Computer Networks - Lecture 19 & 20

- Author: Ruben Schenk
- Date: 14.06.2021
- Contact: ruben.schenk@inf.ethz.ch

6.2 Error Detection and Correction

Bit-errors are introduced by signal attenuation and electromagnetic noise and are, in general, not avoidable when using certain types of media for transport. Our aim is to detect and possibly even correct these errors at a very low level, namely at the link layer. A possibility is to add `redundancy`, i.e., check bits that allow some errors to be detected. Adding even more check bits even makes it possible to directly correct some errors.

Goal: Structure code to detect many errors with few check bits and modest computational effort.

6.2.1 Intuition & Usage

A `codeword` consists of D data bits and R check bits. The sender computes R based on the data D and sends the codeword of $D + R$ bits (concatenated). A receiver then, upon receiving $D + R$ bits, recomputes R' based on D. If R' doesn't match R, then there is an error somewhere.

Note: For data bits D, the set of correct codewords should only be a tiny fraction of the set of all codewords, such that the probability of a randomly chosen codeword being correct is very small.

Hamming Distance

Let the `distance` of two codewords $D + R_1$ and $D + R_2$ be the number of bits that need to be flipped to turn $D + R_1$ into $D + R_2$ (or vice versa).

> The `hamming distance` of a code is the minimum distance between a pair of codewords.

> A code of hamming distance $d + 1$ can `detect` up to d errors.

> A code of hamming distance $2d + 1$ can `correct` up to d errors by mapping to the closest codeword.

6.2.2 Error Detection

Parity Bit

Take D data bits and add only a single `check bit` c such that c is the sum of all D bits modulo 2, i.e.

$c \equiv_2 \sum_{i=1}^{D} x_i$

where x_i denotes the i-th data bit.

The distance of the code is 2, it can thus detect exactly one error (but not locate it) and correct none.

Checksums

Idea: Sum up data in N-bit words, that gives a stronger protection than parity and is widely used in TCP/IP/UDP etc. `Internet checksum` works as follows:

Sending side:

1. Arrange data in 16-bit words.
2. Put 0 in the checksum position and add the words.
3. Add any carryover back to get 16 bits.
4. The checksum is now given by the complement (negation).

Receiving side:

1. Arrange the data in 16-bit words.
2. Add the words and the checksum.
3. Add any carryover back to get 16 bits.
4. Negate the result and check if it is 0, if not, then an error occurred.

Cyclic Redundancy Check (CRC)

Let the data D consist of d data bits. Sender and receiver then agree on a $k+1$ bit pattern, which is called the generator G (and can be expressed as a polynomial over the finite field $GF(2)$). The sender will then choose r bits R to append to D such that the resulting $d+r$ bit pattern is congruent modulo 2 to G, i.e., $D+R \equiv_2 G$.
The receiver then checks whether the received $d+r$ bits are divisible by G with a 0 remainder. If not, an error occurred.

6.2.3 Error Correction

The general problem that makes error correction so hard is that check bits aren't reliable either. If we can construct a code of Hamming Distance $\geq 2d+1$, then codewords containing at most d bit errors are uniquely mapped to the closest valid codeword.

Hamming Code

A `hamming code` is a code with hamming distance 3. It can hence correct 1 bit error. For a message of n bits, choose k such that $n = 2^k - k - 1$ holds. We insert check bits at positions of powers of 2, starting with position 1.
When written in binary, the positions of the check bits have exactly one bit set to 1. The check bit $p8$ for example records the parity of all positions that have the 4-th bit set to 1. Check bit $p4$ records parity of all positions that have the 3rd bit set to 1.

Now to decode, we recompute the check bits, arrange them as a binary number, called the `syndrome`. It tells us the error positions where the bit needs to be flipped. If the syndrome is 0, no error occurred.

