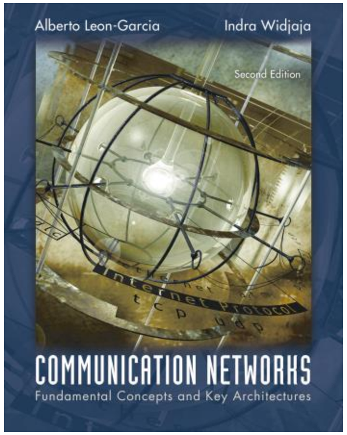
Chapter 5 Peer-to-Peer Protocols and Data Link Layer



PART I: Peer-to-Peer Protocols

Peer-to-Peer Protocols and Service Models

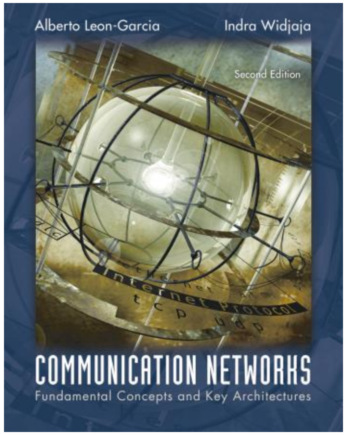
ARQ Protocols and Reliable Data Transfer

Flow Control

Timing Recovery

TCP Reliable Stream Service & Flow Control

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



PART II: Data Link Controls

Framing

Point-to-Point Protocol

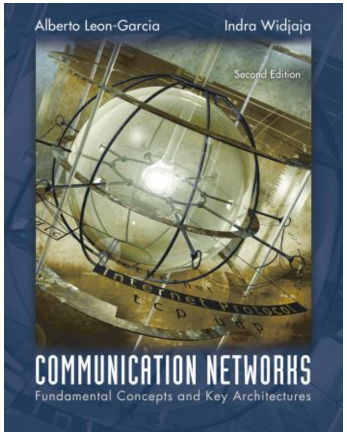
High-Level Data Link Control

*Link Sharing Using Statistical Multiplexing

Chapter Overview

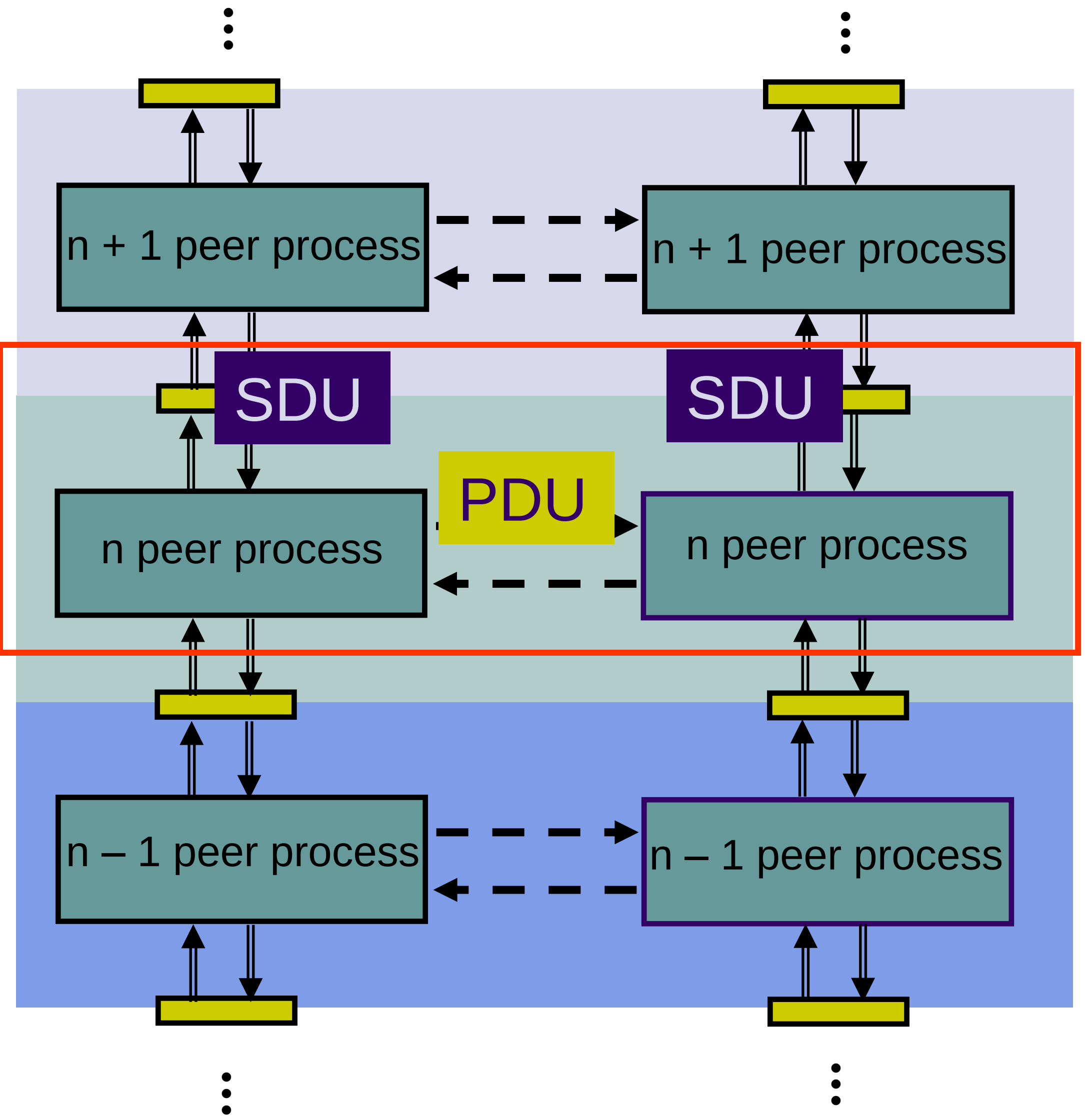
- Peer-to-Peer protocols: many protocols involve the interaction between two peers
  - Service Models are discussed & examples given
  - Detailed discussion of ARQ provides example of development of peer-to-peer protocols
  - Flow control, TCP reliable stream, and timing recovery
- Data Link Layer
 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

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



Peer-to-Peer Protocols and Service Models

Peer-to-Peer Protocols



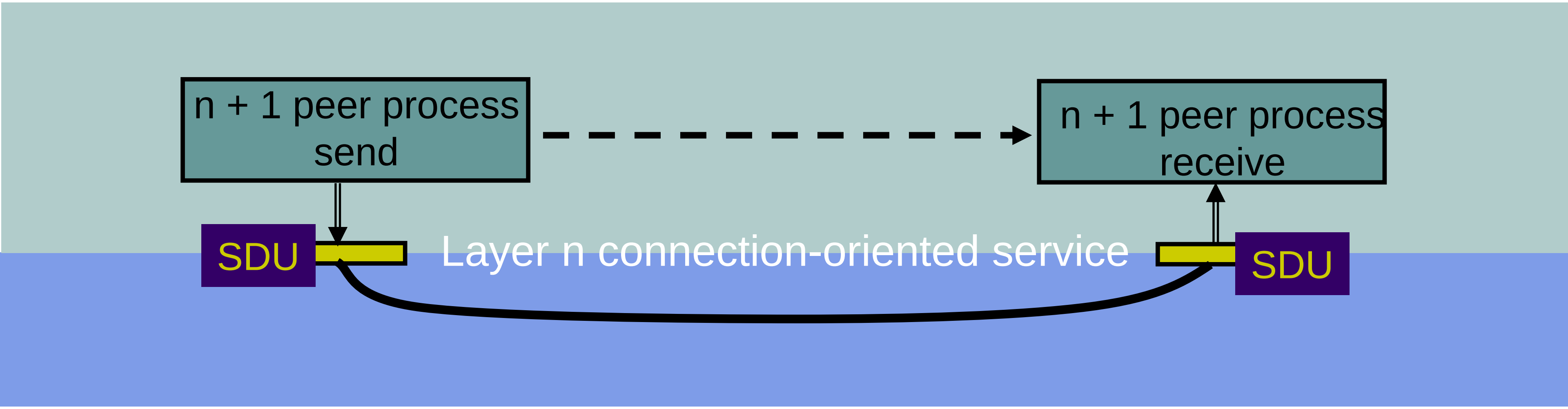
- *Peer-to-Peer processes* execute layer-n protocol to provide service to layer-(n+1)
- Layer-(n+1) peer calls layer-n and passes Service Data Units (SDUs) for transfer
- Layer-n peers exchange Protocol Data Units (PDUs) to effect transfer
- Layer-n delivers SDUs to destination layer-(n+1) peer

Service Models

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

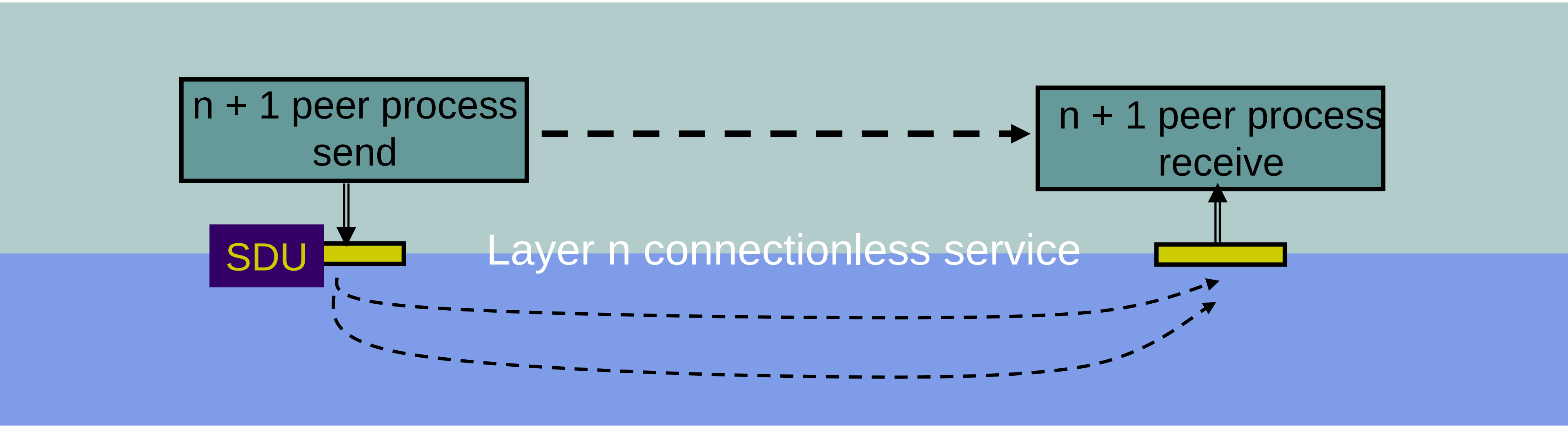
Connection-Oriented Transfer Service

- Connection Establishment
  - Connection must be established between layer-(n+1) peers
  - Layer-n protocol must: Set initial parameters, e.g. sequence numbers; and Allocate resources, e.g. buffers
- Message transfer phase
  - Exchange of SDUs
- Disconnect phase
- Example: TCP, PPP



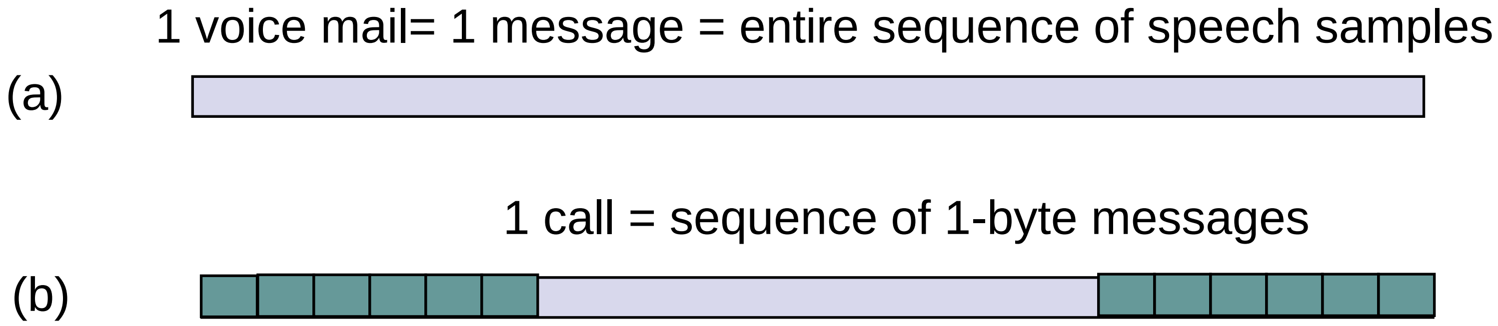
Connectionless Transfer Service

- No Connection setup, simply send SDU
- Each message send independently
- Must provide all address information per message
- Simple & quick
- Example: UDP, IP



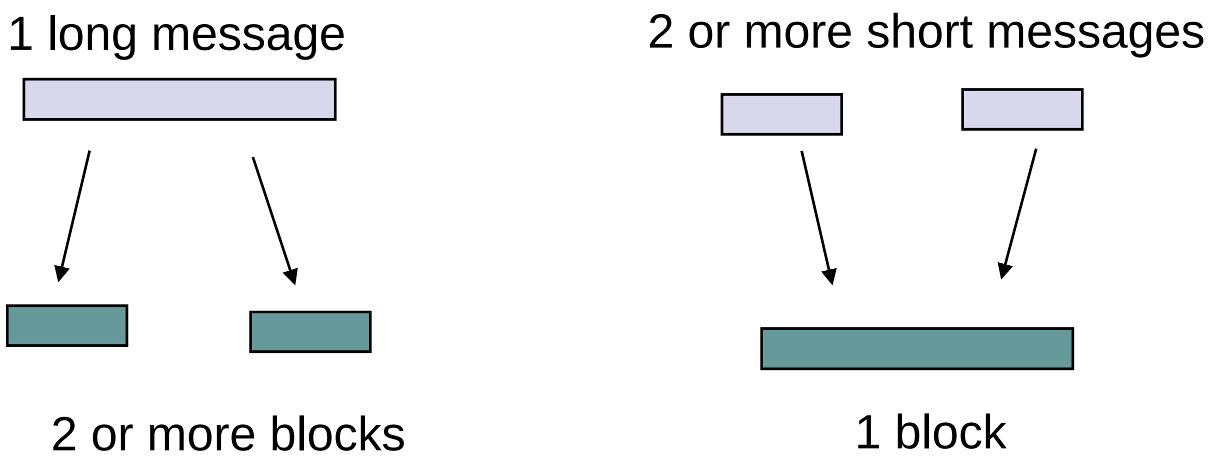
Message Size and Structure

- What message size and structure will a service model accept?
  - Different services impose restrictions on size & structure of data it will transfer
  - Single bit? Block of bytes? Byte stream?
  - Ex: Transfer of voice mail = 1 long message
  - Ex: Transfer of voice call = byte stream



Segmentation & Blocking

- To accommodate arbitrary message size, a layer may have to deal with messages that are too long or too short for its protocol
- *Segmentation & Reassembly*: a layer breaks long messages into smaller blocks and reassembles these at the destination
- *Blocking & Unblocking*: a layer combines small messages into bigger blocks prior to transfer



Reliability & Sequencing

- *Reliability*: Are messages or information stream delivered error-free and without loss or duplication?
- *Sequencing*: Are messages or information stream delivered in order?
- *ARQ protocols* combine error detection, retransmission, and sequence numbering to provide reliability & sequencing
- Examples: TCP and HDLC

Pacing and Flow Control

- Messages can be lost if receiving system does not have sufficient buffering to store arriving messages
- If destination layer-(n+1) does not retrieve its information fast enough, destination layer-n buffers may overflow
- *Pacing & Flow Control* provide backpressure mechanisms that control transfer according to availability of buffers at the destination
- Examples: TCP and HDLC

Timing

- Applications involving voice and video generate units of information that are related temporally
- Destination application must reconstruct temporal relation in voice/video units
- Network transfer introduces delay & jitter
- *Timing Recovery* protocols use *timestamps* & *sequence numbering* to control the delay & jitter in delivered information
- Examples: RTP & associated protocols in Voice over IP

Multiplexing

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

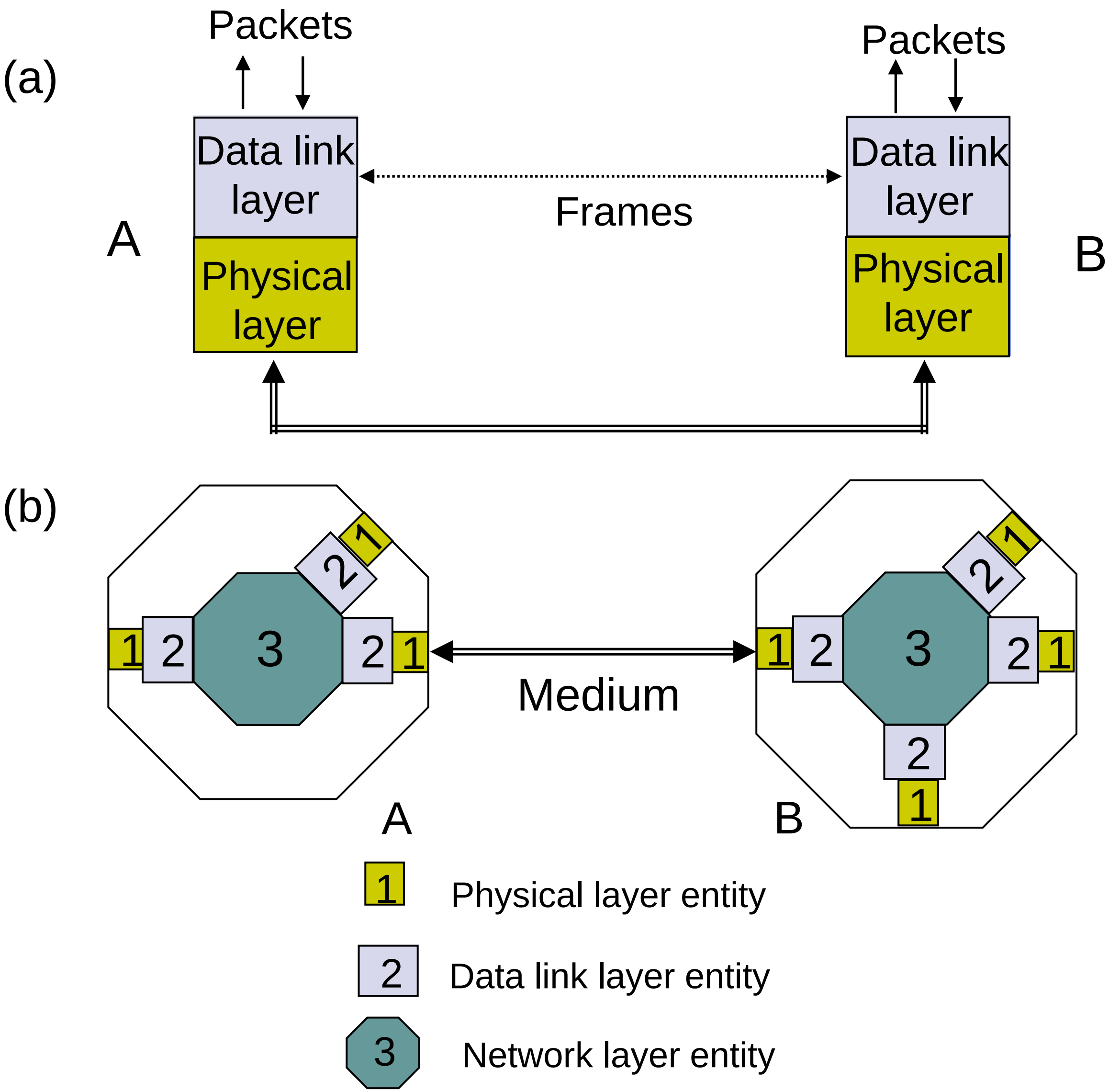
Privacy, Integrity, & Authentication

- *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

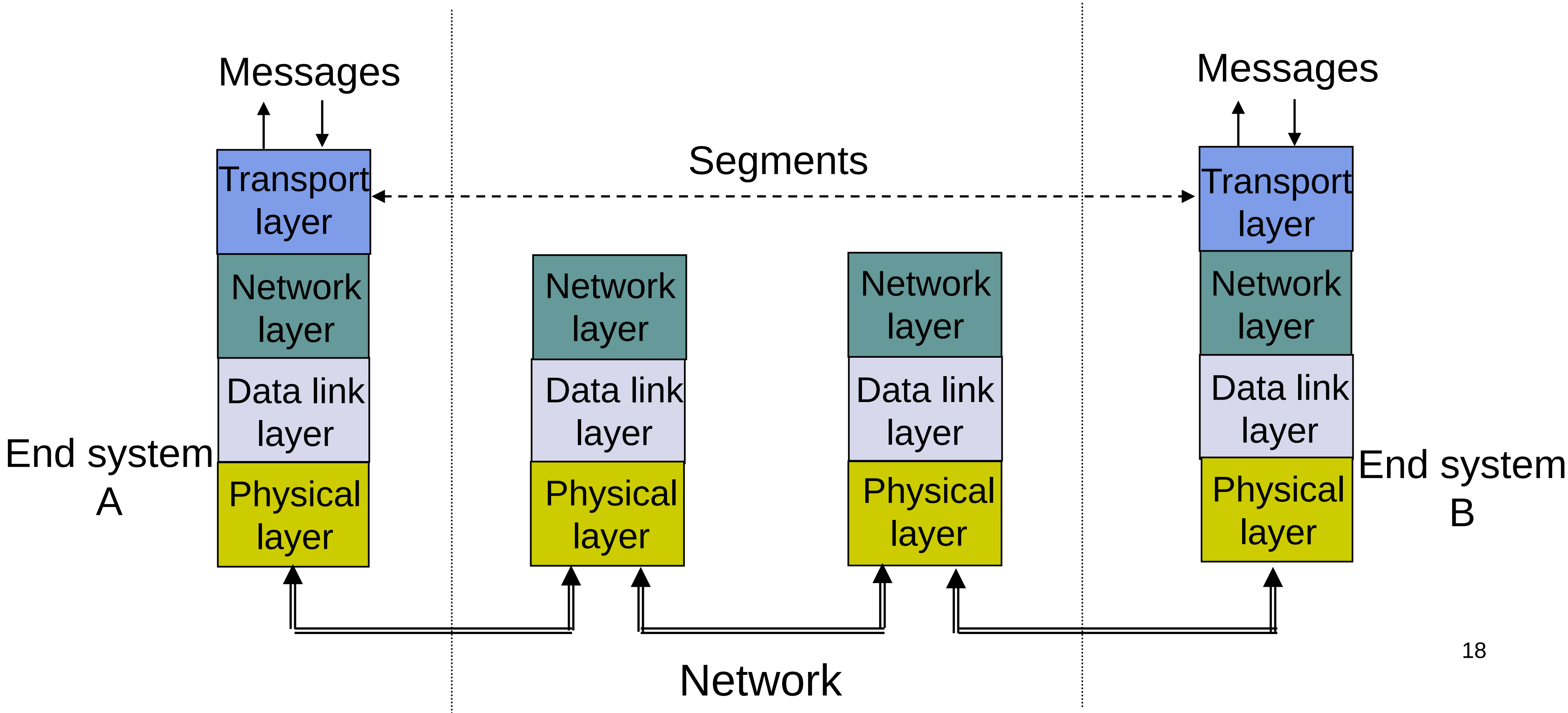
Error control in Data Link Layer



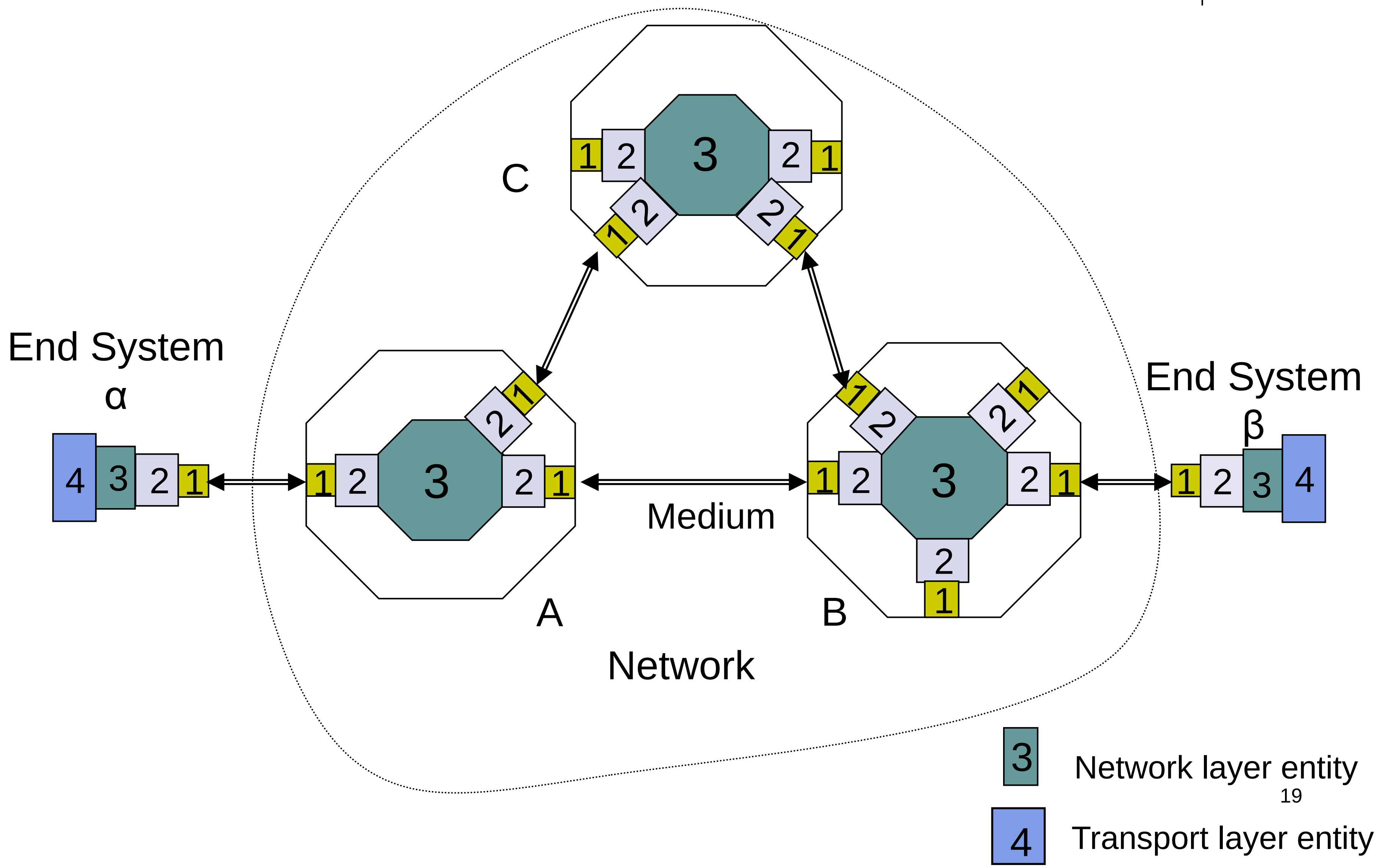
- Data Link operates over wire-like, directly-connected systems
- Frames can be corrupted or lost, but arrive in order
- Data link performs error-checking & retransmission
- Ensures error-free packet transfer between two systems

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

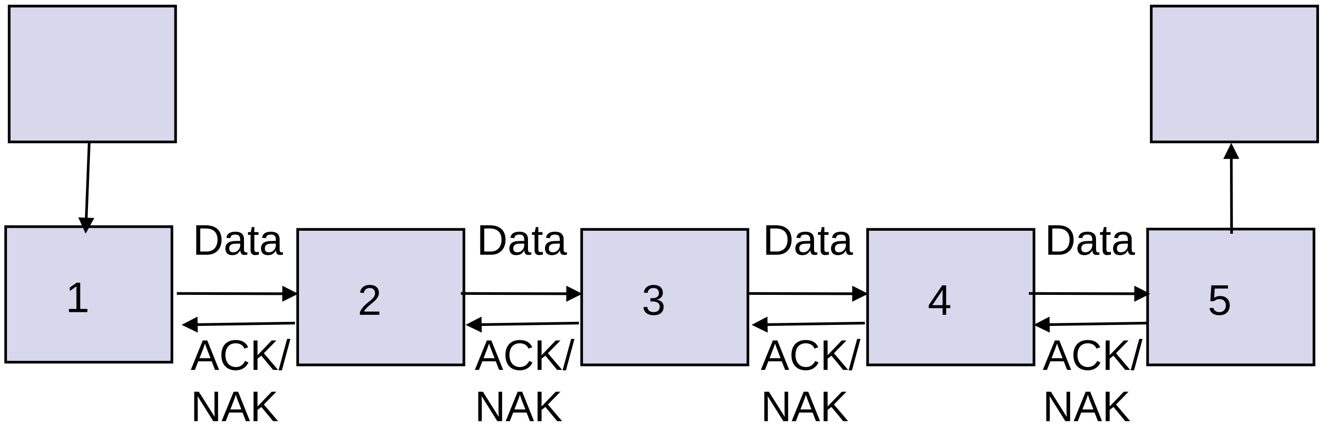


- Segments can experience long delays, can be lost, or arrive out-of-order because packets can follow different paths across network
- End-to-end error control protocol more difficult



End-to-End Approach Preferred

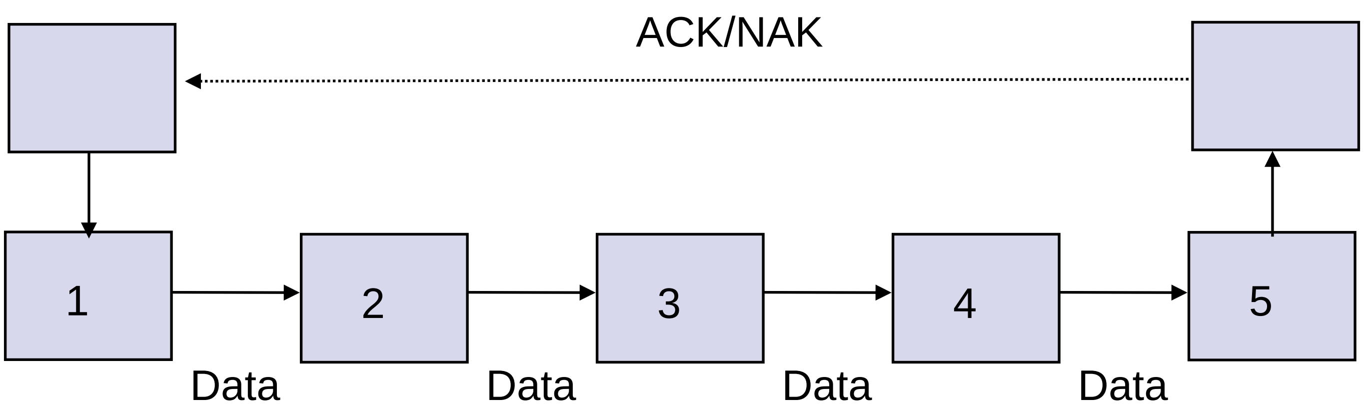
Hop-by-hop



Hop-by-hop cannot ensure E2E correctness

Faster recovery

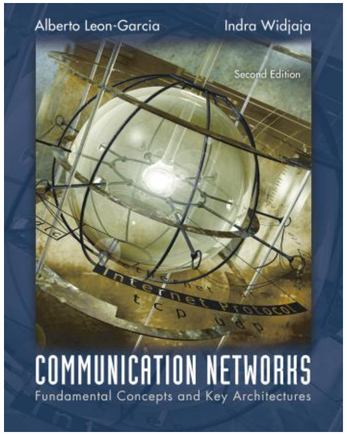
End-to-end



Simple inside the network

More scalable if complexity at the edge

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



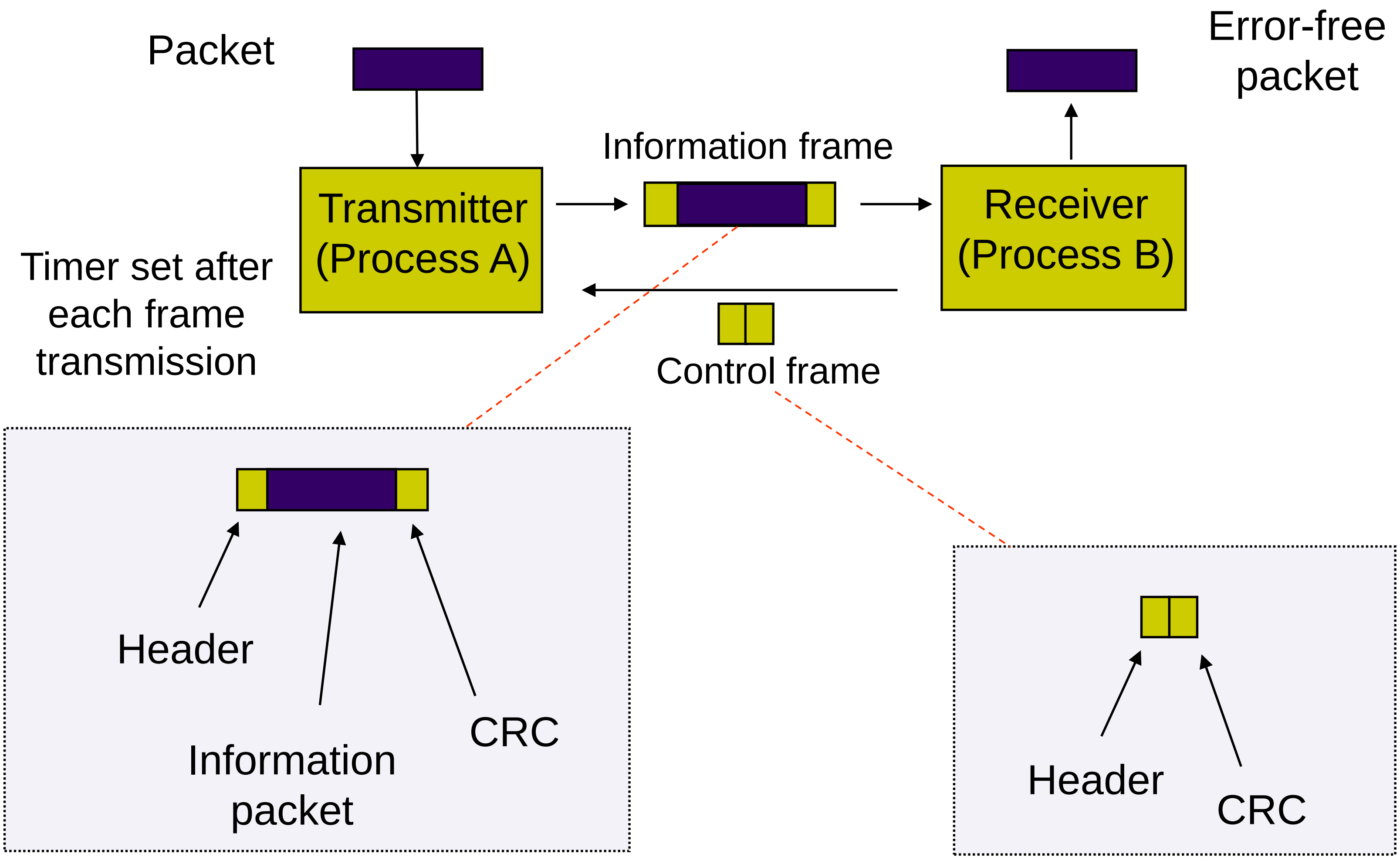
ARQ Protocols and Reliable Data Transfer

