Data Link Control Protocols

Dr Steve Gordon
ICT, SIIT

Contents

- Flow Control
  - Stop-and-wait
  - Sliding Window
- Error Control
  - Stop-and-wait ARQ
  - Go-Back-N ARQ
  - Selective-Reject ARQ
- Example Protocols

Concept

Concept

Technology

Data Link Control Protocols

- Need layer of logic above Physical Layer
  - Physical layer concentrates on sending signals over transmission link
  - More control and management is needed to send data over data communications link
    - Frame synchronization: start and end of each frame
    - Flow control: ensure sender does not send too fast for receiver
    - Error control: correct bit errors introduced by transmission system
    - Addressing: must specify identity of two stations communicating
    - Control and data: receiver must distinguish between control and data information
    - Link management: setup and maintain the link
  - Hence, data link layer (and data link control protocols)
  - We will focus on Flow Control and Error Control

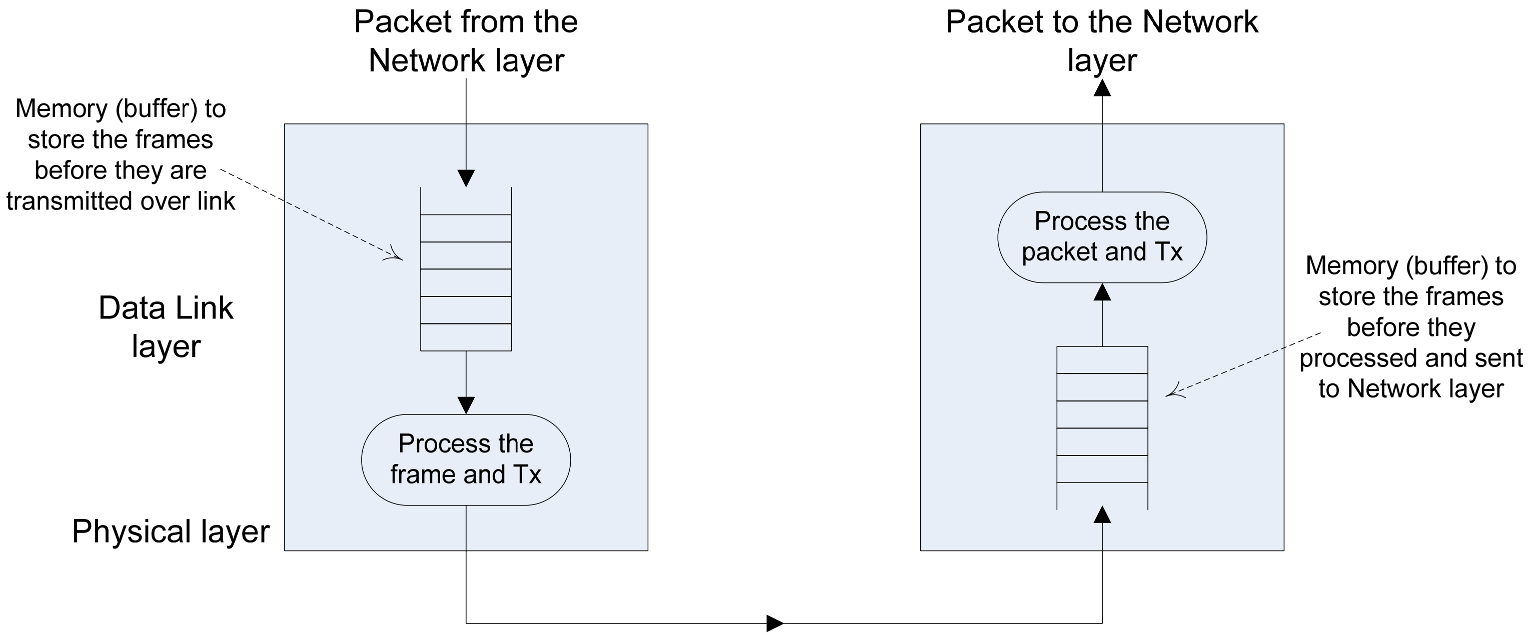
Flow Control

Stop-and-Wait

Sliding Window

Flow Control

- Receivers typically have a finite amount of memory (buffer space) to store received data before processing

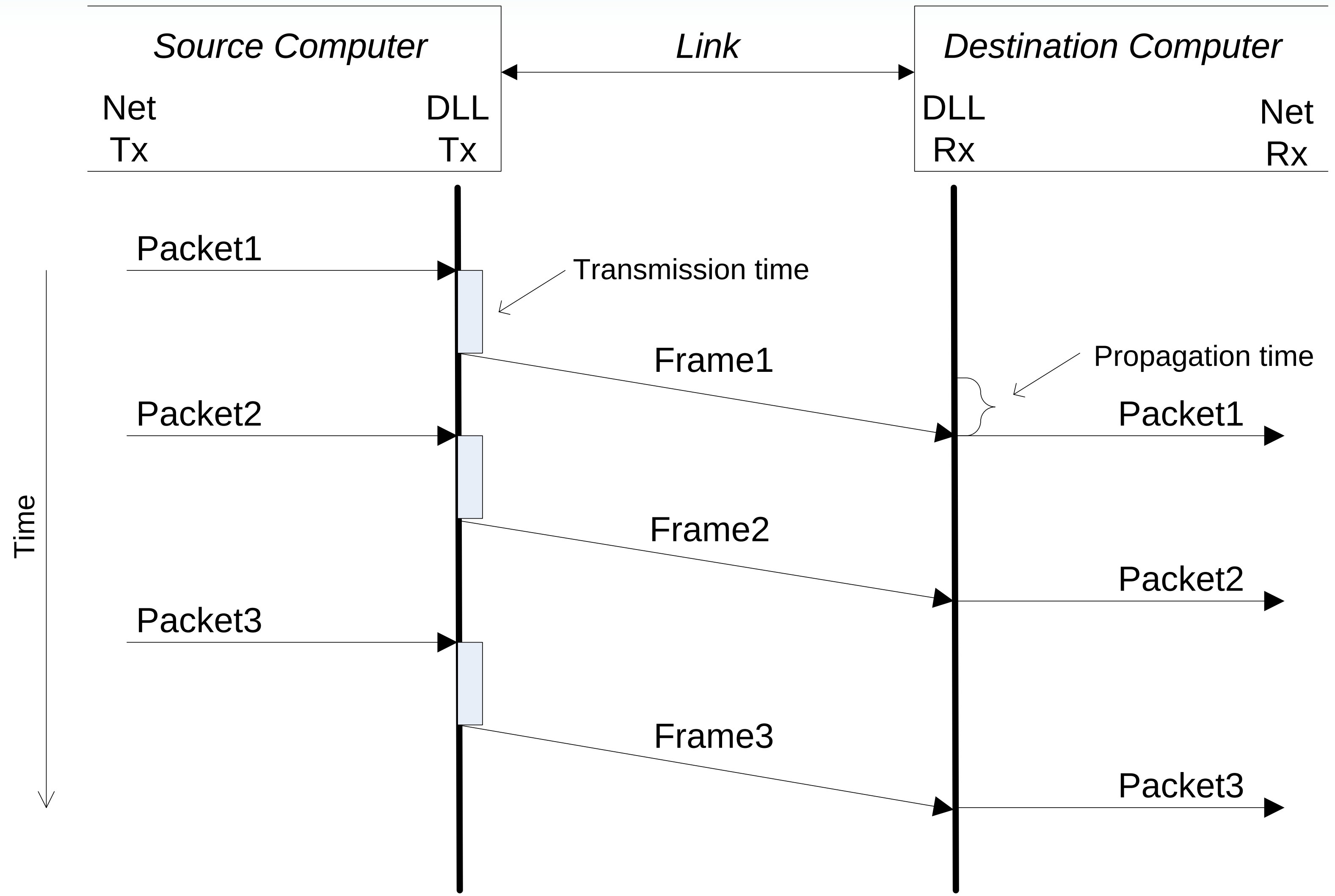


Flow Control

- Flow control aims to ensure sending entity does not overwhelm receiving entity
  - If sender sends too fast for receiver, then buffer may overflow
    - Result of buffer overflow: data is lost, possibly need to retransmit, which reduces performance
  - Flow control tries to prevent buffer overflow
  - Flow control is influenced by:
    - Transmission time
    - Propagation time
- Assumes there are no errors but varying delays

- Note: flow control is used at the data link layer to control the sending of data over a link. But it is also used at other layers, especially transport layer, to control sending of data over a network. Similar concepts and protocol mechanisms are used.

