Computer Networks Homework 2
TCP Congestion Control

Date: 2015/11/25

Instructor: **Prof. Ai-Chun Pang**

TA: **Guan-Yu Chen**
Ming-Fan Chen

Introduction

Introduction

- Target
  - Application layer reliable transfer / congestion control
  - Implement TCP by **UDP**
  - Socket programming

UDP	TCP
Unreliable Unordered delivery	Reliable In-order delivery Congestion control

Introduction

- Architecture
  - Sender / Agent / Receiver

Introduction

- Sender / Receiver
  - Send / receive file by UDP
  - Provide reliable transmission
  - Congestion control

- Agent
  - Forward data & ACK packets
  - Randomly drop data packet
  - Compute loss rate

Requirement

Requirement (1/7)

- Reliable transmission
  - Data & ACK
  - Time out & retransmission
  - Sequence number
  - Completeness and correctness of transmitted file

Requirement (2/7)

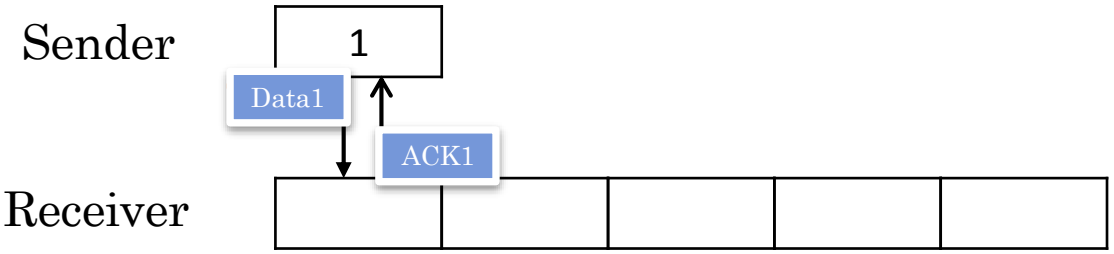
- Congestion control [**Sender side**]
  - Slow start
    - Send single packet in the beginning
    - When below the threshold, congestion window increase exponentially until packet loss, i.e., $1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow \ldots$
    - When larger than or equal to the threshold, congestion window increase linearly until packet loss, i.e., $16 \rightarrow 17 \rightarrow 18 \rightarrow \ldots$
  - Packet loss / time out
    - Set threshold to $\max(\left\lfloor \frac{Congestion\ Window}{2} \right\rfloor, 1)$
    - Set congestion window to 1
    - Retransmit
      - From the first "un-ACKed" packet

Requirement (3/7)

- Buffer handling [**Receiver side**]
  - Buffer overflow
    - Drop packet if "out of range" of buffer
  - Flush(write) to the file
    - Only when both buffer overflows and all packets in range are received

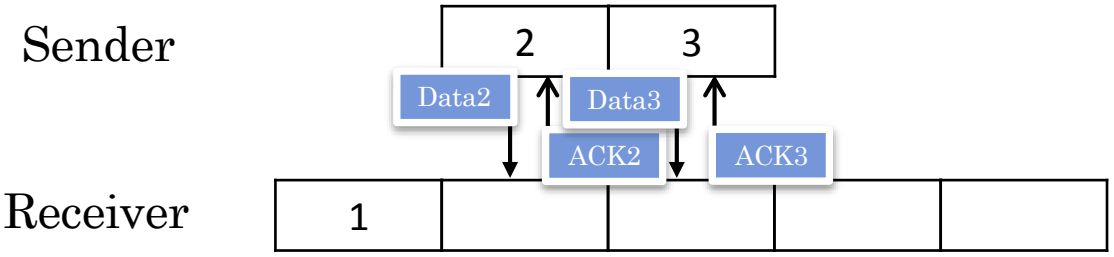
Requirement (3/7)

- Example
  - Sender sends Data 1
    - Congestion window = 1; Threshold = 2
  - Receiver sends ACK 1



