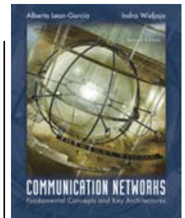
Chapter 5
Peer-to-Peer Protocols and Data Link Layer

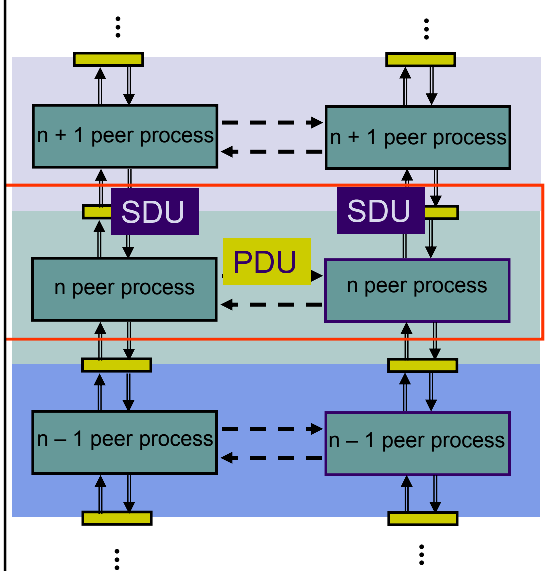


PART I: Peer-to-Peer Protocols
PART II: Data Link Controls

Chapter Overview

- Peer-to-Peer protocols: many protocols involve the interaction between two peers
  - Service Models are discussed & examples given
  - Detailed discussion of ARQ provides example of development of peer-to-peer protocols
  - Flow control, TCP reliable stream, and timing recovery
- Data Link Layer
  - Framing
  - PPP & HDLC protocols
  - Queuing foundations

Peer-to-Peer Protocols



- *Peer-to-Peer processes* execute layer-n protocol to provide service to layer-(n+1)
- Layer-(n+1) peer calls layer-n and passes Service Data Units (SDUs) for transfer
- Layer-n peers exchange Protocol Data Units (PDUs) to effect transfer
- Layer-n delivers SDUs to destination layer-(n+1) peer

Service Models

- The *service model* specifies the information transfer service layer-n provides to layer-(n+1)
- The most important distinction is if the service is:
  - Connection-oriented
  - Connectionless
- Service model possible features:
  - Arbitrary message size or structure
  - Sequencing and Reliability
  - Timing and Flow control
  - Multiplexing
  - Privacy, integrity, and authentication

Reliability & Sequencing

- *Reliability*: what transmission is reliable?
  - *Sequencing*: Are messages or information stream delivered in order?
  - *duplication*
- How to provide reliable communication?
  - Examples: TCP and HDLC
- *ARQ protocols* combine error detection, retransmission, and sequence numbering to provide reliability & sequencing

Flow Control

- What and Why flow control?
  - If destination layer-(n+1) does not retrieve its information fast enough, destination layer-n buffers may overflow
- *Flow Control* provide backpressure mechanisms that control transfer according to availability of buffers at the destination
- Examples: TCP and HDLC

Timing

- Why timing is important in multimedia applications?
  - Applications involving voice and video generate units of information that are related temporally
  - Network transfer introduces delay & jitter
  - Destination application must reconstruct temporal relation in voice/video units
- How to reconstruct timing information?
  - Timing Recovery protocols use *timestamps & sequence numbering* to control the delay & jitter in delivered information
- Examples: RTP & associated protocols in VoIP

Multiplexing

- *Multiplexing* enables multiple layer-(n+1) users to share a layer-n service
- What it needs?
  - A multiplexing tag is required to identify specific users at the destination
- Examples: UDP, IP

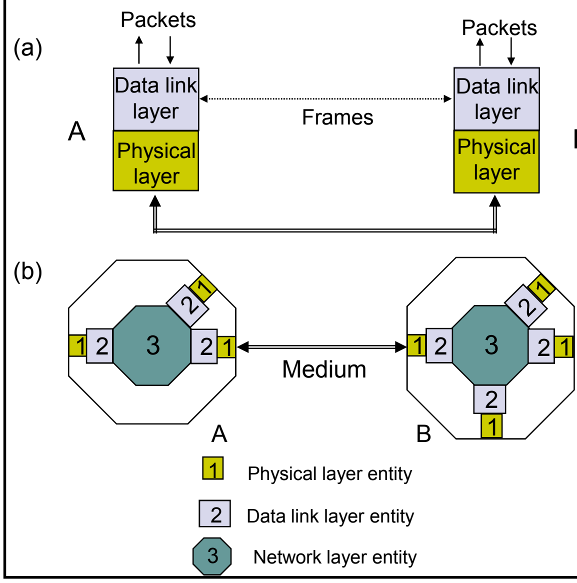
Privacy, Integrity, & Authentication

- *Privacy*: ensuring that information transferred cannot be read by others
- *Integrity*: ensuring that information is not altered during transfer
- *Authentication*: verifying that sender and/or receiver are who they claim to be
- *Security protocols* provide these services and are discussed in Chapter 11
- Examples: IPSec, SSL

End-to-End vs. Hop-by-Hop

- A service feature can be provided by implementing a protocol
  - end-to-end across the network
  - across every hop in the network
- Example:
  - Perform error control at every hop in the network or only between the source and destination?
  - Perform flow control between every hop in the network or only between source & destination?
- We next consider the tradeoffs between the two approaches

