Data Link Layer: Sliding Window Protocols
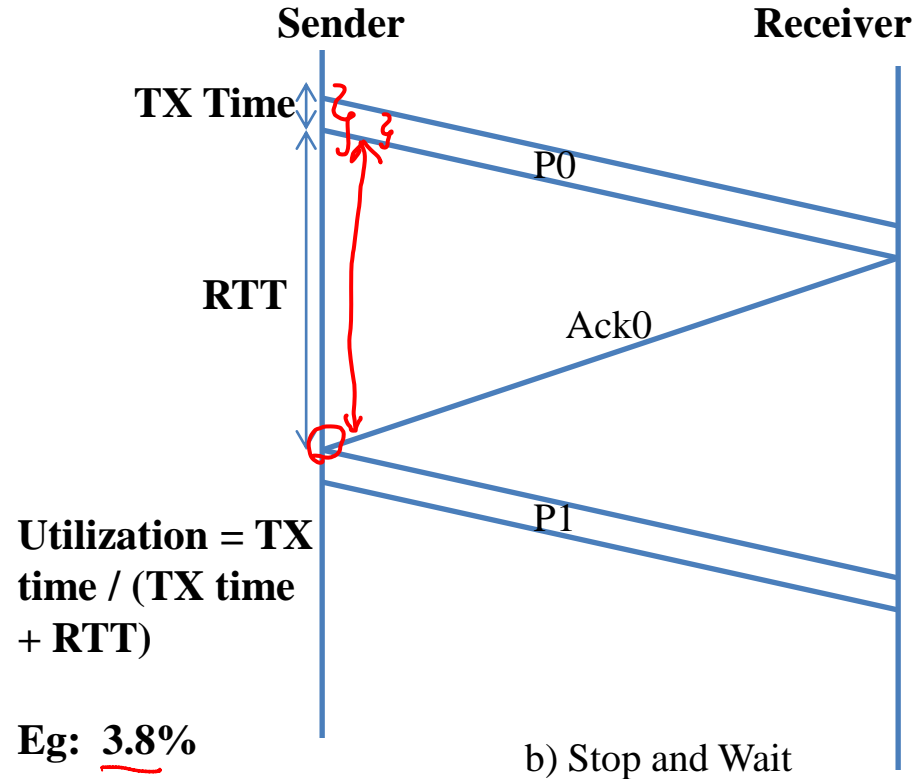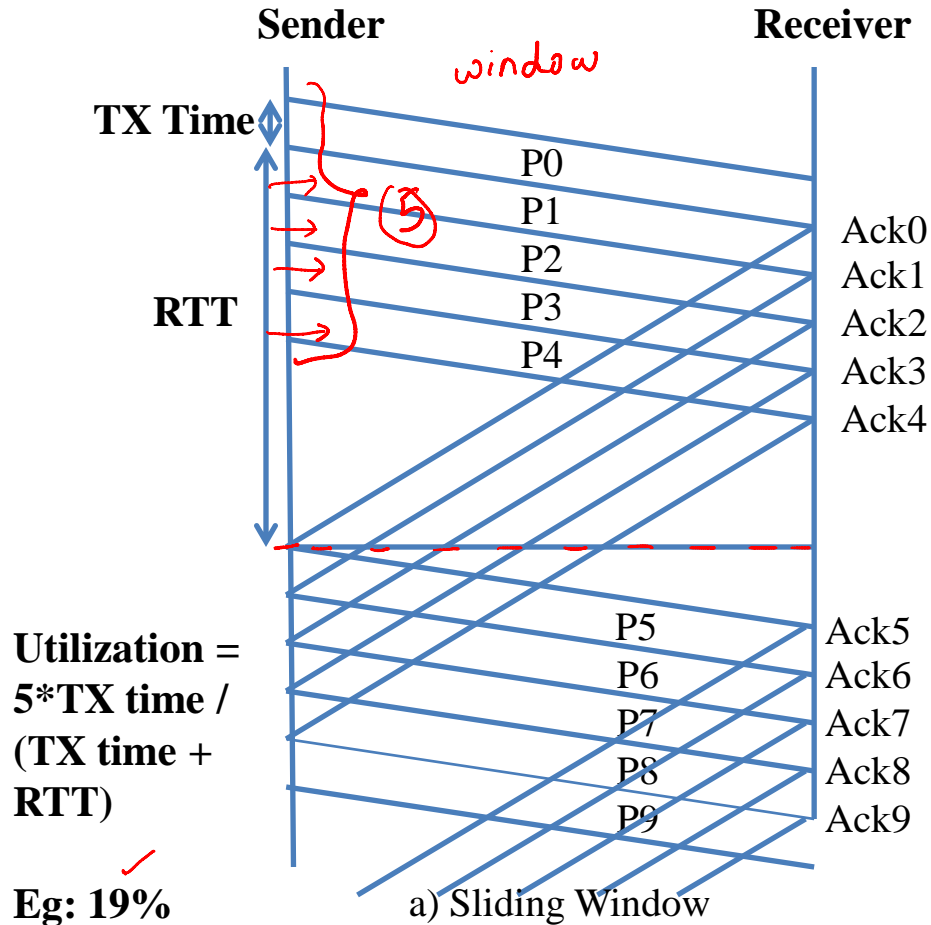
Kameswari Chebrolu

