IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

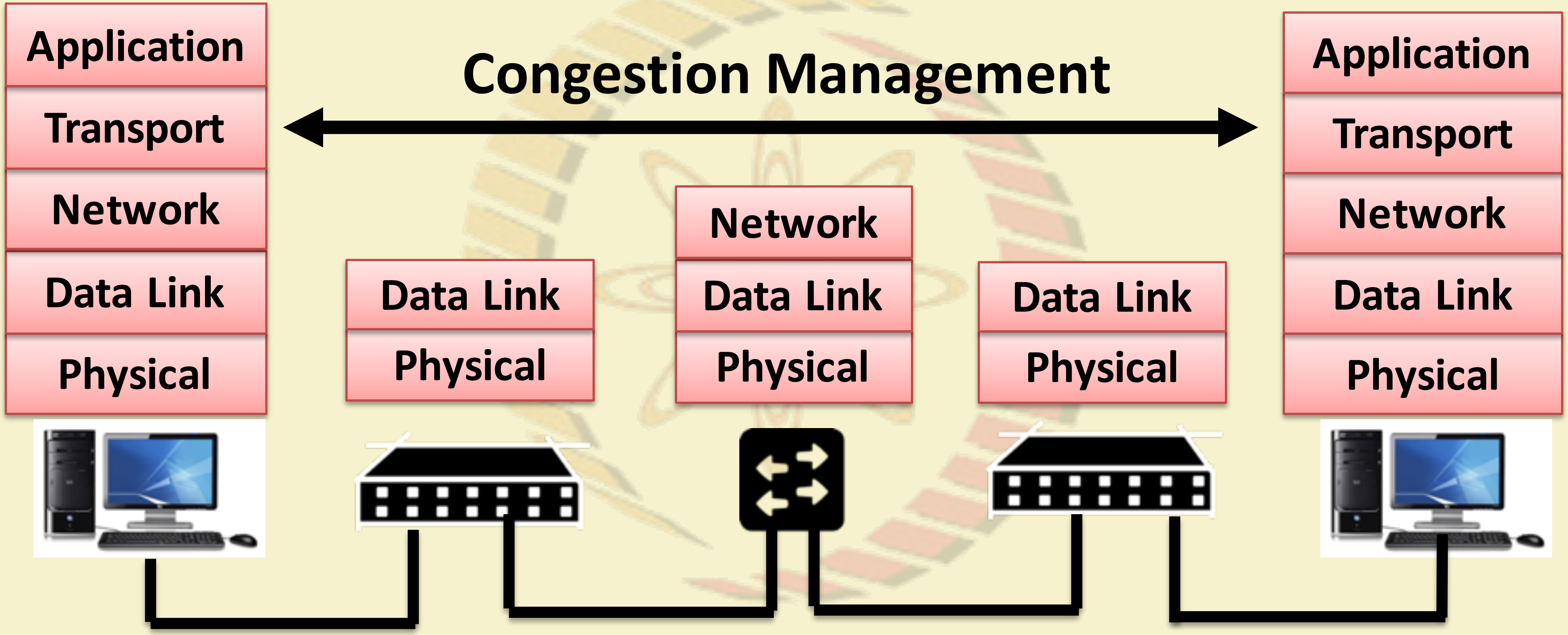
NPTEL

COMPUTER NETWORKS AND INTERNET PROTOCOLS

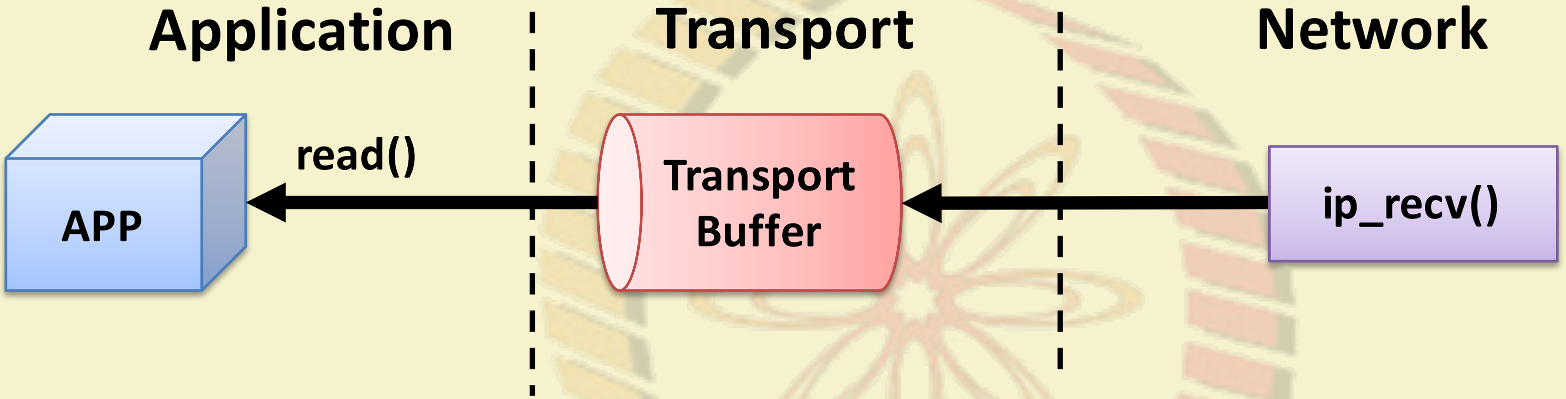
SOUMYA K GHOSH
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