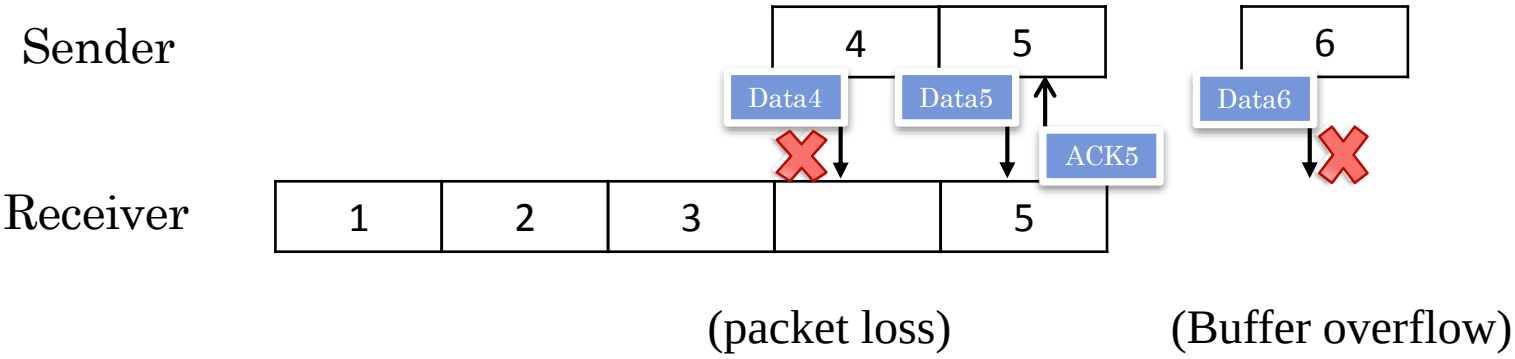
Requirement (3/7)

- Example
  - Sender sends Data 2, 3
    - Congestion window = 2; Threshold = 2
  - Receiver sends ACK 2, 3



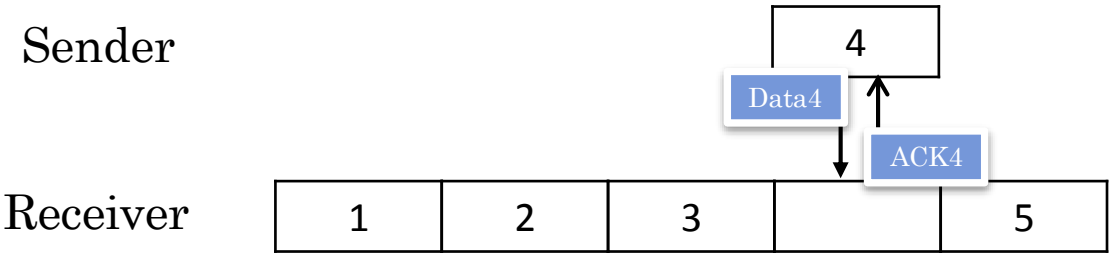
Requirement (3/7)

- Example
  - Sender sends Data 4, 5, 6
    - Congestion window = 3; Threshold = 2
  - Receiver sends ACK 5, drops Data 6



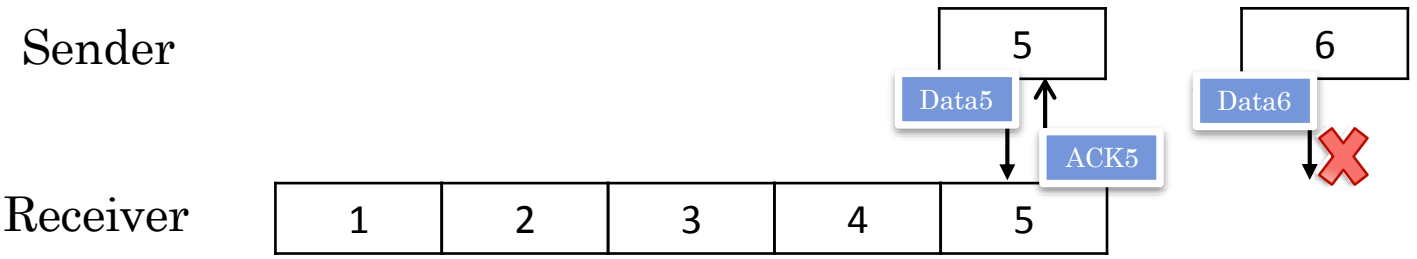
Requirement (3/7)

- Example
  - Sender sends Data 4
    - Congestion window = 1; Threshold = 1
  - Receiver sends ACK 4



Requirement (3/7)

