IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

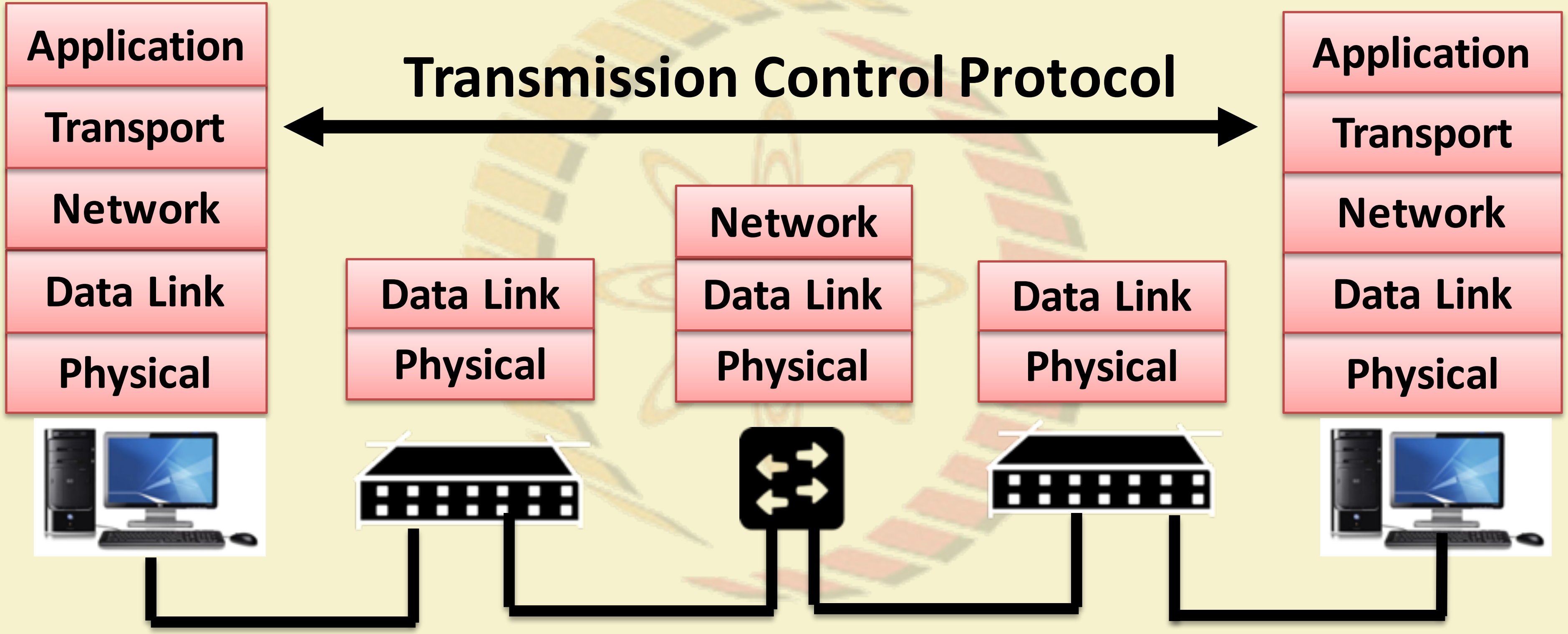
NPTEL

# COMPUTER NETWORKS AND INTERNET PROTOCOLS

SOUMYA K GHOSH
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

SANDIP CHAKRABORTY
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

# Transmission Control Protocol IV (Congestion Control)



## TCP Congestion Control

- Based on implementation of AIMD using a window and with packet loss as the binary signal
- TCP maintains a **Congestion Window (CWnd) –** number of bytes the sender may have in the network at any time
- **Sending Rate = Congestion Window / RTT**
- **Sender Window (SWnd) = Min (CWnd, RWnd)**
- RWnd – Receiver advertised window size

## 1986 Congestion Collapse

- In 1986, the growing popularity of Internet led to the first occurrence of congestion collapse – a prolonged period during which goodput dropped precipitously (more than a factor of 100)