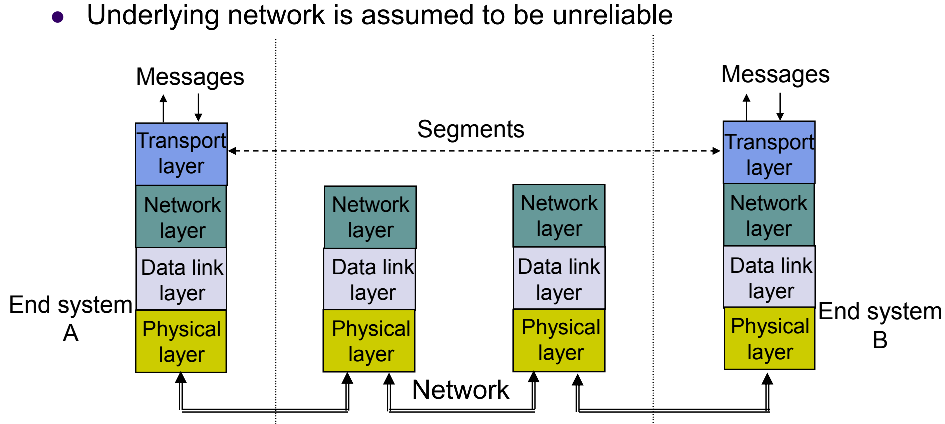
Error control in Data Link Layer



- Data Link operates over *wire-like*, directly-connected systems
- Frames can be corrupted or lost, but arrive in order
- Data link performs error-checking & retransmission
- Ensures error-free packet transfer between two systems

Error Control in Transport Layer

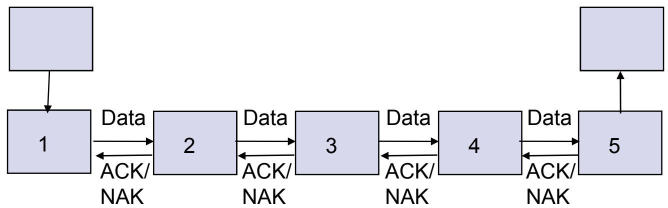
- Transport layer protocol (e.g. TCP) sends segments across network and performs end-to-end error checking & retransmission
- Underlying network is assumed to be unreliable



Can segments experience long delays, be lost, out-of-order? Why?

Which Approach Preferred

Hop-by-hop (HDLC)

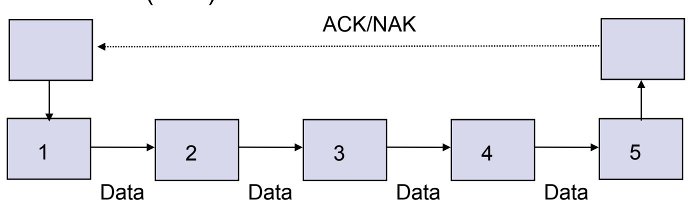


Hop-by-hop cannot ensure E2E correctness

Faster recovery

In situations with infrequent or frequent errors, which one?

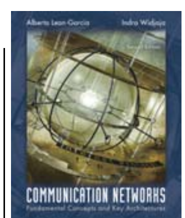
End-to-end (TCP)



Simple inside the network

More scalable if complexity at the edge

Chapter 5
Peer-to-Peer Protocols and Data Link Layer



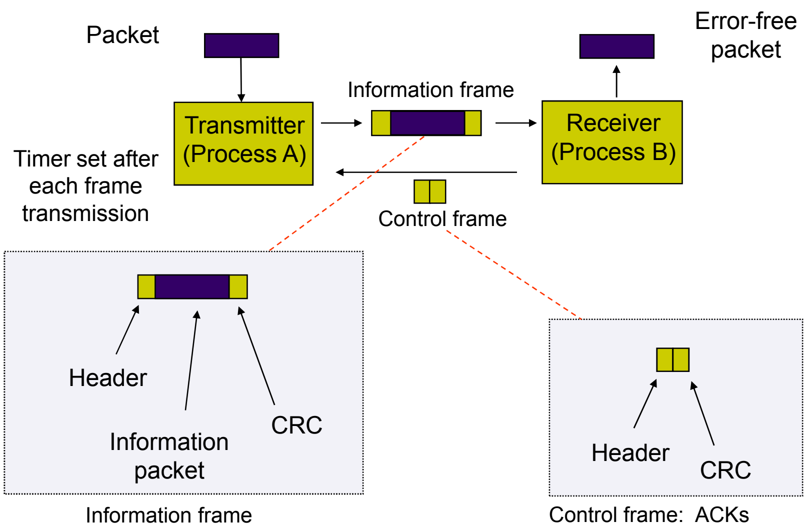
ARQ Protocols and Reliable Data Transfer

Automatic Repeat Request (ARQ)

- *Purpose*: to ensure a sequence of information packets is delivered in order and without errors or duplications despite transmission errors & losses
- Sliding window: a set of Seq.# corresponding to frames permitted to send / receive.
- We will look at:
  - Stop-and-Wait ARQ
  - Go-Back N ARQ
  - Selective Repeat ARQ
- Basic elements of ARQ:
  - *Error-detecting code* with high error coverage
  - *ACKs* (positive acknowledgments)
  - *NAKs* (negative acknowledgments)
  - *Timeout mechanism*

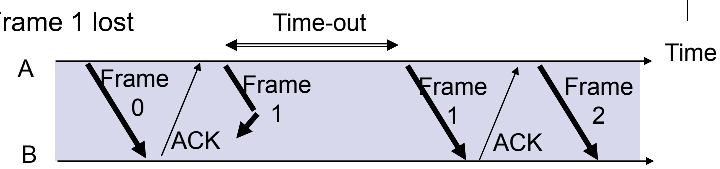
Stop-and-Wait ARQ

Transmit a frame, wait for ACK

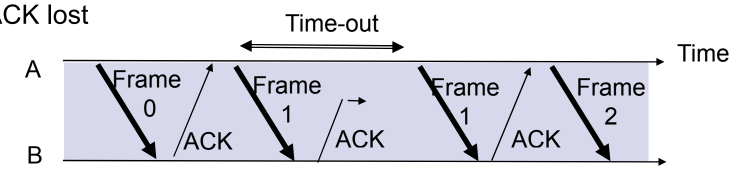


Need for Sequence Numbers

(a) Frame 1 lost



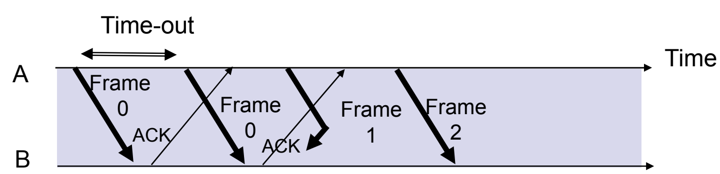
(b) ACK lost



- In cases (a) & (b) the transmitting station A acts the same way
- But in case (b) the receiving station B accepts frame 1 twice
- Question: How is the receiver to know the second frame is also frame 1?
- Answer: ***Add frame sequence number in header***
- S_{last} is sequence number of most recent transmitted frame