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
- 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 acknowlegments)
  - *Timeout mechanism*

Stop-and-Wait ARQ

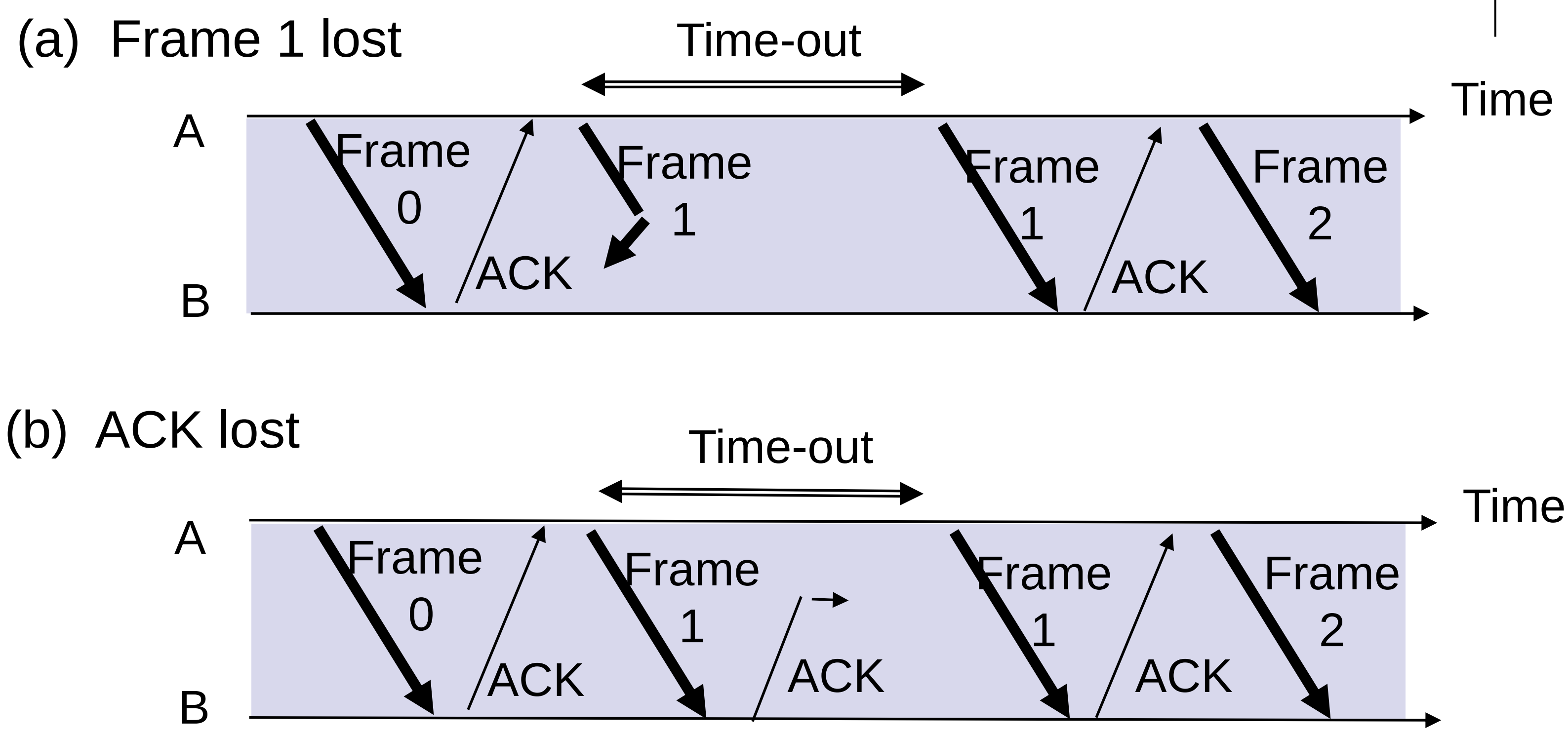
Transmit a frame, wait for ACK



Information frame

Control frame: ACKs

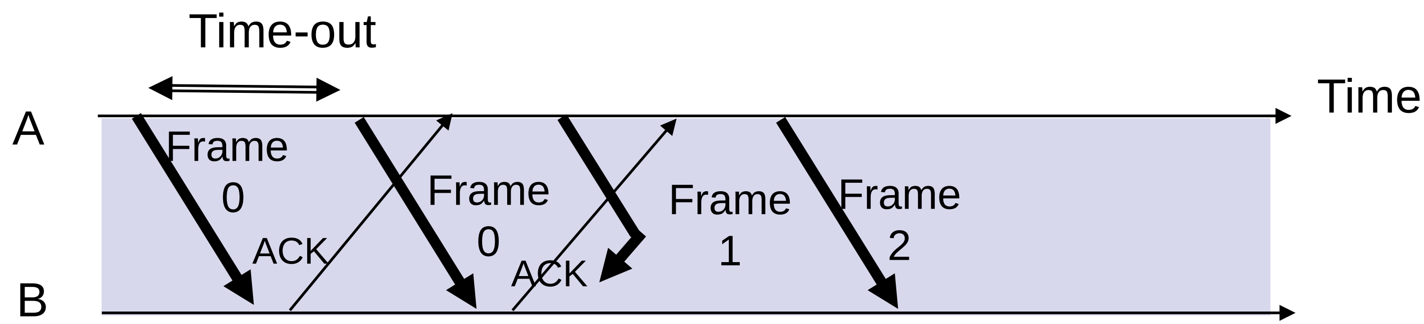
Need for Sequence Numbers



- 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

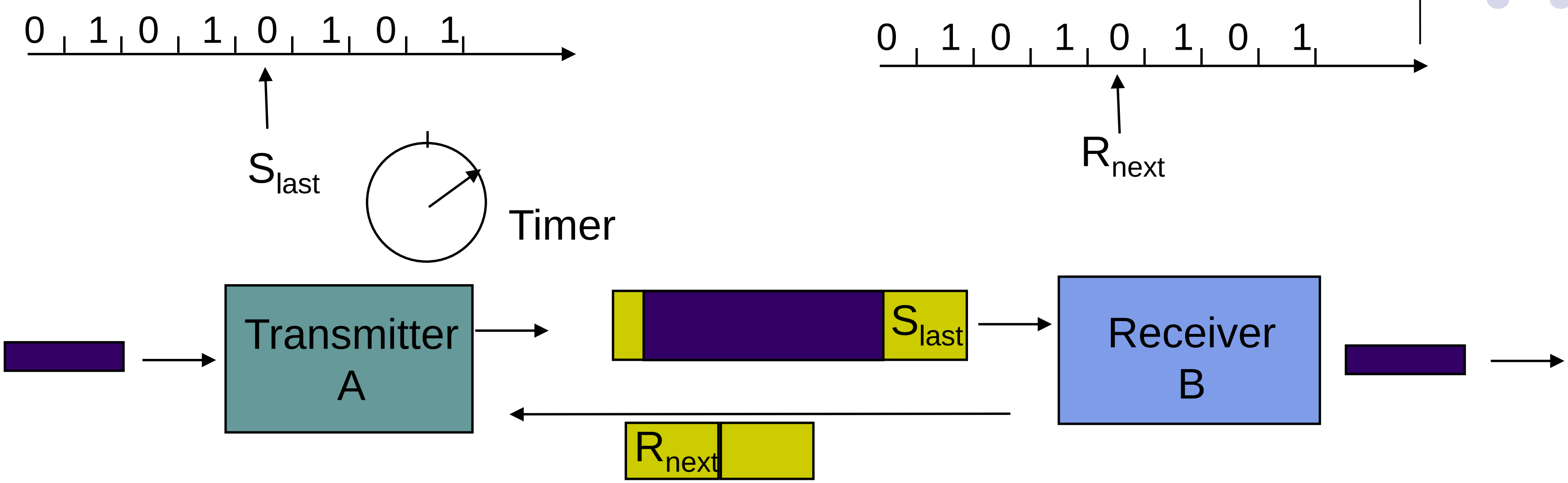
Sequence Numbers

(c) Premature Time-out

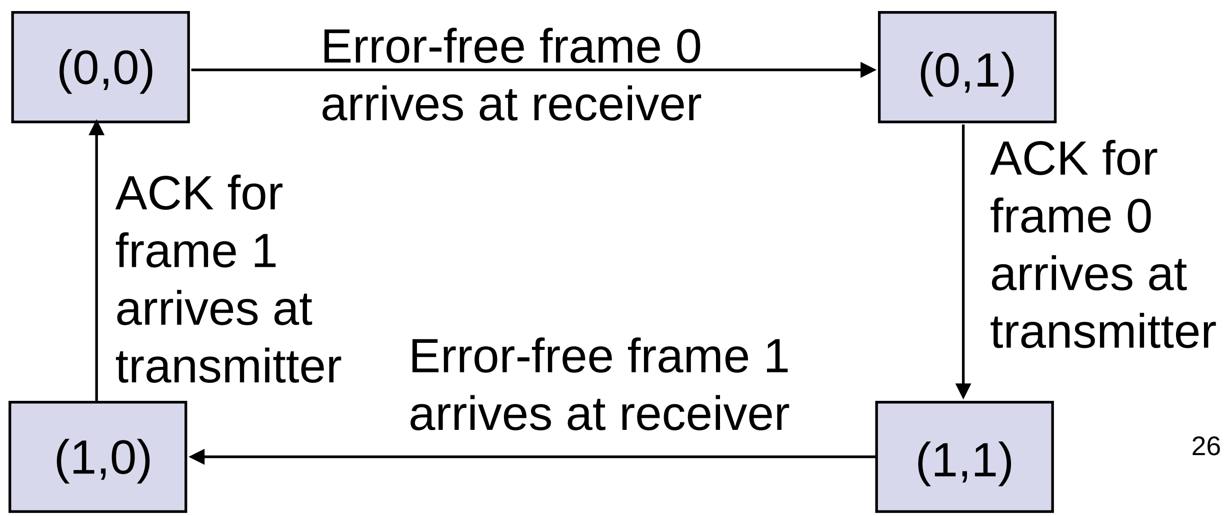


- The transmitting station A misinterprets duplicate ACKs
- Incorrectly assumes second ACK acknowledges Frame 1
- Question: How is the receiver to know second ACK is for frame 0?
- Answer: ***Add frame sequence number in ACK header***
- R_{next} is sequence number of next frame expected by the receiver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

1-Bit Sequence Numbering Suffices



Global State:
(S_{last}, R_{next})



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

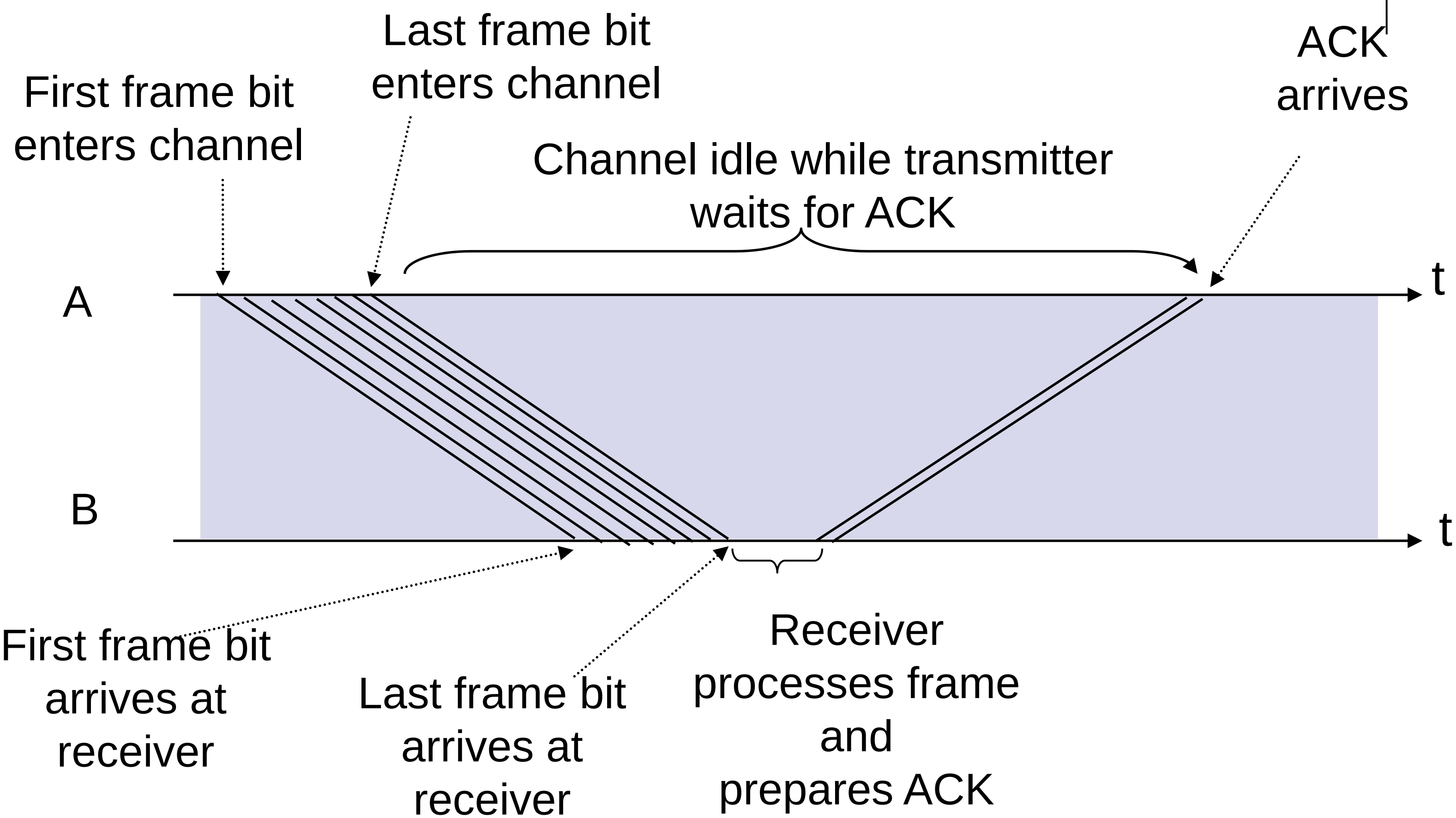
Always in Ready State

- 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},
  - 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

Applications of Stop-and-Wait ARQ

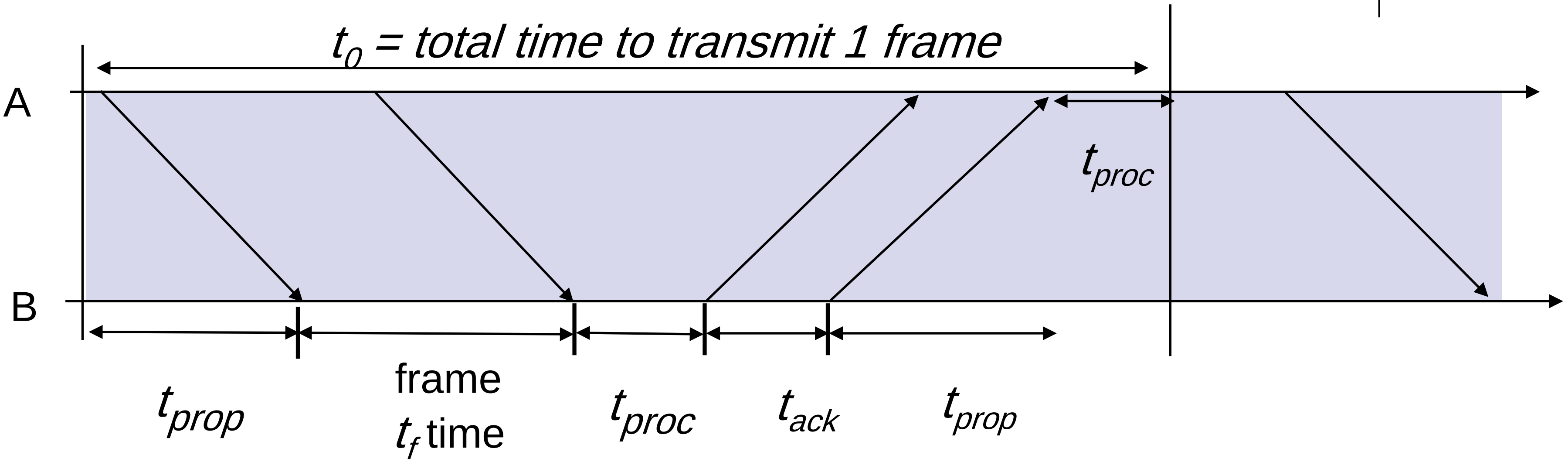
- IBM *Binary Synchronous Communications protocol* (Bisync): character-oriented data link control
- *Xmodem*: modem file transfer protocol
- *Trivial File Transfer Protocol* (RFC 1350): simple protocol for file transfer over UDP

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



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

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

bits/info frame

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	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}

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

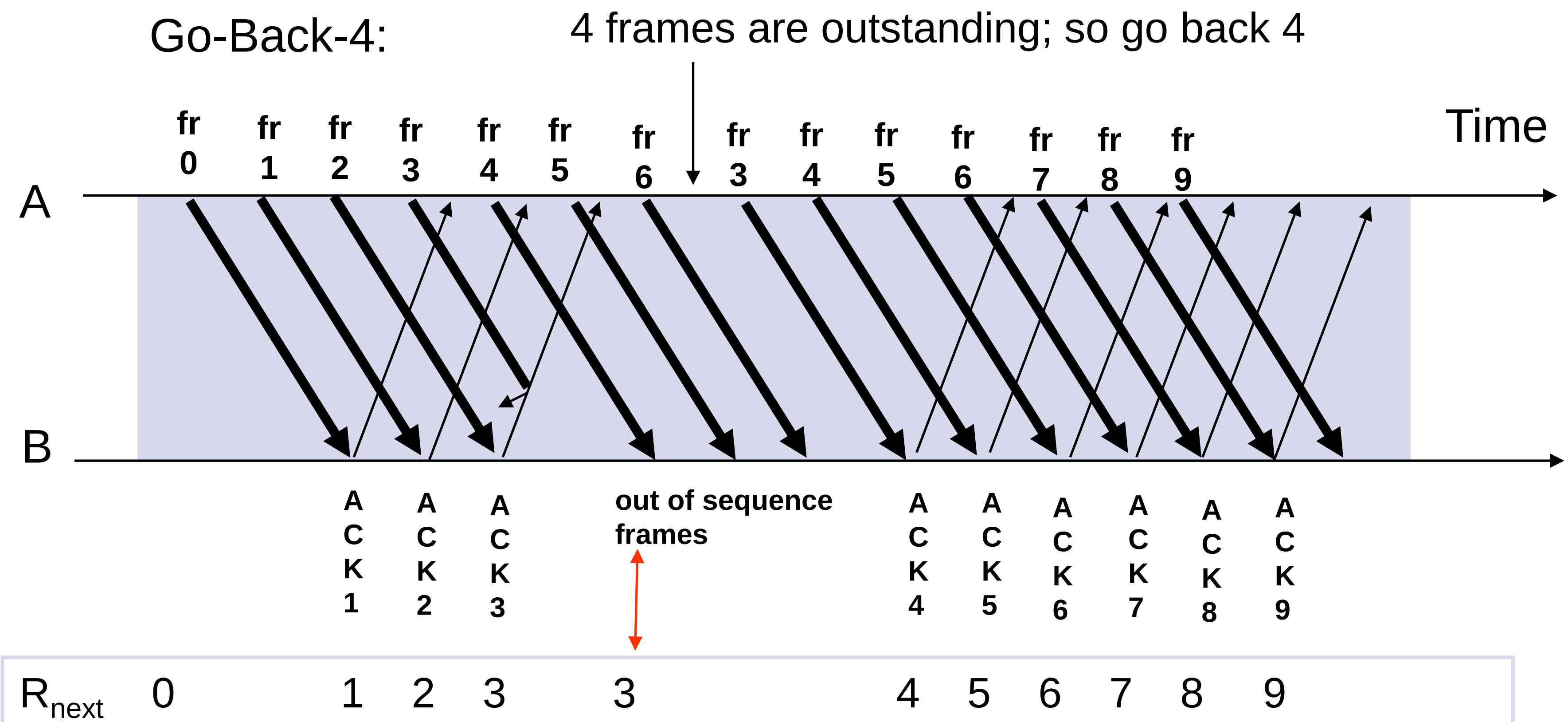
$1 - P_f$ **Efficiency**	0	10^{-6}	10^{-5}	10^{-4}
1 Mbps & 1 ms	1 88%	0.99 86.6%	0.905 79.2%	0.368 32.2%

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

Go-Back-N

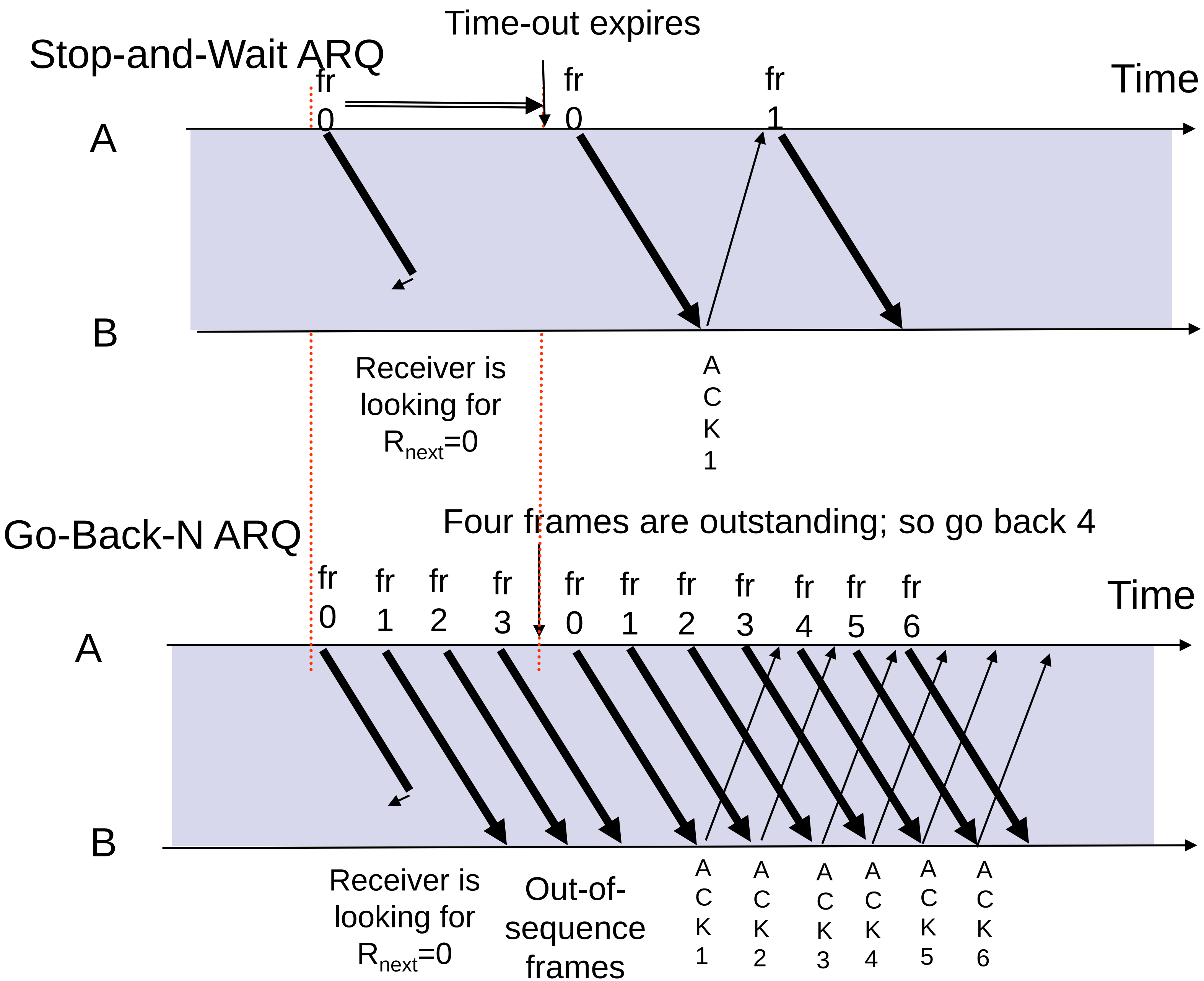
- Improve Stop-and-Wait by not waiting!
- Keep channel busy by continuing to send frames
- Allow a window of up to W_s outstanding frames
- Use *m*-bit sequence numbering
- If ACK for oldest frame arrives before window is exhausted, we can continue transmitting
- If window is exhausted, pull back and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

Window size long enough to cover round trip time

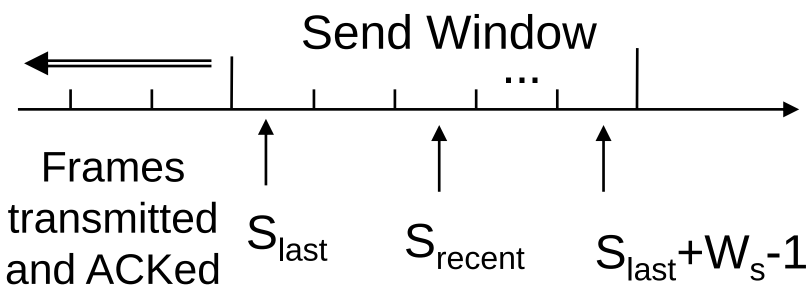


Go-Back-N with Timeout

- Problem with Go-Back-N as presented:
  - If frame is lost and source does not have 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

Go-Back-N Transmitter & Receiver

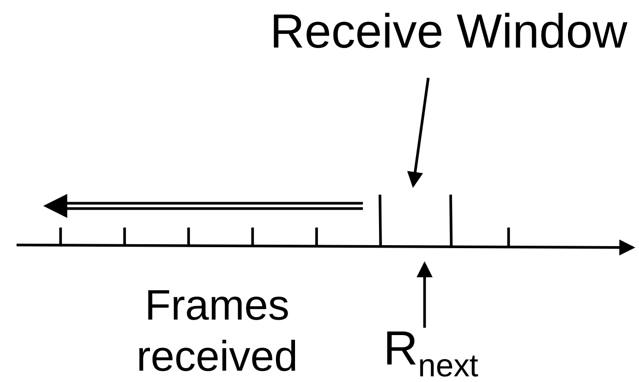
Transmitter



Buffers

Timer	Buffer	
Timer	S_{last}	oldest un-ACKed frame
Timer	$S_{last}+1$	
	...	
Timer	S_{recent}	most recent transmission
	...	
	$S_{last}+W_s-1$	max Seq # allowed

Receiver

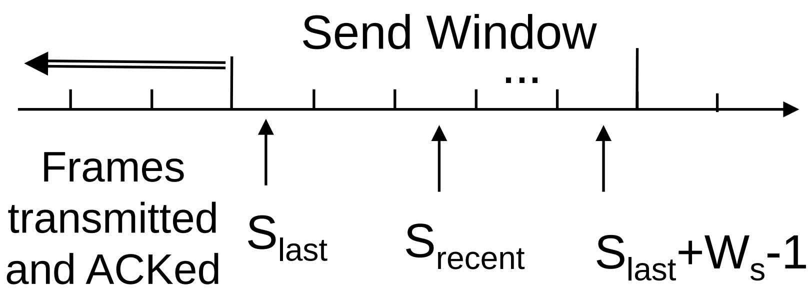


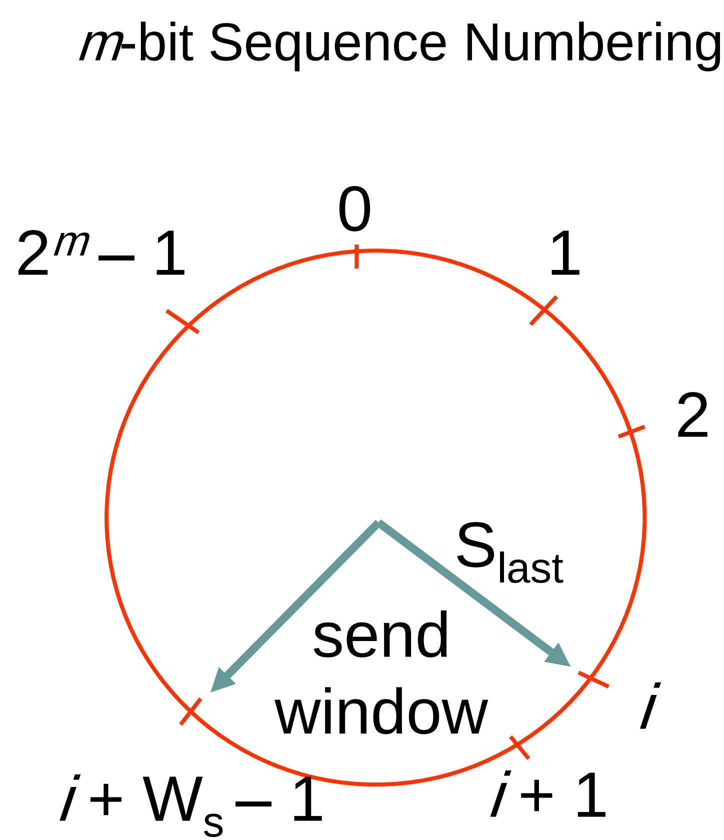
Receiver will only accept a frame that is error-free and that has sequence number R_{next}

When such frame arrives R_{next} is incremented by one, so the *receive window slides forward* by one

Sliding Window Operation

Transmitter

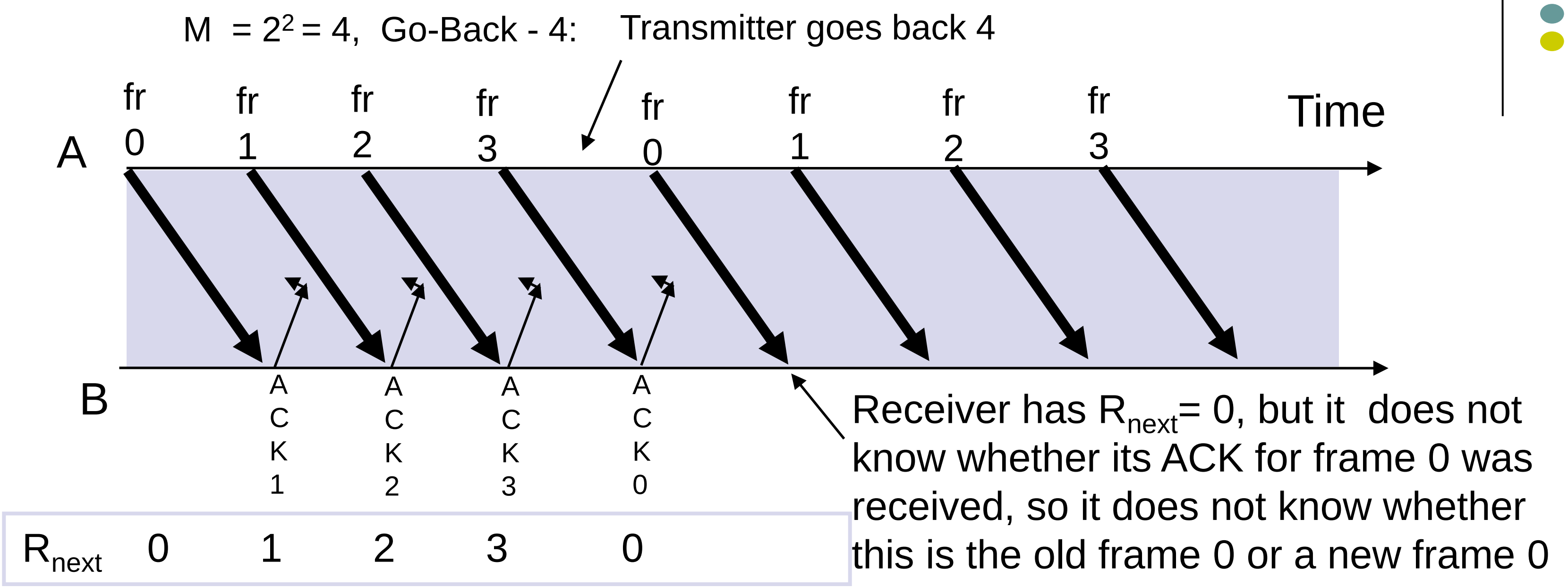


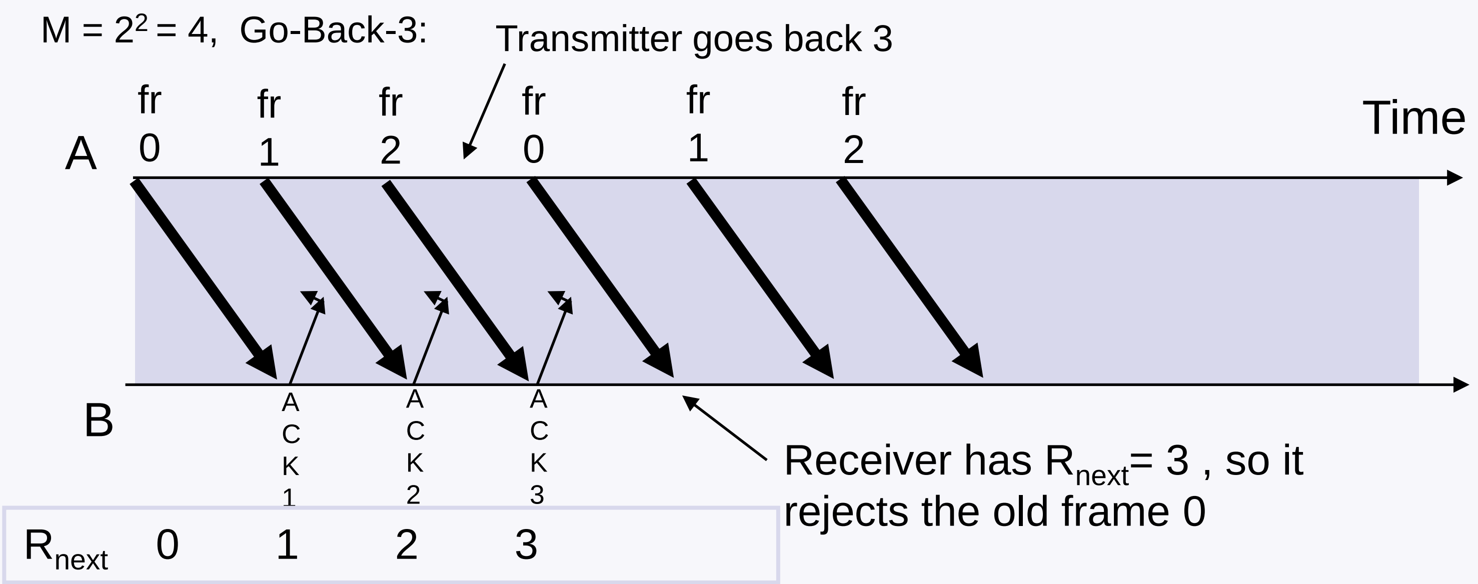


Transmitter waits for error-free ACK frame with sequence number S_{last} +1

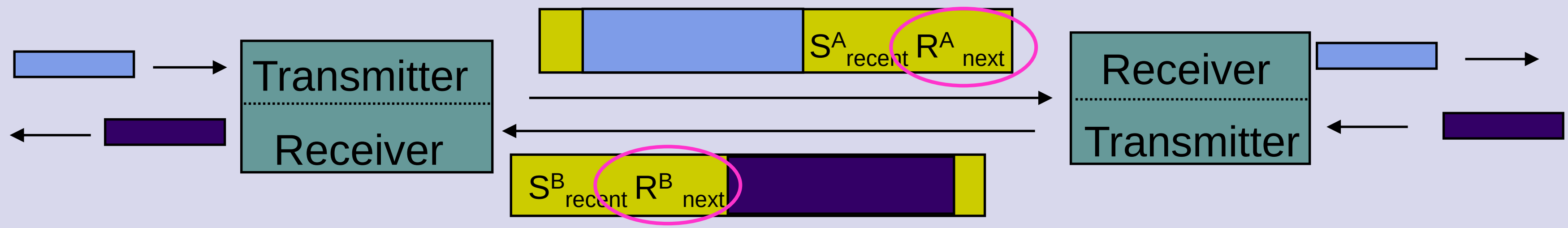
When such ACK frame arrives, S_{last} is incremented by one, and the *send window slides forward* by one

