

Data Link Layer: Reliable Data Transmission

Xiaoming Fu

Telematics group
University of Göttingen, Germany

Data Link Layer

- **Reliable transmission of packets over a link**
 - Framing: determine the start and end of packets
 - Error detection and correction (*see first part*)
 - Automatic Repeat ReQuest (ARQ)
 - PPP
- Sharing a broadcast channel: multiple access
 - Channel partitioning: TDMA, FDMA, CDMA
 - Random access: Aloha, CSMA, CSMA/CD
 - Taking turns: Token Ring, Token Bus
 - LAN examples: Ethernet and Wireless LAN
- Interconnection devices (*self-learning*)
 - Hubs, bridges, switches, and routers

Credits:

- Eytan Modiano, MIT
 - James Kurose & Keith Ross: Computer Networking(2nd Ed.), Addison-Wesley, 2002
- WS 2003/04, fu@informatik.cs.uni-goettingen.de

2

Framing

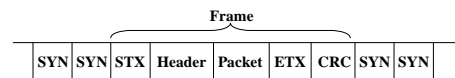
- Q: what is the lowest layer in the OSI model speaks sort of "protocol"?

010100111010100100101010100111000100

- Q: where is the DATA?
- A: Framing techniques (synchronization at data link layer) are needed: *protocol*
 - Method 1: Character oriented framing
 - Method 2: Length counts
 - » fixed length
 - Method 3: Bit oriented protocols (flags)

3

Character Based Framing



- SYN is synchronous idle
- STX is start text
- ETX is end text
- **Standard character codes such as ASCII and EBCDIC contain special communication characters that cannot appear in data**
- **Entire transmission is based on a character code**

4

Issues with Character Based Framing

- Character code dependent
 - How do you send binary data?
- Frames must be integer number of characters
- Errors in control characters are messy

NOTE: Primary Framing method from 1960 to ~1975

Length field approach (DECNET)

- Use a header field to give the length of the frame (in bits or bytes)
 - Receiver can count until the end of the frame to find the start of the next frame
 - Receiver looks at the respective length field in the next packet header to find that packet's length
- Length field must be $\log_2(\text{Max_Size_Packet}) + 1$ bits long
 - This restricts the packet size to be used
- Issues with length counts
 - Difficult to recover from errors
 - Resynchronization is needed after an error in the length count

Fixed Length Packets (e.g., ATM)

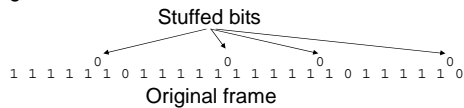
- All packets are of the same size
 - In ATM networks all packets are 53 Bytes
- Requires synchronization upon initialization
- Issues:
 - Message lengths are not multiples of packet size
 - Last packet of a message must contain idle fill (efficiency)
 - Synchronization issues
 - Fragmentation and re-assembly is complicated at high rates

Bit Oriented Framing (Flags)

- A flag is some fixed string of bits to indicate the start and end of a packet
 - A single flag can be used to indicate both the start and the end of a packet
- In principle, any string could be used, but appearance of flag must be prevented somehow in data
 - Standard protocols use the 8-bit string 01111110 as a flag
 - Use 01111111..1110 (<16 bits) as abort under error conditions
 - Constant flags or 1's is considered an idle state
- Thus 01111111 is the actual bit string that must not appear in data
- INVENTED ~ 1970 by IBM for SDLC (synchronous data link protocol)

BIT STUFFING (Transmitter)

- Used to remove flag from original data
- A 0 is stuffed after each consecutive five 1's in the original frame



- Why is it necessary to stuff a 0 in 0111110?
 - If not, then 01111101111 -> 0111110111 0111111111 -> 0111110111
 - How do you differentiate at the receiver?

DESTUFFING (Receiver)

- If 0 is preceded by 011111 in bit stream, remove it
- If 0 is preceded by 0111111, it is the final bit of the flag.

