Computer Network(CSC 503)

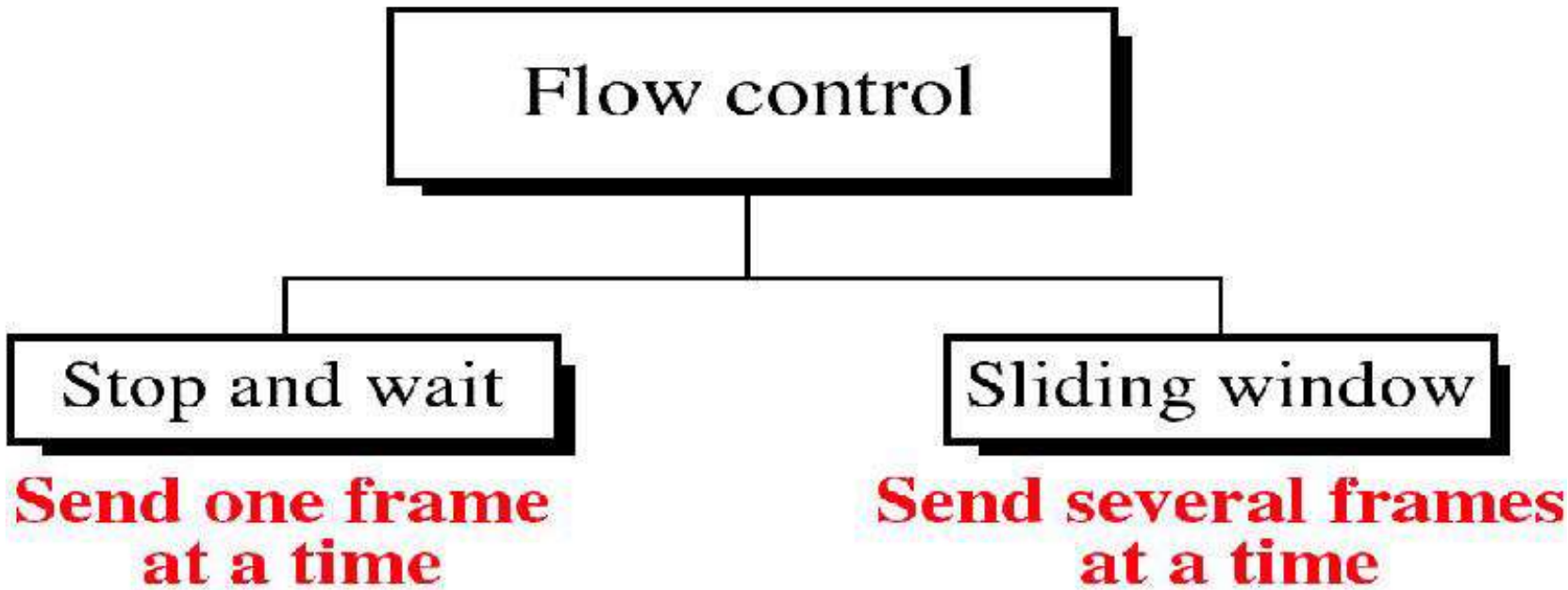
Shilpa Ingoley

Lecture 16 and 17

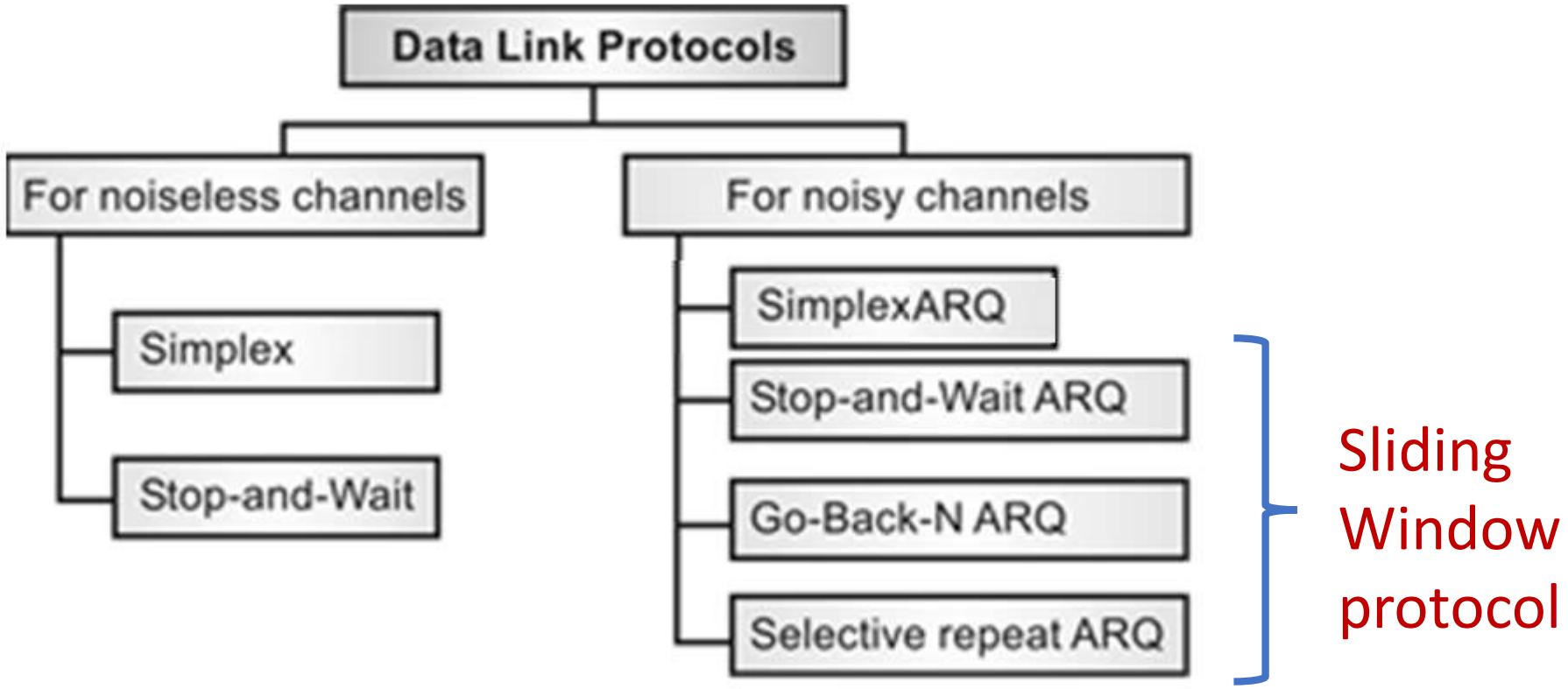
Flow control

- Receiver has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data.
- Receiver must inform the sender before the limits are reached and request that the transmitter to send fewer frames or stop temporarily.
- Flow control coordinates the amount of data that can be sent before receiving acknowledgement
- It is one of the most important functions of data link layer.
- Flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the receiver.

Contd...


Flow control
Stop and wait
Sliding window
Send one frame at a time
Send several frames at a time

Types of Protocols



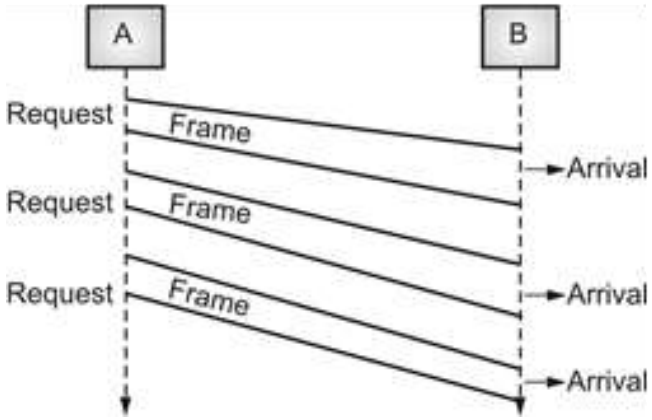
Stop and wait protocol

Types of Stop and Wait Protocol:

- An Unrestricted Simplex Protocol
- A Simplex Stop-and-Wait Protocol
- A Simplex Protocol for a Noisy Channel

1. An Unrestricted Simplex Protocol(Utopia): [Protocol 1]

- Data are transmitted in one direction only - **unidirectional**
- The transmitting (Tx) and receiving (Rx) hosts are always ready
- Infinite buffer space is available
- No errors occur; i.e. no damaged frames and no lost frames (perfect channel)
- The sender is in an infinite while loop just pumping data out onto the line as fast as it can.
- Receiver can **never overwhelmed** with incoming frames.



Contd…

The body of the loop consists of **three** actions:

1 Go fetch a packet from the network layer,

2 Construct an outbound frame and

3. Send it on the way.

- ⏵Only the info field of the frame is used by this protocol, because the other fields have to do with error and flow control and there are no errors and flow control restrictions here.
- ⏵The receiver waits for something to happen, the only possibility being the arrival of an undamaged frame.
- ⏵It passes to the network layer and the data link layer settles back to wait for the next frame.

2. A simplex stop-and-wait protocol: [Protocol 2]

- In this protocol we assume that data are transmitted in one direction only
  - No errors occur (perfect channel)
  - The receiver can only process the received information at a finite rate
- These assumptions imply that the transmitter **cannot send frames at a rate faster than the receiver can process them**.

Contd…

- The problem here is how to prevent the sender from flooding the receiver.

- A general solution to this problem is to have the receiver provide some sort of feedback to the sender.