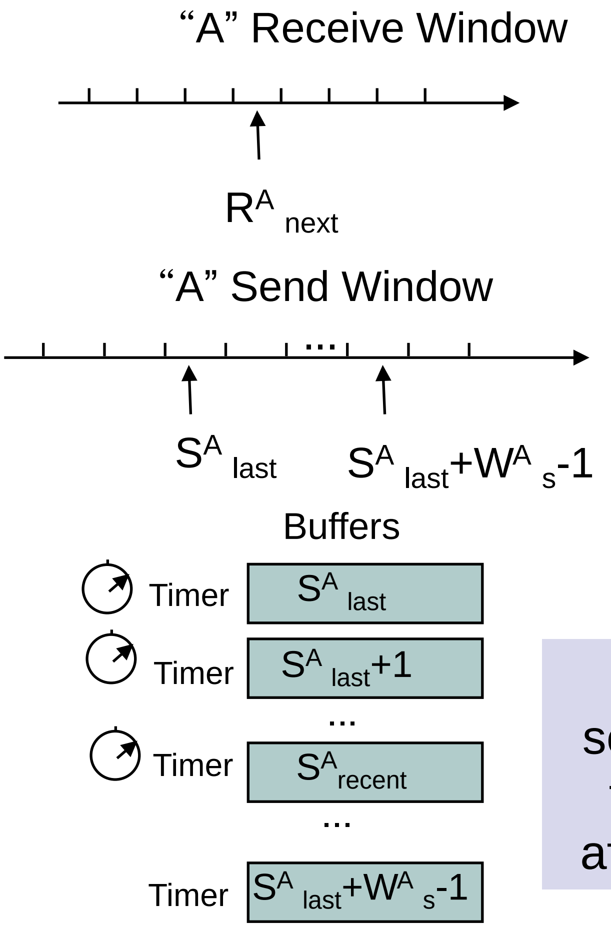
Maximum Allowable Window Size is $W_s = 2^m - 1$

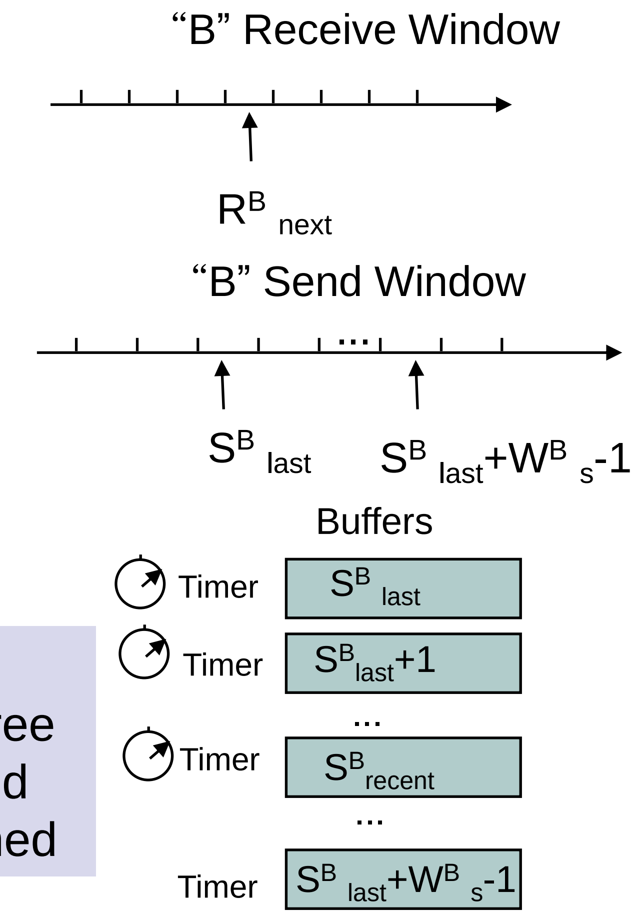




ACK Piggybacking in Bidirectional GBN





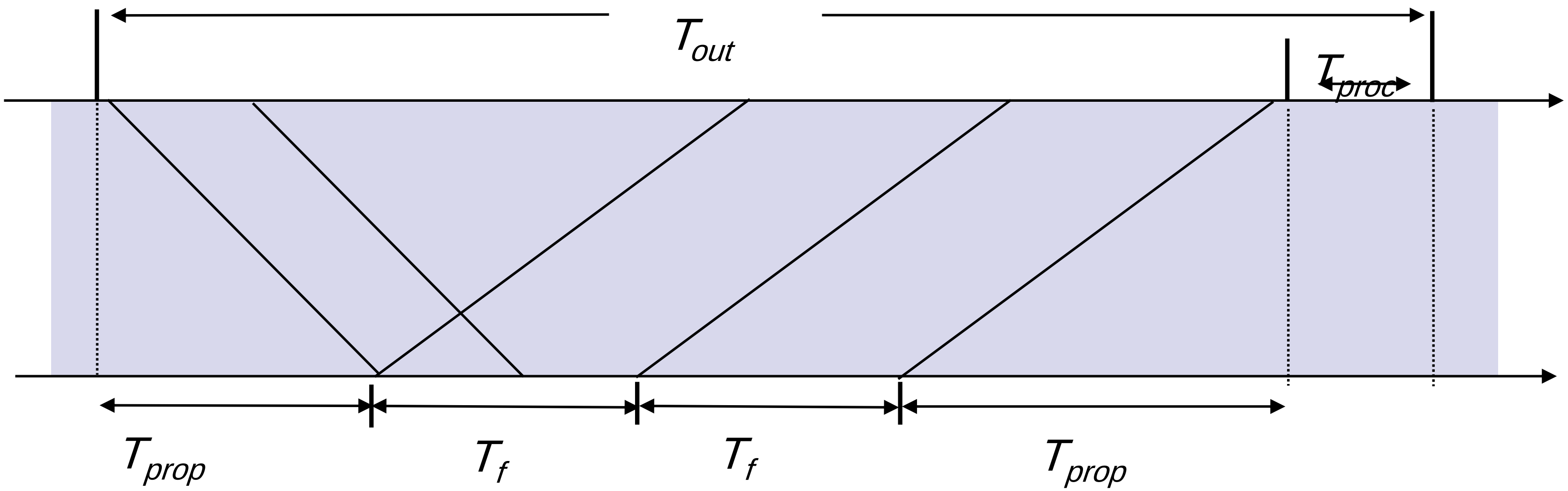


Note: Out-of-sequence error-free frames discarded after R_{next} examined

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 + 1 processing time: $2\ T_{prop} + T_{proc}$
  - A frame that begins transmission right before our frame arrives T_f
  - Next frame carries the ACK, T_f
- 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		
$2(t_{prop} + t_{proc})$	**2 x Delay x BW**	**Window**
1 ms	1000 bits	1
10 ms	10,000 bits	2
100 ms	100,000 bits	11
1 second	1,000,000 bits	101

Efficiency of Go-Back-N

- GBN is completely efficient, if W_s large enough to keep channel busy, and if channel is error-free
- Assume P_f frame loss probability, then time to deliver a frame is:
  - t_f if first frame transmission succeeds $(1 – P_f)$
  - $t_f + W_s t_f / (1-P_f)$ if the first transmission does not succeed P_f

$$t_{GBN} = t_f(1 - P_f) + P_f\{t_f + \frac{W_s t_f}{1 - P_f}\} = t_f + P_f \frac{W_s t_f}{1 - P_f} \quad \text{and}$$

$$\eta_{GBN} = \frac{\frac{n_f - n_o}{t_{GBN}}}{R} = \frac{1 - \frac{n_o}{n_f}}{1 + (W_s - 1)P_f}(1 - P_f)$$

Delay-bandwidth product determines W_s

Example: Impact Bit Error Rate on GBN

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

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

1 Mbps x 100 ms = 100000 bits = 10 frames → Use $W_s = 11$

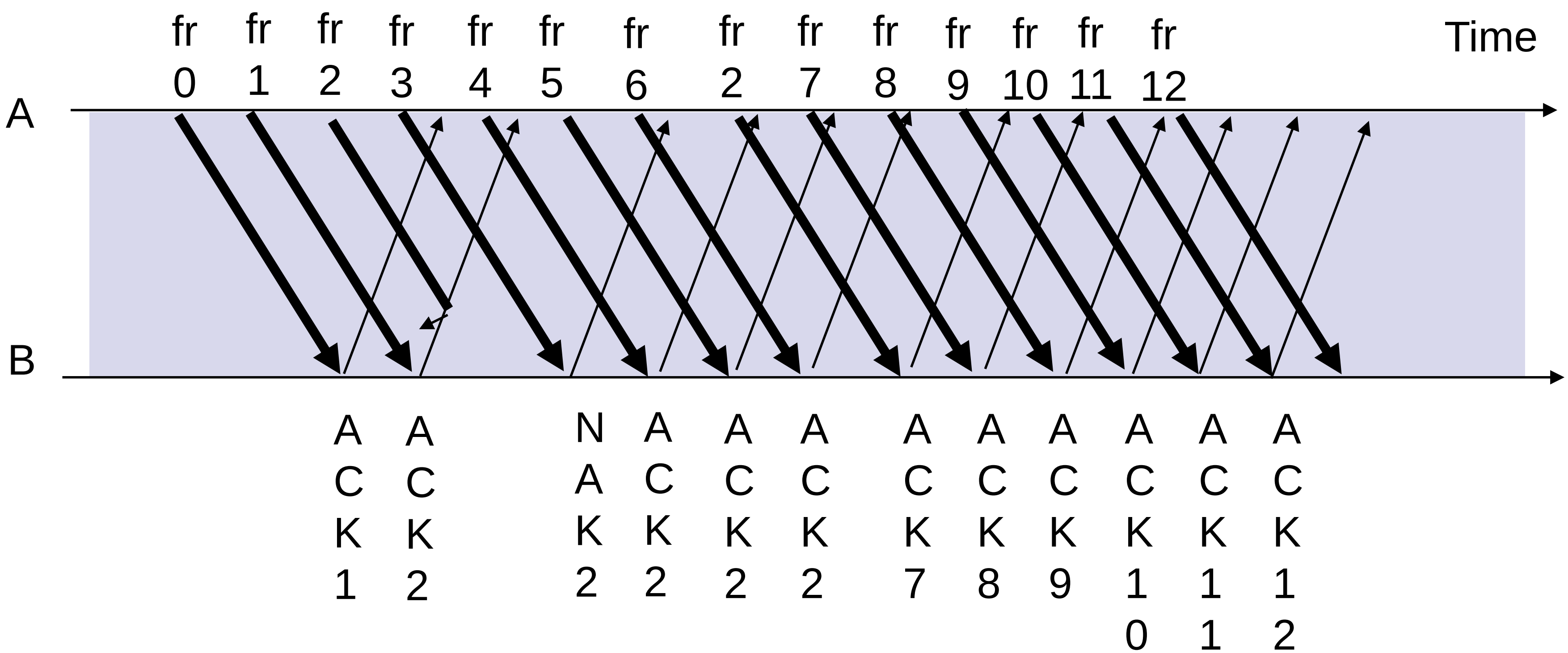
Efficiency	0	10^{-6}	10^{-5}	10^{-4}
S&W	8.9%	8.8%	8.0%	3.3%
GBN	98%	88.2%	45.4%	4.9%

- *Go-Back-N significant improvement over Stop-and-Wait for large delay-bandwidth product*
- *Go-Back-N becomes inefficient as error rate increases*

Selective Repeat ARQ

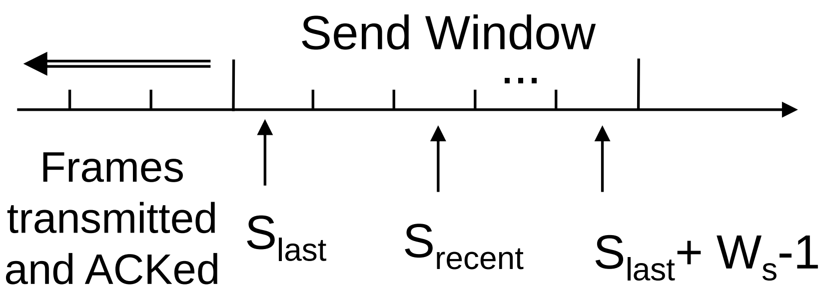
- Go-Back-N ARQ inefficient because *multiple* frames are resent when errors or losses occur
- Selective Repeat retransmits *only an individual frame*
  - Timeout causes individual corresponding frame to be resent
  - NAK causes retransmission of oldest un-acked frame
- Receiver maintains a *receive 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



Selective Repeat ARQ

Transmitter



Buffers

Timer S_{last}

Timer $S_{last}+ 1$

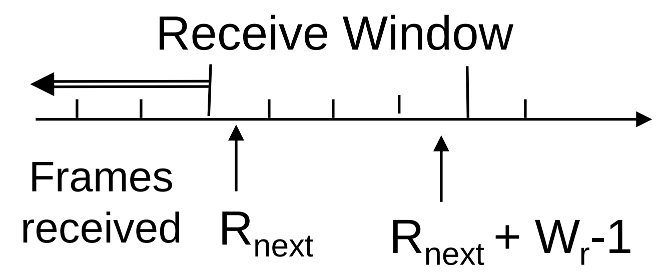
...

Timer S_{recent}

...

$S_{last}+ W_s - 1$

Receiver



Buffers

$R_{next}+ 1$

$R_{next}+ 2$

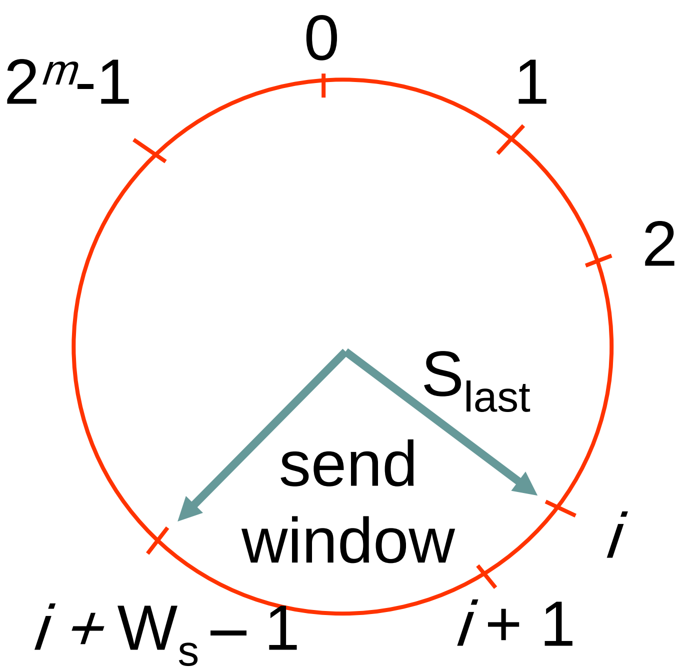
...

$R_{next}+ W_r- 1$

max Seq # accepted

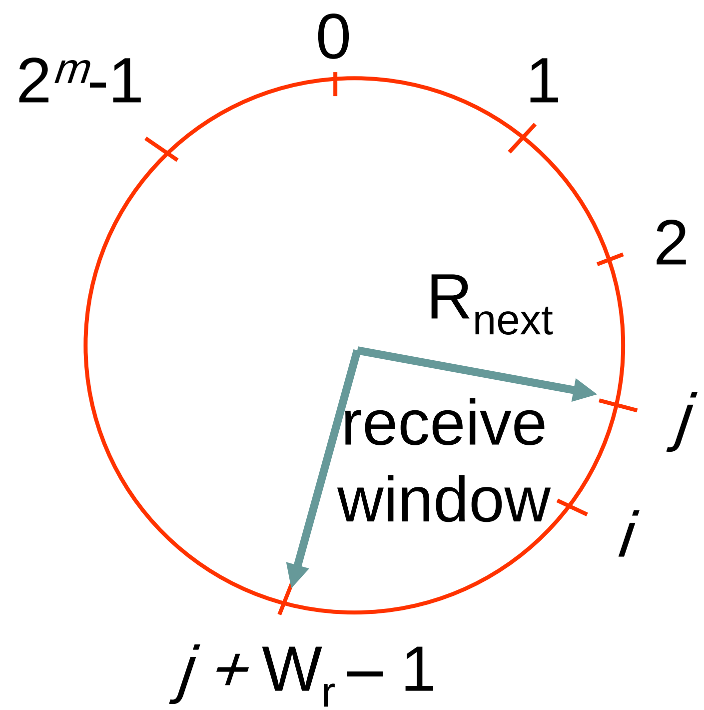
Send & Receive Windows

Transmitter



Moves k forward when ACK arrives with $R_{next} = S_{last} + k$
$k = 1, \ldots, W_s$

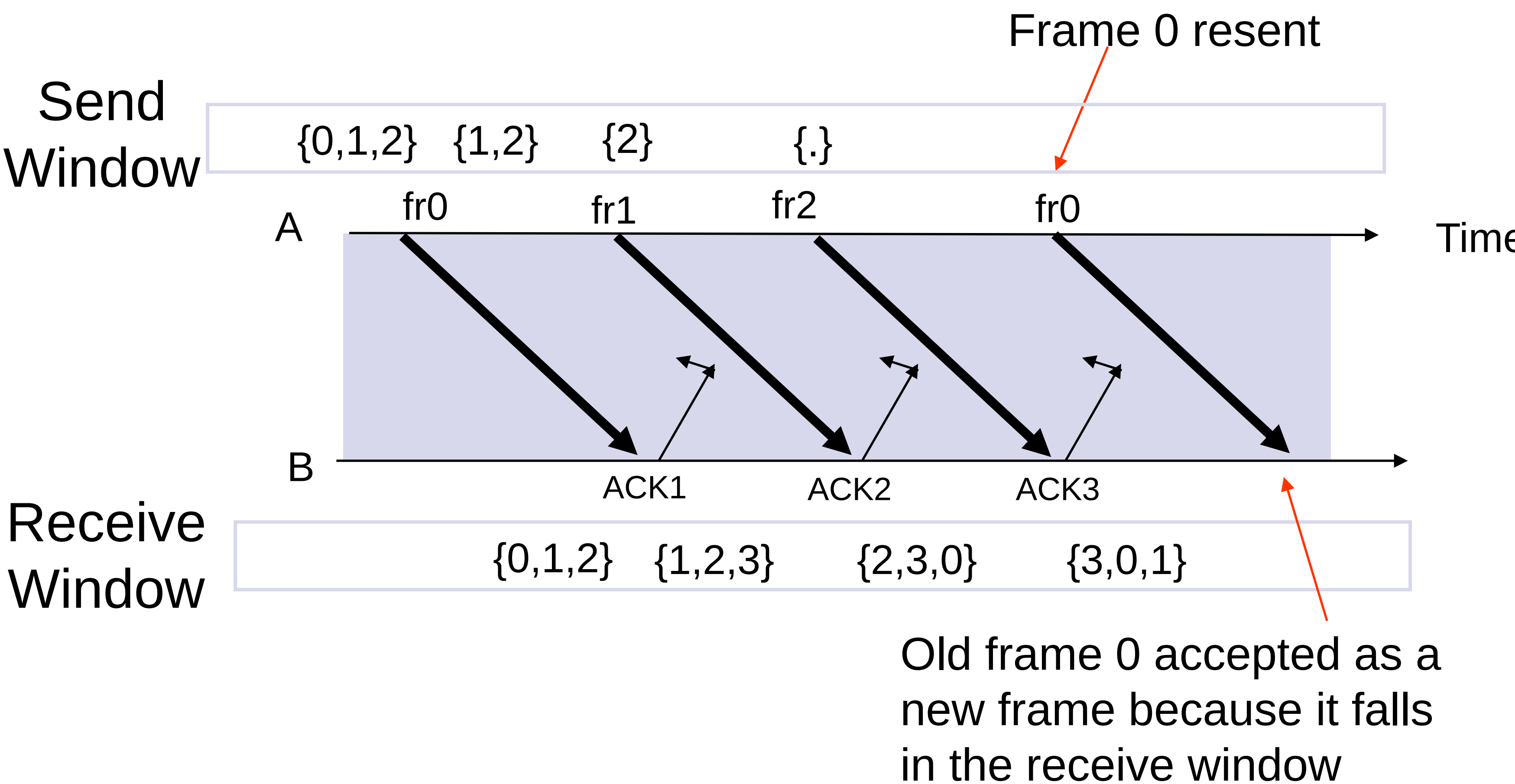
Receiver



Moves forward by 1 or more when frame arrives with Seq. # = R_{next}

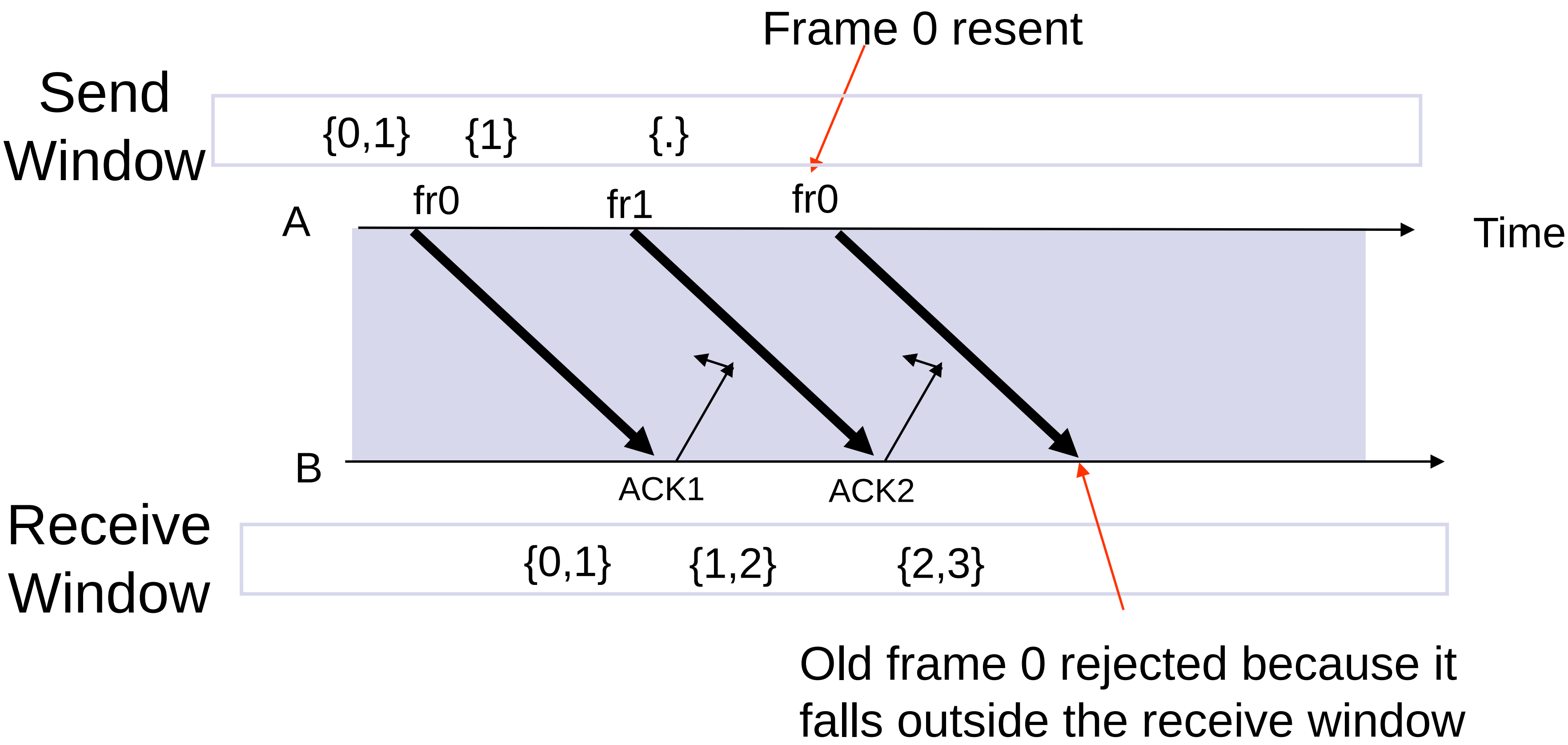
What size W_s and W_r allowed?

- Example: $M=2^2=4$, $W_s=3$, $W_r=3$



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

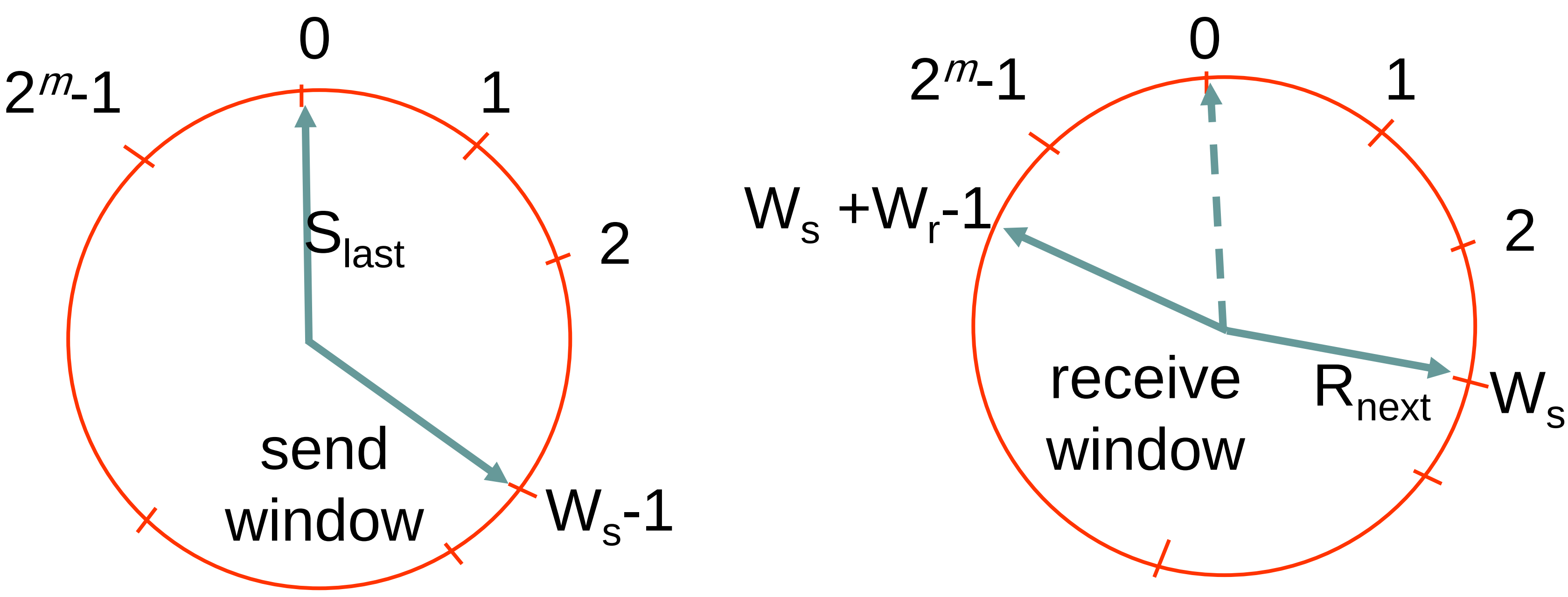
- Example: $M=2^2=4$, $W_s=2$, $W_r=2$



Why $W_s + W_r = 2^m$ works

- Transmitter sends frames 0 to Ws-1; send window empty
- All arrive at receiver
- All ACKs lost
- Transmitter resends frame 0

- Receiver window starts at $\{0, \ldots, W_r\}$
- Window slides forward to $\{W_s, \ldots, W_s+W_r-1\}$
- Receiver rejects frame 0 because it is outside receive window



Applications of Selective 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:
  - $t_f / (1-P_f)$

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

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

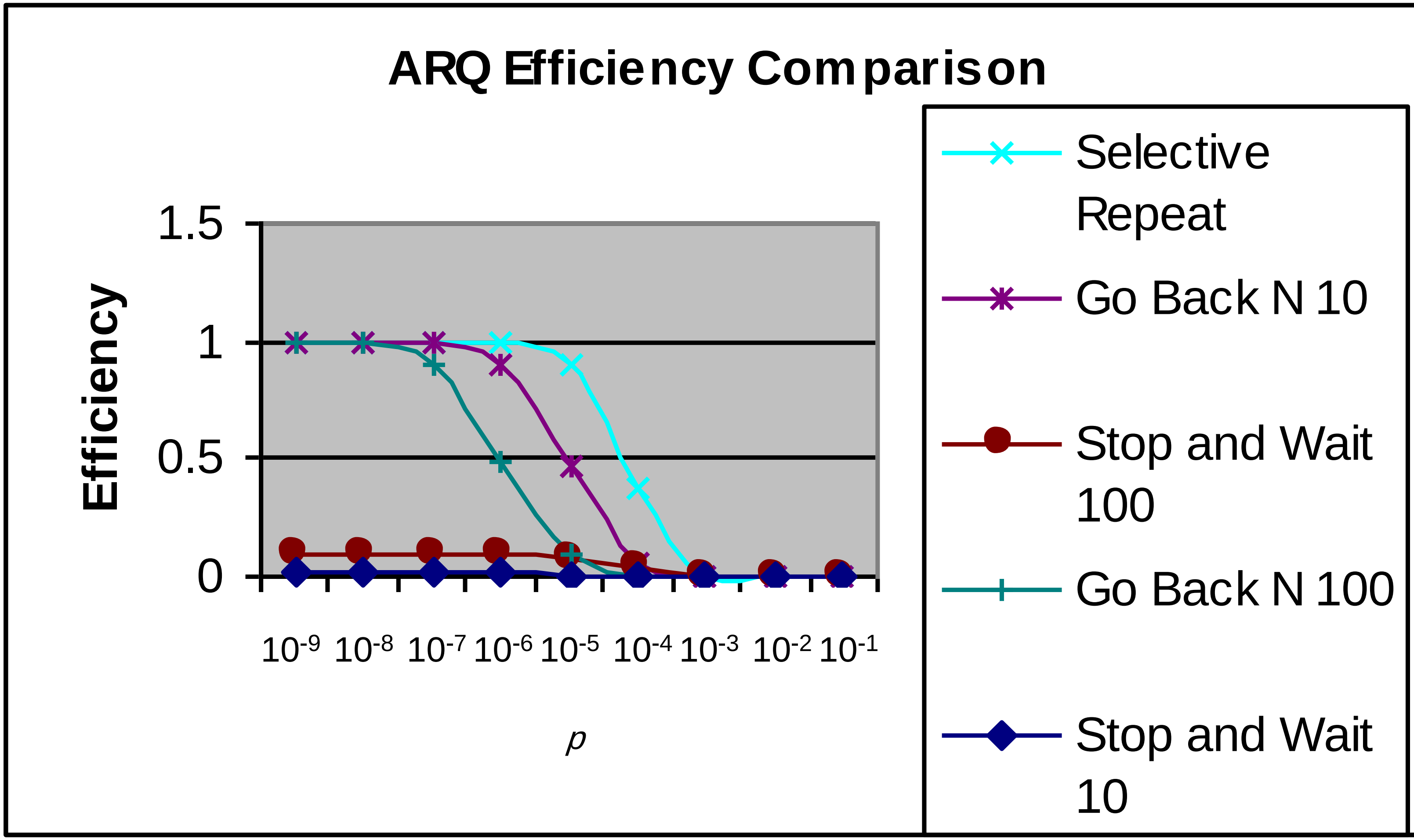
$$\eta_{GBN} = \frac{1-P_f}{1+(W_S-1)P_f} = \frac{1-P_f}{1+LP_f}$$

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

Stop-and-Wait:

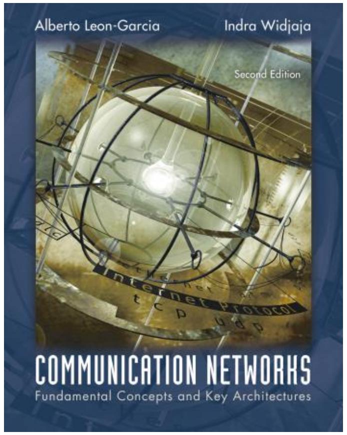
$$\eta_{SW} = \frac{(1-P_f)}{1+\frac{n_a}{n_f}+\frac{2(t_{prop}+t_{proc})R}{n_f}} \approx \frac{1-P_f}{1+L}$$