Example: Bits to be removed are underlined below
1001111101100111011111011001111110
flag

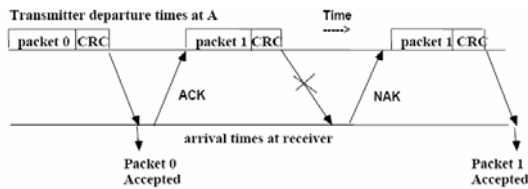
Framing Errors

- All framing techniques are sensitive to errors
 - An error in a length count field causes the frame to be terminated at the wrongpoint (and makes it tricky to find the beginning of the next frame)
 - An error in DLE, STX, or ETX causes the same problems
 - An error in a flag, or a flag created by an error causes a frame to disappear or an extra frame to appear
- Flag approach is least sensitive to errors because a flag will eventually appear again to indicate the end of a next packet
 - Only thing that happens is that an erroneous packet was created
 - This erroneous packet can be removed through an error detection technique
- Error detection & correction: Parity check, Cyclic Redundancy Check (CRC)

Automatic Repeat ReQuest

- When the receiver detects errors in a packet, how does it let the transmitter know to re-send the corresponding packet?
- Systems which automatically request the retransmission of missing packets or packets with errors are called **ARQ systems**.
- Three common schemes
 - Stop & Wait
 - Go Back N
 - Selective Repeat

Pure Stop and Wait Protocol



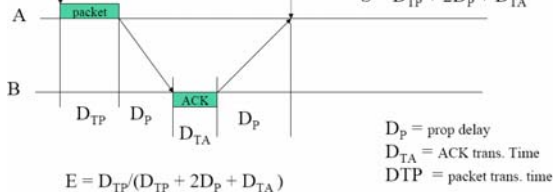
- Problem: Lost Packets
 - Sender will wait forever for an acknowledgement
- Packet may be lost due to framing errors

Efficiency of stop and wait

Let S = total time between the transmission of a packet and reception of its ACK

D_{TP} = transmission time of the packet

Efficiency (no errors) = D_{TP}/S
 $S = D_{TP} + 2D_P + D_{TA}$



D_P = prop delay
 D_{TA} = ACK trans. Time
 D_{TP} = packet trans. time

Stop and wait in the presence of errors

Let P = the probability of an error in the transmission of a packet or in its acknowledgment

$$S = D_{TP} + 2D_P + D_{TA}$$

TO = the timeout interval

X = the amount of time that it takes to transmit a packet and receive its ACK. This time accounts for retransmissions due to errors

$$E[X] = S + TO * P / (1 - P), \text{ Efficiency} = D_{TP} / E[X]$$

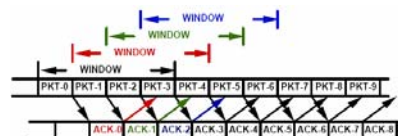
Where,

$TO = D_{TP}$ in a full duplex system

$TO = S$ in a half duplex system

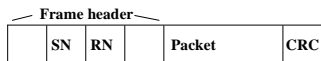
Go Back N ARQ (Sliding Window)

- Stop and Wait is inefficient when propagation delay is larger than the packet transmission time
 - Can only send one packet per round-trip time
- Go Back N allows the transmission of new packets before earlier ones are acknowledged
- Go back N uses a window mechanism where the sender can send packets that are within a "window" (range) of packets
 - The window advances as acknowledgements for earlier packets are received



Features of Go Back N

- Window size = N
 - Sender cannot send packet $i+N$ until it has received the ACK for packet i
- Receiver operates just like in Stop and Wait
 - Receive packets in order
 - Receiver cannot accept packet out of sequence
 - Send $RN = i + 1 \rightarrow$ ACK for all packets up to and including i
- Use of piggybacking
 - When traffic is bi-directional RN's are piggybacked on packets going in the other direction
 - Each packet contains a SN field indicating that packet's sequence number and a RN field acknowledging packets in the other direction



Go Back N ARQ

- The transmitter has a "window" of N packets that can be sent without acknowledgements
- This window ranges from the last value of RN obtained from the receiver (denoted SN_{min}) to $SN_{min}+N-1$
- When the transmitter reaches the end of its window, or times out, it goes back and retransmits packet SN_{min}