Error-Free Frame Transmission

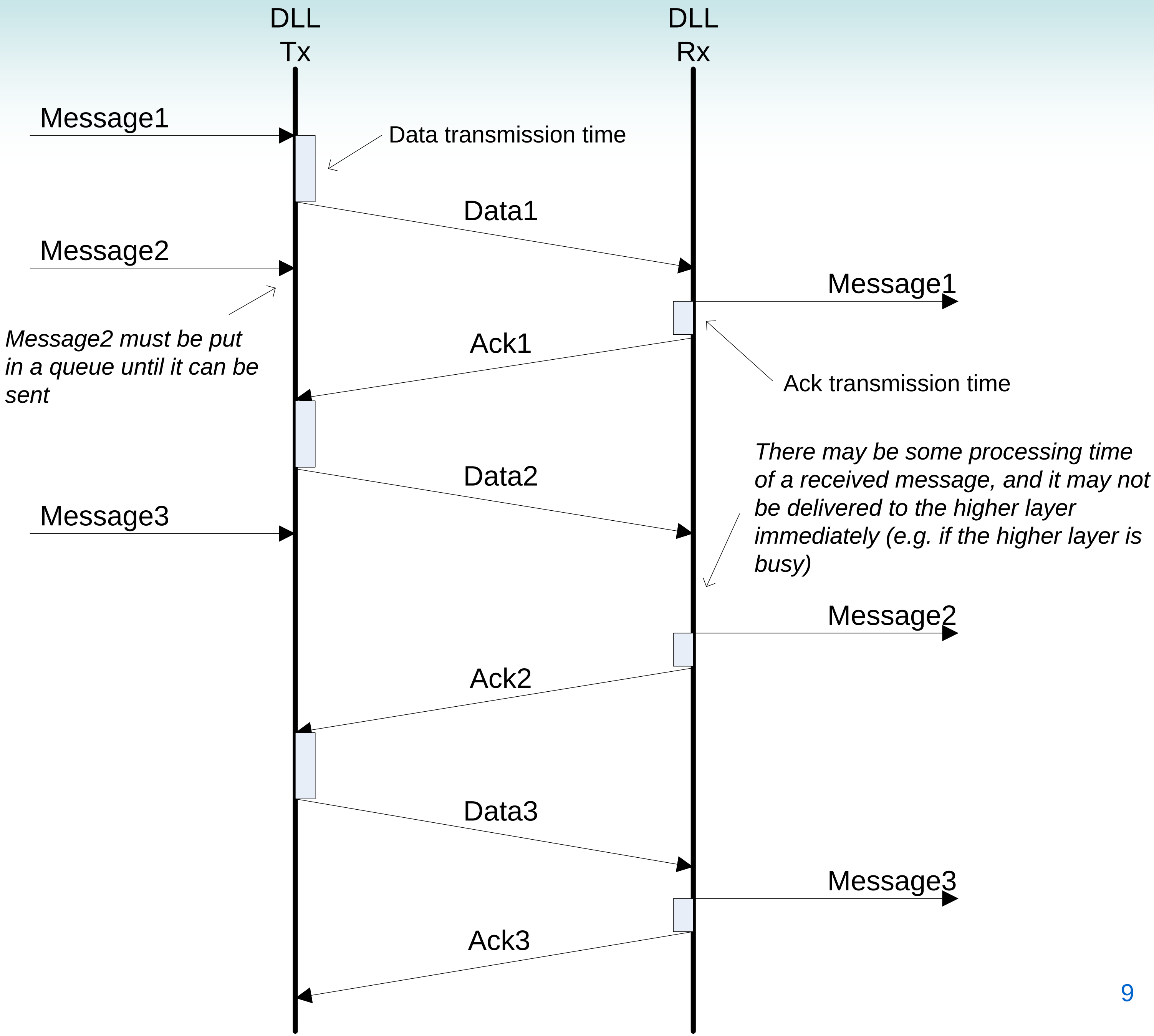


We may call the *Packet* a *Message*

Stop and Wait Flow Control

- Stop and Wait Flow Control Protocol:
  - Frame types:
    - Data: contains the information to be sent
    - ACKnowledgment: acknowledges receipt of data
  - Rules:
    - Source transmits a frame
      - Source waits for ACK before sending next frame
    - Destination receives frame and replies with an ACK
    - Destination can stop the flow of data by not sending ACK

Stop and Wait Flow Control

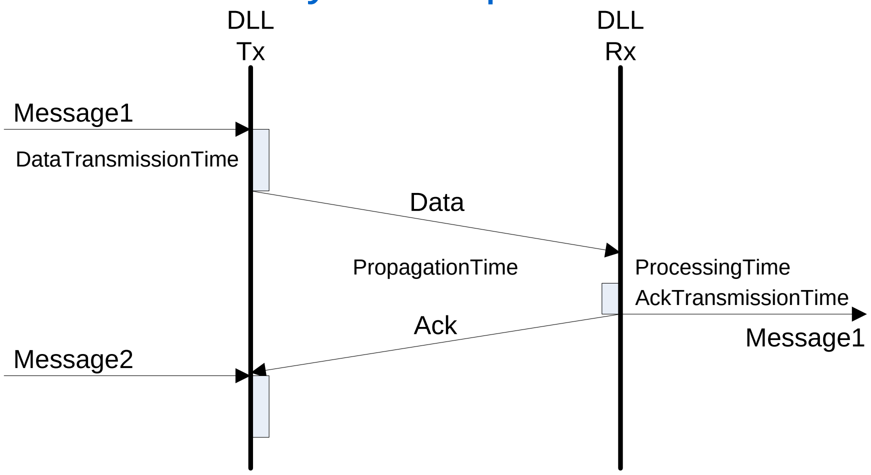


Example: Stop and Wait Performance

- Source Network layer has three 1000 bytes messages to be sent immediately
- When the Destination DLL receives a message, it is sent (after 1μs processing delay) to the Network layer (and an ACK is sent)
- A Data frame contains the message plus 20 byte header
- An ACK frame is 20 bytes
- Link:
  - Data rate: 1Mb/s
  - Distance: 2km
  - Velocity: 2×10^8 m/s

Stop and Wait Efficiency

- Flow control (and any useful communications protocol) introduces overheads:
  - Time to send headers, time to send control information (ACKs), time to wait for an ACK, …
- One measure of performance is throughput
  - Rate at which useful data is received at the destination
  - A related measure is *efficiency* (or utilisation) of a link:
    - What portion of the link is used to send useful data
  - Example: if a link has a data rate of 1Mb/s, but the maximum throughput using a protocol on that link is 800kb/s, then the protocol is 80% efficient
- What is the best case efficiency for Stop and Wait Flow Control?



Stop and Wait Efficiency

- Lets assume:
  - Processing Time is 0 (since it is typically very small compared to transmission and propagation times)
  - AckTransmissionTime is 0 (if ACK is 20 bytes and DATA 1000 bytes, then AckTransmissionTime will be very small compared to DataTransmissionTime)

$$Eff = \frac{DataTransmissionTime}{DataTransmissionTime + AckTransmissionTime + 2 \times PropagationTime}$$

$$= \frac{DataTransmissionTime}{DataTransmissionTime + 2 \times PropagationTime}$$

$$= \frac{1}{1 + 2 \times \frac{p}{d}}$$

  - where p = PropagationTime and d = DataTransmissionTime
- Conclusions:
  - Efficient when DataTransmissionTime is much larger than PropagationTime
  - Inefficient for links with: high data rate, large distance between sender and receiver, small data packets

An Aside: What is a good packet size?