ARQ Efficiencies



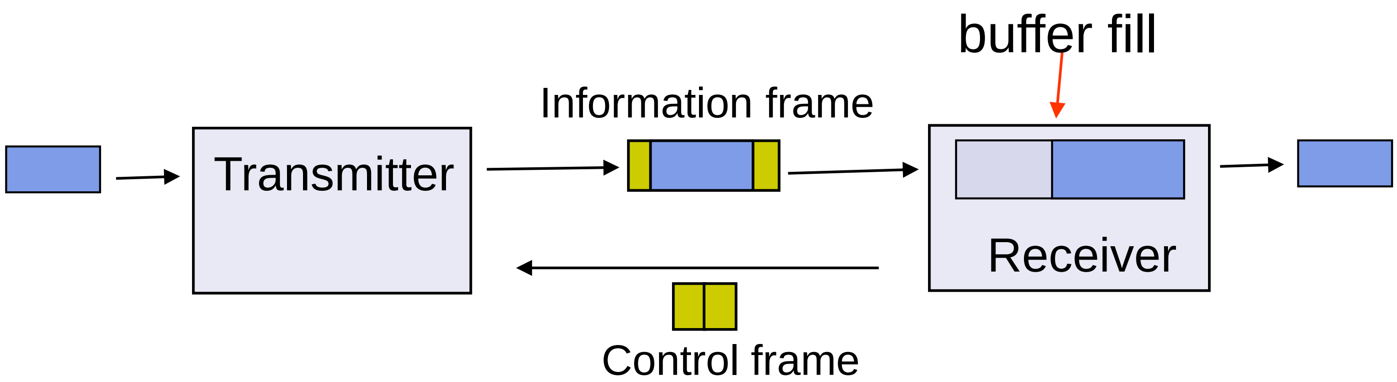
Delay-Bandwidth product = 10, 100

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



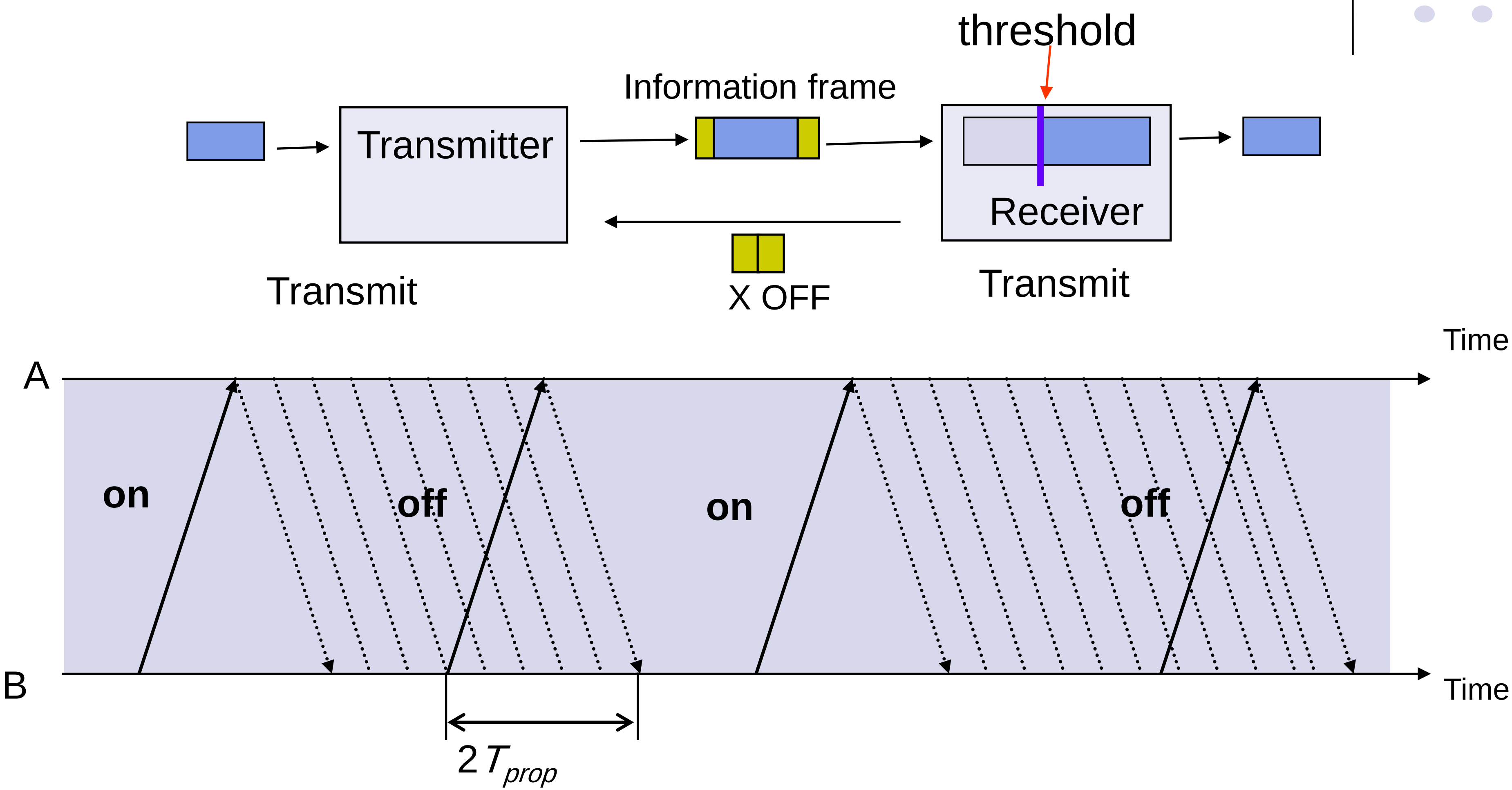
Flow Control

Flow Control



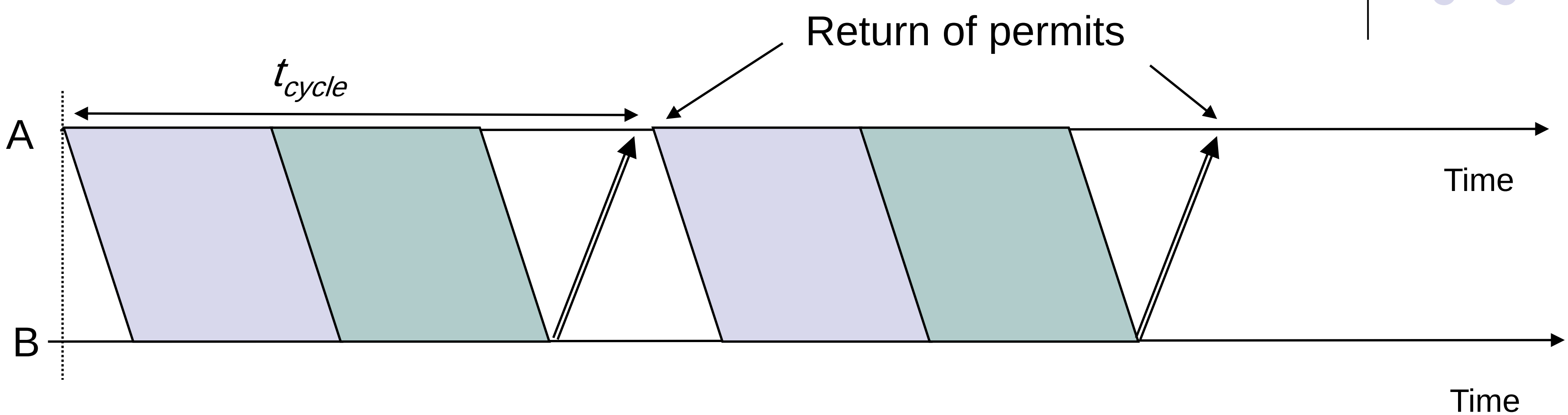
- Receiver has limited buffering to store arriving frames
- Several situations cause buffer overflow
  - Mismatch between sending rate & rate at which user can retrieve data
  - Surges in frame arrivals
- *Flow control* prevents buffer overflow by regulating rate at which source is allowed to send information

X ON / X OFF



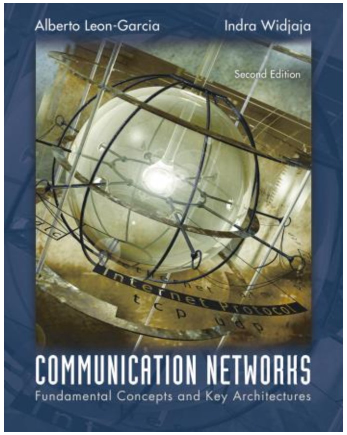
Threshold must activate OFF signal while 2 T_{prop} R bits still remain in buffer

Window Flow Control



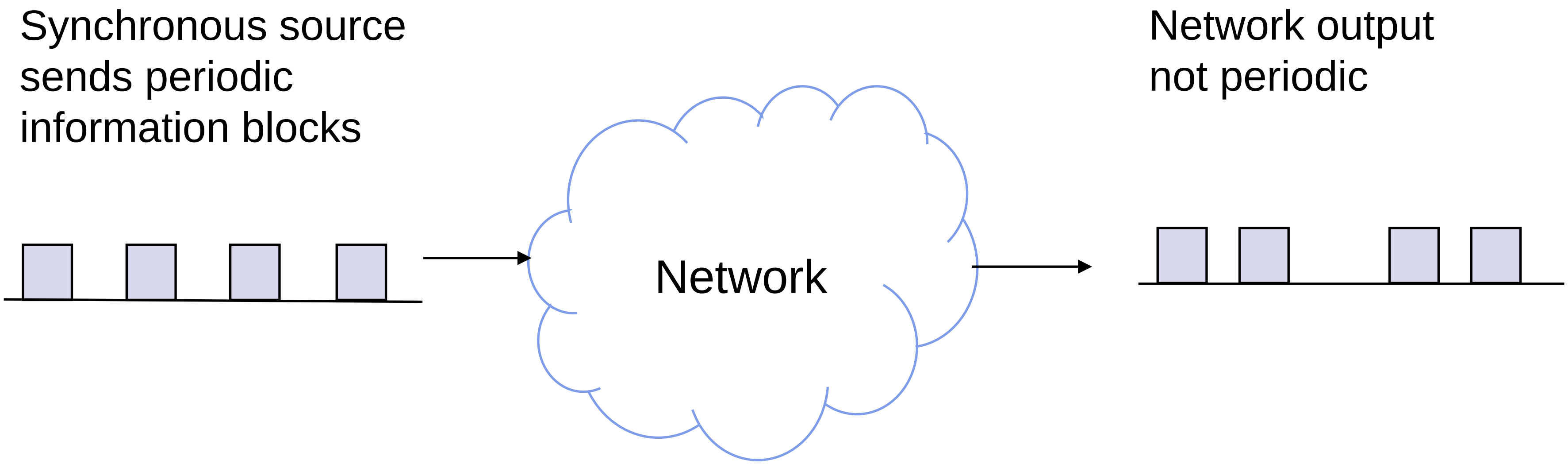
- Sliding Window ARQ method with W_s equal to buffer available
  - Transmitter can never send more than W_s frames
- ACKs that slide window forward can be viewed as permits to transmit more
- Can also pace ACKs as shown above
  - Return permits (ACKs) at end of cycle regulates transmission rate
- Problems using sliding window for both error & flow control
  - Choice of window size
  - Interplay between transmission rate & retransmissions
  - TCP separates error & flow control

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



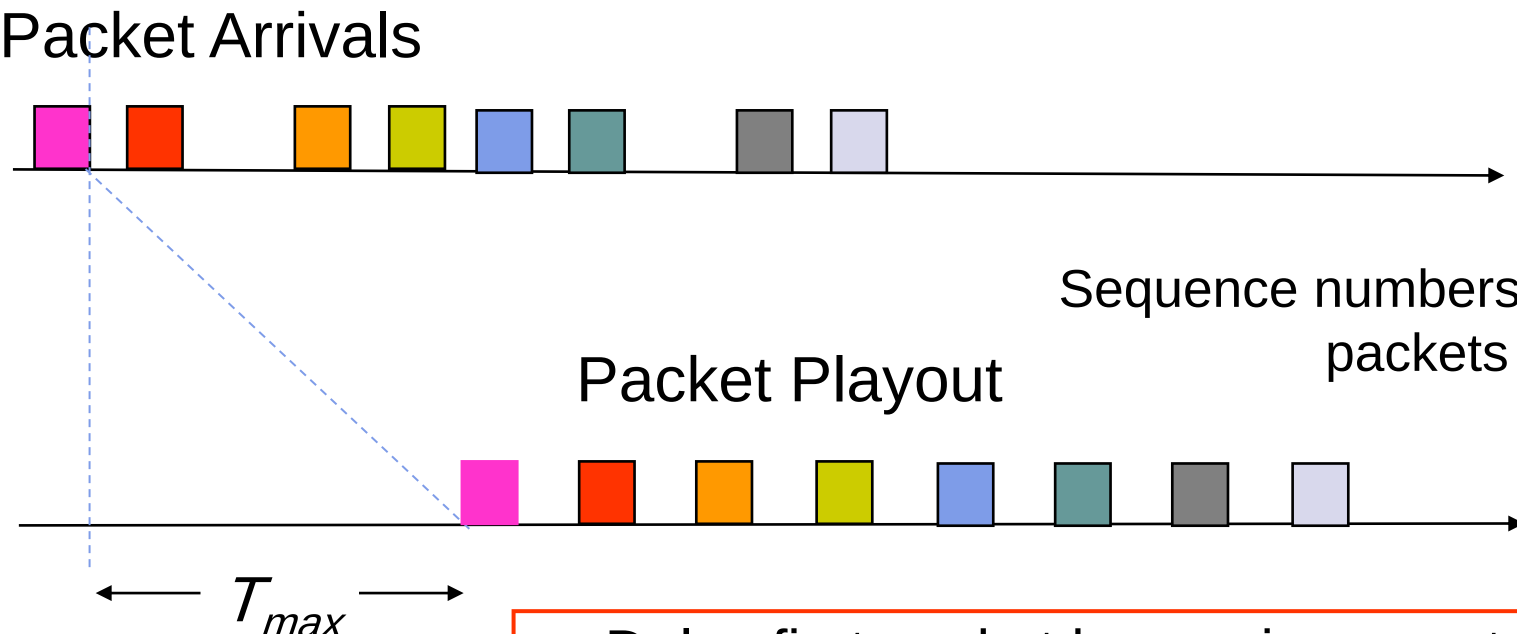
Timing Recovery

Timing Recovery for Synchronous Services



- Applications that involve voice, audio, or video can generate a synchronous information stream
- Information carried by equally-spaced fixed-length packets
- Network multiplexing & switching introduces random delays
  - Packets experience variable transfer delay
  - Jitter (variation in interpacket arrival times) also introduced
- Timing recovery re-establishes the synchronous nature of the stream

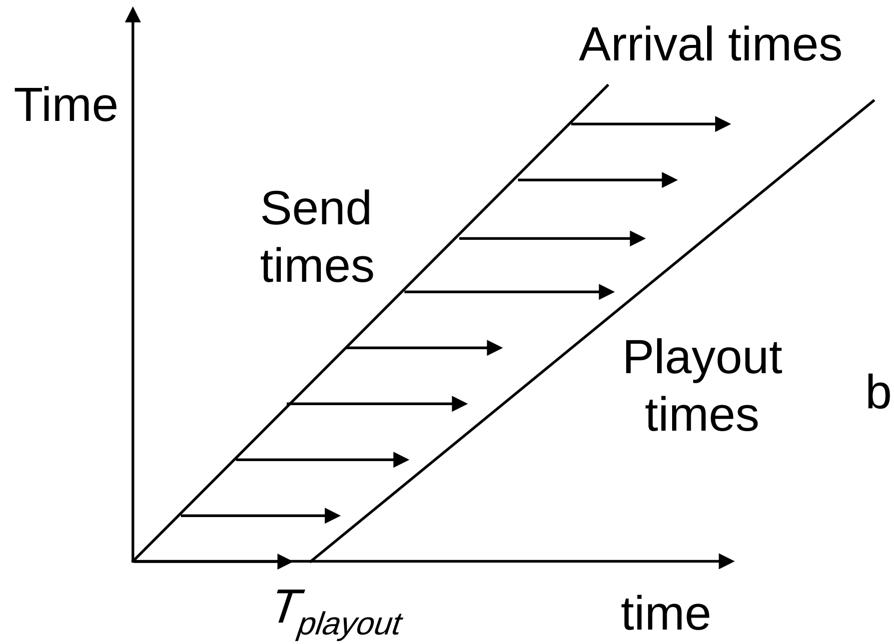
Introduce Playout Buffer



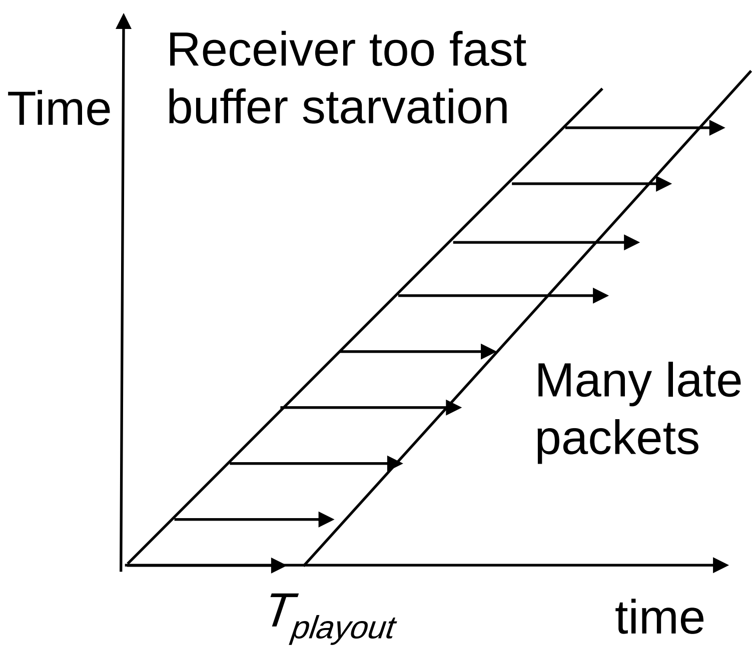
Sequence numbers help order packets

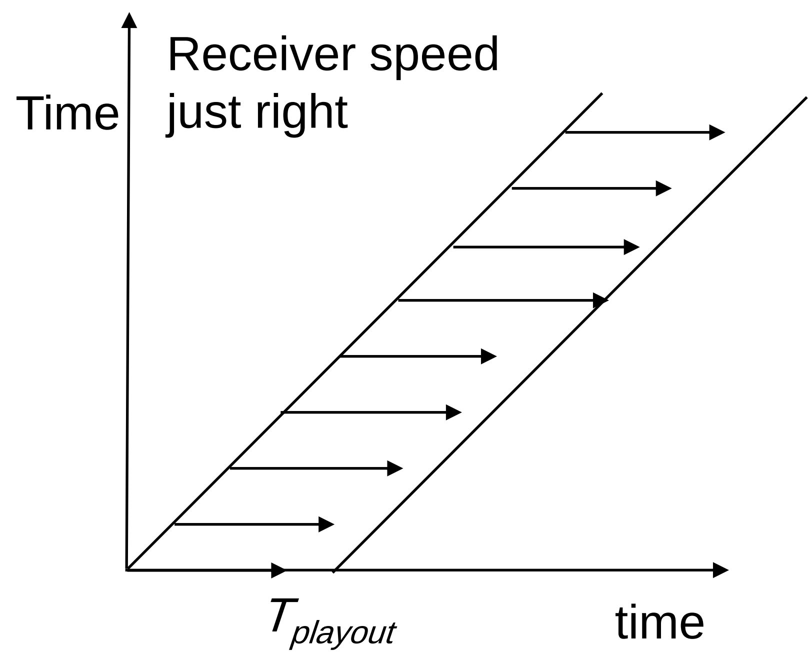
- Delay first packet by maximum network delay
- All other packets arrive with less delay
- Playout packet uniformly thereafter

Playout clock must be synchronized to transmitter clock

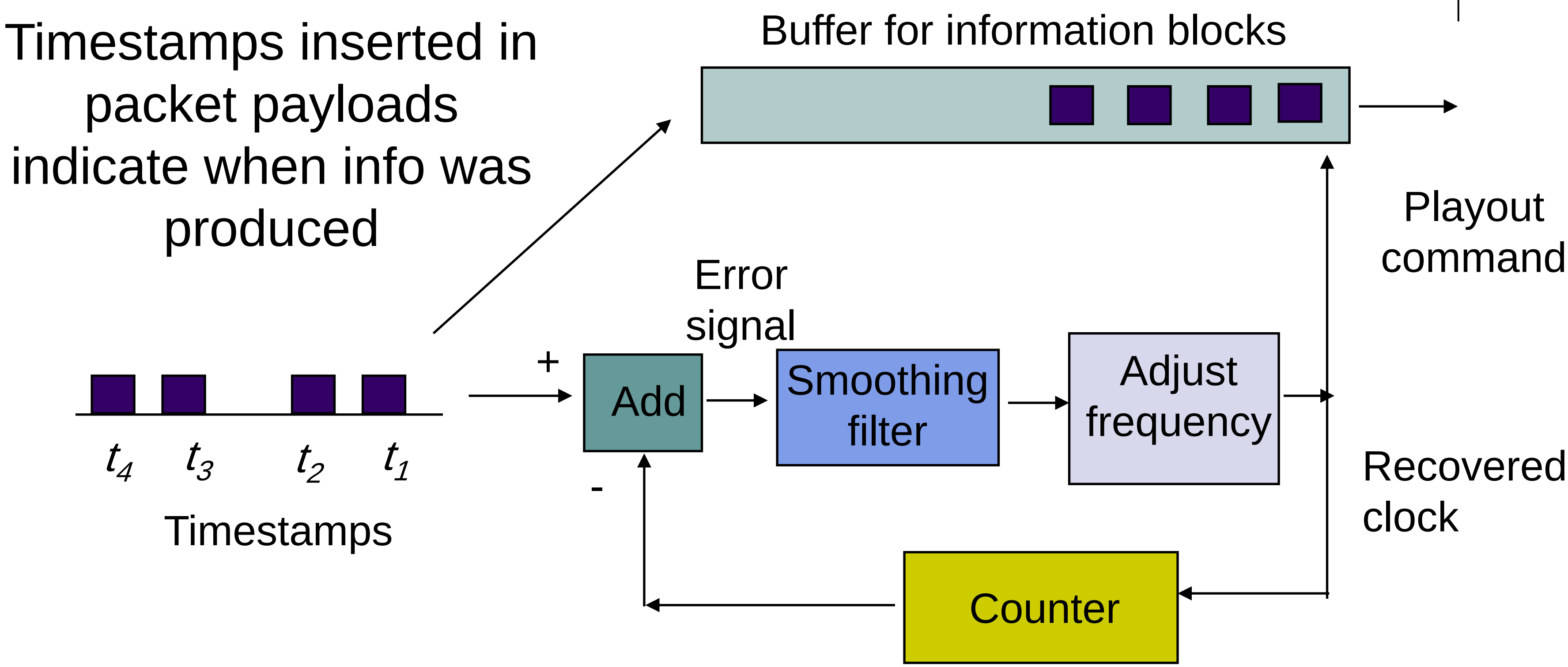


Receiver too slow;
buffer fills and overflows



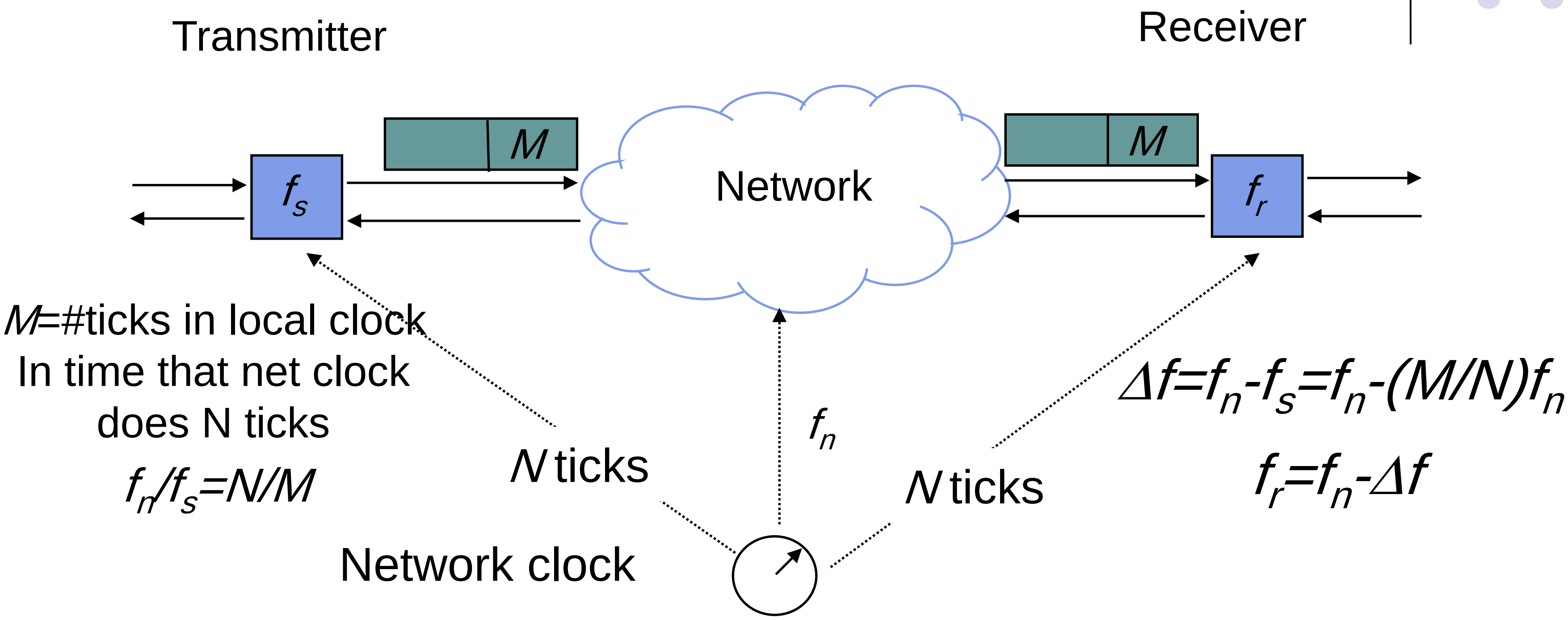


Clock Recovery



- Counter attempts to replicate transmitter clock
- Frequency of counter is adjusted according to arriving timestamps
- Jitter introduced by network causes fluctuations in buffer & in local clock

Synchronization to a Common Clock

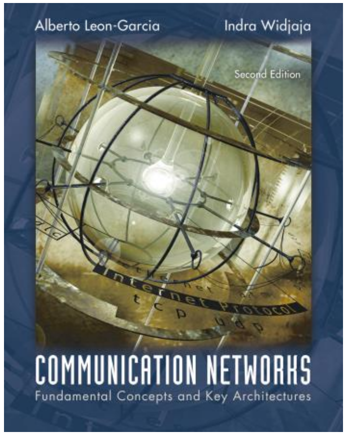


- Clock recovery simple if a common clock is available to transmitter & receiver
  - E.g. SONET network clock; Global Positioning System (GPS)
- Transmitter sends Δf of its frequency & network frequency
- Receiver adjusts network frequency by Δf
- Packet delay jitter can be removed completely

Example: Real-Time Protocol

- RTP (RFC 1889) designed to support real-time applications such as voice, audio, video
- RTP provides means to carry:
  - Type of information source
  - Sequence numbers
  - Timestamps
- Actual timing recovery must be done by higher layer protocol
  - MPEG2 for video, MP3 for audio

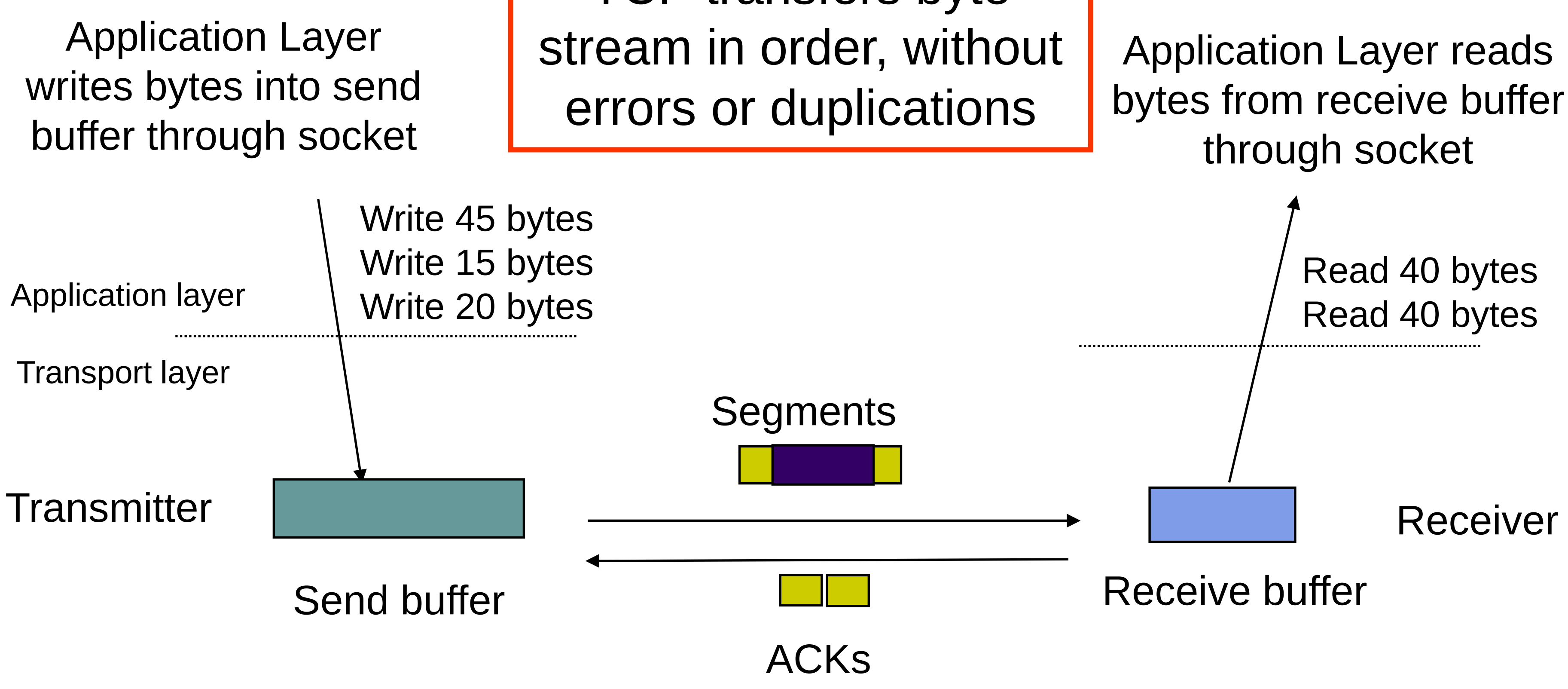
Chapter 5
Peer-to-Peer Protocols and Data Link Layer



TCP Reliable Stream Service & Flow Control

TCP Reliable Stream Service

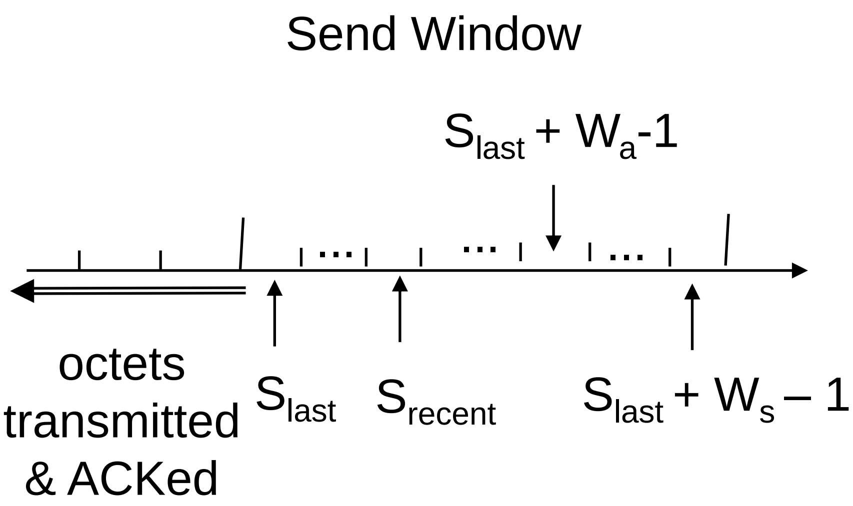
TCP transfers byte stream in order, without errors or duplications



TCP ARQ Method

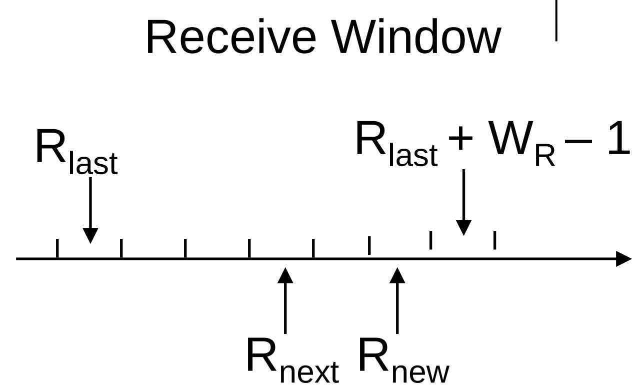
- TCP uses *Selective Repeat ARQ*
  - Transfers byte stream without preserving boundaries
- Operates over best effort service of IP
  - Packets can arrive with errors or be lost
  - Packets can arrive out-of-order
  - Packets can arrive after very long delays
  - Duplicate segments must be detected & discarded
  - Must protect against segments from previous connections
- Sequence Numbers
  - Seq. # is number of first byte in segment payload
  - Very long Seq. #s (32 bits) to deal with long delays
  - Initial sequence numbers negotiated during connection setup (to deal with very old duplicates)
  - Accept segments within a receive window

Transmitter



S_{last} oldest unacknowledged byte
S_{recent} highest-numbered transmitted byte
$S_{last}+W_a-1$ highest-numbered byte that can be transmitted
$S_{last}+W_s-1$ highest-numbered byte that can be accepted from the application

Receiver

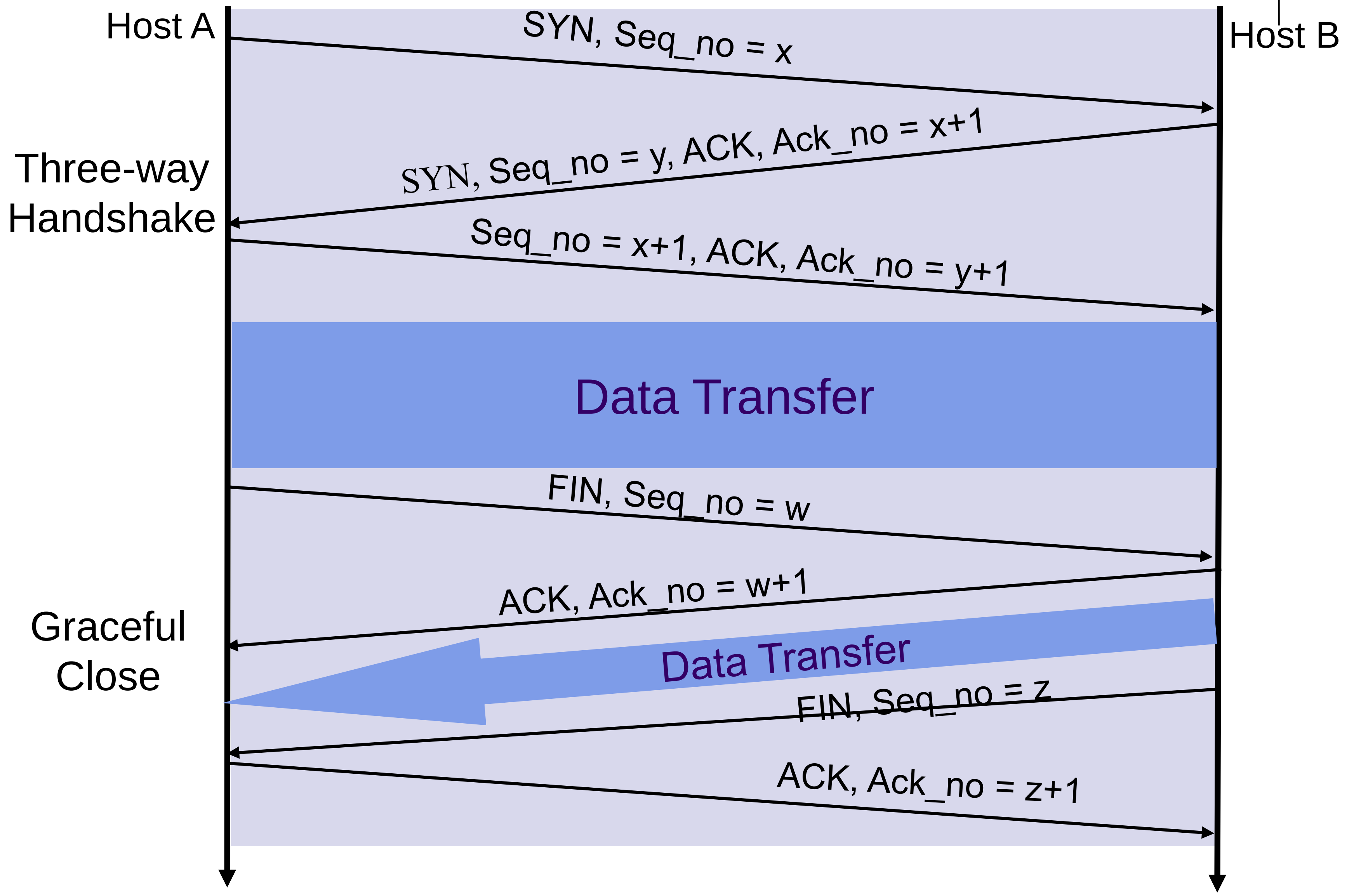


R_{last} highest-numbered byte not yet read by the application
R_{next} next expected byte
R_{new} highest numbered byte received correctly
$R_{last}+W_R-1$ highest-numbered byte that can be accommodated in receive buffer

