, because concurrent programs are notorious for the hidden and sophisticated bugs they contain. A mathematical approach to the study of concurrency and communication is referred to as *Communicating Sequential Processes* (CSP). Concurrency can also be modelled using finite state machines like Mealy- and Moore machines. Mealy- and Moore machines are in use as design tools in digital electronics systems, which we encounter in the form of hardware used in telecommunications or electronic devices in general.

This kind of design can be a bit of a challenge to say the least, so it is important to keep things simple. For the Internet protocols, in particular and in retrospect, this meant a basis for protocol design was needed to allow decomposition of protocols into much simpler, cooperating protocols.

Concurrent programming

A *concurrent program* is an abstraction of cooperating processes suitable for formal treatment and study. The goal of the abstraction is to prove correctness of the program assuming the existence of some basic synchronization or data exchange mechanisms provided by the operating system (or other software) or hardware. The mechanisms are complex, so more convenient higher level *primitives* are implemented with these mechanisms. The primitives are used to construct the concurrent program. The basic primitive for synchronization is the *semaphore*. All other primitives (locks, reentrant mutexes, semaphores, monitors, message passing, tuple space) can be defined using semaphores. The semaphore is sufficiently elementary to be successfully studied by formal methods.

In order to synchronize or exchange data the processes must communicate by means of either a *shared memory*, used to store data or access-restricted procedures, or the

sending/receiving of signals (*message passing*) using a shared transmission medium. Most third generation operating systems implement separate processes that use special instructions to ensure only one process can execute the restricted procedures. On distributed systems there is no common central memory so the communications are always by means of message passing. In this case the processes simply have to wait for each other (*synchronization by rendezvous*) before exchanging data.

Conceptually, the concurrent program consists of several sequential processes whose execution sequences are interleaved. The execution sequences are divided into sections. A section manipulating shared resources is called a *critical section*. The interleaving scheme makes no timing assumptions other than that no process halts in its critical section and that ready processes are eventually scheduled for execution. For correct operation of the program, the critical sections of the processes need to be properly sequenced and synchronized. This is achieved using small code fragments (*protocols*) at the start and the end of the critical sections. The code fragments determine whether the critical sections of two communicating processes should execute in parallel (*rendezvous of processes*) or should be executed sequentially (*mutual exclusion of processes*).
A concurrent program is correct if it does not violate some safety property such as mutual exclusion or rendezvous of critical sections and does not suffer of liveness properties such as deadlock or lockout. Correctness of the concurrent program can only be shown using a mathematical argument. Specifications of concurrent programs can be formulated using formal logics (like CSP) which make it possible to prove properties of the programs. Incorrectness can be shown using execution scenarios.

Mutual exclusion is extensively studied in the *mutual exclusion problem*. The rendezvous is studied in the *producer-consumer problem* in which a producer process only produces data if and only if the consumer process is ready to consume the data. Although both problems only involve two processes, their solutions require rather complex algorithms (Dekker's algorithm, Lamport's bakery algorithm). The readers-writers problem is a generalization of the mutual exclusion problem. The dining philosophers problem is a classical problem sufficiently difficult to expose many of the potential pitfalls of newly proposed primitives.

A *basis* for protocol design

Systems do not use a single protocol to handle a transmission. Instead they use a set of cooperating protocols, sometimes called a protocol family or protocol suite. To cooperate the protocols have to communicate with each other, so some kind of conceptual framework is needed to make this communication possible. Also note that software is needed to implement both the 'xfer-mechanism' and a protocol (no protocol, no communication).

In literature there are numerous references to the analogies between computer communication and programming. By analogy we could say that the aforementioned 'xfer-mechanism' is comparable to a *cpu*; a 'xfer-mechanism' performs communications and a *cpu* performs computations and the 'framework' introduces something that allows

the protocols to be designed independent of one and another by providing separate execution environments for the protocols. Furthermore, it is repeatedly stated that *protocols are to computer communication what programming languages are to computation*.

Protocol layering

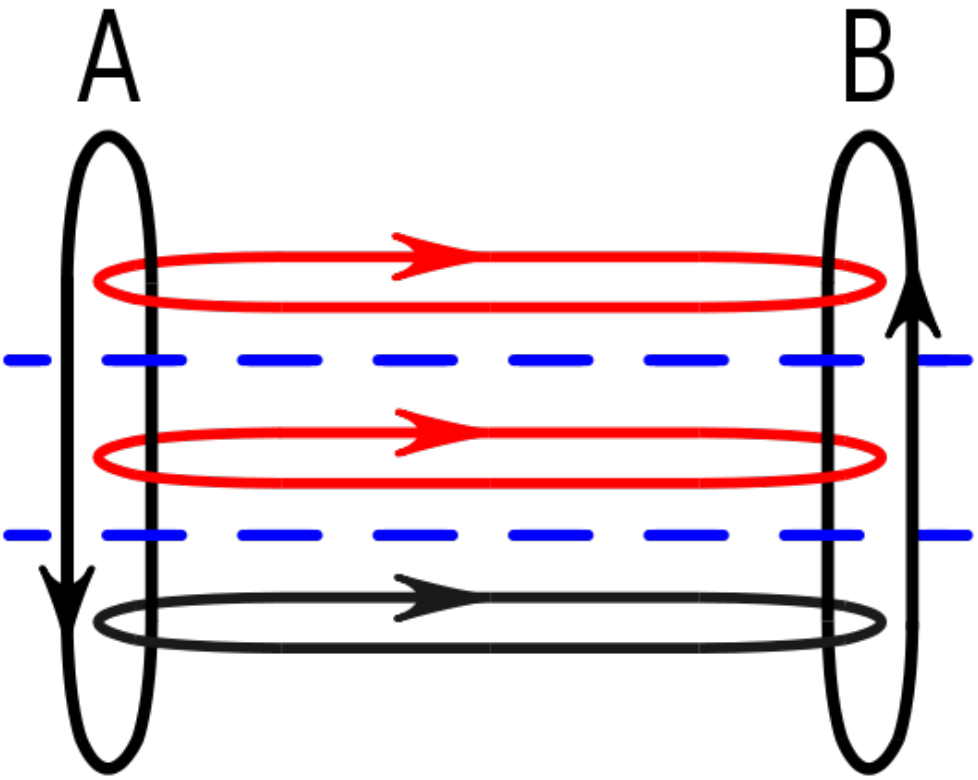


Figure 3. Message flows using a protocol suite.

Protocol layering now forms the basis of protocol design. It allows the decomposition of single, complex protocols into simpler, cooperating protocols, but it is also a functional decomposition, because each protocol belongs to a functional class, called a *protocol layer*. The protocol layers each solve a distinct class of communications problems. The Internet protocol suite consists of the following layers: application-, transport-, internet- and network interface-functions. Together, the layers make up a *layering scheme* or

model.
In computations, we have algorithms and data, and in communications, we have protocols and messages, so the analog of a data flow diagram would be some kind of message flow diagram. To visualize protocol layering and protocol suites, a diagram of the message flows in and between two systems, A and B, is shown in figure 3.
The systems both make use of the same protocol suite. The vertical flows (and protocols) are *in system* and the horizontal message flows (and protocols) are *between* systems. The message flows are governed by rules, and dataformats specified by protocols. The blue lines therefore mark the boundaries of the (horizontal) protocol layers.

The vertical protocols are not layered because they don't obey the *protocol layering principle* which states that *a layered protocol is designed so that layer* n *at the destination receives exactly the same object sent by layer* n *at the source*. The horizontal protocols are *layered protocols* and all belong to the protocol suite. Layered protocols allow the protocol designer to concentrate on one layer at a time, without worrying about how other layers perform.
The vertical protocols neednot be the same protocols on both systems, but they have to satisfy some minimal assumptions to ensure the protocol layering principle holds for the layered protocols. This can be achieved using a technique called *Encapsulation*.
Usually, a message or a stream of data is divided into small pieces, called *messages* or *streams*, *packets*, *IP datagrams* or *network frames* depending on the layer in which the pieces are to be transmitted. The pieces contain a *header area* and a *data area*. The data in the header area identifies the source and the destination on the network of the packet, the protocol, and other data meaningful to the protocol like CRC's of the data to be send, data length, and a timestamp.
The rule enforced by the vertical protocols is that the pieces for transmission are to be *encapsulated* in the data area of all lower protocols on the sending side and the reverse is to happen on the receiving side. The result is that at the lowest level the piece looks like this: 'Header1,Header2,Header3,data' and in the layer directly above it: 'Header2,Header3,data' and in the top layer: 'Header3,data', both on the sending and receiving side. This rule therefore ensures that the protocol layering principle holds and effectively virtualizes all but the lowest transmission lines, so for this reason some message flows are coloured red in figure 3.
To ensure both sides use the same protocol, the pieces also carry data identifying the protocol in their header.

The design of the protocol layering and the network (or Internet) architecture are interrelated, so one cannot be designed without the other. Some of the more important features in this respect of the Internet architecture and the network services it provides are described next.

- The Internet offers *universal interconnection*, which means that any pair of computers connected to the Internet is allowed to communicate. Each computer is identified by an *address* on the Internet. All the interconnected physical networks appear to the user as a single large network. This interconnection scheme is called an *internetwork* or *internet*.

- Conceptually, an *Internet addresses* consists of a *netid* and a *hostid*. The netid identifies a network and the hostid identifies a host. The term host is misleading in that an individual computer can have multiple network interfaces each having its own Internet address. An Internet Address identifies a connection to the network, not an individual computer. The netid is used by routers to decide where to send a packet.
- *Network technology independence* is achieved using the low-level *address resolution protocol* (ARP) which is used to map Internet addresses to physical addresses. The mapping is called *address resolution*. This way physical addresses are only used by the protocols of the network interface layer. The TCP/IP protocols can make use of almost any underlying communication technology.

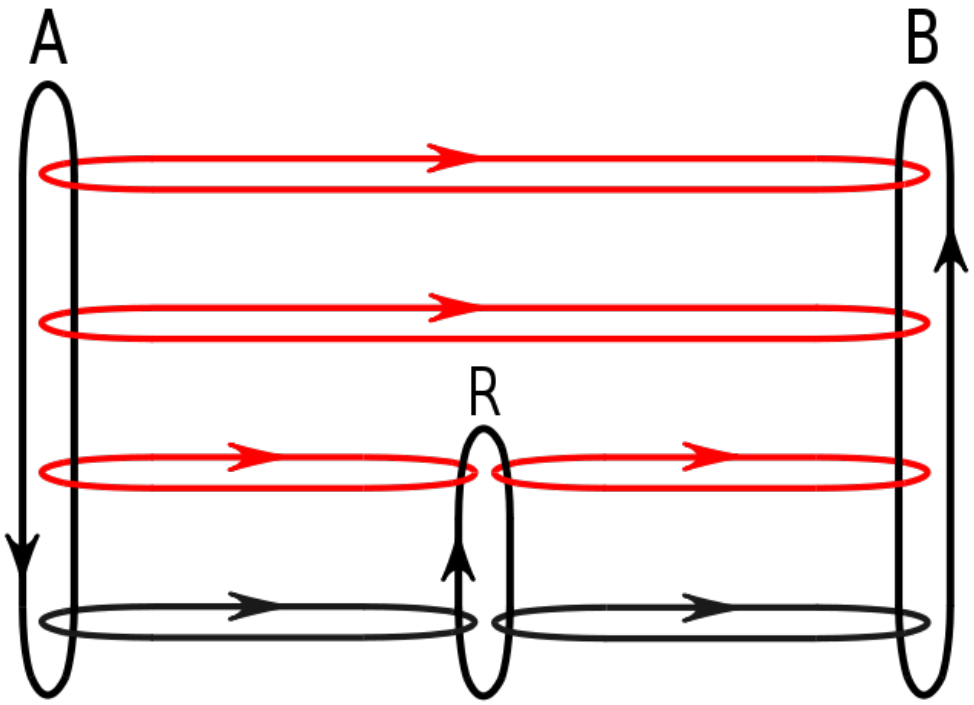


Figure 4. Message flows in the presence of a router

Physical networks are interconnected by routers. Routers forward packets between interconnected networks making it possible for hosts to reach hosts on other physical networks. The message flows between two communicating system A and B in the presence of a router R are illustrated in figure 4. Datagrams are passed from router to router until a router is reached that can deliver the datagram on a physically attached network (called *direct delivery*). To decide whether a datagram is to be delivered directly or is to be send to a router closer to the

destination, a table called the *IP routing table* is consulted. The table consists of pairs of networkids and the paths to be taken to reach known networks. The path can be an indication that the datagram should be delivered directly or it can be the address of a router known to be closer to the destination. A special entry can specify that a default router is chosen when there are no known paths.

