

Computer Communication Networks

driver is in the kernel space

DLC幾乎全軟件

Chapter 3: Data Link Layer

impt

framing
error control(ARQ)

Prof. Xudong Wang

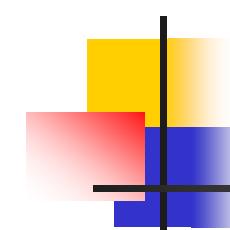
Wireless Networking and Artificial Intelligence Lab

UM-SJTU Joint Institute

Shanghai Jiao Tong University

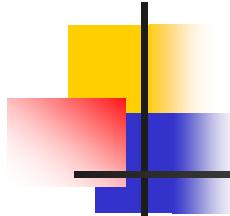
Shanghai, China

<http://wanglab.sjtu.edu.cn>



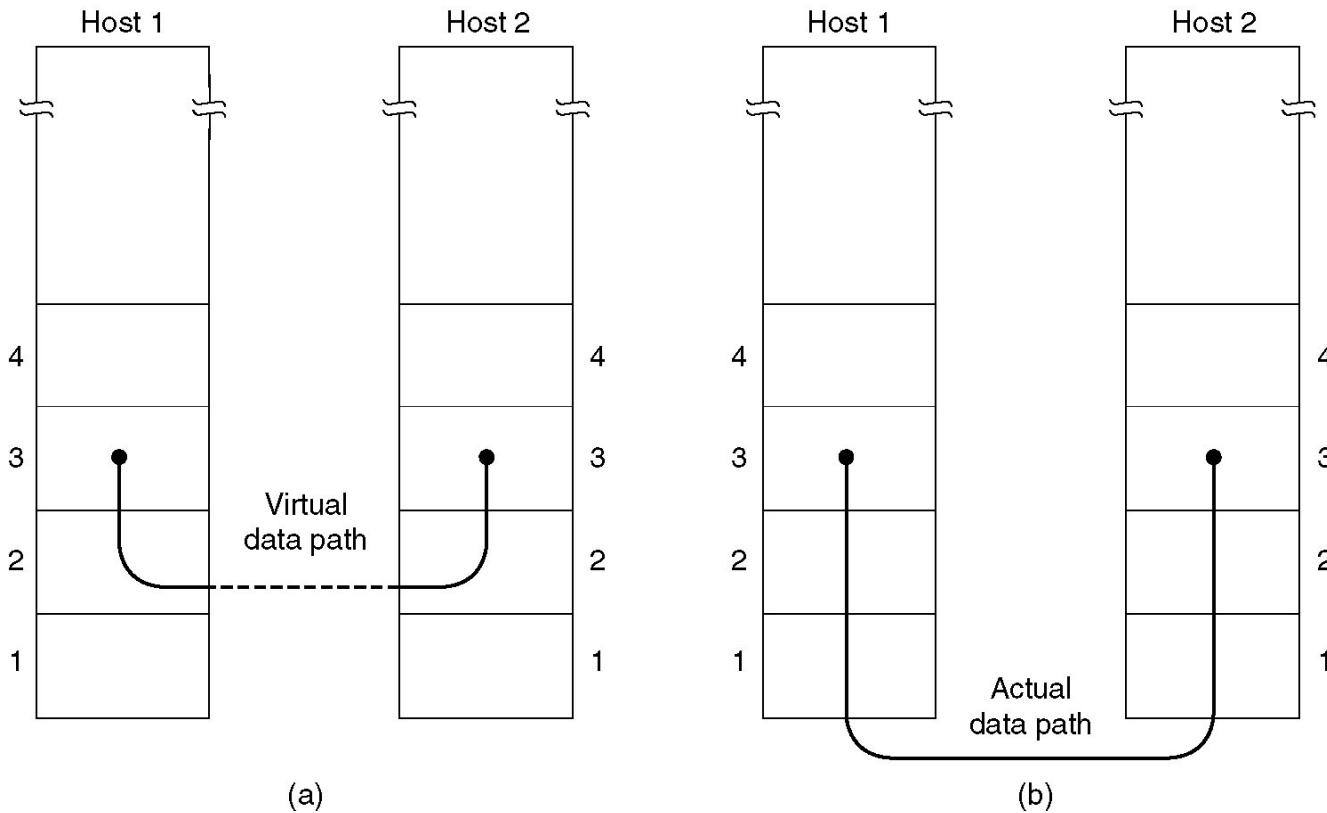
Outline

- Protocol Stack
 - Relation to network and physical layers
 - Sub-layers
 - Logic link control layer
 - medium access control layer
- Logical Link Control Functions
 - Framing
 - Error control
 - Flow control
 - Multiplexing
 - Link Maintenance
 - Security: Authentication & Encryption
 - Link layer switching and bridging
- Medium Access Control Function
 - *Chapter 4*



Protocol Stack of the Data Link Layer

Protocol Stack of the Data Link Layer



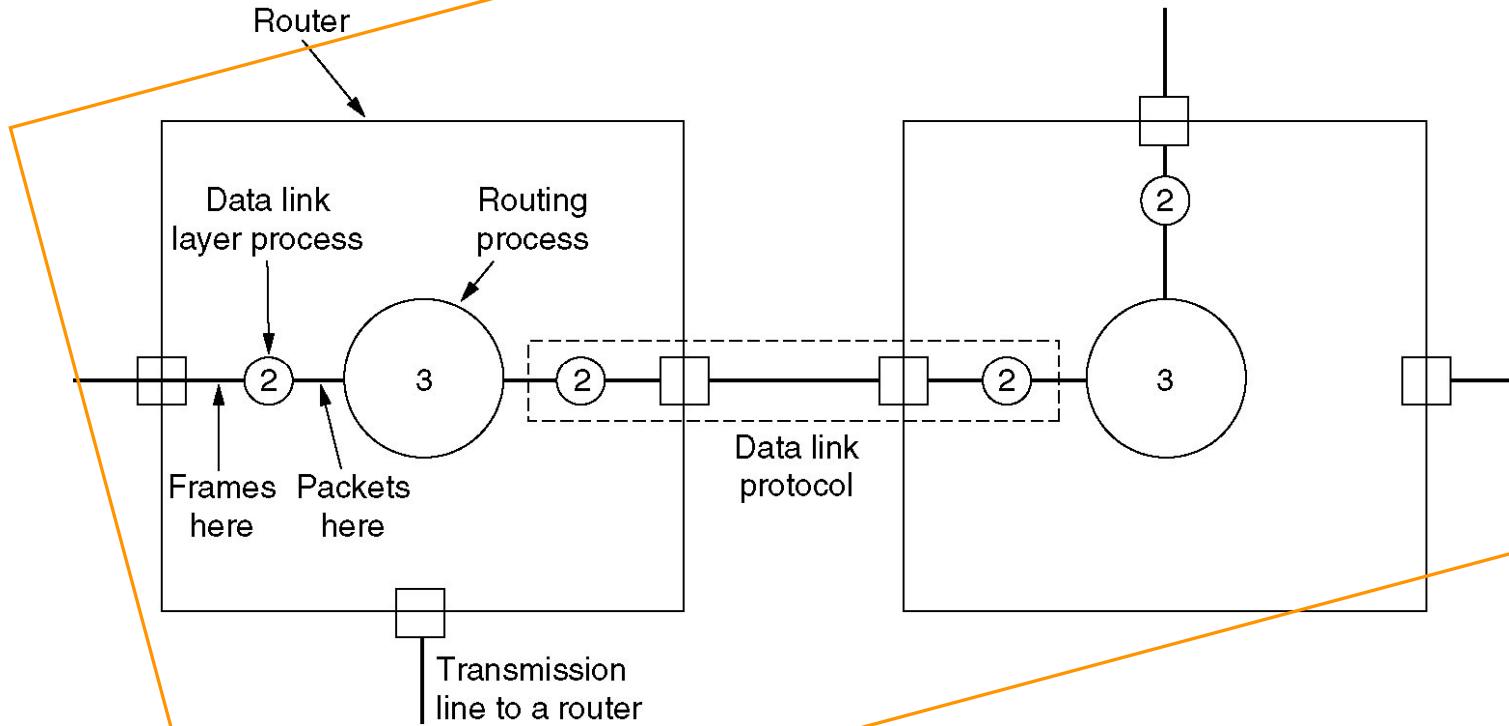
- (a) **Virtual communication**
- (b) Actual communication



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

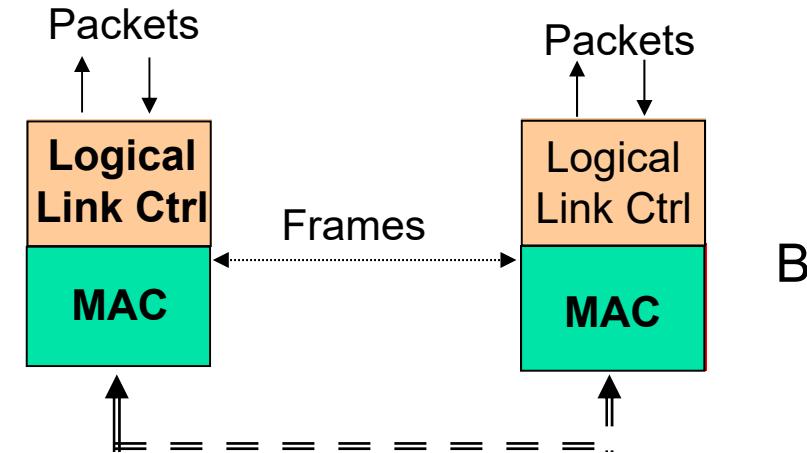
- A service feature can be provided by implementing a protocol
 - end-to-end across the network
 - across every hop in the network
- Example:
 - Perform error control at every hop in the network or only between the source and destination?
 - Perform flow control between every hop in the network or only between source & destination?
- *Usually data link layer is a hop-by-hop approach*

Placement of the Data Link Protocol



Sub-Layers and Functions of the Data Link Layer

- Sub-Layers
 - Logical Link Control
 - Medium Access Control
- Applications
 - Direct Links or point to point
 - No MAC
 - Broadcast: LANs, wireless
- Features
 - Losses & errors, but no out-of-sequence frames
 - Service types
 - unacknowledged connectionless
 - acknowledged connectionless
 - unacknowledged connection

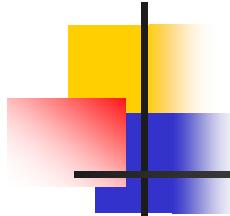




Functions of the Data Link Upper Layer

- Provide Services to the Network Layer

- **Framing**
- **Error control** green ones are impt
- **Flow control** physical layer only cares about bits
- Multiplexing
- Link maintenance
- Timing recovery
- Security: Authentication & Encryption

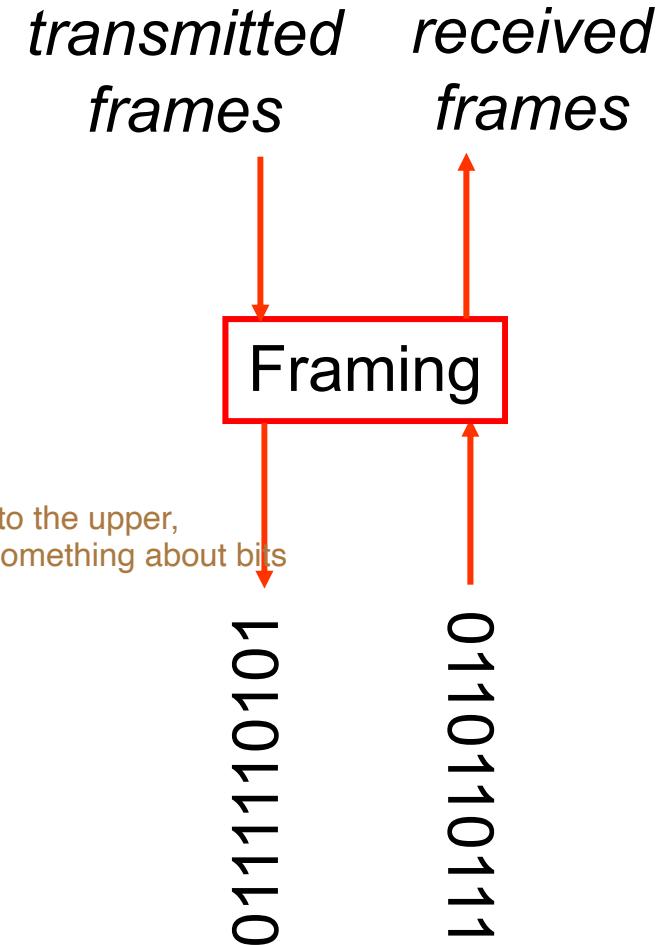


Framing in the Data Link Layer

3kinds
character oriented
byte oriented
bit oriented

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
 - Other fields



Character-Oriented Framing

Only the STX after the DLE is considered the start point
only the ETX after DLE is considered the end point
eg. 如果你的data有兩個DLE
你就需要發送4個DLE after framing

- 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 STX (DLE ETX) in frame
 - All DLEs occur in pairs except at frame boundaries

Data to be sent



bit streams in reality

we need to put some boundaries

STX單獨出現在中間，前面應該不用加DLE吧

After stuffing and framing



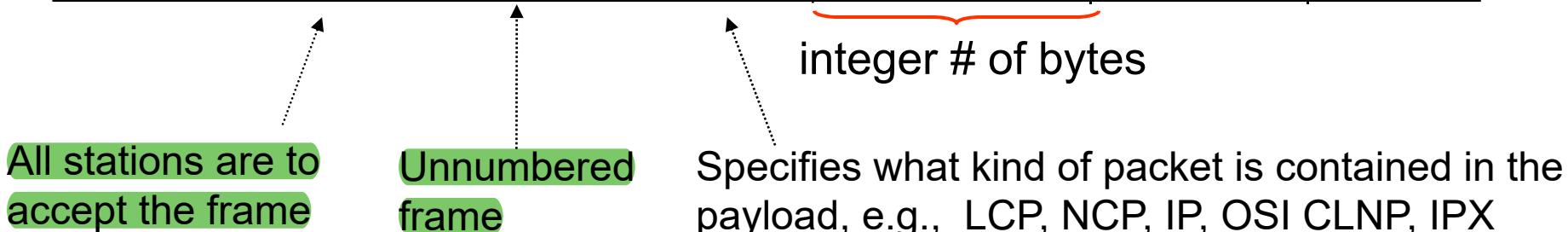
I know
this is
a frame

this is
original
text

Byte-Stuffing in PPP (1)

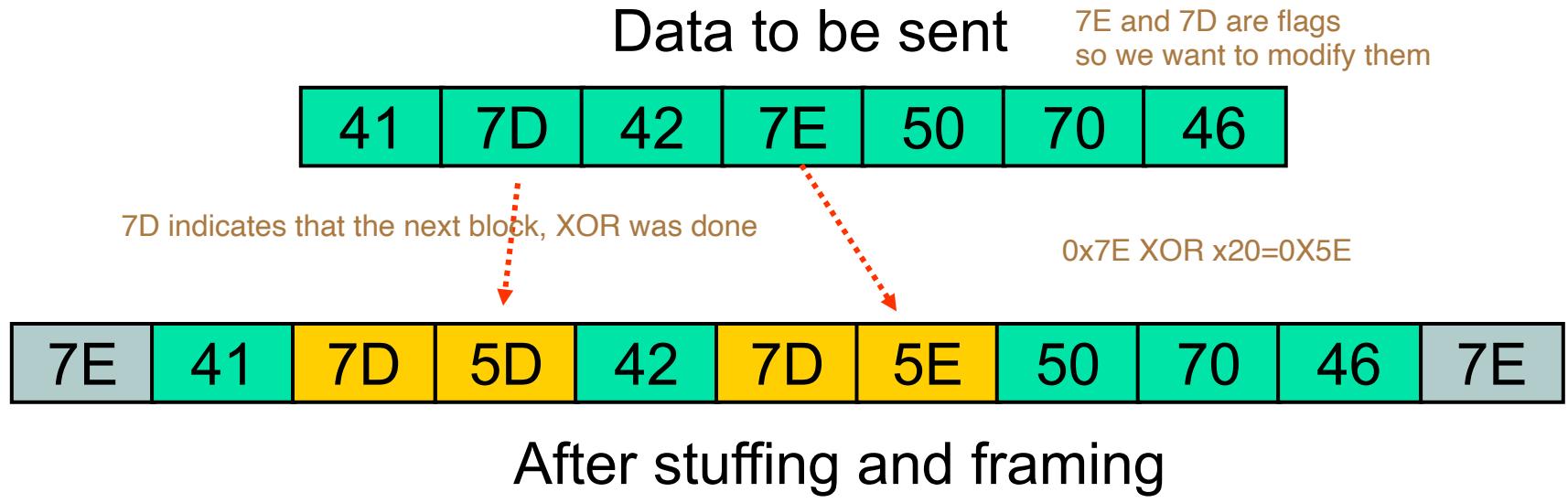
- PPP uses similar frame structure as HDLC, except
 - Protocol type field
 - Payload contains an *integer* number of bytes
- PPP uses the same flag, but uses *byte stuffing*
- Problems with PPP byte stuffing
 - Size of frame varies unpredictably due to byte insertion
 - Malicious users can inflate bandwidth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

Flag 01111110	Address 1111111	Control 00000011	Protocol	Information	CRC	Flag 01111110
------------------	--------------------	---------------------	----------	-------------	-----	------------------



Byte-Stuffing in PPP (2)

- 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)

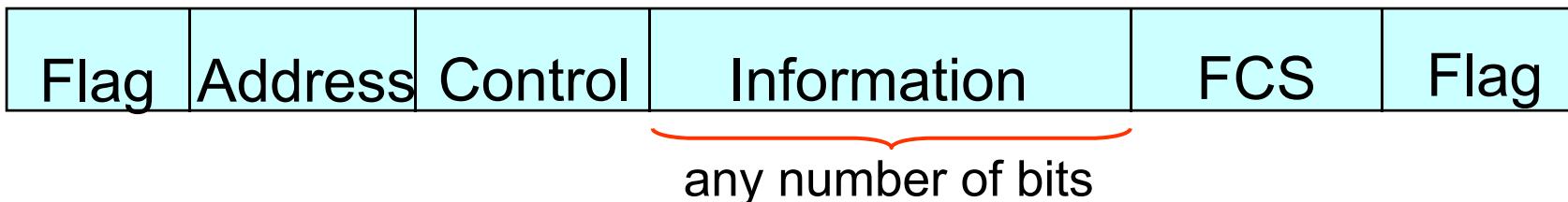


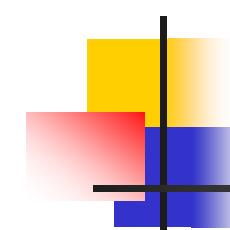
Framing & Bit Stuffing

not allowed to send 6 bits consecutively, cuz it's a flag
if met with 5 consecutive 1s, put a 0 next

- 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

HDLC frame





Example: Bit stuffing & de-stuffing

(a)

Data to be sent

01101111111100

After stuffing and framing

01111110 0110111101111000 01111110

(b)

Data received

0111111000011101111101111101100111110

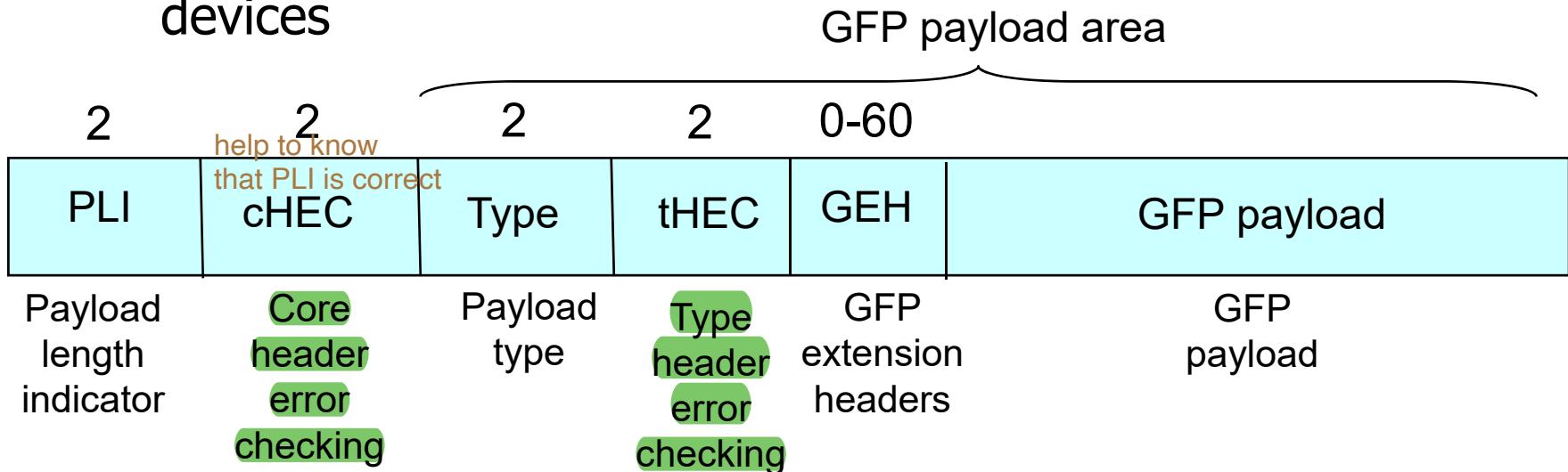
This is a flag,
it is 7E

After destuffing and deframing

000111011111-11111-110

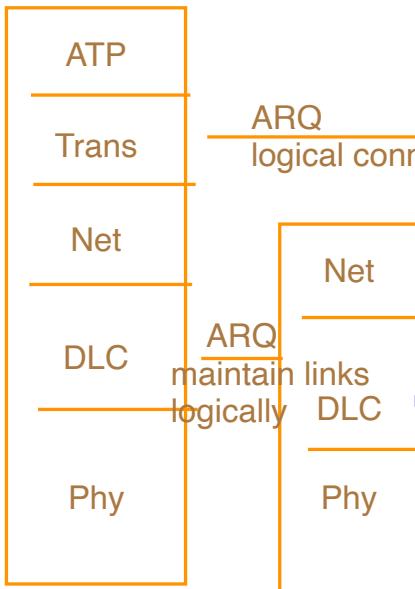
Generic Framing Procedure

- 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

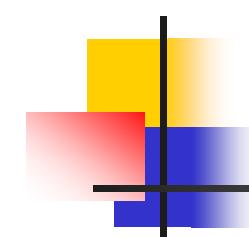
- Synchronization in three-states
 - *Hunt state*: examine 4-bytes to see if CRC ok
 - If no, move forward by one-byte
 - If yes, move to pre-sync state
 - *Pre-sync state*: tentative PLI indicates next frame
 - If N successful frame detections, move to sync state
 - If no match, go to hunt state
 - *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



Error Control in the Data Link Layer (ARQ) very important

One hop
cuz it
might be
linked to
a router??
or may be
a router??