Sequence Numbers

(c) Premature Time-out



- The transmitting station A misinterprets duplicate ACKs
- Incorrectly assumes second ACK acknowledges Frame 1
- Question: How is the receiver to know second ACK is for frame 0?
- Answer: ***Add frame sequence number in ACK header***
- R_{next} is sequence number of next frame expected by the receiver (proactive ACK scheme)

Q: What is the minimum # of bits required for the Seq#?

Stop-and-Wait ARQ

Transmitter

Ready state

- Await request from higher layer for packet transfer
- When request arrives, transmit frame with updated S_{last} and CRC
- Go to Wait State

Wait state

- Wait for ACK or timer to expire; block requests from higher layer
- If timeout expires
  - retransmit frame and reset timer
- If ACK received:
  - If sequence number is incorrect or if errors detected: ignore ACK
  - If sequence number is correct ($R_{next} = S_{last} + 1$): accept frame, go to Ready state

Receiver

Always in Ready State

- Wait for arrival of new frame
- When frame arrives, check for errors
- If no errors detected and sequence number is correct ($S_{last}=R_{next}$), then
  - accept frame,
  - update R_{next}, ⟵ How?
  - send ACK frame with R_{next},
  - deliver packet to higher layer
- If no errors detected and wrong sequence number
  - discard frame
  - send ACK frame with R_{next}
- If errors detected
  - discard frame

Can the protocol accept out-of-order frames? Why?

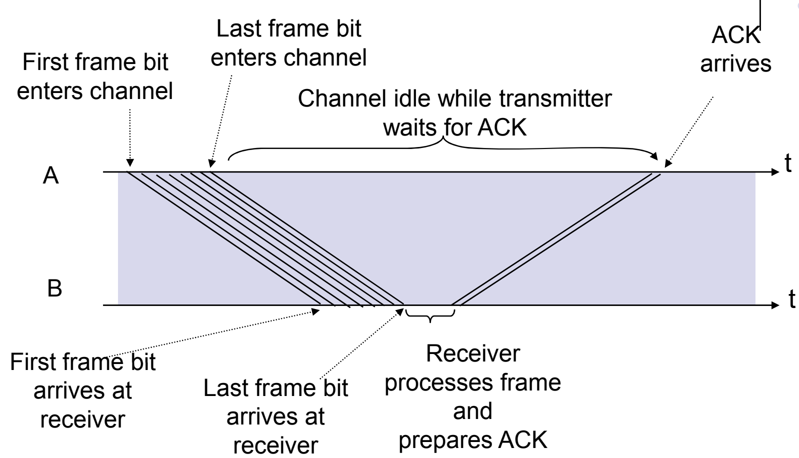
Applications of Stop-and-Wait ARQ

- IBM *Binary Synchronous Communications protocol* (Bisync): character-oriented data link control
- *Xmodem*: modem file transfer protocol
- *Trivial File Transfer Protocol* (RFC 1350): simple protocol for file transfer over UDP

S&W ARQ Performance

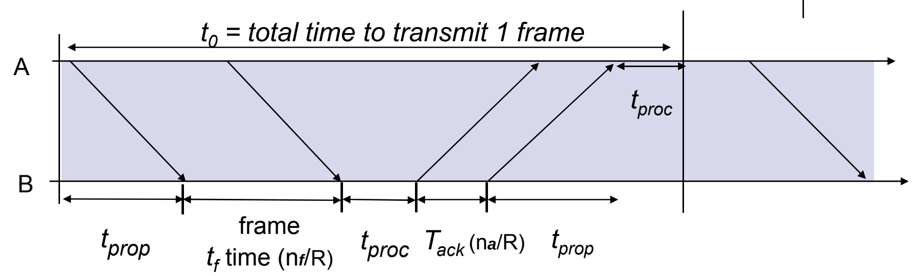
- Q1: consider a 50-kbps channel with a 500-msec round-trip delay. Frame size is 1000-bit. What is the best bandwidth utilization (efficiency) of the one-bit sliding window protocol (stop-and-wait)?
- Q2: A channel has a bit rate of 10 kbps and a propagation delay of 40 msec. For what range of frame sizes does 1-bit sliding window give an bandwidth utilization (efficiency) of at least 50%? (*L*/(*L*+bR))

Stop-and-Wait Efficiency



- 10000 bit frame @ 1 Mbps takes 10 ms to transmit
- If wait for ACK = 1 ms, then efficiency = 10/11= 91%
- If wait for ACK = 20 ms, then efficiency =10/30 = 33%

Stop-and-Wait Model



$$t_0 = 2t_{prop} + 2t_{proc} + t_f + t_{ack}$$

bits/info frame

$$= 2t_{prop} + 2t_{proc} + \frac{n_f}{R} + \frac{n_a}{R}$$

bits/ACK frame

channel transmission rate

S&W Efficiency on Error-free channel

Effective transmission rate:

bits for header & CRC

$$R_{eff}^0 = \frac{\text{number of information bits delivered to destination}}{\text{total time required to deliver the information bits}} = \frac{n_f - n_o}{t_0},$$

