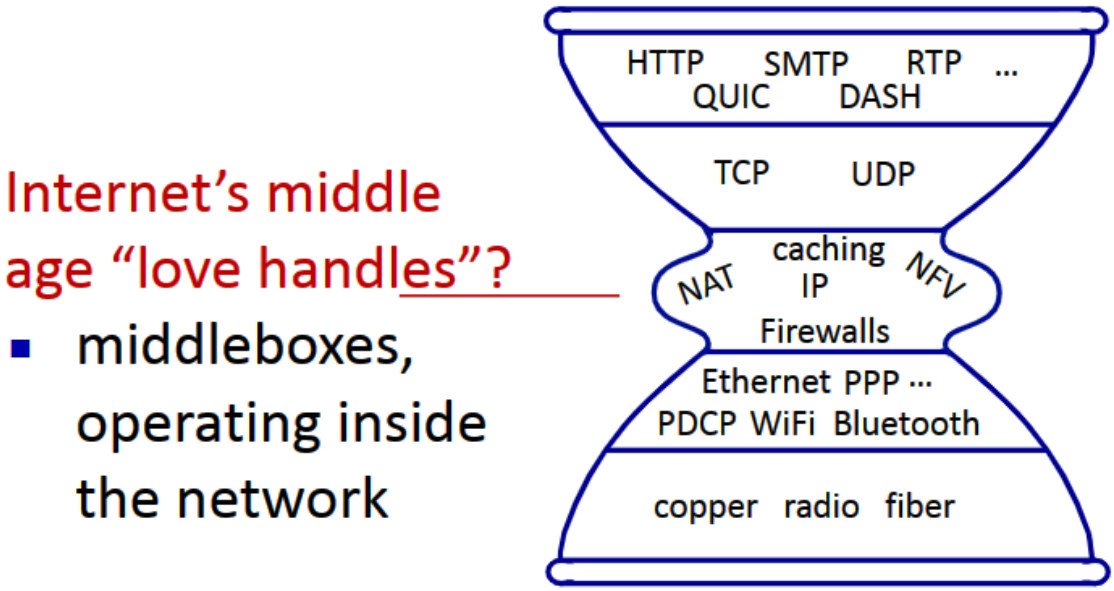
CSEE4119 Computer Networking Final Exam Revision

This document contains all the important knowledge point in: (1) Homework 1-5 (2) Midterm Exam (3) Sample Midterm Exam

General Knowledge



Application Layer:

- Communicate with the application
- Protocols: HTTP, SMTP, DNS.....

Transport layer:

- Get service from network layer, and provide service to application layer
  - Receive message from application layer, then do segmentation, add header, finally send messages to network layer.
  - Receive message from network layer, then reassembles segmented data, read header to identify the port, forwards message to application layer with specific port.
- Protocols: TCP, UDP

Network layer:

- Transport packets form source to destination.
  - Receive packets from transport layer, and encapsulates in a datagram. Then push the datagram to link layer.
  - Receive packets and decapsulates datagram, send packets to appropriate transport layer.
- Protocols: IP

Link layer:

- Node-to-node delivery of data. Encode, decode and organize the outgoing and incoming data
- Protocols: PPP, PDCP, Wi-Fi, Bluetooth

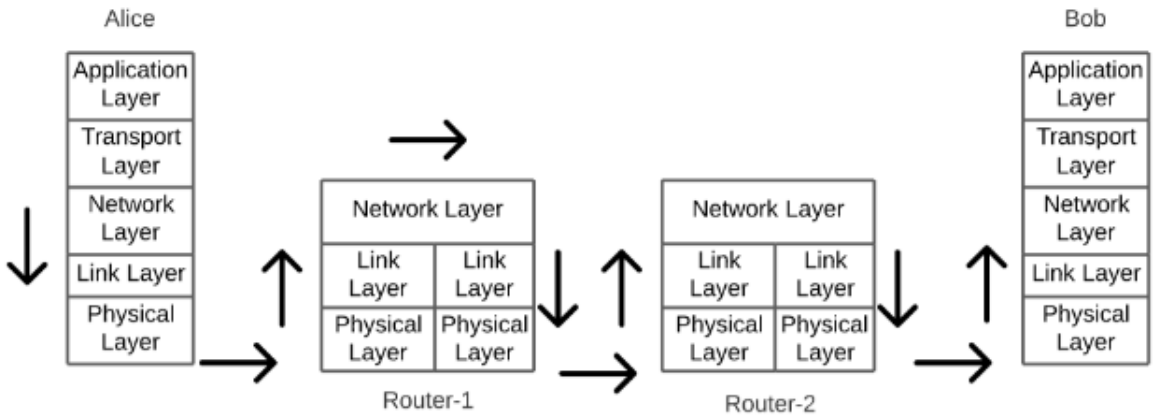
Homework 1

Protocol Layers and Service Models

Layered Internet Architecture:

- Advantage:
  - Has great compatibility between devices, systems and networks.
  - Keep each layer isolated, so allow change the implementation of a service without affecting other components.
  - Does not require developer know all the information about the whole model.
  - Easy to troubleshooting.
- Disadvantage:
  - Each layer will create more overhead since they all add data to the packet.
  - Can make useful information in a certain layer invisible and inaccessible to other layers.
  - One layer may duplicate lower-layer functionality

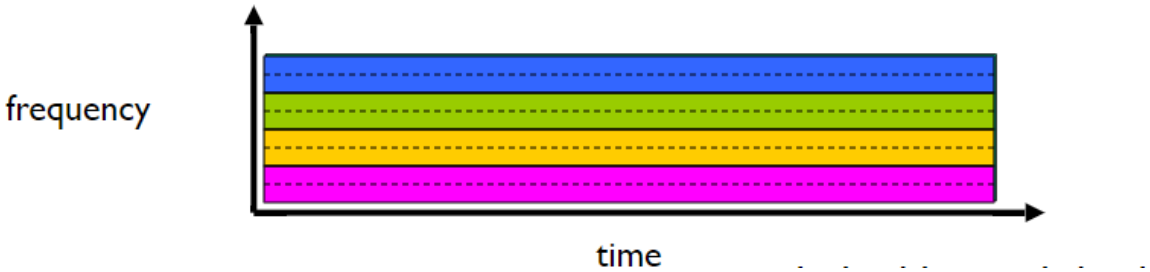
Encapsulation:



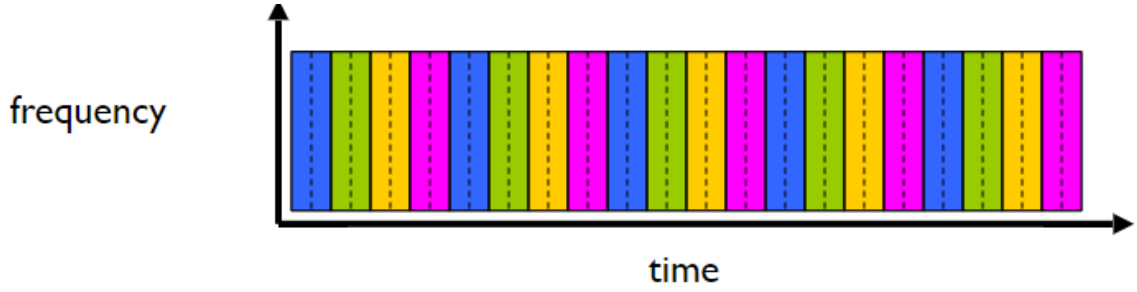
Circuit & Packet Switching

Circuit Switching:

- FDM (Frequency Division Multiplexing) 频分复用



- TDM (Time Division Multiplexing) 时分复用 (时间切片)



Packet Switching:

The reliability of the link could be calculated as the equation below.

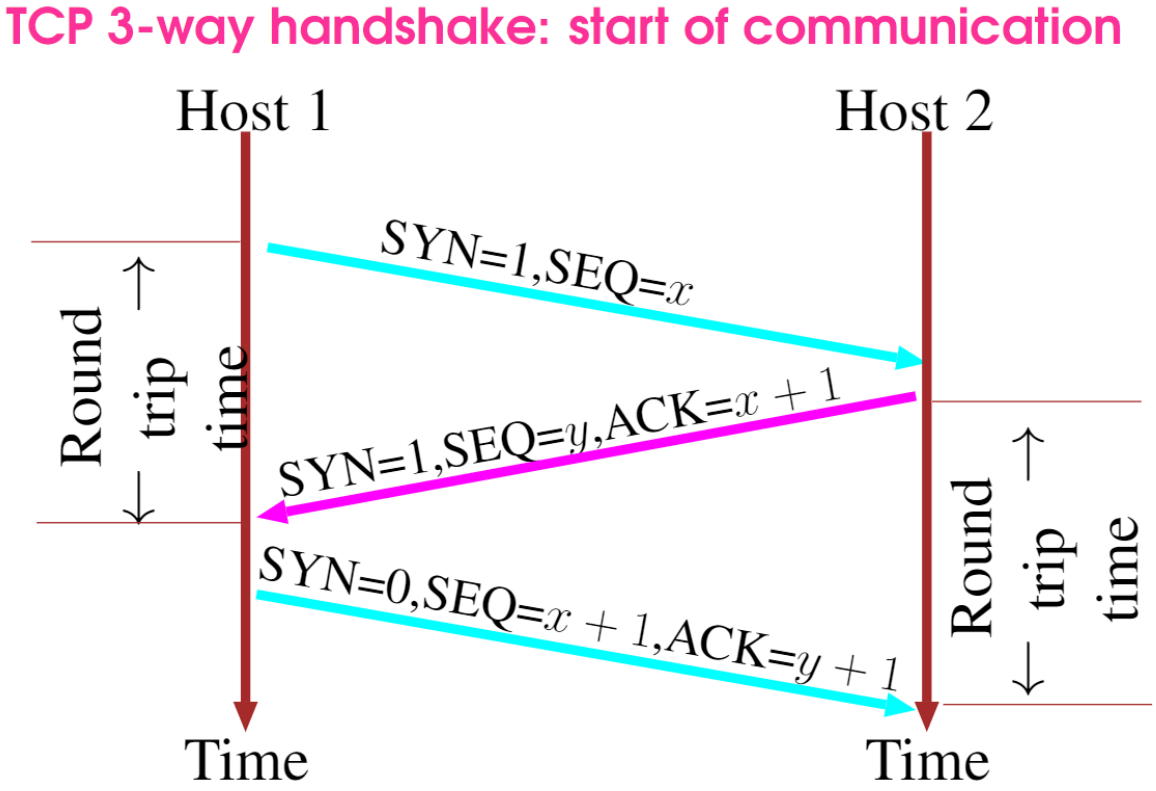
$$\sum_{i=0}^{i=T} \binom{N}{i} p^i (1-p)^{N-i}$$

Where N is the total number of user, i is the number of concurrent user, p is the active time percentage.

Homework 2

HTTP

- **UDP**:
  Provides: Unreliable data transfer.
  Do not provide: reliable, flow control, congestion control, throughput guarantee, security and connection setup.
- **TCP**:
  Provides: Reliable transport, flow control, congestion control.
  Do not provide: timing, minimum throughput guarantee, security.
- **Round Trip Time (RTT) definition**:
  Time for a small packet to travel client to server and back