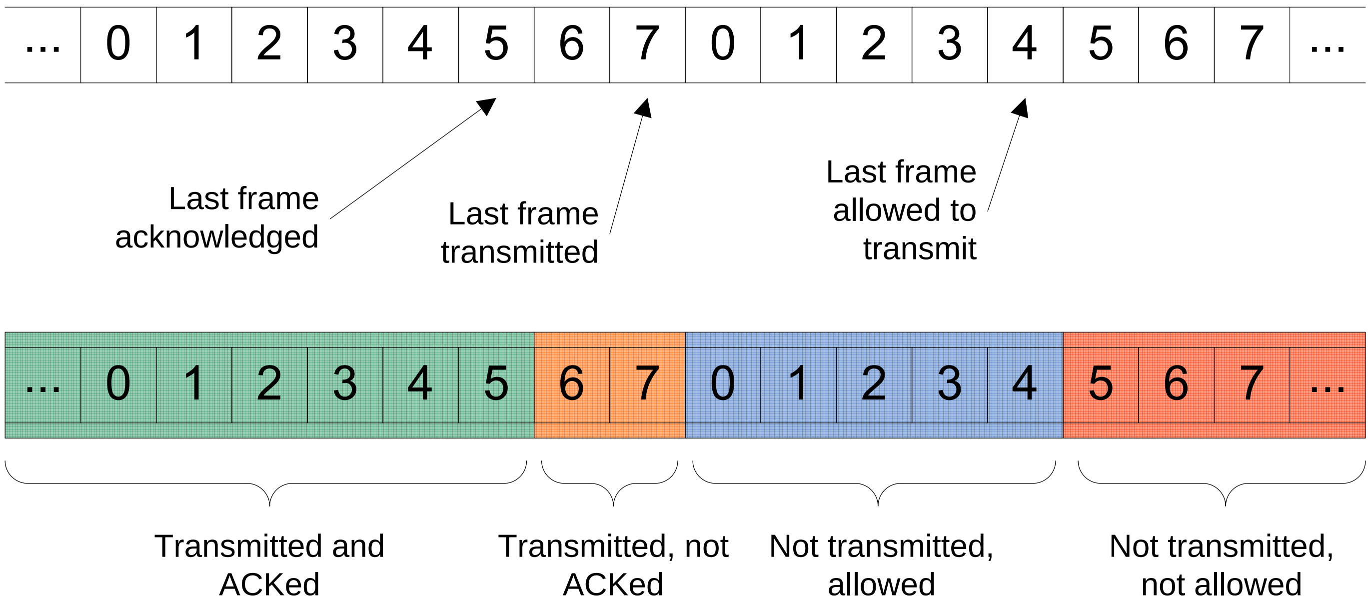
- Protocols will often break data into smaller packets. Why use smaller packets (instead of one large packet)?
  - Buffer sizes of receivers may be limited
    - E.g. if a buffer at the receiver is 4000 bytes, and the sender has 5000 bytes to send:
      - If one larger 5000 byte packet, then the receiver cannot receive/buffer the message
      - If several small 1000 byte packets, then the first 4 packets can be received, processed and then the last packet received and processed
  - Higher overhead (due to retransmission) if errors in large packets
    - E.g. if 1 bit error every 5000 bytes
      - If one larger 5000 byte packet, then may contain an error, and entire 5000 bytes must be resent
      - If several small 1000 byte packets, then 1 of the 5 packets may contain an error, and only that 1 packet has to be resent
  - Smaller packets can make shared access fair amongst users
    - E.g. if a shared LAN has 11 users that take turns transmitting at 400kb/s
      - If one larger 5000 byte packet, then takes 100ms to send. Each user must wait 1sec before the get an opportunity to send
      - If several small 1000 byte packets, then each user waits 200ms before their next opportunity to send
- The best packet size depends on overheads, and desired throughput and delay performance
  - Smaller packets means the header and ACKs contribute a larger overhead

Sliding Window Flow Control

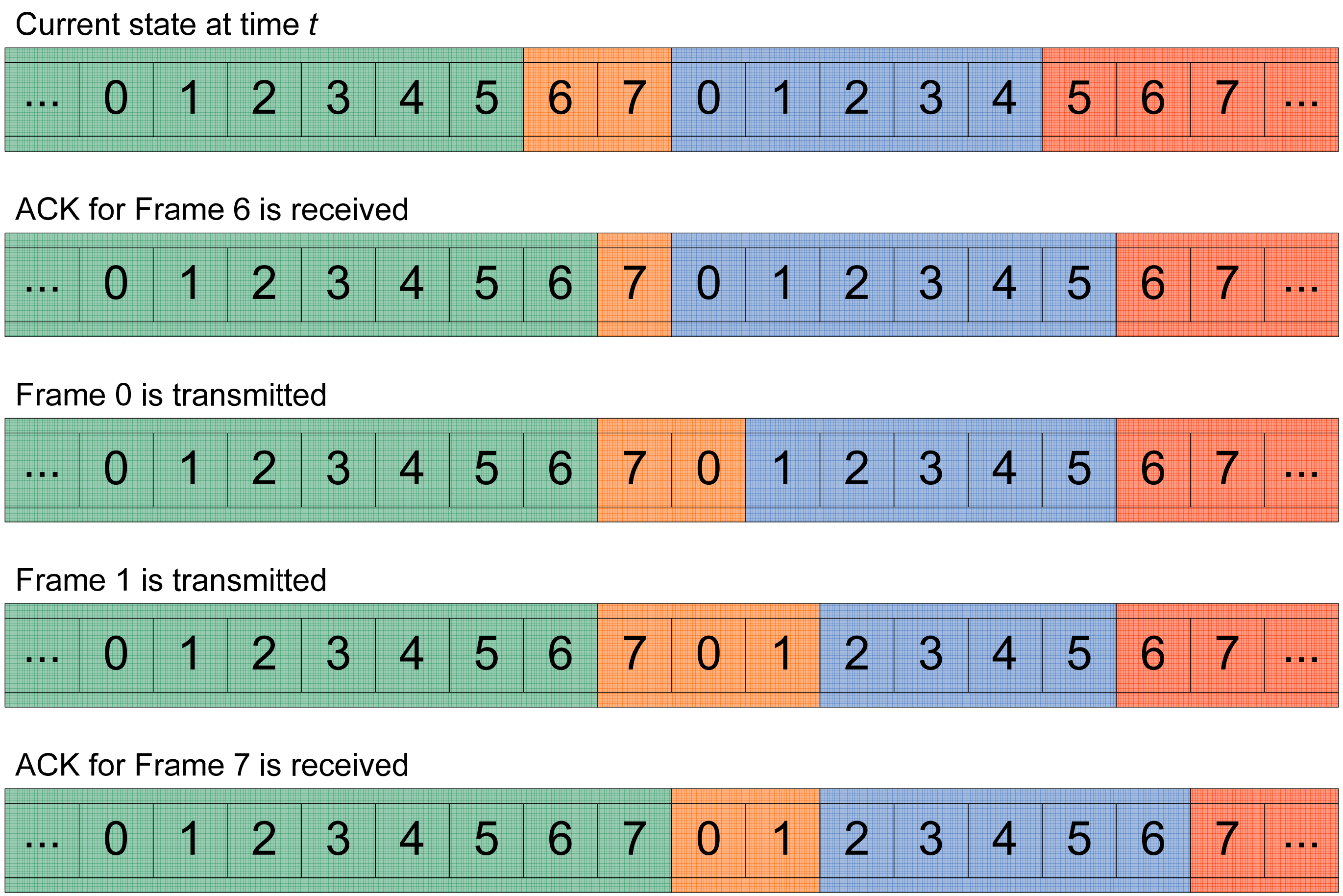
- Stop and Wait only allows 1 frame to be in transit (between sender and receiver)
- Sliding Window allows *multiple* numbered frames to be in transit
  - Each frame has a sequence number
    - Used to keep track of the frames that have been sent and acknowledged
    - Sequence number is carried in a header (therefore limited in size)
    - A *k* bit sequence number; frames are numbered: 0, 1, 2, …, 2^k-1, 0, 1, 2, …
  - Receiver has buffer for W frames
    - W refers to “window”
    - Maximum size of window, W, is 2^k -1 (reason discussed later)
  - Transmitter sends up to W frames without receiving ACK
    - ACK from receiver includes sequence number of the next frame expected
    - Receiver can ACK frames without permitting further transmission
      - Send a special ACK (Not Ready)
      - Must send a normal ACK to resume
  - If have full-duplex link, can piggyback ACKs on data
    - Piggyback = send the acknowledgement information with the data

Sliding Window: Source

- Source uses 3 variables to keeps track of:
  - Frames that have been transmitted and acknowledged
  - Frames that have been transmitted, but not acknowledged
  - Frames that have not been transmitted, but we are allowed to send
  - Frames that have not been transmitted, and we are not allowed to send

