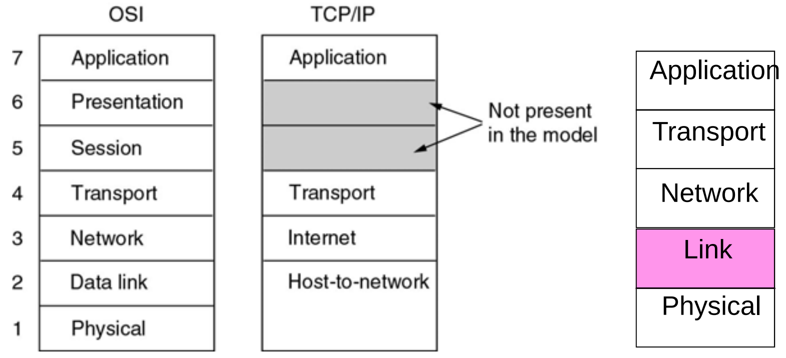
Week 3 – Data Link Layer

COMP90007
Internet Technologies

The Data Link Layer in OSI and TCP/IP



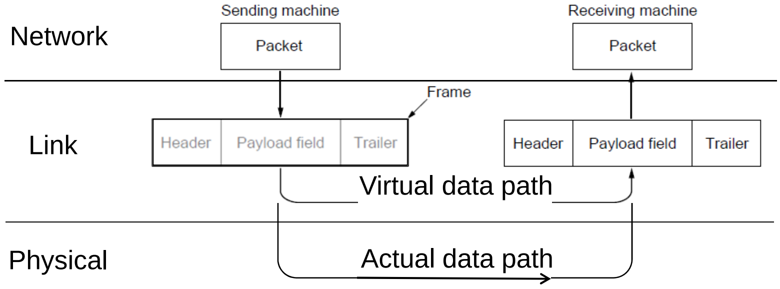
- Reliable, efficient communication of “frames” between two adjacent machines.
- Handles transmission errors and flow control.

Functions & Methods of the Data Layer

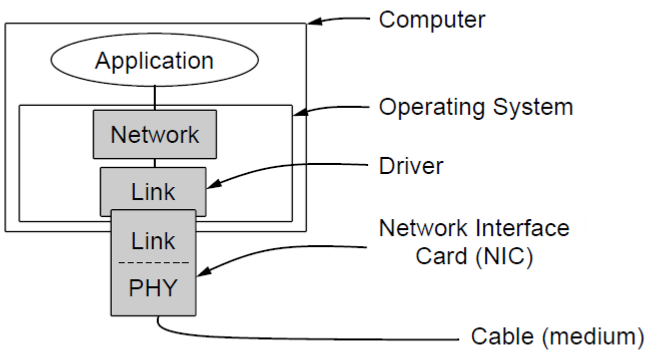
- Functions of the data link layer:
  1. Provide a well-defined service interface to network layer
  2. Handling transmission errors
  3. Data flow regulation
- Primary method:
  - Take **packets from network layer**, and encapsulate them **into frames** (containing a header, a payload, a trailer)

Relation Between Packets and Frames

Link layer accepts **packets** from the network layer, and encapsulates them into **frames** that it sends using the physical layer; reception is the opposite process



Typical Implementation



Type of Services

- <u>**Connection-Oriented vs Connectionless**</u>: Whether a connection is setup before sending a message
- <u>**Acknowledged vs Unacknowledged**</u>: Whether the service provider give the service user an acknowledgement upon delivering the message

Services Provided to Network Layer

- Principal concern is transferring data from network layer on source host to network layer on destination host
- Services provided:
  - Unacknowledged connectionless service
  - Acknowledged connectionless service
  - Acknowledged connection-oriented service

Unacknowledged Connectionless Service

- Source host transmits independent frames to recipient host with no acknowledgement
- No logical connection establishment or release
- No lost frame recovery mechanism (or left to higher levels)
- E.g. Ethernet LANs (No logical connection is established beforehand or released afterward)
- Real-time traffic, e.g., voice

Acknowledged Connectionless Service

- Source host transmits independent frames to recipient host with acknowledgement
- No logical connection establishment or release
- Each frame individually acknowledged (retransmission if lost or errors)
- E.g. Wireless – IEEE 802.11 WiFi