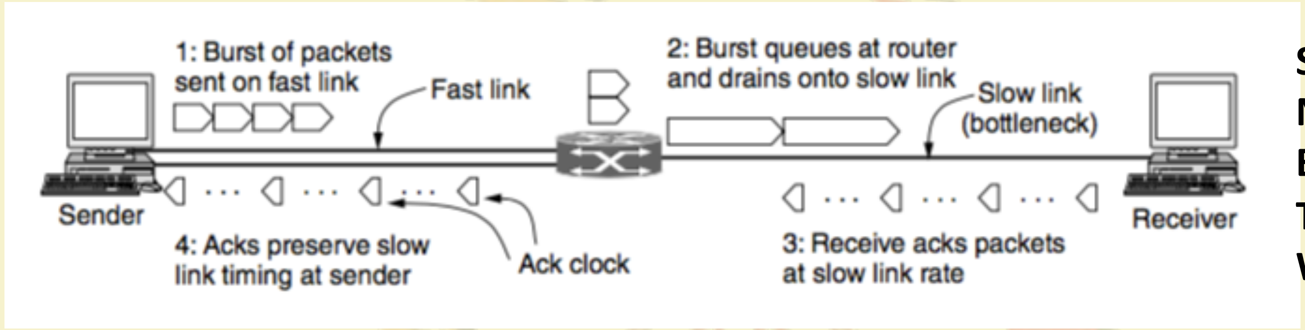
- Early TCP Congestion Control algorithm – Effort by Van Jancobson (1988)

- **Challenge for Jacobson** – Implement congestion control without making much change in the protocol (made it instantly deployable)

- **Packet loss is a suitable signal for congestion – use timeout to detect packet loss. Tune CWnd based on the observation from packet loss**

## Adjust CWnd based on AIMD

- **One of the most interesting ideas – use ACK for clocking**


1: Burst of packets sent on fast link
Fast link
2: Burst queues at router and drains onto slow link
Slow link (bottleneck)
Sender
Receiver
4: Acks preserve slow link timing at sender
Ack clock
3: Receive acks packets at slow link rate

**Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell**

- ACK returns to the sender at about the rate that packets can be sent over the slowest link in the path.

- Trigger CWnd adjustment based on the rate at which ACK are received.

## Increase Rate Exponentially at the Beginning – The Slow Start

- AIMD rule will take a very long time to reach a good operating point on fast networks if the CWnd is started from a small size.

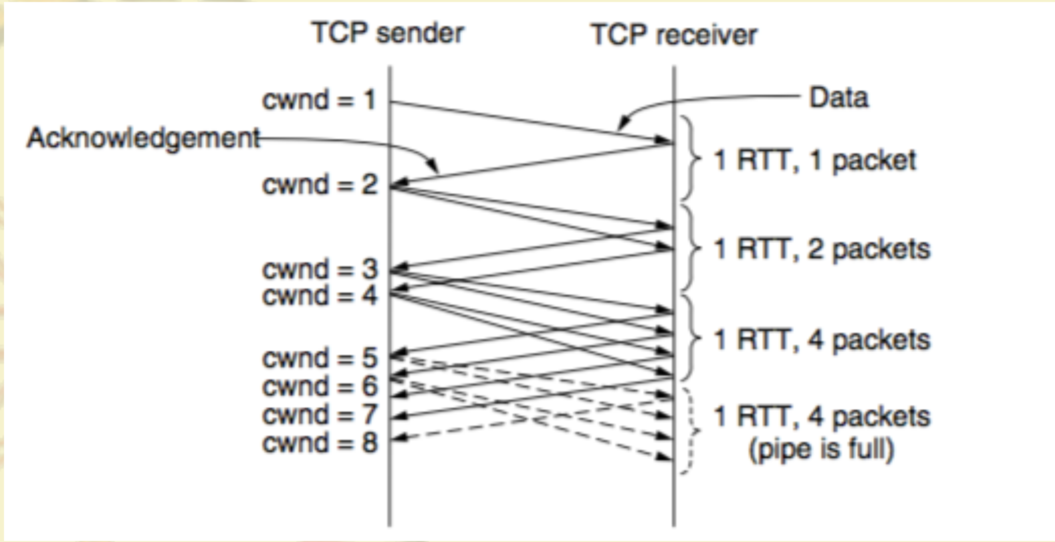
- A 10 Mbps link with 100 ms RTT
  - Appropriate CWnd = BDP = 1 Mbit
  - 1250 byte packets -> 100 packets to reach BDP
  - CWnd starts at 1 packet, and increased 1 packet at every RTT
  - 100 RTTs are required 10 sec before the connection reaches to a moderate rate