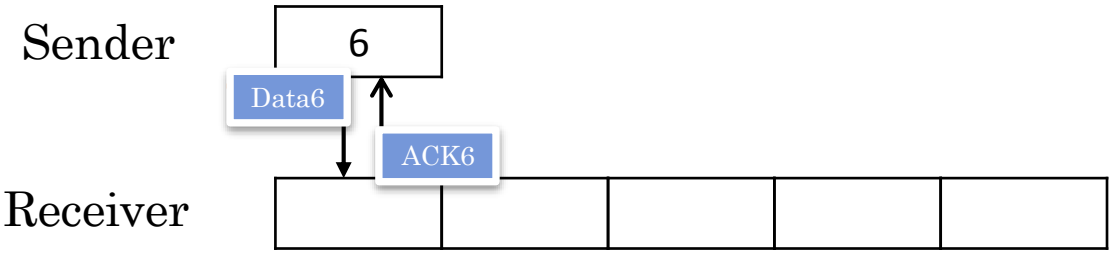
- Example
  - Sender sends Data 5, 6
    - Congestion window = 2; Threshold = 1
  - Receiver sends ACK 5, drops Data 6, flush buffer (to file)



(Buffer overflow)

Requirement (3/7)

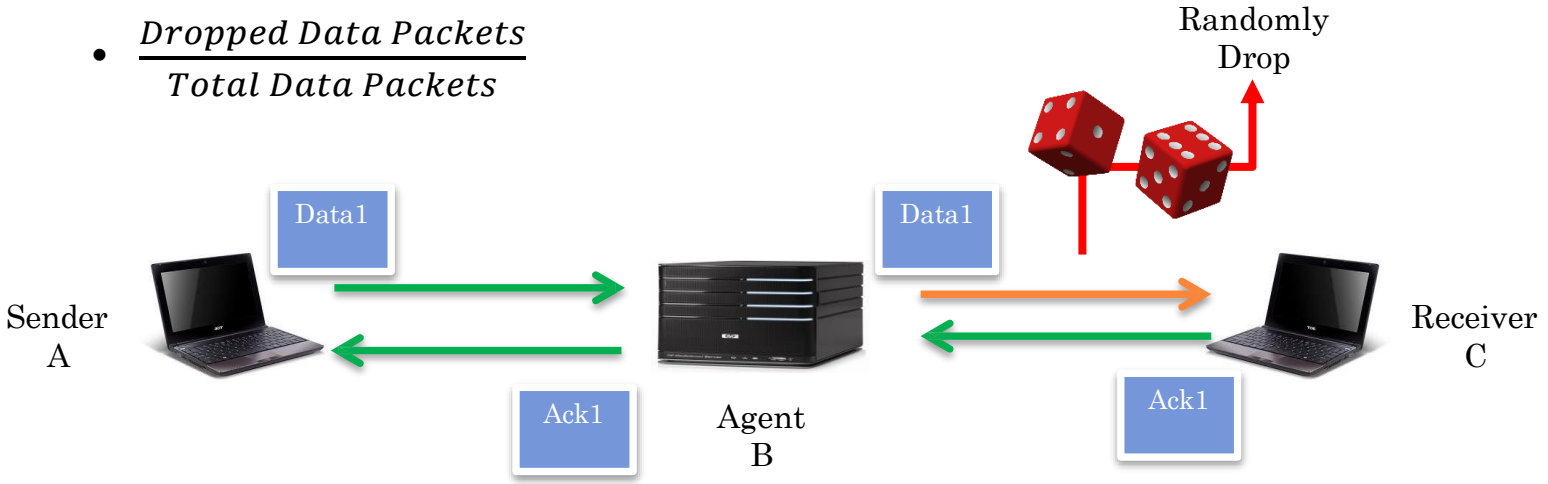
- Example
  - Sender sends Data 6
    - Congestion window = 1; Threshold = 1
  - Receiver sends ACK 6



And so on …….

Requirement (4/7)

- Agent
  - Forward data & ACK packets
  - Randomly drop data packet [DO NOT DROP ACK PACKETS]
  - Compute loss rate
    - $\frac{Dropped\ Data\ Packets}{Total\ Data\ Packets}$



Requirement (5/7)

- Show message
  - Sender
    - send, recv, data, ack, fin, finack, sequence number, time out, resnd, winSize, threshold
  - Receiver
    - send, recv, data, ack, fin, finack, sequence number, ignr, drop, flush
  - Agent
    - get, fwd, data, ack, fin, finack, sequence number, drop, loss rate

- Show message

```
send    data    #1,     winSize = 1
recv    ack     #1
send    data    #2,     winSize = 2
send    data    #3,     winSize = 2
recv    ack     #2
recv    ack     #3
send    data    #4,     winSize = 3
send    data    #5,     winSize = 3
send    data    #6,     winSize = 3
recv    ack     #5
time    out,            threshold = 1
resnd   data    #4,     winSize = 1
recv    ack     #4
resnd   data    #5,     winSize = 2
resnd   data    #6,     winSize = 2
recv    ack     #5
time    out,            threshold = 1
resnd   data    #6,     winSize = 1
recv    ack     #6
send    fin
recv    finack
```