Experience

A man who carries a cat by the tail learns something he can learn in no other way.
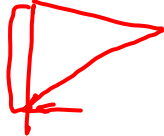
-- Mark Twain

Recap

- Incrementally built the framework for a Reliable Data Transfer (RDT) protocol

  – Required Functionality

  – Stop and Wait Protocol: Works correctly but performance is poor

- Sliding Window Protocols: Can improve performance considerably

General Idea

Sender   **Receiver**

window

TX Time

RTT

P0
P1 ⑤
P2
P3
P4

Ack0
Ack1
Ack2
Ack3
Ack4

**Utilization =
5*TX time /
(TX time +
RTT)**

P5   Ack5
P6   Ack6
P7   Ack7
P8   Ack8
P9   Ack9

Eg: 19%

a) Sliding Window

Sender   **Receiver**

TX Time

RTT

P0

Ack0

**Utilization = TX
time / (TX time
+ RTT)**

P1

Eg: 3.8%

b) Stop and Wait
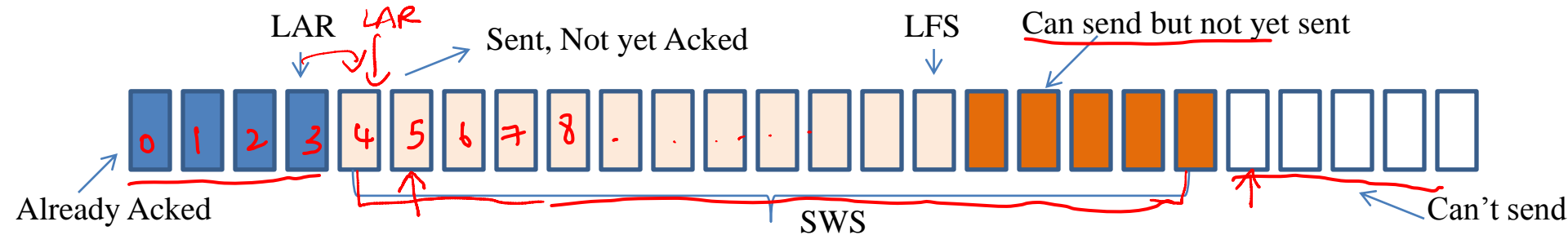
Bandwidth-Delay Product

- Sender can send a maximum of W packets (window size) without waiting for an ACK

- What can the window size be?

  - To keep utilization maximum, sender can send 'roughly' upto Data-Rate*RTT before hearing an ack

  - Window size = Bandwidth * RTT bits

    - Divide by packet size to get the number of packets

Sliding Window

- At any given time, no more than W packets can be outstanding (their status not known at sender)

- As status of packets gets known at sender, the window of sequence number slides to encompass newer sequence numbers

  - Permits sender to send subsequent packets

Sender Side

- Assign a sequence number to each frame

- Maintain 3 variables:

  - Send Window Size (SWS): upper bound on number of outstanding frames

  - LAR: Seq. No of Last Acknowledgment Received; Advance LAR when ACK arrives

  - LFS: Sequence number of Last Frame Sent

- Maintain Invariant: LFS-LAR <= SWS

LAR   LAR   Sent, Not yet Acked   LFS   Can send but not yet sent

0 1 2 3 4 5 6 7 8 . . . . .
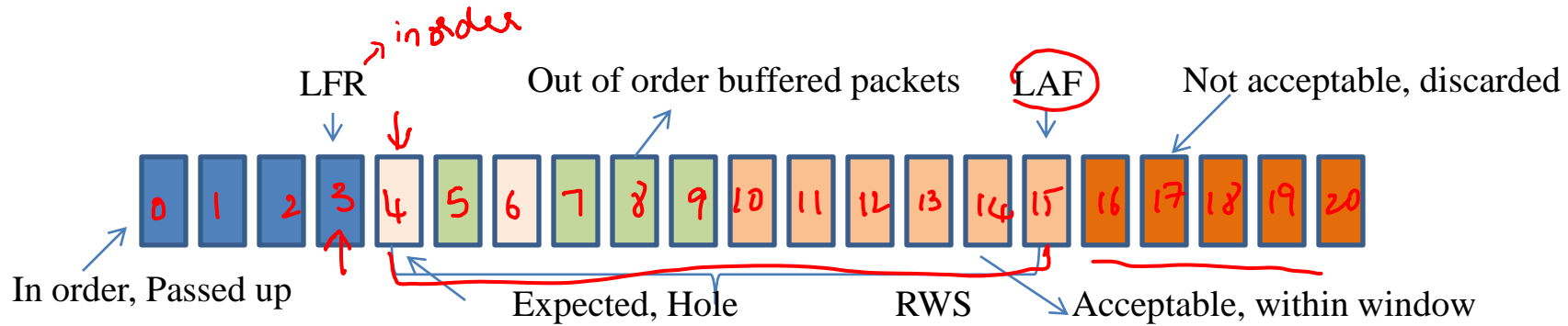
Already Acked

SWS

Can't send

Receiver Side

- Maintains the following three variables

  – Received Window Size (RWS): upper bound on the number of out of order frames

  – LAF denotes sequence number of last acceptable frame

  – LFR denotes sequence number of last frame received

- Set LAF = LFR + RWS

RWS > SWS ?
RWS ≤ SWS
RWS = 1 ?

Receiver accept only in order packets

LFR — in order

Out of order buffered packets

LAF

Not acceptable, discarded

In order, Passed up

Expected, Hole

RWS

Acceptable, within window

Consequences

- Need a range of sequence numbers (two won't suffice)

  $w = 1$

- Sender has to buffer more than one frame (buffer all transmitted but not yet acked)

  $RWS \le SWS$

- Receiver may also have to buffer (out of order frames)

- Range of protocols based on sliding window

  - Go Back N, Selective Repeat, TCP

  - Actions taken on events (like frame/ack rcvd, duplicate ack, timeout etc) differ

Action at the sender

- On receiving a packet from higher layer:

  - Send the packet or buffer it or discard it?

- On receiving ACK:

  - Send packet?

  - If so, which one?

- Timeout: Retransmit packet, update timer

Action at Receiver

- On receiving a packet:

  - Should I accept it or discard?

  - Whether accepted or not, should I generate ACK?

 - If ACK generated, what should I acknowledge in it?

Receiver Side Details

- Frame "SeqNum" arrives
  - If SeqNum <= LFR or SeqNum > LAF, discard frame
  - If LFR < SeqNum <= LAF, accept

- Always good to send an ACK (improves performance by early detection of losses)

  - Exception: depends on protocol (e.g. Go-Back-N doesn't send Ack when frames are discarded)

- Types of ACK: Cumulative Ack; Selective ACK

  - Tradeoff: Simplicity vs Performance

Cumulative ACK

- SeqNumToAck

  - Largest sequence number received such that all frames with sequence number less than SeqNumToAck have also been received

  - LFR is set to this value

Example

- Frames 0,1,2,3,5,7 arrive in that order

- State: Receiver acked frame 3, assume RWS is 4

  - When 5 arrives: Buffer 5, ack 3, LFR=3, LAF=7

  - When 7 arrives, Buffer 7, ack 3, LFR=3, LAF=7

  - When 4 arrives (say due to sender retransmission), it fills the hole,

    - Accept 4 and pass up packets till 5, ack 5, LFR=5, LAF=9

  - When 6 arrives, accept/pass up, ack 7, LFR=7, LAF=11
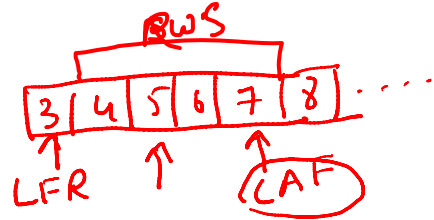
Selective ACK