## Increase Rate Exponentially at the Beginning – The Slow Start

- Slow Start - Exponential increase of rate to avoid slow convergence
  - Rate is not slow at all ! 
  - CWnd is doubled at every RTT

# TCP Slow Start

- Every ACK segment allows two more segments to be sent
- For each segment that is acknowledged before the retransmission timer goes off, the sender adds one segment's worth of bytes to the congestion window.



**Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell**

## Slow Start Threshold

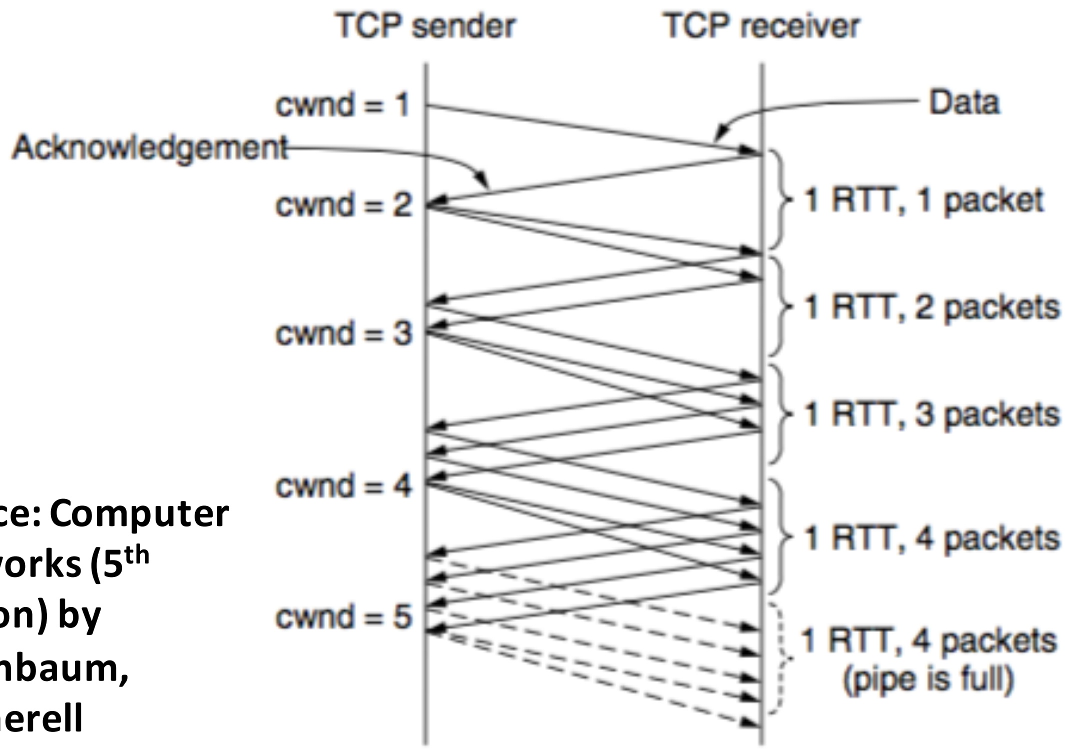
- Slow start causes exponential growth, eventually it will send too many packets into the network too quickly.
- To keep slow start under control, the sender keeps a threshold for the connection called the **slow start threshold (ssthresh).**
- Initially ssthresh is set to BDP (or arbitrarily high), the maximum that a flow can push to the network.
- Whenever a packet loss is detected by a RTO, the ssthresh is set to be half of the congestion window

