Communication Networks I (16:332:543)

Lecture no 2, September 3, 1998

Lecture Summary by Cristina Comaniciu

Keywords: layered architecture, OSI reference model, Data Link layer, frame, header, trailer, code word, error detection, minimum distance, burst error detection capability, ARQ, Stop and Wait.

The layered architecture discussed in the previous lecture promotes the concept of modularity. As a consequence, one can treat each system module as a black box described in terms of the input, output and the functional relation between input and output. This allows separate design and analysis for each network layer, provided that known interfaces between layers are defined.

The first layer in the OSI reference model which is analyzed in this course is the Data Link layer. The Data Link layer perceives the Physical layer as an unreliable bit pipe and its function is to enforce reliability by providing error-free packets to the Network layer.

In order to achieve this, the Data Link layer constructs frames by adding a header (containing overhead control bits) in front of the data packet received from the Network layer and a trailer (containing control bits for error detection) at the end of the packet (see Figure 1.).

![Figure 1: Frame Content](image)

When a frame is received from the Physical layer, the Data Link uses the supplementary information from the trailer bits to detect errors. It is assumed that the receiver can detect the beginning and the end of the frame. The detection capability is determined by the number of bits in the trailer together with the encoding technique for the supplementary information.

More specifically, if we define the code word as being a valid frame, the detection capability is given by the minimum distance between any two codewords. The minimum distance $d = \ldots$

---

1 In the text book Data Link layer appears as Data Link Control layer
2 Some fundamentals for Physical layer are presented in the textbook, in section 2.2
3 In practice there will be always a nonzero error probability which can be made negligible by error detection and correction. The order of magnitude for an error to be considered negligible depends on the application
smallest number of errors that changes a code word into another. For a collection of code words having minimum distance \( d \), up to \( d - 1 \) errors can be detected\(^4\).

We define the length of a burst of errors, \( B \), as the number of bits from the 1st error to the last error (inclusive).

The burst error detection capability is the largest \( B \) such that all bursts of length \( B \) can be detected.

Let the number of bits in the trailer be equal to \( L \) and the number of data +header bits equal to \( K \). Thus the frame has \( K + L \) bits. Several examples of error detection mechanisms are discussed below:

1. **Single Parity Check**: For this case, \( L = 1 \) and if \( N_1 \) is the number of ones in the \( K \) bits (trailer+header), the actual value for the bit in the trailer is:

   \[
   b = \begin{cases} 
   1, & \text{if } N_1 \text{ is odd} \\
   0, & \text{if } N_1 \text{ is even} 
   \end{cases}
   \]

   Therefore \( b \) represents the \( \text{mod} \ 2 \) sum of all \( K \) bits. This encoding allows detection for all odd number of errors but does not detect any even number of errors. The minimum distance for this code is \( d = 2 \).

   For practical situations, Single Parity Check is ineffective due to the fact that it will detect errors in approximately half of the cases (the probability for an even number of errors is equal to the probability for an odd number of errors). This is due to the fact that a practical model for errors should consider bursts of errors instead of using an independent bit error model. Examples of situations in which burst errors occur: burst noise, fading for wireless systems (when the signal goes through a deep fade a string of consecutive bits is affected\(^5\)), error in the reception of a single sample for the physical channel which results in several bits errored (several bits are encoded into a single sample), etc.

2. **Horizontal and Vertical Check Bits**:

   In this case the codeword is arranged in a two dimensional array as in Figure 2.

   ![Figure 2: Horizontal and Vertical Check Bits](image)

   The code can detect an odd number of errors on each row and on each column. Thus, if an even number of errors occurs in a row, it may be detected by the columns’ parity check bits.

---

\(^4\)The code will detect any configuration of up to \( d - 1 \) errors, although some particular configurations of more than \( d - 1 \) errors might be detected.

\(^5\)An additional encoding technique for reducing the effect of long strings of bit errors is interleaving.
A rectangular pattern of errors (as in Figure 3 a) fails to be detected. Moreover, the code fails to detect relative short bursts of errors (see Figure 3 b for an example of a shortest burst, length of a row +2, undetected by the code). The horizontal and vertical parity check code has minimum distance \( d = 4 \).