- *All networks are treated equal*. A LAN, a WAN or a point-to-point link between two computers are all considered as one network.
- A *Connectionless packet delivery (or packet-switched) system (or service)* is offered by the Internet, because it adapts well to different hardware, including best-effort delivery mechanisms like the *ethernet*. Connectionless delivery means that the messages or streams are divided in pieces that are *multiplexed* separately on the high speed intermachine connections allowing the connections to be used concurrently. Each piece carries information identifying the destination. The delivery of packets is said to be *unreliable*, because packets may be lost, duplicated, delayed or delivered out of order without notice to the sender or receiver. Unreliability arises only when resources are exhausted or underlying networks fail. The unreliable connectionless delivery system is defined by the *Internet Protocol* (IP). The protocol also specifies the *routing* function, which chooses a path over which data will be send. It is also possible to use TCP/IP protocols on *connection oriented systems*. Connection oriented systems build up *virtual circuits* (paths for exclusive use) between senders and receivers. Once build up the IP datagrams are send as if they were data through the virtual circuits and forwarded (as data) to the IP protocol modules. This technique, called *tunneling*, can be used on X.25 networks and ATM networks.
- A *reliable stream transport service* using the unreliable connectionless packet delivery service is defined by the *transmission control protocol* (TCP). The services are layered as well and the application programs residing in the layer above it, called the *application services*, can make use of TCP. Programs wishing to interact with the packet delivery system itself can do so using the *user datagram protocol* (UDP).

Software layering

Having established the protocol layering and the protocols, the protocol designer can now resume with the software design. The software has a layered organization and its relationship with protocol layering is visualized in figure 5.

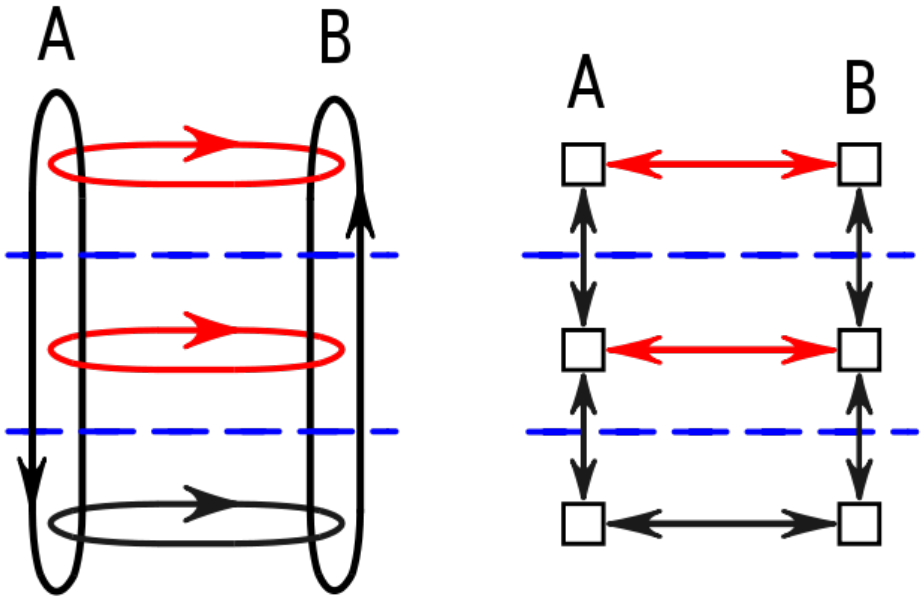


Figure 5: Protocol and software layering

The software modules implementing the protocols are represented by cubes. The information flow between the modules is represented by arrows. The (top two horizontal) red arrows are virtual. The blue lines mark the layer boundaries.

To send a message on system A, the top module interacts with the module directly below it and hands over the message to be encapsulated. This module reacts by encapsulating the message in its own data area and filling in its header data in accordance with the protocol it implements and interacts with the module below it by handing over this newly formed message whenever appropriate. The bottom module directly interacts with the bottom module of system B, so the message is send across. On the receiving system B the reverse happens, so ultimately (and assuming there were no transmission errors or protocol violations etc.) the message gets delivered in its original form to the topmodule of system B.
On protocol errors, a receiving module discards the piece it has received and reports back the error condition to the original source of the piece on the same layer by handing the error message down or in case of the bottom module sending it across.
The division of the message or stream of data into pieces and the subsequent reassembly are handled in the layer that introduced the division/reassembly. The reassembly is done at the destination (i.e. not on any intermediate routers).

TCP/IP software is organized in four layers.

- *Application layer*. At the highest layer, the services available across a TCP/IP internet are accessed by application programs. The application chooses the style of transport to be used which can be a sequence of individual messages or a continuous stream of bytes. The application program passes data to the *transport layer* for delivery.
- *Transport layer*. The transport layer provides communication from one application to another. The transport layer may regulate flow of information and provide reliable transport, ensuring that data arrives without error and in sequence. To do so, the receiving side sends back acknowledgments and the sending side retransmits lost pieces called packets. The stream of data is divided into packets by the module and each packet is passed along with a destination address to the next layer for transmission. The layer must accept data from many applications concurrently and therefore also includes codes in the packet header to identify the sending and receiving application program.
- *Internet layer*. The Internet layer handles the communication between machines. Packets to be send are accepted from the transport layer along with an identification of the receiving machine. The packets are encapsulated in IP datagrams and the datagram headers are filled. A routing algorithm is used to determine if the datagram should be delivered directly or send to a router. The datagram is passed to the appropriate network interface for transmission. Incoming datagrams are checked for validity and the routing algorithm is used to decide whether the datagram should be processed locally or forwarded. If the datagram is addressed to the local machine, the datagram header is deleted and the appropriate transport protocol for the packet is chosen. ICMP error and control messages are handled as well in this layer.
- *Network interface layer*. The network interface layer is responsible for accepting IP datagrams and transmitting them over a specific network. A network interface may consist of a device driver or a complex subsystem that uses its own data link protocol.

Program translation has been divided into four subproblems: compiler, assembler, link editor, and loader. As a result, the translation software is layered as well, allowing the software layers to be designed independently. Noting that the ways to conquer the complexity of program translation could readily be applied to protocols because of the analogy between programming languages and protocols. The designers of the TCP/IP protocol suite were keen on imposing the same layering on the software framework. This can be seen in the TCP/IP layering by considering the translation of a *pascal program* (message) that is compiled (function of the application layer) into an *assembler program* that is assembled (function of the transport layer) to *object code* (pieces) that is linked (function of the Internet layer) together with *library object code* (routing table) by the link editor, producing *relocatable machine code* (datagram) that is passed to the loader which fills in the memory locations (ethernet addresses) to produce *executeable code* (network frame) to be loaded (function of the network interface layer) into physical memory (transmission medium). To show just how closely the analogy fits, the terms

between parentheses in the previous sentence denote the relevant analogs and the terms written *cursively* denote data representations. Program translation forms a linear sequence, because each layer's output is passed as input to the next layer. Furthermore, the translation process involves multiple data representations. We see the same thing happening in protocol software where multiple protocols define the datarepresentations of the data passed between the software modules.

The network interface layer uses physical addresses and all the other layers only use IP addresses. The boundary between network interface layer and Internet layer is called the *high-level protocol address boundary*. The modules below the application layer are generally considered part of the operating system. Passing data between these modules is much less expensive than passing data between an application program and the transport layer. The boundary between application layer and transport layer is called the *operating system boundary*.

Strict layering

Strictly adhering to a layered model, a practice known as strict layering, is not always the best approach to networking. Strict layering, can have a serious impact on the performance of the implementation, so there is at least a trade-off between simplicity and performance. Another, perhaps more important point can be shown by considering the fact that some of the protocols in the Internet Protocol Suite cannot be expressed using the TCP/IP model, in other words some of the protocols behave in ways not described by the model. To improve on the model, an offending protocol could, perhaps be split up into two protocols, at the cost of one or two extra layers, but there is a hidden caveat, because the model is also used to provide a conceptual view on the suite for the intended users. There is a trade-off to be made here between preciseness for the designer and clarity for the intended user.

Protocol development

For communication to take place, protocols have to be agreed upon. Recall that in digital computing systems, the rules can be expressed by algorithms and datastructures, raising the opportunity of hardware independency. Expressing the algorithms in a portable programming language, makes the protocolsoftware operating system independent. The sourcecode could be considered a protocol specification. This form of specification, however is not suitable for the parties involved.
For one thing, this would enforce a source on all parties and for another, proprietary software producers would not accept this. By describing the software interfaces of the modules on paper and agreeing on the interfaces, implementers are free to do it their way. This is referred to as source independency. By specifying the algorithms on paper and detailing hardware dependencies in an unambiguous way, a *paper draft* is created, that when adhered to and published, ensures interoperability between software and hardware. Such a paper draft can be developed into a *protocol standard* by getting the approval of a *standards organization*. To get the approval the paper draft needs to enter and successfully complete the *standardization process*. This activity is referred to as *protocol*

development. The members of the standards organization agree to adhere to the standard on a voluntary basis. Often the members are in control of large market-shares relevant to the protocol and in many cases, standards are enforced by law or the government, because they are thought to serve an important public interest, so getting approval can be very important for the protocol. It should be noted though that in some cases protocol standards are not sufficient to gain widespread acceptance i.e. sometimes the sourcecode needs to be disclosed enforced by law or the government in the interest of the public.

The need for protocol standards

The need for protocol standards can be shown by looking at what happened to the bi-sync protocol (BSC) invented by IBM. BSC is an early link-level protocol used to connect two separate nodes. It was originally not intended to be used in a multinode network, but doing so revealed several deficiencies of the protocol. In the absence of standardization, manufacturers and organizations felt free to 'enhance' the protocol, creating incompatible versions on their networks. In some cases, this was deliberately done to discourage users from using equipment from other manufacturers. There are more than 50 variants of the original bi-sync protocol. One can assume, that a standard would have prevented at least some of this from happening.
In some cases, protocols gain market dominance without going through a standardization process. Such protocols are referred to as *de facto standards*. De facto standards are common on emerging markets, niche markets, or markets that are monopolized (or oligopolized). They can hold a market in a very negative grip, especially when used to scare away competition. From a historical perspective, standardization should be seen as a measure to counteract the ill-effects of de facto standards. Positive exceptions exist; a 'de facto standard' operating system like GNU/Linux does not have this negative grip on its market, because the sources are published and maintained in an open way, thus inviting competition. Standardization is therefore not the only solution for *open systems interconnection*.

Standards organizations

Some of the standards organizations of relevance for communications protocols are the International Organization for Standardization (ISO), the International Telecommunications Union (ITU), the Institute of Electrical and Electronics Engineers (IEEE), and the Internet Engineering Task Force (IETF). The IETF maintains the protocols in use on the Internet. The IEEE controls many software and hardware protocols in the electronics industry for commercial and consumer devices. The ITU is an umbrella organization of telecommunications engineers designing the public switched telephone network (PSTN), as well as many radio communication systems. For marine electronics the NMEA standards are used. The World Wide Web Consortium (W3C) produces protocols and standards for Web technologies.

International standards organizations are supposed to be more impartial than local organizations with a national or commercial self-interest to consider. Standards

organizations also do research and development for standards of the future. In practice, the standards organizations mentioned, cooperate closely with each other.