Sender

```
get     data    #1
fwd     data    #1,     loss rate = 0.0000
get     ack     #1
fwd     ack     #1
get     data    #2
fwd     data    #2,     loss rate = 0.0000
get     data    #3
fwd     data    #3,     loss rate = 0.0000
get     ack     #2
fwd     ack     #2
get     ack     #3
fwd     ack     #3
get     data    #4
drop    data    #4,     loss rate = 0.2500
get     data    #5
fwd     data    #5,     loss rate = 0.2000
get     data    #6
fwd     data    #6,     loss rate = 0.1667
get     ack     #5
fwd     ack     #5
get     data    #4
fwd     data    #4,     loss rate = 0.1429
get     ack     #4
fwd     ack     #4
get     data    #5
fwd     data    #5,     loss rate = 0.1250
get     data    #6
fwd     data    #6,     loss rate = 0.1111
get     ack     #5
fwd     ack     #5
get     data    #6
fwd     data    #6,     loss rate = 0.1000
get     ack     #6
fwd     ack     #6
get     fin
fwd     fin
get     finack
fwd     finack
```

Agent

```
recv    data    #1
send    ack     #1
recv    data    #2
send    ack     #2
recv    data    #3
send    ack     #3
recv    data    #5
send    ack     #5
drop    data    #6
recv    data    #4
send    ack     #4
ignr    data    #5
send    ack     #5
drop    data    #6
flush
recv    data    #6
send    ack     #6
recv    fin
send    finack
flush
```

Receiver

Requirement (6/7)

- Settings (1/2)
  - Sender
    - Arguments: IP, port, path of source file, … etc.
    - Default threshold: **16**
    - Input file may include binary or text file
  - Receiver
    - Arguments: IP, port, path of destination file, … etc.
    - Default buffer size: **32**
  - Agent
    - Arguments: IP, port, loss rate, … etc.

Requirement (6/7)

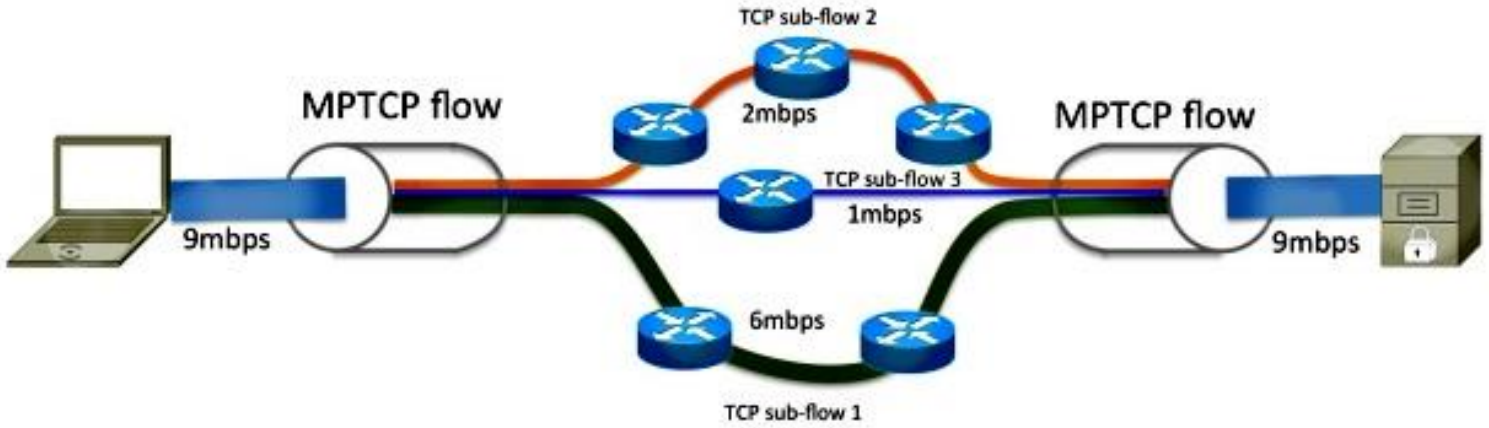
- Settings (2/2)
  - File size
    - more than 0.5 MB (= $0.5 * 10^3$ KB)
  - Data packet size (payload)
    - 1 KB (= 10^3 bytes)
  - Time out
    - Less than or equal to 1 second ($\leq 1\ sec$)