Transmission efficiency:

$$\eta_0 = \frac{R_{eff}}{R} = \frac{\frac{n_f - n_o}{t_0}}{R} = \frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}}.$$

Effect of frame overhead

Effect of ACK frame

Effect of ***Delay-Bandwidth Product***

Example: Impact of Delay-Bandwidth Product

n_f=1250 bytes = 10000 bits, n_a=n_o=25 bytes = 200 bits

2xDelayxBW Efficiency	1 ms 200 km (RTT dist.)	10 ms 2000 km	100 ms 20000 km	1 sec 200000 km
1 Mbps	10^3 88%	10^4 49%	10^5 9%	10^6 1%
1 Gbps	10^6 1%	10^7 0.1%	10^8 0.01%	10^9 0.001%

Stop-and-Wait does not work well for very high speeds or long propagation delays

S&W Efficiency in Channel with Errors

- Let 1 – P_f = probability frame arrives w/o errors
- Avg. # of transmissions to first correct arrival is then 1/ (1–P_f)
- "If 1-in-10 get through without error, then avg. 10 tries to success"
- Avg. Total Time per frame is then $t_0/(1 - P_f)$

$$\eta_{SW} = \frac{R_{eff}}{R} = \frac{\dfrac{n_f - n_o}{t_0 / (1-P_f)}}{R} = \frac{1 - \dfrac{n_o}{n_f}}{1 + \dfrac{n_a}{n_f} + \dfrac{2(t_{prop} + t_{proc})R}{n_f}}(1 - P_f)$$

Effect of frame loss

Example: Impact Bit Error Rate

n_f=1250 bytes = 10000 bits, n_a=n_o=25 bytes = 200 bits
Find efficiency for random bit errors with p=0, 10^{-6}, 10^{-5}, 10^{-4}

$$1 - P_f = (1-p)^{n_f} \approx e^{-n_f p} \text{ for large } n_f \text{ and small } p$$

$1 - P_f$ **Efficiency**	0	10^{-6}	10^{-5}	10^{-4}
1 Mbps & 1 ms	1 88%	0.99 86.6%	0.905 79.2%	0.368 32.2%

Bit errors impact performance as $n_f p$ *approach 1*

S&W Performance Summary

- General model: Given: the channel capacity *b* bits/sec, the frame size *L* bits, the round-trip propagation time *R* sec. What is the time required to transmit a single frame? And what is the line utilization?

  The combination of a long transit time (high RTT), high bandwidth, and short frame length gives bad efficiency to the protocol.

  Piggybacking: why not have a free ride of ACK?

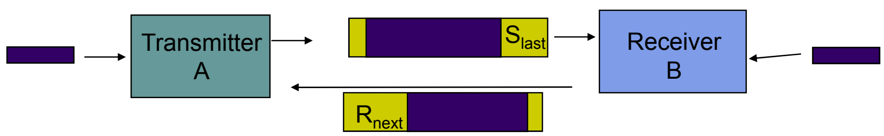
  Pipelining: why not sending more frames before ACKs back?

  What is the price of pipelining, compared to S&W ARQ?

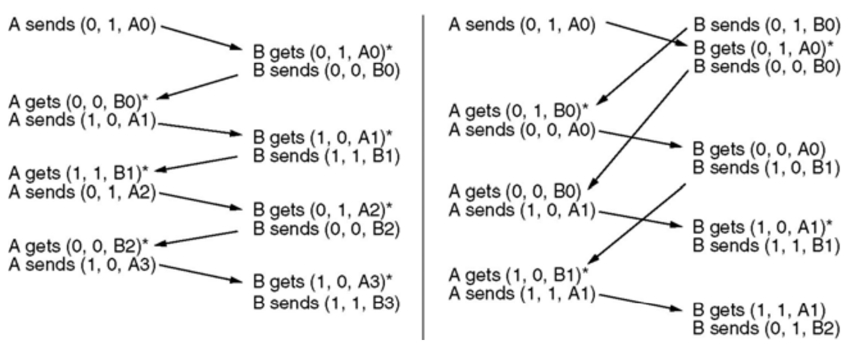
  What is the window size required if pipelining?

Piggybacking

- Since in the two-way transmission, data frames and ACK frames are interleaving, why not have a free ride of ACK upon a data delivering?
  - How a receiver can tell if the frame is data or ACK?
  - For piggybacking, how long should the data link layer wait for a packet onto which to piggyback the ACK?



Piggybacking Example



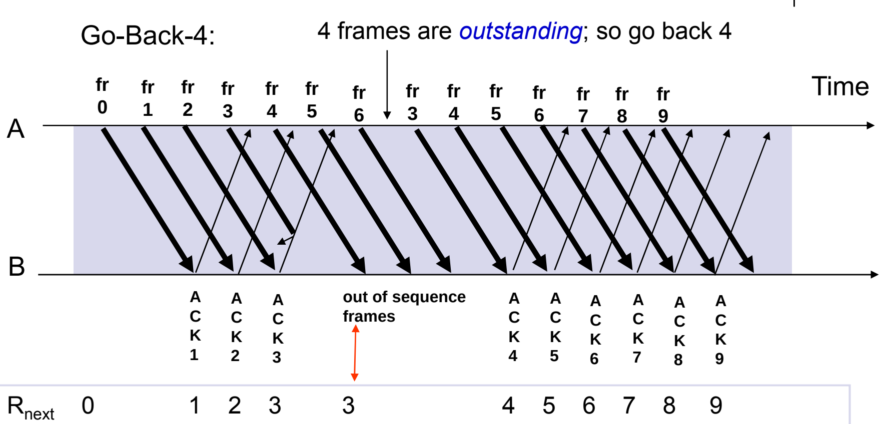
Two scenarios for protocol 4. (l) Normal case. (r) Abnormal case (simultaneous sending). The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

What is the problem with the case (b)?