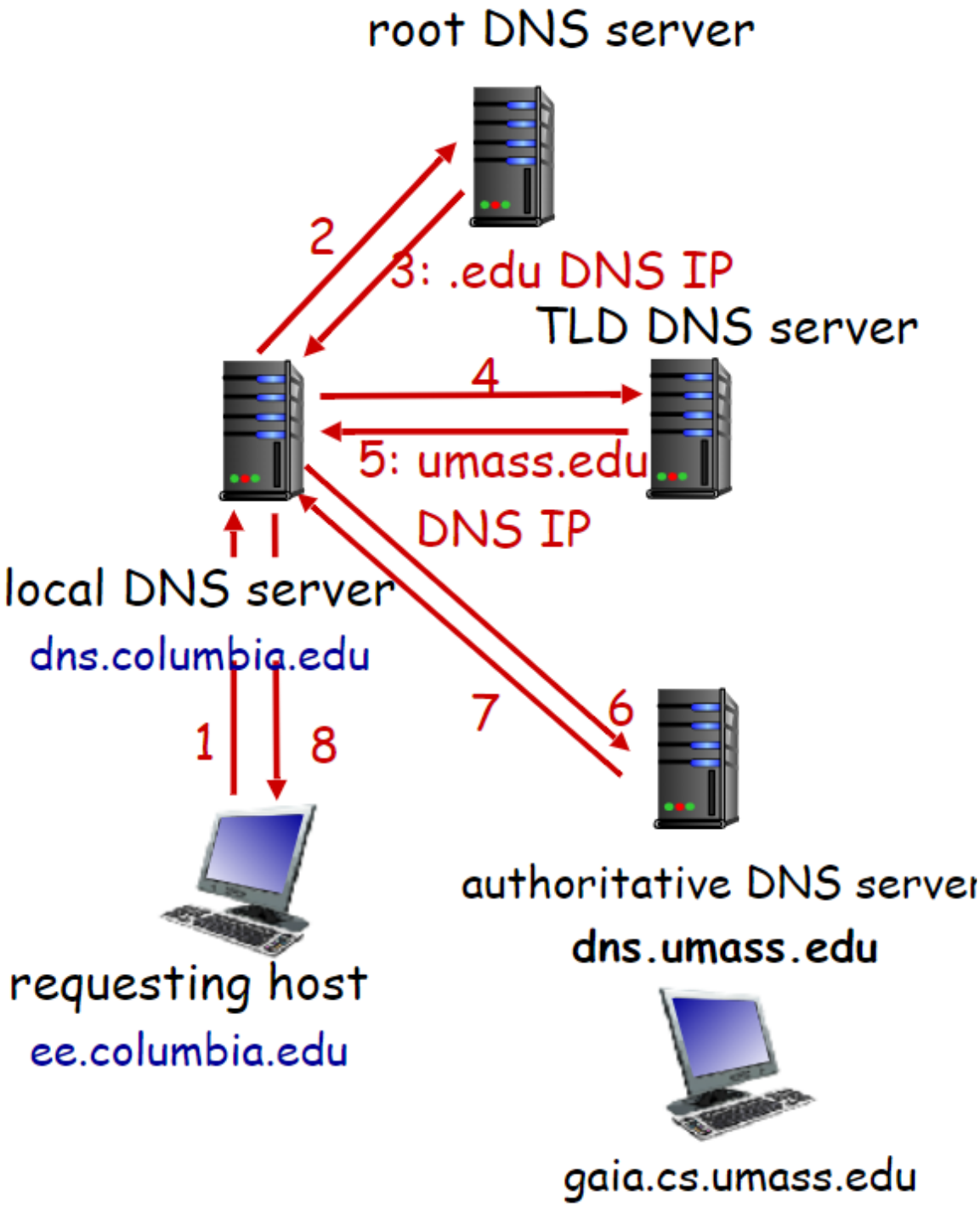
- **Non-persistent HTTP 非持久性HTTP**:
  At most one object sent over TCP connection, connection then closed. Downloading multiple objects required multiple connections
- **Persistent HTTP 持久性HTTP**:
  Multiple objects can be sent over single TCP connection between client and server
- **Non-persistent HTTP issues:**
  - requires 2RTT per object
  - OS overhead for each TCP connection 每个TCP链接都会消耗OS的资源
  - browser often open parallel TCP connection to fetch referenced objects
- **Persistent HTTP:**
  - server leaves connection open after sending response
  - subsequent HTTP messages between same client/server sent over connection
  - client sends requests as soon as it encounters a referenced object
  - as little as one RTT for all the reference objects

DNS and CDN

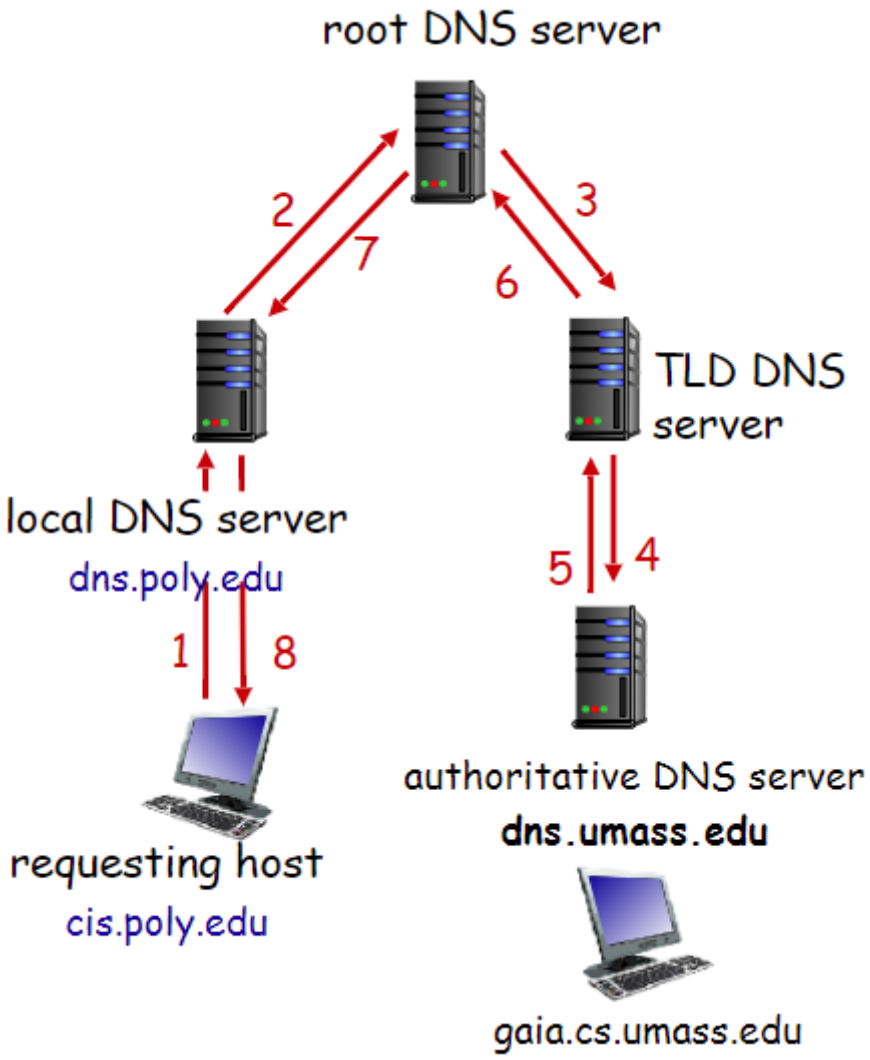
Why DNS use UDP:

- The DNS query and response message are shot and they can fit in one UDP segment
- UDP can avoid handshake
- Using UDP can increase performance and decrease latency

Iterative query:



Recursive query:



CDN

Anycast Pros:
CDN inherently allows content to be served near users, allows lower latency and more throughput.
Increase robustness since if one of the server down, other servers still available.
Anycast has lower fail over time. Provide better DDOS protection. Simple to configure

Anycast Cons:
The server that user actually use is at the mercy of other networks/BGP.
Harder to control or predict which site that users will takes, difficult to load balancing.
Hard to perform off-net deployment.

DNS redirection Pros:
CDN inherently allows content to be served near users, allows lower latency and more throughput.
Increase robustness since if one of the server down, other servers still available.
Allows the CDN to control with group area of the user should go to which PoP to assign server with the load balancing or RTT.
Allow off-net server deployment.

DNS redirection Cons:

DNS records and TTL may not be respected. The user might be far away from the recursive resolver.

Longer failover time and worse protection for DDOS.

The DNS system can be complex.

> Anycast特征：从不同的地理位置被重定向到不同的IP
>
> DNS Redirection特征：不同的地理位置被重定向到相同的IP，或只有少数几个不同的IP供切换

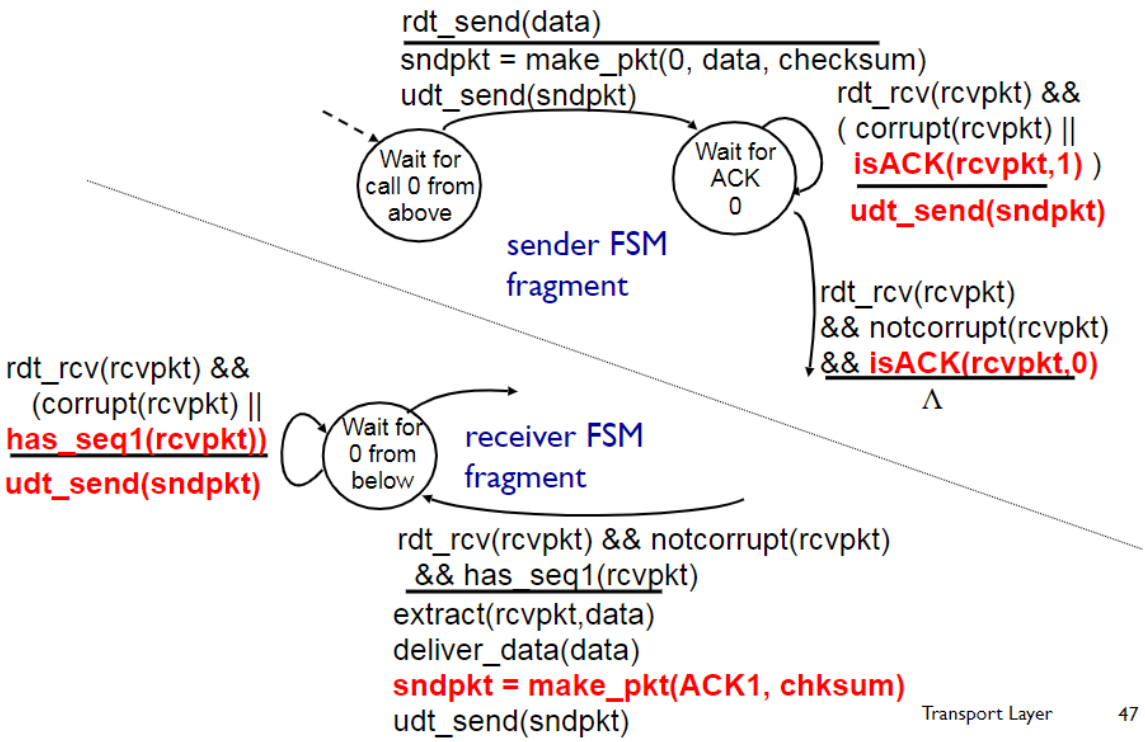
Homework 3

Reliable Data Transfer (rdt)

- Incrementally develop sender, receiver sides of Reliable Data Transfer protocol (rdt)
- Consider only unidirectional data transfer
  - but control info will flow on both directions
- Use finite state machines (FSM) to specify sender and receiver

rdt 2.2: a NAK-free protocol

- Same functionality as rdt2.1, using ACKs only
- Instead of NAK, receiver sends ACK for last packet received OK
  - receiver must explicitly include sequence number of packet being ACKed
- Duplicate ACK at sender results in same action as NAK: retransmit current packet