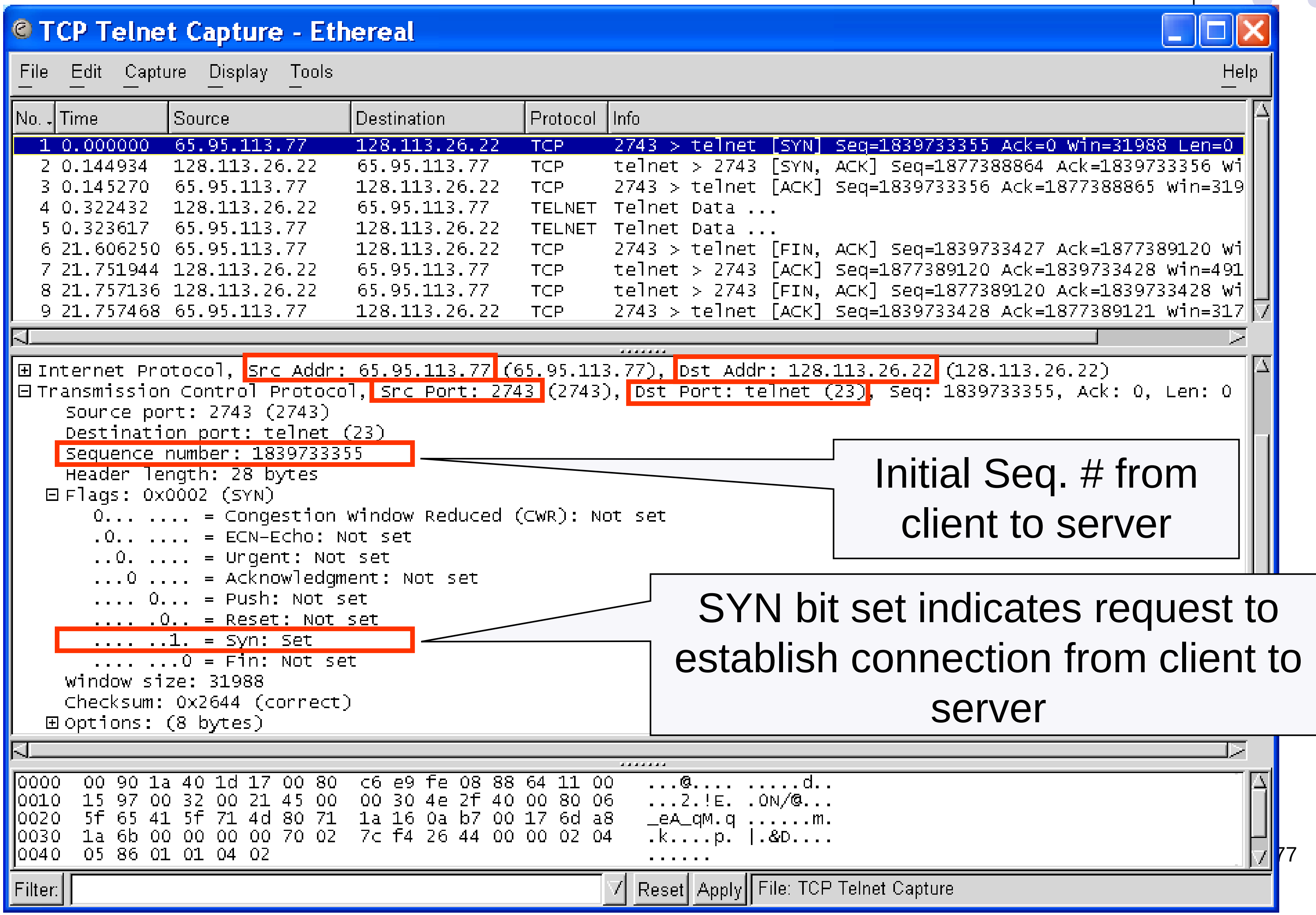
TCP Connections

- TCP Connection
  - One connection each way
  - Identified uniquely by Send IP Address, Send TCP Port #, Receive IP Address, Receive TCP Port #
- Connection Setup with Three-Way Handshake
  - Three-way exchange to negotiate initial Seq. #'s for connections in each direction
- Data Transfer
  - Exchange segments carrying data
- Graceful Close
  - Close each direction separately

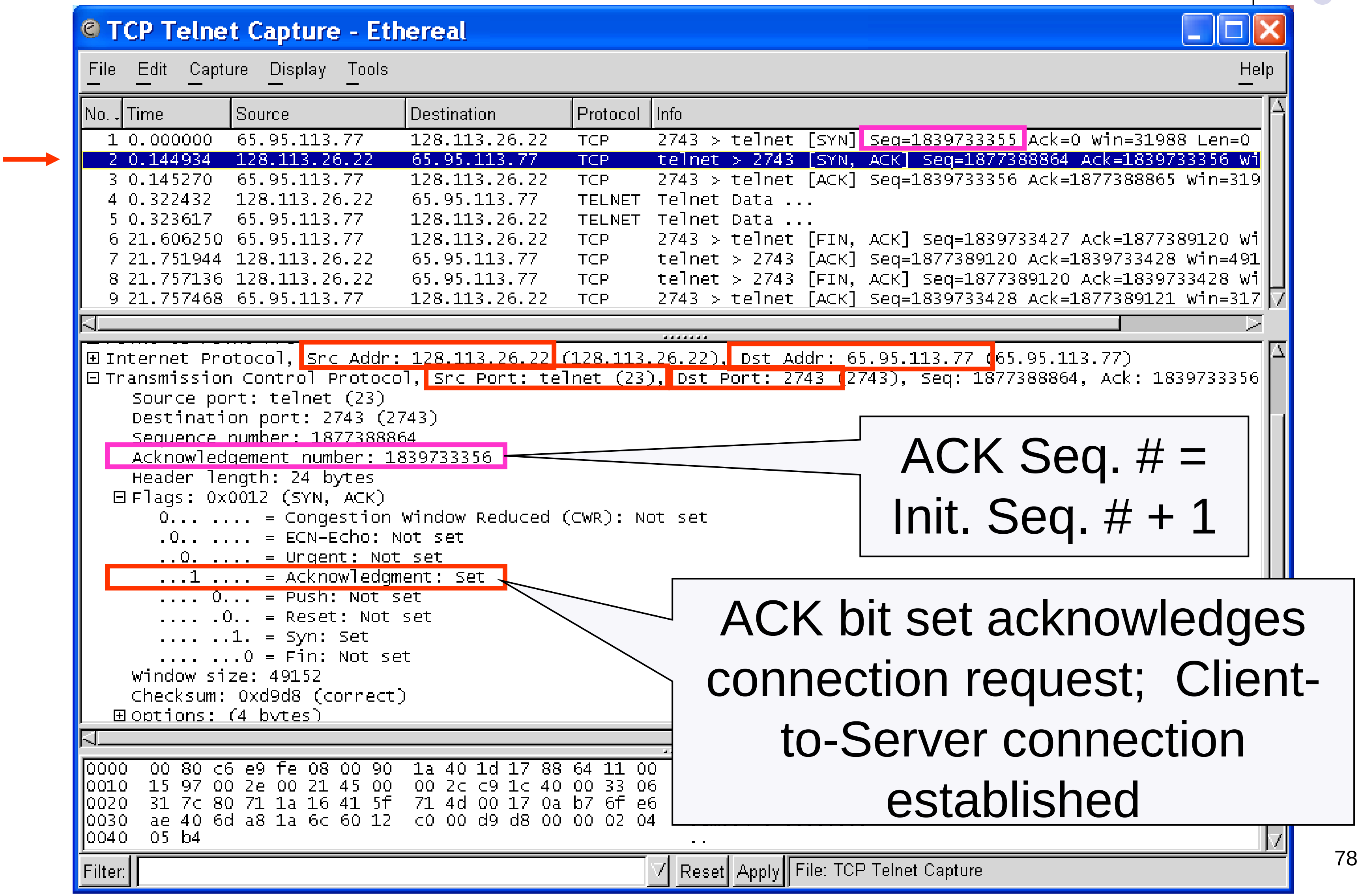
Three Phases of TCP Connection



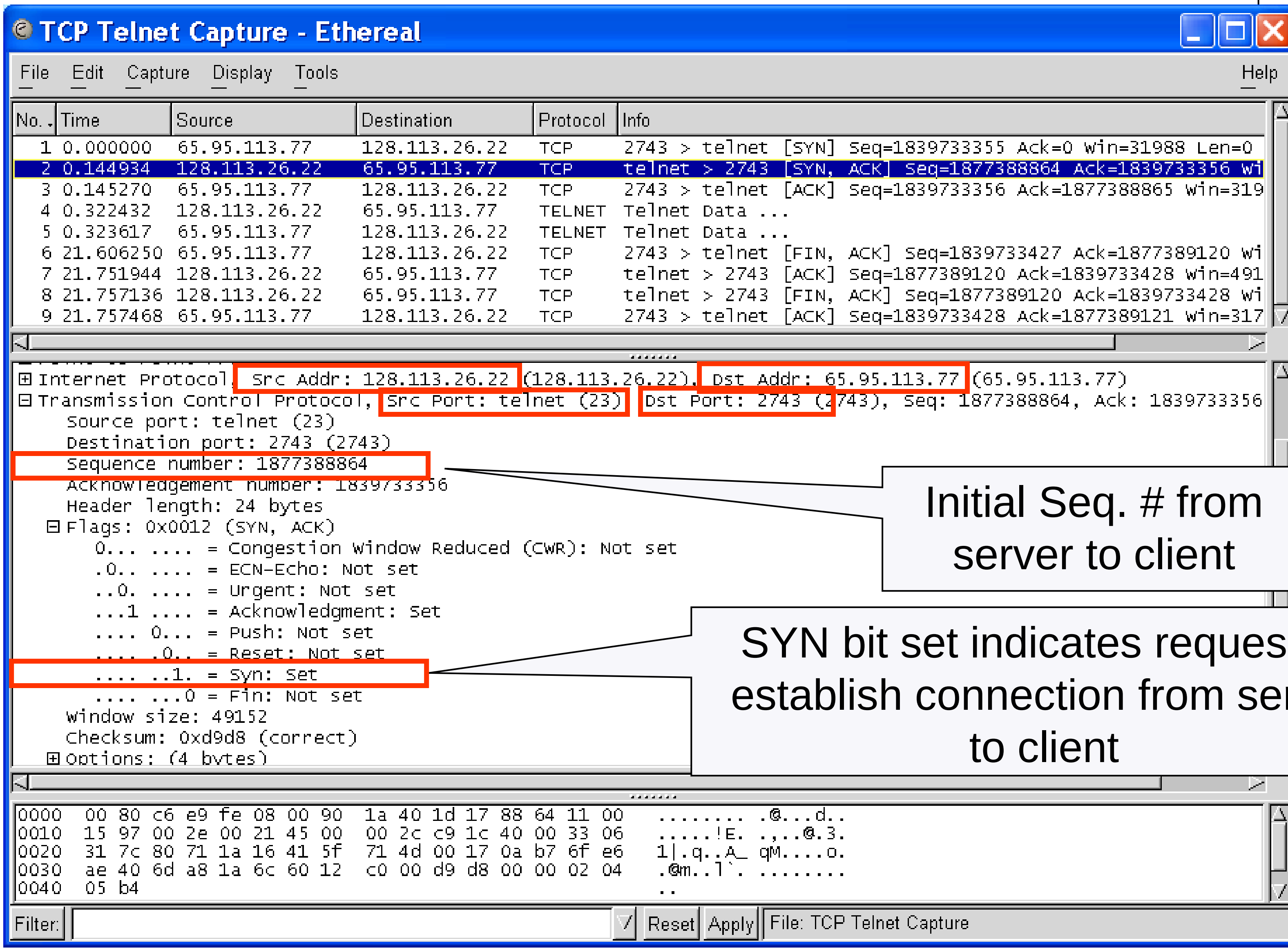
1st Handshake: Client-Server Connection Request



2nd Handshake: ACK from Server



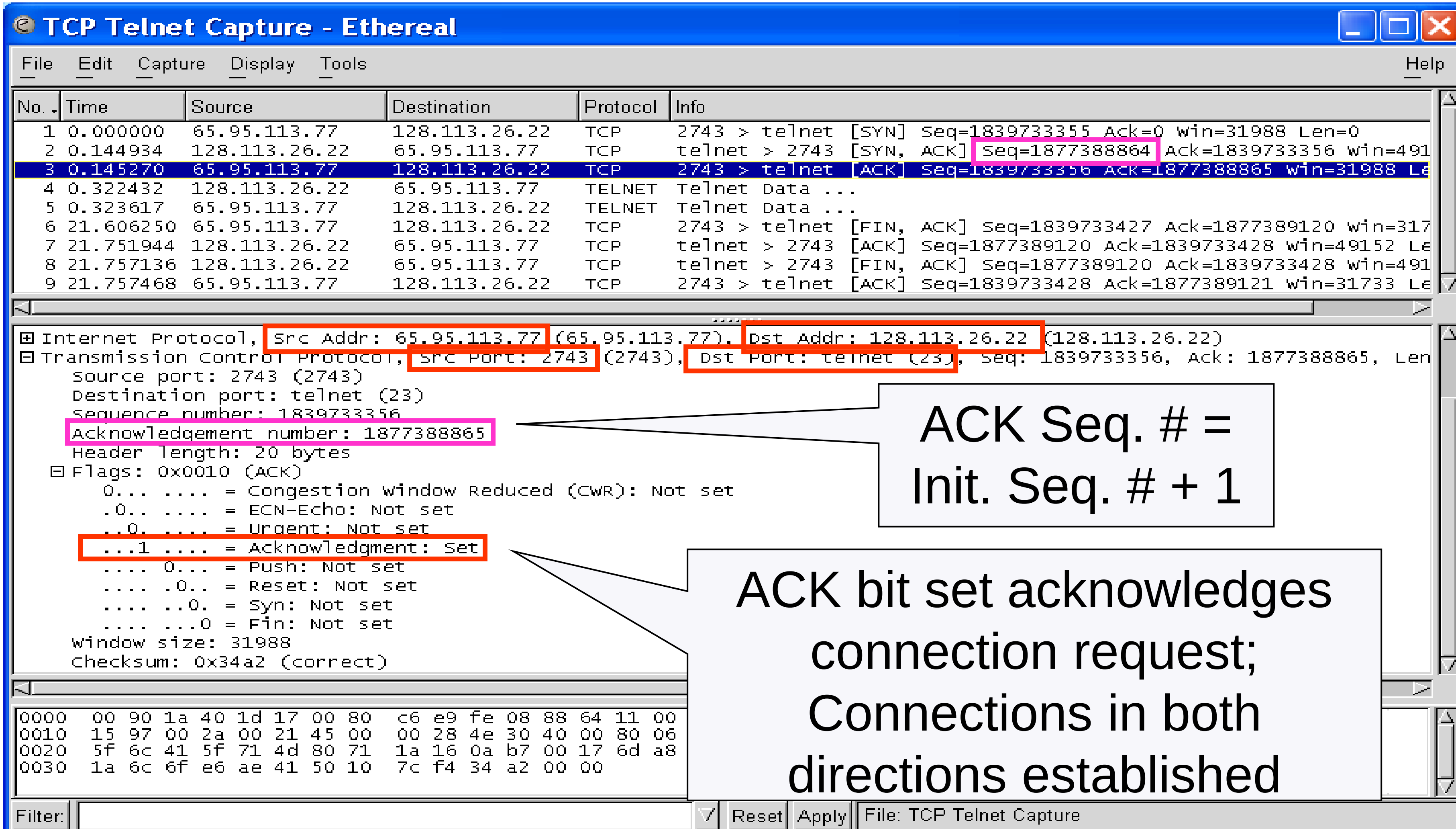
2nd Handshake: Server-Client Connection Request



Initial Seq. # from server to client

SYN bit set indicates request to establish connection from server to client

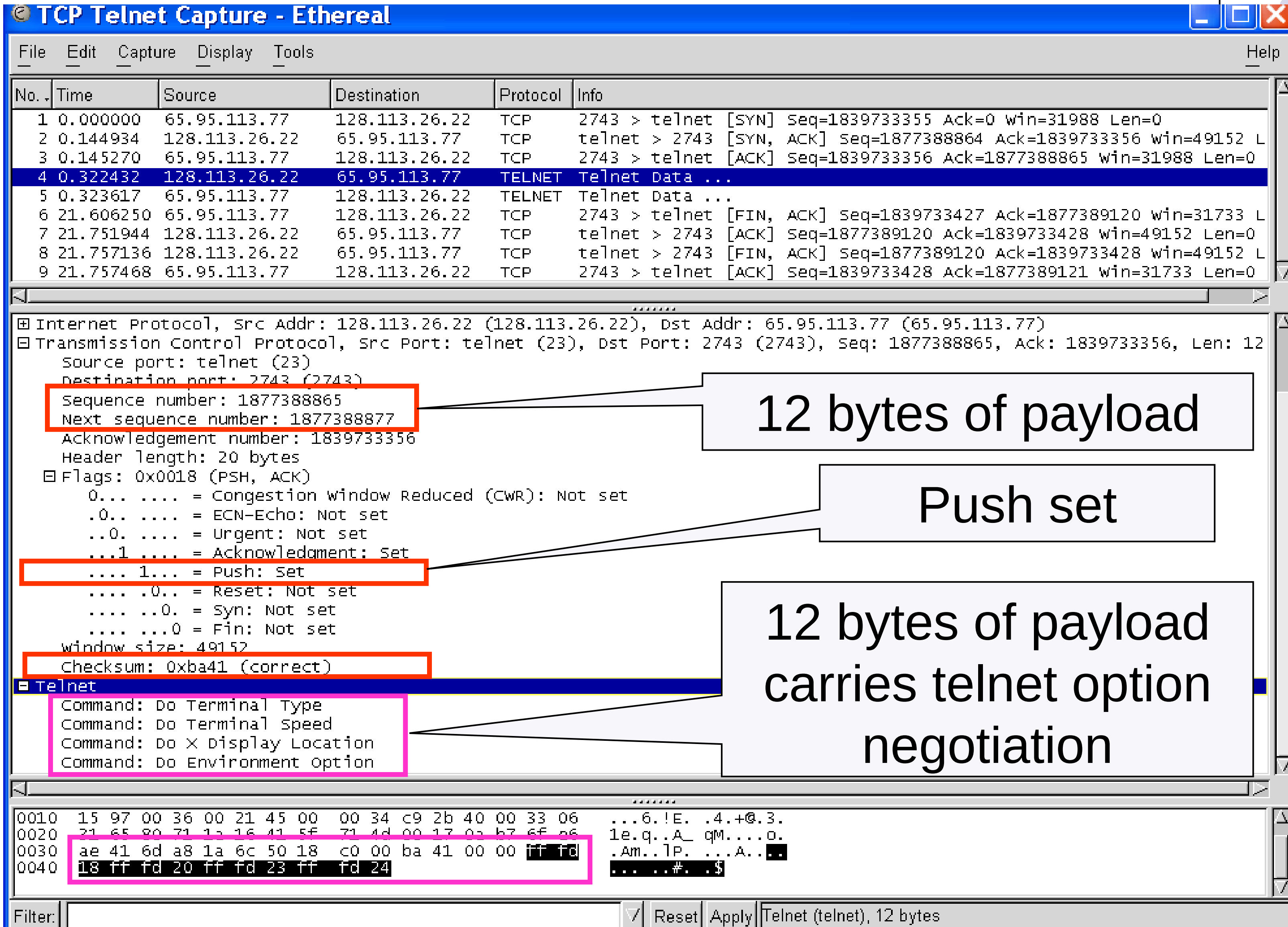
3rd Handshake: ACK from Client



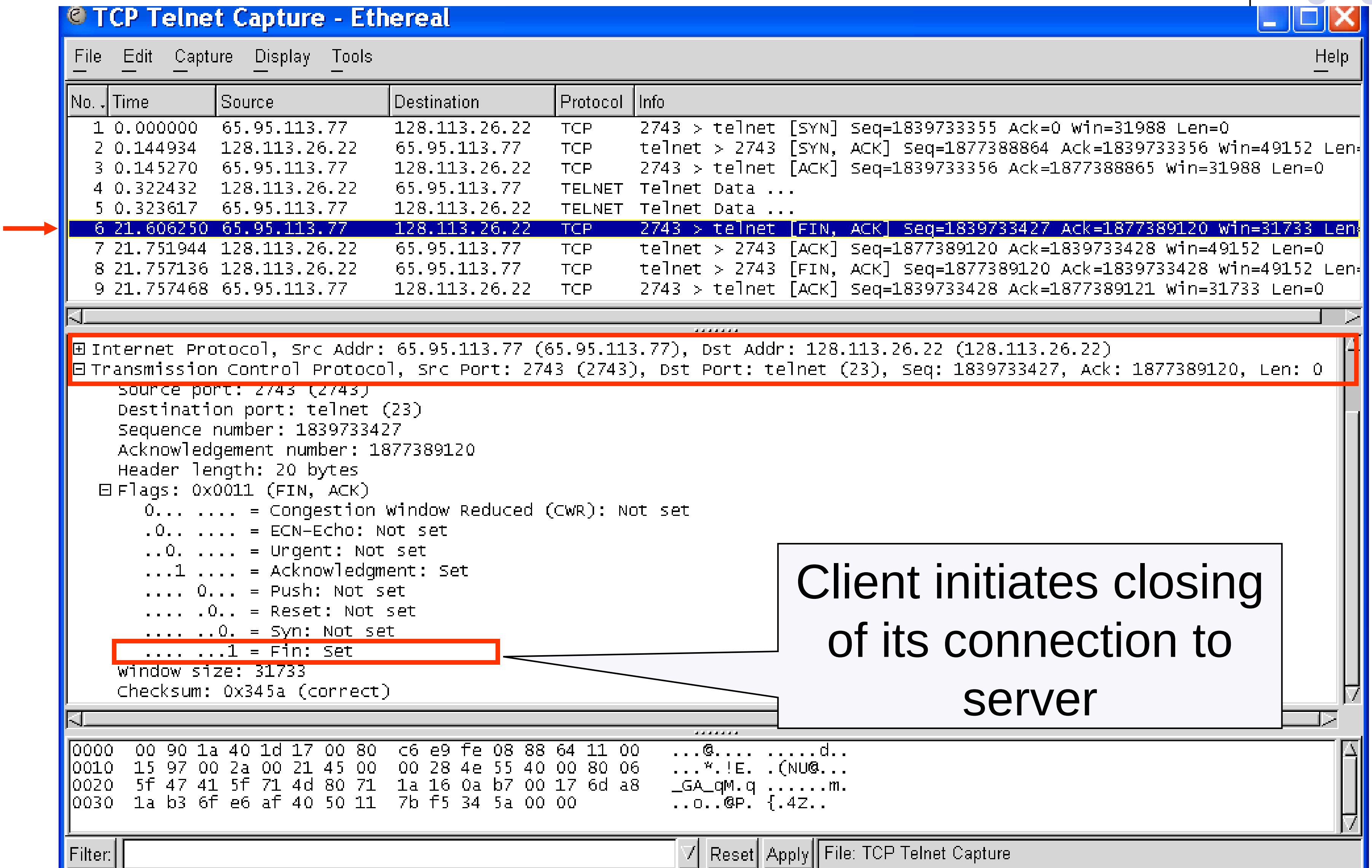
TCP Data Exchange

- Application Layers write bytes into buffers
- TCP sender forms segments
  - When bytes exceed threshold or timer expires
  - Upon PUSH command from applications
  - Consecutive bytes from buffer inserted in payload
  - Sequence # & ACK # inserted in header
  - Checksum calculated and included in header
- TCP receiver
  - Performs selective repeat ARQ functions
  - Writes error-free, in-sequence bytes to receive buffer

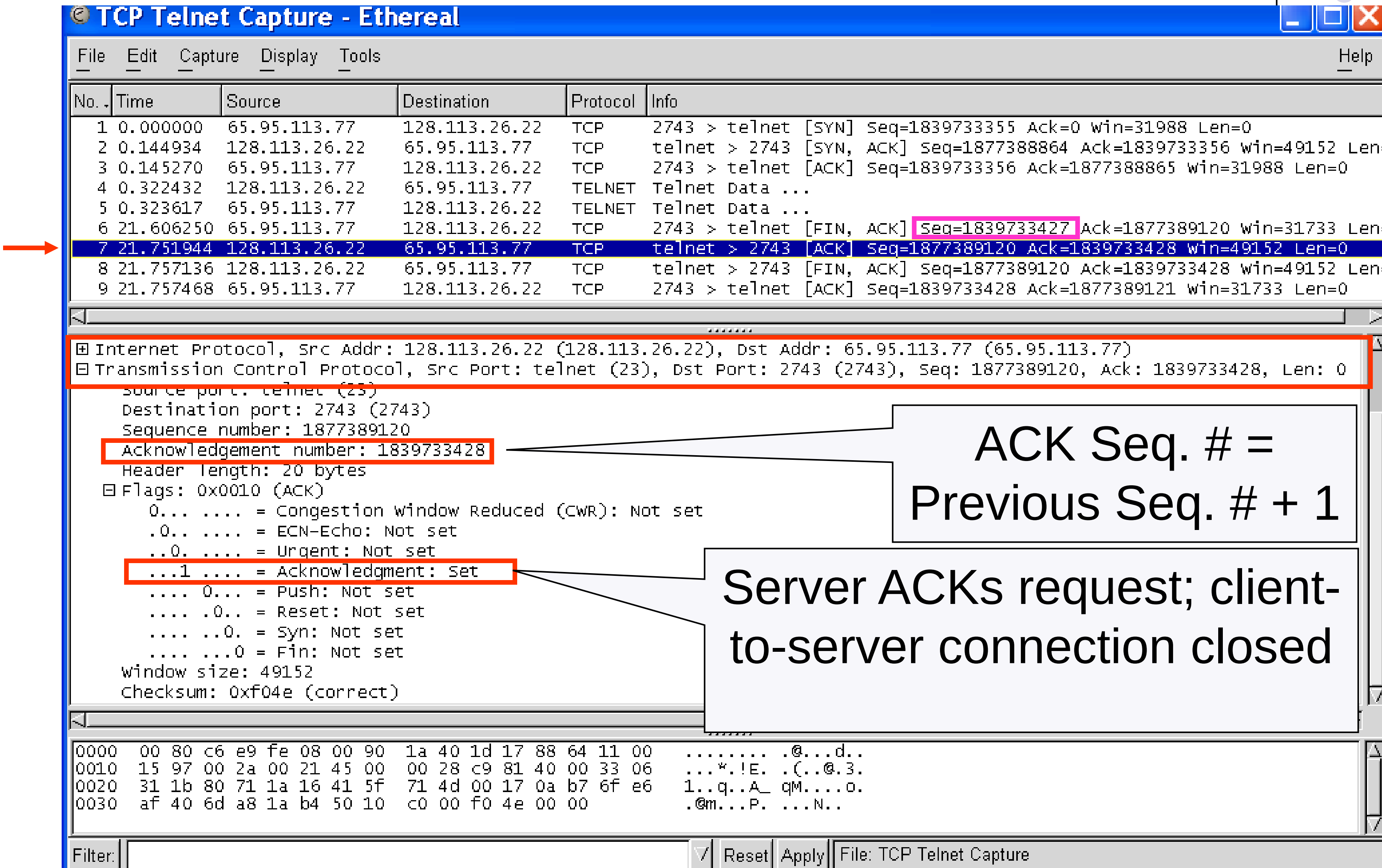
Data Transfer: Server-to-Client Segment



Graceful Close: Client-to-Server Connection



Graceful Close: Client-to-Server Connection



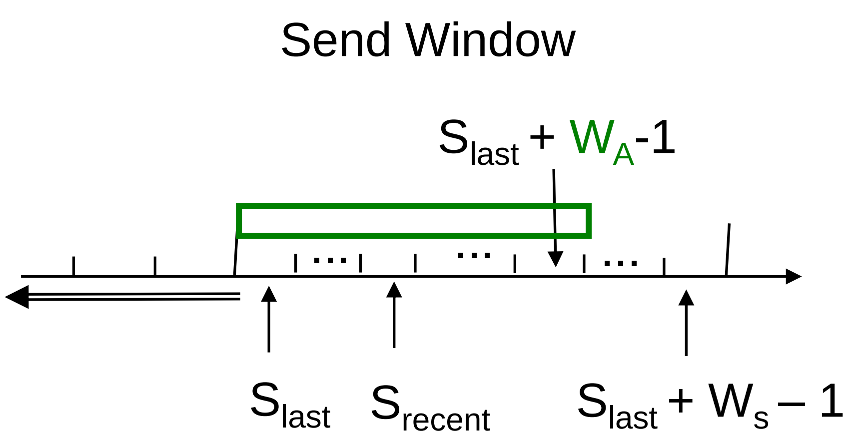
Flow Control

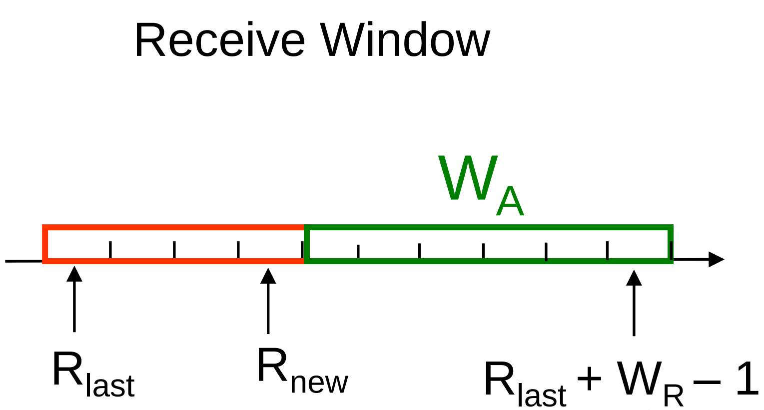
- TCP receiver controls rate at which sender transmits to prevent buffer overflow
- TCP receiver advertises a window size specifying number of bytes that can be accommodated by receiver

$$W_A = W_R - (R_{new} - R_{last} + 1)$$

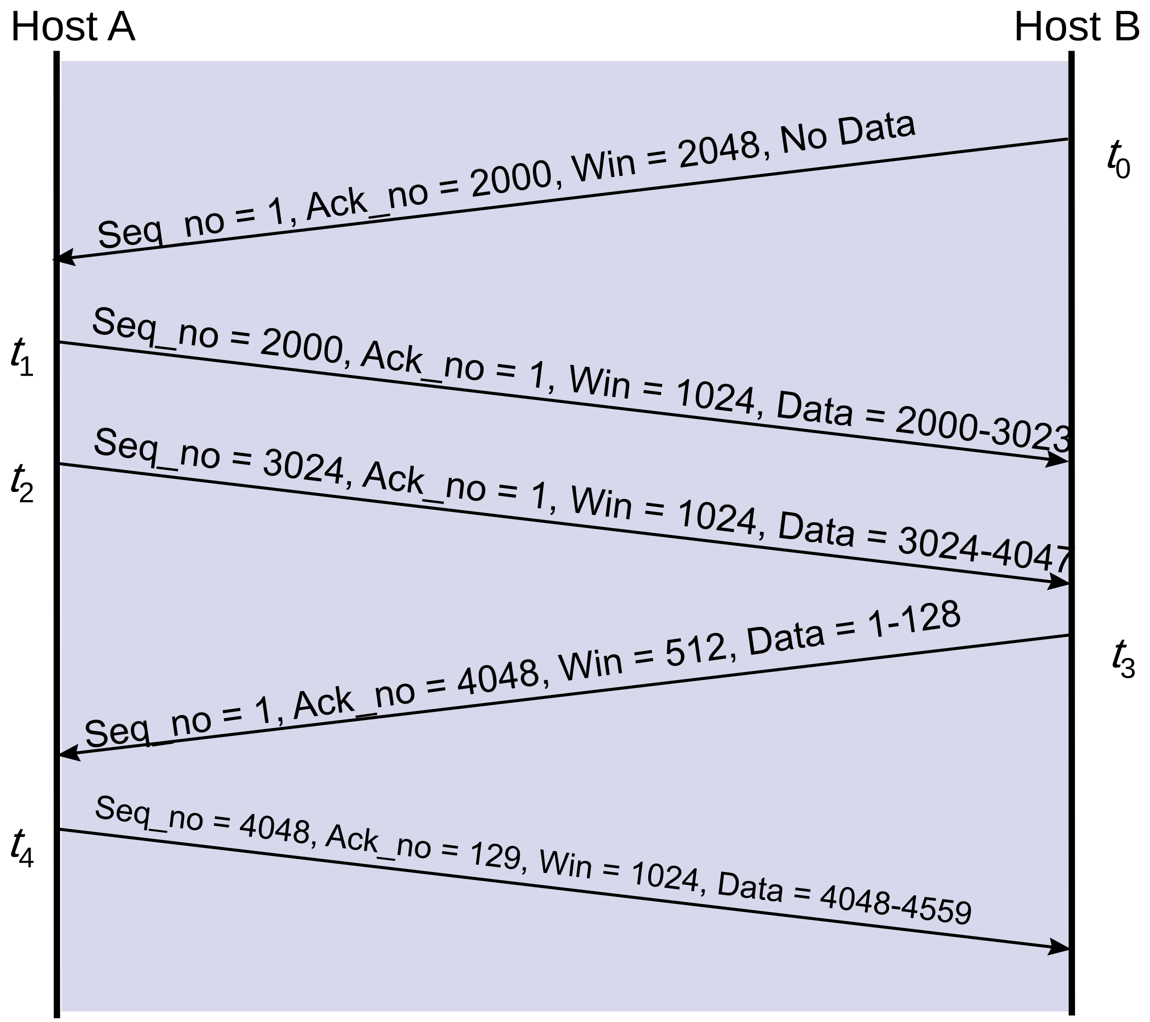
- TCP sender obliged to keep # outstanding bytes below W_A

$$(S_{recent} - S_{last}) \leq W_A - 1$$





TCP window flow control



TCP Retransmission Timeout

- TCP retransmits a segment after timeout period
  - Timeout too short: excessive number of retransmissions
  - Timeout too long: recovery too slow
  - Timeout depends on RTT: time from when segment is sent to when ACK is received
- Round trip time (RTT) in Internet is highly variable
  - Routes vary and can change in mid-connection
  - Traffic fluctuates
- TCP uses adaptive estimation of RTT
  - Measure RTT each time ACK received: τ_n

$$t_{RTT}(\text{new}) = \alpha\, t_{RTT}(\text{old}) + (1 - \alpha)\, \tau_n$$

  - $\alpha = 7/8$ typical

RTT Variability

- Estimate variance σ^2 of RTT variation
- Estimate for timeout:

$$t_{out} = t_{RTT} + k\,\sigma_{RTT}$$

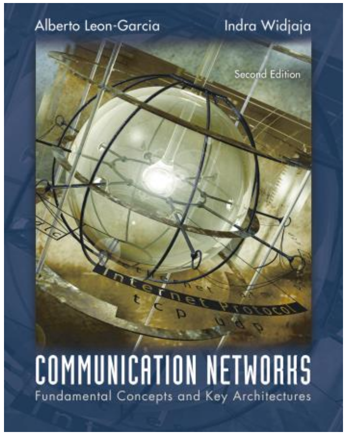
- If RTT highly variable, timeout increase accordingly
- If RTT nearly constant, timeout close to RTT estimate