3. Parity Check Codes

The concept used for the horizontal and vertical parity check code can be extended to arbitrary parity check codes. In general, a parity check code will have \( K \) bits in the data string plus \( L \) additional control bits (see Figure 1).

An accurate error probability (defined as the probability of an undetected frame error) is hard to compute since all possible undetectable error patterns should be considered, and a realistic error model should also assume that the bit errors are not independent.

Therefore, the effectiveness of a code for error detection is given by:

- minimum distance of the code,
- burst detecting capability,
- probability that a completely random string will be accepted as a codeword.

This probability is computed as:

\[
P = \frac{\text{total number of possible combinations for the } K \text{ data bits}}{\text{total number of possible combinations for the } K + L \text{ frame bits}} = \frac{2^K}{2^{K+L}} = 2^{-L}
\]

The goal is to make this probability very small. This can be achieved by increasing \( L \). The penalty is an increase in the packet overhead.

Note: The parity checks codes widely used at the Data Link layer are the \textit{Cyclic Redundancy Checks} \(^6\) (CRC). CRCs provide a systematic way of creating nearly random parity checks and have the advantage of simple encoding and decoding procedure.

ARQ

We assume now that the receiver can detect the beginning and the end of the frame, and that the frames that are received in error can be detected. Thus, Data Link will still have to solve problems such as: detected errored frames and lost frames. To overcome this problems a family

---

\(^6\)Description of CRC can be found in the textbook, section 2.3.4
of protocols (ARQ = Automatic Repeat reQuest) is employed, which ensures retransmission of errored or lost messages until they are correctly received.

It is assumed that each frame is delayed by an arbitrary and variable time before arriving at the receiver, some frames might be lost, but the frames that arrive, will arrive in the same order as transmitted.

The simplest ARQ protocol is *Stop and Wait*:

The basic idea for the *Stop and Wait* protocol is to ensure that each packet has been received correctly before transmitting the next packet. The algorithm can be summarized as follows:

- A sends a packet
- B receives the packet and sends an ACK (Acknowledgment) or a NAK (Negative Acknowledgment)
- If A gets an ACK, sends the next packet,
- If A gets an NAK, repeats previous packet,
- If A waits too long (time-out), A retransmits previous packet.

A potential problem with this strategy is illustrated in Figure 4.

Figure 4: Unnumbered packets problem in Stop and Wait ARQ

A long delay for receiving the ACK from B determines A to time-out and to repeat packet 0, but B is expecting packet 1, since it knows that the acknowledgment for 0 was already sent. The confusion can be avoided by numbering the data packets. *SN* associated with each data packet is the sender’s sequence number.

If we consider further that the link from B to A can experience the same types of errors as the link from A to B, it is possible that an ACK might be lost. Therefore, each time it receives a correct packet B has to send again ACK. In Figure 5, the second ACK for packet 0 arrives after packet 1 has been transmitted. Frame 1 is errored, but A will interpret the second ACK for packet 0 as ACK for packet 1 and will send packet 2. Packet 1 will never arrive at B.

The solution is to number the ACKs as well. By convention, when B sends an ACK it has a number *RN* = number of the next requested packet (Figure 6).

---

7Since it is possible to transmit ACKs from B to A on frames from B to A ("piggyback" ACKs) the scenario of receiving an NAK instead of an ACK is not plausible, since that assumes that a whole frame is received in error and undetected. This is conflicting with the initial assumption that all errored frames can be detected.
Figure 5: Unnumbered ACks problem in Stop and Wait ARQ

Figure 6: Numbering for ACKs in Stop and Wait ARQ

In Figure 7, the example is revisited considering also RN's numbers. The algorithm seems to work well, in the sense that B receives all transmitted packets in a finite time. The proof for the correctness of Stop and Wait ARQ will be discussed in the next lecture.

Figure 7: Stop and Wait ARQ example

References
[1] Dr. Roy Yates, *Communication Networks I. Course Notes*