- For out of order packets: ACK <u>sequence</u> number of whatever packet accepted

- In previous example: 0,1,2,3,5,7

    – When 5 arrives, ack 3, sack 5

    – When 7 arrives, ack 3, sack <u>7 5</u>

    – When 4 arrives, ack 5, sack 7

    – When 6 arrives, ack 7

Sender Side Details

- On receiving packet from higher layers:

  - Determine next sequence number to use (SeqNum)
  - If SeqNum – LAR <= SWS, send the packet, else buffer or inform application

Sender Side Details

- On receiving ACK: action function of ACK type

  - Can get very complex (e.g. TCP variants)

  - Duplicate Acks and selective acks can be used to determine packet loss, which in turn trigger retransmission

    $\rightarrow$ empty the pipe

    - Timeout is a backup mechanism

  - Simple Version: Use cumulative ack to advance window

    - If SeqNum in ACK > LAR, LAR = SeqNum in ACK

    - If SeqNum of Buffered packets <= LAR+SWS, send them

Sequence Number Space

- How many sequence numbers do you need?
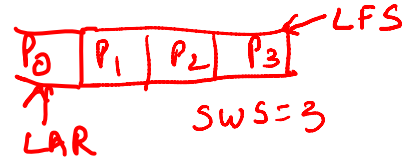
- Suppose RWS = 1, SWS = 3

  2 seq #
  w = 1

  – Maxseqno = 4 i.e. 0,1,2, 3?

  0
  P0 P1 P2 P3 P4

  | | P₃ | r |

  ↑
  LFR   Expecting P₄

- Suppose RWS=SWS=3
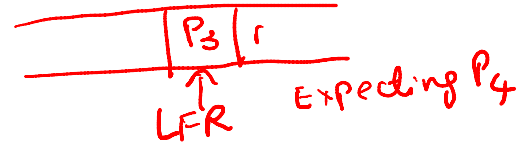
  LFS
  | P₀ | P₁ | P₂ | P₃ |
  ↑
  LAR   SWS=3

  – Maxseqno=4 sufficient?

  sender: 0,1,2 next
                  ↓
  receiver: 3,0,1   previous
  Ack's lost, Sender retx 0-1,2

- What is the general rule?

  RWS, SWS ?

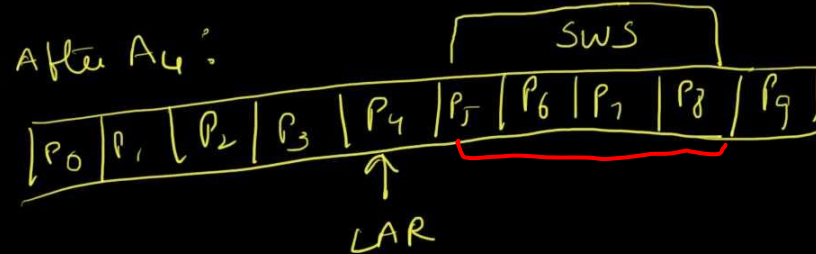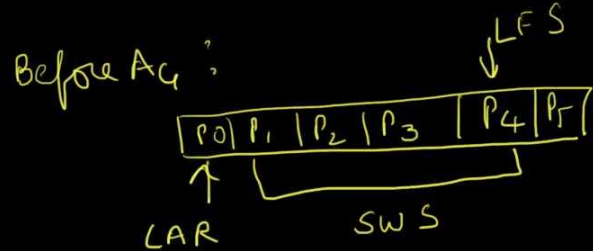# Simple Sliding Window Example

* SWS = RWS = 4
* on receipt of a frame,
  receiver sents cumm. Ack
* Timeout = 2 RTT
* Retransmission @ Sender