Acknowledged Connection-Oriented Service

- Source host transmits independent frames to recipient host after connection establishment and with acknowledgement
- Connection established and released (communicate rate and details of message)
- Frames numbered, counted, acknowledged with logical order enforced
- Unreliable links such as satellite channel or long distance telephone circuit

Framing

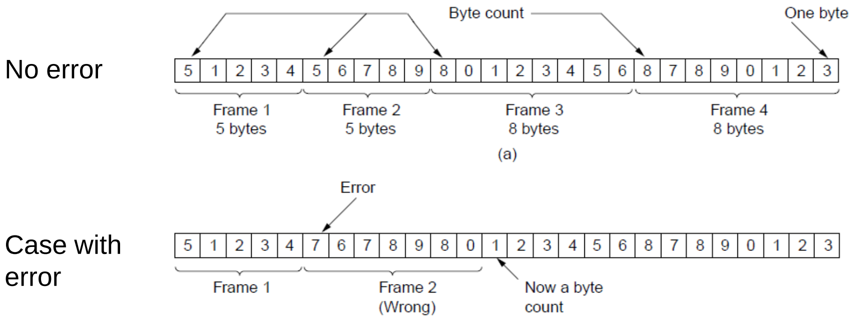
- Physical layer provides no guarantee a raw stream of bits is error free
- Framing is the method used by data link layer to break raw bit stream into discrete units and generate a checksum for the unit
- Checksums can be computed and embedded at the source, then computed and compared at the destination checksum = f(payload)
- The primary purpose therefore, of framing, is to provide some level of reliability over the unreliable physical layer

Framing Methods

- Framing methods:
  - Character(Byte) count
  - Flag bytes with byte stuffing
  - Start and end flags with bit stuffing
- Most data link protocols use a combination of character count and one other method

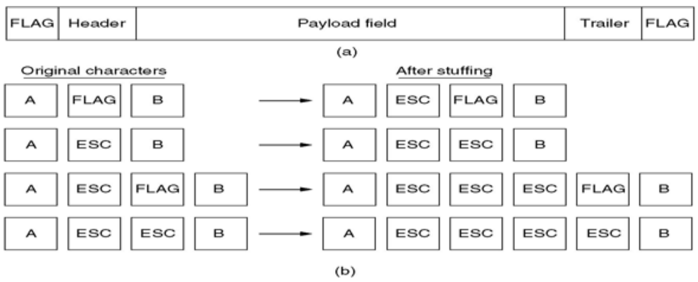
Character Counts

- Uses a field in the frame header to specify the number of characters in a frame



Flag Bytes with Byte Stuffing

- Each frame starts and ends with a special byte -“flag byte”



Start and End flags with Bit stuffing

- Frames contain an arbitrary number of bits and allow character codes
- With an arbitrary number of bits per character
- Each frame begins and ends with a special bit pattern

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 — The original data

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0 — Sent data

Stuffed bits

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 — Destuffing at receiver

Insert 0 after five ones (11111)

Error Control

- Ensuring that a garbled message by the physical layer is not considered as the original message by the receiver by adding check bits
- Error Control deals with
  - Detecting the error
  - Correcting the error
  - Re-transmitting lost frames
- Link layer deals with bit errors

Error Detection and Correction

- Physical media may be subject to errors
- Errors may occur randomly or in bursts
- Bursts of errors are easier to detect but harder to resolve
- Resolution needs to occur before handing data to network layer
- Key issues
  - ***Fast*** mechanism and ***low*** computational overhead
  - Detection of ***different kinds of error***
  - ***Minimum*** amount of extra bits send with the data

Example

- Repeat the bits, if a copy is different than the other there is an error
  - 01101 -> 000 111 111 000 111
- How many different errors can this detect? 2
- How many errors can this correct? 1
- What is the minimum number of errors that can fail the algorithm? 3
  (Note: the algorithm can detect up to 2 errors and retransmission is needed but completely failed with 3 errors)
- What is the overhead? Sending 1 bit 3 times

Error Bounds – Hamming distance

Code turns data of *n* bits into codewords of *n+k* bits

Hamming distance is the minimum bit flips to turn one valid codeword into any other valid one.