Go-Back-N

- Improve Stop-and-Wait by not waiting!
- Keep channel busy by continuing to send frames
- Allow a window of up to W_s outstanding frames
  - Receiver's window size is often 1
- Use *m*-bit sequence numbering
- If ACK for oldest frame arrives before window is exhausted, we can continue transmitting
- If window is exhausted, pull back and retransmit all outstanding frames
- Alternative: Use timeout

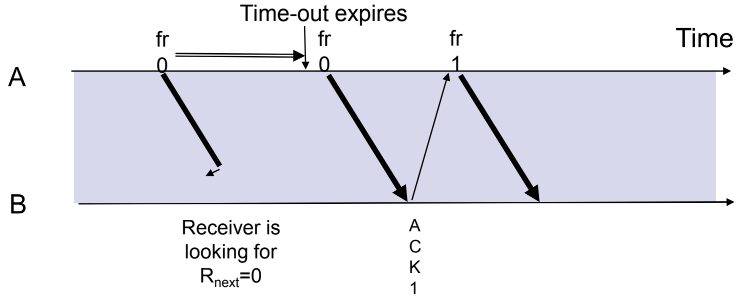
Go-Back-N ARQ



- Frame transmission are *pipelined* to keep the channel busy
- Frame with errors and subsequent out-of-sequence frames are ignored
- Transmitter is forced to go back when window of 4 is exhausted

Go-Back-N with Timeout

- Problem with Go-Back-N as presented:
  - Window size should be long enough to cover round trip time
  - If a frame is lost and source does not have a frame to send, then window will not be exhausted and recovery will not commence
- Use a timeout with each frame
  - When timeout expires, resend all outstanding frames



Maximum Window Size

- Given N-bit seq. numbers, what is the maximum number of frames that can be outstanding in "go back N"?

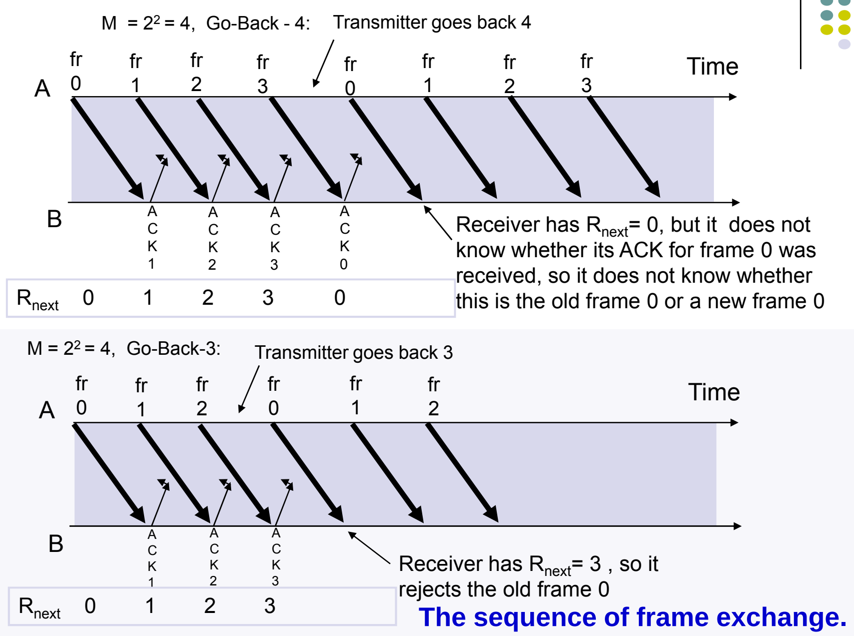
MAX_SEQ = 2^m – 1 while there are 2^m sequence numbers.
Should the maximum number be MAX_SEQ or MAX_SEQ + 1?

Example: m = 2; sequence numbers: 0, 1, 2, 3

1) The sender sends 4 frames 0 through 3
2) A piggybacked ACK for frame 3 eventually back to sender
3) The sender sends another 4 frames (0 through 3)
4) Another piggybacked ACK for frame 3 back to sender

Can the sender tell if all 4 frames in second batch got received or all got lost?

Maximum Allowable Window Size is $W_s = 2^m-1$

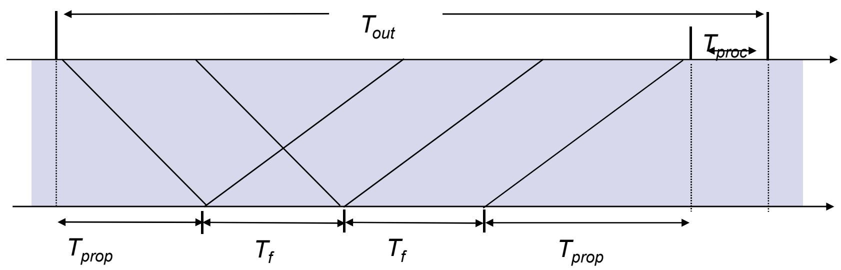


The sequence of frame exchange.

Applications of Go-Back-N ARQ

- *HDLC* (High-Level Data Link Control): bit-oriented data link control
- *V.42 modem*: error control over telephone modem links

Required Timeout & Window Size



- Timeout value should allow for:
  - Two propagation times + two transmission times + 1 processing time: $2\ T_{prop} + 2\ T_f + T_{proc}$
    - *Assume receiver starts transmission right after receiving*
- W_s should be large enough to keep channel busy for T_{out}