The standardization process

The standardization process starts off with ISO commissioning a sub-committee workgroup. The workgroup issues working drafts and discussion documents to interested parties (including other standards bodies) in order to provoke discussion and comments. This will generate a lot of questions, much discussion and usually some disagreement on what the standard should provide and if it can satisfy all needs (usually not). All conflicting views should be taken into account, often by way of compromise, to progress to a *draft proposal* of the working group.
The draft proposal is discussed by the member countries' standard bodies and other organizations within each country. Comments and suggestions are collated and national views will be formulated, before the members of ISO vote on the proposal. If rejected, the draft proposal has to consider the objections and counter-proposals to create a new draft proposal for another vote. After a lot of feedback, modification, and compromise the proposal reaches the status of a *draft international standard*, and ultimately an *international standard*.
The process normally takes several years to complete. The original paper draft created by the designer will differ substantially from the standard, and will contain some of the following 'features':

- Various optional modes of operation, for example to allow for setup of different packet sizes at startup time, because the parties could not reach consensus on the optimum packet size.
- Parameters that are left undefined or allowed to take on values of a defined set at the discretion of the implementor. This often reflects conflicting views of some of the members.
- Parameters reserved for future use, reflecting that the members agreed the facility should be provided, but could not reach agreement on how this should be done in the available time.
- Various inconsistencies and ambiguities will inevitably be found when implementing the standard.

International standards are reissued periodically to handle the deficiencies and reflect changing views on the subject.

Future of standardization (OSI)

A lesson learned from ARPANET (the predecessor of the Internet) is that standardization of protocols is not enough, because protocols also need a framework to operate. It is therefore important to develop a general purpose, future-proof framework suitable for *structured protocols* (such as layered protocols) and their standardization. This would prevent protocol standards with overlapping functionality and would allow clear definition of the responsibilities of a protocol at the different levels (layers). This gave

rise to the ISO *Open Systems Interconnection reference model* (RM/OSI), which is used as a framework for the design of standard protocols and services conforming to the various layer specifications.

In the OSI model, communicating systems are assumed to be connected by an underlying physical medium providing a basic (and unspecified) transmission mechanism. The layers above it are numbered (from one to seven); the n^{th} layer is referred to as (n)-layer. Each layer provides service to the layer above it (or at the top to the application process) using the services of the layer immediately below it. The layers communicate with each other by means of an interface, called a *service access point*. Corresponding layers at each system are called *peer entities*. To communicate, two peer entities at a given layer use a (n)-protocol, which is implemented by using services of the (n-1)-layer. When systems are not directly connected, intermediate peer entities (called *relays*) are used. An *address* uniquely identifies a service access point. The address naming domains need not be restricted to one layer, so it is possible to use just one naming domain for all layers. For each layer there are two types of standards: protocol standards defining how peer entities at a given layer communicate, and service standards defining how a given layer communcates with the layer above it.

In the original version of RM/OSI, the layers and their functionality are (from highest to lowest layer):

- The *application layer* may provide the following services to the application processes: identification of the intended communication partners, establishment of the necessary authority to communicate, determination of availability and authentication of the partners, agreement on privacy mechanisms for the communication, agreement on responsibility for error recovery and procedures for ensuring data integrity, synchronization between cooperating application processes, identification of any constraints on syntax (e.g. character sets and data structures), determination of cost and acceptable quality of service, selection of the dialogue discipline, including required logon and logoff procedures.
- The *presentation layer* may provide the following services to the application layer: a request for the establishment of a session, data transfer, negotiation of the syntax to be used between the application layers, any necessary syntax transformations, formatting and special purpose transformations (e.g. data compression and data encryption).
- The *session layer* may provide the following services to the presentation layer: establishment and release of session connections, normal and expedited data exchange, a quarantine service which allows the sending presentation entity to instruct the receiving session entity not to release data to its presentation entity without permission, interaction management so presentation entities can control whose turn it is to perform certain control functions, resynchronization of a session connection, reporting of unrecoverable exceptions to the presentation entity.
- The *transport layer* provides reliable and transparent data transfer in a cost effective way as required by the selected quality of service. It may support the

multiplexing of several transport connections on to one network connection or split one transport connection into several network connections.
- The *network layer* does the setup, maintenance and release of network paths between transport peer entities. When relays are needed, routing and relay functions are provided by this layer. The quality of service is negotiated between network and transport entities at the time the connection is setup. This layer is also responsible for (network) congestion control.
- The *data link layer* does the setup, maintenance and release of data link connections. Errors occurring in the physical layer are detected and may be corrected. Errors are reported to the network layer. The exchange of data link units (including flow control) is defined by this layer.
- The *physical layer* describes details like the electrical characteristics of the physical connection, the transmission techniques used, and the setup, maintenance and clearing of physical connections.

In contrast to the TCP/IP layering scheme, which assumes a connectionless network, RM/OSI assumed a connection oriented network. Connection oriented networks are more suitable for wide area networks and connectionless networks are more suitable for local area networks. Using connections to communicate implies some form of session and (virtual) circuits, hence the (in the TCP/IP model lacking) session layer. The constituent members of ISO were mostly concerned with wide area networks, so development of RM/OSI concentrated on connection oriented networks and connectionless networks were only mentioned in an addendum to RM/OSI. At the time, the IETF had to cope with this and the fact that the Internet needed protocols which simple were not there. As a result the IETF developed its own standardization process based on "rough consensus and running code". The standardization process is described by RFC2026.
Nowadays, the IETF has become a standards organization for the protocols in use on the Internet. RM/OSI has extended its model to include connectionless services and because of this, both TCP and IP could be developed into international standards.

Taxonomies

Classification schemes for protocols usually focus on domain of use and function. As an example of domain of use, connection-oriented protocols and connectionless protocols are used on connection-oriented networks and connectionless networks respectively. For an example of function consider a tunneling protocol, which is used to encapsulate packets in a high-level protocol, so the packets can be passed across a transport system using the high-level protocol.

A *layering scheme* combines both function and domain of use. The dominant layering schemes are the ones proposed by the IETF and by ISO. Both layering schemes are considered less than perfect, but they do represent current progress in protocol design and development. Despite the fact that the underlying assumptions of the layering schemes are different enough to warrant distinguishing the two, it is a common practice to compare the two by relating common protocols to the layers of the two schemes. For an example of this practice see: List of network protocols.

The layering scheme from the IETF is called *Internet layering* or *TCP/IP layering*. The functionality of the layers has been described in the section on software layering and an overview of protocols using this scheme is given here on Internet protocols.
The layering scheme from ISO is called *the OSI model* or *ISO layering*. The functionality of the layers has been described in the section on the future of standardization and an overview of protocols using this scheme is given here on OSI protocols.

Common types of protocols

The Internet Protocol is used in concert with other protocols within the Internet Protocol Suite. Prominent members of which include:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Internet Control Message Protocol (ICMP)
- Hypertext Transfer Protocol (HTTP)
- Post Office Protocol (POP3)
- File Transfer Protocol (FTP)
- Internet Message Access Protocol (IMAP)

Other instances of high level interaction protocols are:

- IIOP
- RMI
- DCOM
- DDE
- SOAP

Chapter 7

Uploading and Downloading

In computer networks, to *download* means to receive data to a local system from a remote system, or to initiate such a data transfer. Examples of a remote system from which a download might be performed include a webserver, FTP server, email server, or other similar systems. A *download* can mean either any file that is offered for downloading or that has been downloaded, or the process of receiving such a file.

It has become more common to mistake and confuse the meaning of downloading and installing or simply combine them incorrectly together.

The inverse operation, *uploading*, can refer to the sending of data from a local system to a remote system such as a server or another client with the intent that the remote system should store a copy of the data being transferred, or the initiation of such a process. The words first came into popular usage among computer users with the increased popularity of Bulletin Board Systems (BBSs), facilitated by the widespread distribution and implementation of dial-up access the in the 1970s.

Download

A symbol for downloading to a hard drive.

The use of the terms **uploading and downloading** often imply that the data sent or received is to be stored permanently, or at least stored more than temporarily. In contrast, the term downloading is distinguished from the related concept of *streaming*, which indicates the receiving of data that is used near immediately as it is received, while the transmission is still in progress and which may not be stored long-term, whereas in a process described using the term downloading, this would imply that the data is only usable when it has been received in its entirety. Increasingly, websites that offer streaming media or media displayed in-browser, such as YouTube, and which place restrictions on the ability of users to save these materials to their computers after they have been received, say that *downloading* is not permitted. In this context, "download" implies specifically "receive and save" instead of simply "receive". However, it is also important to note that "downloading" is not the same as "transferring" (i.e., sending/receiving data between two storage devices would be a transferral of data, but receiving data from the Internet would be considered a download of data).

Sideload

When applied to local transfers (sending data from one local system to another local system), it is often difficult to decide if it is an upload or download, as both source and destination are in the local control of the user. Technically if the user uses the receiving device to initiate the transfer then it would be a download and if they used the sending device to initiate it would be an upload. However, as most non-technical users tend to use the term download to refer to any data transfer, the term "sideload" is sometimes being used to cover all local to local transfers to end this confusion.

Remote upload

When there is a transfer of data from a remote system to another remote system, the process is called "remote uploading". This is used by some online file hosting services.

Remote uploading is also used in situations where the computers that need to share data are located on a distant high speed local area network, and the remote control is being performed using a comparatively slow dialup modem connection.

For example:

- The user remotely accesses a file hosting service at MyRemoteHost.
- The user finds a public file at PublicRemoteHost and wants to keep a copy in their MyRemoteHost.
- To have it done they "remote upload" the file from PublicRemoteHost to MyRemoteHost.
- None of the hosts are located on the user's local network.

Without remote uploadin functionality, the user would be required to download the file first to their local host and then re-upload it to the remote file hosting server.

Where the connection to the remote computers is via a dialup connection, the transfer time required to download locally and then re-upload could increase from seconds, to hours or days.

Chapter 8

Differential Coding & Analog Transmission

Differential Coding

In digital communications, **differential coding** is a technique used to provide *unambiguous* signal reception when using some types of modulation. It makes data to be transmitted to depend not only on the current bit (or symbol), but also on the previous one.

The common types of modulation that require differential coding include phase shift keying and quadrature amplitude modulation.

Purposes of differential coding

To demodulate BPSK one needs to make a local oscillator *synchronous* with the remote one. This is accomplished by a carrier recovery circuit. However, a carrier can be recovered in different ways, depending upon a valid phases count (2 for BPSK).

For this coding, if a carrier is recovered incorrectly, the received data are inverted.

Assuming that x_i is a bit intended for transmission, and y_i is a bit actually transmitted (differentially encoded), if

$$y_i = y_{i-1} \oplus x_i, \qquad (1)$$

is transmitted, then on the decoding side

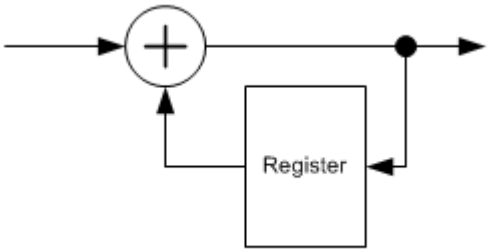
$$x_i = y_i \oplus y_{i-1}. \qquad (2)$$

can be reconstructed, where $\oplus$ indicates binary or modulo-2 addition.

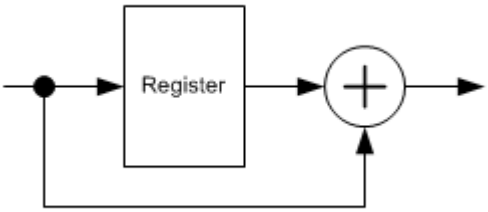
Now x_i depends only on a difference between y_i and y_{i-1} and not on their values. So, whether the data stream is inverted or not, the decoded data will always be correct.

When data is transmitted over twisted-pair wires, it is easy to accidentally insert an extra half-twist in the cable between the transmitter and the receiver. When this happens, the received data are inverted. There are several different line codes designed to be **polarity insensitive** -- whether the data stream is inverted or not, the decoded data will always be correct. The line codes with this property include differential Manchester encoding, bipolar encoding, NRZI, biphase mark code, coded mark inversion, and MLT-3 encoding.

Conventional differential coding



A differential encoder



A differential decoder

A method illustrated above can deal with a data stream inversion (it is called *180° ambiguity*). Sometimes it is enough (e.g. if BPSK is used or if other ambiguities are detected by other circuits, such as a Viterbi decoder or a frame synchronizer) and sometimes it isn't.