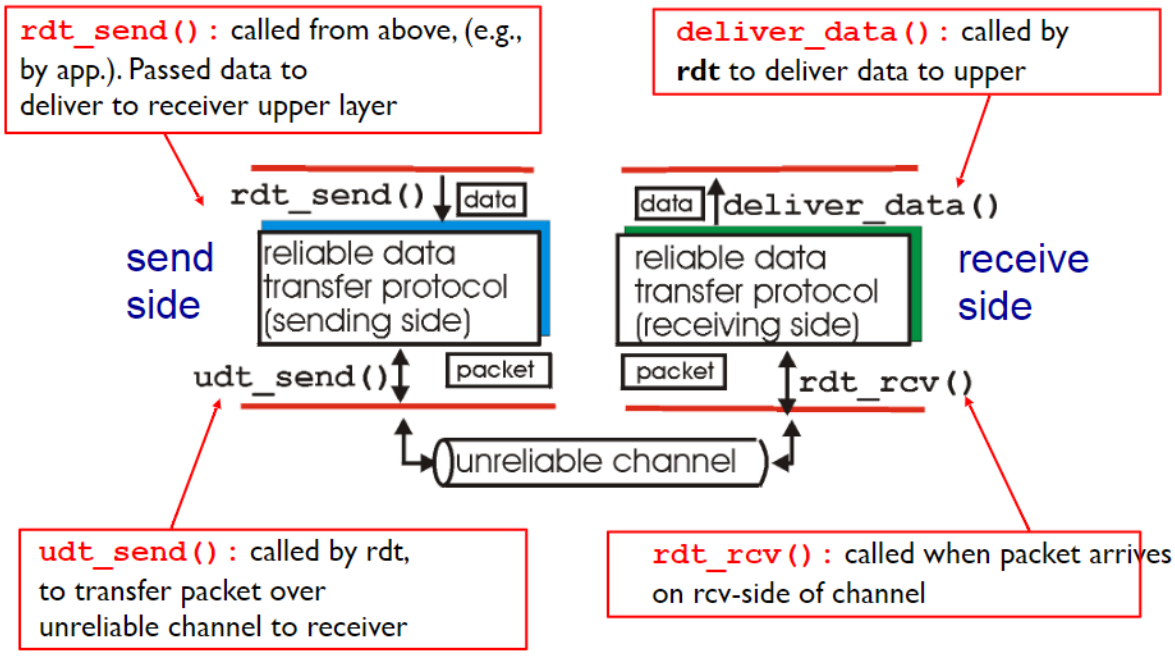
> 若receiver收到了受损数据，则不发送当前数据的NAK，而是发送上一个无损数据的ACK，以达到同样的效果。

sender收到上一个数据的两次ACK，便能知道当前的数据未发送成功

rdt 3.0: channels with errors and loss

New assumption:

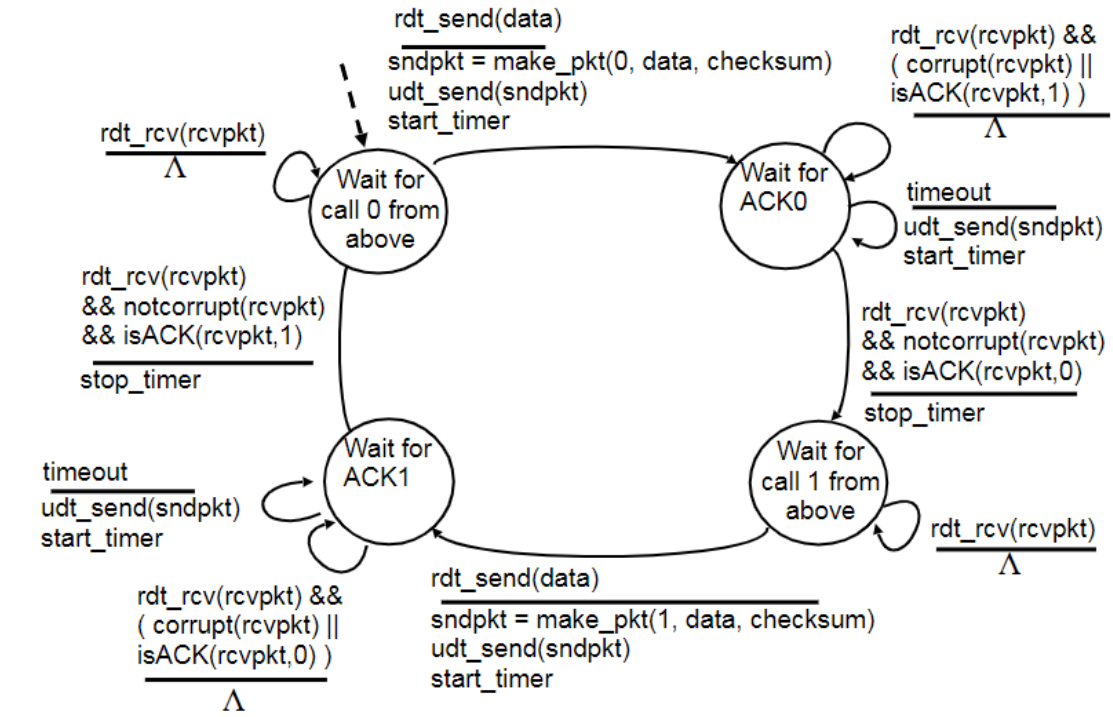
- **underlying channel** can also loss packets (data, ACKs)
  - checksum, sequence number, ACKs, retransmission will help, but not enough