- The process could be as follows:

- The receiver send an acknowledge frame back to the sender telling the sender that the last received frame has been processed.
- Permission to send the next frame is granted. The sender, after having sent a frame, must wait for the acknowledge frame from the receiver before sending another frame. **This protocol is known as stop-and-wait.**

Contd...

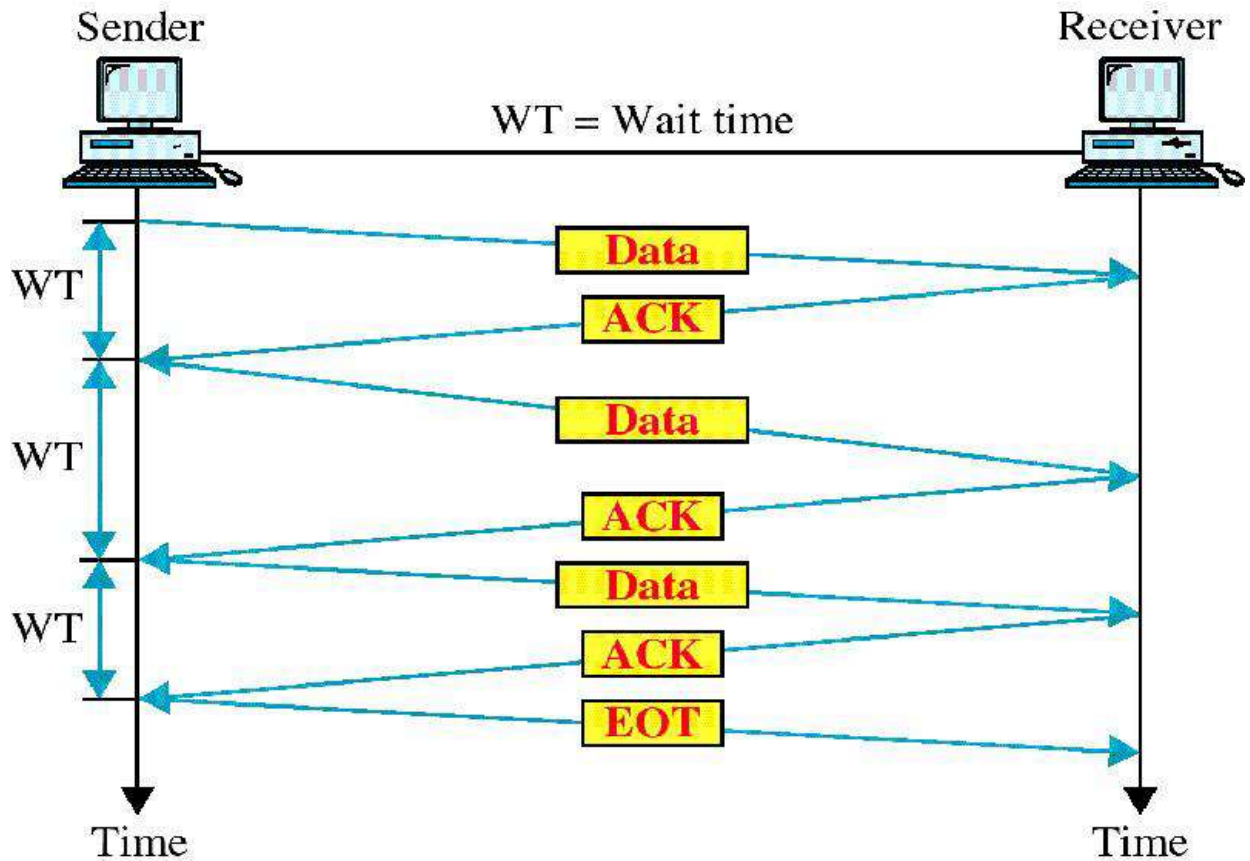


Fig: Stop-and-wait Protocol

Advantages and Disadvantages of Stop-and-wait

Advantage

1. The Stop-and-wait method is **simple as each frame is checked and acknowledged** before the next frame is sent.

Disadvantage

- 1. Stop-and-wait technique is **inefficient to use** as each frame must travel across all the way to the receiver and an **acknowledgement travels** all the way **before the next frame is sent.**
- 2. Each frame sent and received uses the **entire time needed to traverse the link.**

Protocols for Noisy channel

3.A simplex protocol for a noisy channel:[Protocol:3]

- In this protocol the unreal "**error free**" assumption in protocol 2 is dropped. Frames may be either damaged or lost completely.
- The sender would send a frame and the receiver would send an **ACK frame only if the frame is received correctly.** If the frame is in error the receiver simply ignores it; the transmitter would time out and would retransmit it.
- One fatal flaw with the above scheme is that if the ACK frame is lost or damaged, duplicate frames are accepted at the receiver without the receiver knowing it.