Generally speaking, a *differential coding* applies to *symbols* (these are not necessary the same symbols as used in the modulator). To resolve *180° ambiguity* only, bits are used as these symbols. When dealing with *90° ambiguity*, pairs of bits are used, and triplets of bits are used to resolve *45° ambiguity* (e.g. in 8PSK).

A *differential encoder* provides the (1) operation, a *differential decoder* - the (2) operation.

Both differential encoder and differential decoder are discrete linear time-invariant systems. The former is recursive and IIR, the latter is non-recursive and thus FIR. They can be analyzed as digital filters.

A *differential encoder* is similar to an analog integrator. It has an impulse response

$$h(k) = \begin{cases} 1, & \text{if } k \geq 0 \\ 0, & \text{if } k < 0 \end{cases}$$

and a transfer function

$$H(z) = \frac{1}{1 - z^{-1}}.$$

A *differential decoder* is thus similar to an analog differentiator, its impulse response being

$$h(k) = \begin{cases} 1, & \text{if } k = 0 \\ -1, & \text{if } k = 1 \\ 0, & \text{otherwise} \end{cases}$$

and its transfer function

$$H(z) = 1 - z^{-1}.$$

Note that in binary (modulo-2) arithmetic, addition and subtraction (and positive and negative numbers) are equivalent.

Generalized differential coding

Using the relation $y_{i-1} \oplus x_i = y_i$ is not the only way of carrying out differential encoding. More generally, it can be any function $u = F(y,x)$ provided that an equation $u_0 = F(y_0,x)$ has one and only one solution for any y_0 and u_0.

Applications

Differential coding is widely used in satellite and radio relay communications together with PSK and QAM modulations.

Drawbacks

Differential coding has one significant drawback: it leads to error multiplication. That is, if one symbol such as y_i was received incorrectly, two incorrect symbols x_i and x_{i+1} would be at the differential decoder's output, see: $x_i = y_i \oplus y_{i-1}$and $x_{i+1} = y_{i+1} \oplus y_i$. This approximately doubles the BER at signal-to-noise ratios for which errors rarely occur in consecutive symbols.

Other techniques to resolve a phase ambiguity

Differential coding is not the only way to deal with a phase ambiguity. The other popular technique is to use *sync-words* for this purpose. That is, if a *frame synchronizer* detects repeated inverted sync-words, it inverts the whole stream. This method is used in DVB-S.

Analog Transmission

Analog transmission is a transmission method of conveying voice, data, image, signal or video information using a continuous signal which varies in amplitude, phase, or some other property in proportion to that of a variable. It could be the transfer of an analog source signal using an analog modulation method such as FM or AM, or no modulation at all.

Some textbooks also consider passband data transmission using a digital modulation methods such as ASK, PSK and QAM, i.e. a sinewave modulated by a digital bit-stream, as analog transmission and as an analog signal. Others define that as digital transmission and as a digital signal. Baseband data transmission using line codes, resulting in a pulse train, are always considered as digital transmission, although the source signal may be a digitized analog signal.

Modes of transmission

Analog transmission can be conveyed in many different fashions:

- twisted-pair or coax cable
- fiber-optic cable
- Via air
- Via water

There are two basic kinds of analog transmission, both based on how they modulate data to combine an input signal with a carrier signal. Usually, this carrier signal is a specific frequency, and data is transmitted through its variations. The two techniques are amplitude modulation (AM), which varies the amplitude of the carrier signal, and frequency modulation (FM), which modulates the frequency of the carrier.

Types of analog transmissions

Most analog transmissions fall into one of several categories. Until recently, most telephony and voice communication was primarily analog in nature, as was most television and radio transmission. Early telecommunication devices utilized analog-to-digital conversion devices called modulator/demodulators, or modems, to convert analog data to digital data and back.

Benefits and drawbacks

Analog transmission is still very popular, in particular for shorter distances, due to significantly lower costs and complex multiplexing and timing equipment is unnecessary, and in small "short-haul" systems that simply do not need multiplexed digital transmission.

However, in situations where a signal often has high signal-to-noise ratio and cannot achieve source linearity, or in long distance, high output systems, analog is unattractive due to attenuation problems. Furthermore, as digital techniques continue to be refined, analog systems are increasingly becoming legacy equipment.

Recently, some nations, such as the Netherlands, have completely ceased analog transmissions on certain media, such as television, for the purposes of the government saving money.

Chapter 9

Bit Error Rate

In digital transmission, the number of **bit errors** is the number of received bits of a data stream over a communication channel that have been altered due to noise, interference, distortion or bit synchronization errors.

The **bit error rate** or **bit error ratio** (**BER**) is the number of bit errors divided by the total number of transferred bits during a studied time interval. BER is a unitless performance measure, often expressed as a percentage.

The **bit error probability** p_e is the expectation value of the BER. The BER can be considered as an approximate estimate of the bit error probability. This estimate is accurate for a long time interval and a high number of bit errors.

Example

As an example, assume this transmitted bit sequence:

0 1 1 0 0 0 1 0 1 1,

and the following received bit sequence:

0 0 1 0 1 0 1 0 0 1,

The number of bit errors (the underlined bits) is in this case 3. The BER is 3 incorrect bits divided by 10 transferred bits, resulting in a BER of 0.3 or 30%.

Packet error rate

The **packet error rate** (PER) (or symbol or block error rate) is the number of incorrectly transferred data packets (etc.) divided by the number of transferred packets. A packet is assumed to be incorrect if at least one bit is incorrect. The expectation value of the PER is denoted **packet error probability** p_p, which for a data packet length of N bits can be expressed as:

$p_p = 1 - (1 - p_e)^N,$

assuming that the bit errors are independent of each other. For small bit error probabilities, this is approximately:

$p_p \approx p_e N$

Factors affecting the BER

In a communication system, the receiver side BER may be affected by transmission channel noise, interference, distortion, bit synchronization problems, attenuation, wireless multipath fading, etc.

The BER may be improved by choosing a strong signal strength (unless this causes cross-talk and more bit errors), by choosing a slow and robust modulation scheme or line coding scheme, and by applying channel coding schemes such as redundant forward error correction codes.

The *transmission BER* is the number of detected bits that are incorrect before error correction, divided by the total number of transferred bits (including redundant error codes). The *information BER*, approximately equal to the **decoding error probability**, is the number of decoded bits that remain incorrect after the error correction, divided by the total number of decoded bits (the useful information). Normally the transmission BER is larger than the information BER. The information BER is affected by the strength of the forward error correction code.

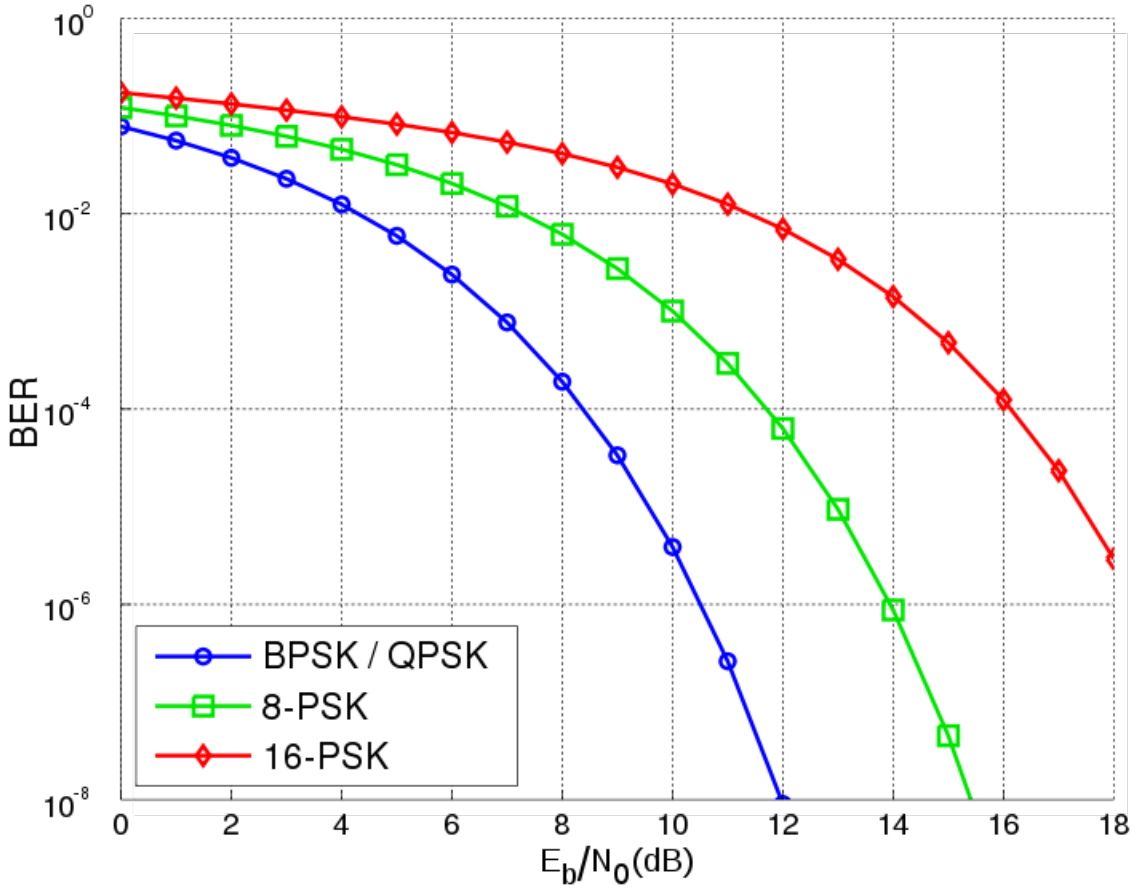
Analysis of the BER

The BER may be analyzed using stochastic computer simulations. If a simple transmission channel model and data source model is assumed, the BER may also be calculated analytically. An example of such a data source model is the Bernoulli source.

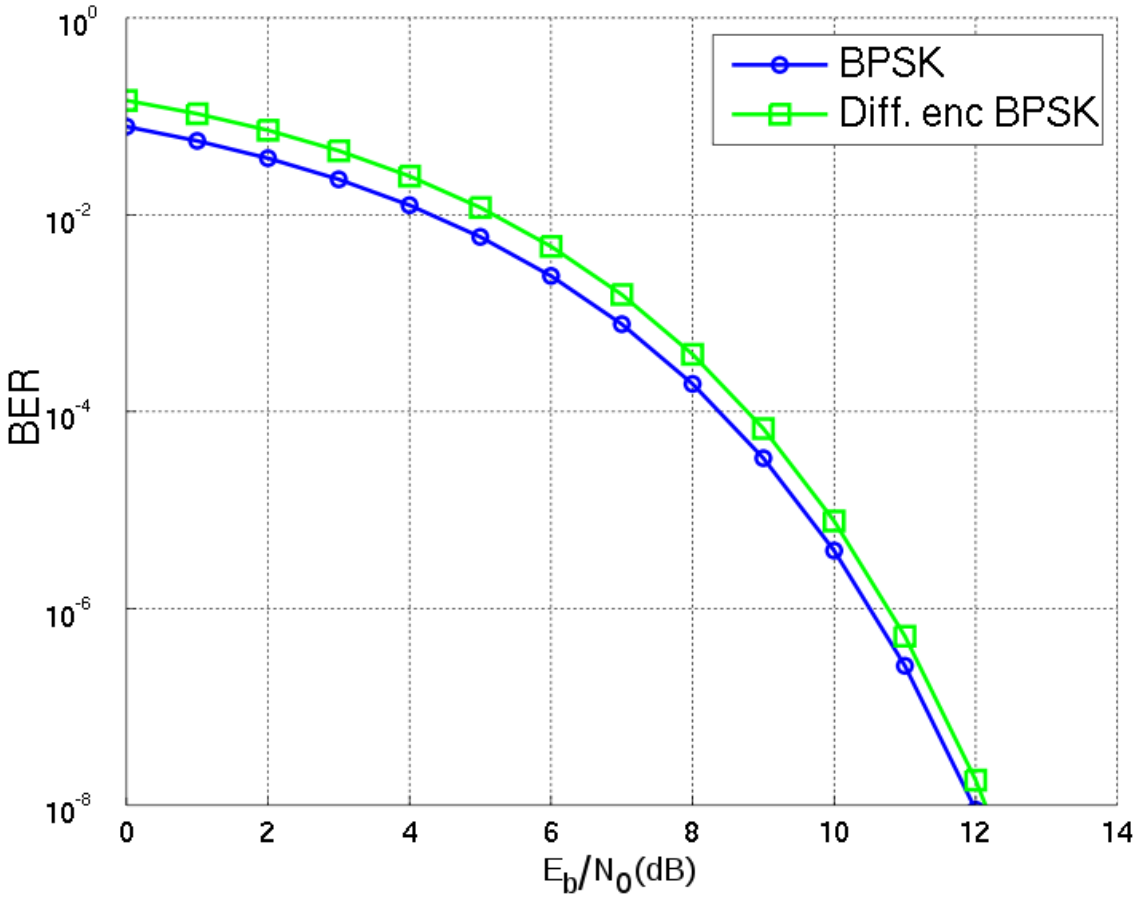
Examples of such simple channel models are:

- Binary symmetric channel (used in analysis of decoding error probability in case of non-bursty bit errors on the transmission channel)
- Additive white gaussian noise (AWGN) channel without fading.