- Approximate estimation of deviation

$$d_{RTT}(new) = \beta\, d_{RTT}(old) + (1-\beta)\,|\,\tau_n - t_{RTT}\,|$$

$$\boxed{t_{out} = t_{RTT} + 4\, d_{RTT}}$$

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



PART II: Data Link Controls

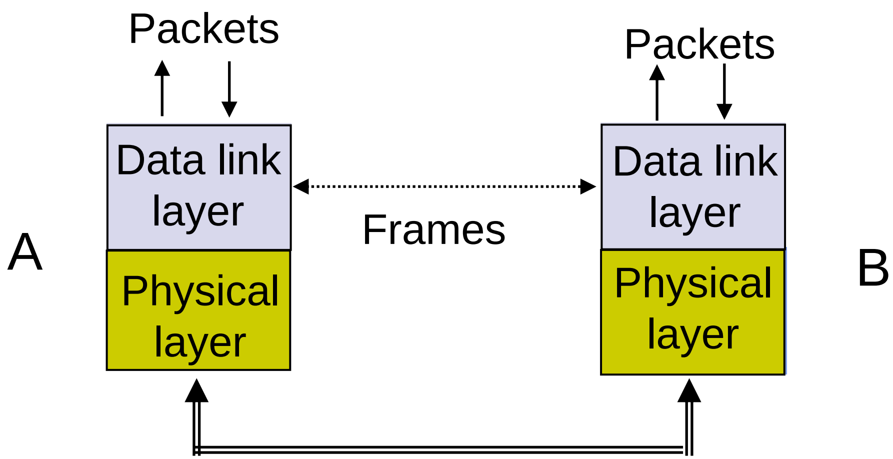
Framing

Point-to-Point Protocol

High-Level Data Link Control

Link Sharing Using Statistical Multiplexing

Data Link Protocols



Data Links Services

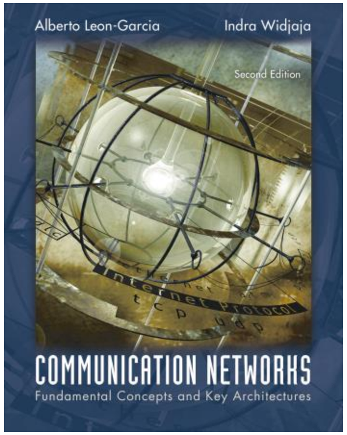
- Framing
- Error control
- Flow control
- Multiplexing
- Link Maintenance
- Security: Authentication & Encryption

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

Examples

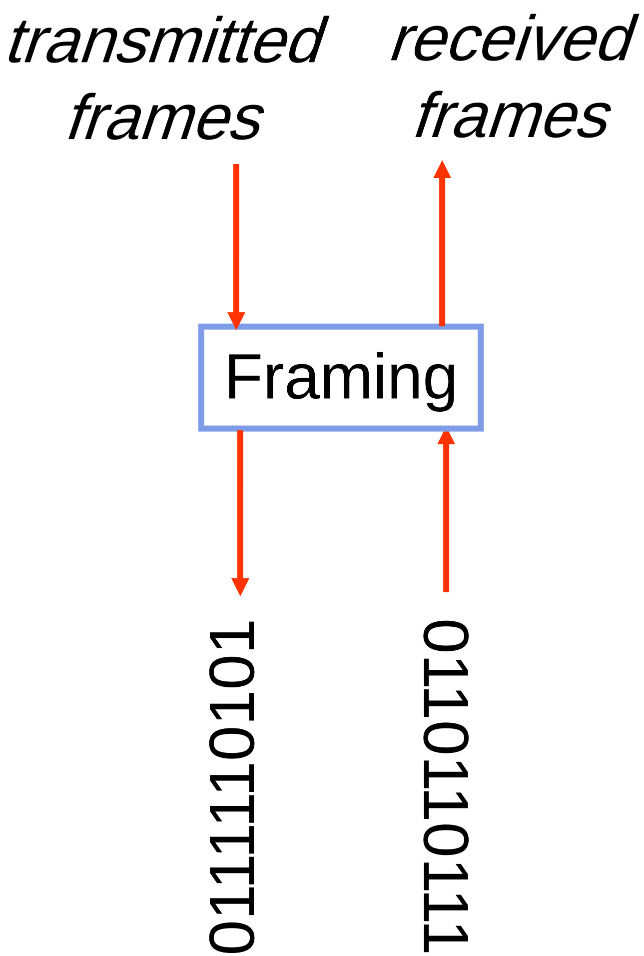
- PPP
- HDLC
- Ethernet LAN
- IEEE 802.11 (Wi Fi) LAN

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



Framing

Framing



- Mapping stream of physical layer bits into frames
- Mapping frames into bit stream
- Frame boundaries can be determined using:
  - Character Counts
  - Control Characters
  - Flags
  - CRC Checks

Character-Oriented Framing

Data to be sent

A	DLE	B	ETX	DLE	STX	E

After stuffing and framing

DLE	STX	A	DLE	DLE	B	ETX	DLE	DLE	STX	E	DLE	ETX

- Frames consist of integer number of bytes
  - Asynchronous transmission systems using ASCII to transmit printable characters
  - Octets with HEX value <20 are nonprintable
- Special 8-bit patterns used as control characters
  - STX (start of text) = 0x02; ETX (end of text) = 0x03;
- Byte used to carry non-printable characters in frame
  - DLE (data link escape) = 0x10
  - DLE STX (DLE ETX) used to indicate beginning (end) of frame
  - Insert extra DLE in front of occurrence of DLE in frame
  - All DLEs occur in pairs except at frame boundaries

Framing & Bit Stuffing

HDLC frame

Flag	Address	Control	Information	FCS	Flag

any number of bits

- 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: Bit stuffing & de-stuffing

(a)

Data to be sent

0110111111111100

After stuffing and framing

*01111110*011011111011111000*01111110*

(b)

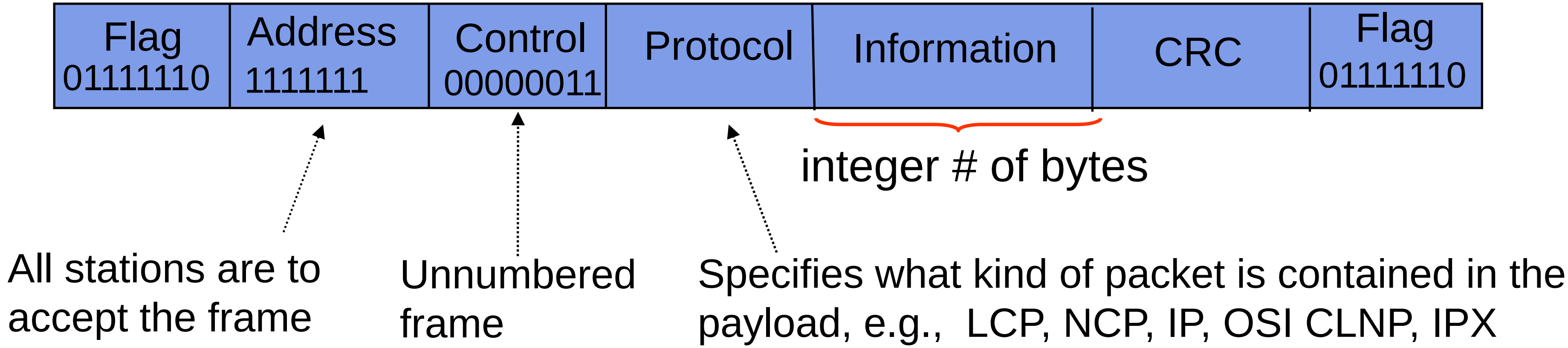
Data received

01111110000111011111011111011001111110

After destuffing and deframing

000111011111-11111-110

PPP Frame



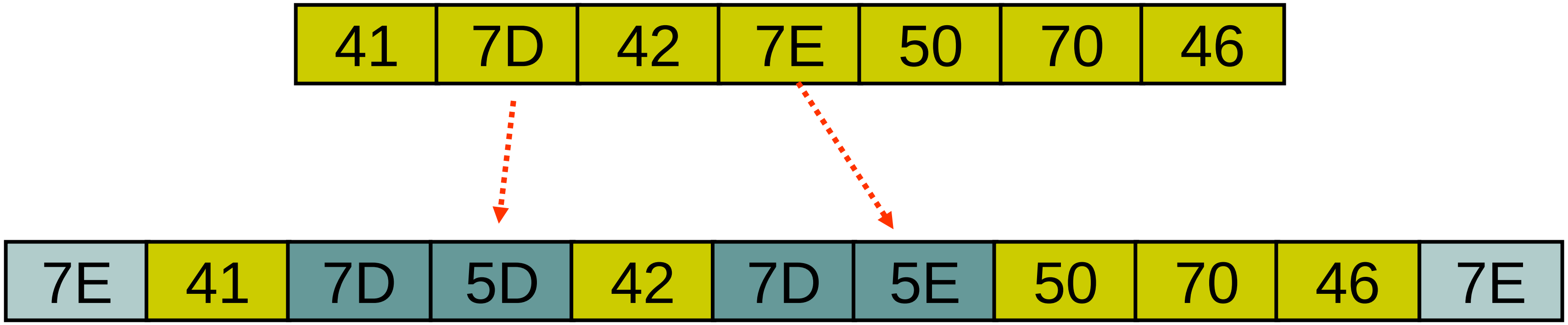
- PPP uses similar frame structure as HDLC, except
  - Protocol type field
  - Payload contains an *integer* number of bytes
- PPP uses the same flag, but uses *byte stuffing*
- Problems with PPP byte stuffing
  - Size of frame varies unpredictably due to byte insertion
  - Malicious users can inflate bandwidth by inserting 7D & 7E

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



After stuffing and framing

Generic Framing Procedure

GFP payload area

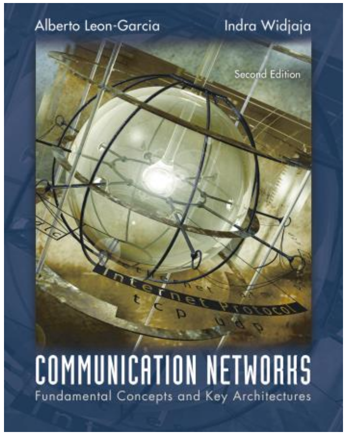
2	2	2	2	0-60	
PLI	cHEC	Type	tHEC	GEH	GFP payload
Payload length indicator	Core header error checking	Payload type	Type header error checking	GFP extension headers	GFP payload

- GFP combines frame length indication with CRC
  - PLI indicated length of frame, then simply count characters
  - cHEC (CRC-16) protects against errors in count field (single-bit error correction + error detection)
- GFP designed to operate over octet-synchronous physical layers (e.g. SONET)
  - Frame-mapped mode for variable-length payloads: Ethernet
  - Transparent mode carries fixed-length payload: storage devices

GFP Synchronization & Scrambling

- Synchronization in three-states
  - *Hunt state*: examine 4-bytes to see if CRC ok
    - If no, move forward by one-byte
    - If yes, move to pre-sync state
  - *Pre-sync state*: tentative PLI indicates next frame
    - If N successful frame detections, move to sync state
    - If no match, go to hunt state
  - *Sync state*: normal state
    - Validate PLI/cHEC, extract payload, go to next frame
    - Use single-error correction
    - Go to hunt state if non-correctable error
- Scrambling
  - Payload is scrambled to prevent malicious users from inserting long strings of 0s which cause SONET equipment to lose bit clock synchronization (as discussed in line code section)

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



Point-to-Point Protocol

PPP: Point-to-Point Protocol

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

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

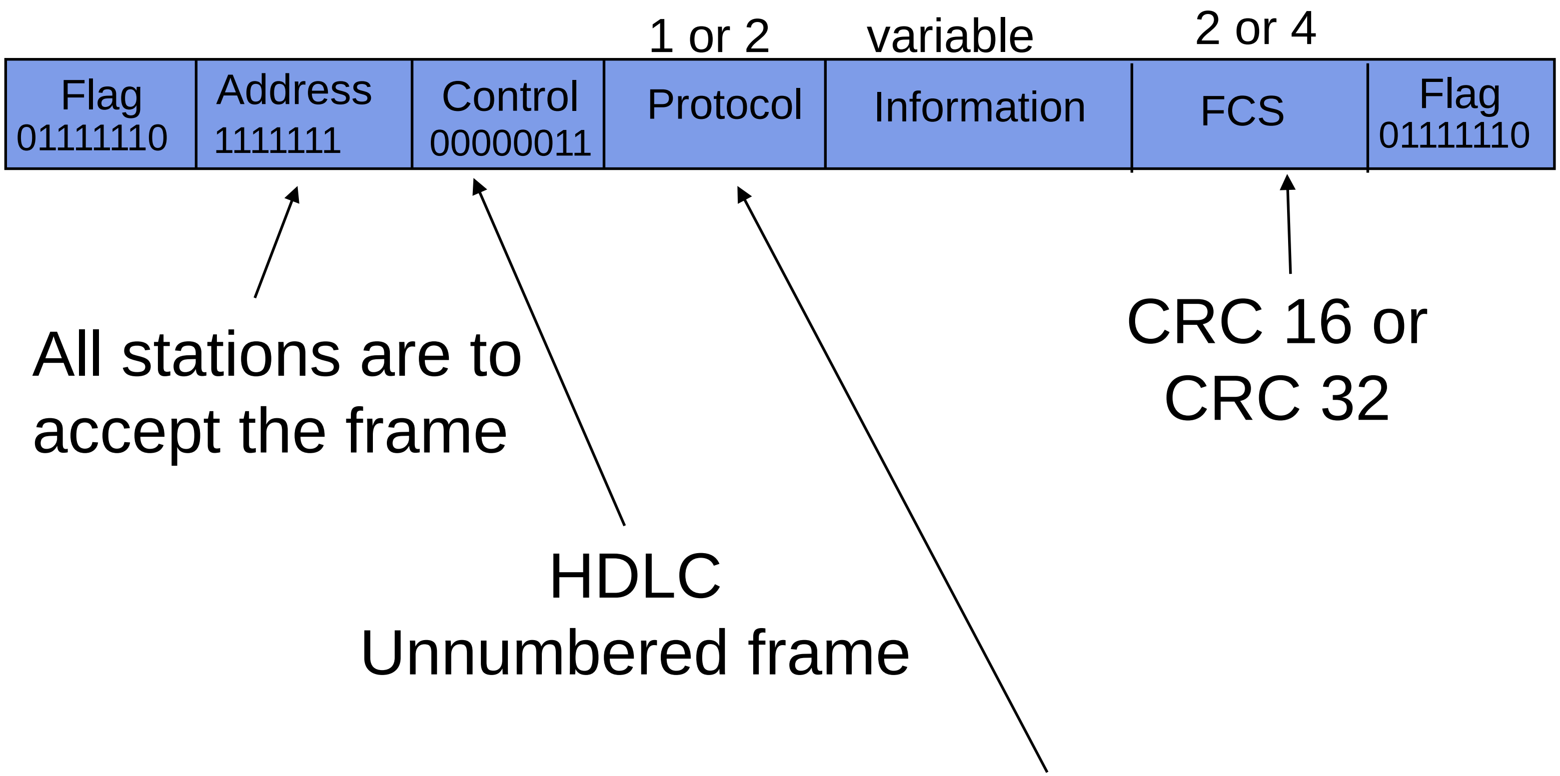
PPP Applications

PPP used in many point-to-point applications

- Telephone Modem Links 30 kbps
- Packet over SONET 600 Mbps to 10 Gbps
  - IP→PPP→SONET

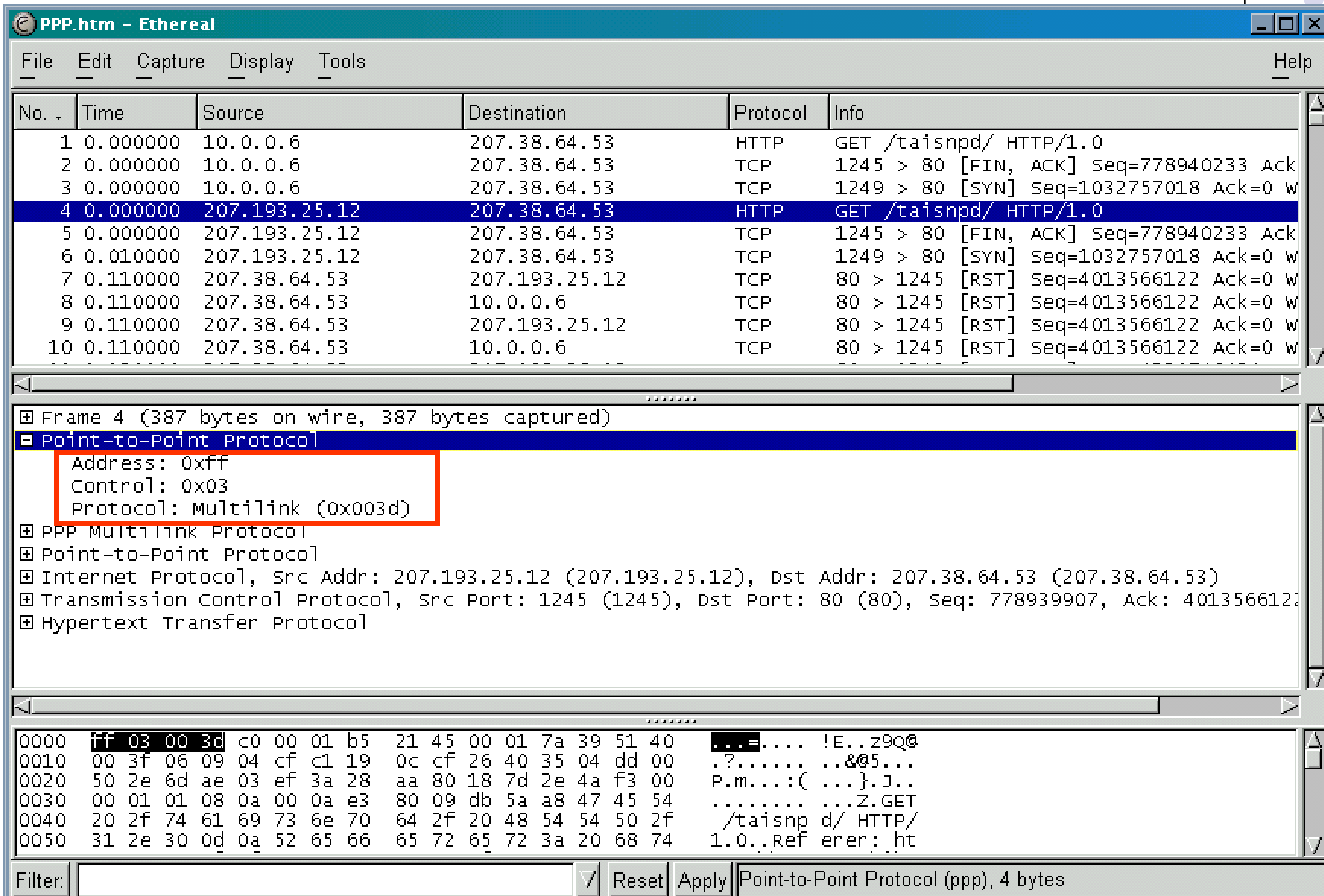
- PPP is also used over shared links such as Ethernet to provide LCP, NCP, and authentication features
  - PPP over Ethernet (RFC 2516)
  - Used over DSL

PPP Frame Format



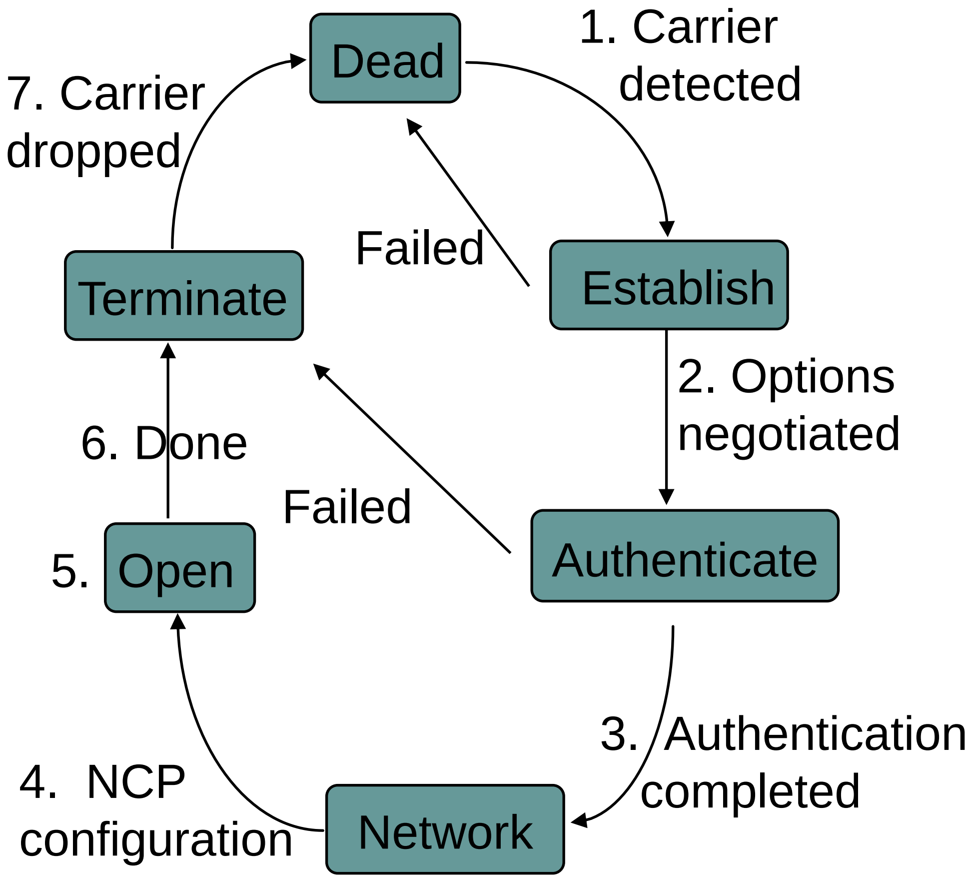
- PPP can support multiple network protocols simultaneously
- Specifies what kind of packet is contained in the payload
  - e.g. LCP, NCP, IP, OSI CLNP, IPX...

PPP Example



PPP.htm - Ethereal
File Edit Capture Display Tools Help
No. Time Source Destination Protocol Info
1 0.000000 10.0.0.6 207.38.64.53 HTTP GET /taisnpd/ HTTP/1.0
2 0.000000 10.0.0.6 207.38.64.53 TCP 1245 > 80 [FIN, ACK] Seq=778940233 Ack
3 0.000000 10.0.0.6 207.38.64.53 TCP 1249 > 80 [SYN] Seq=1032757018 Ack=0 W
4 0.000000 207.193.25.12 207.38.64.53 HTTP GET /taisnpd/ HTTP/1.0
5 0.000000 207.193.25.12 207.38.64.53 TCP 1245 > 80 [FIN, ACK] Seq=778940233 Ack
6 0.010000 207.193.25.12 207.38.64.53 TCP 1249 > 80 [SYN] Seq=1032757018 Ack=0 W
7 0.110000 207.38.64.53 207.193.25.12 TCP 80 > 1245 [RST] Seq=4013566122 Ack=0 W
8 0.110000 207.38.64.53 10.0.0.6 TCP 80 > 1245 [RST] Seq=4013566122 Ack=0 W
9 0.110000 207.38.64.53 207.193.25.12 TCP 80 > 1245 [RST] Seq=4013566122 Ack=0 W
10 0.110000 207.38.64.53 10.0.0.6 TCP 80 > 1245 [RST] Seq=4013566122 Ack=0 W
Frame 4 (387 bytes on wire, 387 bytes captured)
Point-to-Point Protocol
Address: 0xff
Control: 0x03
Protocol: Multilink (0x003d)
PPP Multilink Protocol
Point-to-Point Protocol
Internet Protocol, Src Addr: 207.193.25.12 (207.193.25.12), Dst Addr: 207.38.64.53 (207.38.64.53)
Transmission Control Protocol, Src Port: 1245 (1245), Dst Port: 80 (80), Seq: 778939907, Ack: 401356612
Hypertext Transfer Protocol
0000 ff 03 00 3d c0 00 01 b5 21 45 00 01 7a 39 51 40 ...=.... !E..z9Q@
0010 00 3f 06 09 04 cf c1 19 0c cf 26 40 35 04 dd 00 .?...... ..&@5...
0020 50 2e 6d ae 03 ef 3a 28 aa 80 18 7d 2e 4a f3 00 P.m...:(...}.J..
0030 00 01 01 08 0a 00 0a e3 80 09 db 5a a8 47 45 54 ....Z.GET
0040 20 2f 74 61 69 73 6e 70 64 2f 20 48 54 54 50 2f /taisnp d/ HTTP/
0050 31 2e 30 0d 0a 52 65 66 65 72 65 72 3a 20 68 74 1.0..Ref erer: ht
Filter: Reset Apply Point-to-Point Protocol (ppp), 4 bytes

PPP Phases



Home PC to Internet Service Provider

1. PC calls router via modem
2. PC and router exchange LCP packets to negotiate PPP parameters
3. Check on identities
4. NCP packets exchanged to configure the network layer, e.g. TCP/IP (requires IP address assignment)
5. Data transport, e.g. send/receive IP packets
6. NCP used to tear down the network layer connection (free up IP address); LCP used to shut down data link layer connection
7. Modem hangs up

PPP Authentication

- Password Authentication Protocol
  - Initiator must send ID & password
  - Authenticator replies with authentication success/fail
  - After several attempts, LCP closes link
  - Transmitted unencrypted, susceptible to eavesdropping
- Challenge-Handshake Authentication Protocol (CHAP)
  - Initiator & authenticator share a secret key
  - Authenticator sends a challenge (random # & ID)
  - Initiator computes cryptographic checksum of random # & ID using the shared secret key
  - Authenticator also calculates cryptocgraphic checksum & compares to response
  - Authenticator can reissue challenge during session

Example: PPP connection setup in dialup modem to ISP

PPP LCP and NCP Negotiation - Ethereal

File Edit Capture Display Tools Help

No.	Time	Source	Destination	Protocol	Info
1	0.000000	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP LCP	PPP LCP Configuration Request
2	2.999526	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP LCP	PPP LCP Configuration Request
3	3.130440	20:52:45:43:56:00	20:52:45:43:56:00	PPP LCP	PPP LCP Configuration Reject
4	3.130495	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP LCP	PPP LCP Configuration Request
5	3.243457	20:52:45:43:56:00	20:52:45:43:56:00	PPP LCP	PPP LCP Configuration Ack
6	5.096025	20:52:45:43:56:00	20:52:45:43:56:00	PPP LCP	PPP LCP Configuration Request
7	5.096072	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP LCP	PPP LCP Configuration Reject
8	5.220084	20:52:45:43:56:00	20:52:45:43:56:00	PPP LCP	PPP LCP Configuration Request
9	5.220127	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP LCP	PPP LCP Configuration Ack
10	5.220155	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP PAP	PPP PAP Authenticate-Request
11	5.423283	20:52:45:43:56:00	20:52:45:43:56:00	PPP PAP	PPP PAP Authenticate-Ack
12	5.423367	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP IPCP	PPP IPCP Configuration Request
13	5.423390	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP CCP	PPP CCP Configuration Request
14	5.428998	20:52:45:43:56:00	20:52:45:43:56:00	PPP IPCP	PPP IPCP Configuration Request
15	5.429038	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP IPCP	PPP IPCP Configuration Ack
16	5.558729	20:52:45:43:56:00	20:52:45:43:56:00	PPP IPCP	PPP IPCP Configuration Reject
17	5.558785	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP IPCP	PPP IPCP Configuration Request
18	5.564373	20:52:45:43:56:00	20:52:45:43:56:00	PPP LCP	PPP LCP Protocol Reject
19	5.699896	20:52:45:43:56:00	20:52:45:43:56:00	PPP IPCP	PPP IPCP Configuration Nak
20	5.699938	20:53:45:4e:44:00	20:53:45:4e:44:00	PPP IPCP	PPP IPCP Configuration Request

LCP Setup

PAP

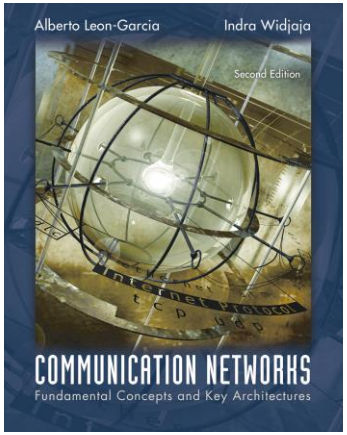
IP NCP setup

```
⊞ Frame 9 (38 bytes on wire, 38 bytes captured)
⊞ Ethernet II, Src: 20:53:45:4e:44:00, Dst: 20:53:45:4e:44:00
⊟ PPP Link Control Protocol
    Code: Configuration Ack (0x02)
    Identifier: 0x83
    Length: 24
  ⊟ Options: (20 bytes)
      Async Control Character Map: 0x000a0000 (DC1 (XON), DC3 (XOFF))
    ⊟ Authentication protocol: 4 bytes
        Authentication protocol: Password Authentication Protocol (0xc023)
      Magic number: 0x218ad821
      Protocol field compression
      Address/control field compression
```

```
0000  20 53 45 4e 44 00 20 53  45 4e 44 00 c0 21 02 83    SEND. S END..!..
0010  00 18 02 06 00 0a 00 00  03 04 c0 23 05 06 21 8a   ........ ...#..!.
0020  d8 21 07 02 08 02                                   .!....
```

Filter: Reset Apply File: PPP LCP and NCP Negotiation

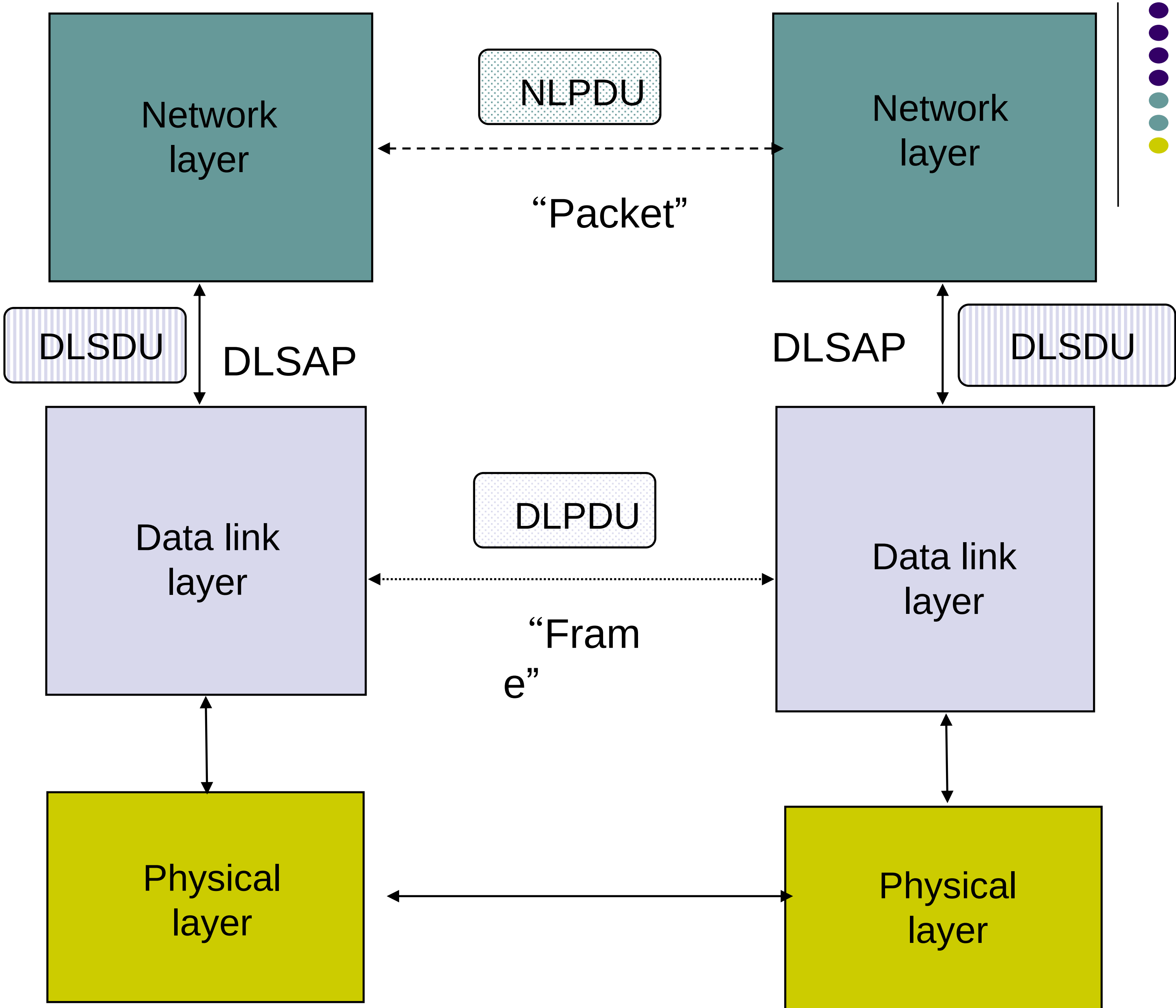
Chapter 5
Peer-to-Peer Protocols and Data Link Layer



High-Level Data Link Control

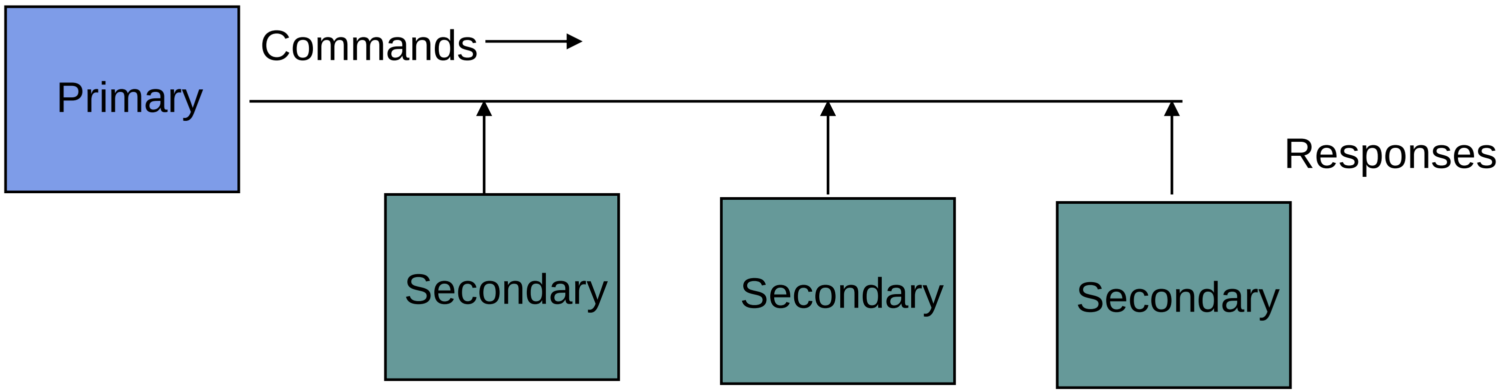
High-Level Data Link Control (HDLC)

- Bit-oriented data link control
- Derived from IBM Synchronous Data Link Control (SDLC)
- Related to Link Access Procedure Balanced (LAPB)
  - LAPD in ISDN
  - LAPM in cellular telephone signaling

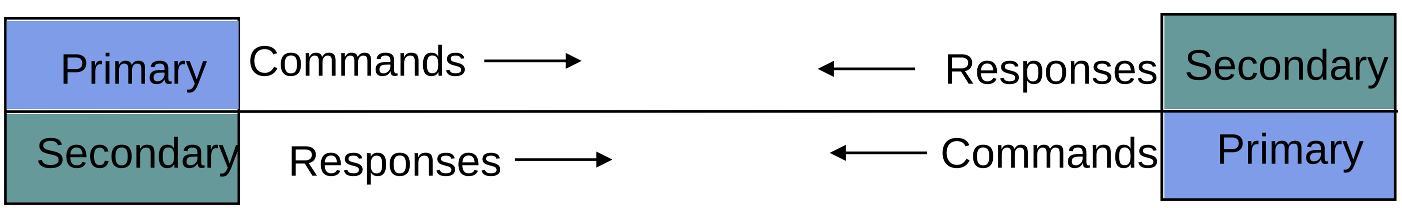

Network layer
NLPDU
Network layer
"Packet"
DLSDU
DLSAP
DLSAP
DLSDU
Data link layer
DLPDU
Data link layer
"Fram e"
Physical layer
Physical layer