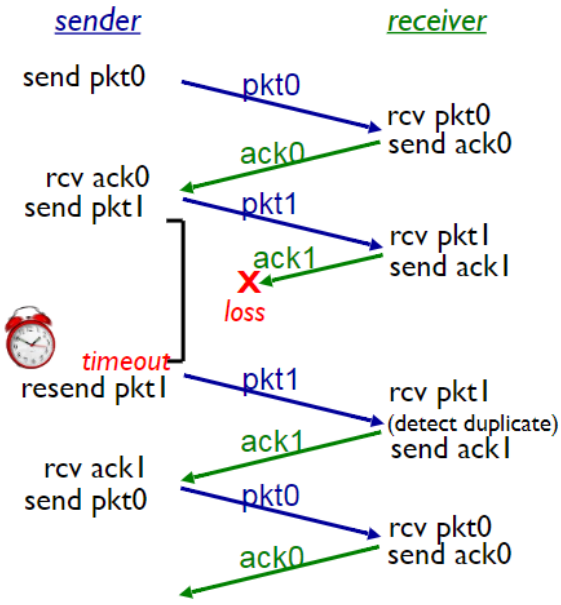


Approach: sender waits 'reasonable' amount of time for ACK

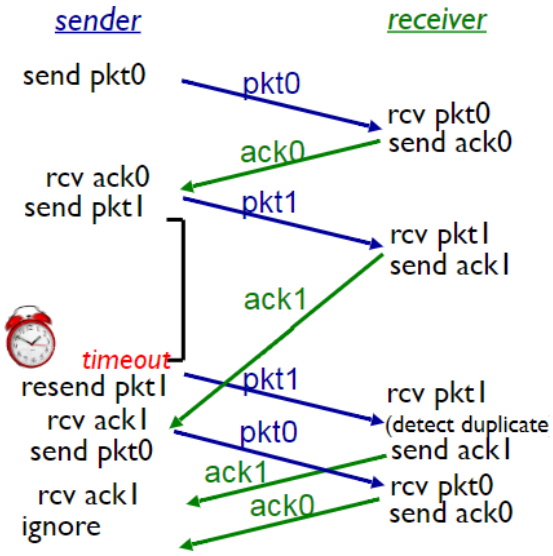
- retransmits if no ACK received in this time
- if packet (or ACK) just delayed (not lost)
  - retransmission will be duplicate, but sequence number is already handled this (in rdt 2.1)
  - receiver must specify sequence number of packet being ACKed
- requires countdown timer

rdt3.0 sender



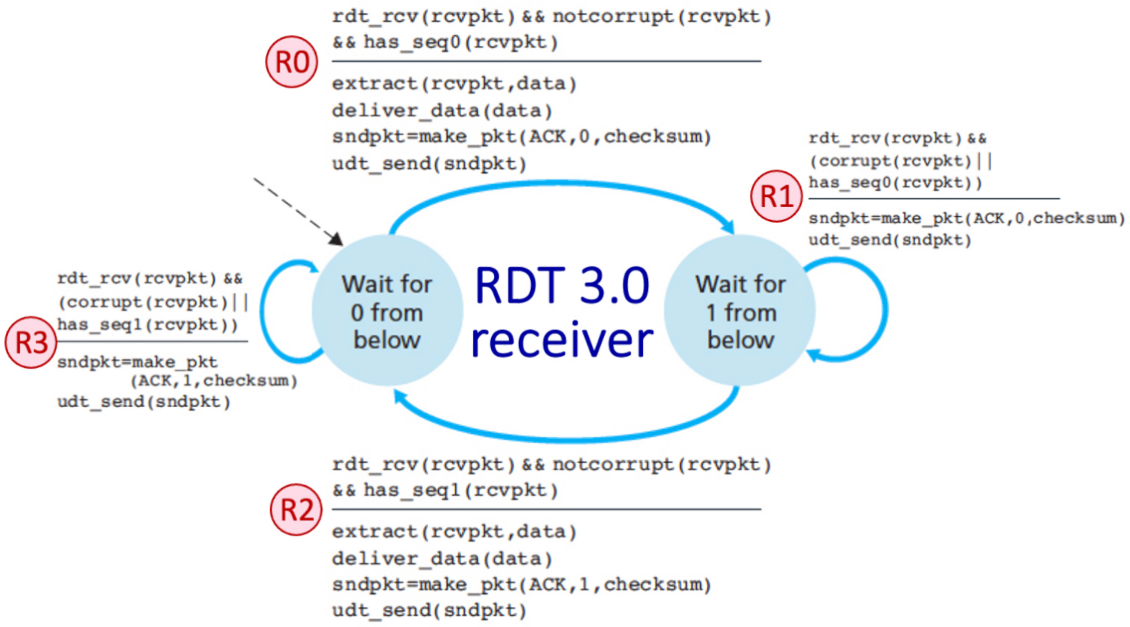


(c) ACK loss

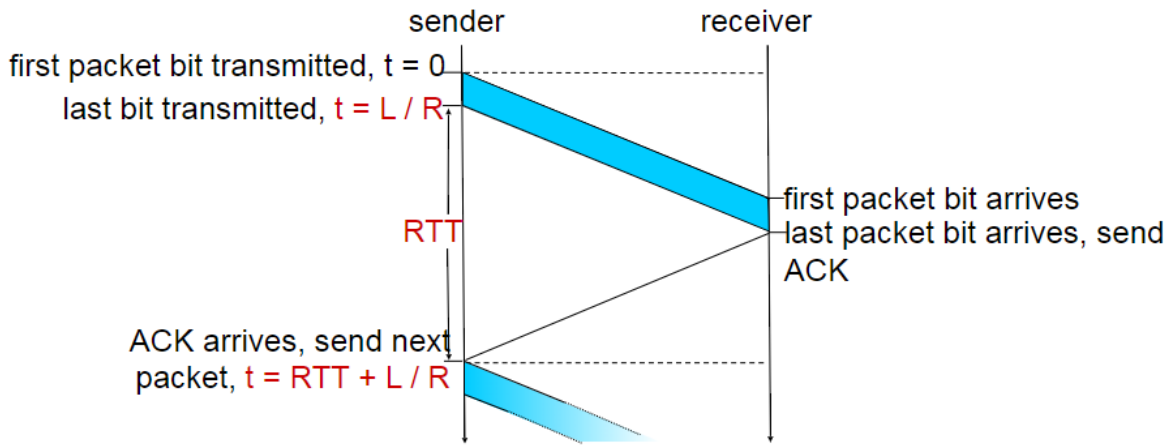


(d) premature timeout/ delayed ACK

rdt3.0 receiver



rdt3.0: stop-and-wait operation



L = packet size, R = transmission rate, RTT = Round Trip Time

Utilisation

- rdt3.0 is correct, but not enough in performance
- For example: for 1 Gbps link, 15 ms propagation delay, sending 8000 bit packet:
  - $D_{trans} = \frac{L}{R} = \frac{8000\ bits}{10^9\ bits/sec} = 0.008\ ms$
  - U_{sender}: utilisation–fraction of time sender busy sending
    $U_{sender} = \frac{L/R}{RTT+L/R} = \frac{0.008}{30.008} = 0.00027$, where RTT should be twice of the propagation delay
  - if RTT = 30 ms, 1 KB packet every 30 ms: 33KB/sec throughput over 1 Gbps link.