A worst case scenario is a completely random channel, where noise totally dominates over the useful signal. This results in a transmission BER of 50% (provided that a Bernoulli binary data source and a binary symmetrical channel are assumed, see below).



Bit-error rate curves for BPSK, QPSK, 8-PSK and 16-PSK, AWGN channel.



BER comparison between BPSK and differentially-encoded BPSK with gray-coding operating in white noise.

In a noisy channel, the BER is often expressed as a function of the normalized carrier-to-noise ratio measure denoted Eb/N0, (energy per bit to noise power spectral density ratio), or Es/N0 (energy per modulation symbol to noise spectral density).

For example, in the case of QPSK modulation and AWGN channel, the BER as function of the Eb/N0 is given by: *BER* = 1 / 2*erfc*(*Eb* / *N*0 / *sqrt*(2)).

People usually plot the BER curves to describe the functionality of a digital communication system. In optical communication, BER(dB) vs. Received Power(dBm) is usually used; while in wireless communication, BER(dB) vs. SNR(dB) is used.

Measuring the bit error ratio helps people choose the appropriate forward error correction codes. Since most such codes correct only bit-flips, but not bit-insertions or bit-deletions, the Hamming distance metric is the appropriate way to measure the number of bit errors. Many FEC coders also continuously measure the current BER.

A more general way of measuring the number of bit errors is the Levenshtein distance. The Levenshtein distance measurement is more appropriate for measuring raw channel

performance before frame synchronization, and when using error correction codes designed to correct bit-insertions and bit-deletions, such as Marker Codes and Watermark Codes.

Mathematical draft

The BER is the likelihood of a bit misinterpretation due to electrical noise $w(t)$. Considering a bipolar NRZ transmission, we have

$x_1(t) = A + w(t)$ for a "1" and $x_0(t) = -A + w(t)$ for a "0". Each of $x_1(t)$ and $x_0(t)$ has a period of T.

Knowing that the noise has a bilateral spectral density $\frac{N_0}{2}$,

$x_1(t)$ is $\mathcal{N}\left(A, \frac{N_0}{2T}\right)$

and $x_0(t)$ is $\mathcal{N}\left(-A, \frac{N_0}{2T}\right)$.

Returning to BER, we have the likelihood of a bit misinterpretation $p_e = p(0 \mid 1)p_1 + p(1 \mid 0)p_0$.

$$p(1|0) = 0.5\,\mathrm{erfc}\left(\frac{A+\lambda}{\sqrt{N_o/T}}\right) \text{ and } p(0|1) = 0.5\,\mathrm{erfc}\left(\frac{A-\lambda}{\sqrt{N_o/T}}\right)$$

where λ is the threshold of decision, set to 0 when $p_1 = p_0 = 0.5$.

We can use the average energy of the signal $E = A^2T$ to find the final expression :

$$p_e = 0.5\,\mathrm{erfc}\left(\sqrt{\frac{E}{N_o}}\right).$$

±§

Bit error rate test

BERT or **bit error rate test** is a testing method for digital communication circuits that uses predetermined stress patterns consisting of a sequence of logical ones and zeros generated by a pseudorandom binary sequencer.

A BERT typically consists of a test pattern generator and a receiver that can be set to the same pattern. They can be used in pairs, with one at either end of a transmission link, or singularly at one end with a loopback at the remote end. BERTs are typically stand-alone specialised instruments, but can be Personal Computer based. In use, the number of errors, if any, are counted and presented as a ratio such as 1 in 1,000,000, or 1 in 1e06.

Common types of BERT stress patterns

- **PRBS** (Pseudo Random binary sequence) – A pseudorandom binary sequencer of N Bits. These pattern sequences are used to measure jitter and eye mask of TX-Data in electrical and optical data links.
- **QRSS** (Quasi Random Signal Source) – A pseudorandom binary sequencer which generates every combination of a 20-bit word, repeats every 1,048,575 bits, and suppresses consecutive zeros to no more than 14. It contains high-density sequences, low-density sequences, and sequences that change from low to high and vice versa. This pattern is also the standard pattern used to measure jitter.
- **3 in 24** – Pattern contains the longest string of consecutive zeros (15) with the lowest ones density (12.5%). This pattern simultaneously stresses minimum ones density and the maximum number of consecutive zeros. The D4 frame format of 3 in 24 may cause a D4 Yellow Alarm for frame circuits depending on the alignment of one bits to a frame.
- **1:7** – Also referred to as "1 in 8". It has only a single one in an 8-bit repeating sequence. This pattern stresses the minimum ones density of 12.5% and should be used when testing facilities set for B8ZS coding as the 3 in 24 pattern increases to 29.5% when converted to B8ZS.
- **Min/Max** – Pattern rapid sequence changes from low density to high density. Most useful when stressing the repeater's ALBO feature.
- **All Ones (or Mark)** – A pattern composed of ones only. This pattern causes the repeater to consume the maximum amount of power. If DC to the repeater is regulated properly, the repeater will have no trouble transmitting the long ones sequence. This pattern should be used when measuring span power regulation. An unframed all ones pattern is used to indicate an AIS (also known as a Blue Alarm).
- **All Zeros** – A pattern composed of zeros only. It is effective in finding equipment misoptioned for AMI, such as fiber/radio multiplex low-speed inputs.
- **Alternating 0s and 1s** - A pattern composed of alternating ones and zeroes.
- **2 in 8** – Pattern contains a maximum of four consecutive zeros. It will not invoke a B8ZS sequence because eight consecutive zeros are required to cause a B8ZS substitution. The pattern is effective in finding equipment misoptioned for B8ZS.
- **Bridgetap** - Bridge taps within a span can be detected by employing a number of test patterns with a variety of ones and zeros densities. This test generates 21 test patterns and runs for 15 minutes. If a signal error occurs, the span may have one or more bridge taps. This pattern is only effective for T1 spans that transmit the signal raw. Modulation used in HDSL spans negates the Bridgetap patterns' ability to uncover bridge taps.

- **Multipat** - This test generates 5 commonly used test patterns to allow DS1 span testing without having to select each test pattern individually. Patterns are: All Ones, 1:7, 2 in 8, 3 in 24, and QRSS.
- **T1-DALY** and **55 OCTET** - Each of these patterns contain fifty-five (55), eight bit octets of data in a sequence that changes rapidly between low and high density. These patterns are used primarily to stress the ALBO and equalizer circuitry but they will also stress timing recovery. 55 OCTET has fifteen (15) consecutive zeroes and can only be used unframed without violating ones density requirements. For framed signals, the T1-DALY pattern should be used. Both patterns will force a B8ZS code in circuits optioned for B8ZS.

Bit error rate tester

A **bit error rate tester** (BERT), also known as a **bit error ratio tester** or **bit error rate test solution** (BERTs) is electronic test equipment used to test the quality of signal transmission of single components or complete systems.

The main building blocks of a BERT are:

- Pattern Generator, which transmits a defined test pattern to the DUT or test system
- Error detector connected to the DUT or test system, to count the errors generated by the DUT or test system
- Clock signal generator to synchronize the pattern generator and the error detector
- Digital communication analyser is optional to display the transmitted or received signal
- Electrical-optical converter and optical-electrical converter for testing optical communication signals.

Chapter 10

Bit Rate

In telecommunications and computing, **bitrate** (sometimes written **bit rate**, **data rate** or as a variable R or f_b) is the number of bits that are conveyed or processed per unit of time.

The bit rate is quantified using the bits per second (**bit/s** or **bps**) unit, often in conjunction with an SI prefix such as kilo- (kbit/s or kbps), mega- (Mbit/s or Mbps), giga- (Gbit/s or Gbps) or tera- (Tbit/s or Tbps). Note that, unlike many other computer-related units, 1 kbit/s is traditionally defined as 1,000 bit/s, not 1,024 bit/s, etc., also before 1999 when SI prefixes were introduced for units of information in the standard IEC 60027-2.

The formal abbreviation for "bits per second" is "bit/s" (not "bits/s", see writing style for SI units). In less formal contexts the abbreviations "b/s" or "bps" are often used, though this risks confusion with "bytes per second" ("B/s", "Bps"). 1 Byte/s (Bps or B/s) corresponds to 8 bit/s (bps or b/s).

Bit rates at various protocol layers

Physical layer gross bit rate

In digital communication systems, the **gross bitrate**, **raw bitrate**, data signaling rate or **uncoded transmission rate** is the total number of physically transferred bits per second over a communication link, including useful data as well as protocol overhead. The gross bit rate is related to, but should not be confused with, the symbol rate in baud, symbols/s or pulses/s. Gross bit rate can be used interchangeably with "baud" *only* when each modulation transition of a data transmission system carries exactly one bit of data; something not true for modern modem modulation systems and modern LANs, for example.

For most line codes and modulation methods:

$$\text{Symbol rate} \leq \text{Gross bit rate}$$

More specifically, a line code representing the data using pulse-amplitude modulation with 2^N different voltage levels, or a digital modulation method using 2^N different symbols, for example 2^N amplitudes, phases or frequencies, can transfer N bit/symbol, or N bit/pulse. This results in:

$$\text{Gross bit rate} = \text{Symbol rate} \cdot N$$

The exception from the above is some self-synchronizing line codes, for example Manchester coding and return-to-zero (RTZ) coding, where each bit is represented by two pulses (signal states), resulting in:

$$\text{Gross bit rate} = \text{Symbol rate}/2$$

A theoretical upper bound for the symbol rate in baud, symbols/s or pulses/s for a certain analog bandwidth in hertz is given by the Nyquist law:

$$\text{Symbol rate} \leq \text{Nyquist rate} = 2 \cdot \text{bandwidth}$$

In practice this upper bound can only be approached for line coding schemes (or baseband transmission) and for so-called vestigal sideband digital modulation. Most other digital carrier-modulated schemes (or passband transmission schemes), for example ASK, PSK and QAM, can be characterized as double sideband modulation, resulting in the following approximative relation:

$$\text{Symbol rate} \leq \text{Bandwidth}$$

Physical layer net bit rate

The physical layer **net bitrate**, **peak bitrate**, **useful bit rate**, **information rate**, **payload rate**, **coded transmission rate**, **effective data rate** or wire speed (informal language) of a digital communication link is the capacity excluding the physical layer protocol overhead, for example time division multiplex (TDM) framing bits, redundant forward error correction (FEC) codes, equalizer training symbols and other channel coding. Error-correcting codes are common especially in wireless communication systems and broadband modem standards. The relationship between the gross bit rate and net bit rate is affected by the FEC code rate according to the following.

$$\text{Net bit rate} \leq \text{Gross bit rate} \cdot \text{code rate}$$

Some operational systems indicate the "**connection speed**" (informal language) of a network access technology or communication device. The connection speed of a technology that involves forward error correction typically refers to the physical layer *net bit rate* in accordance with the above definition.