- Example with 4 codewords of 10 bits (n=2, k=8):
  - 0000000000, Hamming distance is 5
  - 0000011111,
  - 1111100000, and
  - 1111111111

Bounds for a code with distance:

- 2d+1 – can correct d errors (e.g., 2 errors above)
- d+1 – can detect d errors (e.g., 4 errors above)

Error Bounds

Q: Why can a code with distance *2d+1* detect up to *d* errors?

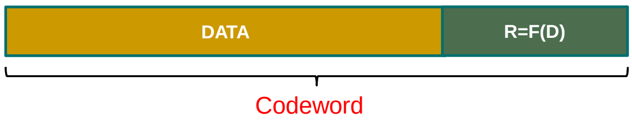
- Errors are corrected by mapping a received invalid codeword to the nearest valid codeword, i.e., the one that can be reached with the fewest bit flips
- If there are more than d bit flips, then the received codeword may be closer to another valid codeword than the codeword that was sent

Example: Sending 0000000000 with 2 flips might give 1100000000 which is closest to 0000000000, correcting the error.

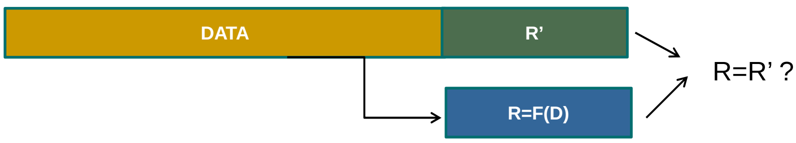
But with 3 flips 1110000000 might be received, which is closest to 1111100000, which is still an error

How it works?

- Sender calculates R check bits using a function of data bits:



- Receiver: Receive the codeword and calculates the same function on the data and match the results with received data check bits:



Error Detecting Codes

- More efficient in some transmission media – e.g. where low error rates occur (copper wires)
- ***Parity*** (1 bit): XOR all the data bits and add the result as the check bit (Hamming distance=2)
- ***Checksum*** (16 bit): Add 16 bits of data and calculate 1's complement and add to the data as the check bits (Hamming distance=2)
- ***Cyclical Redundancy Check*** *(CRC)* – Use division by a k bits polynomial in base-2's representation (Standard 32 bit CRC: Hamming distance=4)

Error Correcting Codes

- More efficient in noisy transmission media e.g., wireless
- Challenge is that the error can be in the check bits
- Assumption on a specific number of errors occurring in transmission

(Internet) Checksum

- There are different variation of checksum
- Internet Checksum (16-bit word): Sum modulo 2^{16} and adding any overflow of high order bits back into low-order bits

Example of Check Sum

Calculate checksum (5bit word) for data 00110 10001 11001 01011

1. 00110
+10001 = 10111

2. 11001
+10111 = 10000 +1 = 10001

3. 01011
+10001 = 11100

4. The checksum is one's complement of 11100 which is 00011

Data sent: 00110 10001 11001 01011 00011

Cyclic Redundancy Check

- Based on a generator polynomial G(x)
  - Eg. G(x) = x^4+x+1 (10011)
  - Let *r* be the degree of *G*(*x*) (r=4). Append *r* zero bits to the low-order end of the frame so it now contains *m* + *r* bits and corresponds to the polynomial $x^rM(x)$.
  - Divide the bit string corresponding to *G*(*x*) into the bit string corresponding to $x^rM(x)$, using modulo 2 division.
  - Subtract the remainder (which is always *r* or fewer bits) from the bit string corresponding to $x^rM(x)$ using modulo 2 subtraction. The result is the checksummed frame to be transmitted. Call its polynomial *T*(*x*).