## Additive Increase (Congestion Avoidance)

- Whenever ssthresh is crossed, TCP switches from slow start to additive increase.
- Usually implemented with an partial increase for every segment that is acknowledged, rather than an increase of one segment per RTT.
- A common approximation is to increase Cwnd for additive increase as follows:

$$\boldsymbol{CWnd = Cwnd + \frac{MSS \times MSS}{CWnd}}$$

## Additive Increase – Packet Wise Approximation



Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

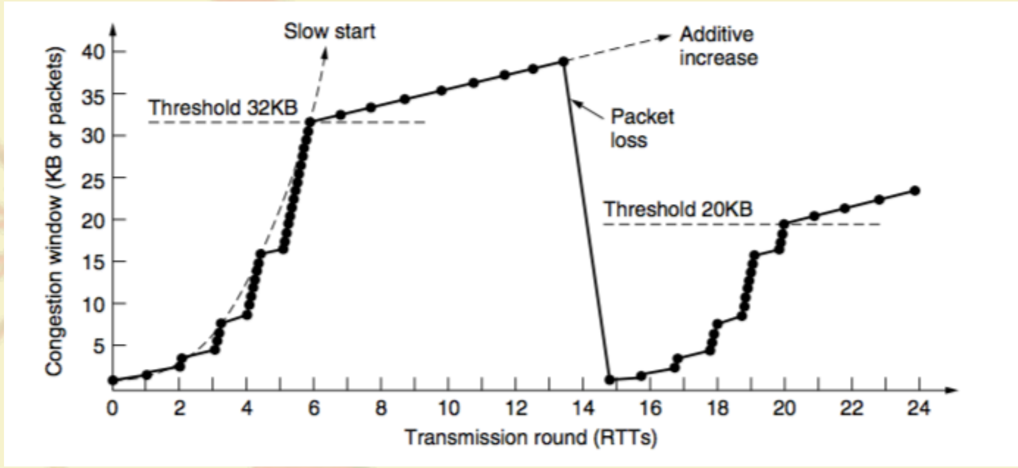
## Triggering an Congestion

- Two ways to trigger a congestion notification in TCP – (1) RTO, (2) Duplicate ACK
- **RTO**: A sure indication of congestion, however time consuming
- **Duplicate ACK:** Receiver sends a duplicate ACK when it receives out of order segment
  - **A loose way of indicating congestion**
  - **TCP arbitrarily assumes that THREE duplicate ACKs (DUPACKs) imply that a packet has been lost – triggers congestion control mechanism**
  - The identity of the lost packet can be inferred – **the very next packet in sequence**
  - **Retransmit the lost packet and trigger congestion control**

## Fast Retransmission – TCP Tohoe

- Use THREE DUPACK as the sign of congestion

- Once 3 DUPACKs have been received,
  - Retransmit the lost packet (**fast retransmission)** – takes one RTT
  - Set ssthresh as half of the current CWnd
  - Set CWnd to 1 MSS



**Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell**

## Fast Recovery – TCP Reno

- **Once a congestion is detected through 3 DUPACKs, do TCP really need to set CWnd = 1 MSS ?**

- DUPACK means that **some segments are still flowing in the network** – a signal for temporary congestion, but not a prolonged one