SANDIP CHAKRABORTY
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

Transport Layer - VII (Buffer Management and Congestion Control)



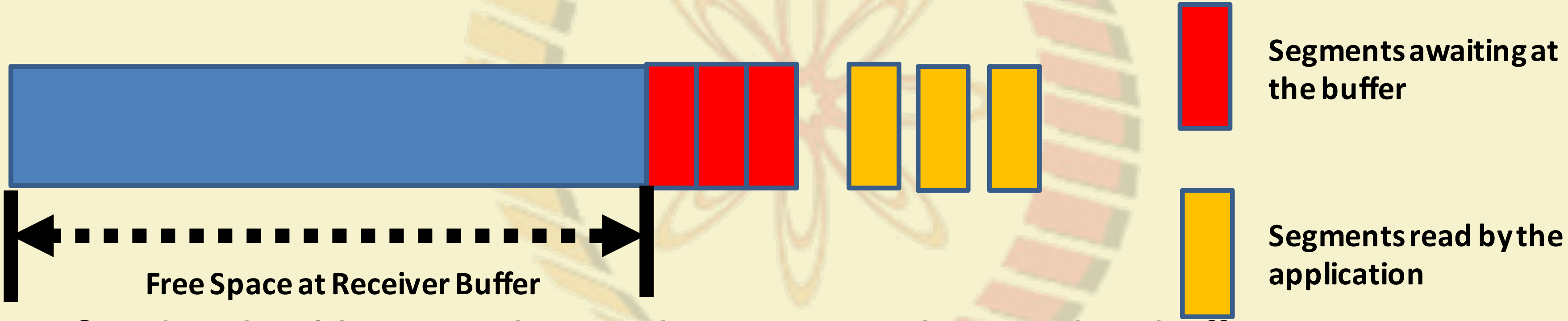
Transport Buffer at the Receiver Side



- There can be rate difference between
  - The rate of application read
  - The rate of transport receive

Dynamic Buffer Management for Window Based Flow Control

- Sender and receiver needs to dynamically adjust buffer allocations
- Based on the rate difference between the **transport entity** and the **application**, the available size of the receiver buffer changes



- Sender should not send more data compared to receiver buffer space – dynamically adjust the window size based on availability of receiver buffer space

Dynamic Buffer Management for Window Based Flow Control

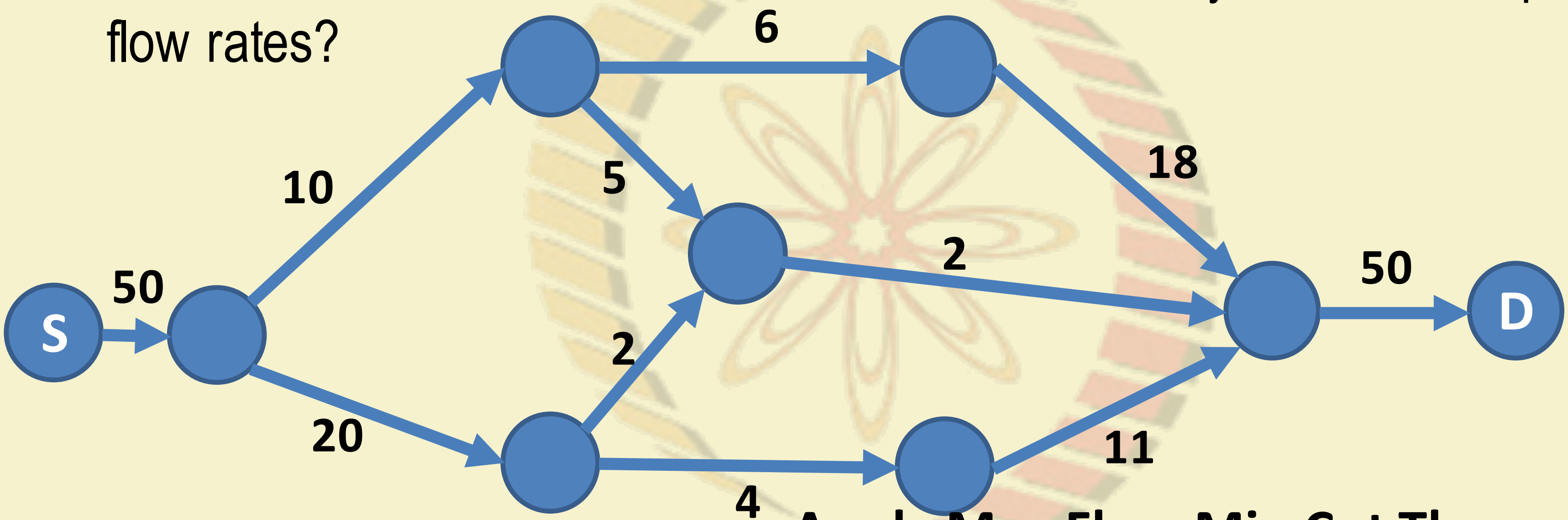
- Receiver forwards available buffer space through ACK