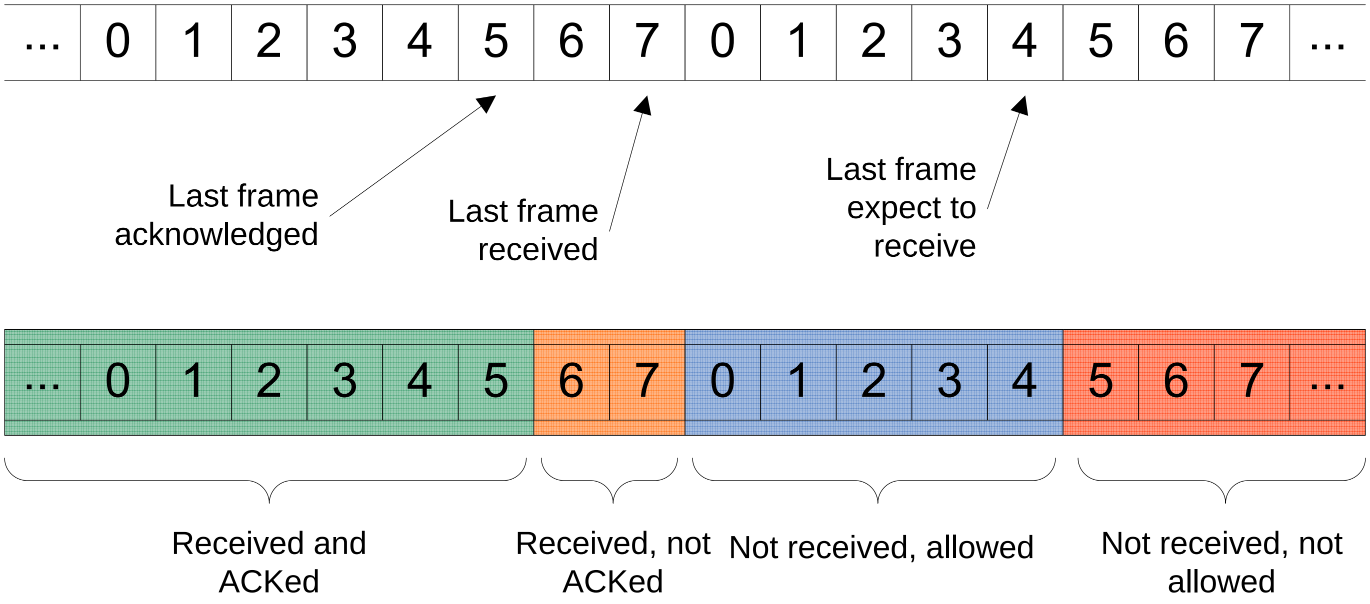


Sliding Window: Source Example

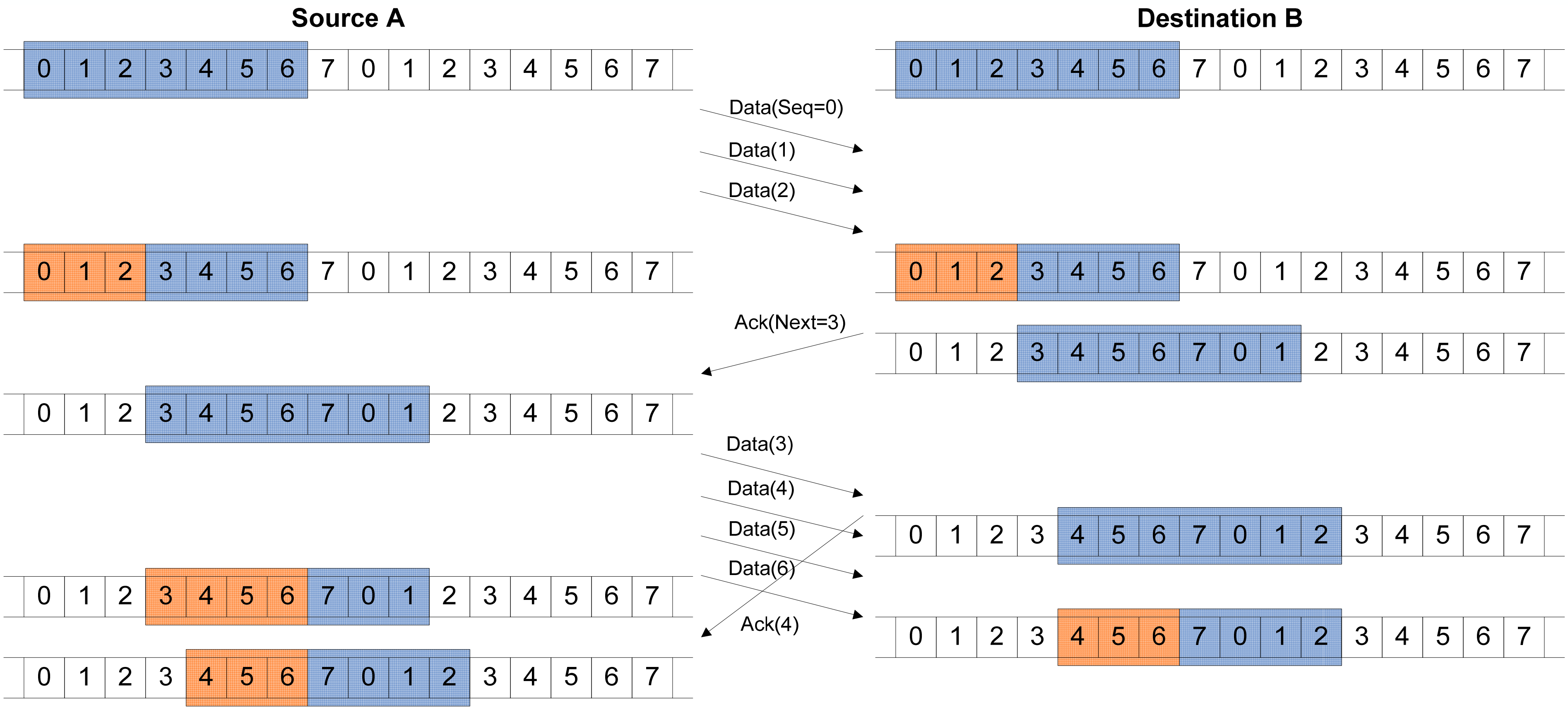


Sliding Window: Destination

- Destination uses 3 variables to keeps track of:
  - Frames that have been received and acknowledged
  - Frames that have been received, but not acknowledged
  - Frames that have not been received, but we expect (or allowed) to receive
  - Frames that have not been received, and we don't expect (or not allowed) to receive



Sliding Window Example



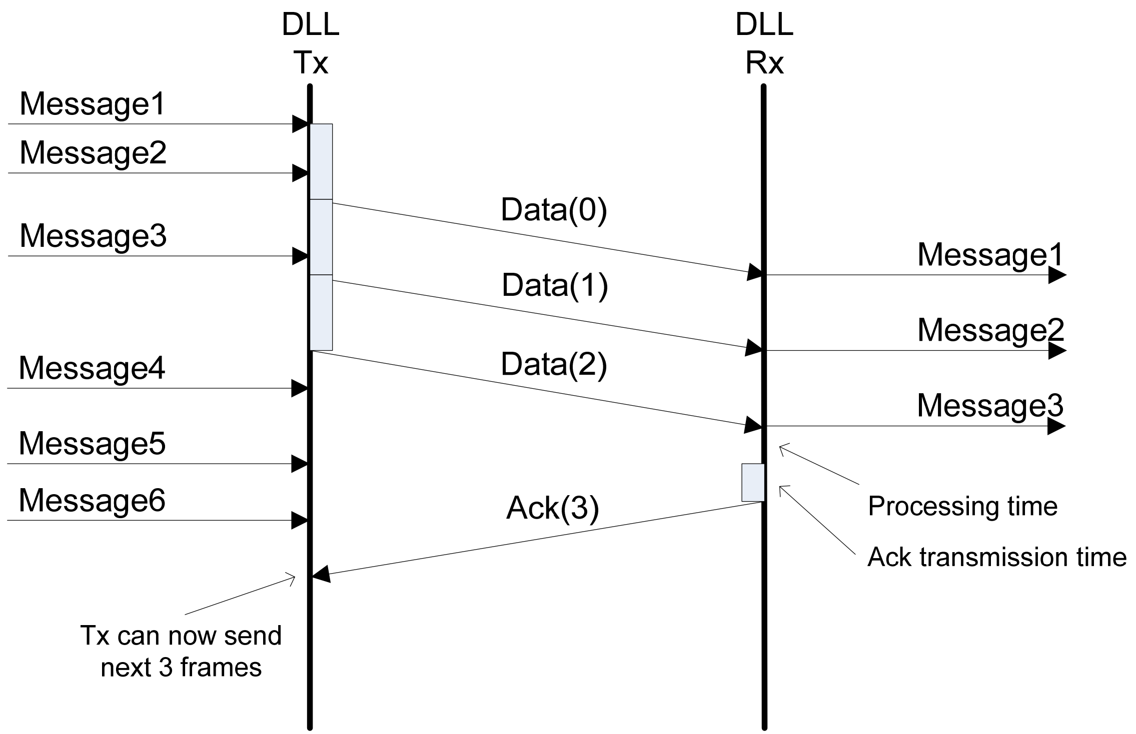
Sliding Window Efficiency

- Lets assume:
  - Processing time and ACK transmission time is 0
  - Receiver sends an ACK after receiving W frames
    - This is not mandatory in Sliding Window; the receiver can choose when to send an ACK

$$Eff = \frac{W \times DataTransmissionTime}{W \times (DataTranssmisionTime) + 2 \times PropagationTime}$$