ARQ also included in the transport layer

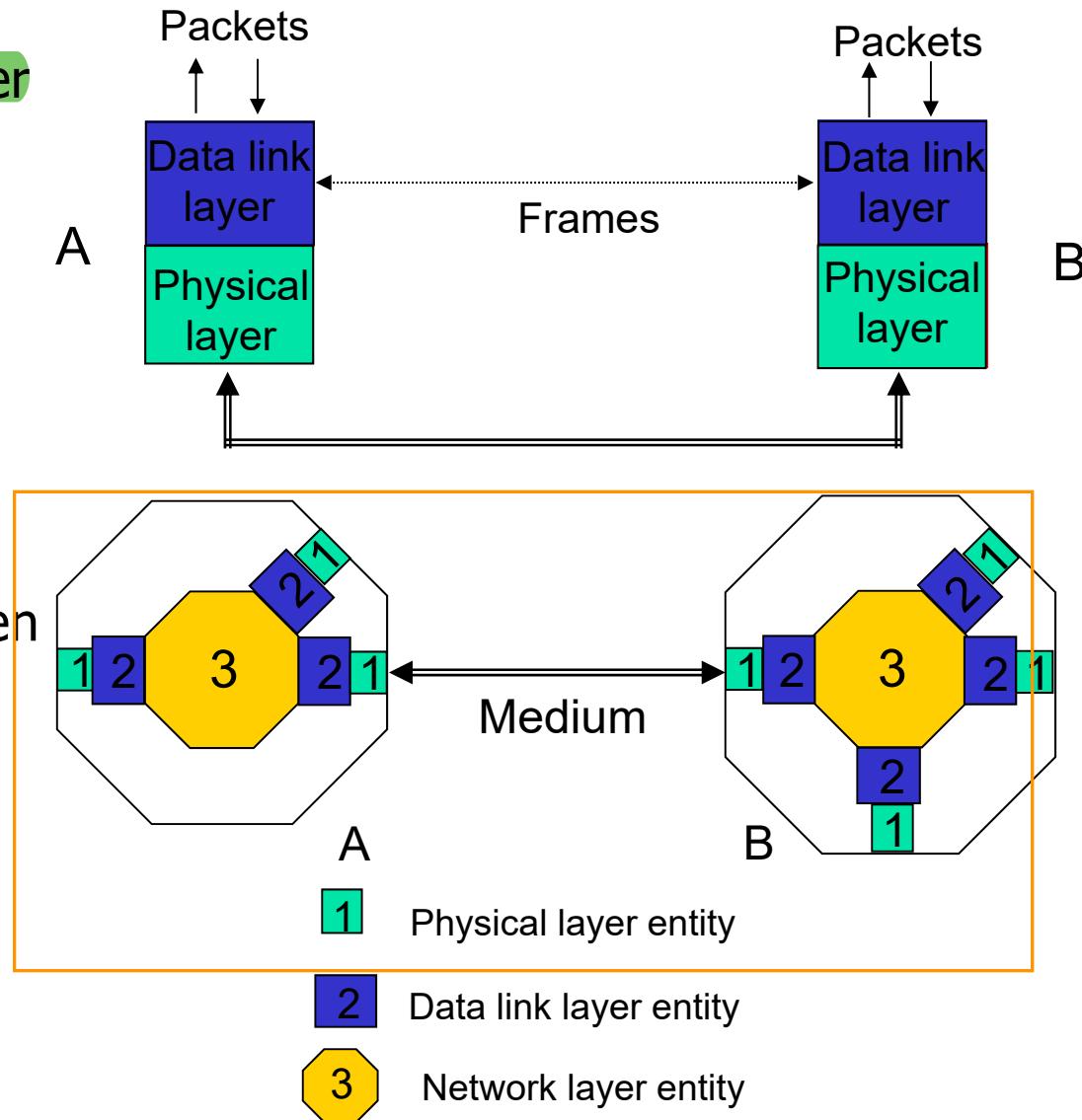


Error Control

- **FEC** Forward Error Correction usually used in the physical layer only
 - Data link FEC versus physical link FEC
- **ARQ** Auto Repeat Request
 - Retransmission after error detection
 - Simplex Stop-and-Wait protocol
 - Sliding window protocol
 - One-bit sliding window
 - Go-back-N ARQ
 - Selective repeat ARQ used in TCP

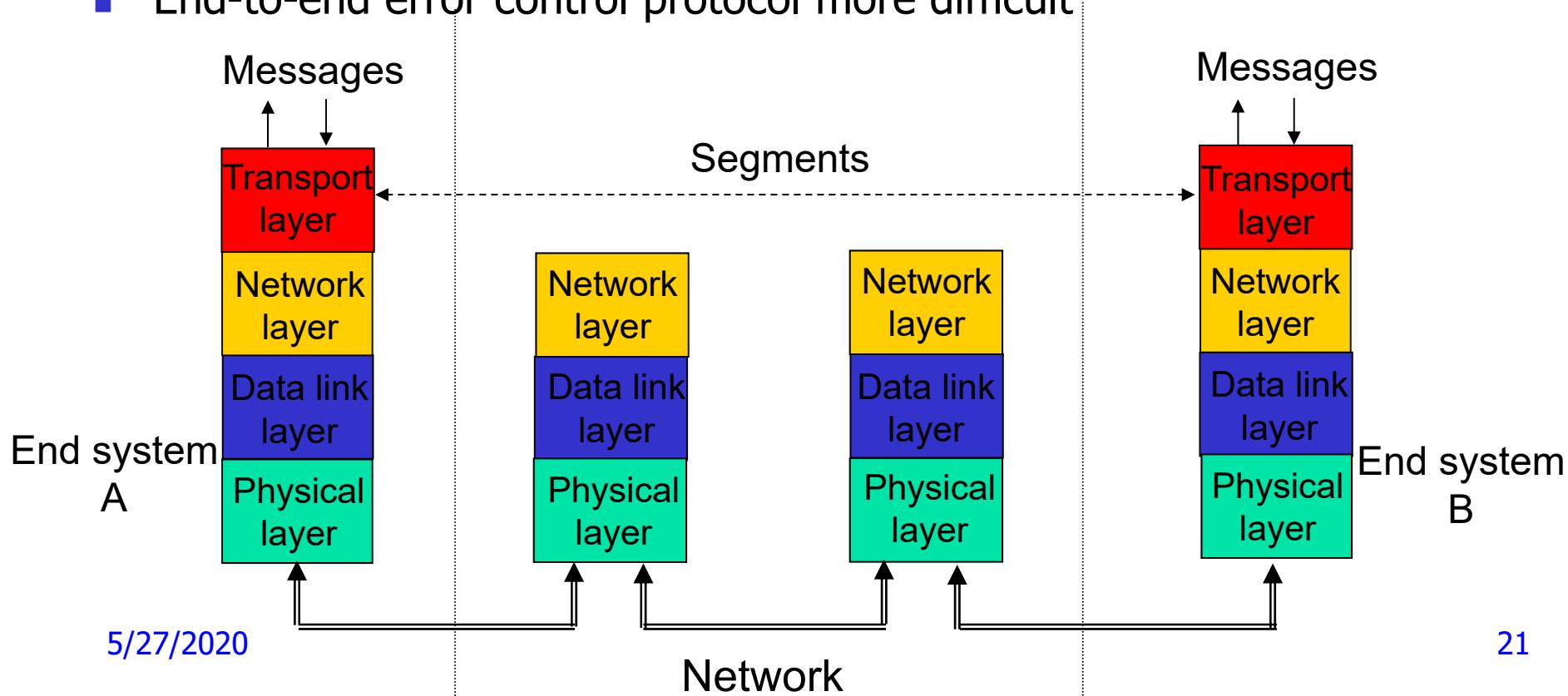
Error control in Data Link Layer

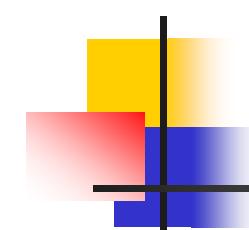
- Data Link operates over wire-like, directly-connected systems
- Frames can be corrupted or lost, but arrive in order
- Data link performs error-checking & retransmission
- Ensures error-free packet transfer between two systems



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





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 the receiver checks
1 no errors
2 in order (check sequence number)
3 no duplications
 - 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*

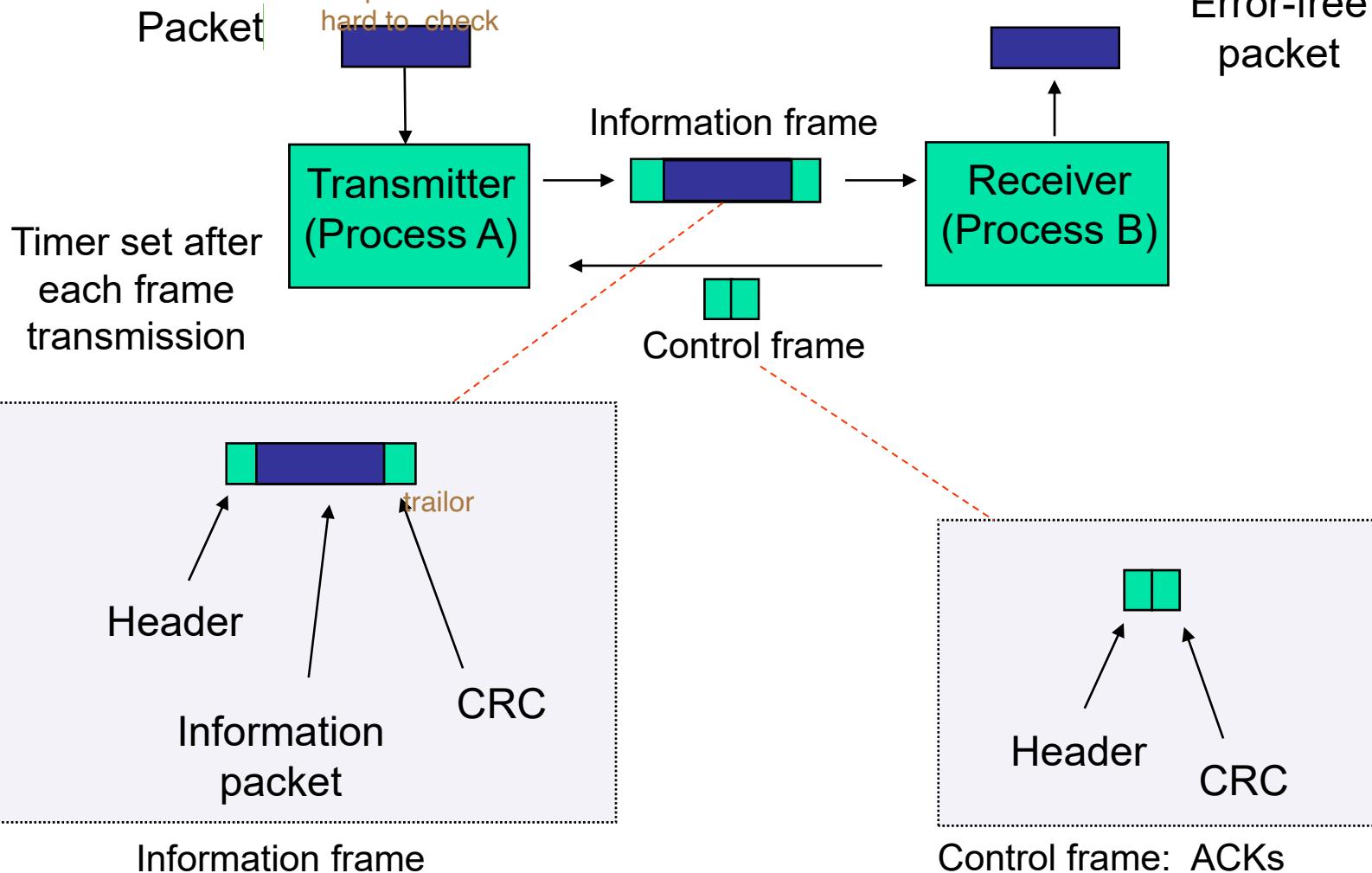
Stop-and-Wait ARQ

Transmit a frame, wait for ACK

wait until sending another package

simple but not efficient

hard to check

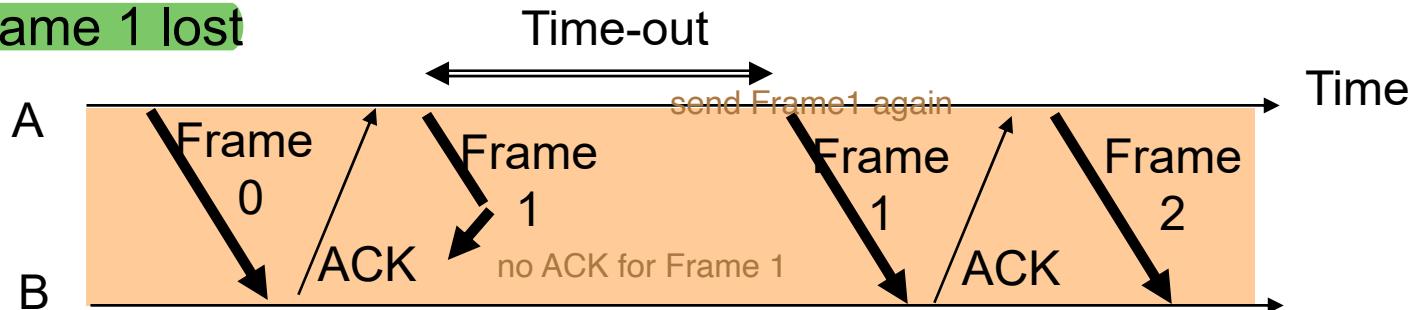


Information frame

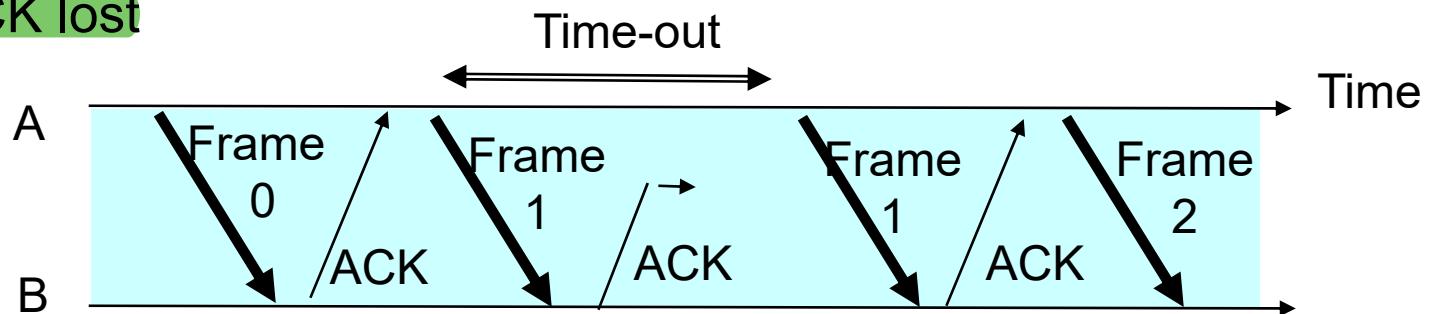
Control frame: ACKs

Need of Sequence Numbers (case 1)

(a) Frame 1 lost



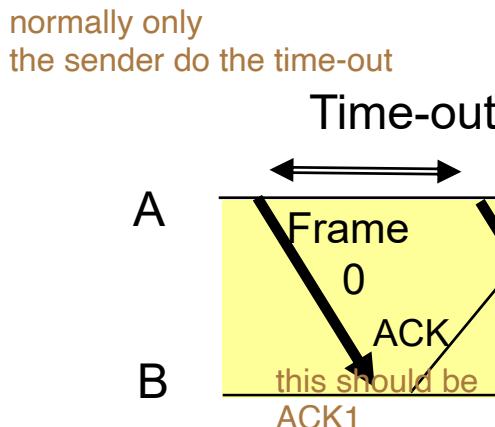
(b) ACK lost



- 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 does the receiver 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

Need of Sequence number (case 2)

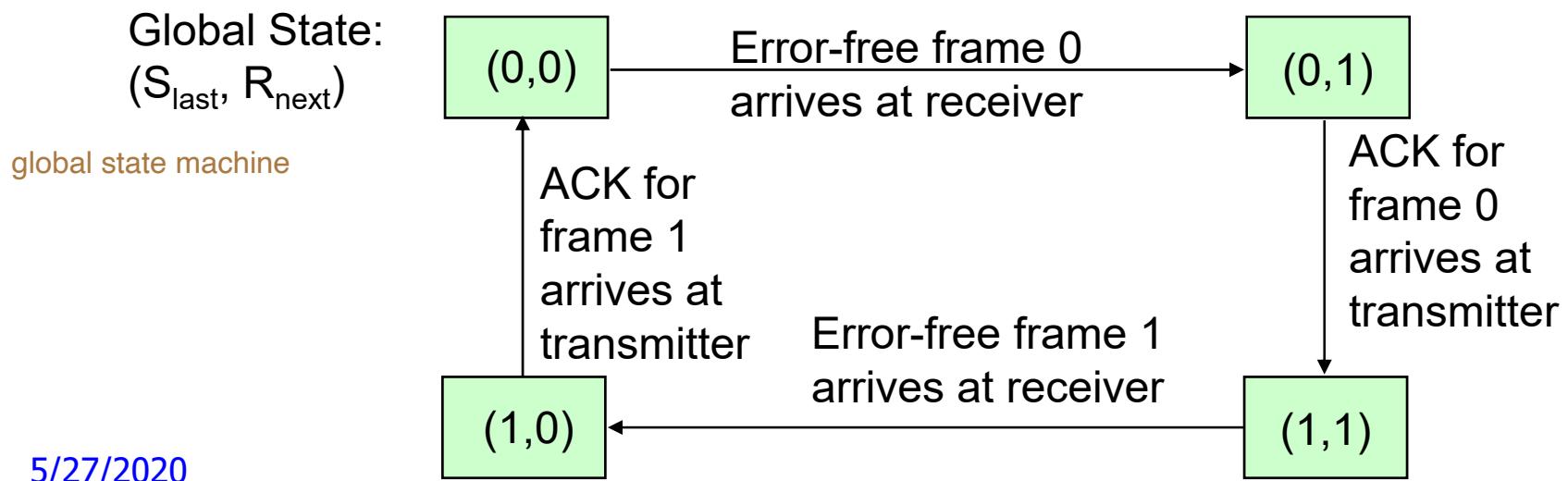
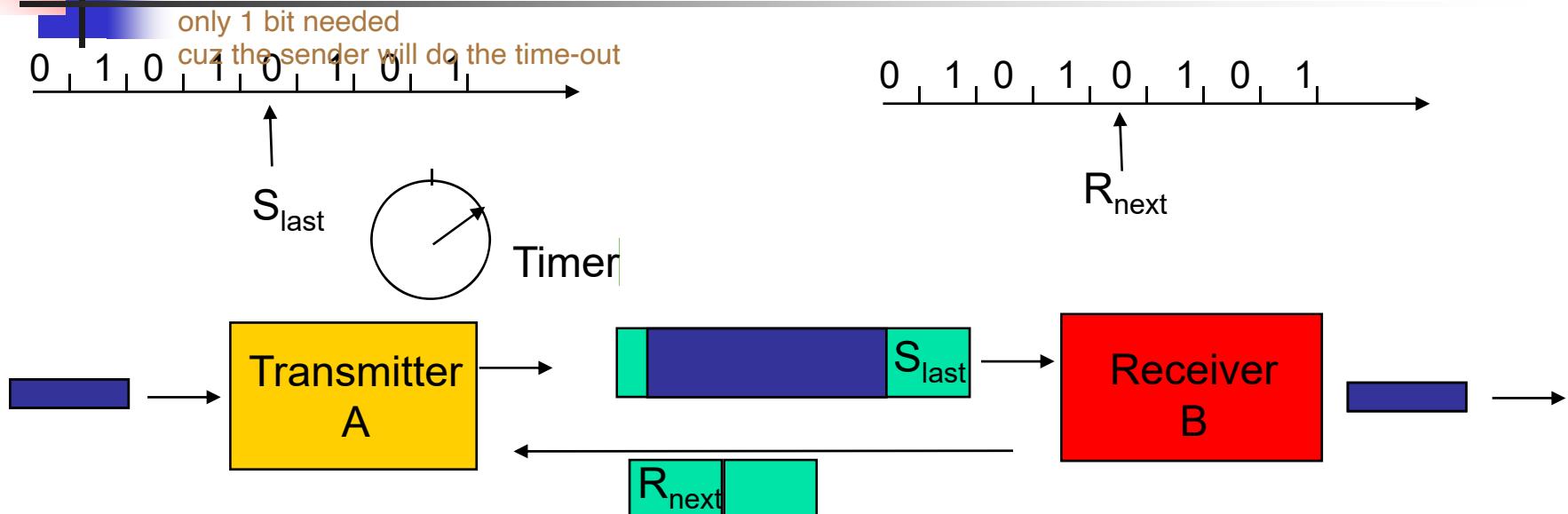
- The transmitting station A misinterprets duplicate ACKs
- Incorrectly assumes second ACK acknowledges Frame 1
- Question: How can the receiver know the second ACK is for frame 0?
- Answer: **Add frame sequence number in ACK header**
- R_{next} is sequence number of next frame expected by the receiver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



In this case, A mistaken the second ACK of Frame 0 as the ACK of Frame 1 and thus it sent out Frame 2. This is wrong because Frame 1 didn't arrive at B successfully.