For example, the net bitrate (and thus the "connection speed") of a IEEE 802.11a wireless network is the net bit rate of between 6 and 54 Mbit/s, while the gross bit rate is between 12 and 72 Mbit/s inclusive of error-correcting codes. The net bit rate of ISDN Basic Rate Interface (2 B-channels + 1 D-channel) of 64+64+16 = 144 kbit/s also refers to the payload data rates, while the signalling rate is 160 kbit/s.

The net bit rate of the Ethernet 100Base-TX physical layer standard is 100 Mbit/s, while the gross bitrate is 125 Mbit/second, due to the 4B5B (four bit over five bit) encoding. In this case, the gross bit rate is equal to the symbol rate or pulse rate of 125 Mbaud, due to the NRZI line code.

In communications technologies without forward error correction and other physical layer protocol overhead, there is no distinction between gross bit rate and physical layer net bit rate. For example, the net as well as gross bit rate of Ethernet 10Base-T is 10 Mbit/s. Due to the Manchester line code, each bit is represented by two pulses, resulting in a pulse rate of 20 Mbaud.

The "connection speed" of a V.92 voiceband modem typically refers to the gross bit rate, since there is no additional error-correction code. It can be up to 56,000 bit/s downstreams and 48,000 bit/s upstreams. A lower bit rate may be chosen during the connection establishment phase due to adaptive modulation - slower but more robust modulation schemes are chosen in case of poor signal-to-noise ratio. Due to data compression, the actual data transmission rate or throughput (see below) may be higher.

The channel capacity, also known as the Shannon capacity, is a theoretical upper bound for the maximum net bitrate, exclusive of forward error correction coding, that is possible without bit errors for a certain physical analog node-to-node communication link.

$$\text{Net bit rate} \leq \text{Channel capacity}$$

The channel capacity is proportional to the analog bandwidth in hertz. This proportionality is called Hartley's law. Consequently the net bit rate is sometimes called digital bandwidth capacity in bit/s.

Note that the term **line rate** in some textbooks is defined as gross bit rate, in others as net bit rate.

Network throughput

The term *throughput*, essentially the same thing as **digital bandwidth consumption**, denotes the achieved average useful bit rate in a computer network over a logical or physical communication link or through a network node, typically measured at a reference point above the datalink layer. This implies that the throughput often excludes data link layer protocol overhead. The throughput is affected by the traffic load from the data source in question, as well as from other sources sharing the same network resources.

Goodput (data transfer rate)

Goodput or **data transfer rate** refers to the achieved average net bit rate that is delivered to the application layer, exclusive of all protocol overhead, data packets retransmissions, etc. For example, in the case of file transfer, the goodput corresponds to the achieved **file transfer rate**. The file transfer rate in bit/s can be calculated as the file size (in bytes), divided by the file transfer time (in seconds), and multiplied by eight.

As an example, the goodput or data transfer rate of a V.92 voiceband modem is affected by the modem physical layer and data link layer protocols. It is sometimes higher than the physical layer data rate due to V.44 data compression, and sometimes lower due to bit-errors and automatic repeat request retransmissions.

If no data compression is provided by the network equipment or protocols, we have the following relation:

$$\text{Goodput} \leq \text{Throughput} \leq \text{Maximum throughput} \leq \text{Net bit rate}$$

for a certain communication path.

Multimedia encoding bit rate

In digital multimedia, *bit rate* often refers to the number of bits used per unit of playback time to represent a continuous medium such as audio or video after source coding (data compression). The encoding bit rate of a multimedia file is the size of a multimedia file in bytes divided by the playback time of the recording (in seconds), multiplied by eight.

In case of realtime streaming multimedia, the encoding bit rate is the goodput that is required to avoid interrupts. For streaming multimedia without interrupts, we have the following relationship:

$$\text{Encoding bit rate} = \text{Required goodput}$$

The term average bitrate is used in case of variable bitrate multimedia source coding schemes.

A theoretical lower bound for the encoding bit rate for lossless data compression is the source information rate, also known as the *entropy rate*.

$$\text{Entropy rate} \leq \text{Multimedia bit rate}$$

Prefixes

When quantifying large bit rates, SI prefixes (also known as Metric prefixes or Decimal prefixes) are used, thus:

1,000 bit/s rate = 1 kbit/s (one kilobit or one thousand bits per second)
1,000,000 bit/s rate = 1 Mbit/s (one megabit or one million bits per second)
1,000,000,000 bit/s rate = 1 Gbit/s (one gigabit or one billion bits per second)

Binary prefixes have almost never been used for bitrates, although they may occasionally be seen when data rates are expressed in bytes per second (e.g. 1 kByte/s or kBps is sometimes interpreted as 1000 Byte/s, sometimes as 1024 Byte/s). A 1999 IEC standard (IEC 60027-2) specifies different abbreviations for Binary and Decimal (SI) prefixes (e.g. 1 kiB/s = 1024 Byte/s = 8192 bit/s, and 1 MiB/s = 1024 kiB/s), but these are still not very common in the literature, and therefore sometimes it is necessary to seek clarification of the units used in a particular context.

Progress trends

These are examples of physical layer net bit rates in proposed communication standard interfaces and devices:

WAN modems	Ethernet LAN	WiFi WLAN	Mobile data
• 1972: Acoustic coupler 300 baud • 1977: 1200 baud Vadic and Bell 212A • 1986: ISDN introduced with two 64 kbit/s channels (160 kbit/s gross bit rate) • 1990: v.32bis modems: 2400 / 4800 / 9600 / 19200 bit/s • 1994: v.34 modems	• 1972: IEEE 802.3 Ethernet 2.94 Mbit/s • 1985: 10b2 10 Mbit/s coax thinwire • 1990: 10bT 10 Mbit/s • 1995: 100bT 100 Mbit/s (125 Mbit/s gross bit rate) • 1999: 1000bT (Gigabit) 1 Gbit/s	WiFi WLANs • 1997: 802.11 2 Mbit/s • 1999: 802.11b 11 Mbit/s • 1999: 802.11a 54 Mbit/s (72 Mbit/s gross bit rate) • 2003: 802.11g 54 Mbit/s (72 Mbit/s gross bit rate) • 2005: 802.11g (proprietary) 108 Mbit/s	• 1G: o 1981: NMT 1200 bit/s • 2G: o 1991: GSM CSD and D-AMPS 14.4 kbit/s o 2003: GSM EDGE 296 kbit/s down, 118.4 kbit/s up • 3G: o 2001: UMTS-FDD (WCDMA) 384 kbit/s o 2007: UMTS HSDPA 14.4 Mbit/s o 2008: UMTS HSPA 14.4 Mbit/s down, 5.76 Mbit/s up o 2009: HSPA+

with 28.8 kbit/s
- 1995: v.90 modems with 56 kbit/s downstreams, 33.6 kbit/s upstreams
- 1999: v.92 modems with 56 kbit/s downstreams, 48 kbit/s upstreams
- 1998: ADSL up to 8 Mbit/s,
- 2003: ADSL2 up to 12 Mbit/s
- 2005: ADSL2+ up to 24 Mbit/s (1.25 Gbit/s gross bit rate)
- 2003: 10GBASE 10 Gbit/s
- 2007: 802.11n 600 Mbit/s (Without MIMO) 28 Mbit/s downstreams (56 Mbit/s with 2x2 MIMO), 22 Mbit/s upstreams
  - 2010: CDMA2000 EV-DO Rev. B 14.7 Mbit/s downstreams
- Pre-4G:
  - 2007: Mobile WiMAX (IEEE 802.16e) 144 Mbit/s down, 35 Mbit/s up.
  - 2009: LTE 100 Mbit/s downstreams (360 Mbit/s with MIMO 2x2), 50 Mbit/s upstreams

Bitrates in multimedia

In digital multimedia, bitrate represents the amount of information, or detail, that is stored per unit of time of a recording. The bitrate depends on several factors:

- The original material may be sampled at different frequencies
- The samples may use different numbers of bits
- The data may be encoded by different schemes
- The information may be digitally compressed by different algorithms or to different degrees

Generally, choices are made about the above factors in order to achieve the desired trade-off between minimizing the bitrate and maximizing the quality of the material when it is played.

If lossy data compression is used on audio or visual data, differences from the original signal will be introduced; if the compression is substantial, or lossy data is decompressed and recompressed, this may become noticeable in the form of compression artifacts. Whether these affect the perceived quality, and if so how much, depends on the compression scheme, encoder power, the characteristics of the input data, the listener's perceptions, the listener's familiarity with artifacts, and the listening or viewing environment.

The bitrates in this section are approximately the *minimum* that the *average* listener in a typical listening or viewing environment, when using the best available compression, would perceive as not significantly worse than the reference standard:

Audio

MP3

- 32 kbit/s – MW (AM) quality
- 96 kbit/s – FM quality - This is questionable since FM broadcast is transmitted in analog 30 Hz-15khz. Similarly one cannot compare directly an LP record to CD using kbit/s.
- 100–160 kbit/s – Standard Bitrate quality; difference can sometimes be obvious (e.g. lack of low frequency quality and high frequency "swashy" effects)
- 192 kbits/s is the highest level supported by most MP3 encoders when ripping from a Compact Disc.
- 224–320 kbit/s – VBR to highest MP3 quality.

Other audio

- 800 bit/s – minimum necessary for recognizable speech (using special-purpose FS-1015 speech codecs)
- 2.15 kbit/s - minimum bitrate available through the open-source Speex codec
- 8 kbit/s – telephone quality (using speech codecs)
- 32-500 kbit/s -- lossy audio as used in Ogg Vorbis
- 256 kbit/s - Digital Audio Broadcasting (DAB) MP2 bit rate required to achieve a high quality signal
- 400 kbit/s–1,411kbit/s – lossless audio as used in formats such as Free Lossless Audio Codec, WavPack or Monkey's Audio to compress CD audio
- 1,411.2 kbit/s – Linear PCM sound format of Compact Disc Digital Audio
- 5,644.8 kbit/s – DSD (A trademarked implementation of PDM) sound format of Super Audio CD

Video

- 16 kbit/s – videophone quality (minimum necessary for a consumer-acceptable "talking head" picture using various video compression schemes)

- 128 – 384 kbit/s – business-oriented videoconferencing quality using video compression
- 1.15 Mbit/s max – VCD quality (using MPEG1 compression)
- 3.5 Mbit/s typ - Standard-definition television quality (with bit-rate reduction from MPEG-2 compression)
- 9.8 Mbit/s max – DVD (using MPEG2 compression)
- 8 to 15 Mbit/s typ – HDTV quality (with bit-rate reduction from MPEG-4 AVC compression)
- 19 Mbit/s approximate - HDV 720p (using MPEG2 compression)
- 24 Mbit/s max - AVCHD (using MPEG4 AVC compression)
- 25 Mbit/s approximate - HDV 1080i (using MPEG2 compression)
- 29.4 Mbit/s max – HD DVD
- 40 Mbit/s max – Blu-ray Disc (using MPEG2, AVC or VC-1 compression)

Chapter 11

Variable Bitrate & Viterbi Decoder

Variable Bitrate

Variable bitrate (**VBR**) is a term used in telecommunications and computing that relates to the bitrate used in sound or video encoding. As opposed to constant bitrate (CBR), VBR files vary the amount of output data per time segment. VBR allows a higher bitrate (and therefore more storage space) to be allocated to the more complex segments of media files while less space is allocated to less complex segments. The average of these rates can be calculated to produce an average bitrate for the file.

MP3, WMA, Vorbis, and AAC audio files can optionally be encoded in VBR. Variable bit rate encoding is also commonly used on MPEG-2 video, MPEG-4 Part 2 video (Xvid, DivX, etc), MPEG-4 Part 10/H.264 video, Theora, Dirac and other video compression formats.

Advantages and disadvantages of VBR

The advantages of VBR are that it produces a better quality-to-space ratio compared to a CBR file of the same data. The bits available are used more flexibly to encode the sound or video data more accurately, with fewer bits used in less demanding passages and more bits used in difficult-to-encode passages.