  due to duplicate ACK
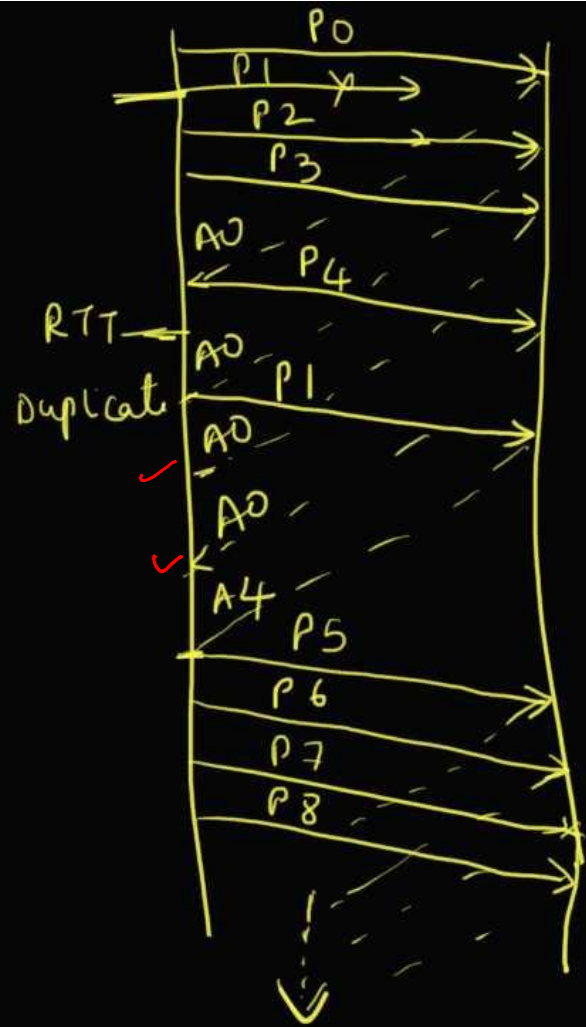
Go Back N

- On TimeOut, sender resends all packets that have been previously sent but unack'ed

- RWS = 1
  - Receiver accepts only in order packet and generates cumulative ack

  - Discards out of order packets, no ack generated

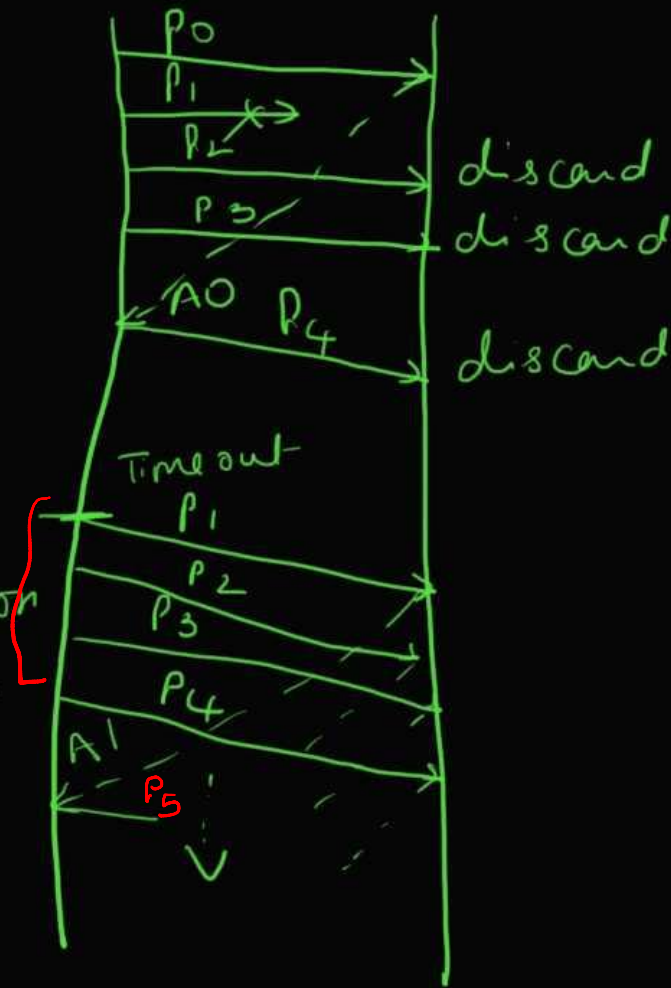- Inefficient but receiver design is very simple