Protocols in which the sender waits for positive acknowledgement before advancing to the next data item are often called PAR(Positive Acknowledgement with Retransmission or ARQ (Automatic Repeat reQuest)

Contd…

- Imagine a situation where the receiver has just sent an ACK frame back to the sender saying that it correctly received and already passed a frame to its host.
- ⏵However, the ACK frame gets lost completely, the sender times out and retransmits the frame.
- ⏵There is no way for the receiver to tell whether this frame is a retransmitted frame or a new frame, so the receiver **accepts this duplicate** happily and transfers it to the host. The protocol thus fails in this aspect.
- ⏵To overcome this problem it is required that the receiver be able to distinguish a frame that it is seeing for the **first time from a retransmission**.

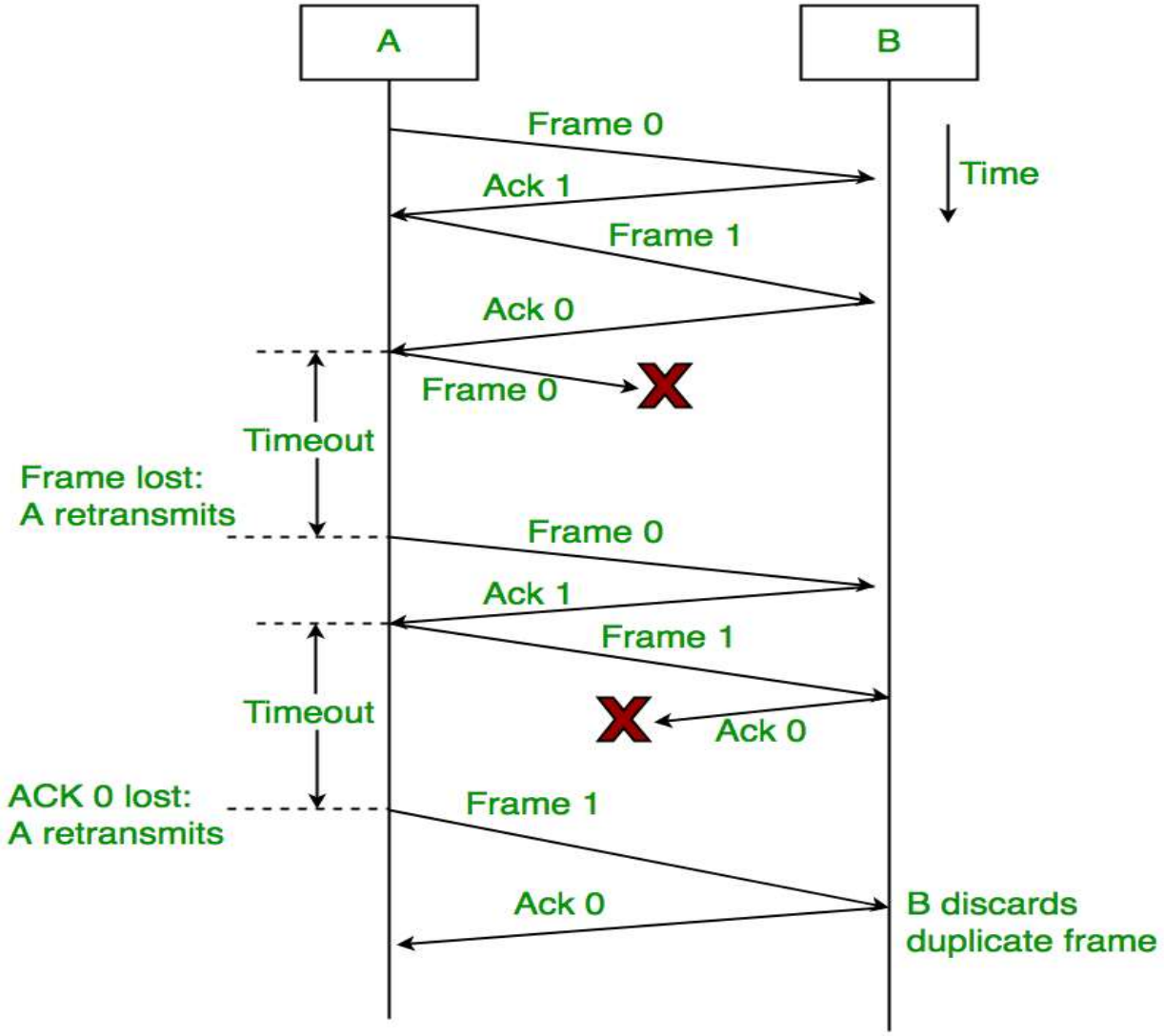
Contd…

- One way to achieve this is to have the sender put **a sequence number** in the header of each frame it sends. The receiver then can check the sequence number of each arriving frame to see if it is a new frame or a duplicate to be discarded.
- ⏵The receiver needs to distinguish only 2 possibilities: a new frame or a duplicate; a 1-bit sequence number is sufficient. At any instant the receiver expects a particular sequence number. Any wrong sequence numbered frame arriving at the receiver is rejected as a duplicate. A correctly numbered frame arriving at the receiver is accepted, passed to the host, and the expected sequence number is incremented by 1

Contd…

- 1) Sender A sends a data frame or packet with sequence number 0.
- 2) Receiver B, after receiving data frame, sends and acknowledgement with sequence number 1 (sequence number of **next expected data frame** or packet)There is only one bit sequence number that implies that both sender and receiver have buffer for one frame or packet only.