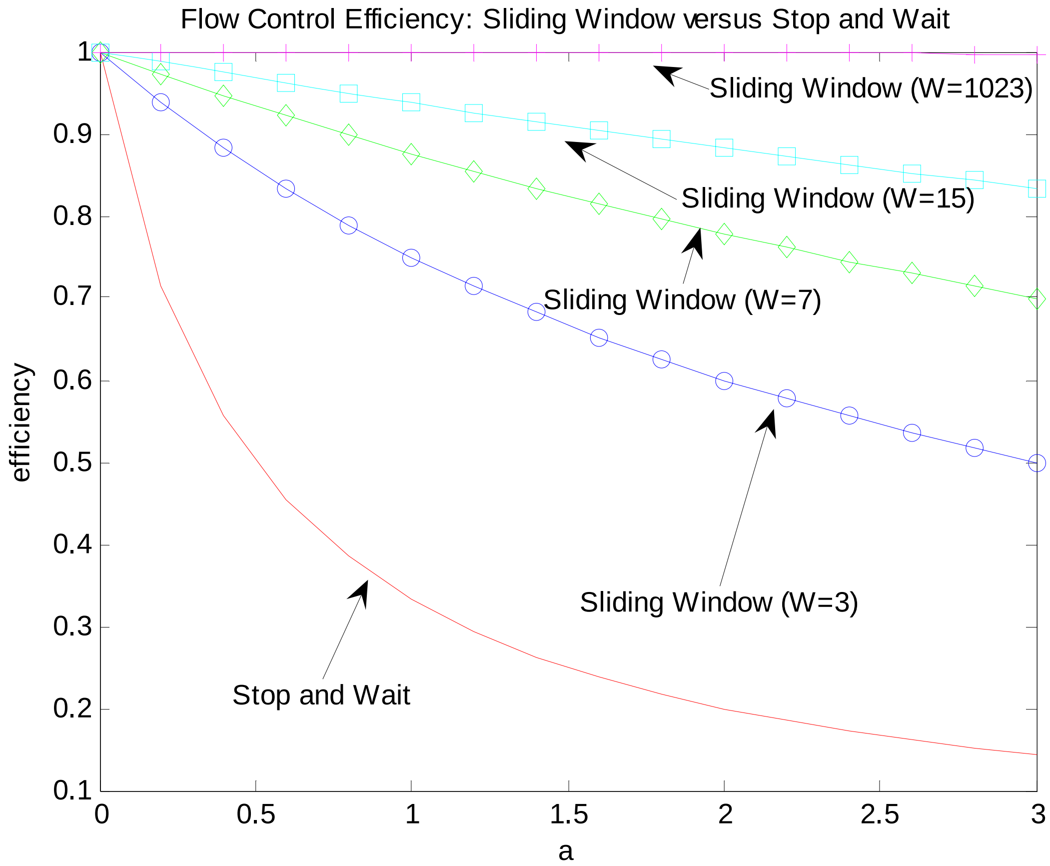
$$= \frac{Wd}{Wd + 2p}$$

$$= \frac{W}{W + 2\,{}^{p}\!/_{d}}$$



Flow Control Efficiency Comparison

- In the plot, a = p/d
  - E.g. if a >1 then propagation time is larger than Data transmission time
- Stop and Wait is equivalent to Sliding Window with W = 1
- Increasing the maximum window size W gives a better efficiency
- Sliding window is more complex; the larger W, the larger buffer needed at receiver



Error Control

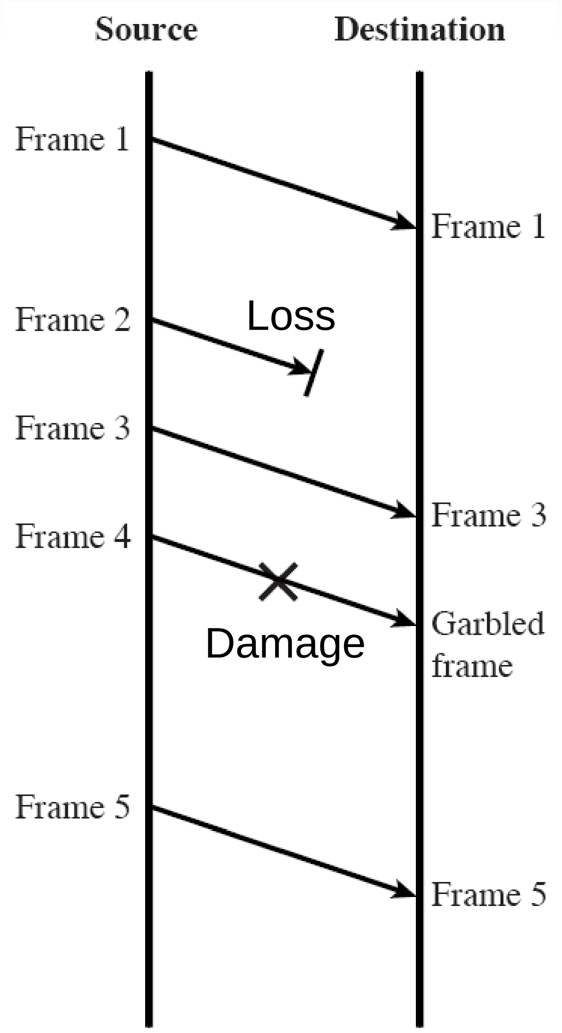
Stop-and-Wait

Go-Back-N

Selective-Reject

Error Control

- We need to detect and correct of errors such as:
  - Lost frames (frame not received)
  - Damaged frames (frame received with errors)
- Common techniques used:
  - Error detection and FEC (discussed in previous topic)
  - Positive acknowledgment: destination returns a positive ACK after successfully receiving error-free frames
  - Retransmission after timeout: source retransmits a frame that has not been ACKed after predetermined time
  - Negative acknowledgement and retransmission: destination returns negative ACK for frames in which an error is detected
- Together, the last 3 techniques are called *automatic repeat request* (ARQ). Three versions:
  - Stop-and-wait ARQ
  - Go-back-N ARQ
  - Selective-reject ARQ