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 octets)
    - In ABM mode, a station can act as primary or secondary so address changes accordingly
  - Control: purpose & functions of frame (1 or 2 octets)
  - Information: contains user data; length not standardized, but implementations impose maximum
  - Frame Check Sequence: 16- or 32-bit CRC

Control Field Format

Information Frame

1	2-4	5	6-8
0	N(S)	P/F	N(R)

Supervisory Frame

1	0	S	S	P/F	N(R)

Unnumbered Frame

1	1	M	M	P/F	M	M	M

- S: Supervisory Function Bits
- N(R): Receive Sequence Number
- N(S): Send Sequence Number
- M: Unnumbered Function Bits
- P/F: Poll/final bit used in interaction between primary and secondary

Information frames

- Each I-frame contains sequence number N(S)
- Positive ACK piggybacked
  - N(R)=Sequence number of *next* frame expected acknowledges all frames up to and including N(R)-1
- 3 or 7 bit sequence numbering
  - Maximum window sizes 7 or 127
- Poll/Final Bit
  - NRM: Primary polls station by setting P=1; Secondary sets F=1 in *last* I-frame in response
  - Primaries and secondaries always interact via *paired* P/F bits

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

Supervisory frames

Used for error (ACK, NAK) and flow control (Don't Send):

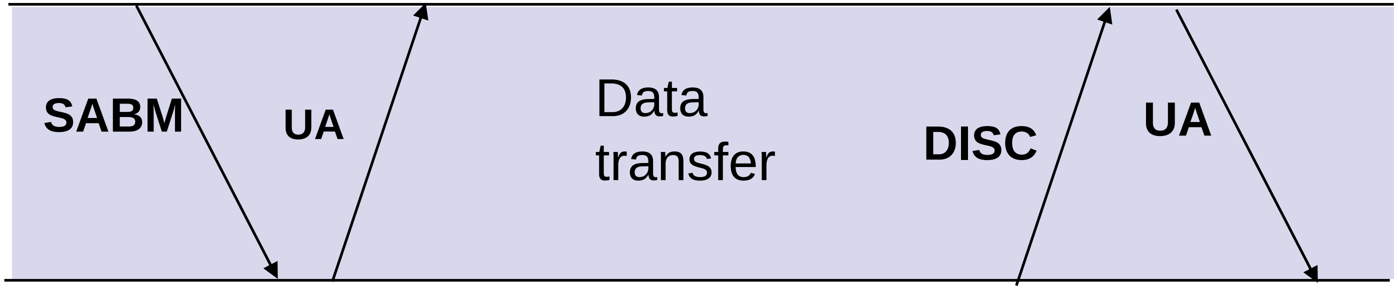
- *Receive Ready* (RR), SS=00
  - ACKs frames up to N(R)-1 when piggyback not available
- *REJECT* (REJ), SS=01
  - Negative ACK indicating N(R) is first frame not received correctly. Transmitter must resend N(R) and later frames
- *Receive Not Ready* (RNR), SS=10
  - ACKs frame N(R)-1 & requests that no more I-frames be sent
- *Selective REJECT* (SREJ), SS=11
  - Negative ACK for N(R) requesting that N(R) be selectively retransmitted

Unnumbered Frames

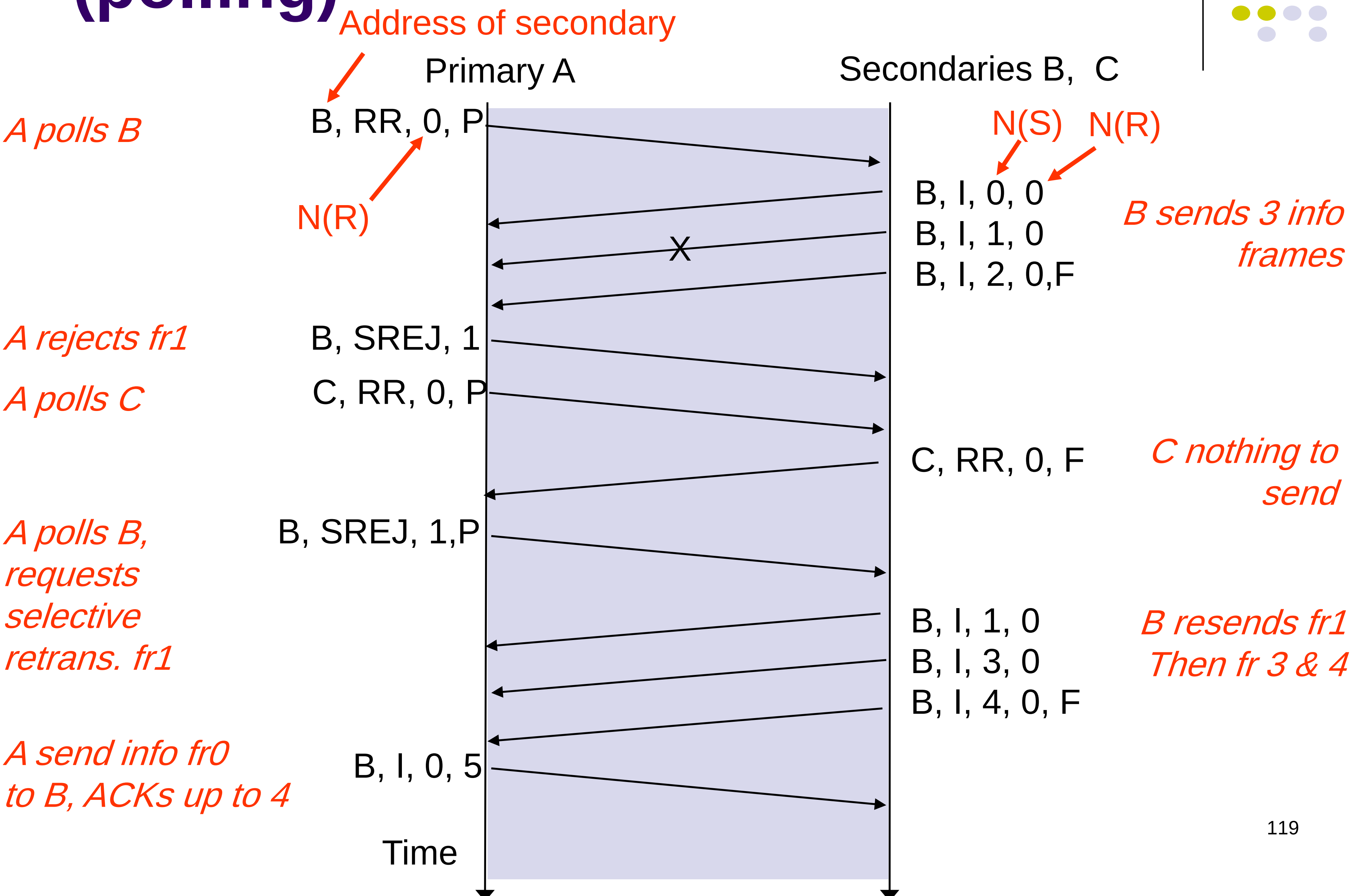
- Setting of Modes:
  - SABM: Set Asynchronous Balanced Mode
  - UA: acknowledges acceptance of mode setting commands
  - DISC: terminates logical link connectio
- Information Transfer between stations
  - UI: Unnumbered information
- Recovery used when normal error/flow control fails
  - FRMR: frame with correct FCS but impossible semantics
  - RSET: indicates sending station is resetting sequence numbers
- XID: exchange station id and characteristics

Connection Establishment & Release

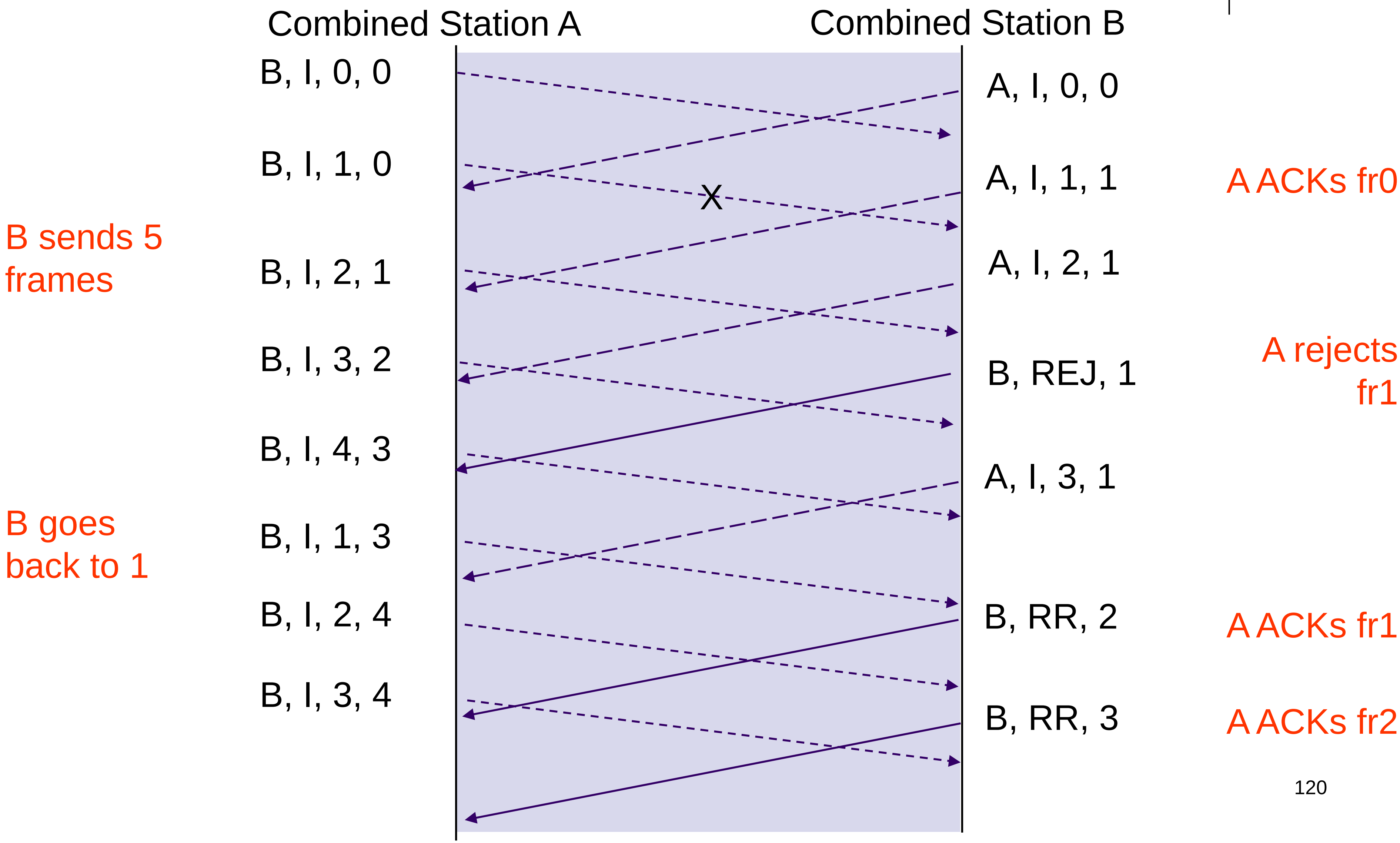
- Unnumbered frames used to establish and release data link connection
- In HDLC
  - Set Asynchronous Balanced Mode (SABM)
  - Disconnect (DISC)
  - Unnumbered Acknowledgment (UA)



Example: HDLC using NRM (polling)

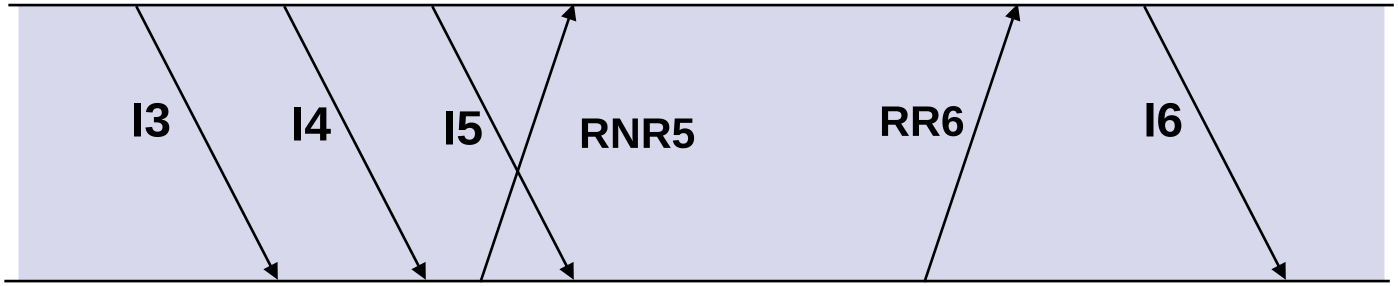


Frame Exchange using Asynchronous Balanced Mode

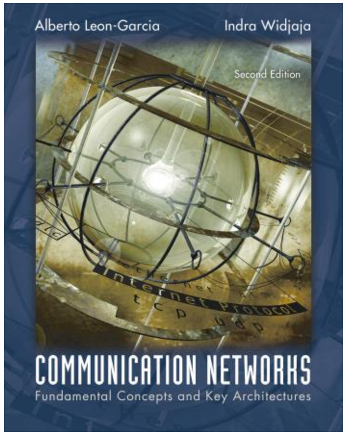


Flow Control

- Flow control is required to prevent transmitter from overrunning receiver buffers
- Receiver can control flow by delaying acknowledgement messages
- Receiver can also use supervisory frames to explicitly control transmitter
  - Receive Not Ready (RNR) & Receive Ready (RR)



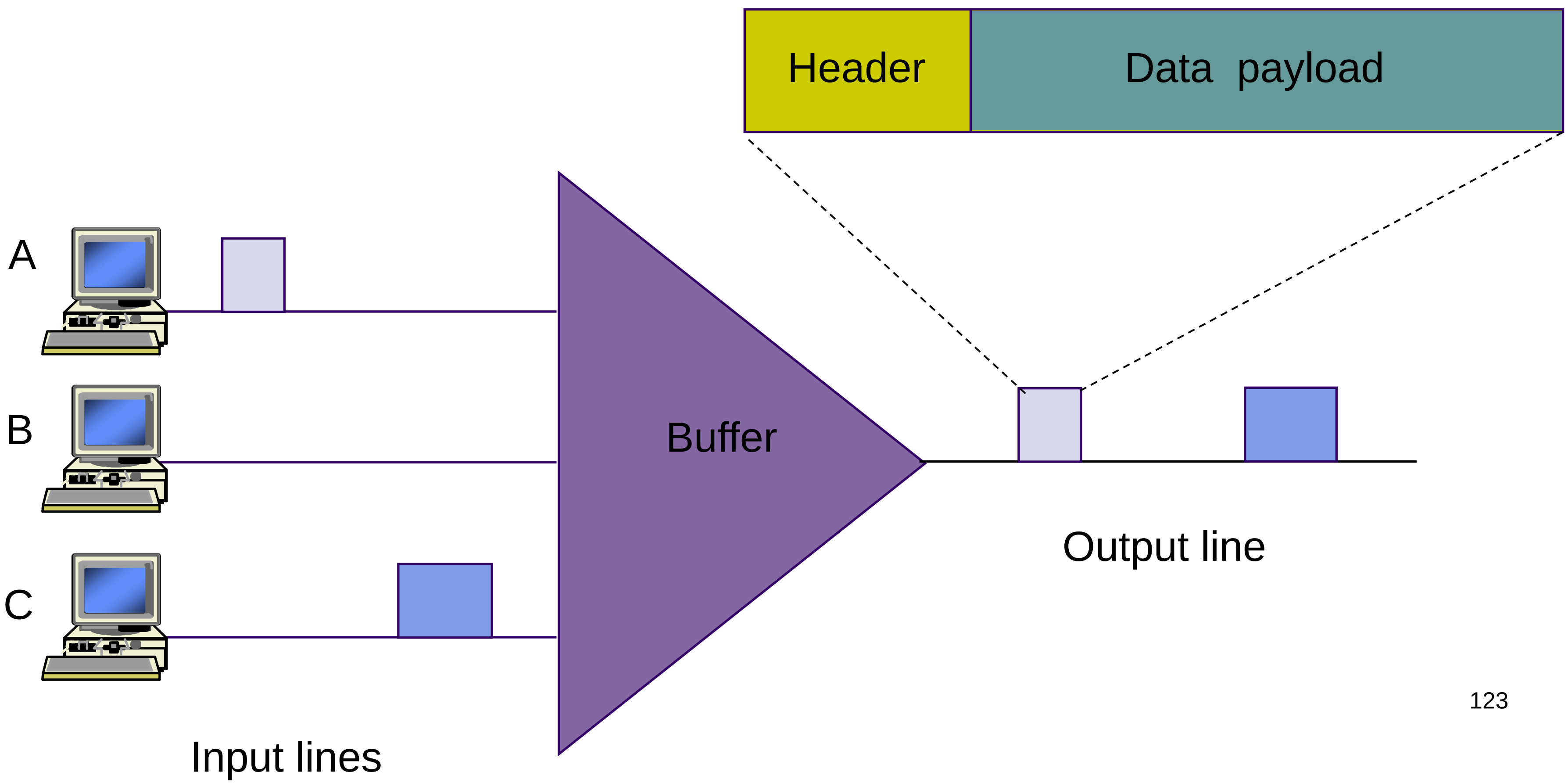
Chapter 5
Peer-to-Peer Protocols and Data Link Layer



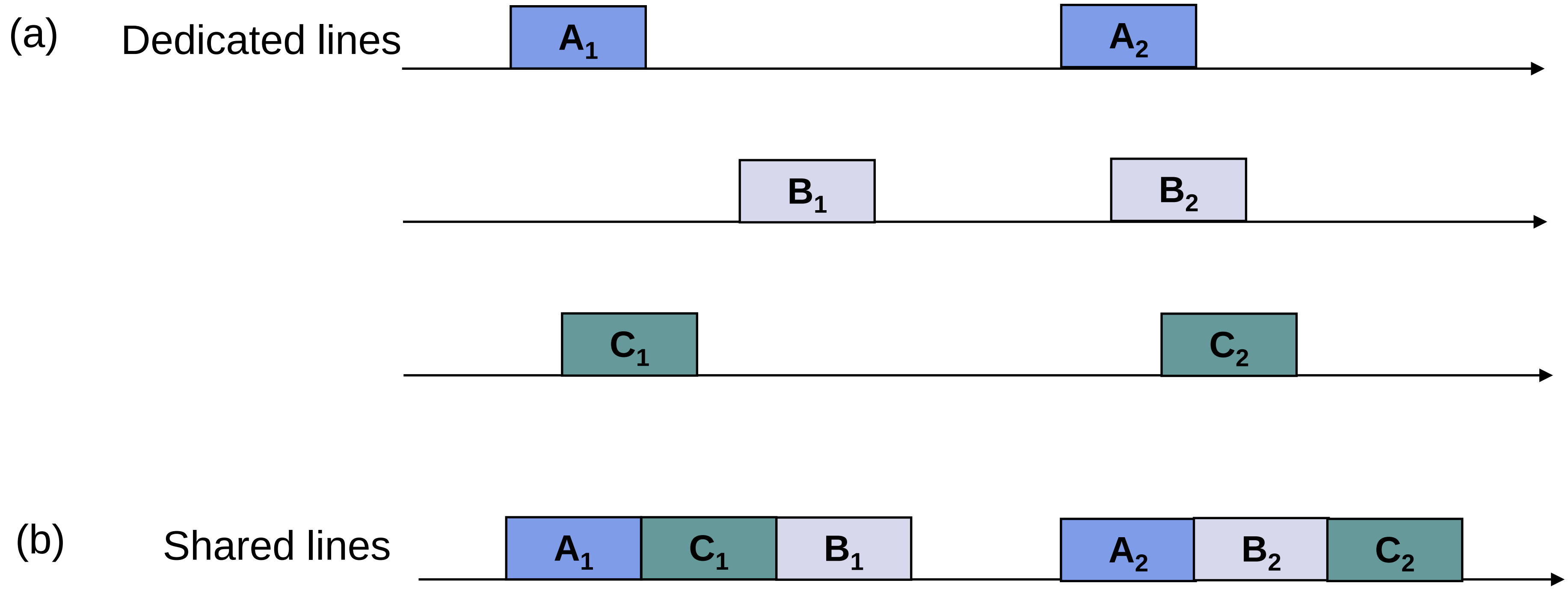
Link Sharing Using Statistical Multiplexing

Statistical Multiplexing

- Multiplexing concentrates bursty traffic onto a shared line
- Greater efficiency and lower cost

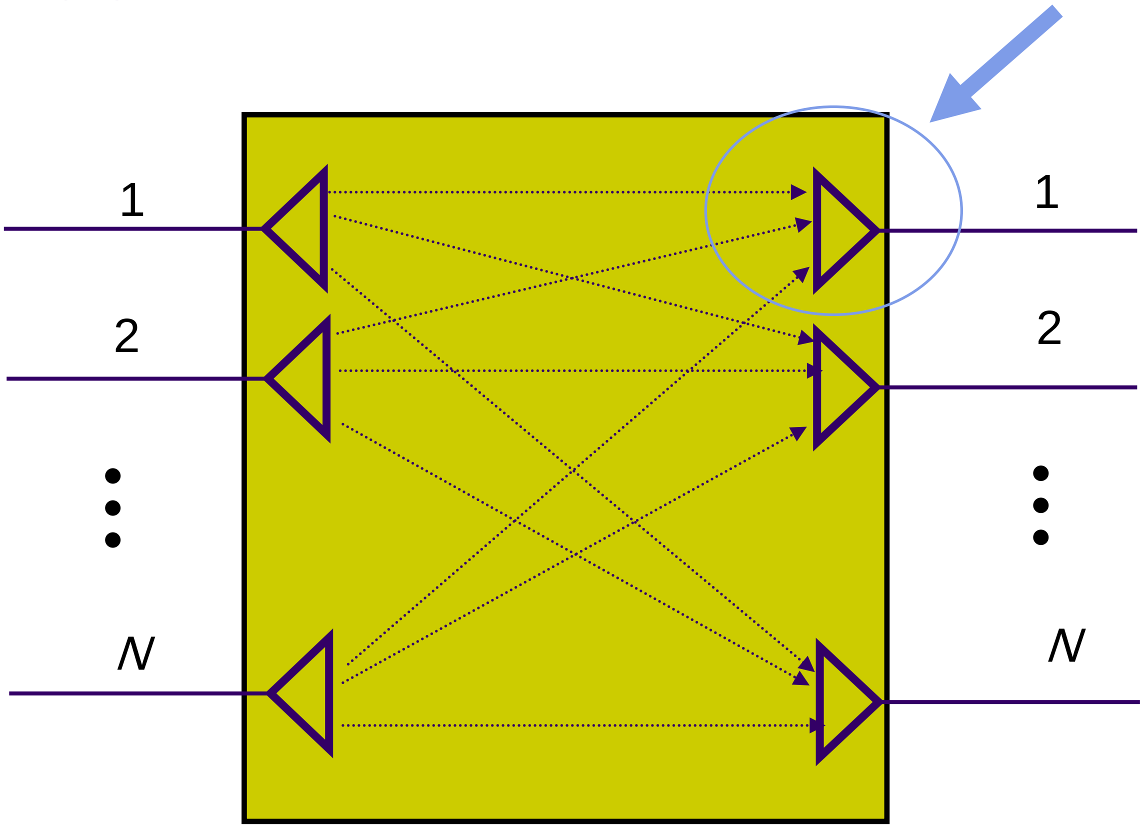


Tradeoff Delay for Efficiency



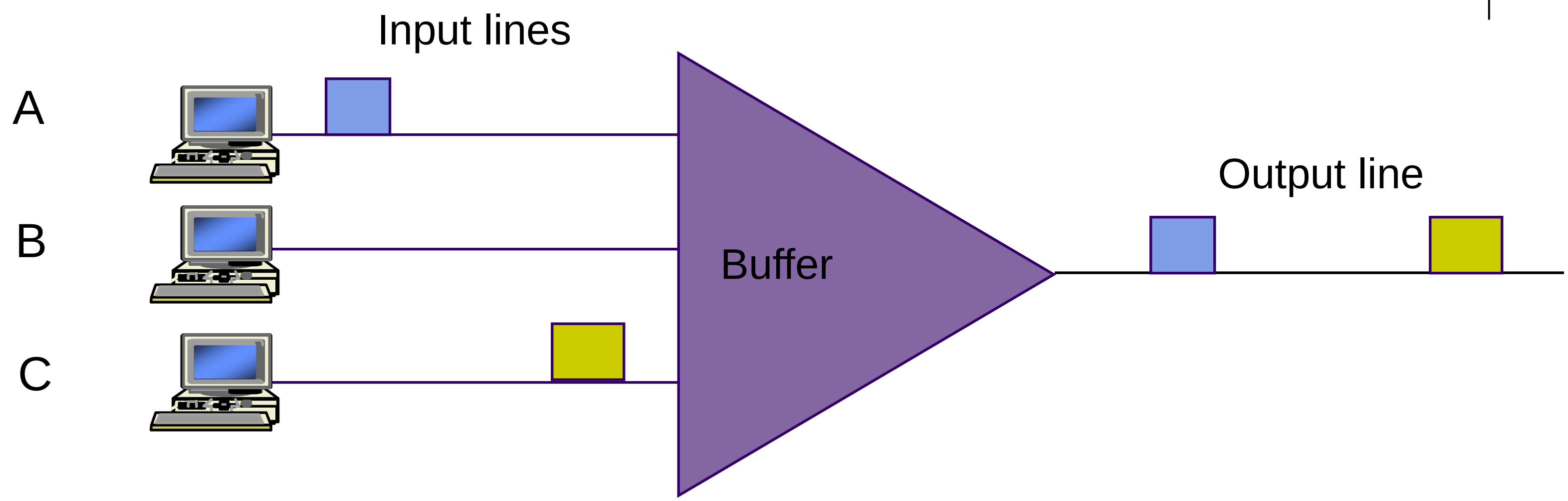
- Dedicated lines involve not waiting for other users, but lines are used inefficiently when user traffic is bursty
- Shared lines concentrate packets into shared line; packets buffered (delayed) when line is not immediately available

Multiplexers inherent in Packet Switches



- Packets/frames forwarded to buffer prior to transmission from switch
- Multiplexing occurs in these buffers

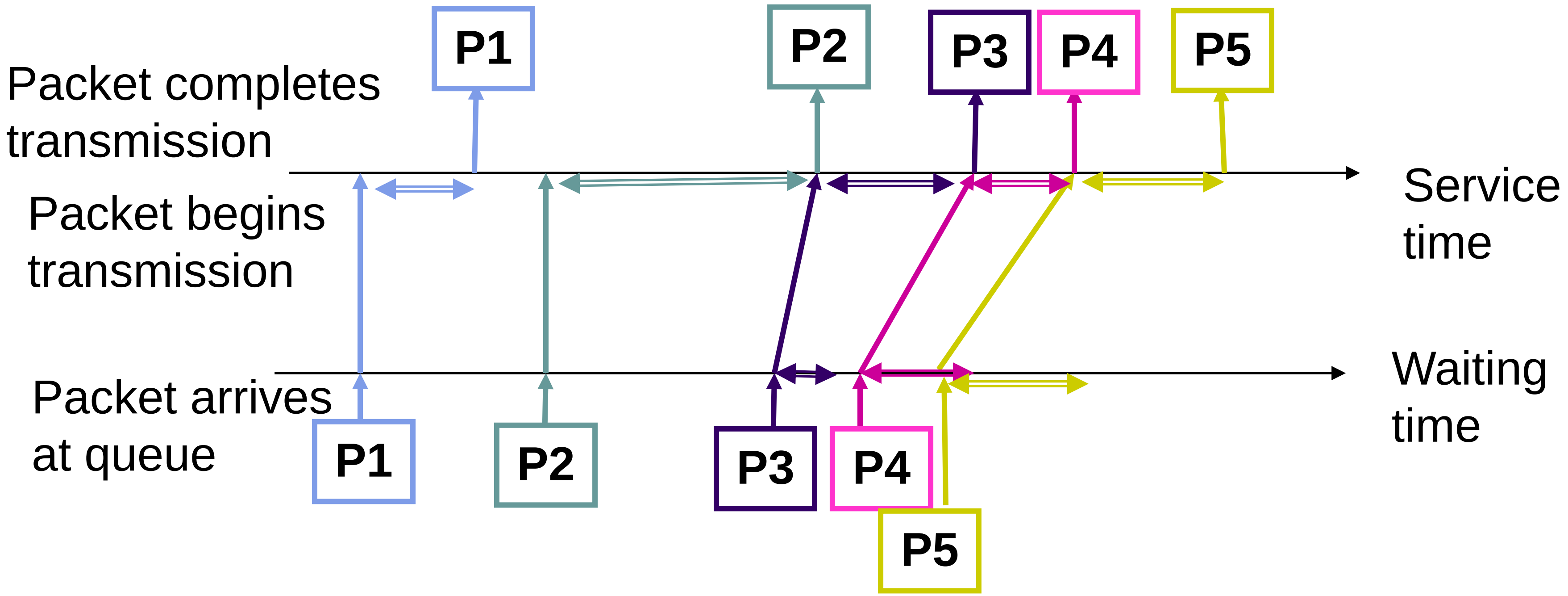
Multiplexer Modeling



- Arrivals: What is the packet interarrival pattern?
- Service Time: How long are the packets?
- Service Discipline: What is order of transmission?
- Buffer Discipline: If buffer is full, which packet is dropped?

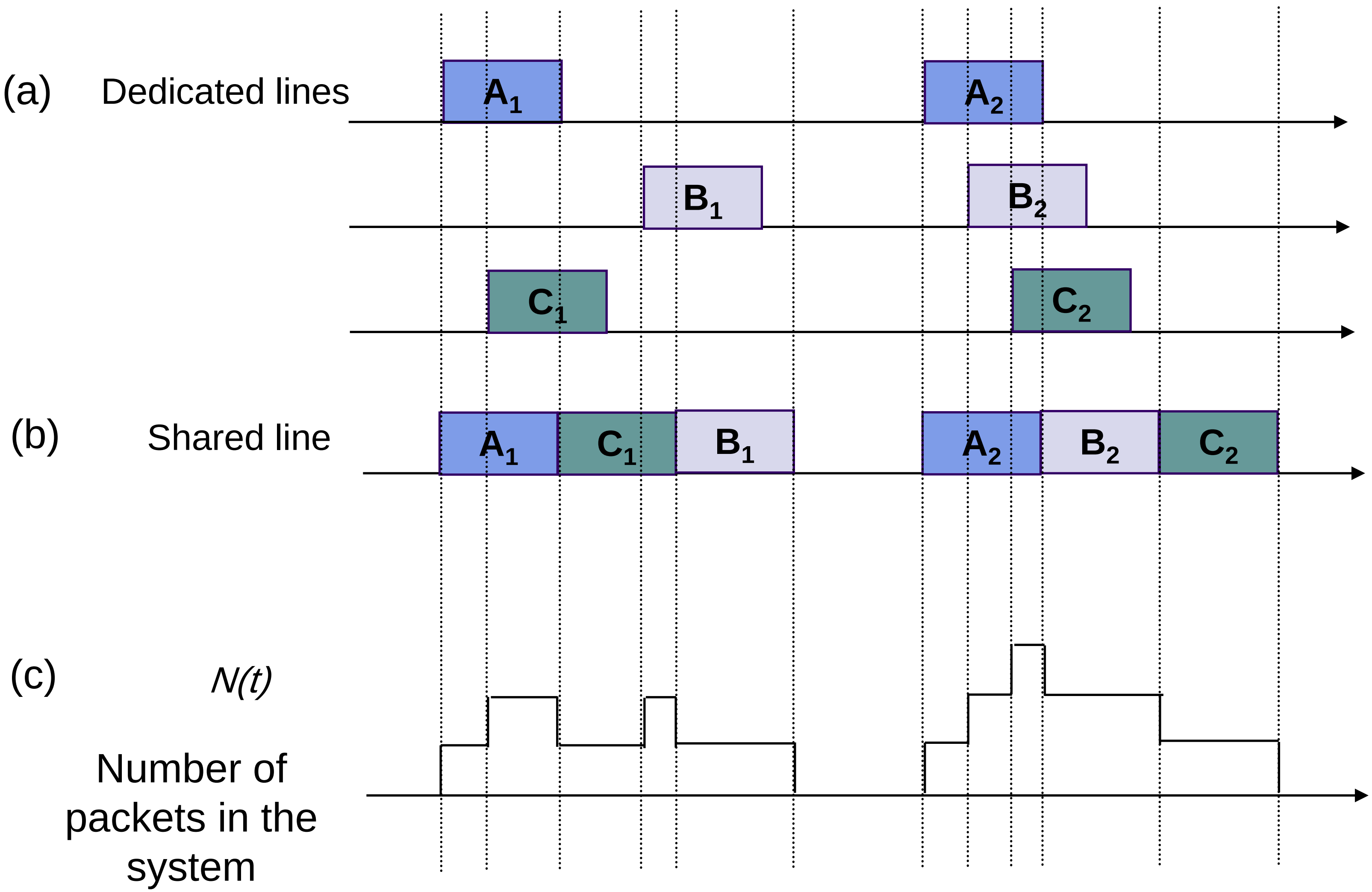
- Performance Measures:
- *Delay Distribution; Packet Loss Probability; Line Utilization*

Delay = Waiting + Service Times



- Packets arrive and wait for service
- Waiting Time: from arrival instant to beginning of service
- Service Time: time to transmit packet
- Delay: total time in system = waiting time + service time