ACK number is the frame expected
= number of the received Frame +1

1-Bit Sequence Numbering Suffixes



Mechanisms of 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, update S_{last} ($S_{last} = R_{next}$), and 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
 - dont send back ACK

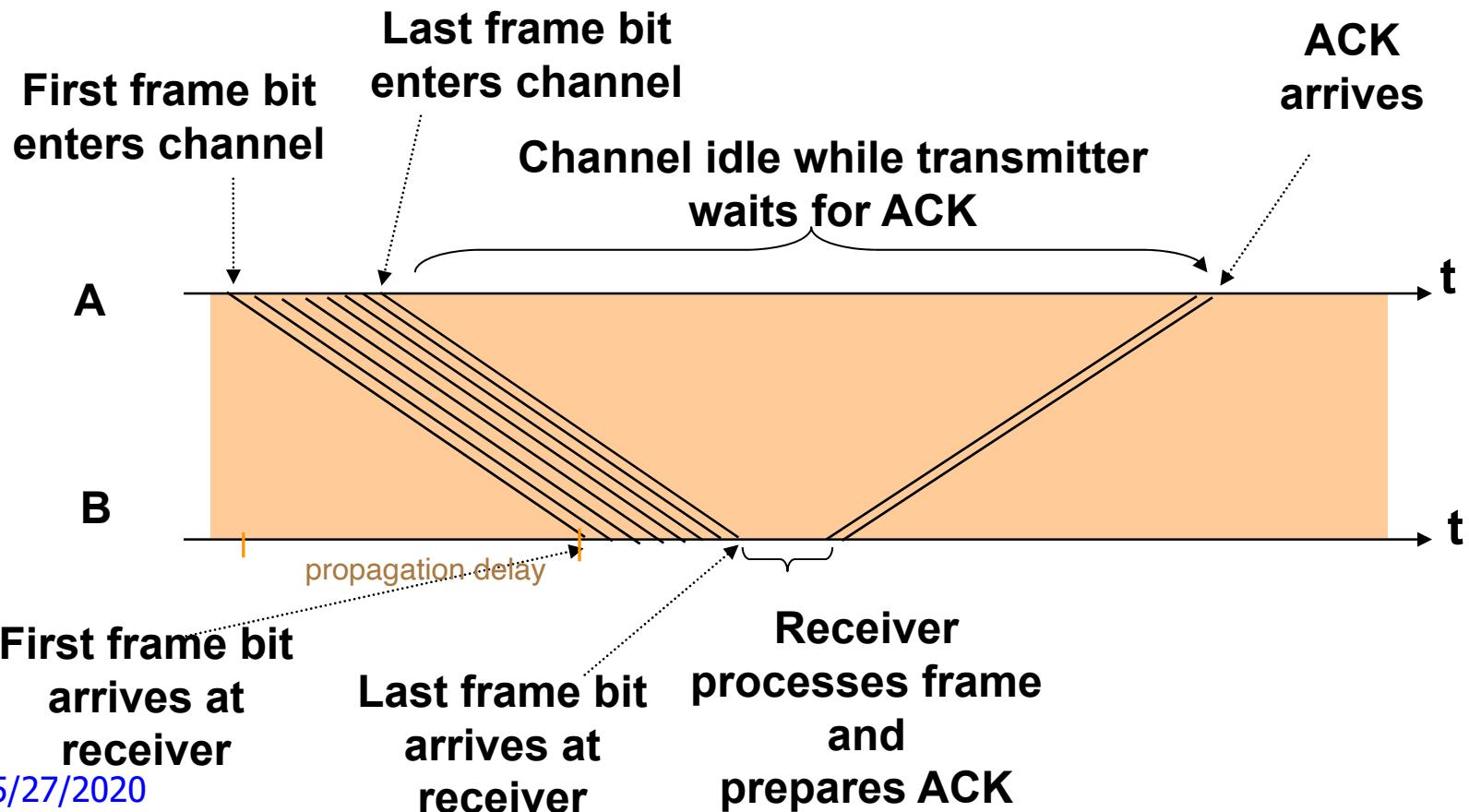


Applications of Stop-and-Wait ARQ

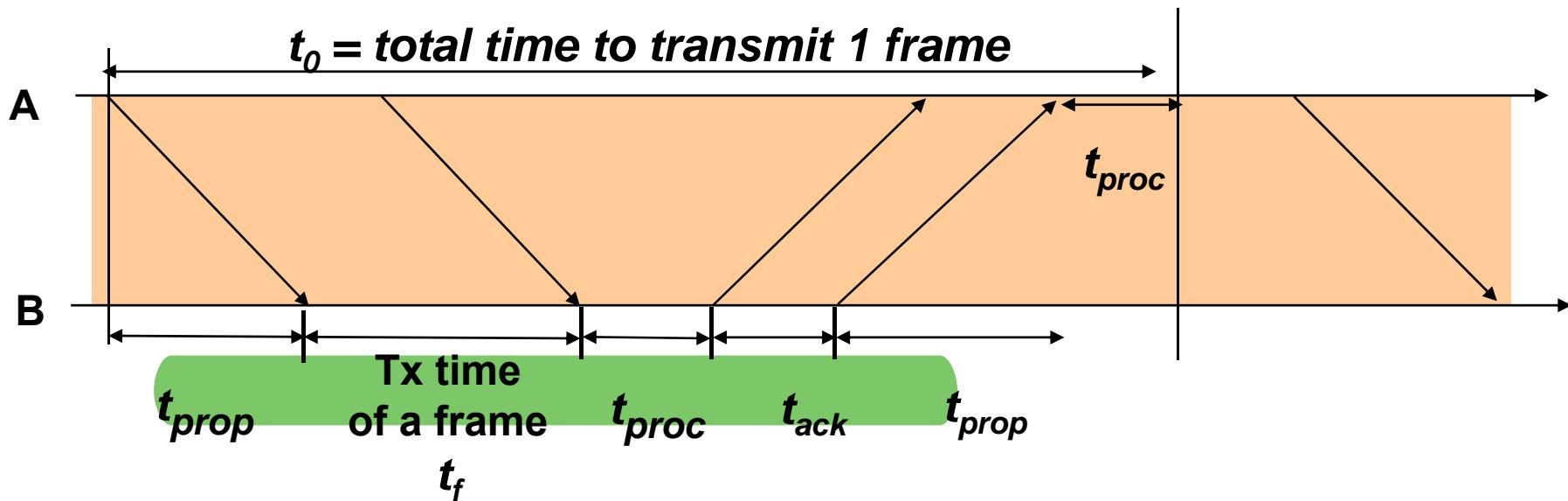
- IBM *Binary Synchronous Communications protocol* (Bisync): character-oriented data link control
- *Xmodem*: modem file transfer protocol
- *Trivial File Transfer Protocol* (RFC 1350): simple protocol for file transfer over UDP
- 802.11 CSMA/CA (part of collision avoidance)

Efficiency of Stop-and-Wait

- 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



$$\begin{aligned} t_0 &= 2t_{prop} + 2t_{proc} + t_f + t_{ack} && \text{bits/info frame} \\ &= 2t_{prop} + 2t_{proc} + \frac{n_f}{R} + \frac{n_a}{R} && \text{bits/ACK frame} \end{aligned}$$

transmission rate

Stop-and-Wait Efficiency on Error-free channel

nawadays, wifi still use wait and delay,
cuz the delay is very small, so we can still handle

data center network

Effective transmission rate:

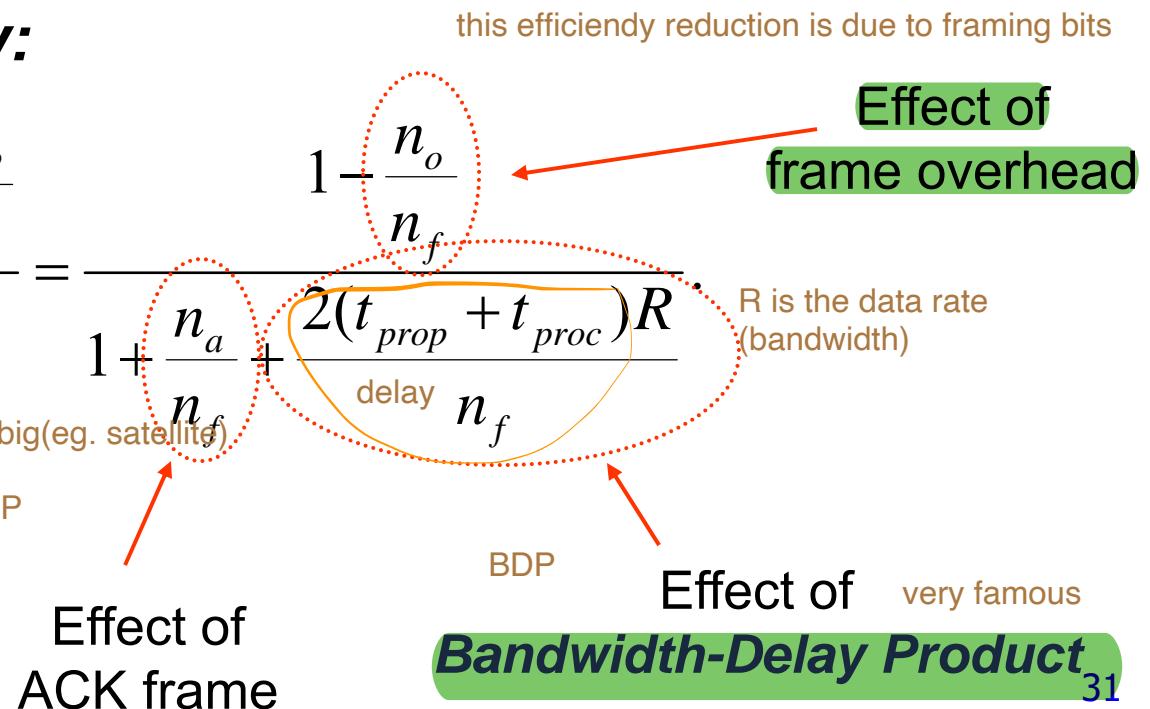
$$R_{\text{eff}}^0 = \frac{\text{number of information bits delivered to destination}}{\text{total time required to deliver the information bits}} = \frac{n_f - n_o}{t_0},$$

bits for header & CRC

Transmission efficiency:

$$\eta_0 = \frac{R_{\text{eff}}}{R} = \frac{\frac{n_f - n_o}{t_0}}{R} =$$

maybe bandwidth and delay are all very big(e.g. satellite)
then efficiency is very small
so we dont do stop and wait for large BDP



Example: Impact of Delay-Bandwidth Product

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

	Delay Efficiency Bandwidth	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

If BDP high, avoid using stop and wait

Stop-and-Wait Efficiency in Channel with Errors

- Let $1 - P_f$ = probability that a frame arrives w/o errors
- Avg. # of transmissions to achieve the first correct arrival is then $1 / (1 - P_f)$ P_f is the probability of frame error
- “If 1-in-10 get through without error, then avg. 10 tries to success” – how to use probability to derive?
- Avg. Total Time per frame is then $t_0 / (1 - P_f)$

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

**Effect of
frame loss**

Example: Impact of Bit Error Rate

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

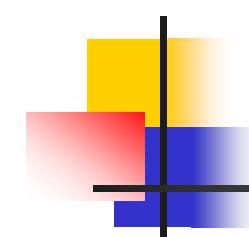
Find efficiency for random bit errors with $p=0, 10^{-6}, 10^{-5}, 10^{-4}$

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

Bandwidth: 1 Mbps & Delay: 1 ms

Bit error	0	10^{-6}	10^{-5}	10^{-4}
$1 - P_f$	1	0.99	0.905	0.368
Efficiency	88%	86.6%	79.2%	32.2%

