WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Беларуская (тарашкевіца)
Català
Čeština
Deutsch
Ελληνικά
Español
فارسی
Français
한국어
हिन्दी
Italiano
עברית
Magyar
日本語
Norsk nynorsk
Polski
Português
Русский
Simple English
Suomi
Svenska
ไทย
Türkçe
Українська
Edit links

Error detection and correction

From Wikipedia, the free encyclopedia
(Redirected from Redundancy check)

Not to be confused with error handling.

This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(August 2008)*

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 in many cases.

Contents [hide]

Definitions [edit]

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.

History [edit]

An early systematic use of error detection was by Jewish scribes in the precise copying of the Jewish bible, beginning before Christ. An emphasis on minute details of words and spellings evolved into the idea of a perfect text in 135 CE, and with it increasingly forceful strictures that a deviation in even a single letter would make a Torah scroll invalid.[1][2] The scribes used methods such as summing the number of words per line and per page (Numerical Masorah), and checking the middle paragraph, word and letter against the original. The page was thrown out if a single mistake was found, and three mistakes on a single page would result in the entire manuscript being destroyed (the equivalent of retransmission on a telecommunications channel). The effectiveness of their methods was verified by the accuracy of copying through the centuries demonstrated by discovery of the Dead Sea Scrolls in 1947–1956,[3] dating from c.408 BCE-75 CE.[4]

Introduction [edit]

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 corrupted. 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 variable, 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.

Implementation [edit]

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)*.

Error detection schemes [edit]

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).

A random-error-correcting code based on minimum distance coding can provide a strict guarantee on the number of detectable errors, but it may not protect against a preimage attack. A repetition code, described in the section below, is a special case of error-correcting codes: although rather inefficient, a repetition code is suitable in some applications of error correction and detection due to its simplicity.

Repetition codes [edit]

Main article: Repetition code

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 are 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.

A repetition code is very inefficient, 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.[5][6]

Parity bits [edit]

Main article: Parity bit

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 [edit]

Main article: Checksum

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 ones'-complement operation 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 Damm algorithm, 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) [edit]

Main article: Cyclic redundancy check

A *cyclic redundancy check (CRC)* is a non-secure hash function designed to detect accidental changes to digital data in computer networks; as a result, it is not suitable for detecting maliciously introduced errors. It is characterized by specification of what is called a *generator polynomial*, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend, such that the remainder becomes the result.

A cyclic code has favorable properties that make it 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 [edit]

Main article: Cryptographic hash function

The output of a *cryptographic hash function*, also known as a *message digest*, can provide strong assurances about data integrity, whether changes of the data are accidental (e.g., due to transmission errors) or maliciously introduced. 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. If an attacker can change not only the message but also the hash value, then a *keyed hash* or message authentication code (MAC) can be used for additional security. Without knowing the key, it is infeasible for the attacker to calculate the correct keyed hash value for a modified message.

Error-correcting codes [edit]

Main article: Forward error correction

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.

Error correction [edit]

Automatic repeat request (ARQ) [edit]

Main article: 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.[7]

For example, ARQ is used on shortwave radio data links in the form of ARQ-E, or combined with multiplexing as ARQ-M.

Error-correcting code [edit]

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 backchannel 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 [edit]

Main article: Hybrid ARQ

Hybrid ARQ is a combination of ARQ and forward error correction. There are two basic approaches:[7]

- 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 [edit]

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 having no return channel cannot use ARQ. Applications that require extremely low error rates (such as digital money transfers) must use ARQ. Reliability and inspection engineering also make use of the theory of error-correcting codes.[8]

Internet [edit]

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 (see also RFC 3819).
- 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 Data-Link 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 [edit]

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.[9] 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.[10] 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 craft 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. Both craft use V2 RSV coding due to ECC system upgrades after 1989.

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 which 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) [edit]

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 [edit]

Error detection and correction codes are often used to improve the reliability of data storage media.[*citation needed*] 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.[11] RAID systems use a variety of error correction techniques, to correct errors when a hard drive completely fails. Filesystems such as ZFS or Btrfs, as well as some RAID implementations, support data scrubbing and resilvering, which allows bad blocks to be detected and (hopefully) recovered before they are used. The recovered data may be re-written to exactly the same physical location, to spare blocks elsewhere on the same piece of hardware, or to replacement hardware.