Example

Data: 1101001 and G(x) = x4+x+1 (10011)

5 bits polynomial add 4 bits as the checksum – so add 0000

```
10011 | 11010010000
        10011
        ------
        010010
          10011
          ------
          000011000
                10011
                ------
                010110
                  10011
                ------
                  00101   =>  Data sent:
                              11010010101
```

Hamming Code

- n=2^k-k-1 (n: number of data, k: check bits)
- Put check bits in positions *p* that are power of 2, starting with position 1
- Check bit in position *p* is parity of positions with a *p* term in their value
- Example: Data: 0101 - > requires 3 check bits

Example

Put check bits in positions p that are power of 2, starting with position 1

- Data: 0101 → requires 3 check bits

Position	P1	P2	P3	P4	P5	P6	P7
Data	?	?	0	?	1	0	1

1. Calculate the parity bits for P1, P2, P4

P1 + P3 + P5 + P7 = ?+0+1+1= 0 (even)
P2 + P3 + P6 + P7 = ?+0+0+1= 1 (odd)
P4 + P5 + P6 + P7 = ?+1+0+1= 0 (even)

Data sent: 0100101

error

Example 1: At the receiver: 0100100

P1 + P3 + P5 + P7 = 0+0+1+0= 1 ×
P2 + P3 + P6 + P7 = 1+0+0+0= 1 ×
P4 + P5 + P6 + P7 = 0+1+0+0= 1 ×

Error bit = P1+P2+P4 = P7

error

Example 2: At the receiver: 0000101

P1 + P3 + P5 + P7 = 0+0+1+1= 0
P2 + P3 + P6 + P7 = 0+0+0+1= 1 ×
P4 + P5 + P6 + P7 = 0+1+0+1= 0

Error bit = P2

Data Transmission

- So far we discussed how to send single messages and now we will look at a series of messages
- A service to send messages should have:
  - Reliability
  - Flow Control

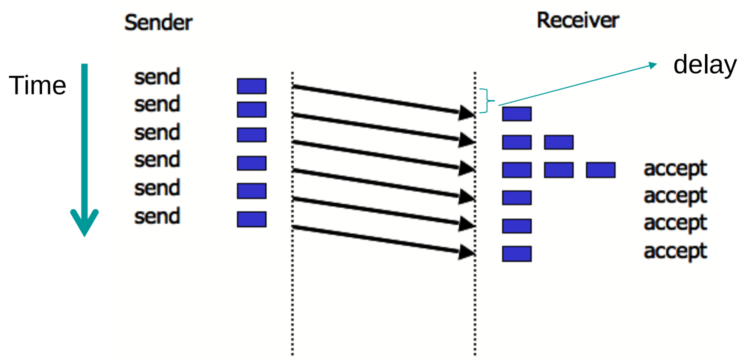
Reliability

- Each layer need to make sure the service provided to other layers is reliable
- Retransmission with error detection is a way of ensuring reliability
- Error correction is another way but has its own shortcomings

Flow Control

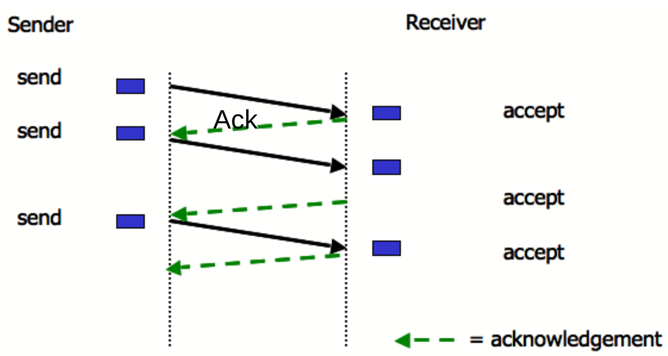
- The fast senders vs slow receivers problem requires a solution
- Principles to control when sender can send next frame
  - Feedback based flow control (usually used in DL layer)
  - Rate based flow control

A Very Simple Protocol



Acknowledged transmission

(think: fast sender / slow receiver)
Data transmitted in one direction
Time is relatively important, buffer space constrained

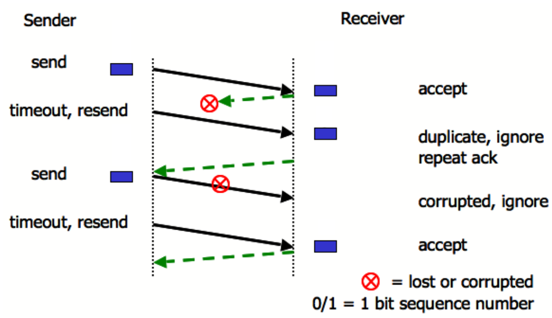


Noisy Channel Protocol

- Frames can be lost either entirely or partially
- Requires distinction between frames already sent/received and those being re-transmitted
- Requires *timeout function* to determine arrival or non-arrival of complete frames