Bit errors impact performance as $n_f p$ approach 1

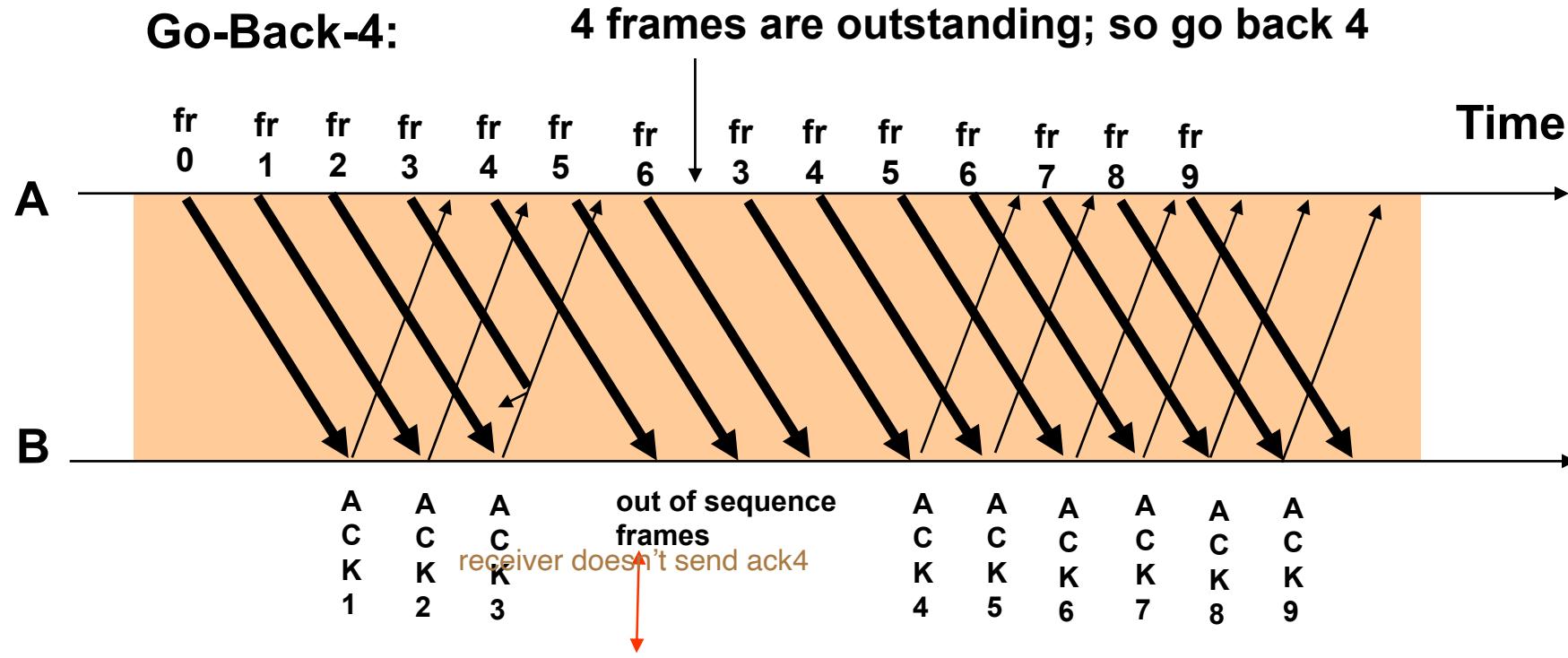


Go-Back-N

- 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

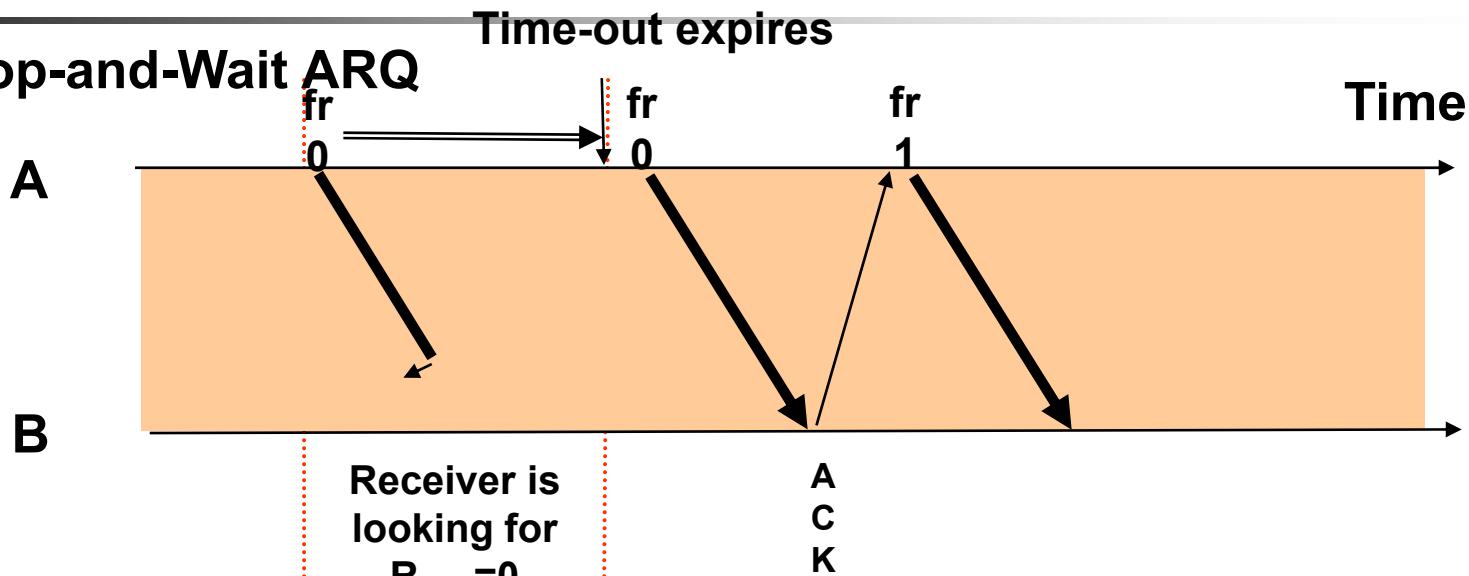
Procedures

- 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

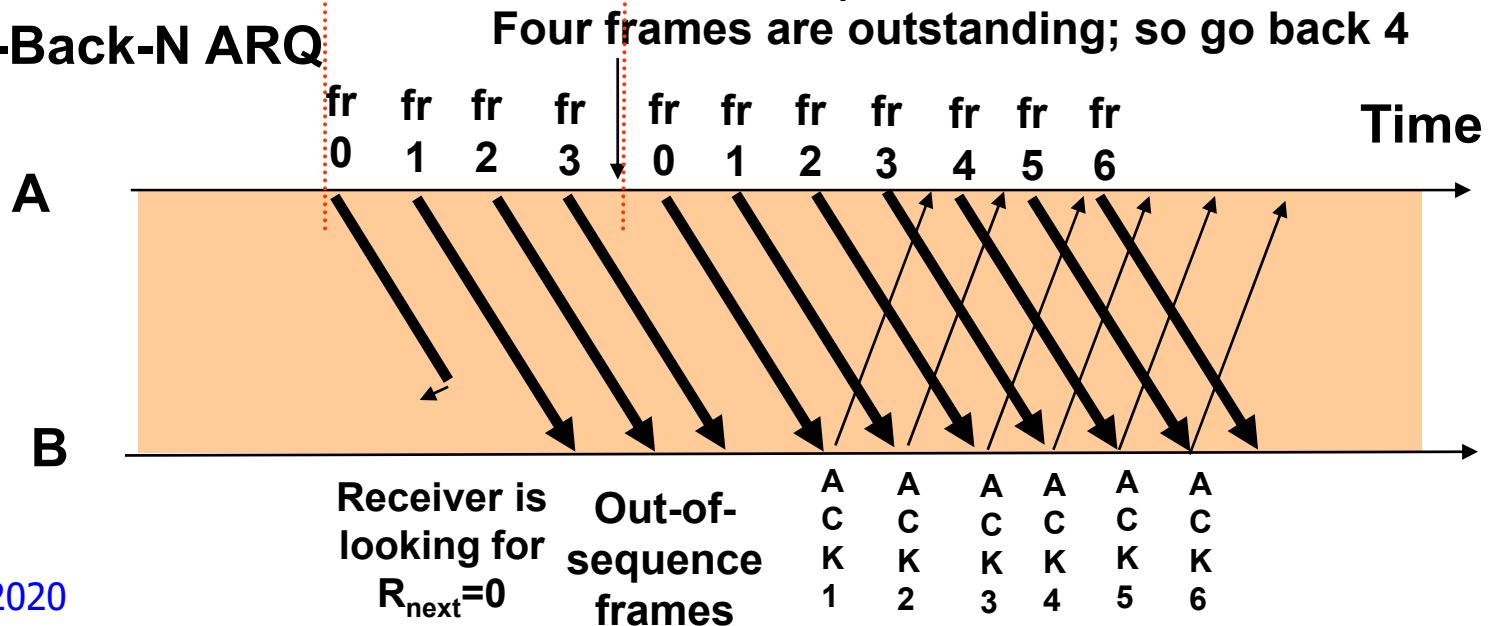


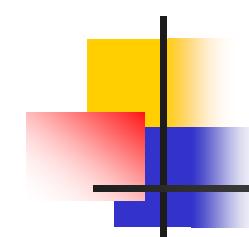
Window Size: Round-Trip Time

Stop-and-Wait ARQ



Go-Back-N ARQ

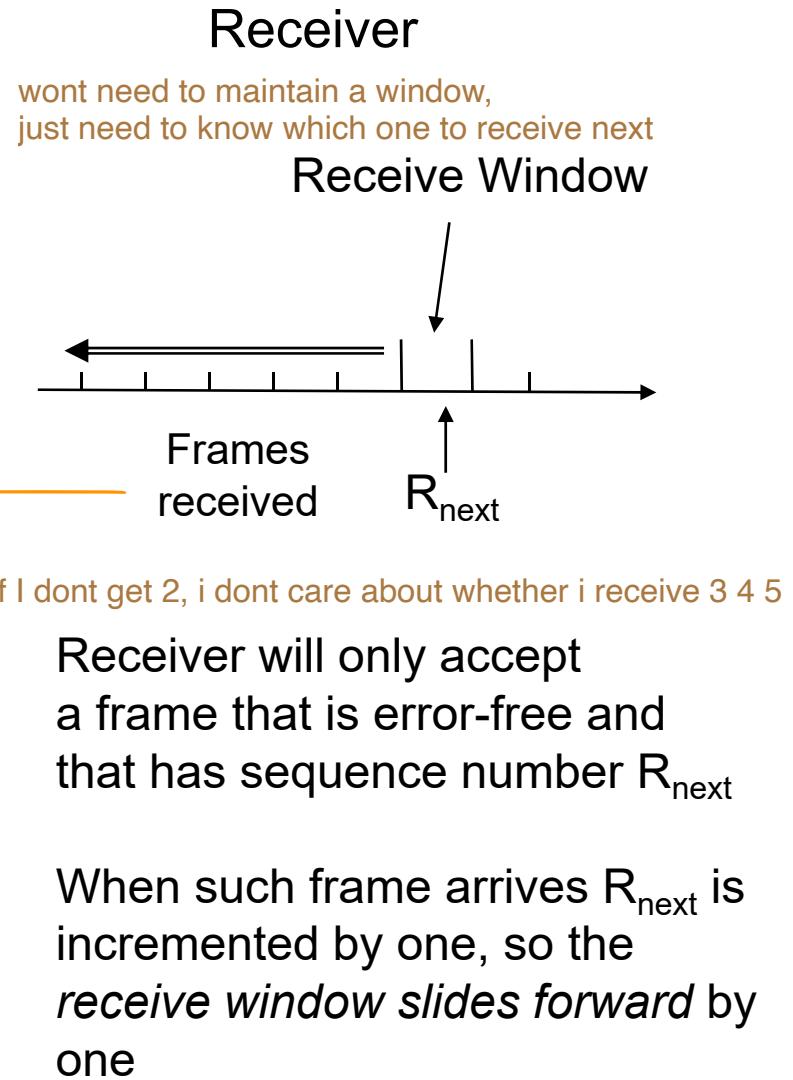
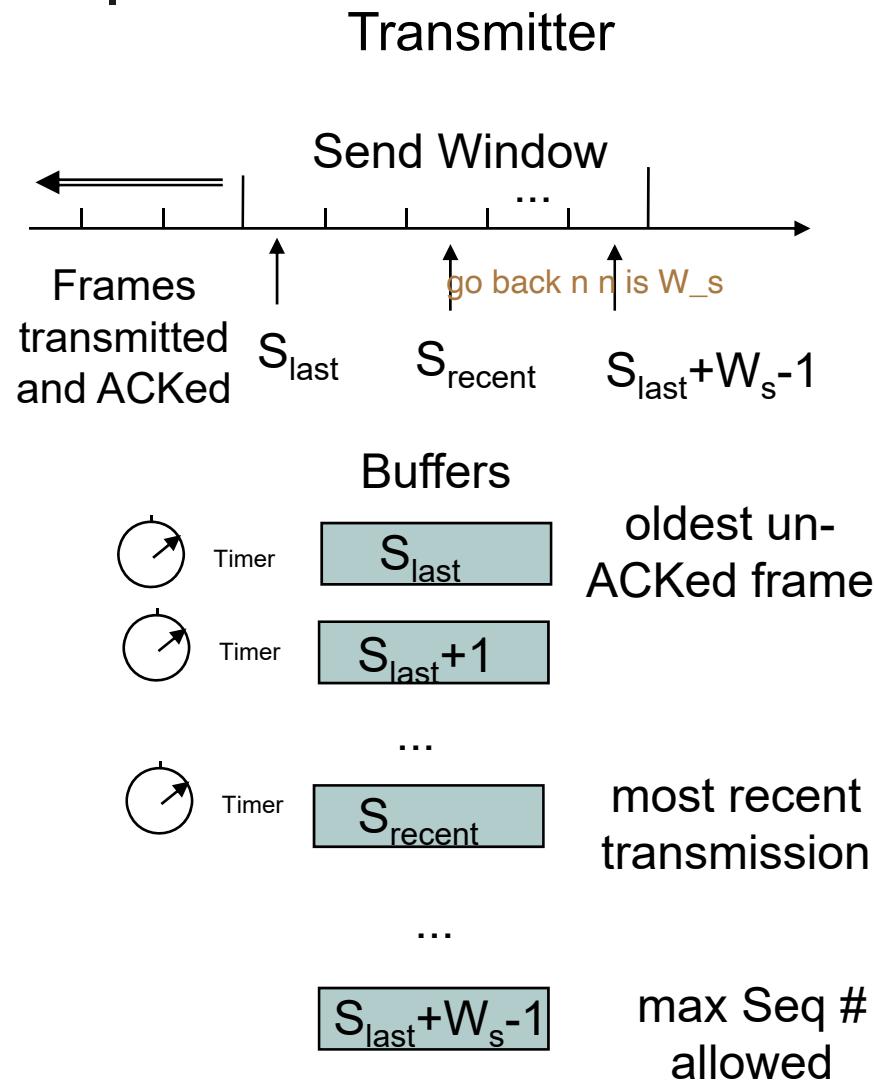




Go-Back-N with Timeout

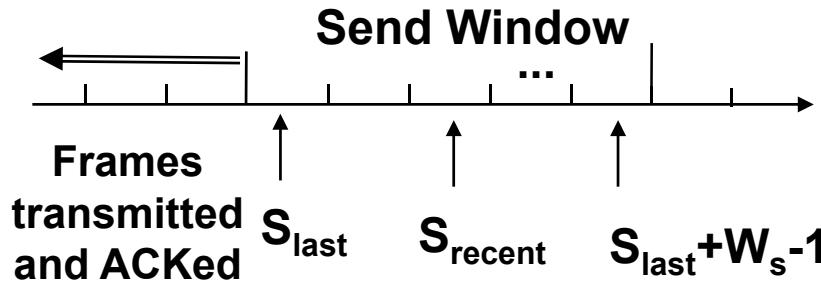
- Problem with Go-Back-N as presented:
 - If frame is lost and source does not have frame to send, then window will not be exhausted and recovery will not commence
- Use a timeout with each frame
 - When timeout expires, resend all outstanding frames

Go-Back-N Transmitter & Receiver



Sliding Window Operation

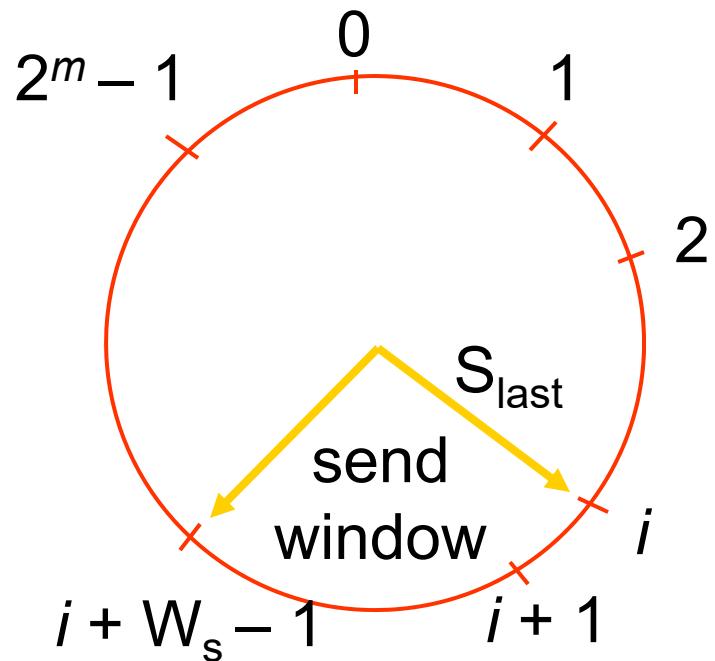
Transmitter



Transmitter waits for error-free ACK frame with sequence number S_{last} ($= R_{next}-1$)

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

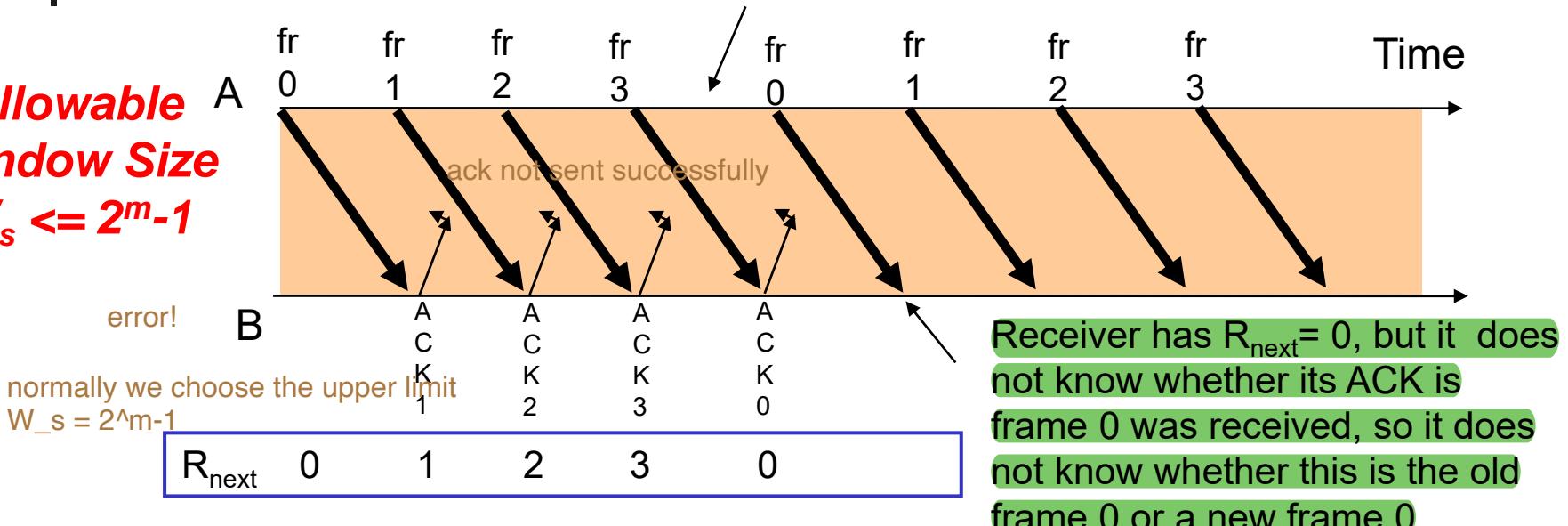
m-bit Sequence Numbering



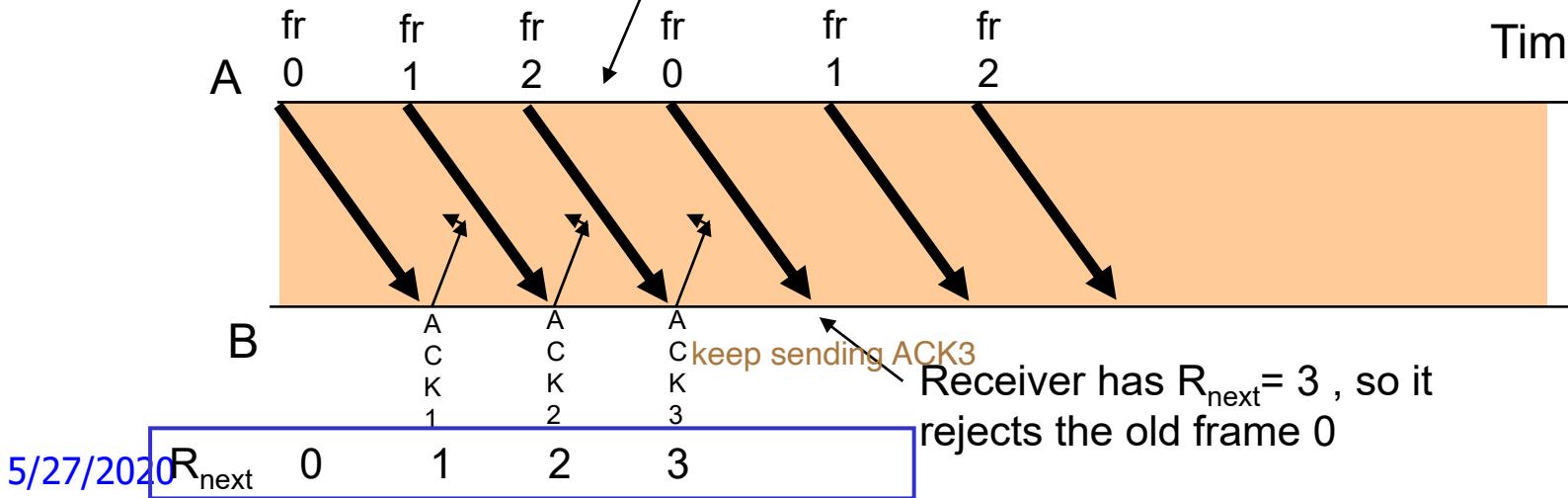
Examples

$M = 2^2 = 4$, Go-Back - 4: Transmitter goes back 4

Allowable Window Size
 $W_s \leq 2^m - 1$

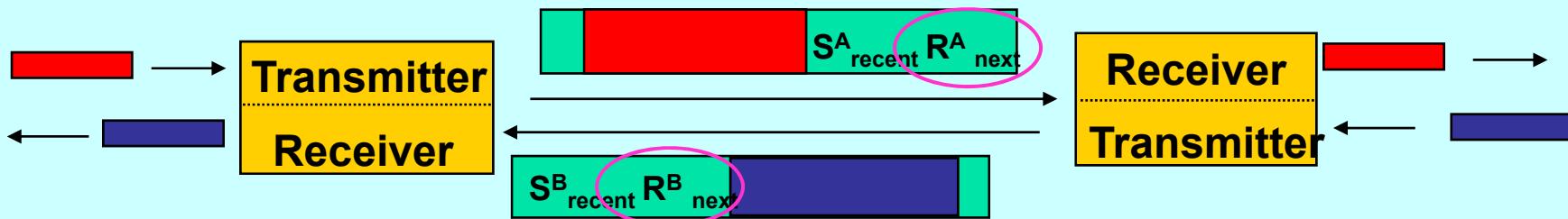


$M = 2^2 = 4$, Go-Back-3: Transmitter goes back 3



TCP use this

ACK Piggybacking in Bidirectional GBN



"A" Receive Window

pack ACK with the data you want to send

"A" Send Window

S^A_{last} $S^A_{last} + W^A_s - 1$

Buffers

- Timer S^A_{last}
- Timer $S^A_{last} + 1$
- ...
- Timer S^A_{recent}
- ...
- Timer $S^A_{last} + W^A_s - 1$

What's the relationship among the 4 windows?

"B" Receive Window

R^B_{next}

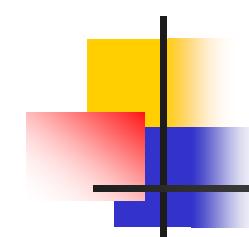
"B" Send Window

S^B_{last} $S^B_{last} + W^B_s - 1$

Buffers

- Timer S^B_{last}
- Timer $S^B_{last} + 1$
- ...
- Timer S^B_{recent}
- ...
- Timer $S^B_{last} + W^B_s - 1$

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

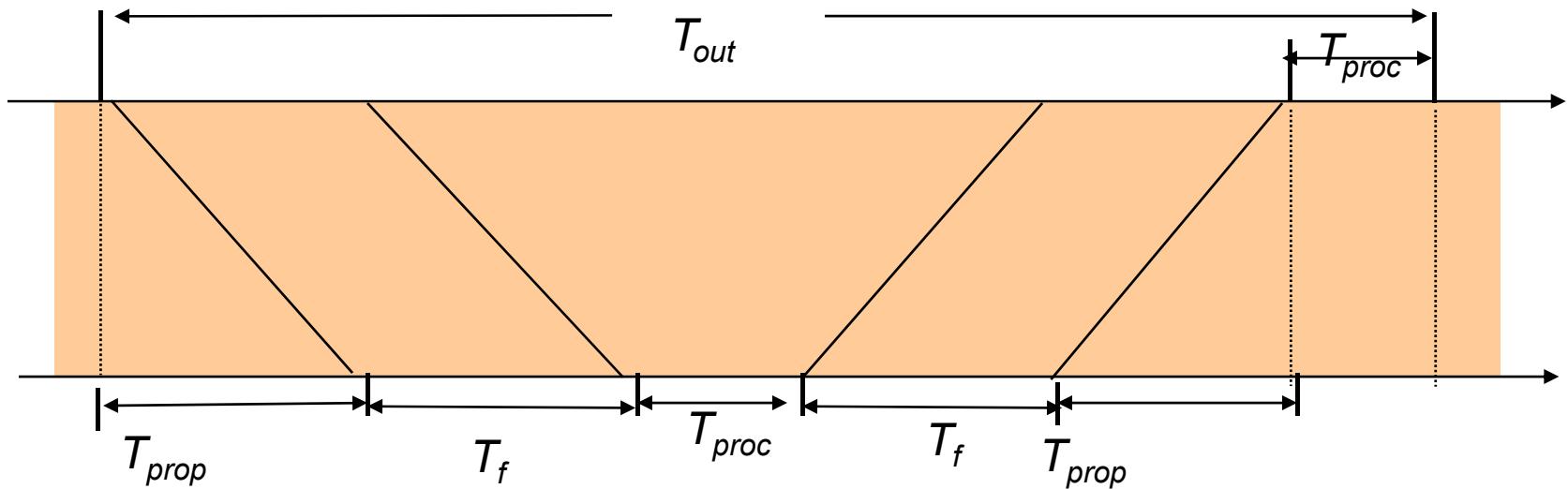


Applications of Go-Back-N ARQ

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

Required Timeout & Window Size

- Timeout value should allow for:
 - 2 propagation times + 2 processing times: $2(T_{prop} + T_{proc})$
 - Transmission time of a frame T_f
 - Transmission time of the next frame carries the ACK, T_f
- W_s should be large enough to keep channel busy for T_{out}



Required Window Size for Bandwidth-Delay Product (BDP)

for larger BDP, larger window size is expected

Frame = 1250 bytes = 10,000 bits, $R = 1 \text{ Mbps}$

$2(t_{\text{prop}} + t_{\text{proc}})$	$2 \times \text{Delay} \times \text{BW}$	Window
1 ms	1000 bits	1
10 ms	10,000 bits	2 $10000/10000+1$
100 ms	100,000 bits	11 $100000/10000+1$
1 second	1,000,000 bits	101 $1000000/10000+1$

How to derive the window size?