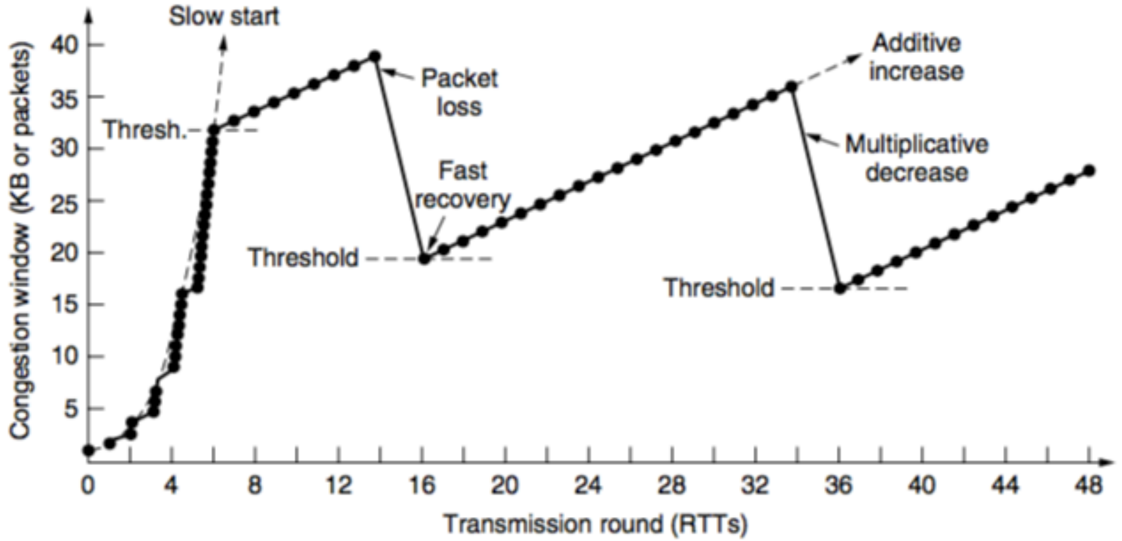
- Immediately transmit the lost segment (**fast retransmit)**, then transmit additional segments based on the DUPACKs received **(fast recovery)**

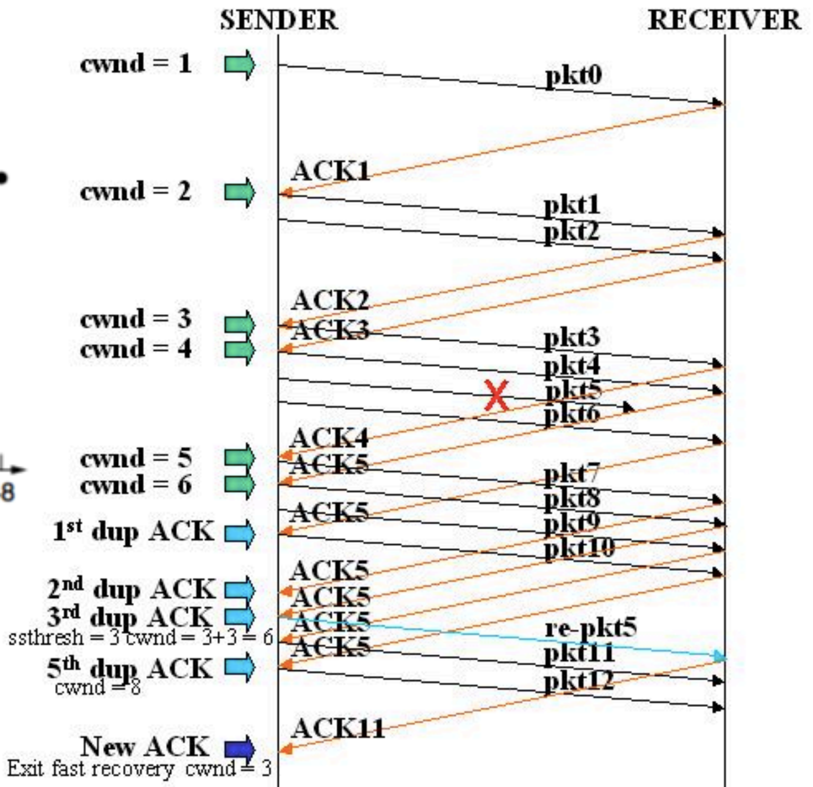
## Fast Recovery – TCP Reno

**Fast recovery:**

1. set ssthresh to one-half of the current congestion window. Retransmit the missing segment.
2. set cwnd = ssthresh + 3.
3. Each time another duplicate ACK arrives, set cwnd = cwnd + 1. Then, send a new data segment if allowed by the value of cwnd.
4. Once receive a new ACK (an ACK which acknowledges all intermediate segments sent between the lost packet and the receipt of the first duplicate ACK), exit fast recovery.
    - This causes setting cwnd to ssthresh (the ssthresh in step 1). Then, continue with linear increasing due to congestion avoidance algorithm.

# Fast Recovery – TCP Reno





thank you!