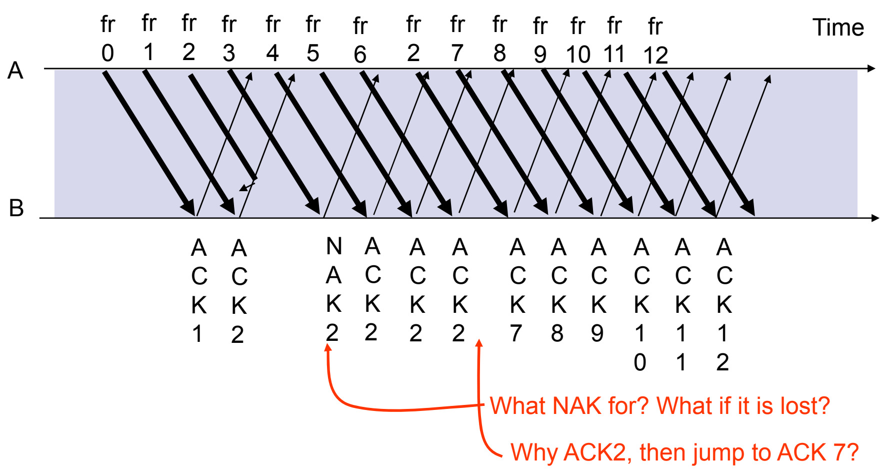
Required Window Size for Delay-Bandwidth Product

Frame = 1250 bytes =10,000 bits, R = 1 Mbps		
Delay: **$2(t_{prop} + t_{proc})$**	**Delay x BW**	**Window** (1 + D * W / L)
1 ms	1000 bits	1
10 ms	10,000 bits	2
100 ms	100,000 bits	11
1 second	1,000,000 bits	101

Selective Repeat ARQ

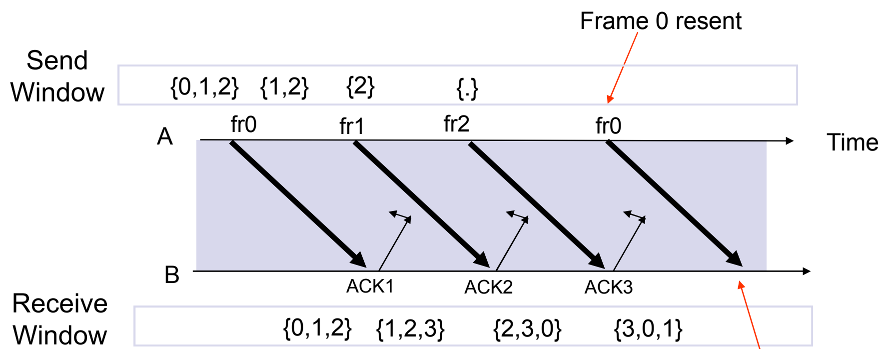
- Why Go-Back-N ARQ inefficient?
  - because *multiple* frames are resent when errors or losses occur
- Selective Repeat retransmits *only an individual frame*
  - Timeout causes individual corresponding frame to be resent
  - NAK causes retransmission of oldest un-acked frame
- Receiver maintains a *receive window* of sequence numbers that can be accepted
  - Error-free, but out-of-sequence frames with sequence numbers within the receive window are buffered
  - Arrival of frame with R_{next} causes window to slide forward by 1 *or more*

Selective Repeat ARQ



What size W_s and W_r allowed?

- Example (as in Go-back N): $M=2^2=4$, $W_s=3$, $W_r=3$



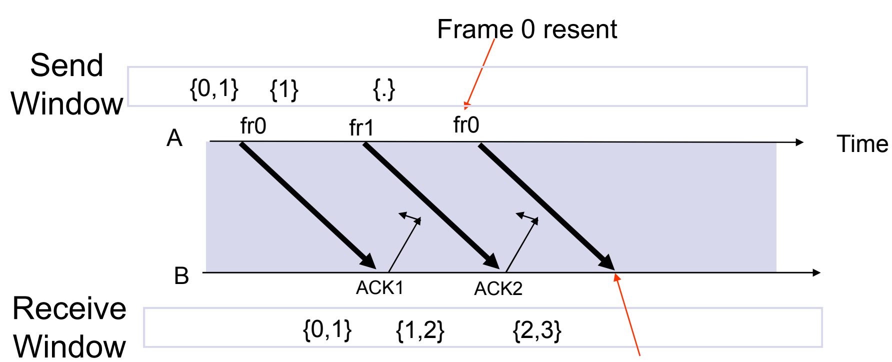
What is the essence of the problem? and what to do?

after the receiver advanced its window, the new range of valid sequence # overlapped with the old one. The receiver cannot distinguish a duplicate from a new frame.

Old frame 0 accepted as a new frame because it falls in the receive window

$W_s + W_r = 2^m$ is maximum allowed

- Example: $M=2^2=4$, $W_s=2$, $W_r=2$

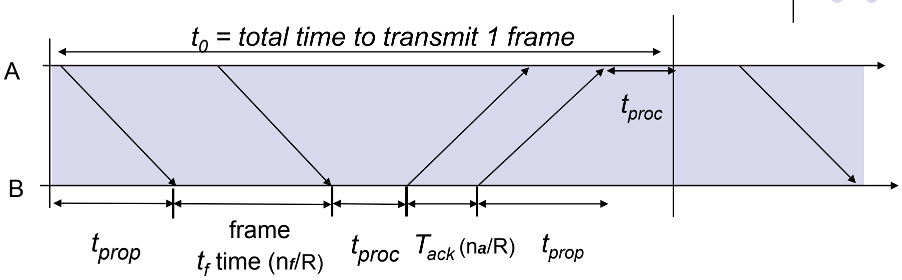


Old frame 0 rejected because it falls outside the receive window

Applications of Sel. Repeat ARQ

- *TCP* (Transmission Control Protocol): transport layer protocol uses variation of selective repeat to provide reliable stream service
- *Service Specific Connection Oriented Protocol*: error control for signaling messages in ATM networks

Efficiency of Selective Repeat



- Assume P_f frame loss probability, then number of transmissions required to deliver a frame is $1/(1-P_f)$:
  - Average transmission time: $t_f/(1-P_f)$

$$\eta_{SR} = \frac{\dfrac{n_f - n_o}{t_f/(1-P_f)}}{R} = \left(1 - \frac{n_o}{n_f}\right)(1-P_f)$$