Requirement (7/7)

- Document
  - Format
    - A4, at most 2 pages
    - Digital PDF file only, “report.pdf”
  - Program
    - Execution environment (language, any library or framework)
    - How to run? (compile, execute, …)
  - Design
    - Details of your design (flow chart, …)
    - Difficulties and solutions

Bonus

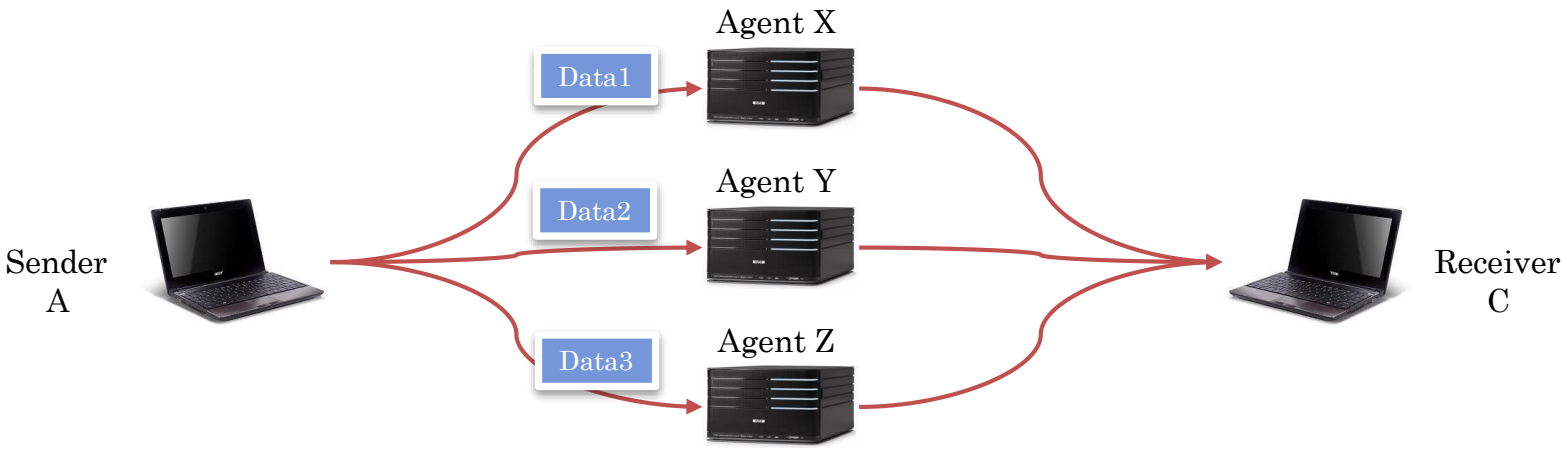
- Multipath TCP
  - Separate single data flow to multiple sub-flows
  - Higher throughput



Cisco: MPTCP and Product Support Overview http://goo.gl/MJm6Uz

Bonus

- Multipath TCP
  - Architecture for this homework
  - Send different packets to different paths



Grading & Submission

Grading & Submission

- Grading (100%+10%)
  - Basic requirement (10%)
    - Socket programming with UDP
    - Language: C
    - Without crash
  - Reliable transmission (20%)
  - Congestion control (25%)
  - Buffer handling (15%)
  - Agent (10%)
  - Message format (5%)
  - Document (5%)
  - Demo (10%)
    - See next slide for more details
  - Bonus (+10%)
    - Multipath TCP

Grading & Submission

- Demo (10%)
  - Please fill demo form (will be announced on course website)
    - Before deadline of homework 2
    - Come to demo on time
      - Discount for those are not on time
  - For those who did not fill demo form
    - You can come to demo on the dates listed in demo form if there exists an empty time slot, or we have free time when you come.
    - You may get at most 90%+10% for all other grading items, except for demo (10%).
    - You are not allowed to demo except for the dates listed in demo form. That is, you will get ZERO score for this homework in total.

Grading & Submission

- Submission
  - Deadline
    - **2015/12/22 (Tue.) 23:59 (UTC+8)**
    - Late submission: 20% off per day
  - Naming
    - [Student ID]_[Version].zip
      Ex: r03944059_v2.zip
  - FTP
    - Hostname: voip.csie.org / Port: 21
    - Username: cn2015
    - Password: cn2015

Questions?