GBN and selective repeat

Go-Back-N:

- sender can have up to N unACKed packets in pipeline
- receiver only sends **cumulative ack**

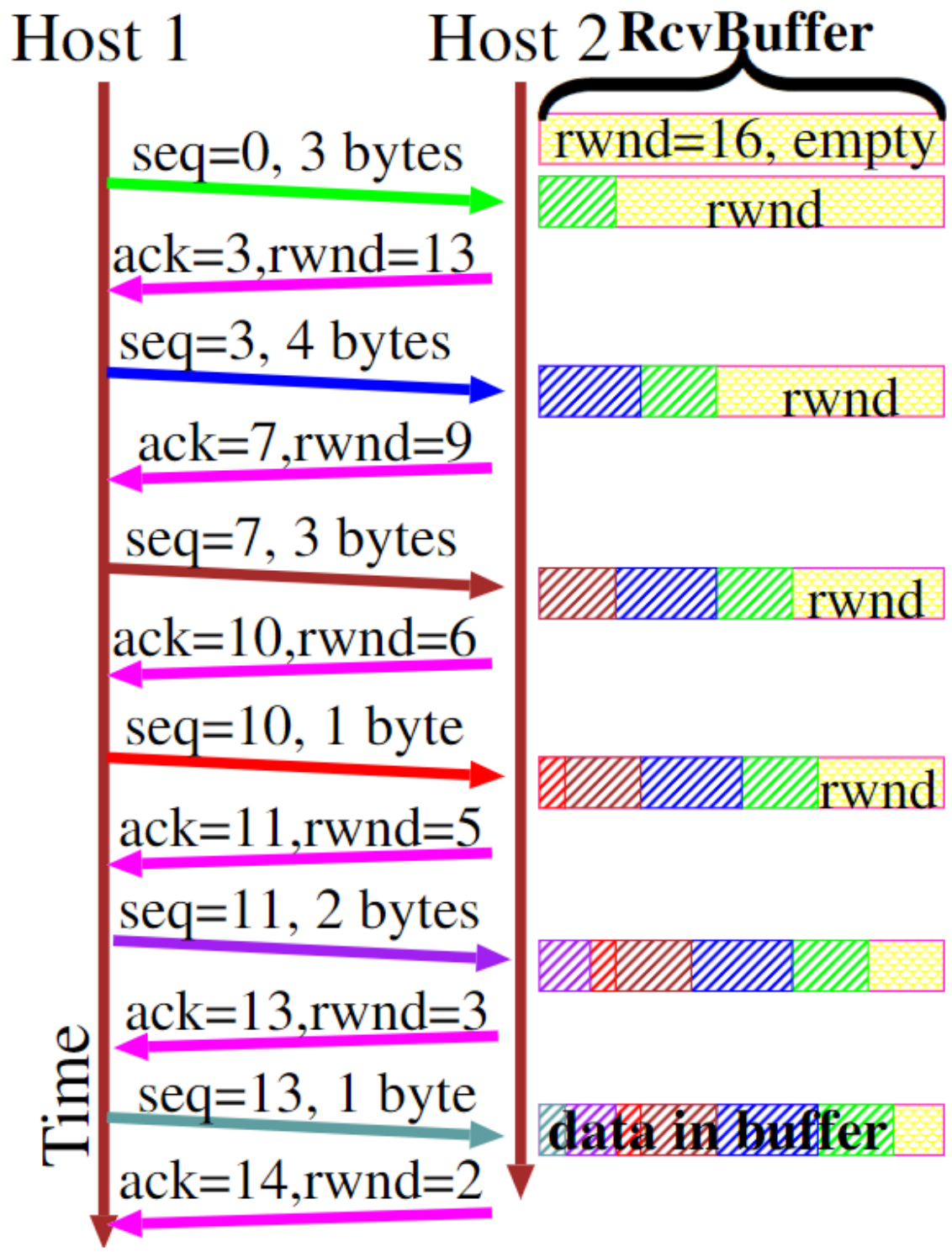
- sender has timer for oldest unacked packet, when timer expires, retransmit **all** unacked packet
- It is possible for the **sender** to receive an ACK for a packet that **fails outside of its current window**.
- It is **not possible** for the sender to have a window that is **further advanced than the receiver's expected seqnum**.

Selective Repeat:

- Sender can have up to N unacked packets in pipeline
- Receiver sends **individual ack** for each packet
- Sender has timer for **each unacked packet**, when timer expires, retransmit **the unacked packet only**
- It is possible for the **sender** to send a packet that **fails outside of receiver's window**
- It is possible for the **sender** to receive an ACK for a packet that **fails outside of its current window**.

TCP Flow Control

- TCP receive window buffer `rwnd` to deal with the size of the bucket (free buffer space)