Example: Impact Bit Error Rate on Selective Repeat

n_f=1250 bytes = 10000 bits, n_a=n_o=25 bytes = 200 bits

Compare S&W, GBN & SR efficiency for random bit errors with p=0, 10^{-6}, 10^{-5}, 10^{-4} and R= 1 Mbps & 100 ms

Efficiency	0	10^{-6}	10^{-5}	10^{-4}
S&W	8.9%	8.8%	8.0%	3.3%
GBN	98%	88.2%	45.4%	4.9%
SR	98%	97%	89%	36%

- *Selective Repeat outperforms GBN and S&W, but efficiency drops as error rate increases*

Comparison of ARQ Efficiencies

Assume n_a and n_o are negligible relative to n_f, and $L = 2(t_{prop}+t_{proc})R/n_f = (W_s-1)$, then

Selective-Repeat:

$$\eta_{SR} = (1-P_f)(1-\frac{n_o}{n_f}) \approx (1-P_f)$$

~~Go-Back-N:~~

For $P_f \approx 0$, SR & GBN same

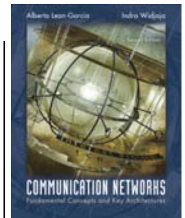
$$\eta_{GBN} = \frac{1-P_f}{1+(W_S-1)P_f} = \frac{1-P_f}{1+LP_f}$$

For $P_f \rightarrow 1$, GBN & SW same

Stop-and-Wait:

$$\eta_{SW} = \frac{(1-P_f)}{1+\frac{n_a}{n_f}+\frac{2(t_{prop}+t_{proc})R}{n_f}} \approx \frac{1-P_f}{1+L}$$

Chapter 5
Peer-to-Peer Protocols and Data Link Layer



PART II: Data Link Controls

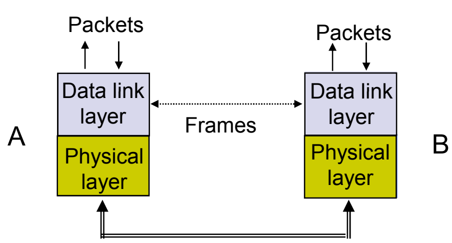
Framing

Point-to-Point Protocol

High-Level Data Link Control

Link Sharing Using Statistical Multiplexing

Data Link Protocols

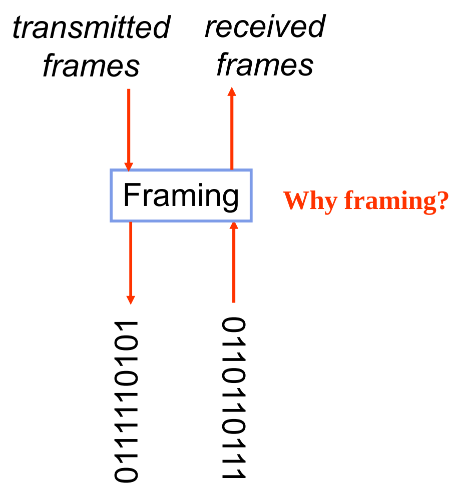


- Directly connected, wire-like
- Losses & errors, but no out-of-sequence frames
- Applications: Direct Links; LANs; Connections across WANs

Data Links Services
- Framing
- Error control
- Flow control
- Multiplexing
- Link Maintenance
- Security: Authentication & Encryption

Examples
- PPP
- HDLC
- Ethernet LAN
- IEEE 802.11 (Wi Fi) LAN

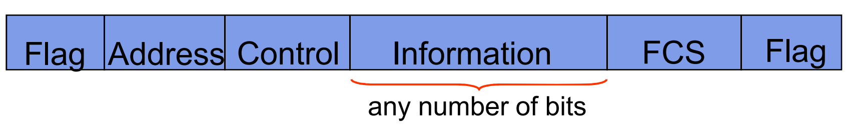
Framing



- Framing: to break the bit stream up into discrete frames and compute the checksum for each frame
  - Mapping stream of physical layer bits into frames
  - Mapping frames into bit stream
- Frame boundaries can be determined using:
  - Control Characters
  - Flags
  - CRC Checks (GFP)

Framing & Bit Stuffing

HDLC frame



- Frame delineated by flag character
- HDLC uses *bit stuffing* to prevent occurrence of flag 01111110 inside the frame
- Transmitter inserts extra 0 after each consecutive five 1s *inside* the frame
- Receiver checks for five consecutive 1s
  - if next bit = 0, it is removed
  - if next two bits are 10, then flag is detected
  - If next two bits are 11, then frame has errors

Example 1

(a) Data to be sent

0110111111111100

After stuffing and framing

*01111110*011011111$\underline{0}$11111$\underline{0}$000*01111110*

(b) Data received

011111101101111101111100001111110

After destuffing and deframing

011011111-11111-00

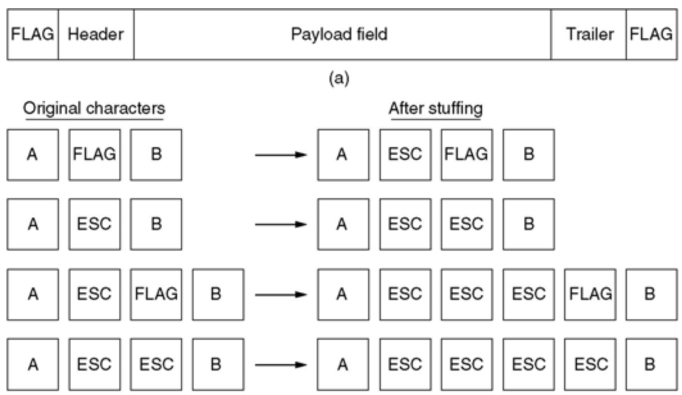
Example 2

Q: How a bit loss be detected in bit stuffing? Can it always be detected?