The disadvantages are that it may take more time to encode, as the process is more complex, and that some hardware might not be compatible with VBR files. VBR may also pose problems when streaming over a dial-up or broadband Internet connection, because the instantaneous bitrate, if unconstrained, may rise above the maximum data transfer speed of the web connection. These problems can be avoided by specifying a maximum instantaneous bitrate during the encoding process.

Also, encryption of VBR-encoded speech (or other signals including video) gives only limited privacy, as the patterns of variation of the bit rate allow recognition of many words and phrases.

In the past, many hardware and software players could not decode variable bitrate files properly, partly because the various VBR encoders used were not well developed. This resulted in common use of CBR over VBR for the sake of compatibility. As of December 2006, devices that support only CBR encoded files are largely obsolete, as the vast majority of modern portable music devices and software support VBR encoded files.

Support for VBR in AAC and MP3 files is found in most modern digital audio players, including those released by Microsoft, Apple Inc., Creative Technology, and SanDisk. Early VBR algorithms occasionally introduced audible artifacts when encoding monotone or minimal tones (for example audiobooks and acoustic music). These artifacts often mimicked a "digital chirp" during the quiet portions of the song or when there was only speaking. As VBR encoding algorithms have improved, these problems have been resolved in subsequent generations of the VBR standard.

Methods of VBR encoding

Note that the choice of a variable bitrate (VBR) method only affects the encoding process. Decoding a VBR stream is performed identically in all cases, regardless of how the encoder chooses to allocate bits.

Multi-pass encoding and single-pass encoding

VBR is created using the so-called *single-pass encoding* or *multi-pass encoding*. Single-pass encoding analyzes and encodes the data "on the fly" and it is also used in the constant bitrate encoding. Single-pass encoding is used when the encoding speed is most important - e.g. for real-time encoding. Single-pass VBR encoding is usually controlled by the *fixed quality* setting or by the *bitrate range* (minimum and maximum allowed bitrate) or by the *average bitrate* setting. Multi-pass encoding is used when the encoding quality is most important. Multi-pass encoding cannot be used in real-time encoding, live broadcast or live streaming. Multi-pass encoding takes much longer than single-pass encoding, because every pass means one pass through the input data (usually through the whole input file). Multi-pass encoding is used only for VBR encoding, because CBR encoding doesn't offer any flexibility to change the bitrate. The most common multi-pass encoding is two-pass encoding. In the first pass of two-pass encoding, the input data are being analyzed and the result is stored in a log file. In the second pass, the collected data from the first pass are used to achieve the best encoding quality. In a video encoding, two-pass encoding is usually controlled by the average bitrate setting or by the bitrate range setting (minimal and maximal allowed bitrate) or by the target video file size setting.

Fixed quality

One means of VBR encoding is *fixed quantizer* or *fixed quality* encoding. It is usually single-pass encoding. The user specifies a given subjective quality value, and the encoder allocates bits as needed to achieve the given level of quality. This ensures the output stream will have consistent quality throughout. A quality level usually has an associated bitrate range. The disadvantage of this encoding method is that the average bitrate (and hence file size) will not be known ahead of time, and achieving a certain average bitrate requires trial and error. This is typically more of a concern for video than for audio, since file sizes are much larger and encoding can take much longer.

Bitrate range

This VBR encoding method allows the user to specify a bitrate range - a minimum and/or maximum allowed bitrate. Some encoders extend this method with an average bitrate. The minimum and maximum allowed bitrate set bounds in which the bitrate may vary. The disadvantage of this method is that the average bitrate (and hence file size) will not be known ahead of time. The bitrate range is also used in some fixed quality encoding methods, but usually without permission to change a particular bitrate.

Average bitrate

Average bitrate (ABR) encoding may be used to ensure the output stream achieves a predictable long-term average bitrate. This is typically implemented using multi-pass encoding, where one or more initial passes are used to collect data on the stream, and a final pass uses that data to achieve uniform quality at the specified average bitrate.

Alternatively, periodic averaging may be used, either by performing ABR on smaller chunks of the output, or by reacting to fluctuations in the ABR by increasing or reducing the overall quality. These can achieve ABR in a single pass, but do not produce the same degree of uniformity as multi-pass ABR. Some encoders use "ABR encoding" and "multi-pass encoding" to refer to single- and multi-pass ABR encoding respectively.

Some encoders also allow the user to specify a maximum allowed bitrate or maximum quality value. This is sometimes called **Constrained Variable Bitrate (CVBR)**, and is typically applied to ABR algorithms.

The disadvantage of single pass ABR encoding (with or without CVBR) is the opposite of fixed quantizer VBR — the size of the output is known ahead of time, but the resulting quality is unknown, although still better than CBR. Specifying a higher average or maximum may simply make the file bigger with no discernible quality effect, and an increased maximum bitrate may introduce stutter when streaming the file. However, reducing these criteria too low will eventually lead to quite drastic losses in quality. The effect on video is typically an increased blockiness, because the frames are no longer being fully detailed in their rendering.

The multi-pass ABR encoding is more similar to fixed quantizer VBR, because a higher average will really increase the quality.

There are no ideal "one-size-fits-all" settings for ABR in video encoding. For low resolution (320 or 640 lines) video encoded with MPEG-1 or MPEG-2, the average bit rate can be as low as 1000 kbit/s and still achieve acceptable results. For a high resolution video such as 1080, this average may need to be 6000 kbit/s or higher. The main factor in determining a minimum video bitrate is how efficiently the video can be encoded. Using more efficient video encodings such as MPEG-4 will help promote a lower bit rate, while significant amounts of motion or white noise will require a higher bit rate to encode without visible artifacts. In the end, the user may have to use trial and error to achieve a minimum file size for a given video stream, by encoding at a given bitrate and then viewing the results.

File size

VBR encoding using the file size setting is usually multi-pass encoding. It allows the user to specify a specific target file size. In the first pass, the encoder analyzes the input file and automatically calculates possible bitrate range and/or average bitrate. In the last pass, the encoder distributes the available bits among the entire video to achieve uniform quality.

Viterbi Decoder

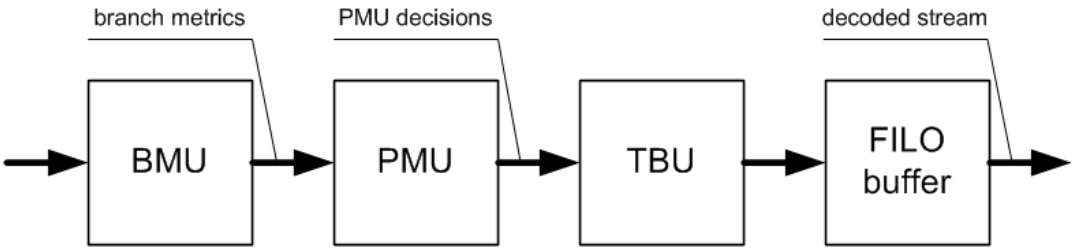
A **Viterbi decoder** uses the Viterbi algorithm for decoding a bitstream that has been encoded using forward error correction based on a convolutional code.

There are other algorithms for decoding a convolutionally encoded stream (for example, the Fano algorithm). The Viterbi algorithm is the most resource-consuming, but it does the maximum likelihood decoding. It is most often used for decoding convolutional codes with constraint lengths k<=10, but values up to k=15 are used in practice.

Viterbi decoding was developed by Andrew J. Viterbi and published in the paper "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", IEEE Transactions on Information Theory, Volume IT-13, pages 260-269, in April, 1967.

There are both hardware (in modems) and software implementations of a Viterbi decoder.

Hardware implementation

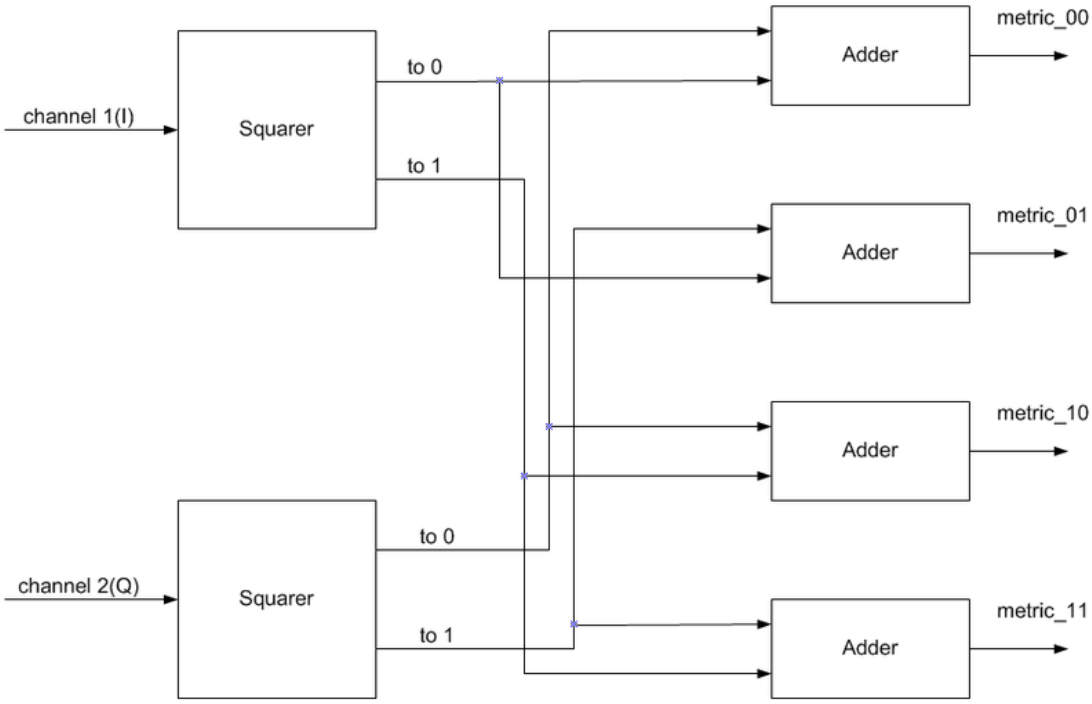


A common way to implement a hardware viterbi decoder

A hardware Viterbi decoder for basic (not perforated) code usually consists of the following major blocks:

- Branch metric unit (BMU)
- Path metric unit (PMU)
- Traceback unit (TBU)

Branch metric unit (BMU)



A sample implementation of a branch metric unit

A branch metric unit's function is to calculate *branch metrics*, which are normed distances between every possible symbol in the code alphabet, and the received symbol.

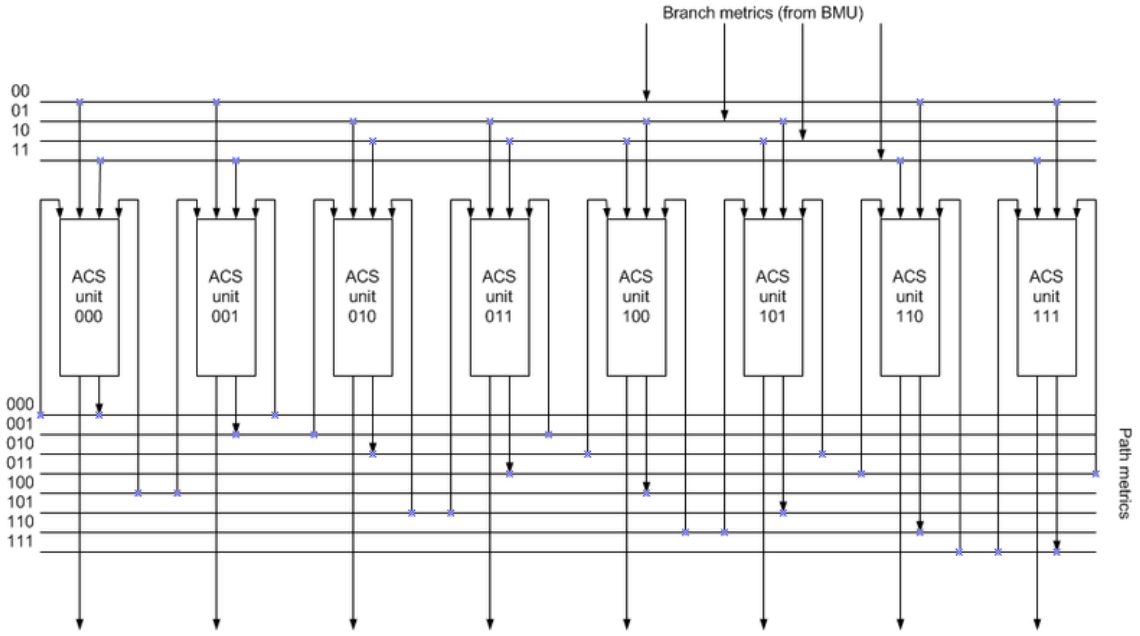
There are hard decision and soft decision Viterbi decoders. A hard decision Viterbi decoder receives a simple bitstream on its input, and a Hamming distance is used as a metric. A soft decision Viterbi decoder receives a bitstream containing information about the *reliability* of each received symbol. For instance, in a 3-bit encoding, this *reliability* information is encoded as follows:

value	meaning
000	strongest **0**
001	relatively strong **0**
010	relatively weak **0**
011	weakest **0**
100	weakest **1**
101	relatively weak **1**
110	relatively strong **1**
111	strongest **1**

Of course, it is not the only way to encode reliability data.