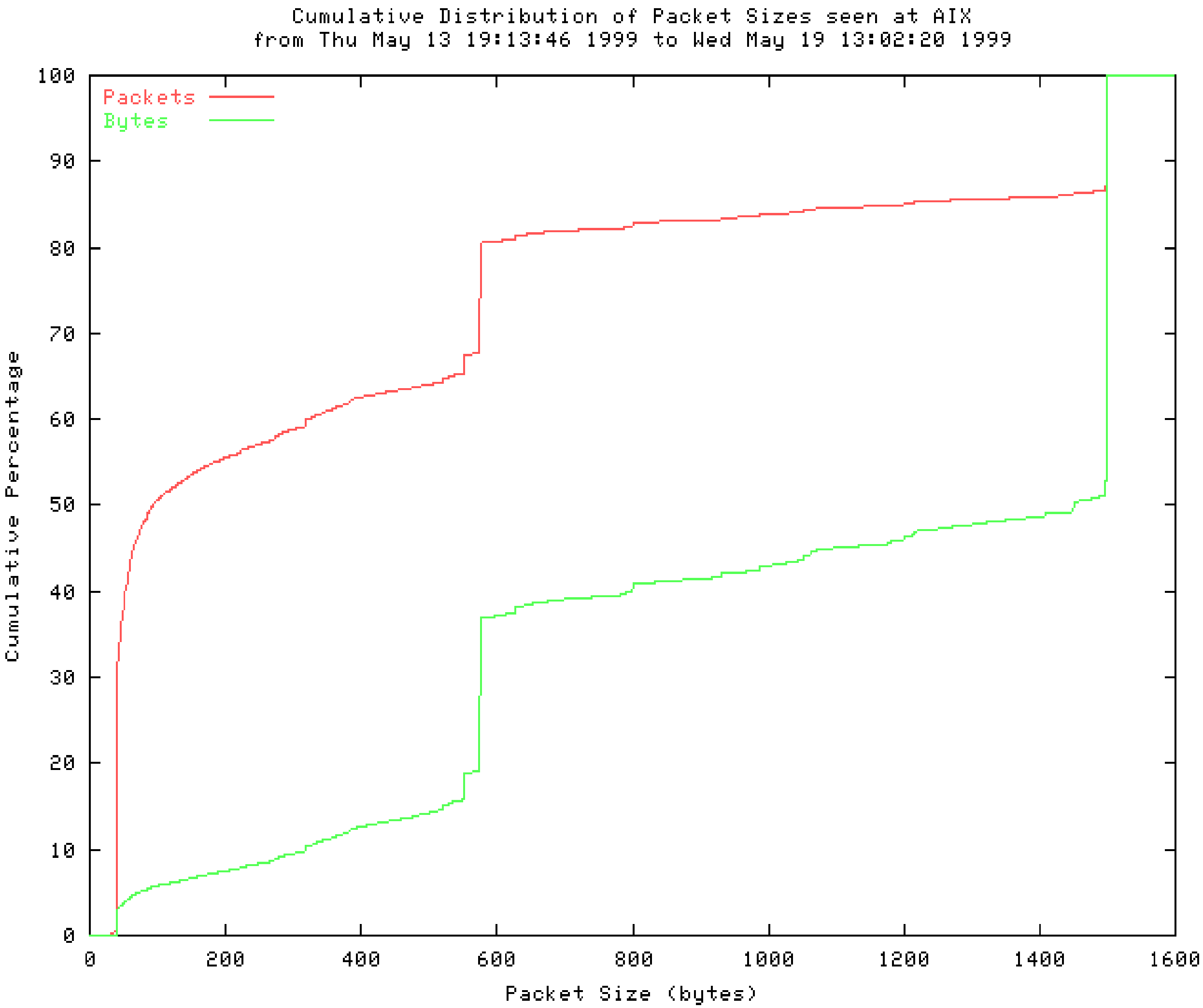
Fluctuations in Packets in the System



Packet Lengths & Service Times

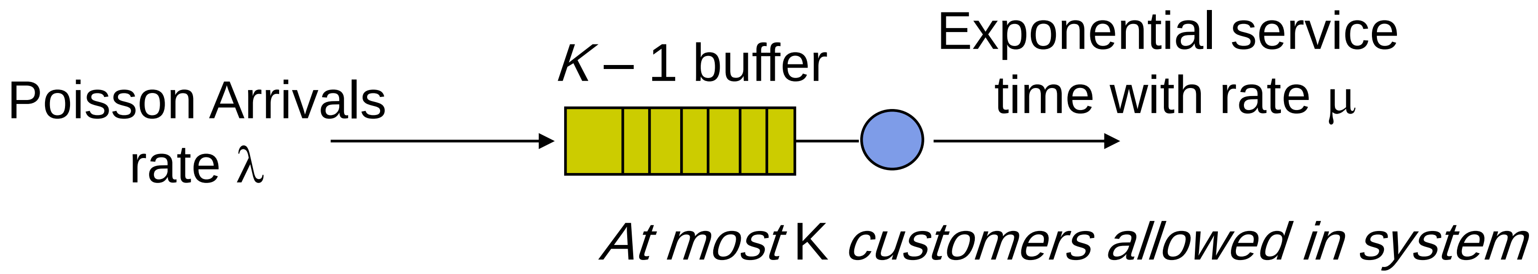
- R bits per second transmission rate
- L = # bits in a packet
- $X = L/R$ = time to transmit ("service") a packet
- Packet lengths are usually variable
  - Distribution of lengths → Dist. of service times
  - Common models:
    - Constant packet length (all the same)
    - Exponential distribution
    - Internet Measured Distributions fairly constant
      - See next chart

Measure Internet Packet Distribution



- Dominated by TCP traffic (85%)
- ~40% packets are minimum-sized 40 byte packets for TCP ACKs
- ~15% packets are maximum-sized Ethernet 1500 frames
- ~15% packets are 552 & 576 byte packets for TCP implementations that do not use path MTU discovery
- Mean=413 bytes
- Stand Dev=509 bytes
- Source: caida.org

M/M/1/K Queueing Model



- 1 customer served at a time; up to $K-1$ can wait in queue
- Mean service time E[X] = $1/\mu$
- Key parameter Load: $\rho = \lambda/\mu$
- When $\lambda \ll \mu$ ($\rho \approx 0$), customers arrive infrequently and usually find system empty, so delay is low and loss is unlikely
- As λ approaches μ ($\rho \to 1$), customers start bunching up and delays increase and losses occur more frequently
- When $\lambda > \mu$ ($\rho > 0$), customers arrive faster than they can be processed, so most customers find system full and those that do enter have to wait about $K-1$ service times

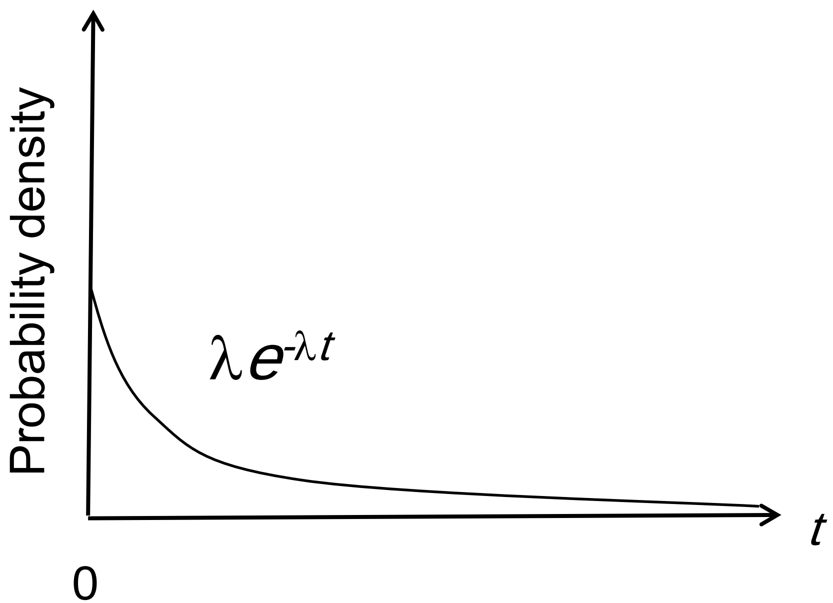
Poisson Arrivals

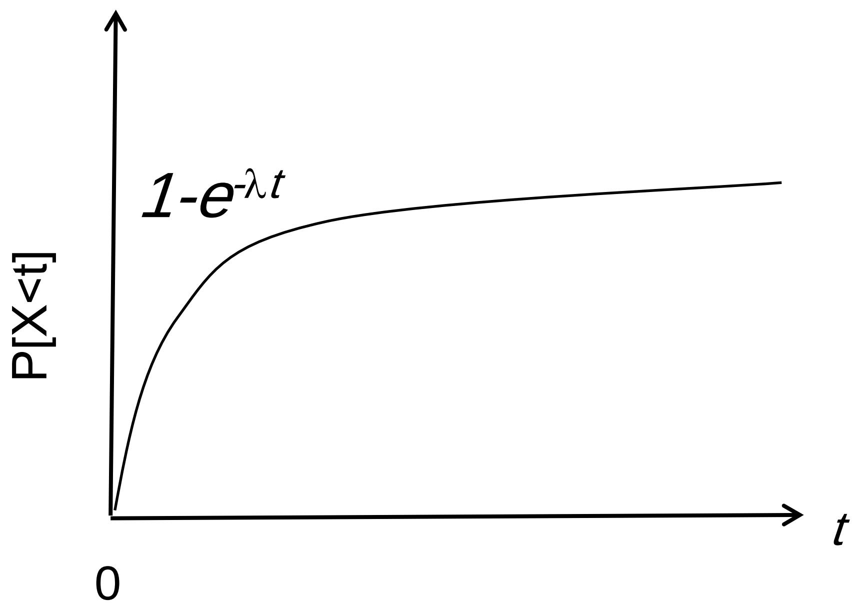
- Average Arrival Rate: λ packets per second
- Arrivals are equally-likely to occur at any point in time
- Time between consecutive arrivals is an exponential random variable with mean 1/ λ
- Number of arrivals in interval of time t is a Poisson random variable with mean λt

$$P\left[\text{ k arrivals in t seconds}\right] = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$$

Exponential Distribution

$$P[X > t] = e^{-t/E[X]} = e^{-\lambda t} \text{ for } t > 0.$$





M/M/1/K Performance Results

(From Appendix A)

Probability of Overflow:

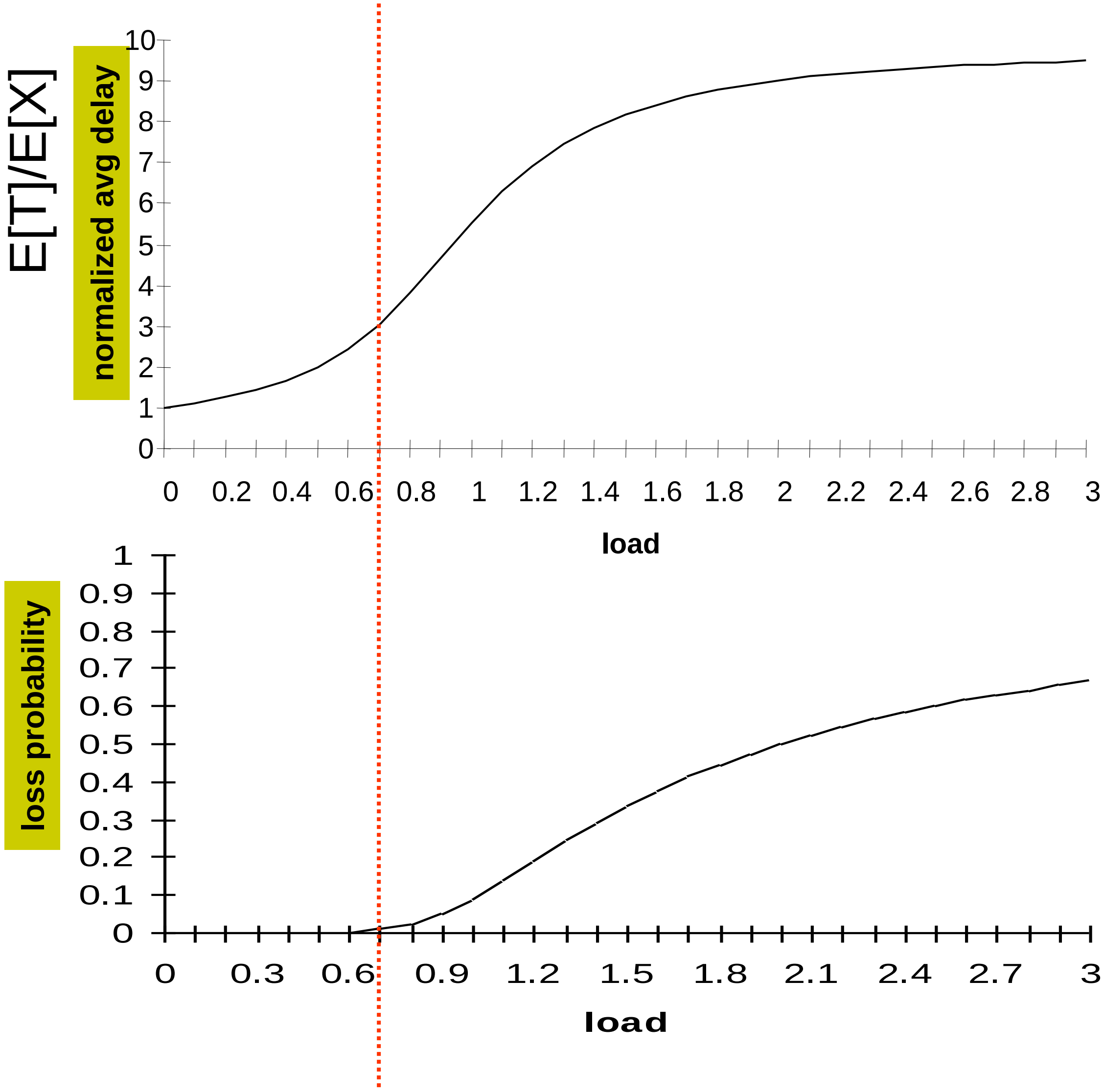
$$P_{loss} = \frac{(1-\rho)\rho^{K}}{1-\rho^{K+1}}$$

Average Total Packet Delay:

$$E[N] = \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}}$$

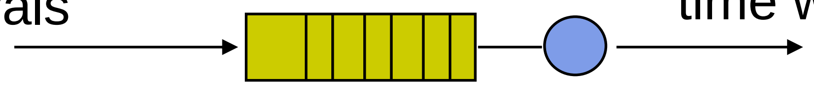
$$E[T] = \frac{E[N]}{\lambda(1-P_K)}$$

M/M/1/10



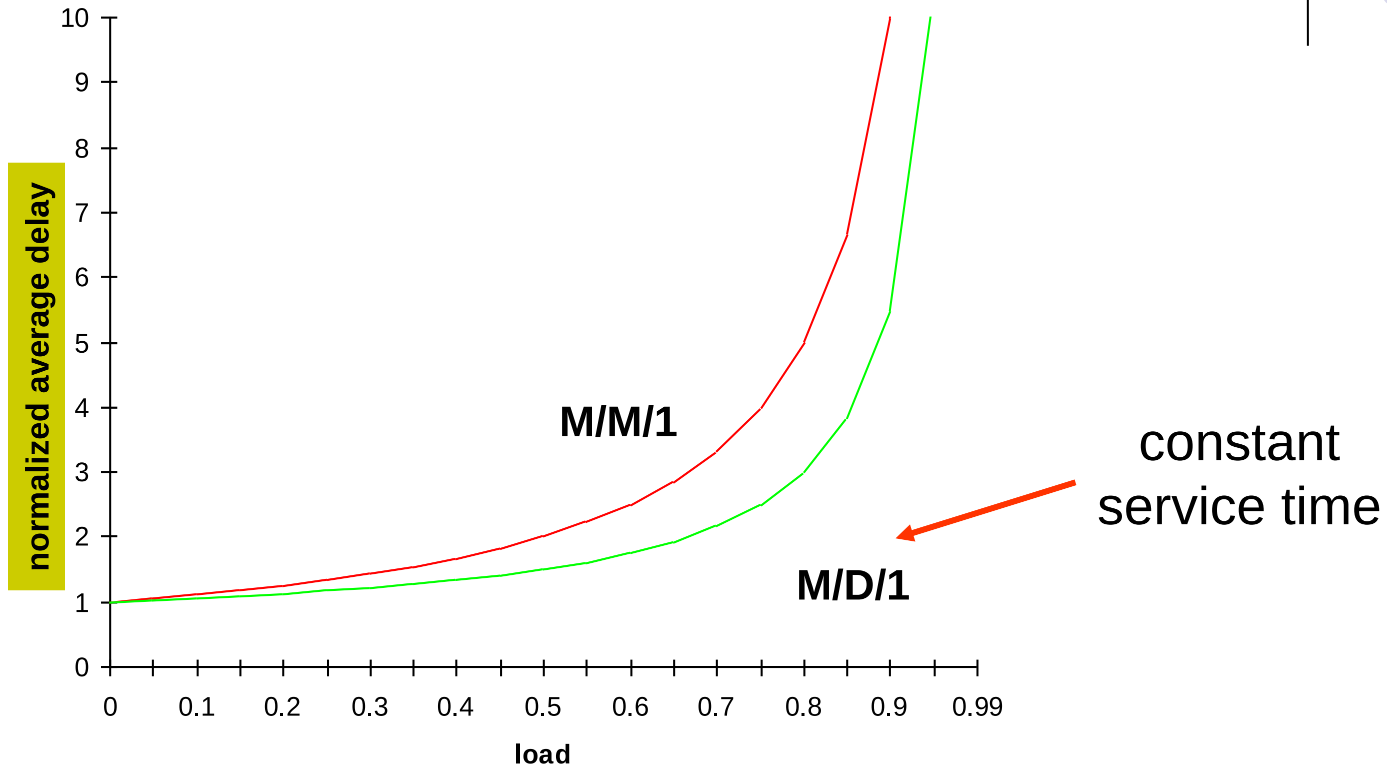
- Maximum 10 packets allowed in system
- Minimum delay is 1 service time
- Maximum delay is 10 service times
- At 70% load delay & loss begin increasing
- What if we add more buffers?

M/M/1 Queue



- P_b=0 since customers are never blocked
- Average Time in system E[T] = E[W] + E[X]
- When $\lambda << \mu$, customers arrive infrequently and delays are low
- As λ approaches μ □□customers start bunching up and average delays increase
- When $\lambda > \mu$□□ customers arrive faster than they can be processed and queue grows without bound (unstable)

Avg. Delay in M/M/1 & M/D/1



$$E[T_M] = \frac{1}{\lambda}\left[\frac{\rho}{1-\rho}\right] = \left[\frac{1}{1-\rho}\right]\frac{1}{\mu} = \left[\frac{\rho}{1-\rho}\right]\frac{1}{\mu} + \frac{1}{\mu} \quad \text{for M/M/1 model.}$$

$$E[T_D] = \left[1 + \frac{\rho}{2(1-\rho)}\right]\frac{1}{\mu} = \left[\frac{\rho}{2(1-\rho)}\right]\frac{1}{\mu} + \frac{1}{\mu} \quad \text{for M/D/1 system.}$$

Effect of Scale

- C = 100,000 bps
- Exp. Dist. with Avg. Packet Length: 10,000 bits
- Service Time: X=0.1 second
- Arrival Rate: 7.5 pkts/sec
- Load: ρ=0.75
- Mean Delay:

$E[T]$ = 0.1/(1-.75) = 0.4 sec

- C = 10,000,000 bps
- Exp. Dist. with Avg. Packet Length: 10,000 bits
- Service Time: X=0.001 second
- Arrival Rate: 750 pkts/sec
- Load: ρ=0.75
- Mean Delay:
- $E[T]$ = 0.001/(1-.75) = 0.004 sec

Reduction by factor of 100

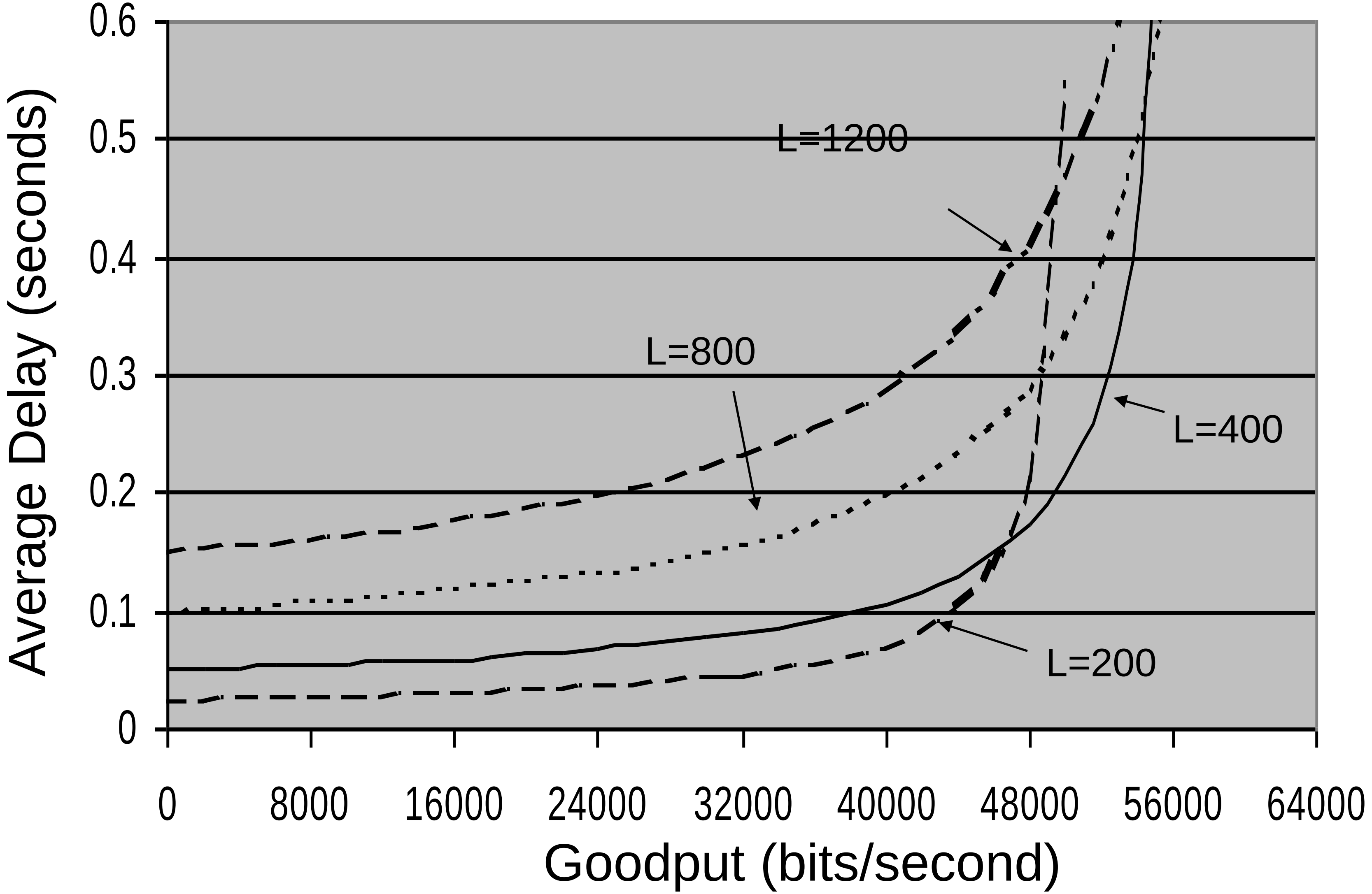
Aggregation of flows can improve Delay & Loss Performance

Example: Header overhead & Goodput

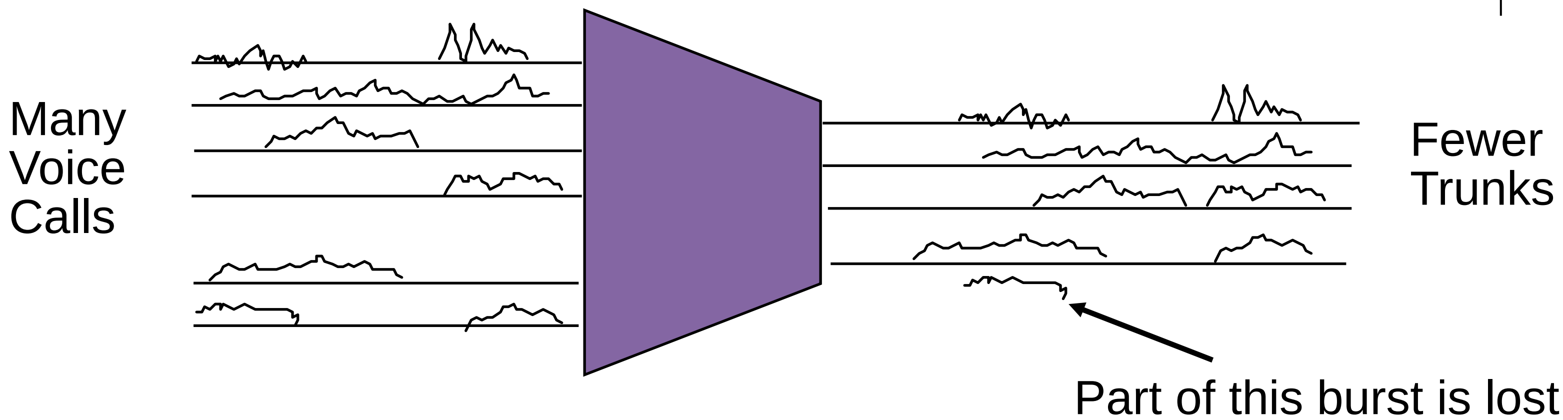
- Let R=64 kbps
- Assume IP+TCP header = 40 bytes
- Assume constant packets of total length
  - L= 200, 400, 800, 1200 bytes
- Find avg. delay vs. goodput (information transmitted excluding header overhead)

- Service rate μ = 64000/8L packets/second
- Total load $\rho = \lambda$ 64000/8L
- Goodput = λ packets/sec x 8(L-40) bits/packet
- Max Goodput = (1-40/L)64000 bps

Header overhead limits maximum goodput

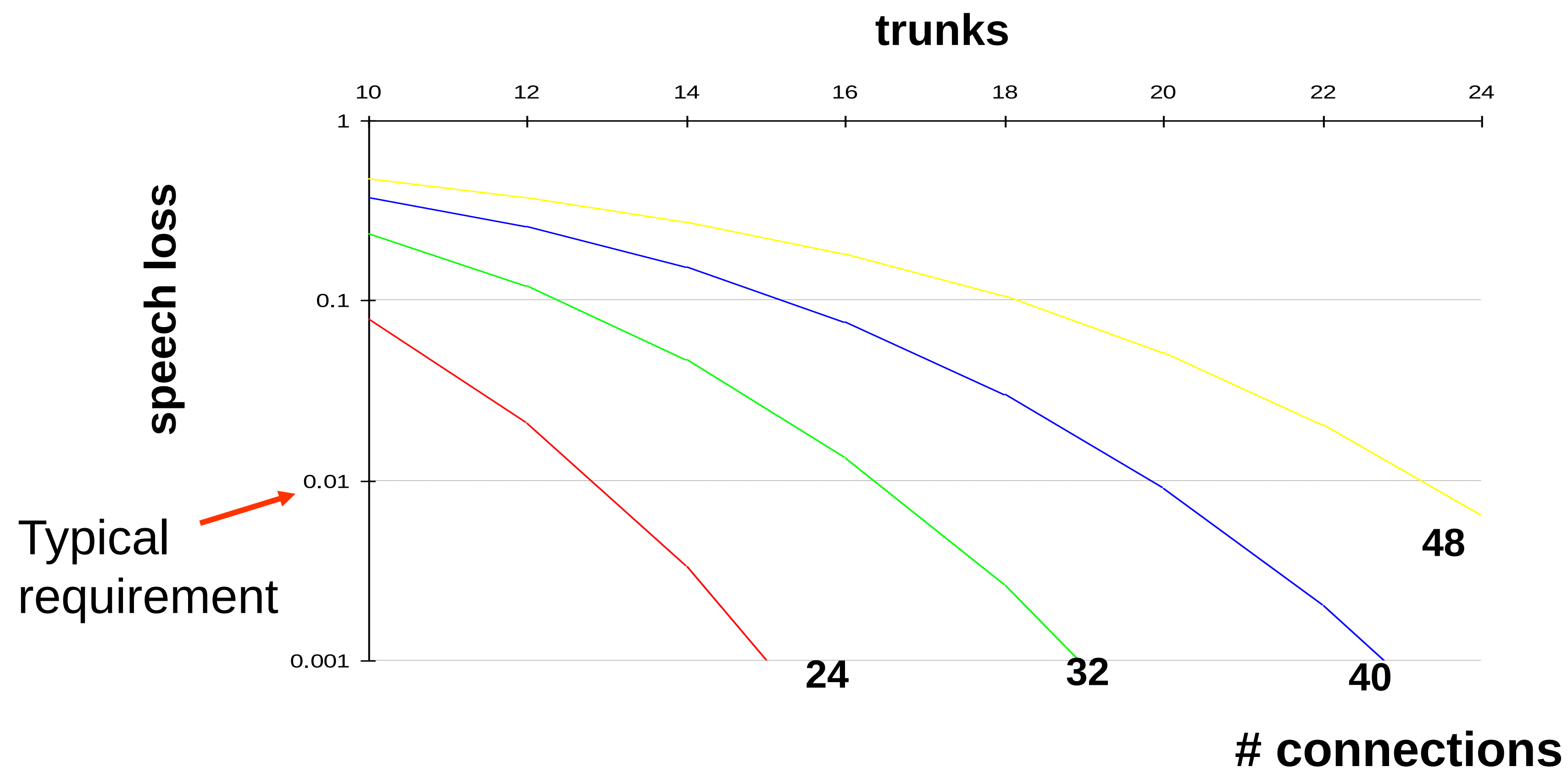


Burst Multiplexing / Speech Interpolation



- Voice active < 40% time
- ***No buffering***, on-the-fly switch bursts to available trunks
- Can handle 2 to 3 times as many calls
- Tradeoff: Trunk Utilization vs. Speech Loss
  - Fractional Speech Loss: fraction of active speech lost
- Demand Characteristics
  - Talkspurt and Silence Duration Statistics
  - Proportion of time speaker active/idle

Speech Loss vs. Trunks



$$\text{speech loss} = \frac{\sum_{k=m+1}^{n} (k-m)\binom{n}{k} p^k (1-p)^{n-k}}{np} \quad \text{where} \quad \binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

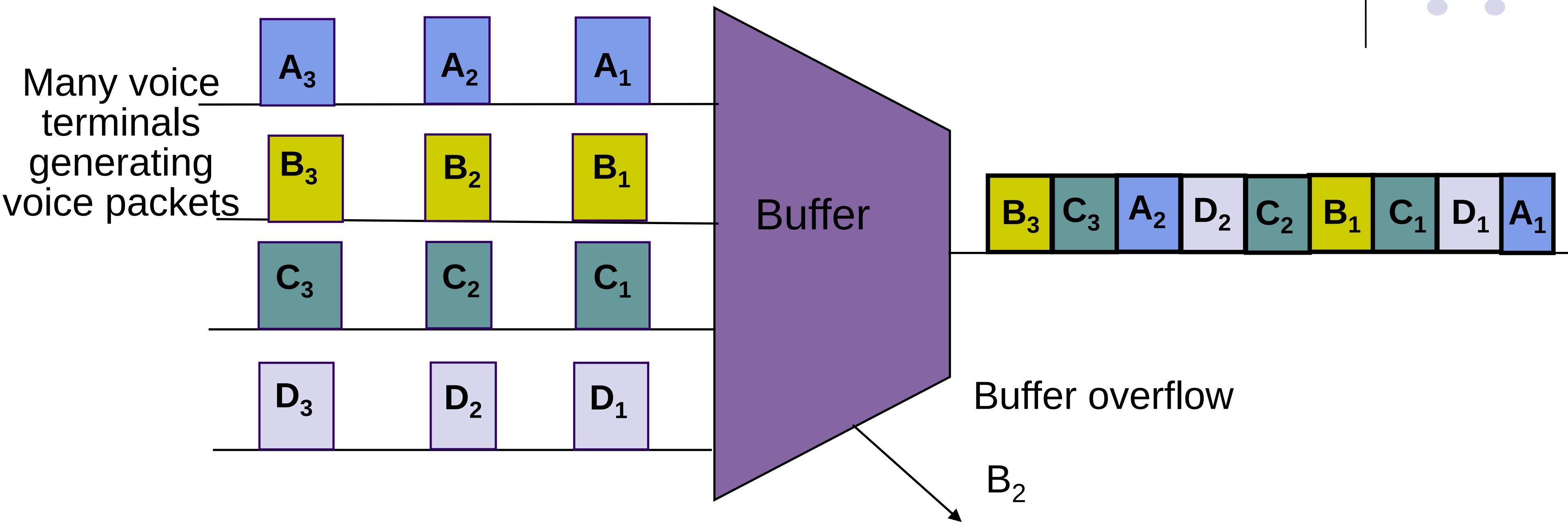
Effect of Scale

- Larger flows lead to better performance
- Multiplexing Gain = # speakers / # trunks

Trunks required for 1% speech loss

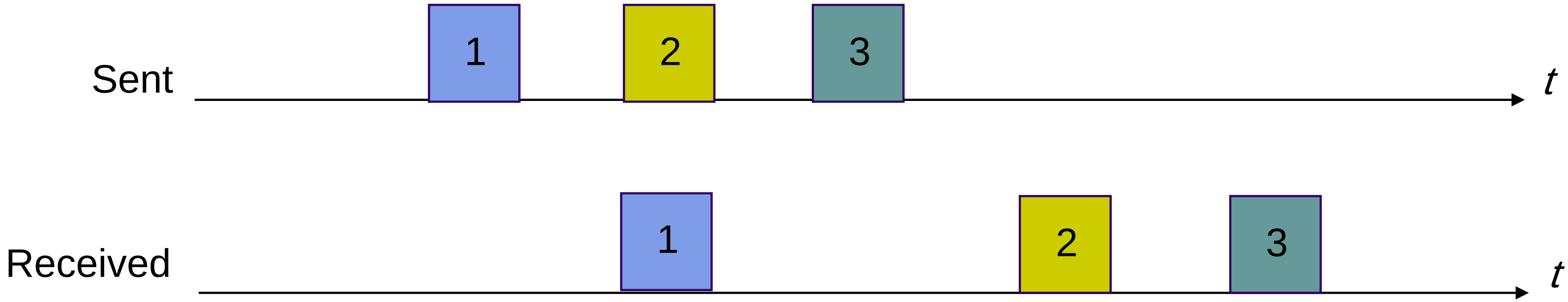
Speakers	Trunks	Multiplexing Gain	Utilization
24	13	1.85	0.74
32	16	2.00	0.80
40	20	2.00	0.80
48	23	2.09	0.83

Packet Speech Multiplexing



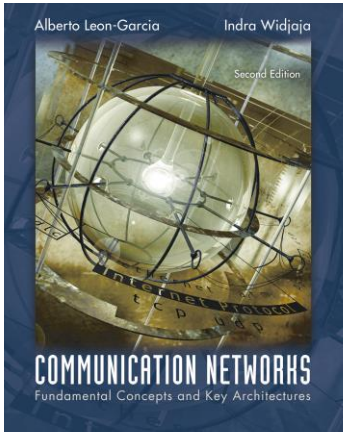
- Digital speech carried by fixed-length packets
- No packets when speaker silent
- Synchronous packets when speaker active
- Buffer packets & transmit over shared high-speed line
- Tradeoffs: Utilization vs. Delay/Jitter & Loss

Packet Switching of Voice



- Packetization delay: time for speech samples to fill a packet
- Jitter: variable inter-packet arrivals at destination
- Playback strategies required to compensate for jitter/loss
  - Flexible delay inserted to produce fixed end-to-end delay
  - Need buffer overflow/underflow countermeasures
  - Need clock recovery algorithm

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



ARQ Efficiency Calculations

Stop & Wait Performance

1 successful transmission

i – 1 unsuccessful transmissions

$$E[t_{total}] = t_0 + \sum_{i=1}^{\infty}(i-1)t_{out}\,P[n_t = i]$$

$$= t_0 + \sum_{i=1}^{\infty}(i-1)t_{out}(1-P_f)^{i-1}P_f$$

$$= t_0 + \frac{t_{out}P_f}{1-P_f} = t_0\frac{1}{1-P_f}.$$

Efficiency:

$$\eta_{SW} = \frac{\dfrac{n_f - n_o}{E[t_{total}]}}{R} = (1-P_f)\frac{1-\dfrac{n_o}{n_f}}{1+\dfrac{n_a}{n_f}+\dfrac{2(t_{prop}+t_{proc})R}{n_f}} = (1-P_f)\eta_0.$$

Go-Back-N Performance

1 successful transmission $i-1$ unsuccessful transmissions

$$E[t_{total}] = t_f + \sum_{i=1}^{\infty}(i-1)W_s t_f P[n_t = i]$$

$$= t_f + W_s t_f \sum_{i=1}^{\infty}(i-1)(1-P_f)^{i-1}P_f$$

$$= t_f + \frac{W_s t_f P_f}{1-P_f} = t_f \frac{1+(W_s-1)P_f}{1-P_f}.$$

Efficiency:

$$\eta_{GBN} = \frac{\frac{n_f - n_o}{E[t_{total}]}}{R} = (1-P_f)\frac{1-\frac{n_o}{n_f}}{1+(W_s-1)P_f}.$$