Stop and Wait Protocol

- Concept of ARQ (Automatic Repeat reQuest)
  - Ack and Timeout
- Stop and Wait
  - One bit Ack



Link Utilization in Stop and Wait Protocols

Principle of efficiency in communication is measured by **Link Utilization (U)**.

Let **B** be the **bit-rate** of the link and **L** the **length of the frame**,

T_f= Time needed to transmit a frame of length L,

T_p= Propagation delay of the channel,

T_a= Time for transmitting an Ack,

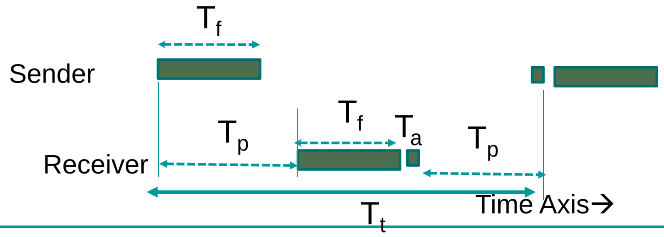
So we have $\mathbf{T_f = L/B}$. We can assume $\mathbf{T_a = 0}$. $\mathbf{T_t = T_f + 2T_p}$.

For example for a Link with B=1Mbps and T_p=50ms and frame size 10Kb:

U= 10000/(10000+0.1*10⁶)=1/11;

U = (Time of transmitting a frame)/(Total time for the transfer) = T_f/T_t

We have $U = T_f / (T_f + 2T_p) = (L/B) / (L/B + 2T_p) = L/ (L+ 2T_p B)$.

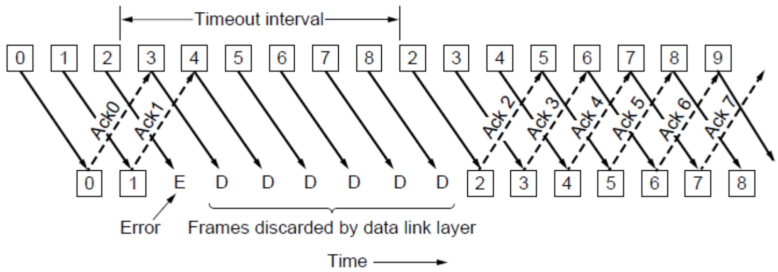


Sliding Window Protocols

- Data is commonly transmitted in both directions simultaneously
- Sender maintains a set of sequence numbers corresponding to frames it is allowed to send (within the "sending window")
- Receiver maintains a set of sequence numbers corresponding to frames it is allowed to accept (within the "receiving window")
- Stop and Wait can be seen as a special case with window size 1

Protocol Using Go-Back-N

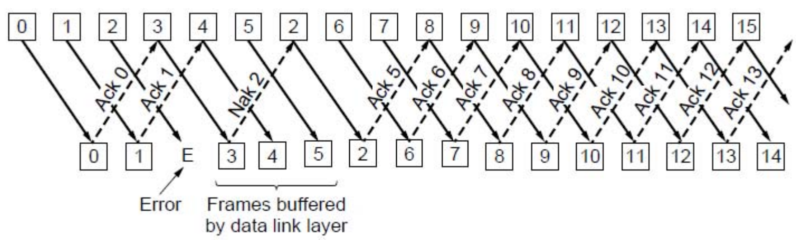
- Long transmission times need to be taken into account when programming timeouts e.g., low bandwidth or long distance
- Senders don't need to wait for acknowledgement for each frame before sending next frame