The *squared* Euclidean distance is used as a metric for soft decision decoders.

Path metric unit (PMU)

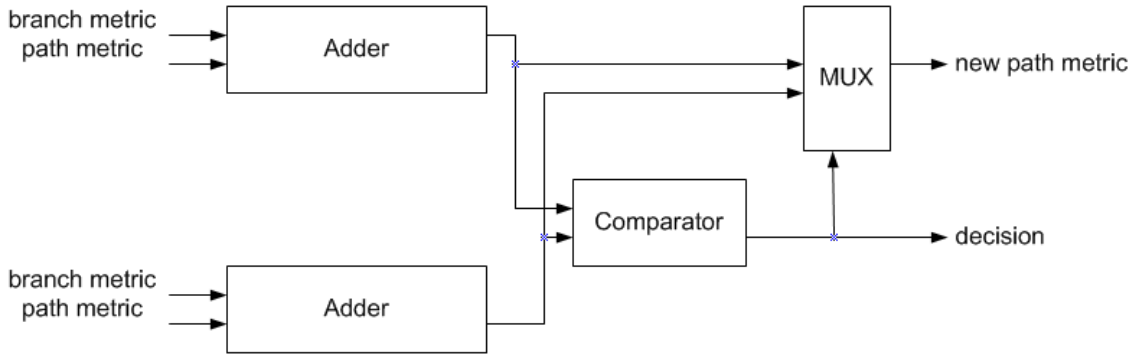


A sample implementation of a path metric unit for a specific K=4 decoder

A path metric unit summarizes branch metrics to get metrics for 2^{K-1} paths, one of which can eventually be chosen as *optimal*. Every clock it makes 2^{K-1} decisions, throwing off wittingly nonoptimal paths. The results of these decisions are written to the memory of a traceback unit.

The core elements of a PMU are *ACS (Add-Compare-Select)* units. The way in which they are connected between themselves is defined by a specific code's trellis diagram.

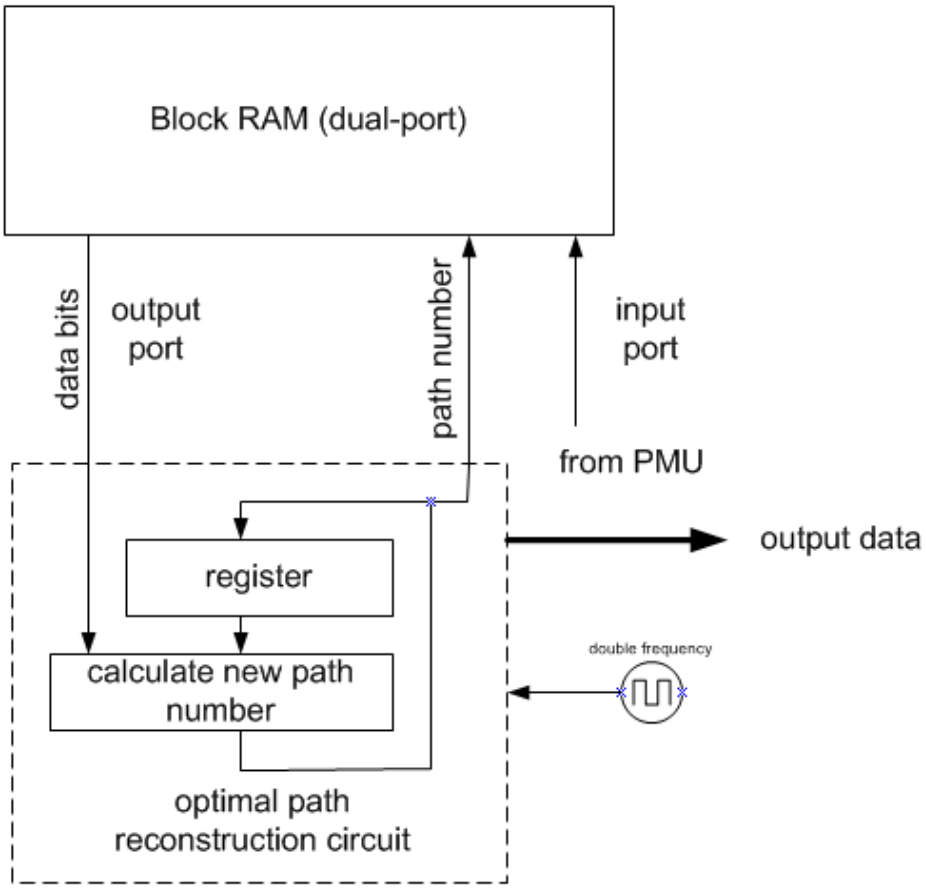
Since branch metrics are always ≥ 0, there must be an additional circuit preventing metric counters from overflow (it isn't shown on the image). An alternate method that eliminates the need to monitor the path metric growth is to allow the path metrics to "roll over", to use this method it is necessary to make sure the path metric accumulators contain enough bits to prevent the "best" and "worst" values from coming within $2^{(n-1)}$ of each other. The compare circuit is essentially unchanged.



A sample implementation of an ACS unit

It is possible to monitor the noise level on the incoming bit stream by monitoring the rate of growth of the "best" path metric, a simpler way to do this is to monitor a single location or "state" and watch it pass "upward" through say four discrete levels within the range of the accumulator. As it passes upward through each of these thresholds, a counter is incremented that reflects the "noise" present on the incoming signal.

Traceback unit (TBU)



A sample implementation of a traceback unit

Back-trace unit restores an (almost) maximum-likelihood path from the decisions made by PMU. Since it does it in inverse direction, a viterbi decoder comprises a FILO (first-in-last-out) buffer to reconstruct a correct order.

Note that the implementation shown on the image requires double frequency. There are some tricks that eliminate this requirement.

Implementation issues

Quantization for soft decision decoding

In order to fully exploit benefits of soft decision decoding, one needs to quantize input signal properly. The optimal quantization zone width is defined by the following formula:

$$T = \sqrt{N_0/2^k},$$

where N_0 is a noise power spectral density, and k is a number of bits for soft decision.

Euclidean metric computation

The squared norm distance (l_2) distance between the received and the actual symbols in the code alphabet may be further simplified into a linear sum/difference form, which makes it less computationally intensive.

Consider a 1/2 convolutional coder, which generates 2 bits (*00*, *01*, *10* or *11*) for every input bit (*1* or *0*). These *Return-to-Zero* signals are translated into a *Non-Return-to-Zero* form shown alongside.

code alphabet	vector mapping
00	*1, 1*
01	*1, -1*
10	*-1, 1*
11	*-1, -1*

Each received symbol may be represented in vector form as $\mathbf{v_r} = \{r_0, r_1\}$, where r_0 and r_1 are soft decision values, whose magnitudes signify the *joint reliability* of the received vector, $\mathbf{v_r}$.

Every symbol in the code alphabet may, likewise, be represented by the vector $\mathbf{v_i} = \{\pm 1, \pm 1\}$.

The actual computation of the Euclidean distance metric is:

$$D = (\overrightarrow{v_r} - \overrightarrow{v_i})^2 = \overrightarrow{v_r}^2 - 2\overrightarrow{v_r}\overrightarrow{v_i} + \overrightarrow{v_i}^2$$

Each square term is a normed distance, depicting the *energy* of the symbol. For ex., the *energy* of the symbol $\mathbf{v_i} = \{\pm 1, \pm 1\}$ may be computed as

$$\overrightarrow{v_i}^2 = (\pm 1)^2 + (\pm 1)^2 = 2$$

Thus, the energy term of all symbols in the code alphabet is constant (at (*normalized*) value 2).

The *Add-Compare-Select* (*ACS*) operation compares the metric distance between the received symbol $\|\mathbf{v_r}\|$ and any 2 symbols in the code alphabet whose paths merge at a node in the corresponding trellis, $\|\mathbf{v_i}^{(0)}\|$ and $\|\mathbf{v_i}^{(1)}\|$. This is equivalent to comparing

$$D_0 = \overrightarrow{v_r}^2 - 2\overrightarrow{v_r}\overrightarrow{v_i^0} + \overrightarrow{v_i^0}^2$$

and

$$D_1 = \overrightarrow{v_r}^2 - 2\overrightarrow{v_r}\overrightarrow{v_i^1} + \overrightarrow{v_i^1}^2$$

But, from above we know that the *energy* of $\mathbf{v_i}$ is constant (equal to (normalized) value of 2), and the *energy* of $\mathbf{v_r}$ is the same in both cases. This reduces the comparison to a minima function between the 2 (middle) *dot product* terms,

$$min(-2\overrightarrow{v_r}\overrightarrow{v_i^0}, -2\overrightarrow{v_r}\overrightarrow{v_i^1}) = max(\overrightarrow{v_r}\overrightarrow{v_i^0}, \overrightarrow{v_r}\overrightarrow{v_i^1})$$

since a *min* operation on negative numbers may be interpreted as an equivalent *max* operation on positive quantities.

Each *dot product* term may be expanded as

$$max(\pm r_0 \pm r_1, \pm r_0 \pm r_1)$$

where, the signs of each term depend on symbols, $\mathbf{v_i^{(0)}}$ and $\mathbf{v_i^{(1)}}$, being compared. Thus, the *squared* Euclidean metric distance calculation to compute the *branch metric* may be performed with a simple add/subtract operation.

Traceback

The general approach to traceback is to accumulate path metrics for up to five times the constraint length (*5 * (K - 1)*), find the node with the largest accumulated cost, and begin traceback from this node.

However, computing the node which has accumulated the largest cost (either the largest or smallest integral path metric) involves finding the *maxima* or *minima* of several (usually 2^{K-1}) numbers, which may be time consuming when implemented on embedded hardware systems.

Most communication systems employ Viterbi decoding involving data packets of fixed sizes, with a fixed bit/byte pattern either at the beginning or/and at the end of the data packet. By using the known bit/byte pattern as reference, the start node may be set to a fixed value, thereby obtaining a perfect Maximum Likelihood Path during traceback.

Limitations

A physical implementation of a viterbi decoder will not yield an *exact* maximum-likelihood stream due to quantization of the input signal, branch and path metrics, and finite *traceback length*. Practical implementations do approach within 1dB of the ideal.

Perforated codes

A hardware viterbi decoder of *perforated codes* is commonly implemented in such a way:

- A deperforator, which transforms the input stream into the stream which looks like an original (imperforated) stream with ERASE marks at the places where bits were erased.
- A basic viterbi decoder understanding these ERASE marks (that is, not using them for branch metric calculation).

Software implementation

One of the most time-consuming operations is an ACS butterfly, which is usually implemented using an assembly language and appropriate instruction set extensions (such as SSE2) to speed up the decoding time.

Applications

The Viterbi decoding algorithm is widely used in the following areas:

- Decoding trellis-coded modulation (TCM), the technique used in telephone-line modems to squeeze high spectral efficiency out of 3 kHz-bandwidth analog telephone lines. The TCM is also used in the PSK31 digital mode for amateur radio and sometimes in the radio relay and satellite communications.
- Automatic speech recognition
- Decoding convolutional codes in satellite communications.
- Computer storage devices such as hard disk drives.