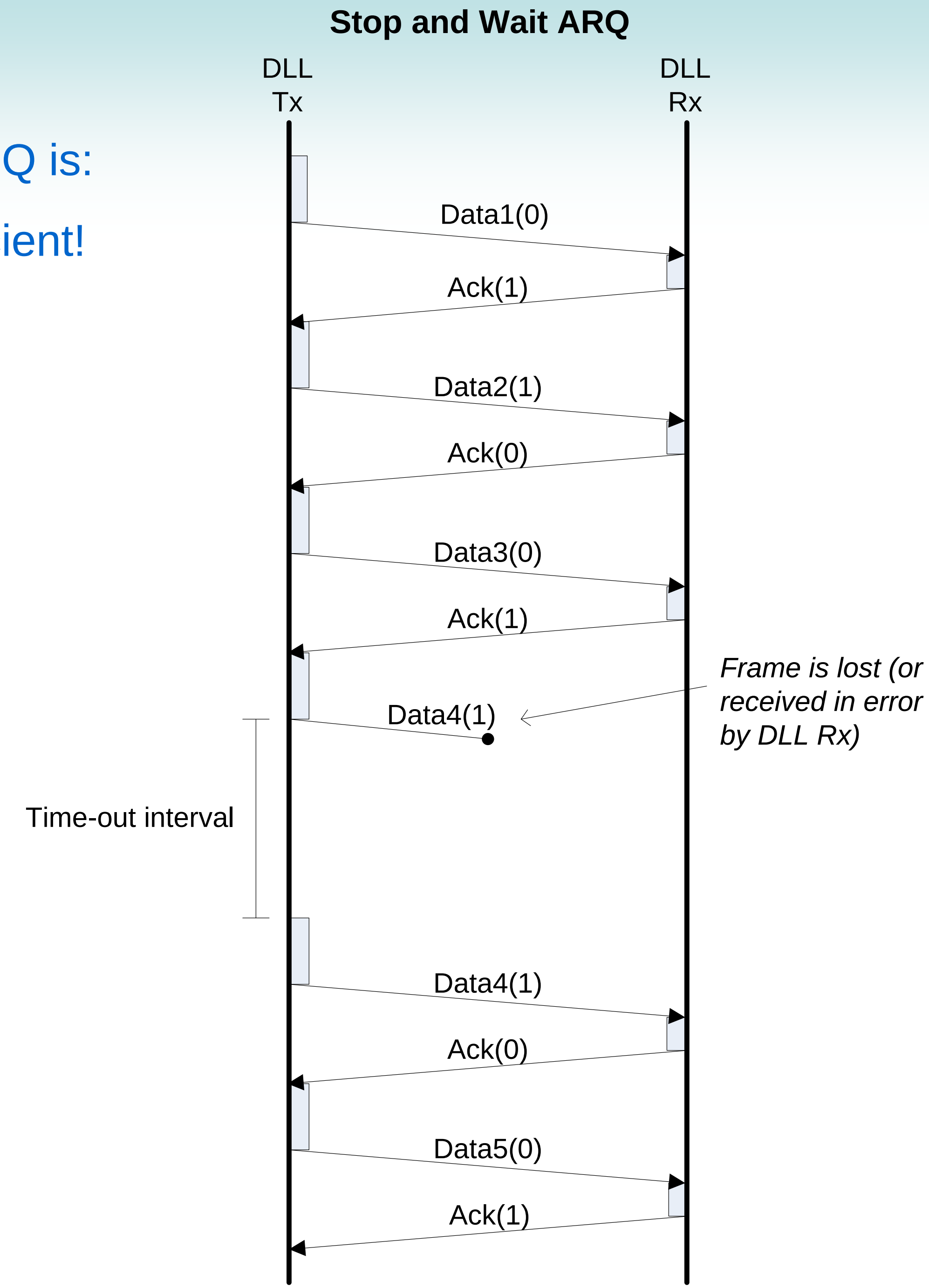


Stop-and-Wait ARQ

- Based on Stop-and-Wait flow control:
  - Source:
    - Transmit a single frame, and start timer for that frame
    - If ACK received, then stop timer and transmit next frame
    - If no ACK received before the timer expires (timeout), then retransmit the frame
  - Destination:
    - If frame received (with no errors), then send an ACK
    - If damaged frame received (containing errors), then discard the frame
- But what if a damaged ACK is received by source:
  - Source will not recognize the damaged ACK and will retransmit
  - Destination will receive two copies of the frame – this is not good!
  - Hence, source labels frames with 0 or 1 (a 1-bit sequence number)
    - Destination includes the Next sequence number expected in ACK
    - E.g. if received Data(0), send ACK(1)
  - This allows the destination to identify if two copies of a frame a received (and therefore not deliver the duplicate frame to the higher layer)

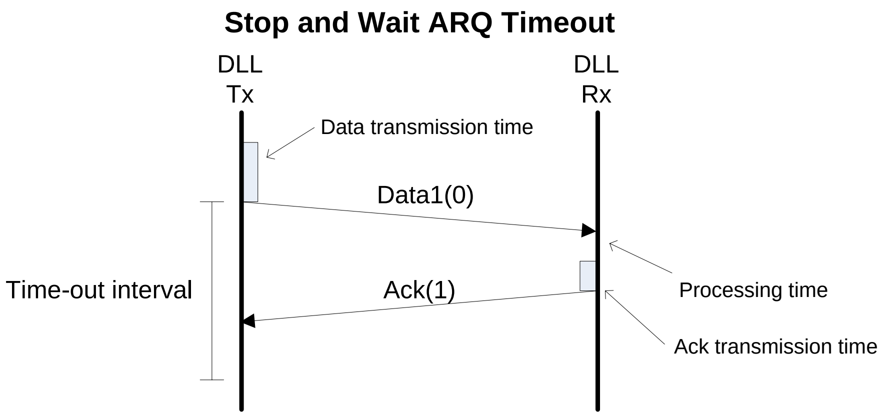
Stop and Wait ARQ is:

Simple, but Inefficient!



An Aside: How long is Timeout interval?

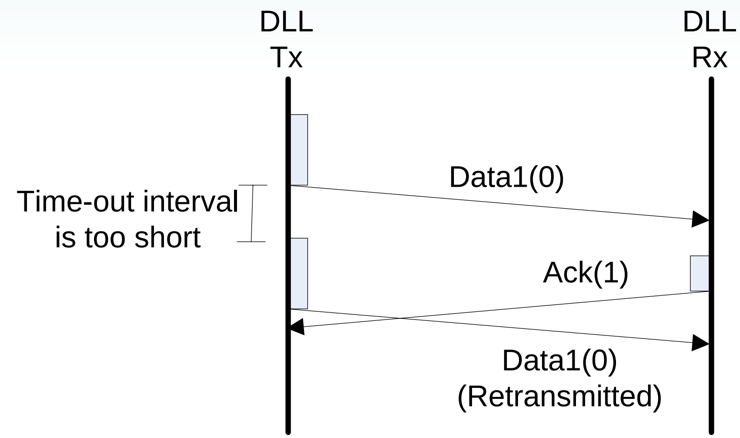
- ARQ protocols set a timer after a frame is transmitted; if the counter reaches a maximum value (timeout interval) then, the frame is retransmitted
- How long should the timeout interval be?



- Timeout interval should be greater than time it takes to receive an ACK
  - DataPropagationTime + ProcessingTime + ACKTransmissionTime + ACKPropagationTime
- Does the sender know these values in practice?
  - ACKTransmissionTime: YES. Usually can predict size of ACK, and know link speed
  - ProcessingTime : NO. This is difficult to predict; it varies depending on receiver
  - PropagationTime (wired link): YES. Usually distance is fixed
  - PropagationTime (wireless): NO. It varies based in distance between sender and receiver
- Sender must estimate an appropriate timeout interval

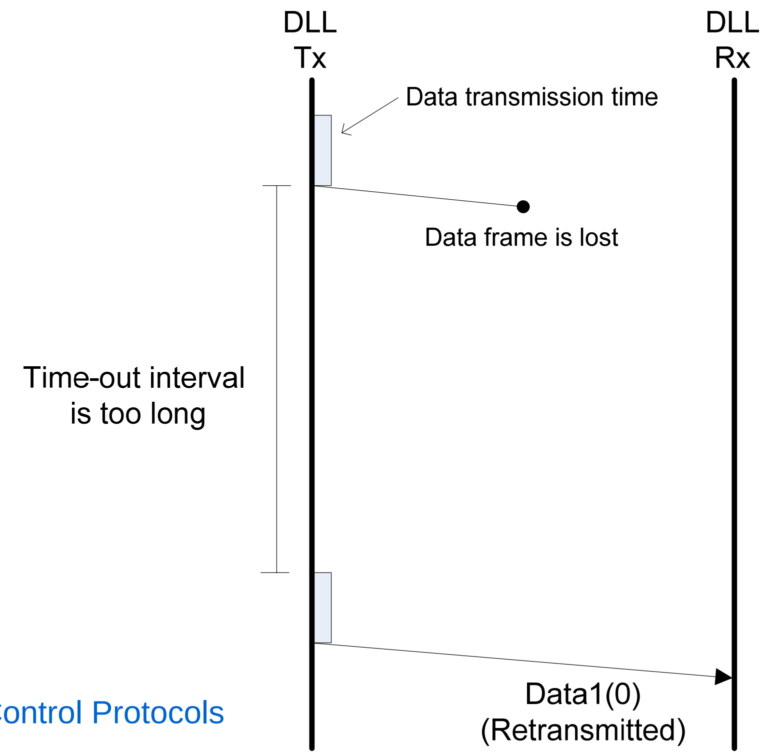
An Aside: How long is Timeout interval?