Stop & Wait Protocol



Contd…

- **Drawback of Stop and wait protocol(A Simplex Stop-and-Wait Protocol and A Simplex Protocol for a Noisy Channel-ARQ)**

  1. Very Inefficient. At a moment , only one frame is in transition.
  2. The sender has to wait at least one round trip time before sending the next frame.
  3. Poor utilization of bandwidth
  4. Poor Performance

Three types of sliding window protocol

1. One-bit sliding window protocol:**[Protocol:4]**

2. Go-back n sliding window protocol (GBN) **:[Protocol:5]**

3. Selective repeat sliding window protocol (SR) **:[Protocol:6]**

Contd...

Assumptions:

- 1.**Data** is transmitted in **both the directions(**Bidirectional**).**
- 2.Requires **full duplex** communication channel.
- 3.No separate ACK sent.
- 4.Uses the concept of piggybacking.(The technique of temporarily delaying the outgoing ACK so they can be hooked onto the next outgoing data frame is known as **piggybacking**).
- 5.Noisy channel.

Contd…

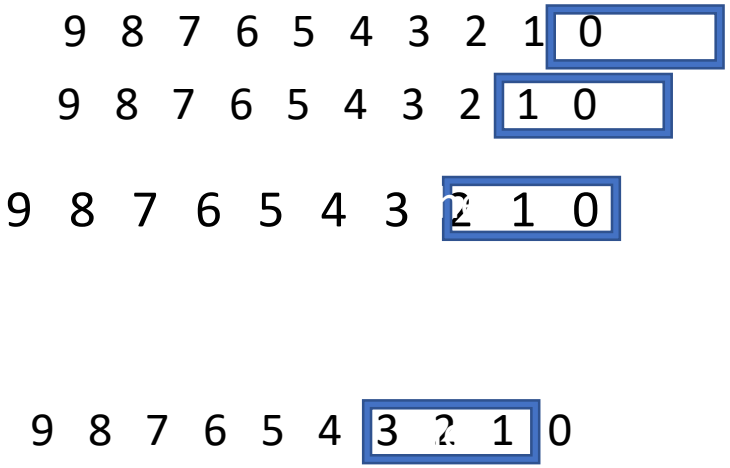
- The Sliding Window is a method of **flow control** in which a **sender can transmit the several frames before getting an acknowledgement.**
- **Multiple frames can be sent one after another** due to which capacity of the communication channel can be utilized efficiently.
- A **single ACK acknowledge multiple frames.**
- Sliding Window refers to imaginary boxes at both the sender and receiver end.
- The window can **hold the frames at either end,** and it provides the **upper limit** on the **number of frames** that can be **transmitted before the acknowledgement.**
- Frames can be **acknowledged** even when the **window is not completely filled**.
- **The window has a specific size in which they are numbered as modulo-n means that they are numbered from 0 to n-1.**
- **Example :** if n = 8, the frames are numbered from 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0,1 …….
- The **size of the window** is represented as **n-1.** Therefore, maximum n-1 frames can be **sent before acknowledgement.**