Flag Byte with Byte Stuffing

- Flag byte starts and ends each frame (used in PPP).

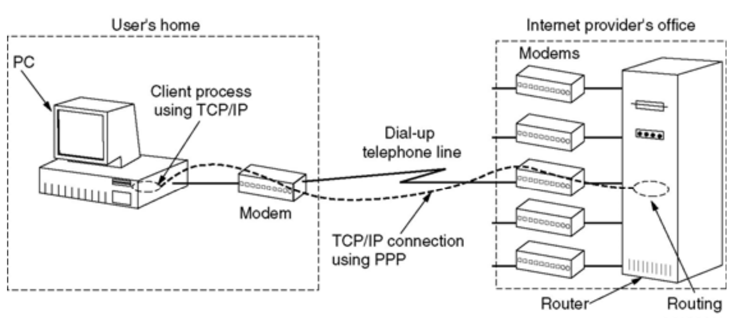
What happens if the payload field contains the flag byte?



(a) A frame delimited by flag bytes.
(b) Four examples of byte sequences before and after stuffing.

PPP: Point-to-Point Protocol

- Data link protocol for point-to-point lines in Internet
  - Router-router; dial-up to router



A home personal computer acting as an internet host.

PPP Functionalities

1. Provides *Framing and Error Detection*
   - Character-oriented HDLC-like frame structure
2. *Link Control Protocol (LCP)*
   - Set up, testing, maintaining, bringing down lines; negotiating options
   - ***Authentication***: key capability in ISP access
3. A family of *Network Control Protocols* (NCP) specific to different network layer protocols
   - IP, OSI network layer, IPX (Novell), Appletalk

PPP Applications

PPP used in many point-to-point applications

- Telephone Modem Links
- Packet over SONET
  - IP→PPP→SONET

- PPP is also used over shared links such as Ethernet to provide LCP, NCP, and authentication features
  - PPP over Ethernet (RFC 2516)
  - Used over DSL

PPP Frame Format

			1 or 2	variable	2 or 4	
Flag 01111110	Address 1111111	Control 00000011	Protocol	Information	CRC	Flag 01111110

integer # of bytes

All stations are to accept the frame

HDLC Unnumbered frame

Specifies what kind of packet is contained in the payload, e.g., LCP, NCP, IP, OSI CLNP, IPX

- PPP uses similar frame structure as HDLC, except
  - Protocol type field
  - Payload contains an *integer* number of bytes
- PPP also uses flag byte 01111110, but uses *byte stuffing*

Byte-Stuffing in PPP

- PPP is character-oriented version of HDLC
- Flag is 0x7E (01111110)
- Control escape 0x7D (01111101)
- Any occurrence of flag or control escape inside of frame is replaced with 0x7D followed by

original octet XORed with 0x20 (00100000)

Data to be sent

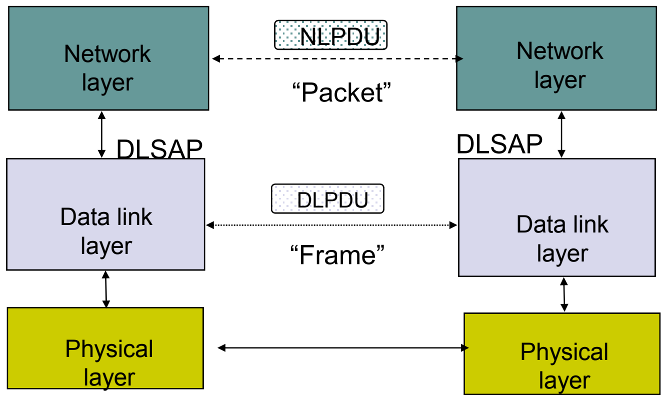
41	7D	42	7E	50	70	46

7E	41	7D	5D	42	7D	5E	50	70	46	7E

After stuffing and framing

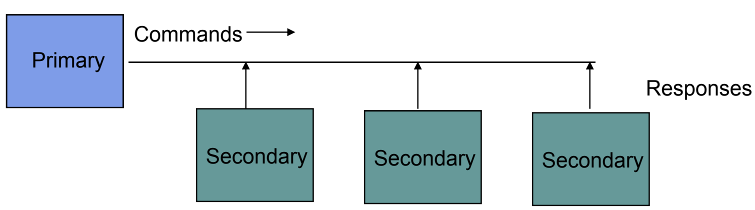
High-Level Data Link Control (HDLC)

- Bit-oriented data link control
  - Derived from IBM Synchronous Data Link Control (SDLC)

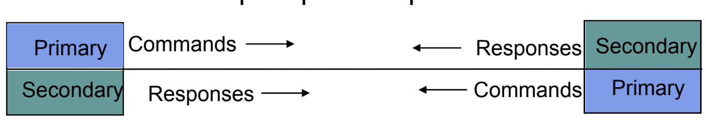


HDLC Data Transfer Modes

- Normal Response Mode
  - Used in polling multidrop lines



- Asynchronous Balanced Mode
  - Used in full-duplex point-to-point links



- Mode is selected during connection establishment

HDLC Frame Format

Flag	Address	Control	Information	FCS	Flag

- Control field gives HDLC its functionality
- Codes in fields have specific meanings and uses
  - Flag: delineate frame boundaries
  - Address: identify *secondary* station (1 or more bytes)
    - In ABM mode, a station can act as primary or secondary so address changes accordingly
  - Control: purpose & functions of frame (1 or 2 bytes)
  - Information: contains user data; length not standardized, but implementations impose maximum
  - Frame Check Sequence: 16- or 32-bit CRC

Error Detection & Loss Recovery

- Frames lost due to loss-of-synch or receiver buffer overflow
- Frames may undergo errors in transmission
- CRCs detect errors and such frames are treated as lost
- Recovery through ACKs, timeouts & retransmission
- Sequence numbering to identify out-of-sequence & duplicate frames
- HDLC provides for options that implement several ARQ methods