Error-correcting memory [edit]

Main article: ECC 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.[12]

In addition to hardware providing features required for ECC memory to operate, operating systems usually contain related reporting facilities that are used to provide notifications when soft errors are transparently recovered. An increasing rate of soft errors might indicate that a DIMM module needs replacing, and such feedback information would not be easily available without the related reporting capabilities. An example is the Linux kernel's *EDAC* subsystem (previously known as *bluesmoke*), which collects the data from error-checking-enabled components inside a computer system; beside collecting and reporting back the events related to ECC memory, it also supports other checksumming errors, including those detected on the PCI bus.[13][14][15]

A few systems also support memory scrubbing.

See also [edit]

Computer science portal

- Berger code
- Forward error correction
- Link adaptation

- List of algorithms for error detection and correction
- List of checksum algorithms
- List of error-correcting codes
- Reliability (computer networking)
- Burst error-correcting code

References [edit]

1. ^ Rambam, The Laws of Tefillin, Mezuzot, and Torah Scrolls, 1:2
2. ^ Menachem Cohen, The Idea of the Sanctity of the Biblical Text and the Science of Textual Criticism in *HaMikrah V'anachnu*, ed. Uriel Simon, HaMachon L'Yahadut U'Machshava Bat-Zmananu and Dvir, Tel-Aviv, 1979.
3. ^ Fagan, Brian M., and Charlotte Beck, *The Oxford Companion to Archeology*, entry on the "Dead sea scrolls", Oxford University Press, 1996.
4. ^ "The Digital Dead Sea Scrolls: Nature and Significance". Israel Museum Jerusalem. Retrieved 2014-10-13.
5. ^ Frank van Gerwen. "Numbers (and other mysterious) stations". Retrieved 12 March 2012.
6. ^ Gary Cutlack (25 August 2010). "Mysterious Russian 'Numbers Station' Changes Broadcast After 20 Years". *Gizmodo*. Retrieved 12 March 2012.
7. ^ *a b* A. J. McAuley, *Reliable Broadband Communication Using a Burst Erasure Correcting Code*, ACM SIGCOMM, 1990.
8. ^ Ben-Gal I., Herer Y. and Raz T. (2003). "Self-correcting inspection procedure under inspection errors" (PDF). IIE Transactions on Quality and Reliability, 34(6), pp. 529-540.
9. ^ K. Andrews et al., *The Development of Turbo and LDPC Codes for Deep-Space Applications*, Proceedings of the IEEE, Vol. 95, No. 11, Nov. 2007.
10. ^ W. William Cary Huffman; Vera S. Pless (2003). *Fundamentals of Error-Correcting Codes*. ISBN 978-0-521-78280-7.
11. ^ My Hard Drive Died. Scott A. Moulton
12. ^ "Using StrongArm SA-1110 in the On-Board Computer of Nanosatellite". Tsinghua Space Center, Tsinghua University, Beijing. Retrieved 2009-02-16.
13. ^ Jeff Layton. "Error Detection and Correction". *Linux Magazine*. Retrieved 2014-08-12.
14. ^ "EDAC Project". *bluesmoke.sourceforge.net*. Retrieved 2014-08-12.
15. ^ "Documentation/edac.txt". *Linux kernel documentation*. kernel.org. 2014-06-16. Retrieved 2014-08-12.

Further reading [edit]

- Shu Lin, Daniel J. Costello, Jr. (1983). *Error Control Coding: Fundamentals and Applications*. Prentice Hall. ISBN 0-13-283796-X.

External links [edit]

- The on-line textbook: Information Theory, Inference, and Learning Algorithms, by David MacKay, contains chapters on elementary error-correcting codes; on the theoretical limits of error-correction; and on the latest state-of-the-art error-correcting codes, including low-density parity-check codes, turbo codes, and fountain codes.
- Compute parameters of linear codes – an on-line interface for generating and computing parameters (e.g. minimum distance, covering radius) of linear error-correcting codes.
- ECC Page
- SoftECC: A System for Software Memory Integrity Checking
- A Tunable, Software-based DRAM Error Detection and Correction Library for HPC
- Detection and Correction of Silent Data Corruption for Large-Scale High-Performance Computing

Categories: Error detection and correction | Computer errors