Efficiency of Go-Back-N

- GBN is completely efficient, if W_s large enough to keep channel busy, and if channel is error-free
- However, assuming frame loss probability P_f , then the 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

$$t_{GBN} = t_f (1 - P_f) + P_f \left\{ t_f + \frac{W_s t_f}{1 - P_f} \right\} = t_f + P_f \frac{W_s t_f}{1 - P_f} \quad \text{and}$$
$$\eta_{GBN} = \frac{\frac{n_f - n_o}{t_{GBN}}}{R} = \frac{1 - \frac{n_o}{n_f}}{1 + (W_s - 1)P_f} (1 - P_f)$$

not success, delay

efficiency reach the upper limit of the transmission if $P_f=0$

performance still depends on BDP

BDP determines W_s

Example: Bit Error Rate Impact 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

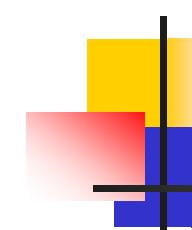
1 Mbps x 100 ms = 100000 bits = 10 frames ? Use $W_s = 11$

However if your channel is nearly error free, the large DBP won't affect you
you can still use go back n

but if large error rate and large DBP, do not use go back n, you the third one

ARQ	Efficiency p	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*



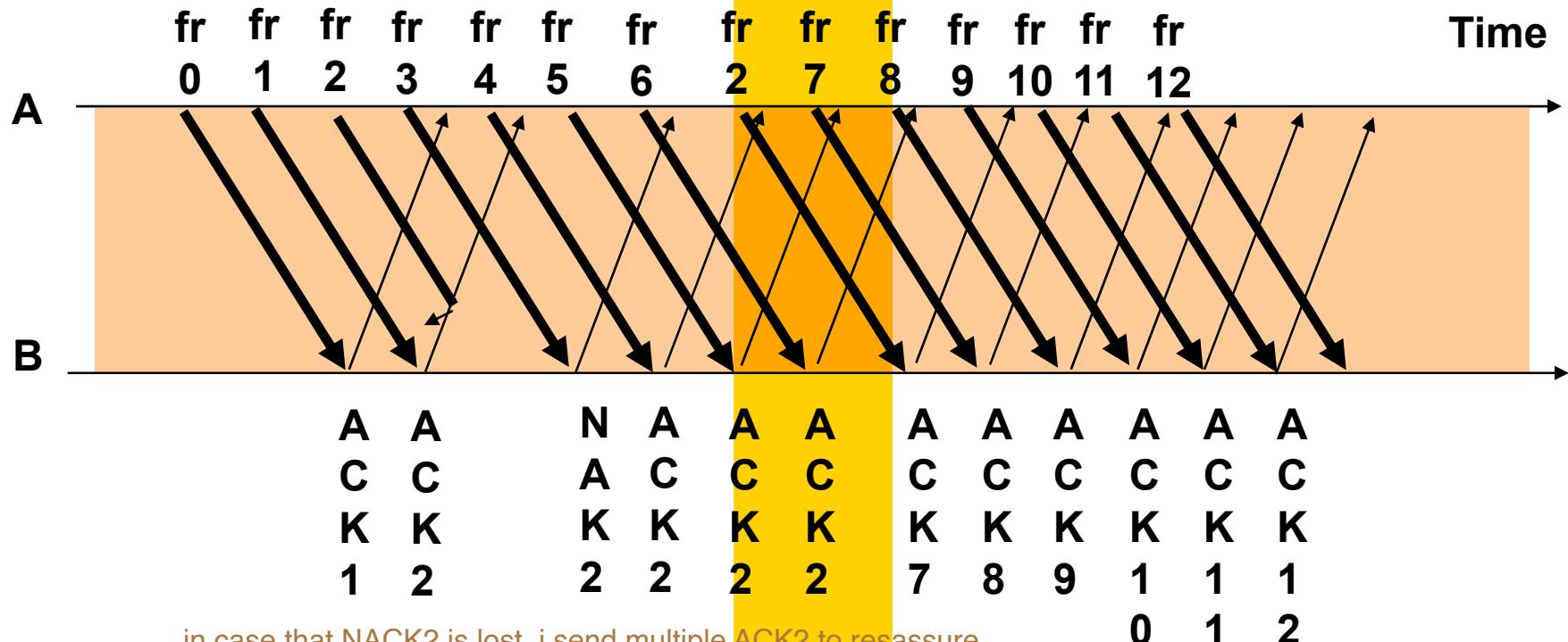
Selective Repeat ARQ

- 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

Mechanism

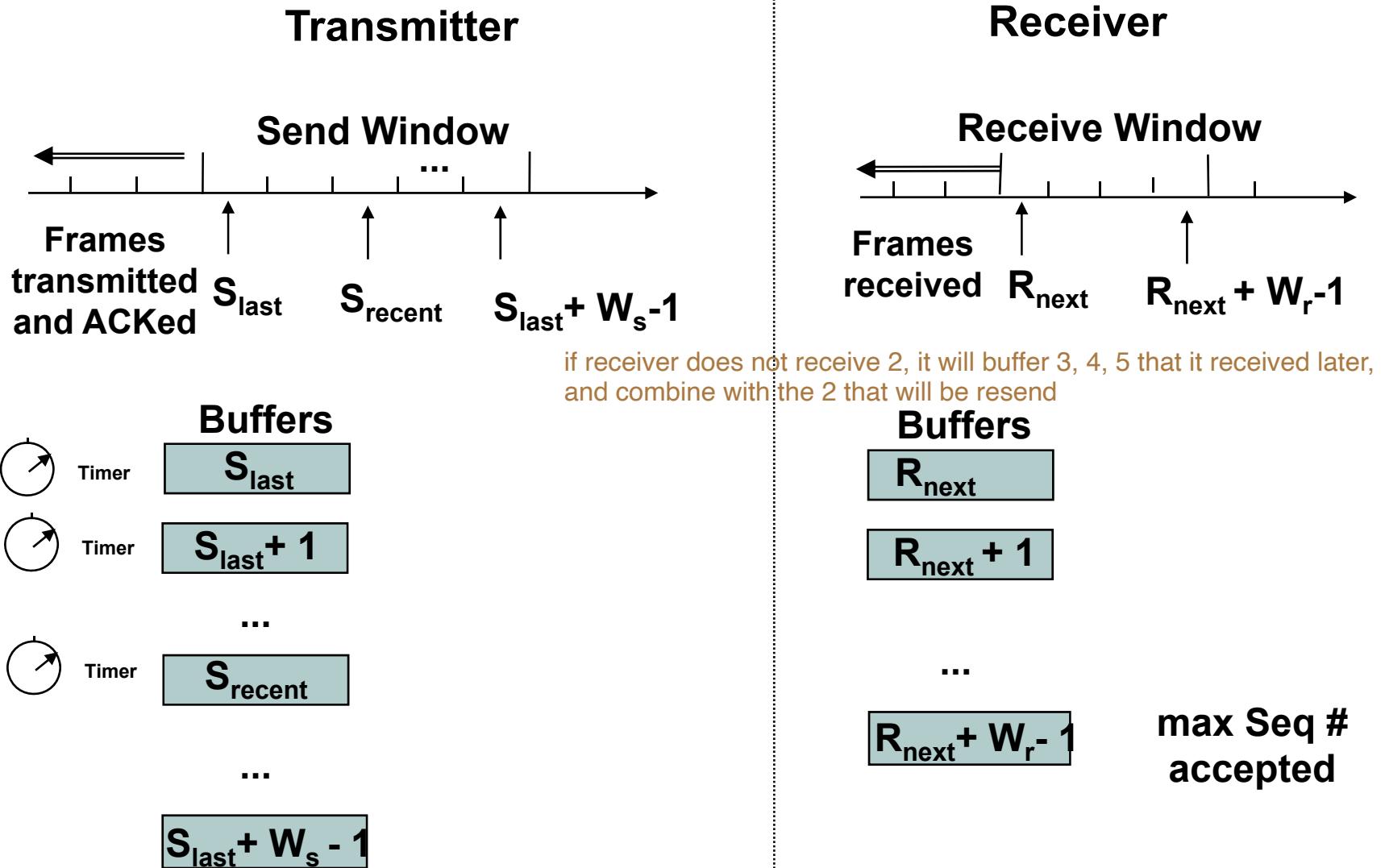
what if fr5 is also lost, we have different ways to handle

ACK always follows NACK to support that



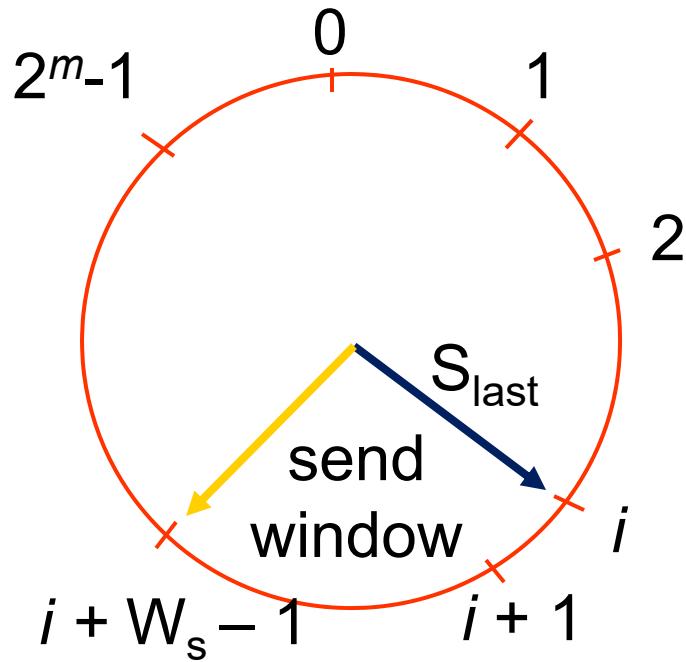
Since fr3 arrives, B knows fr2 is lost.

Transmitter and Receiver of Selective ARQ

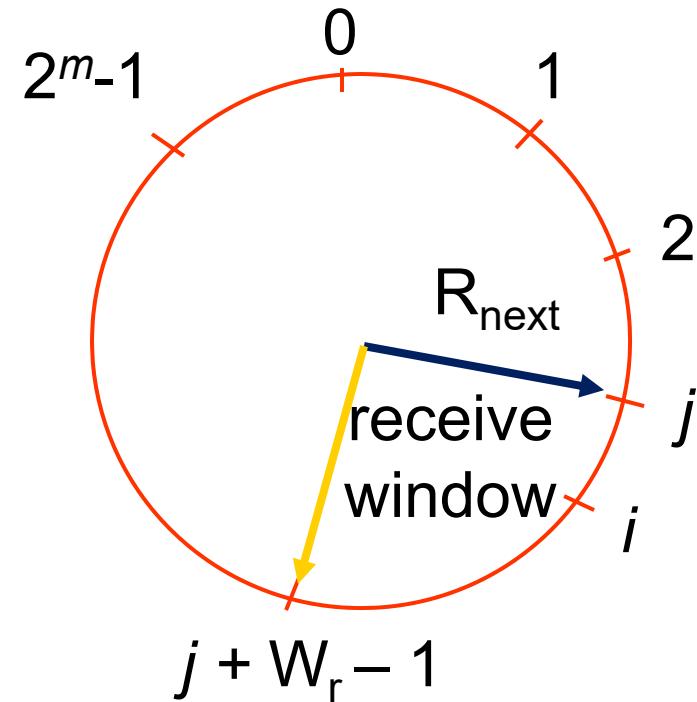


Send & Receive Windows

Transmitter



Receiver



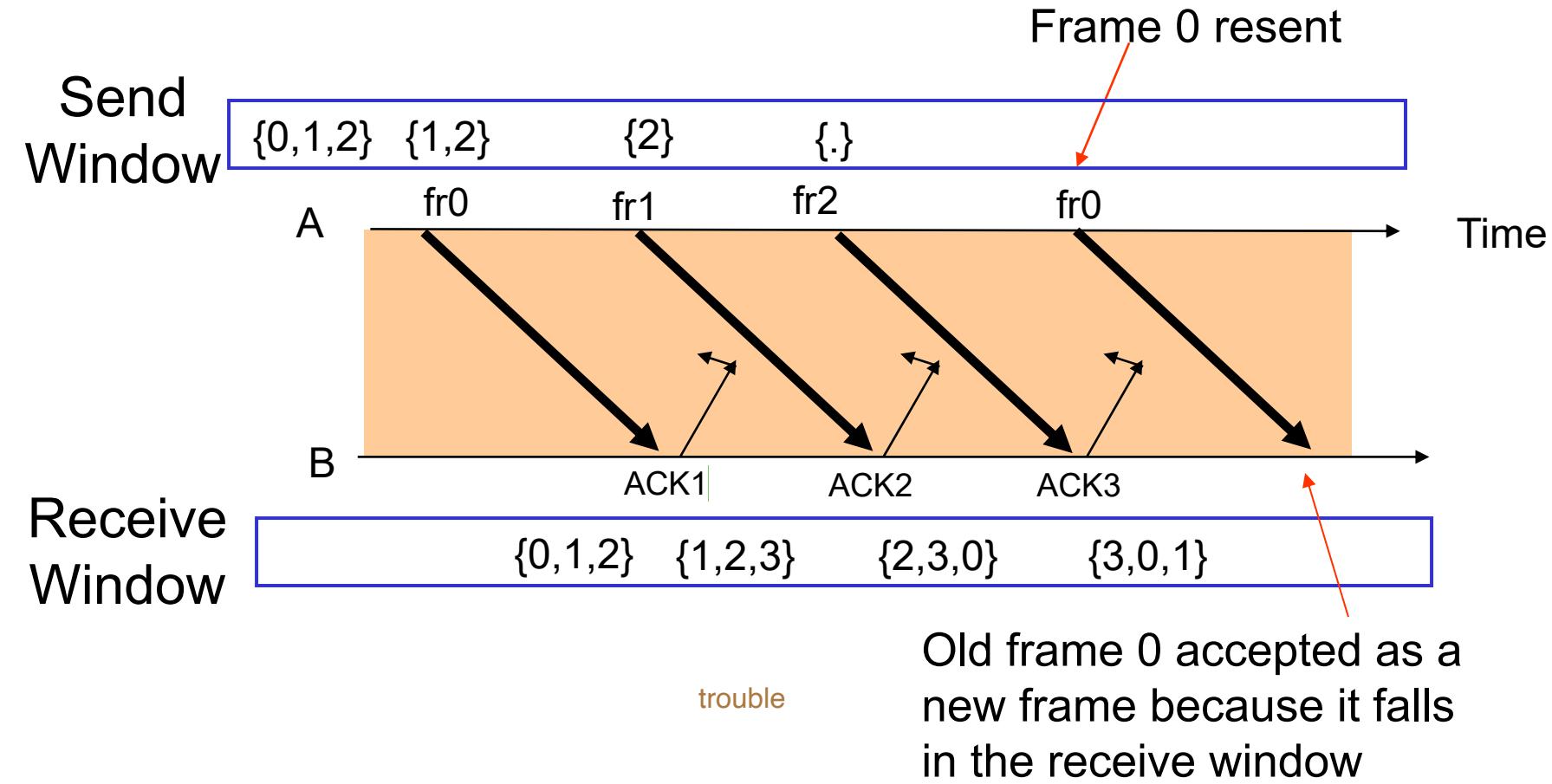
Moves k forward when ACK arrives with $R_{next} = S_{last} + k$
 $k = 1, \dots, W_s - 1$

5/27/2020

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

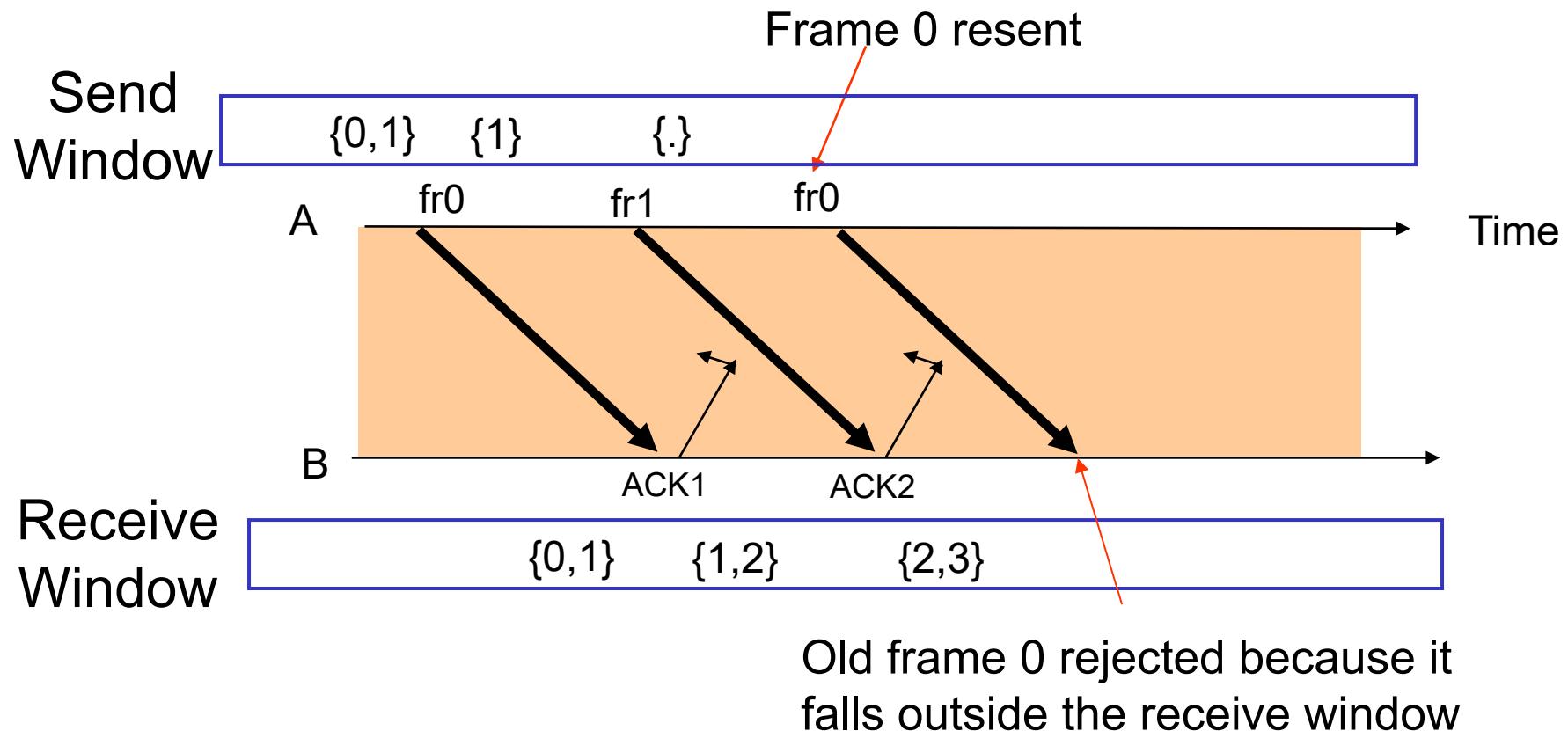
What size W_s and W_r allowed? (Example 1)

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



What size W_s and W_r allowed? (Example 2)

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

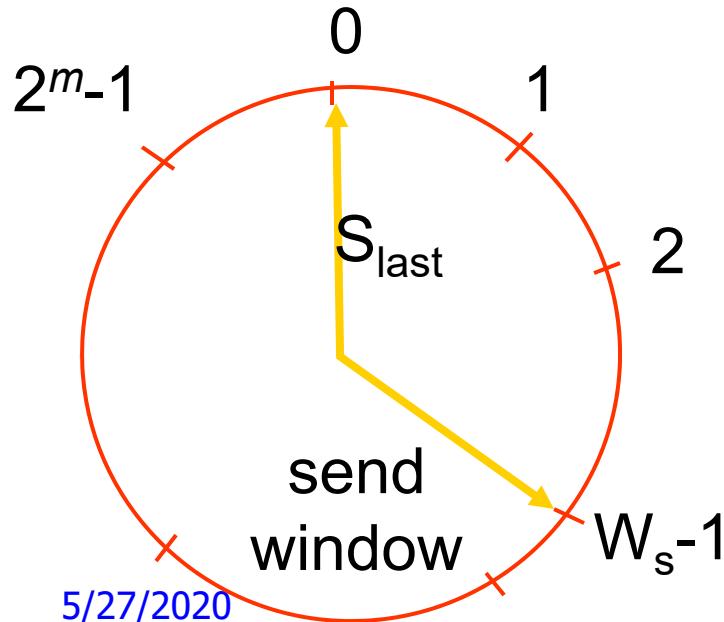


Maximum Windows

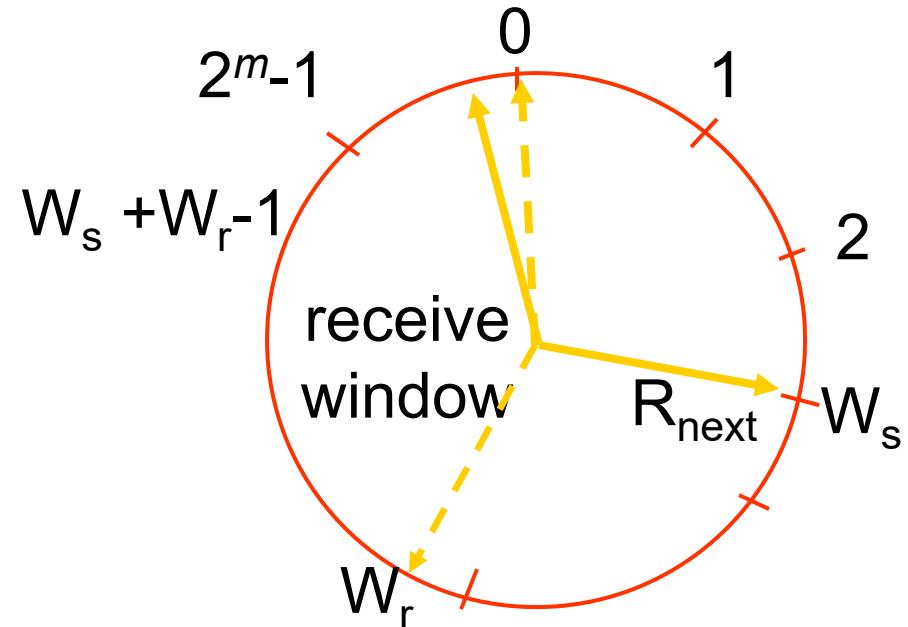
$$W_s + W_r \leq 2^m$$

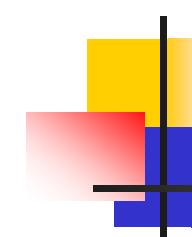
- $W_s + W_r = 2^m$ is maximum allowed for m-bit SN
- Why?