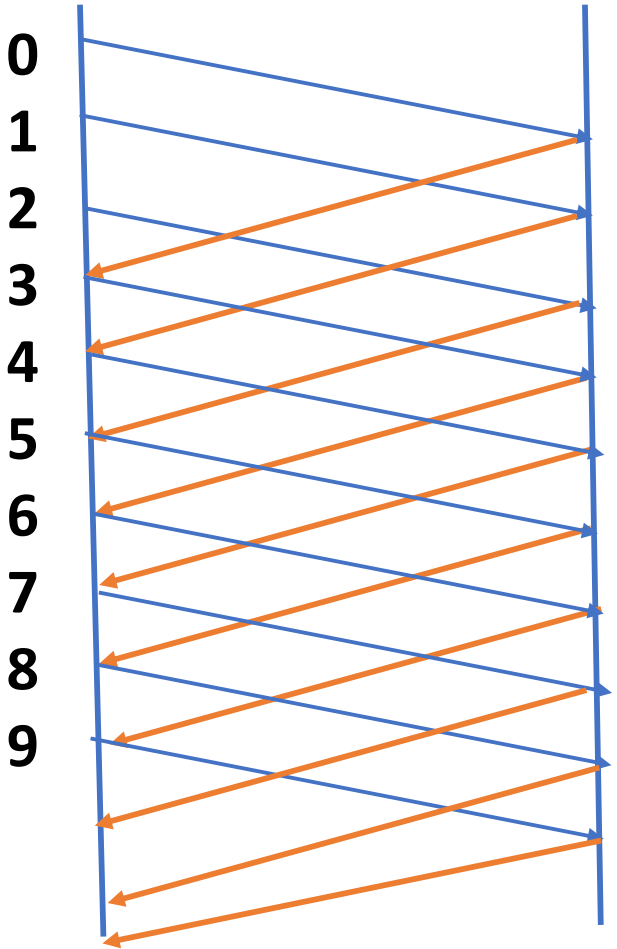
Contd…

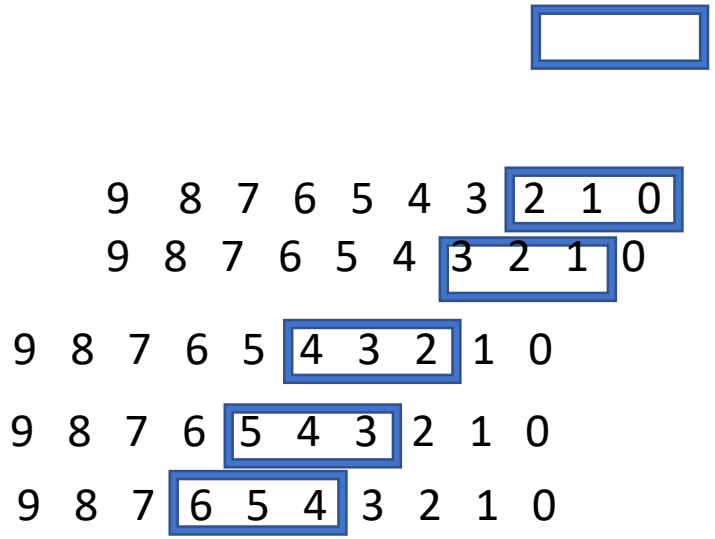
- When the **receiver sends the ACK,** it includes the **number of the next frame** that it wants to receive.
- When the sender sees the **ACK with the number 4**, it gets **know that the frames from 0 through 3** have been received.
- **Receiver Window**
- At the beginning of transmission, the receiver window does not contain n frames, but it contains n-1 spaces for frames
- The receiver window does not represent the number of frames received, but it represents the number of frames that can be received before an ACK is sent.
- If the one frame is received, then the receiver window shrinks and moves the boundary from 0 to 1.