Go Back N

* Receiver discards out of order Packets; NO ACK generated
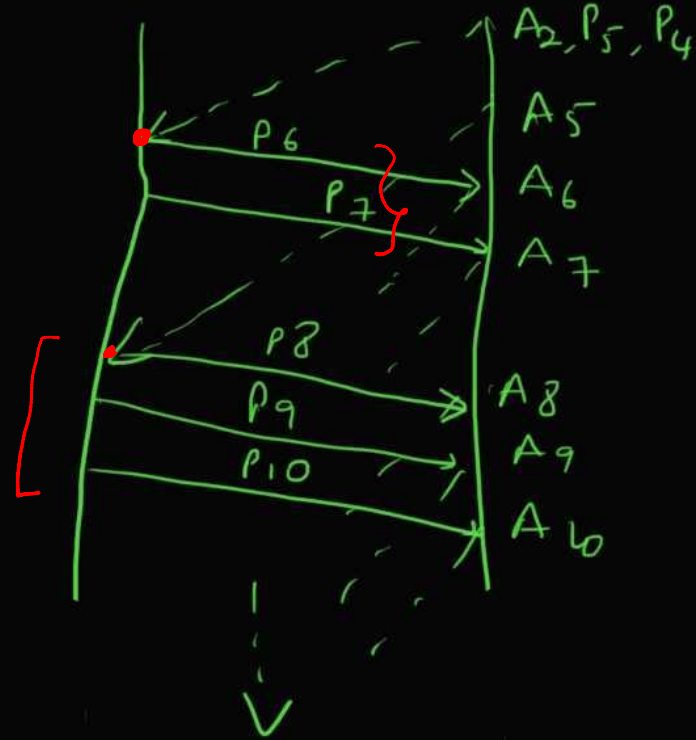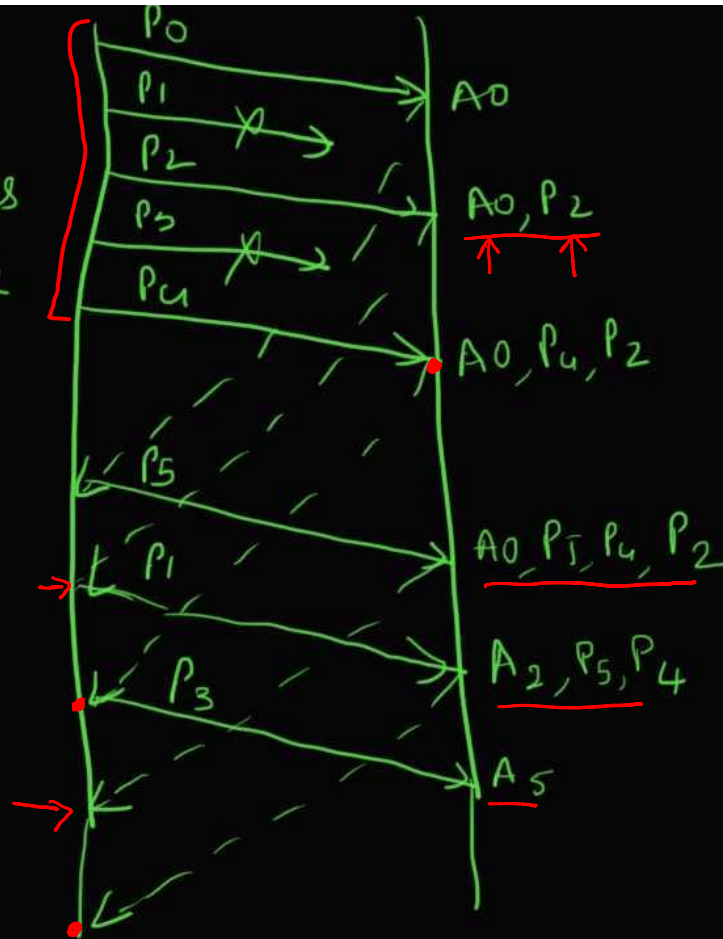
* Generates cumm. ACK for in order Packets

* Sender relies on Timeout for Retransmission

* On timeout, all sent but unacked packets are sent

Selective Repeat

Selective Repeat

* SWS = RWS = 5
* Multiple losses
* Selective Ack

P_0
P_1
P_2
P_3
P_4

A_0
A_0, P_2
A_0, P_4, P_2

P_5
P_1
P_3

A_0, P_1, P_4, P_2
A_2, P_5, P_4
A_5

P_6
P_7

A_2, P_5, P_4
A_5
A_6
A_7

P_8
P_9
P_{10}

A_8
A_9
A_{10}

Flow Control

TCP

- Prevents the <u>sender</u> from overflowing the receiver buffer

  small

- Receiver informs <u>sender how many frames it</u> has room to receive.

  – Sender will always respect this when sending frames

  SwS

Summary

- Sliding Window protocols rely on 'window of packets' to improve performance

    - More complex (range of sequence numbers, types of acks, action at sender/receiver)

- In addition to reliability, they provide

    - In-order delivery ✓

    - Flow control ✓

- Milestone: Achieved point-to-point reliable and efficient data transfer (Building block)