- Transmitter sends frames 0 to $W_s - 1$; send window empty
 - transmitter exhausted the window, so does the receiver
- All arrive at receiver
- All ACKs lost
- Transmitter resends frame 0



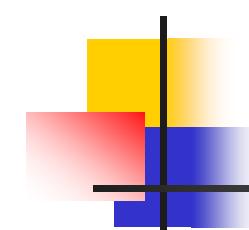
- Receiver window starts at $\{0, \dots, W_r - 1\}$
 - receiver receives, so the window will forward
- Window slides forward to $\{W_s, \dots, W_s + W_r - 1\}$
- Receiver rejects frame 0 because it is outside receive window





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

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

Comparisons of ARQs (Example of BER Impact)

$n_f=1250$ 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 $R= 1 \text{ Mbps} & 100 \text{ ms}$

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*
- *Not sensitive to large BDP*

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: impt

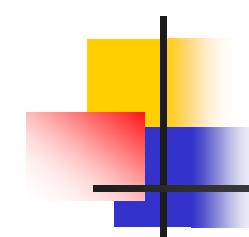
$$\eta_{SR} = (1 - P_f)(1 - \frac{n_o}{n_f}) \approx (1 - P_f)$$

Go-Back-N: *For $P_f \approx 0$, SR & GBN same*

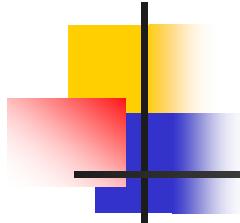
$$\eta_{GBN} = \frac{1 - P_f}{1 + (W_s - 1)P_f} = \frac{1 - P_f}{1 + LP_f}$$

Stop-and-Wait: *For $P_f \rightarrow 1$, GBN & SW same*

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



Efficiency Comparisons

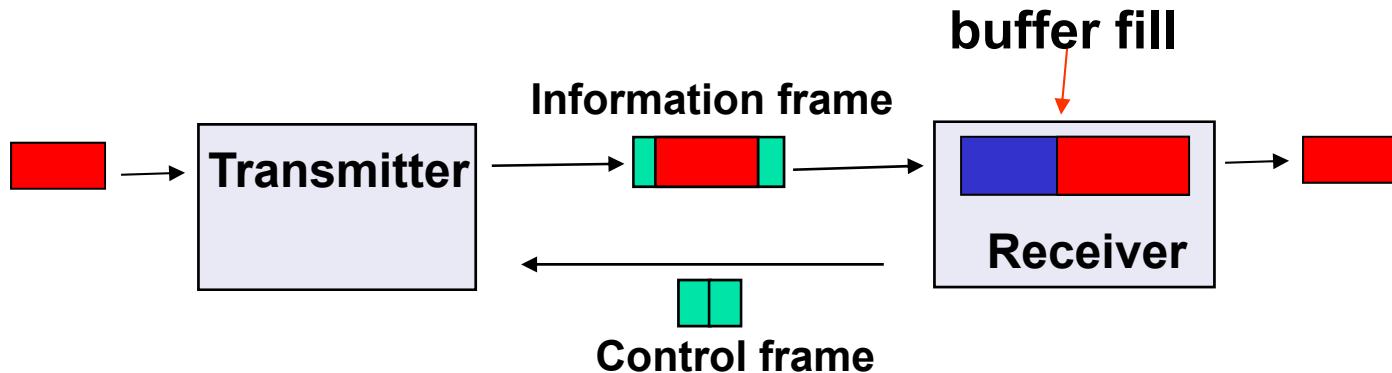


Flow Control in the Data Link Layer

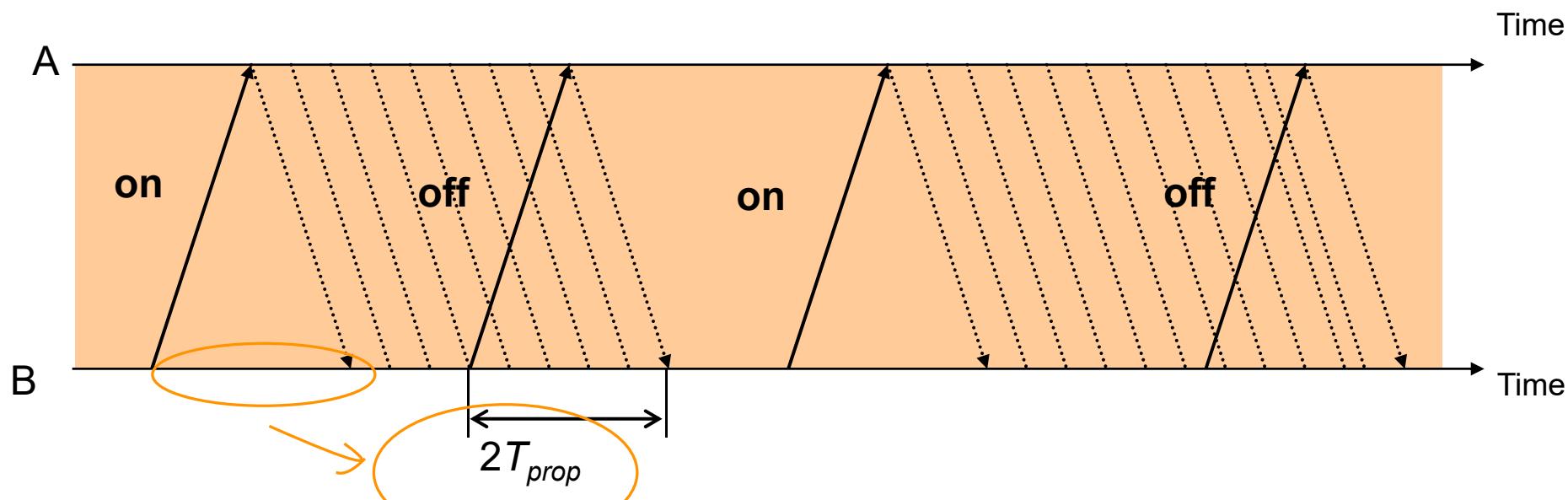
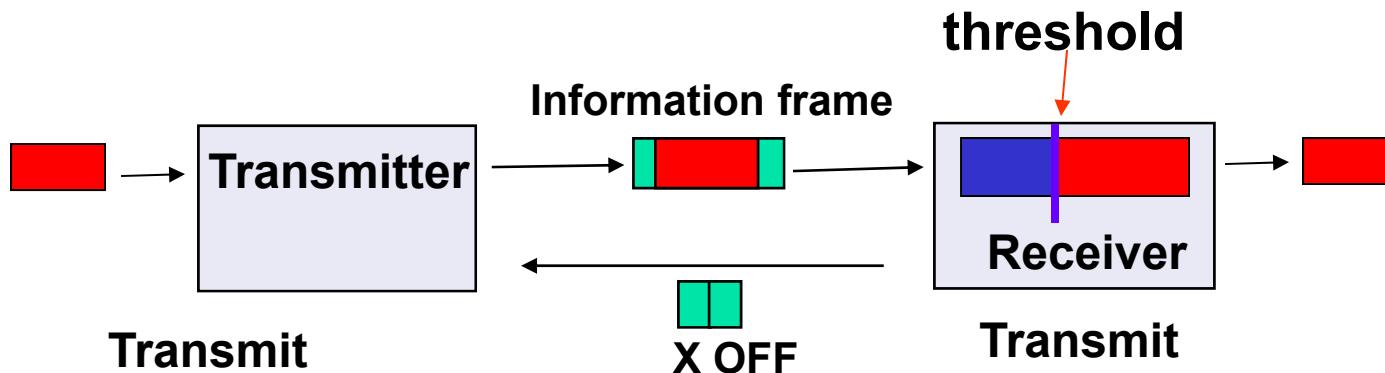
Flow Control

transmitter stronger than receiver, then receiver overload

- Receiver has limited buffering to store arriving frames
needs receiver feedback
- 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

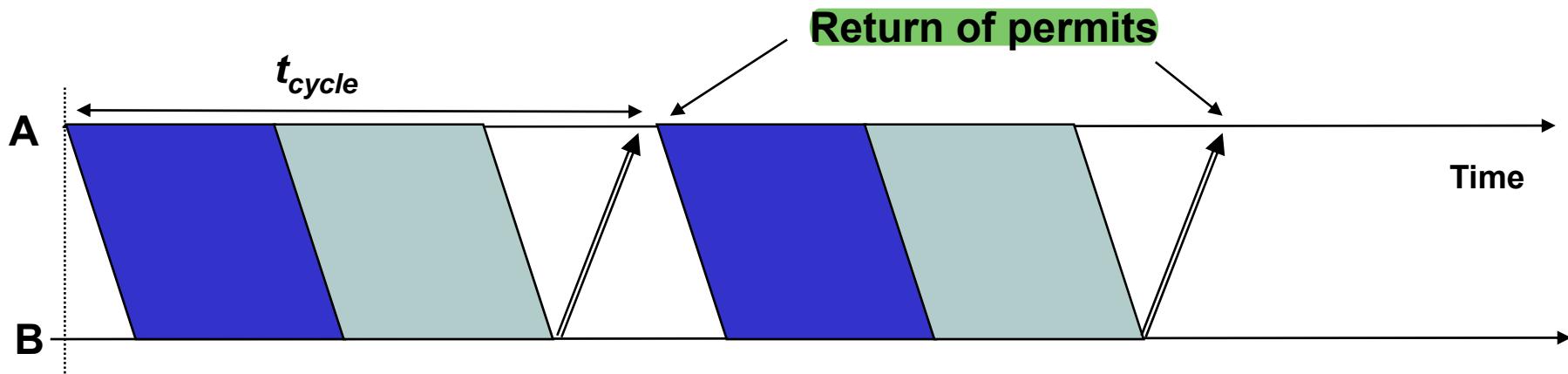


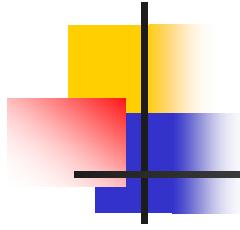
Threshold must activate OFF signal while $2 T_{prop} R$ bits still remain in buffer

Window Flow Control

flow control and ARQ can be done together

- 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
 - ACK considered as the ON
- Can also pace ACKs as shown below
 - Return permits (ACKs) at end of cycle regulates transmission rate
- Problems using sliding window for both error & flow control
 - Choice of window size
 - window size determined by receiver buffer size
 - Interplay between transmission rate & retransmissions
 - TCP separates error & flow control
 - interplay: feedback, commu





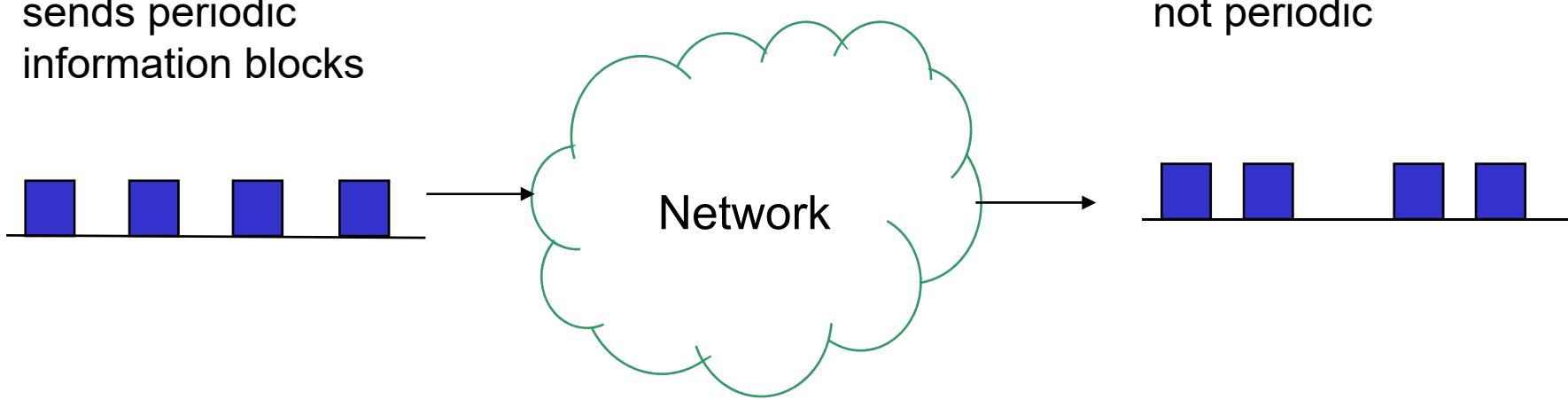
Timing Recovery

Timing Recovery for Synchronous Services

cares about data interval

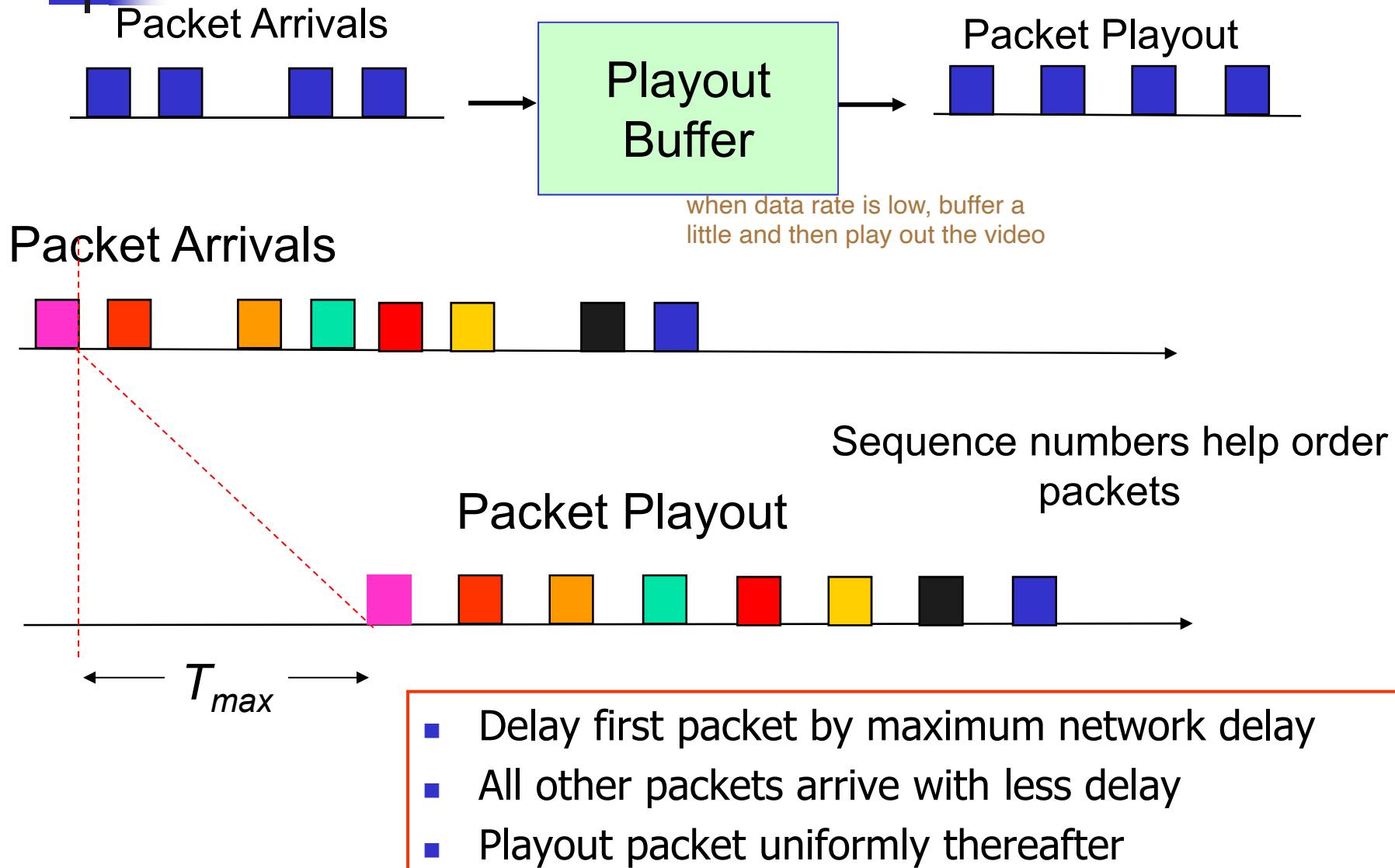
- Applications that involve voice, audio, or video can generate a **synchronous information stream**
- Information carried by equally-spaced fixed-length packets
- Network multiplexing & switching introduces random delays
 - Packets experience variable transfer delay
 - Jitter (variation in interpacket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

Synchronous source
sends periodic
information blocks



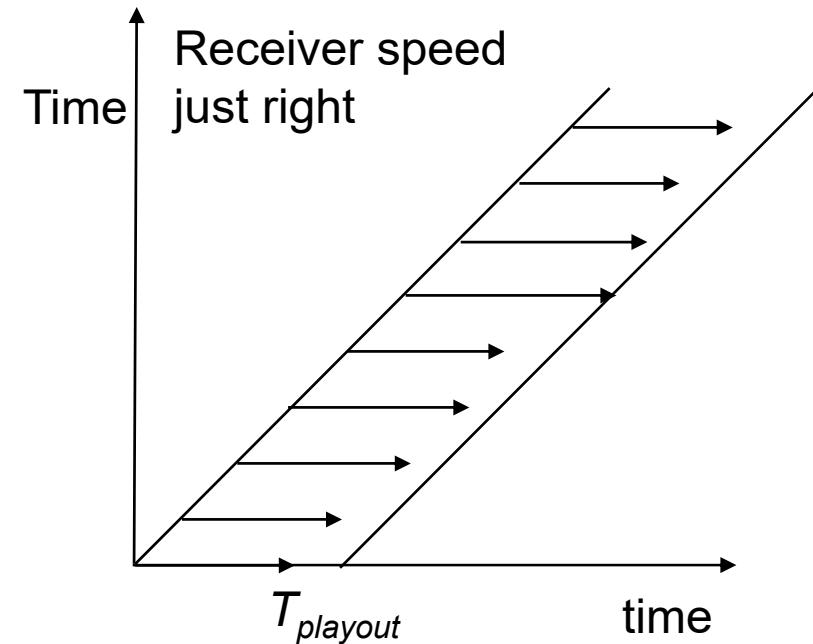
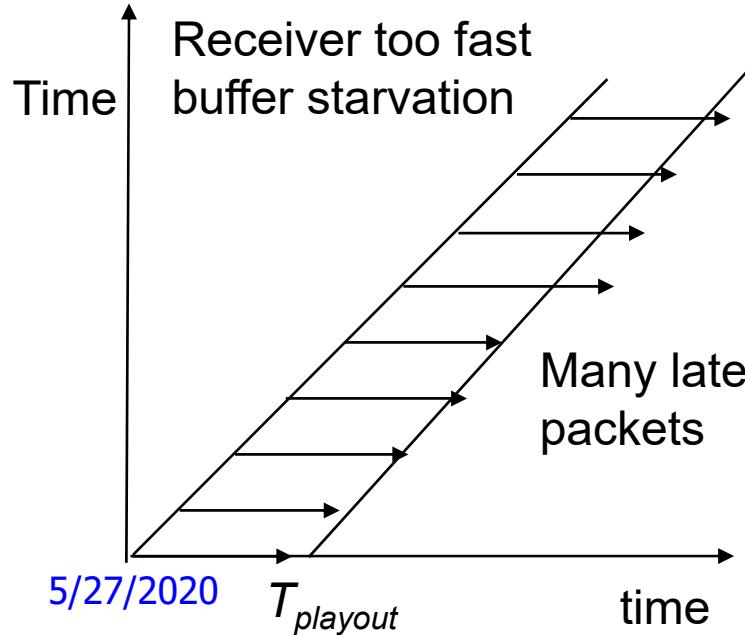
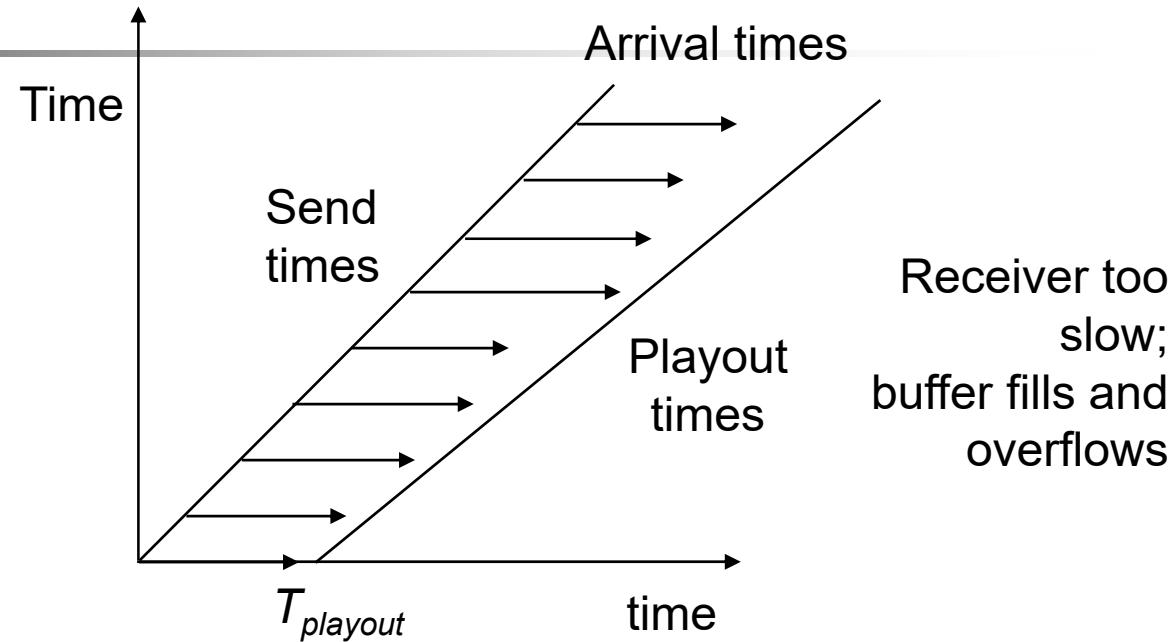
Network output
not periodic

Scheme 1: Playout Buffer



Different Scenarios of Playout and Motivation for Scheme 2

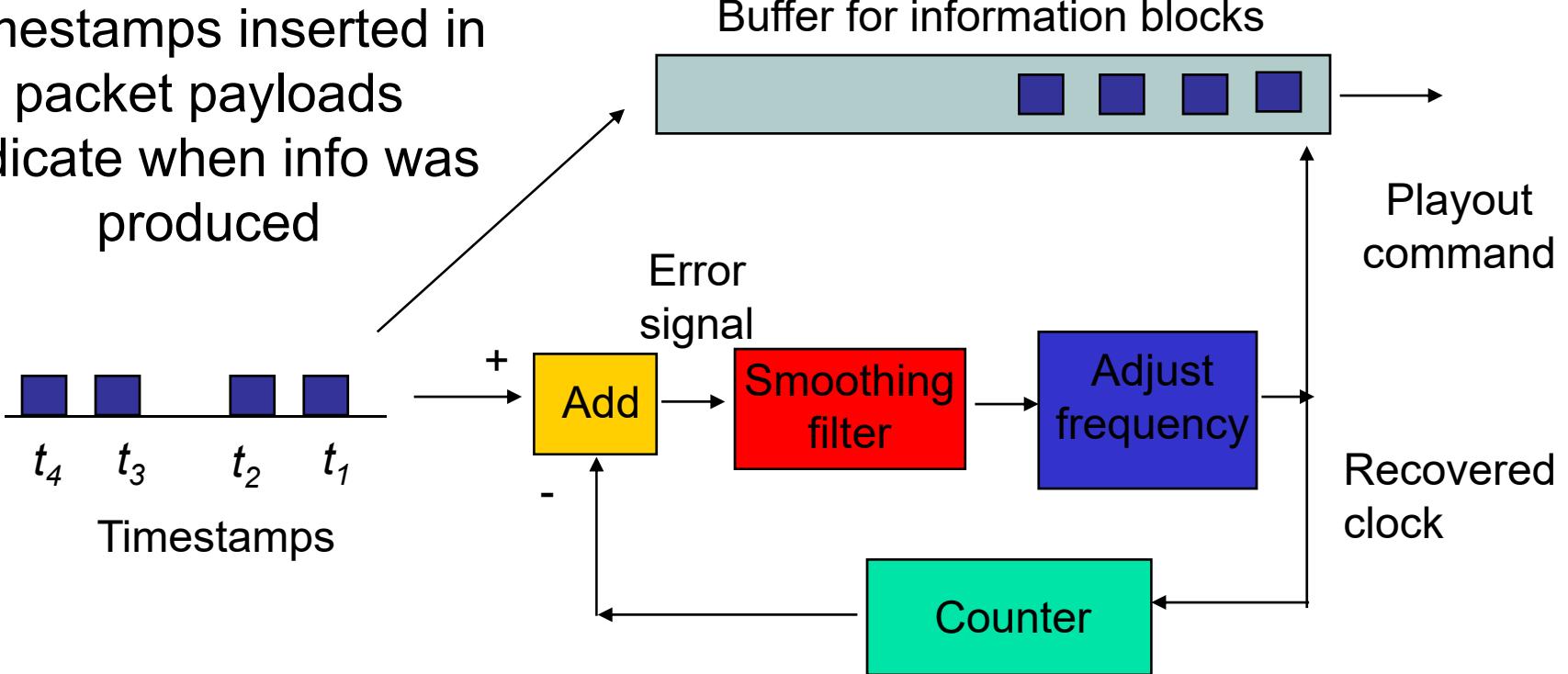
Playout clock must be synchronized to transmitter clock