Let SN_{min} be the smallest number packet not yet ACKed

Let SN_{max} be the number of the next packet to be accepted from the higher layer (i.e., the next new packet to be transmitted)

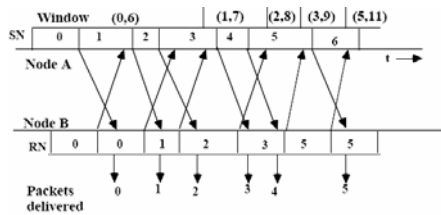
Go Back N: Sender Rules

- $SN_{min} = 0$; $SN_{max} = 0$
- Repeat
 - If $SN_{max} < SN_{min} + N$ (entire window not yet sent)
 - Send packet SN_{max} ;
 - $SN_{max} = SN_{max} + 1$;
 - If packet arrives from receiver with $RN > SN_{min}$
 - $SN_{min} = RN$;
 - If $SN_{min} < SN_{max}$ (there are still some unacknowledged packets) and sender cannot send any new packets
 - Choose some packet between SN_{min} and SN_{max} and re-send it
- The last rule says that when you cannot send any new packets you should re-send an old (not yet ACKed) packet
 - There may be two reasons for not being able to send a new packet
 - Nothing new from higher layer
 - Window expired ($SN_{max} = SN_{min} + N$)
 - No set rule on which packet to re-send
 - Least recently sent

Receiver Rules

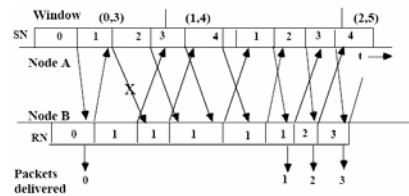
- $RN = 0$;
- Repeat
 - When a good packet arrives, if $SN = RN$
 - Accept packet
 - Increment $RN = RN + 1$
- At regular intervals send an ACK packet with RN
 - Most DLCs send an ACK whenever they receive a packet from the other direction
 - Delayed ACK for piggybacking
- Receiver reject all packets with SN not equal RN
 - However, those packets may still contain useful RN numbers
 - selective repeat: retransmit only those packets that are actually lost (due to errors)

Example of Go Back 7 ARQ



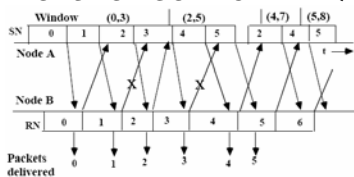
- Note that packet RN-1 must be accepted at B before a frame containing request RN can start transmission at B

RETRANSMISSION BECAUSE OF ERRORS FOR GO ACK 4 ARQ



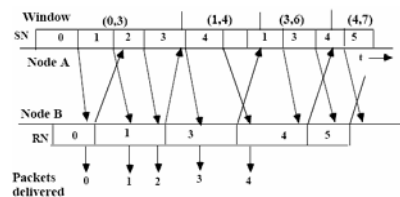
- Note that the timeout value here is take to be the time to send a full window of packets
- Note that entire window has to be retransmitted after an error

RETRANSMISSION DUE TO FEEDBACK ERRORS FOR GO BACK 4 ARQ



- When an error occurs in the reverse direction the ACK may still arrive in time. This is the case here where the packet from B to A with RN=2 arrives in time to prevent retransmission of packet 0
- Packet 2 is retransmitted because RN = 4 did not arrive in time, however it did arrive in time to prevent retransmission of packet 3
 - Was retransmission of packet 4 and 5 really necessary?
 - Strictly no because the window allows transmission of packets 6 and 7 before further retransmissions. However, this is implementation dependent

EFFECT OF LONG FRAMES



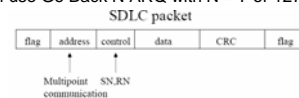
- Long frames in feedback direction slow down the ACKs
 - This causes a transmitter with short frames to wait or go back
- Notice again that the retransmission of packets 3 and 4 was not strictly required because the sender could have sent new packets within the window
 - Again, this is implementation dependent