- If the time-out interval is too short…
  - The sender may retransmit when it doesn't have to
  - Retransmissions reduce efficiency



- If the time-out interval is too long…
  - The sender will wait a long time before retransmitting
  - Waiting (without transmitting any data) reduces the efficiency

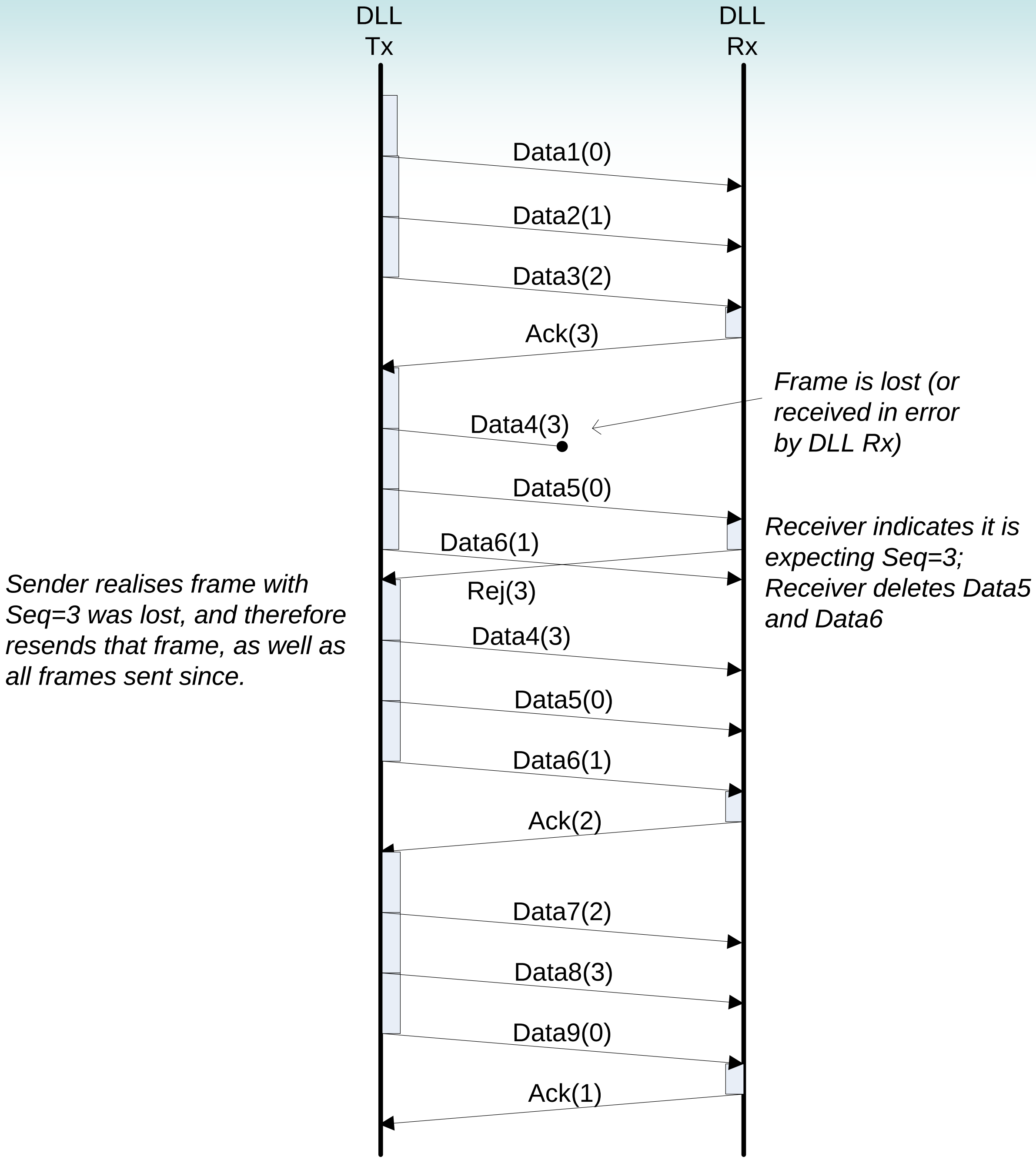
- Choosing a good time-out interval is important for efficiency
- When using networks (later topics), it becomes very hard to estimate a good time-out interval!



Go-Back-N ARQ

- Based on Sliding Window flow control
  - If no error, ACK as in sliding window (contains sequence number of next expected frame)
  - Use window to control number of outstanding frames
  - If error detected by Destination, reply with negative ACK (NACK or rejection, REJ)
    - Destination will discard that frame and all future frames until error frame received correctly
    - Transmitter must go back and retransmit that frame and all subsequent frames (that is, that transmitted but not ACKed)
  - If no response from Destination after timeout, then Source may send special ACK (ACKRequest):
    - The ACKRequest from Source to Destination, is a request for an ACK from the Destination
    - Upon receipt of ACKRequest, the Destination sends an ACK
  - Maximum window size: $2^k - 1$

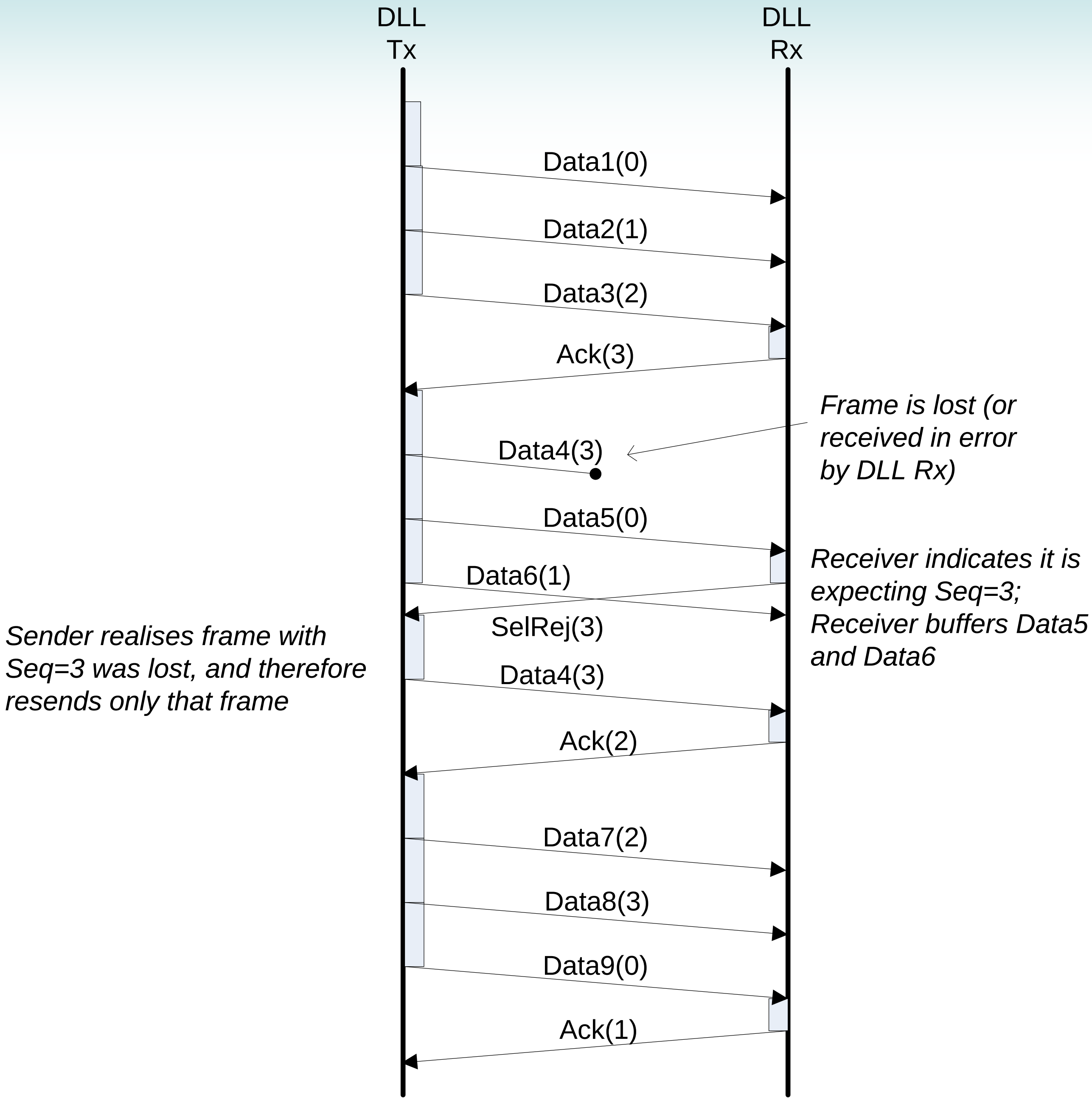
Go-Back-N ARQ


DLL Tx
DLL Rx
Data1(0)
Data2(1)
Data3(2)
Ack(3)
Data4(3)
Frame is lost (or received in error by DLL Rx)
Data5(0)
Data6(1)
Rej(3)
Receiver indicates it is expecting Seq=3; Receiver deletes Data5 and Data6
Sender realises frame with Seq=3 was lost, and therefore resends that frame, as well as all frames sent since.
Data4(3)
Data5(0)
Data6(1)
Ack(2)
Data7(2)
Data8(3)
Data9(0)
Ack(1)