Scheme 2: Clock Recovery Based on Frequency Tracking

Phase locked loop
not covered

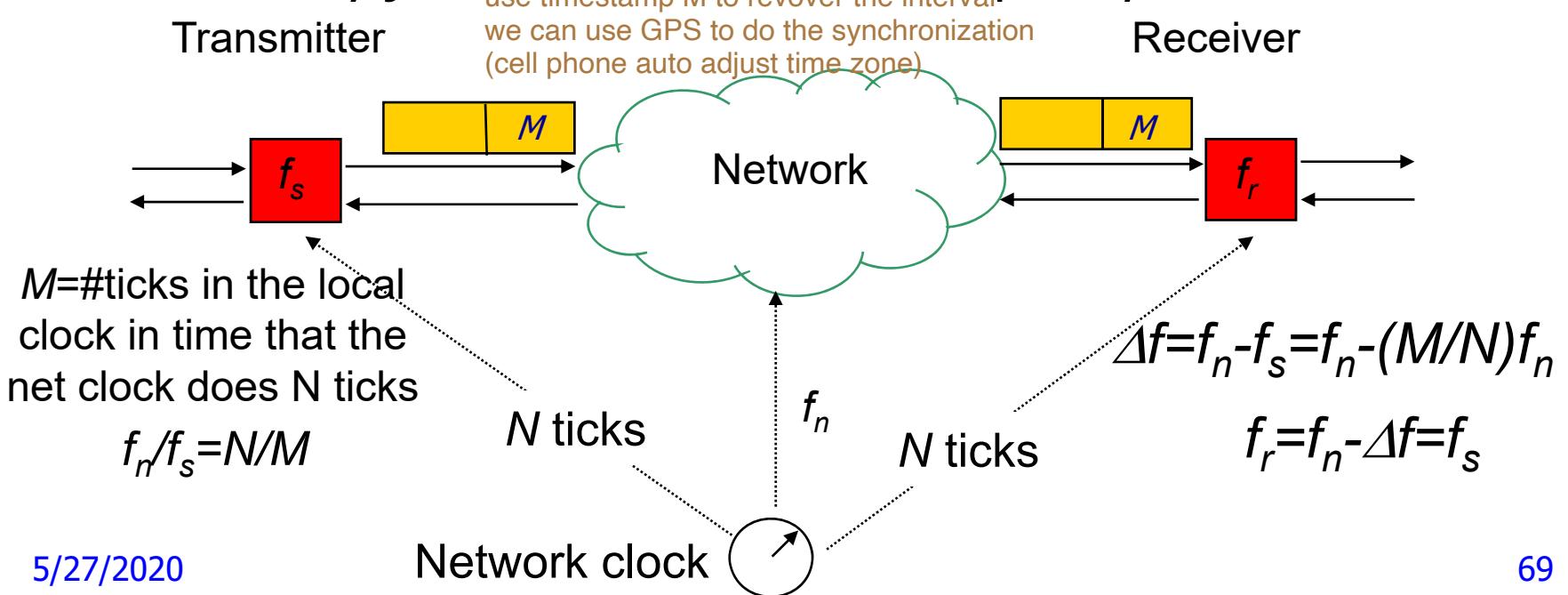
Timestamps inserted in
packet payloads
indicate when info was
produced



- 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

Scheme 3: Synchronization to a Common Clock

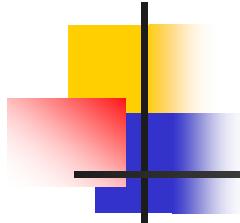
- Clock recovery simple if a common clock is available to transmitter & receiver
 - E.g. SONET network clock; Global Positioning System (GPS)
- Transmitter sends the value of M to the receiver
 - N is fixed, but M is variable according to changing time interval between packets
- Receiver derives Δf and then adjusts network frequency
- Packet delay jitter can be removed completely



Comments on the Playout Clock Frequency at the Receiver

not discussed

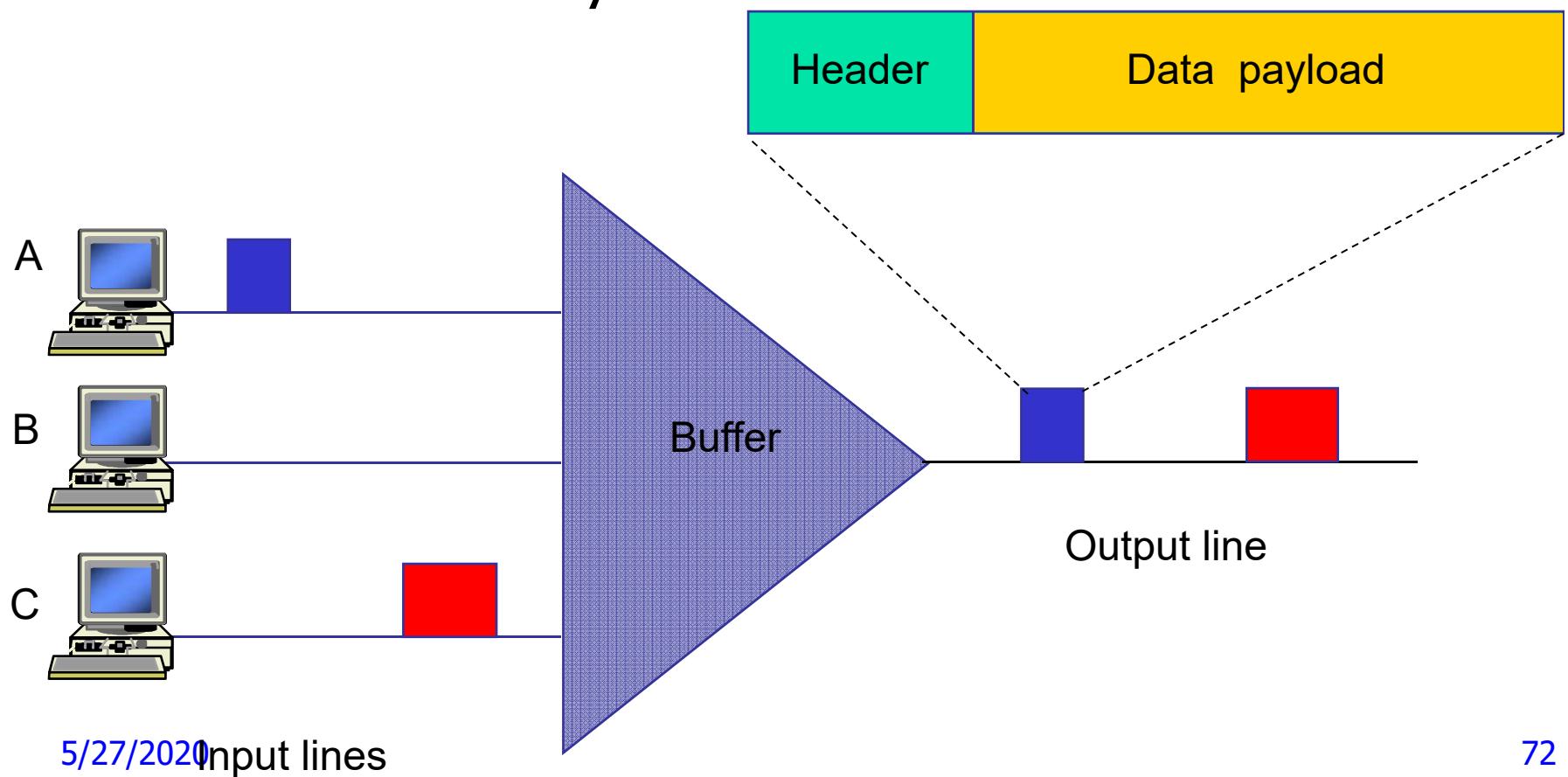
- The playout frequency cannot rely on number of packets in the receiving buffer
 - The variation in buffer length is impacted by network performance (more specifically delay jitter of packets)
- Scheme 1
 - Playout frequency is set according to a predetermined value, which is not adaptive to delay jitter (lead to fast or slow playout)
 - Not good for variable time intervals in original packets; not for interactive
- Scheme 2
 - Rely on timestamp in each packet to derive an adaptive playout frequency
 - E.g., if the next packet playout time is smaller than the timestamp, then decrease frequency; otherwise, increase frequency.
 - Adaptive to variable time intervals (jitter still exists), but too much overhead
- Scheme 3
 - Rely on common clock to accurately find out the playout frequency; low overhead, no jitter



Link Layer Multiplexing

Statistical Multiplexing

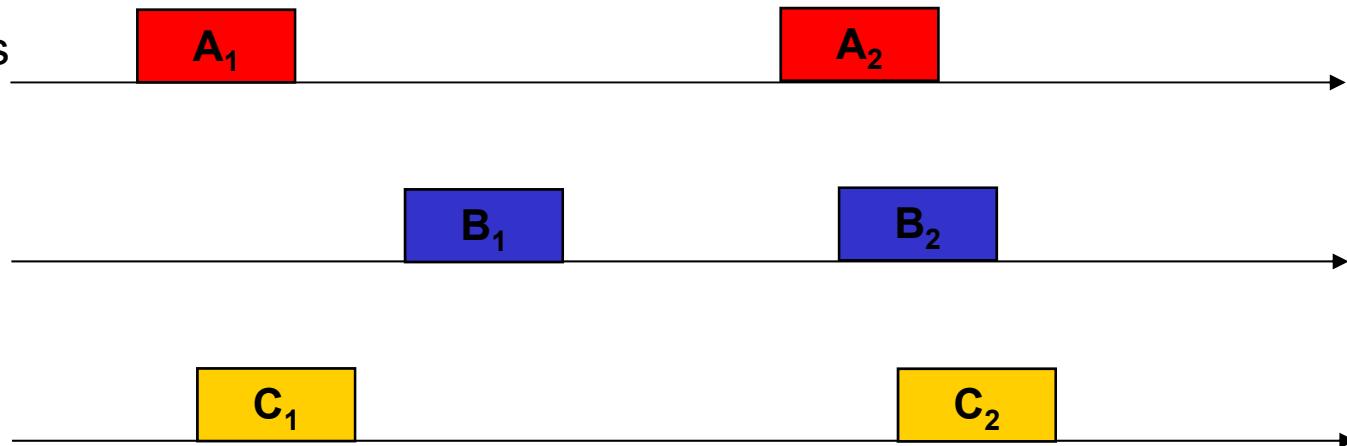
- Multiplexing concentrates bursty traffic onto a shared line
- Greater efficiency and lower cost



Tradeoff Delay for Efficiency

- Dedicated lines involve no 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

(a) Dedicated lines

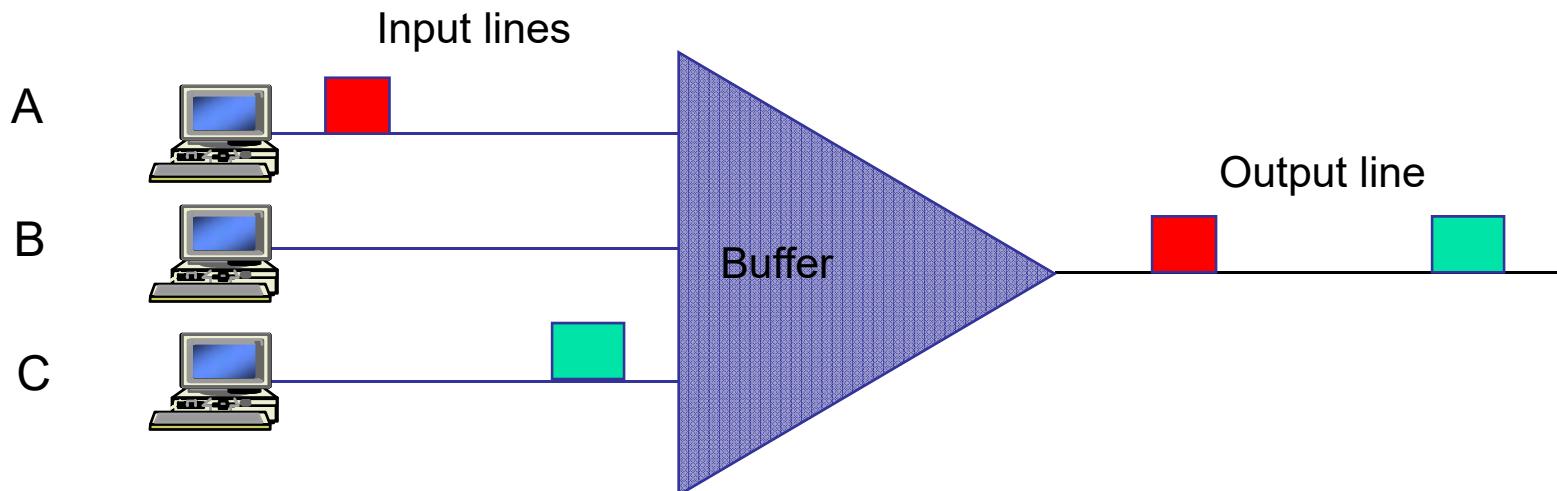


(b) Shared lines



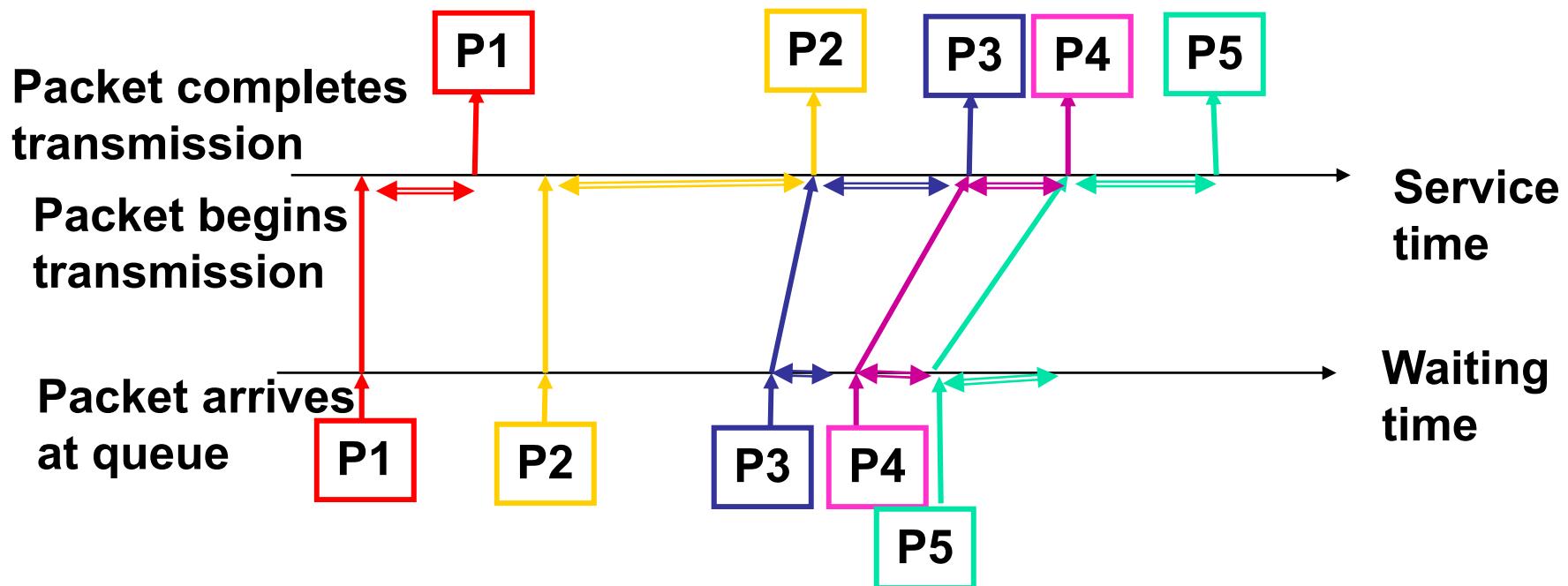
Multiplexer Modeling

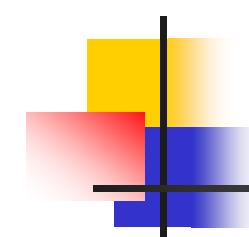
- Arrivals: What is the packet interarrival pattern?
 - Service Time: How long are the packets?
 - Service Discipline: What is order of transmission?
 - Buffer Discipline: If buffer is full, which packet is dropped?
-
- Performance Measures:
 - *Delay Distribution; Packet Loss Probability; Line Utilization*



Delay = Waiting + Service Times

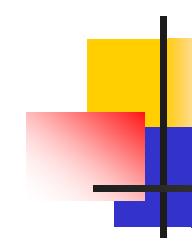
- Packets arrive and wait for service
- Waiting Time: from arrival instant to beginning of service
- Service Time: time to transmit packet
- Delay: total time in system = waiting time + service time





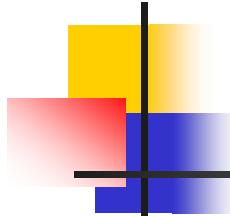
Security

- *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
- Examples: WEP, 802.11i



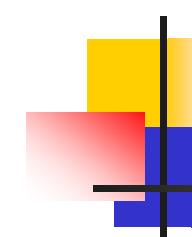
Link Layer Switching and Bridging

- To handle interconnections between different LANs
- *To be studied in next chapter: Chapter 4*



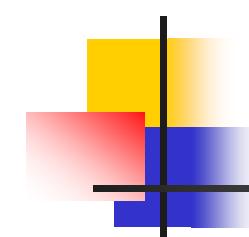
Examples of The Data Link Layer Protocols

framing
ARQ
flow control



Example 1: Point-to-Point Protocol (PPP)

- Data link protocol for point-to-point lines in Internet
 - Router-router; dial-up to router
- Provides *Framing and Error Detection*
 - Character-oriented HDLC-like frame structure
 - Normally HDLC is bit-oriented.
- *Capabilities (why do we need PPP)?*
 - *One Link Control Protocol*
 - *Support Multiple Network Control Protocols simultaneously*
- *Link Control Protocol*
 - Bringing up, testing, bringing down lines; negotiating options
 - E.g., configure/set up a high speed link from multiple low-speed physical links
 - **Authentication:** key capability in ISP access
- A family of *Network Control Protocols* specific to different network layer protocols
 - IP, OSI network layer, IPX (Novell), Appletalk