Concept of Sliding Window

- Ten frames you want to transmit: 0 1 2 3 4 5 6 7 8 9

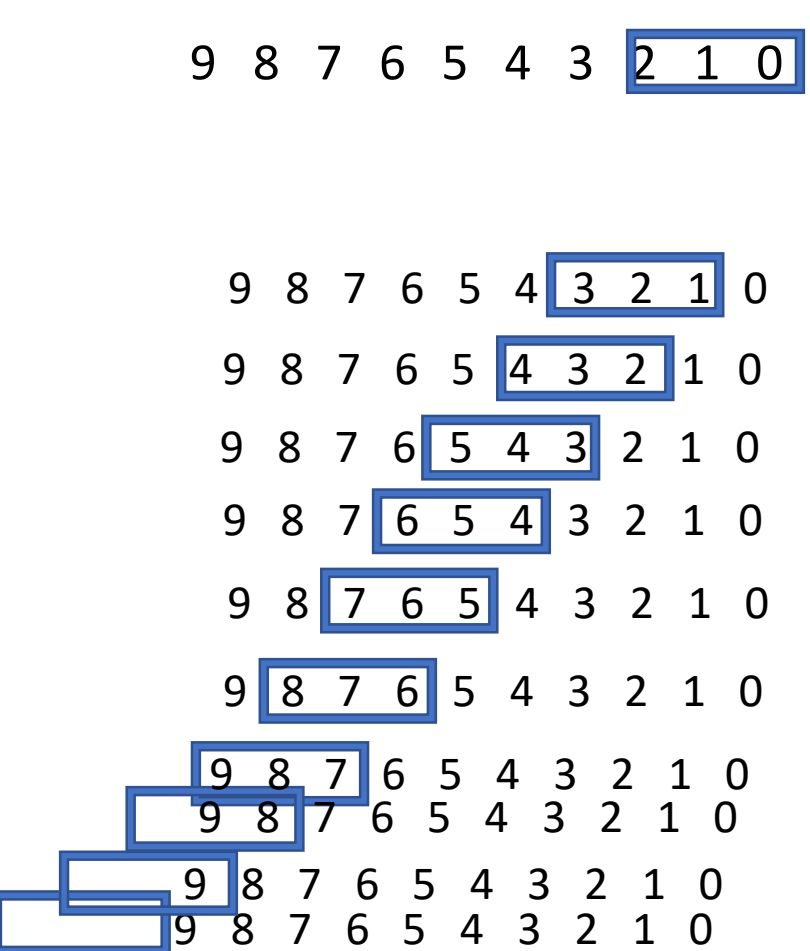


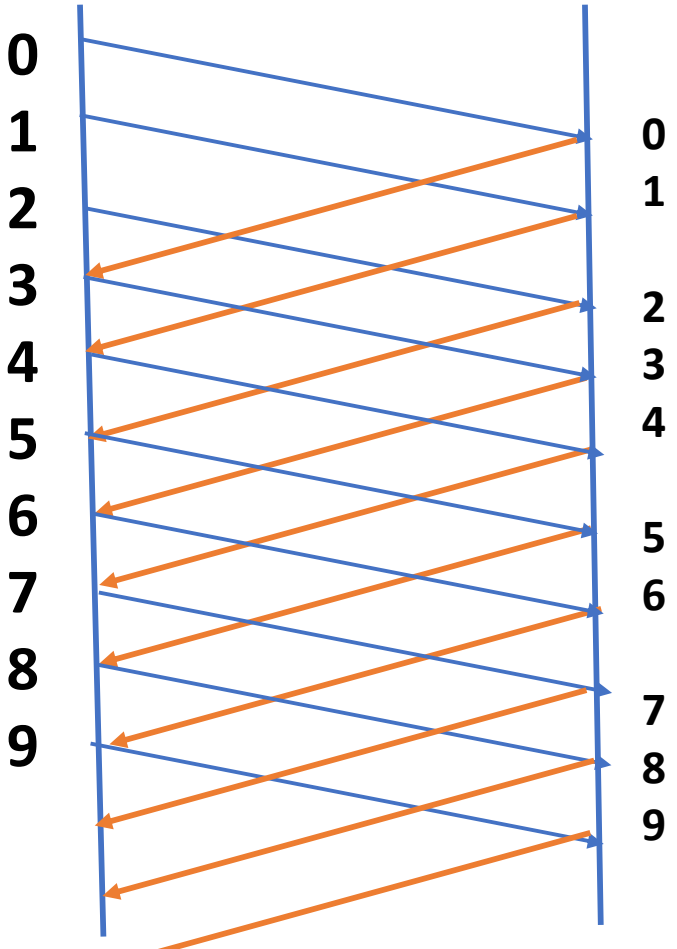


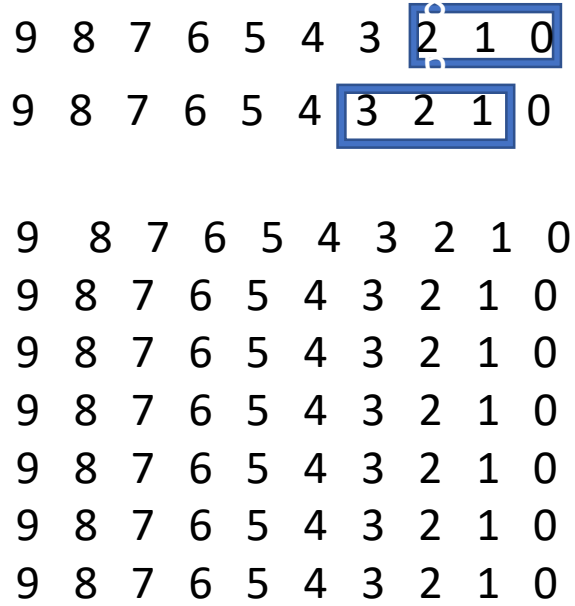


Contd…

- Ten frames you want to transmit: 0 1 2 3 4 5 6 7 8 9





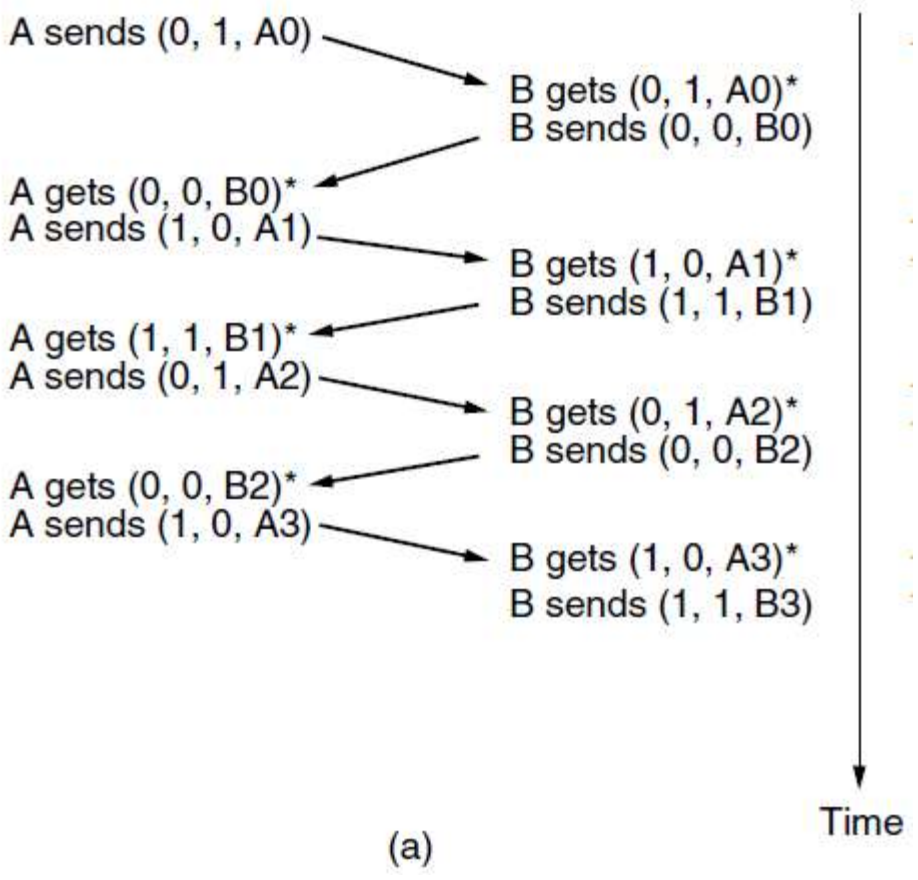


One-bit sliding window protocol:[Protocol :4]

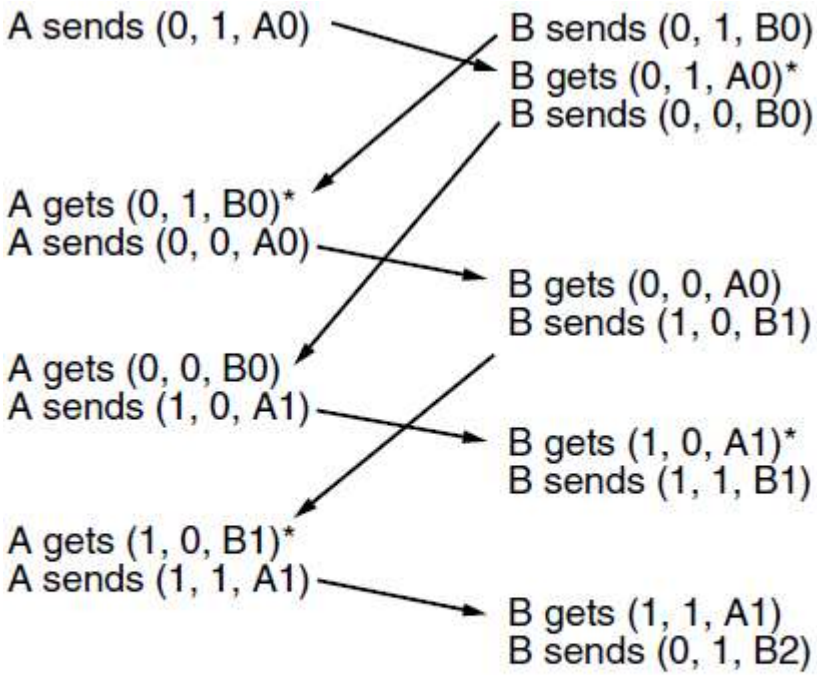
- Sender transmits one frame and waits for it’s ACK, before sending the next frame
  - Uses stop & wait .
  - Window size is 1.
  - Each end simultaneously acts as both sender and receiver .
  - Sender sends frames as tuples.
  - The ack no indicates the last frame received without error.

Two scenarios for protocol 4(One bit sliding window protocol)

Problem if both send initial packet at same time.



(a)



(b)

Fig **(a) Normal case.** **(b) Abnormal case (Duplication of Packet)**

Note: The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

- Here Acknowledge is number of last frame received

If both send simultaneously

- A sends 0,1 B sends 0,1

  B gets 0,1 (got 0, but no ack of my 0)

  B **unnecessarily re-sends**0,0

- A gets 0,1 (got 0, but no ack of my 0) A **unnecessarily re-sends**0,0
- B gets 0,0 (already have 0, **nothing to pass to NB**, but this is good ack of my 0) B sends 1,0
- A gets 0,0 (already have 0, **nothing to pass to NA**, but this is good ack of my 0) A sends 1,0
- B gets 1,0 (got 1, but no ack of my 1) B **unnecessarily re-sends**1,1
- A gets 1,0 (got 1, but no ack of my 1) A **unnecessarily re-sends**1,1
- B gets 1,1 (already have 1, **nothing to pass to NB**, but this is good ack of my 1) B sends 0,1

Contd…

- **Drawback:**
  - Inefficient- Sender waits until an ACK arrives from the receiver.
  - Sends only one frame at a time