Notes on Go Back N

- Requires no buffering of packets at the receiver
- Sender must buffer up to N packets while waiting for their ACK
- Sender must re-send entire window in the event of an error
- Packets can be numbered modulo M where $M > N$
 - Because at most N packets can be sent simultaneously
- Receiver can only accept packets in order
 - Receiver must deliver packets in order to higher layer
 - Cannot accept packet $i+1$ before packet i
 - This removes the need for buffering
 - This introduces the need to re-send the entire window upon error
- The major problem with Go Back N is this need to re-send the entire window when an error occurs. This is due to the fact that the receiver can only accept packets in order

Popular DLCs

- Older protocols (used for modems, e.g., xmodem) used stop and wait and simple checksums
- HDLC, LAPB (X.25), and SDLC are almost the same
 - HDLC/SDLC developed by IBM for IBM SNA networks
 - LAPB developed for X.25 networks
 - They all use bit oriented framing with flag = 01111110
 - They all use a 16-bit CRC for error detection
 - They all use Go Back N ARQ with $N = 7$ or 127 (optional)



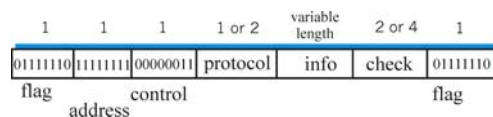
- Another DLC: Point-to-Point Protocol (PPP)

PPP Overview

- one sender, one receiver, one link: easier than broadcast link:
 - no **Media Access Control (MAC)**
 - no need for explicit MAC addressing
 - e.g., dialup link, ISDN line
- **error detection** (no correction)
- **connection liveness**: detect, signal link failure to network layer
- **Error recovery, flow control, data re-ordering all relegated to higher layers!**

PPP Data Frame

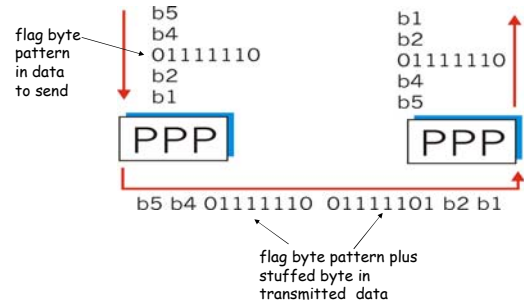
- **Flag**: delimiter (framing)
- **Address**: does nothing (only one option)
- **Control**: does nothing; in the future possible multiple control fields
- **Protocol**: upper layer protocol to which frame delivered (eg, PPP-LCP, IP, IPCP, etc)
- **info**: upper layer data being carried
- **check**: cyclic redundancy check for error detection



Byte Stuffing

- “data transparency” requirement: data field must be allowed to include flag pattern <01111110>
 - **Q:** is received <01111110> data or flag?
- Sender: adds (“stuffs”) extra <01111110> byte after each <01111110> **data** byte
- Receiver:
 - two 01111110 bytes in a row: discard first byte, continue data reception
 - single 01111110: flag byte

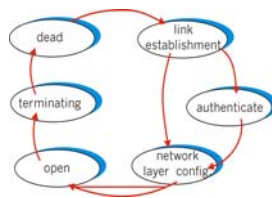
Byte Stuffing



PPP Data Control Protocol

Before exchanging network-layer data, data link peers must

- **configure PPP link** (max. frame length, authentication)
- **learn/configure network layer information**
 - for IP: carry IP Control Protocol (IPCP) msgs (protocol field: 8021) to configure/learn IP address



Homework (Not Mandatory)

1. Understand CRC:
 - A) For the generator string $G=110011$ and data string $M=11100011$ find the CRC and the transmitted string T . (Since G is 6 bits long, $r=5$, and the CRC should be 5 bits long)
 - B) Suppose $G=1001$ and the received $T=1010101$, did any transmission errors occur?
 - C) Suppose $G=101$ and the received $T=1100110$, did any transmission errors occur?
 - D) Suppose $G=1011$ and $M=10010$ Give the shift register implementation of the CRC generator and show the register sequence for generating the CRC with the above value of M .
2. Review all reliability mechanisms provided by DLC. Understand synchronization, ARQ by examples
3. Understand other concepts: packet v.s. circuit switching, layered architectures, connection-oriented v.s. connectionless services, protocols, multiplexing