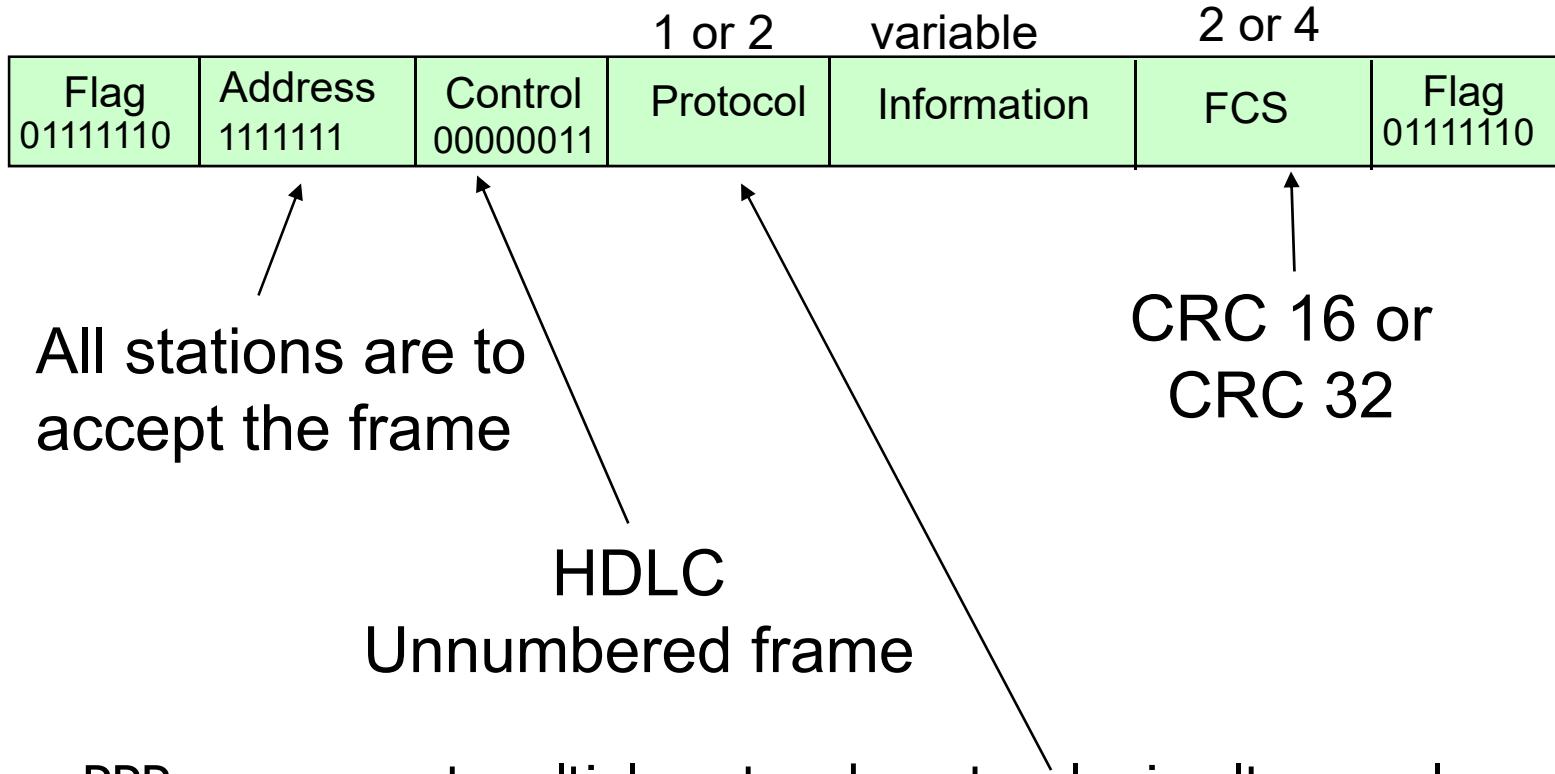


PPP Applications

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



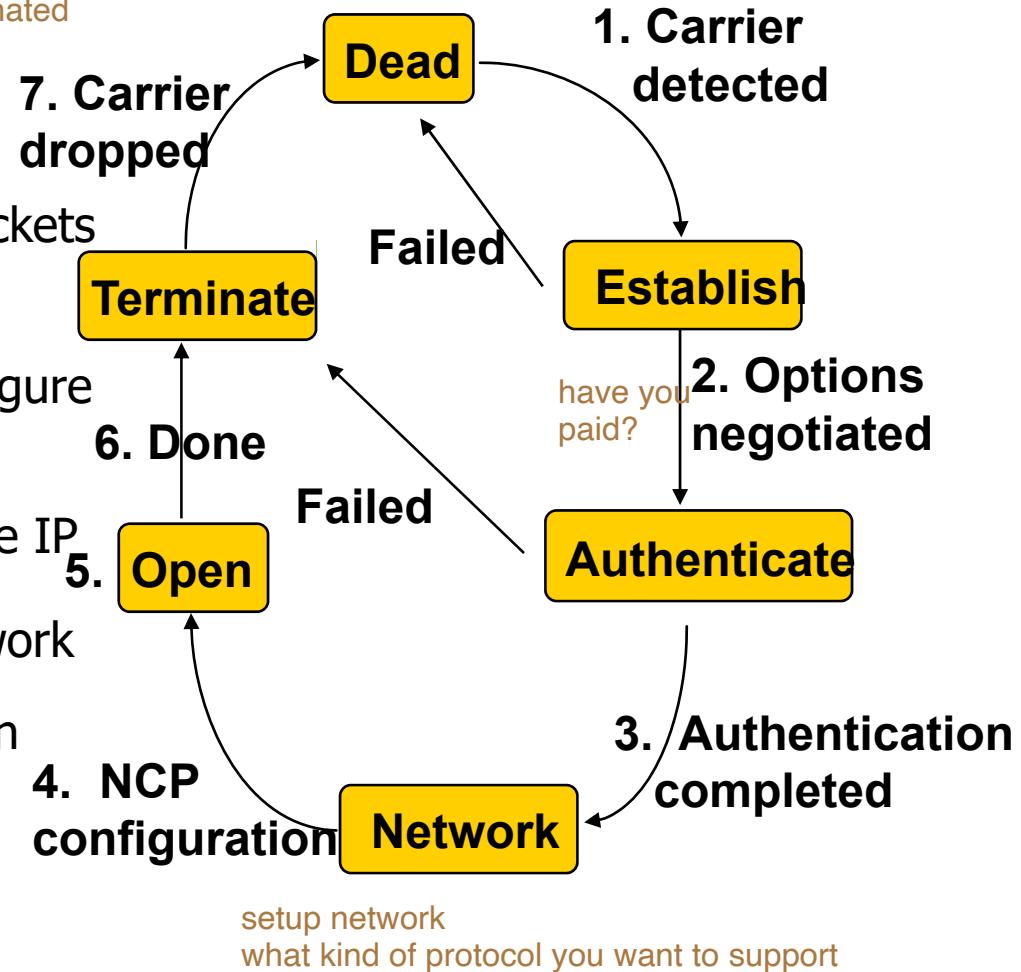
- PPP can support multiple network protocols simultaneously
- Specifies what kind of packet is contained in the payload
 - e.g. LCP, NCP, IP, OSI CLNP, IPX...

PPP Phases

at home, not terminated

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 是 setup link 的 procedure
maintain data link 的是 HDLC



PPP Authentication

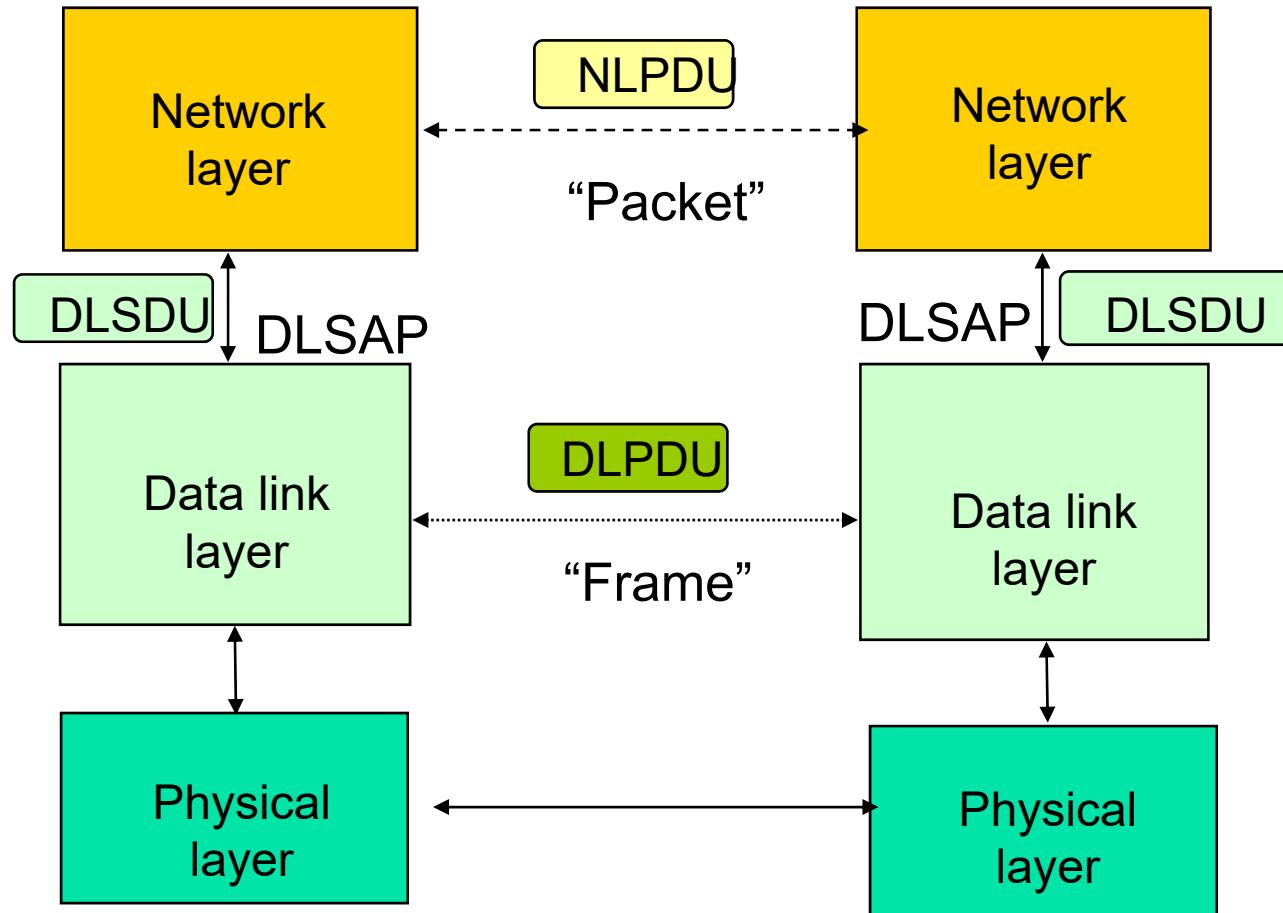
- Password Authentication Protocol
 - Initiator must send ID & password
 - Authenticator replies with authentication success/fail
 - After several attempts, LCP closes link
 - Transmitted unencrypted, susceptible to eavesdropping
- Challenge-Handshake Authentication Protocol (CHAP)
 - Initiator & authenticator 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 cryptographic checksum & compares to response
 - Authenticator can reissue challenge during session



Example 2: 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
- Services for network layer
 - Connection-oriented
 - Connectionless, acknowledged
 - Connectionless, unacknowledged

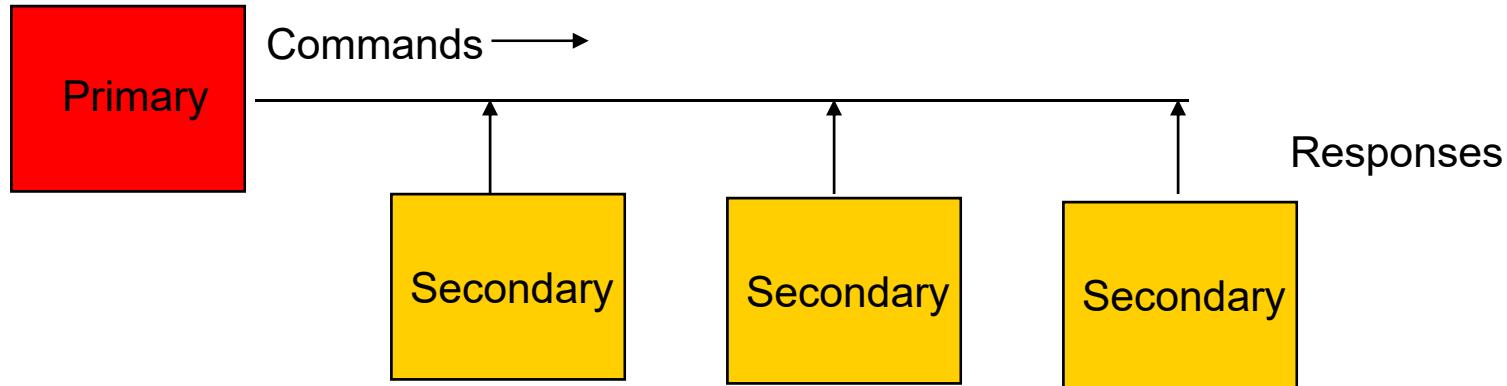
Protocol Stack



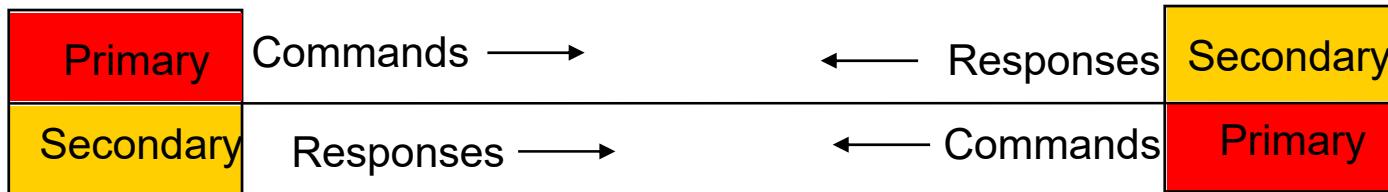
HDLC Data Transfer Modes

Normal Response Mode (NRM)

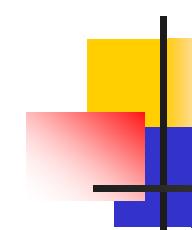
- Used in point-to-point link or polling multidrop lines



- Asynchronous Balanced Mode (ABM)
 - Used in full-duplex point-to-point links



- Mode is selected during connection establishment



HDLC Frame Format

- Control field gives HDLC its functionality
- Codes in fields have specific meanings
 - 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)

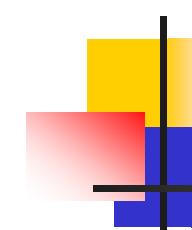
Supervisory Frame see P91

1	0	S	S	P/F	N(R)
---	---	---	---	-----	------

Unnumbered Frame

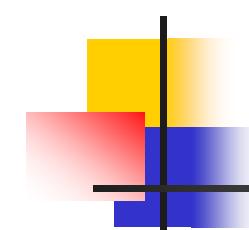
1	1	M	M	P/F	M	M	M
---	---	---	---	-----	---	---	---

- **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
 - $N(R) = \text{Sequence number of } next \text{ frame expected}$ acknowledges all frames up to and including $N(R)-1$
- 3 or 7 bit sequence numbering
 - Maximum window sizes 7 or 127 (extended case: control field = 16 bits) (for Go back N ARQ)
- 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



HDLC Error Detection & Loss Recovery

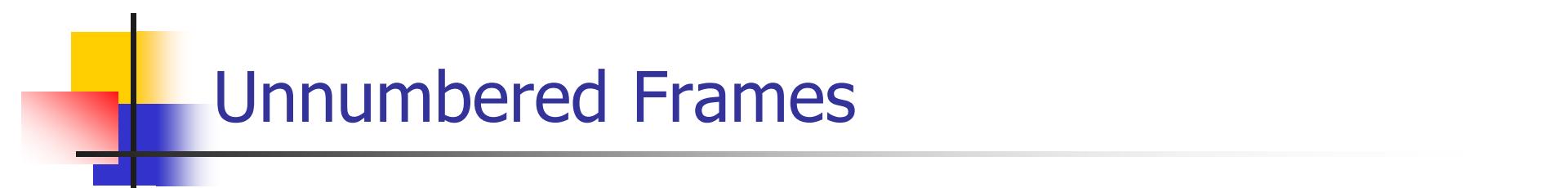
- Frames lost due to loss-of-synch or receiver buffer overflow
- Frames may undergo errors in transmission
- CRCs detect errors and such frames are treated as lost
- Recovery through ACKs, timeouts & retransmission
- Sequence numbering to identify out-of-sequence & duplicate frames
- HDLC provides for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

Supervisory frames

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

- *Receive Ready (RR)*, SS=00 similar to sending ACK wait and
 - ACKs frames up to $N(R)-1$ when piggyback not available
- *REJECT (REJ)*, SS=01 similar to sending NACK to go back n
 - Negative ACK indicating $N(R)$ is the first frame not received correctly. Transmitter must resend $N(R)$ and later frames
 - Could be faster than timeout for go back N, but timeout is still needed
- *Receive Not Ready (RNR)*, SS=10 for flow control
 - ACKs frame $N(R)-1$ & requests that no more I-frames be sent
- *Selective REJECT (SREJ)*, SS=11 selective
 - Negative ACK for $N(R)$ requesting that $N(R)$ be selectively retransmitted just resend the specified frame

flow control and ARQ implemented by the same protocol



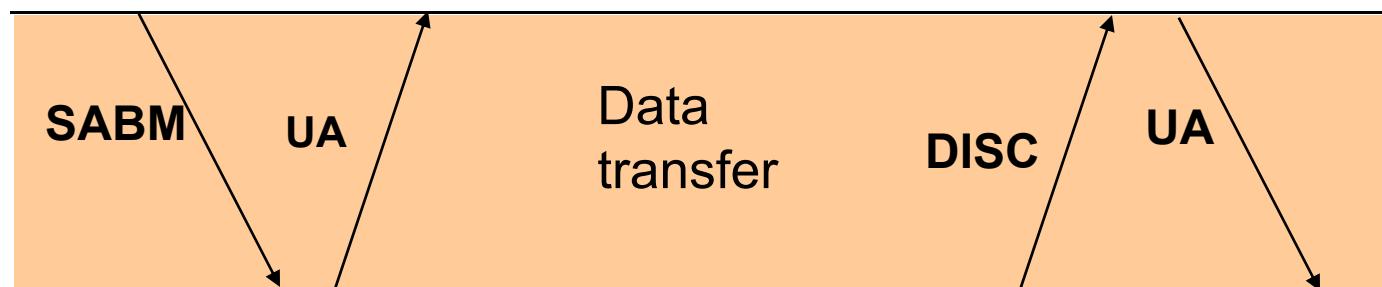
Unnumbered Frames

Multiple control functions done by unnumbered frames

- Setting transfer modes:
 - SABM: Set Asynchronous Balanced Mode
 - UA (Unnumbered Acknowledgement): acknowledges acceptance of mode setting commands
 - DISC (Disconnect): terminates logical link connection
 - ...
- 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 (example of one mode)
 - Set Asynchronous Balanced Mode (SABM)
 - Disconnect (DISC)
 - Unnumbered Acknowledgment (UA)



HDLC using NRM (polling) – Unbalanced Mode

(Mode has been set up by unnumbered frames)

non response mode

different frames used in this

A polls B

Address of secondary

A rejects fr1

A polls C

A polls B,
requests
selective
retrans. fr1

A send info fr0
to B, ACKs up to 4

5/27/2020

Primary A

B, RR, 0, P

polling P

N(R)

B, SREJ, 1

C, RR, 0, P

B, SREJ, 1, P

B, I, 0, 5

Time

RR means ACK

X lost

Secondaries B, C

information frame

N(S) N(R)

B, I, 0, 0

B, I, 1, 0

B, I, 2, 0, F

B sends 3 info

frames not stop and wait

Final bit

B does not respond

C, RR, 0, F

C nothing to
send

B, I, 1, 0

B, I, 3, 0

B, I, 4, 0, F

B resends fr1
Then fr 3 & 4

I for information package (previously only control msgs were sent)

0 is for N(S) (the first information package requested by A)
already received 4 from B, so expecting 5

Frame Exchange using ABM

(Mode has been set up by unnumbered frames)

Async (2 processes can cross over)

A and B are equivalent so-called combined

(when A sends msg to B, A is the primary, vice versa)

B checks that whether packages from A is in sequence
otherwise, package loss and send control frame

Combined Station A

B, I, 0, 0

B, I, 1, 0

B, I, 2, 1

B, I, 3, 2

B, I, 4, 3

B, I, 1, 3

B, I, 2, 4

B, I, 3, 4

Combined Station B

A, I, 0, 0

A, I, 1, 1

A, I, 2, 1

B, REJ, 1

AACKs fr0

A rejects
fr1

you have to repeat sending frame 1

A, I, 3, 1

B, RR, 2

AACKs fr1

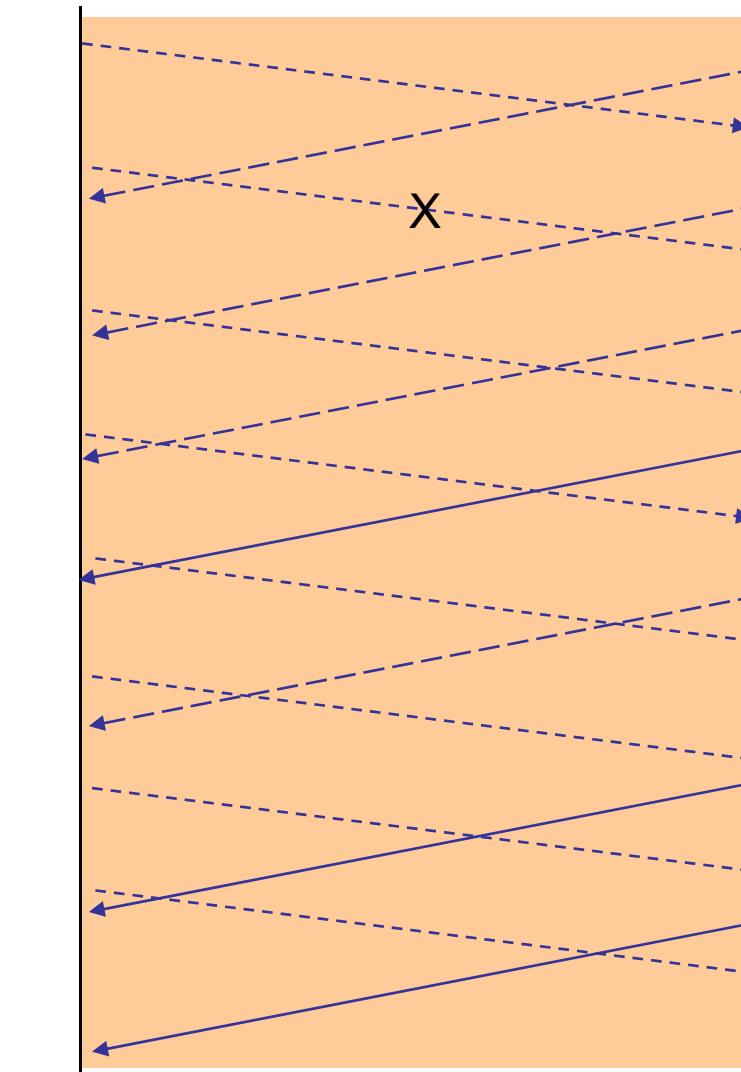
use control frame to confirm that
it does not have more info frame to send

B, RR, 3

AACKs fr2

A
B sends 5
frames

A
B goes
go back n
back to 1



HDLC 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)