6.2.4 Detection vs. Correction

Instead of correcting bit-errors, one could simply try to detect them and, upon detecting an error, requesting a retransmit. This might be more efficient than correction, but it is largely dependent on the setting:

- *Error correction* is needed when we expect error at a small rate or when there is no time for retransmission.
- *Error detection* is more efficient when errors are not expected and when errors are large when they occur.

6.3 Retransmissions

Instead of correcting errors, one can simply detect them and retransmit the frames in which they occurred.

Automatic Repeat Request (ARQ)

ARQ is often used when errors are common or if they must be corrected (e.g. TCP). IT works in the following way:

- Receiver: Automatically ACKs correct frames
- Sender: Automatically retransmits after timeout until ACK is received.

6.4 Multiple Access

`Multiplexing` is the network word for the sharing of a resource. A classic scenario is the sharing of a link among different users with one of the following approaches:

- `Time Division Multiplexing (TDM)`: Users take turns on a fixed schedule which lets them send at a high rate but only during a fraction of time.
- `Frequency Division Multiplexing (FDM)`: Puts users on different frequency bands which lets them send at a low rate but at all the time.

6.4.1 Multiplexing Network Traffic

Network traffic is bursty, the load varies greatly over time, and it is thus very inefficient to allocate the peak need with TDM/FDM. We therefore need `multiple access schemes` that multiplex users according to their demand.

Randomized Multiple Access

Assume a distributed network with no master node, i.e. no-one is in charge. We will now look at a randomized multiple access protocol which is also referred to as the `medium access control (MAC) protocol`, which provided the basis for the classic *Ethernet*.

ALOHA Protocol

The `ALOHA Protocol` connected the Hawaiian islands in the 1960s. The protocol is really simple:

- Nodes just send whenever it has traffic.
- If there was a collision, i.e. no ACK was received, then wait a random time and resend.

Under low load, this works fairly well. However, under high load the efficiency is very bad.

Carrier Sense Multiple Access (CSMA)

CSMA denotes a improvement of ALOHA for LAN. It works by listening for activity before sending. This way we have less collisions, but they can still occur due to sending delays.

- `CSMA/CD` is a CSMA protocol with added *collision detection*. This will reduce the cost of collisions by detecting them and aborting the rest of the frame time.

Let D be the distance between the two furthest away nodes in the network. Then the time window in which a node may hear of a collision is $2D$. We therefore impose a minimum frame size that lasts for $2D$ seconds such that the nodes can0t finish before the collision is detected. This leads to a minimum frame size of 64 bytes for Ethernet.

Another problem arises when a node detects that another node is sending. When it simply waits and sends when the other node is done, this might lead to multiple waiting loads queuing up and even amplifying the problem. One simple solution is, that for N queued senders, each sends with probability $\frac{1}{N}$. The waiting time interval is doubled for each successive collision which leads to a binary exponential backoff.

Wireless Multiple Access

Wireless is much more complicated than the wired case:

- Nodes may have a *different area of coverage*. This may lead to the fact that for some nodes there is interference and for others there isn't. We distinguish between:
  - `Hidden terminals`: Two nodes that cannot reach each other, yet they collide ad some intermediate node.
  - `Exposed terminals`: Two nodes that can reach and thus "hear" each other, yet they don't collide.
- Nodes can't hear while sending and therefore, collision detection is a waste of time.

`Multiple Access with Collision Avoidance (MACA)` is a possible solution. The protocol works with the following rules:

1. A sender node transmits a RTS (Request-To-Send, with frame length).
2. The receiver replies with a CTS (Clear-To-Send, with frame length).
3. Sender transmits the frame while nodes hearing the CTS stay silent.

Note: MACA solves both the hidden and exposed terminal problem.
Note: Collisions on the RTS/CTS are still possible, but less likely.

Token Ring

Another protocol where nodes are arranged in a ring. A token rotates "permission to send" to each node in turn.

- This leads to a fixed overhead, no collisions, predictable, and a regular chance for each node to send.
- However, what if a token is lost, what if a token manager crashes? Overhead is way too high for lower loads.

Note: In practice, it is hard to beat random multiple access protocols.