Receiver window size =1, Sender window size is N

Selective Repeat

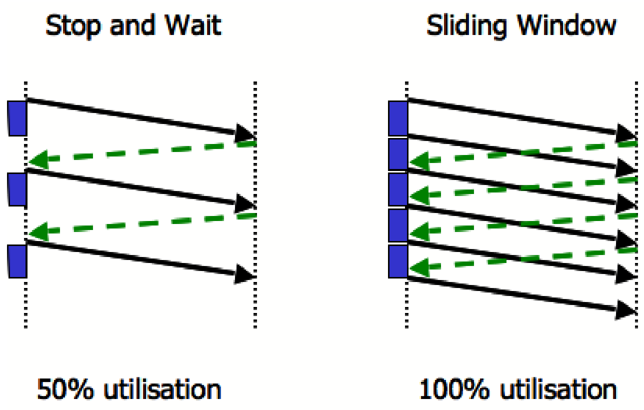
- Receiver accepts frames anywhere in receive window
  - Cumulative ack indicates highest in-order frame
  - NAK (negative ack) causes sender retransmission of a missing frame before a timeout resends window



Go-Back-N vs Selective Repeat

- Go-Back-N: receiver discards all subsequent frames from error point, sending no acknowledgement, until the next frame in sequence
- Selective Repeat: receiver buffers good frames after an error point, and relies on sender to resend oldest unacknowledged frames
- Trade-off between [redacted] and [redacted]

Link Efficiency Compared

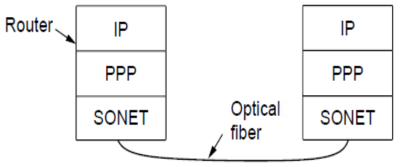


Example Data Link Protocols

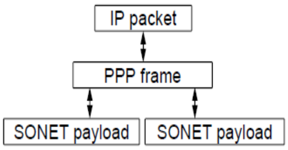
- Packet over SONET
- PPP (Point-to-Point Protocol)
- ADSL (Asymmetric Digital Subscriber Loop)

Packet over SONET

- Packet over SONET is the method used to carry IP packets over SONET optical fiber links
  - Uses PPP (Point-to-Point Protocol) for framing



Protocol stacks



PPP frames may be split over SONET payloads

PPP

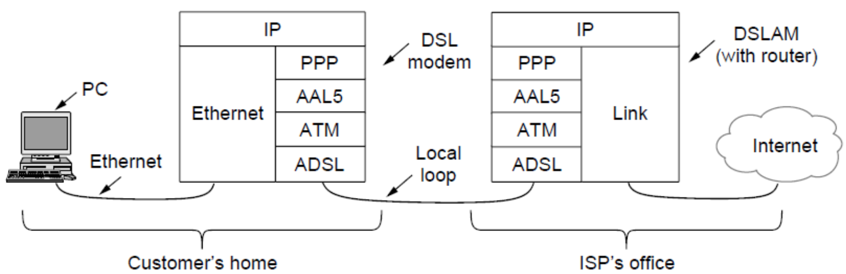
- PPP (Point-to-Point Protocol) is a general method for delivering packets across links
  - Framing uses a flag (0x7E) and byte stuffing
  - “Unnumbered mode” (connectionless unacknowledged service) is used
  - Errors are detected with a checksum

Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

0x21 for IPv4IP packet

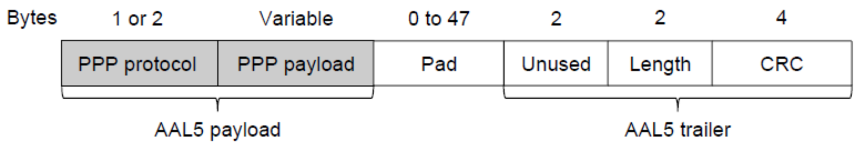
ADSL

- Widely used for broadband Internet over local loops
  - ADSL runs from modem (customer) to DSLAM (ISP)
  - IP packets are sent over PPP and AAL5/ATM (over)



ADSL

- PPP data is sent in AAL5 frames over ATM cells:
  - ATM is a link layer that uses short, fixed-size cells (53 bytes); each cell has a virtual circuit identifier
  - AAL5 is a format to send packets over ATM
  - PPP frame is converted to a AAL5 frame (PPPoA)



AAL5 frame is divided into 48 byte pieces, each of which goes into one ATM cell with 5 header bytes