TCP Congestion Control

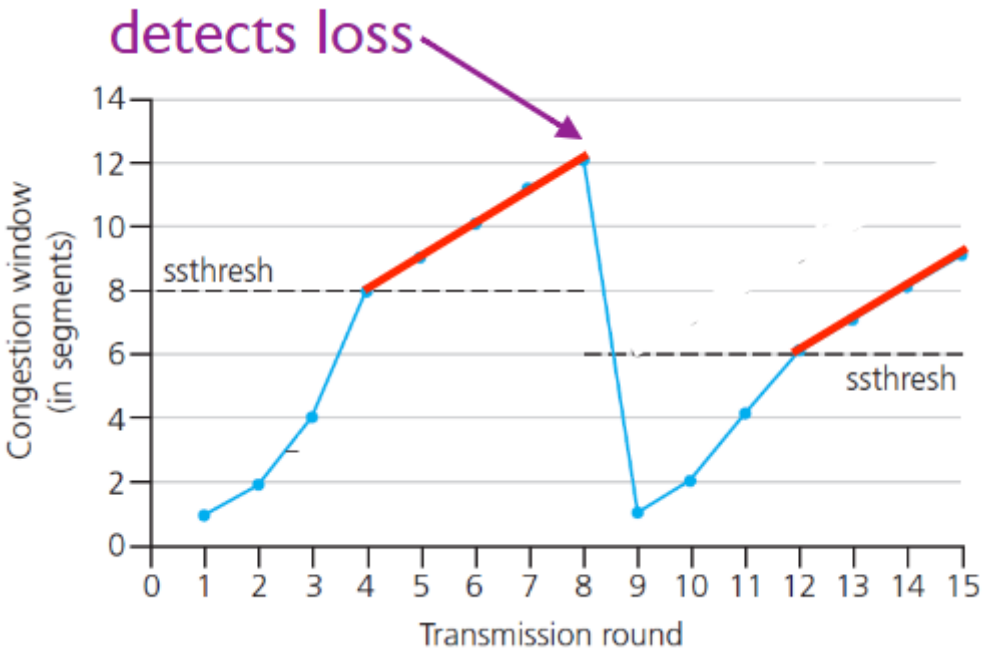
- TCP congestion window buffer `cwnd` includes the packets that are sent but not acked.
  - deal with the narrow part of the pipe(Transmission network)

Slow Start:

- initially `cwnd = 1MSS`
  - MSS -> maximum segment size, usually 1
- double `cwnd` in every RTT
- if loss packet, `cwnd = 1MSS`

Congestion avoidance

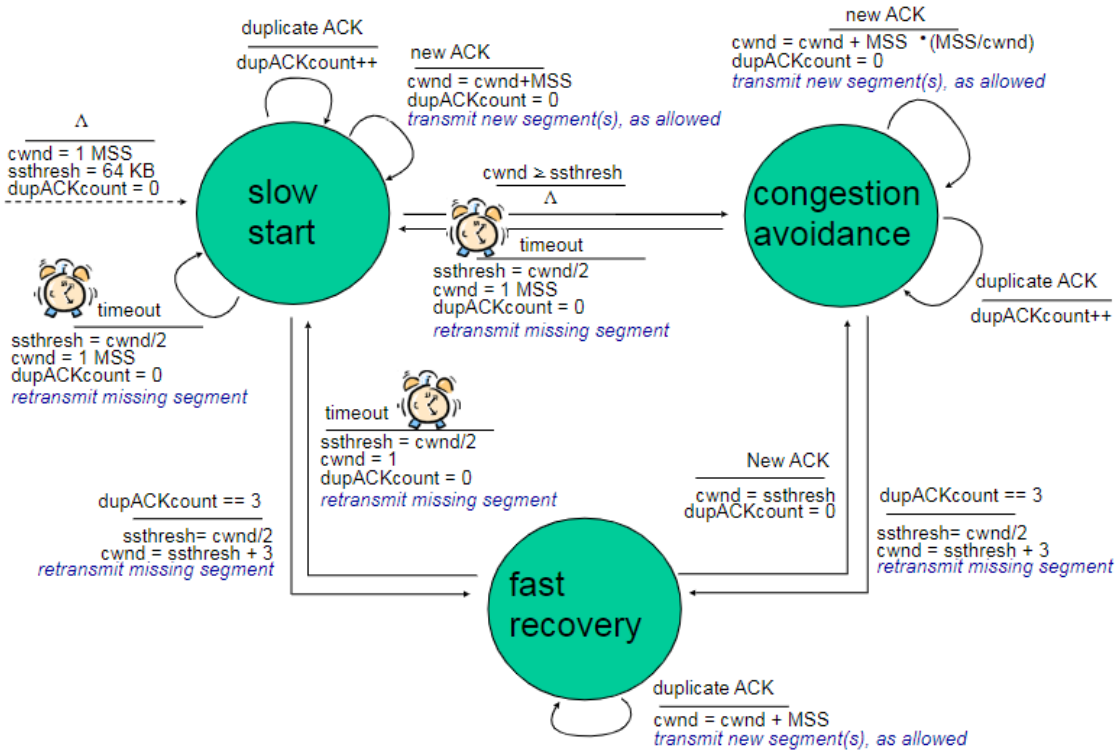
- If `cwnd >= ssthresh`
  - `ssthresh` is the threshold created by a variant number
- Then change the mode to congestion avoidance, for each RTT, `cwnd = cwnd+1`
- Whenever the mode is, if packet loss occurs:
  - `ssthresh = cwnd / 2`
  - `cwnd = 1`
  - switch mode to slow start



Reno's fast recovery

- If **3 duplicate ACKs** detected
  - `ssthresh = cwnd / 2`
  - `cwnd = ssthresh + 3`
  - **REMAIN THE CONGESTION MODE**

> `cwnd` 变为 `ssthresh + 3` 中，3的意义是acknowledge从发送方发出的3个duplicate ACK


duplicate ACK
dupACKcount++
new ACK
cwnd = cwnd+MSS
dupACKcount = 0
transmit new segment(s), as allowed
new ACK
cwnd = cwnd + MSS • (MSS/cwnd)
dupACKcount = 0
transmit new segment(s), as allowed
Λ
cwnd = 1 MSS
ssthresh = 64 KB
dupACKcount = 0
slow start
cwnd ≥ ssthresh
Λ
timeout
ssthresh = cwnd/2
cwnd = 1 MSS
dupACKcount = 0
retransmit missing segment
congestion avoidance
timeout
ssthresh = cwnd/2
cwnd = 1 MSS
dupACKcount = 0
retransmit missing segment
duplicate ACK
dupACKcount++
timeout
ssthresh = cwnd/2
cwnd = 1
dupACKcount = 0
retransmit missing segment
New ACK
cwnd = ssthresh
dupACKcount = 0
dupACKcount == 3
ssthresh= cwnd/2
cwnd = ssthresh + 3
retransmit missing segment
dupACKcount == 3
ssthresh= cwnd/2
cwnd = ssthresh + 3
retransmit missing segment
fast recovery
duplicate ACK
cwnd = cwnd + MSS
transmit new segment(s), as allowed

Homework 4