Selective Reject ARQ

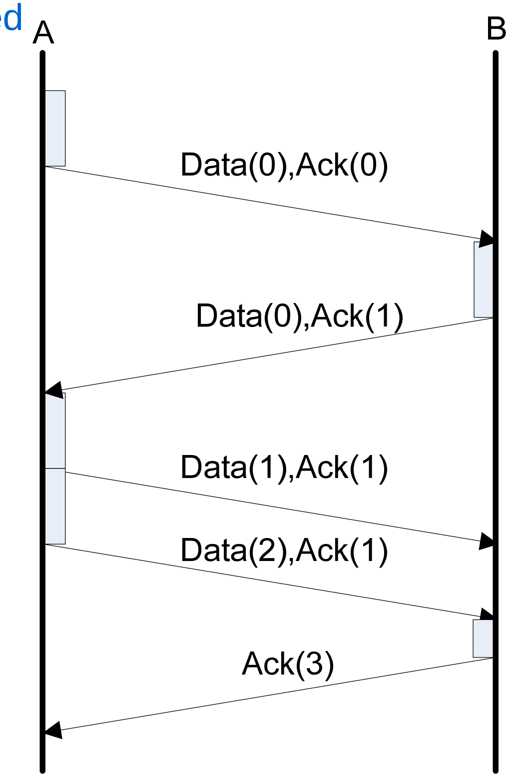
- Also called *selective retransmission* or *selective repeat*
  - Only frames that are rejected or timeout are retransmitted
  - Subsequent frames are accepted by the destination and buffered
  - Maximum window size: $2^{(k-1)}$
- Minimizes retransmission (GOOD)
  - Destination must maintain large enough buffer for frames received out-of-order (BAD)
  - More complex logic in transmitter (BAD)
- Not as widely used as Go-Back-N
  - Useful for satellite links with long propagation delays

Selective-Reject ARQ



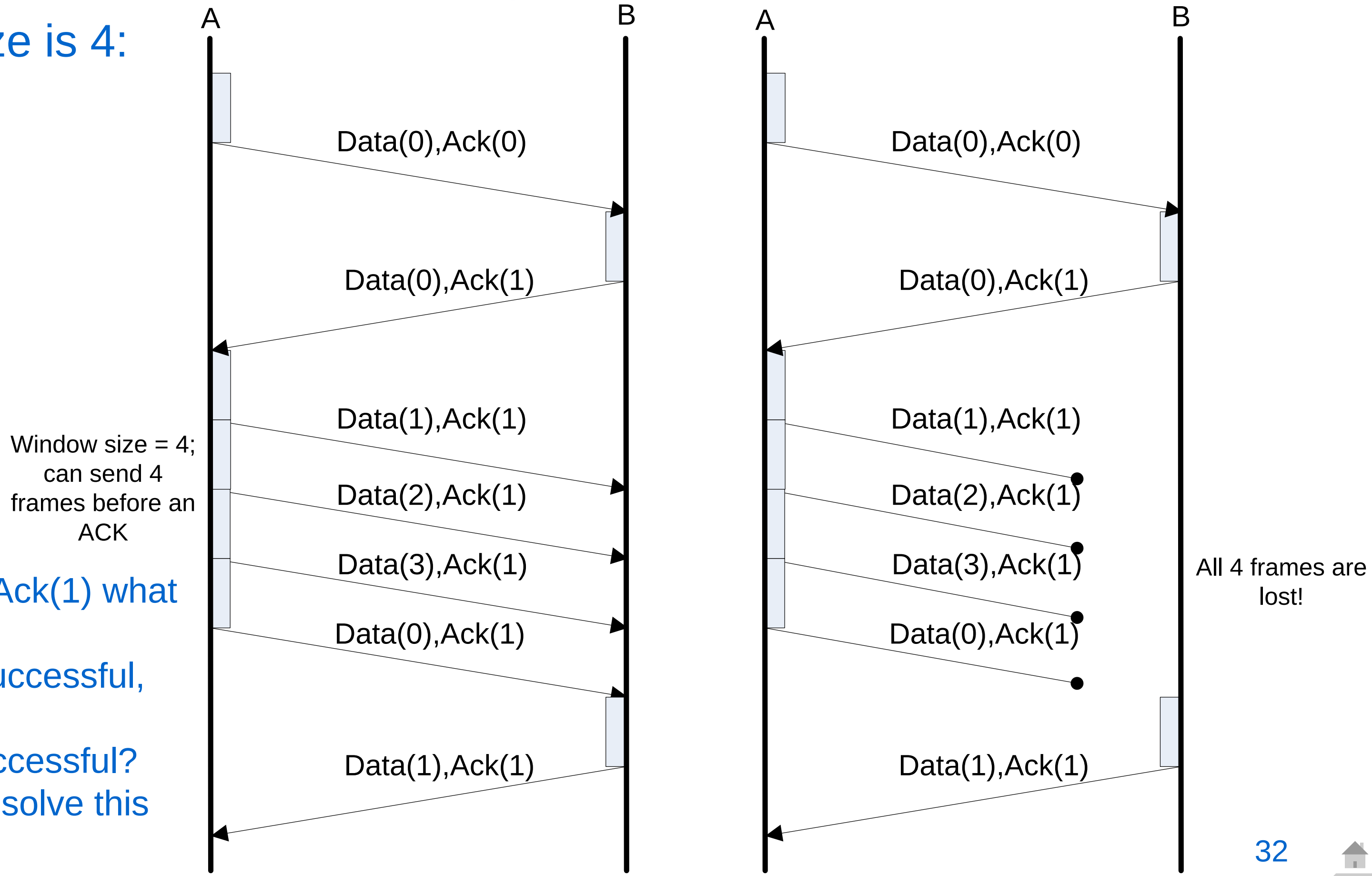
Maximum Window Size

- Go-Back-N (and Sliding Window) have maximum window size of 2^k-1. Why?
  - If data is sent in both directions (A to B, and B to A) then ACKs are piggybacked on Data frames
    - Data frame sent from A to B also contains an ACK from A to B (which acknowledges data sent from B to A)
    - Even if there is nothing new to ACK, if Data is sent, then an ACK for the next expected frame must be included



Maximum Window Size

- If the maximum window size was 2^k, then confusion can occur as to the meaning of the ACKs
- Example: assume k = 2, therefore sequence numbers 0, 1, 2 and 3
- If window size is 4:



When A receives the last Ack(1) what does A know?
a. All 5 Data frames are successful, or
b. Only 1st Data frame successful?
A window size of 2^k-1 can solve this problem

Data Link Layer Protocols

Examples

Example Data Link Layer Protocols

- High Level Data Link Control (HDLC)
  - Developed by ISO, and concepts used in other protocols
  - Provides frame formats, link establishment procedures, flow and error control (e.g. Go-Back-N, Selective Reject)
  - Mainly used for point-to-point links
- Point-to-Point Protocol (PPP)
  - Commonly used by Internet Service Providers: users with dialup connections use PPP for point-to-point link to ISP
  - Uses the Link Control Protocol for link establishment, and Network Control Protocol to negotiate information for specific network layer protocols
  - No flow control, and error control only via CRC
- Local Area Network protocols
  - Ethernet, Wireless LAN (covered in later topic)