	A	Message	B	Comments
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	• • •	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has 1 buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	• • •	<ack = 6, buf = 4>	←	Potential deadlock

Ensure that the ACKs are flowing in the network continuously

Congestion Control in the Network

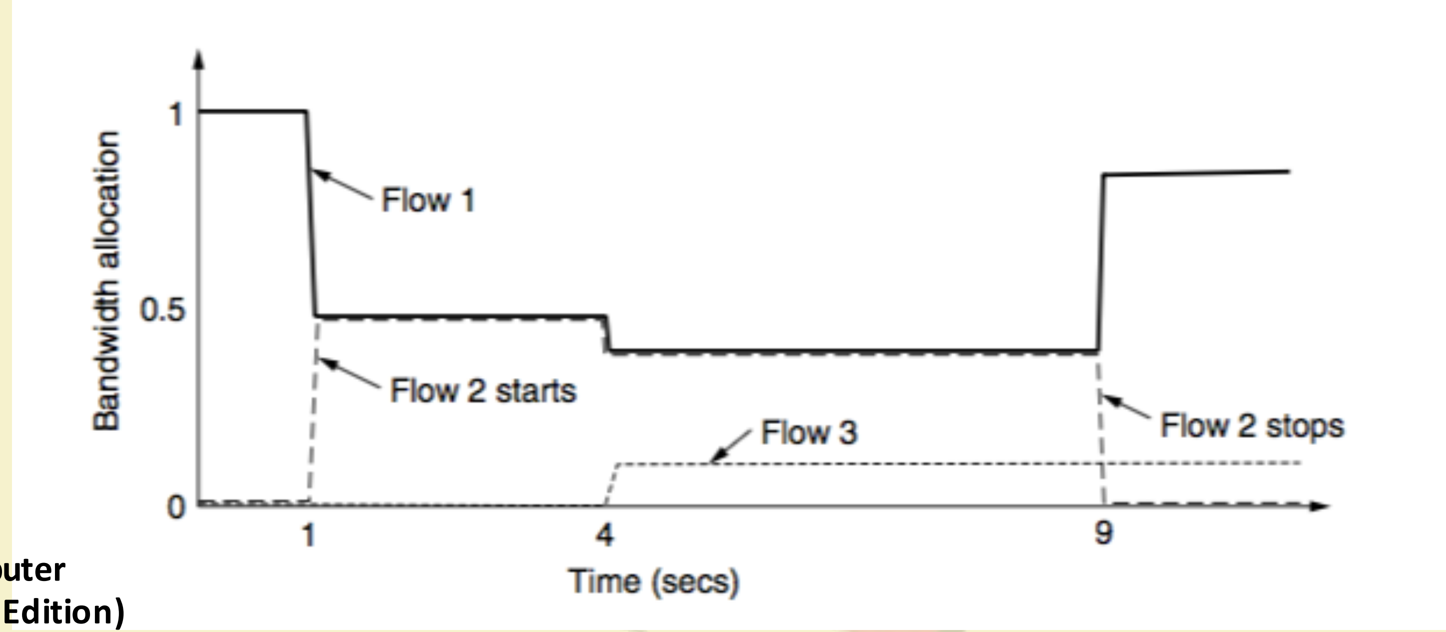
- Consider a centralized network scenario – how can you maintain optimal flow rates?



Apply Max Flow Min Cut Theorem !

But this is hard in a real network …

Congestion Control in the Network



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

Changing Bandwidth Allocation over Time

Congestion Control in the Network

- Flows enter and exit network dynamically – so applying an algorithm for congestion control is difficult

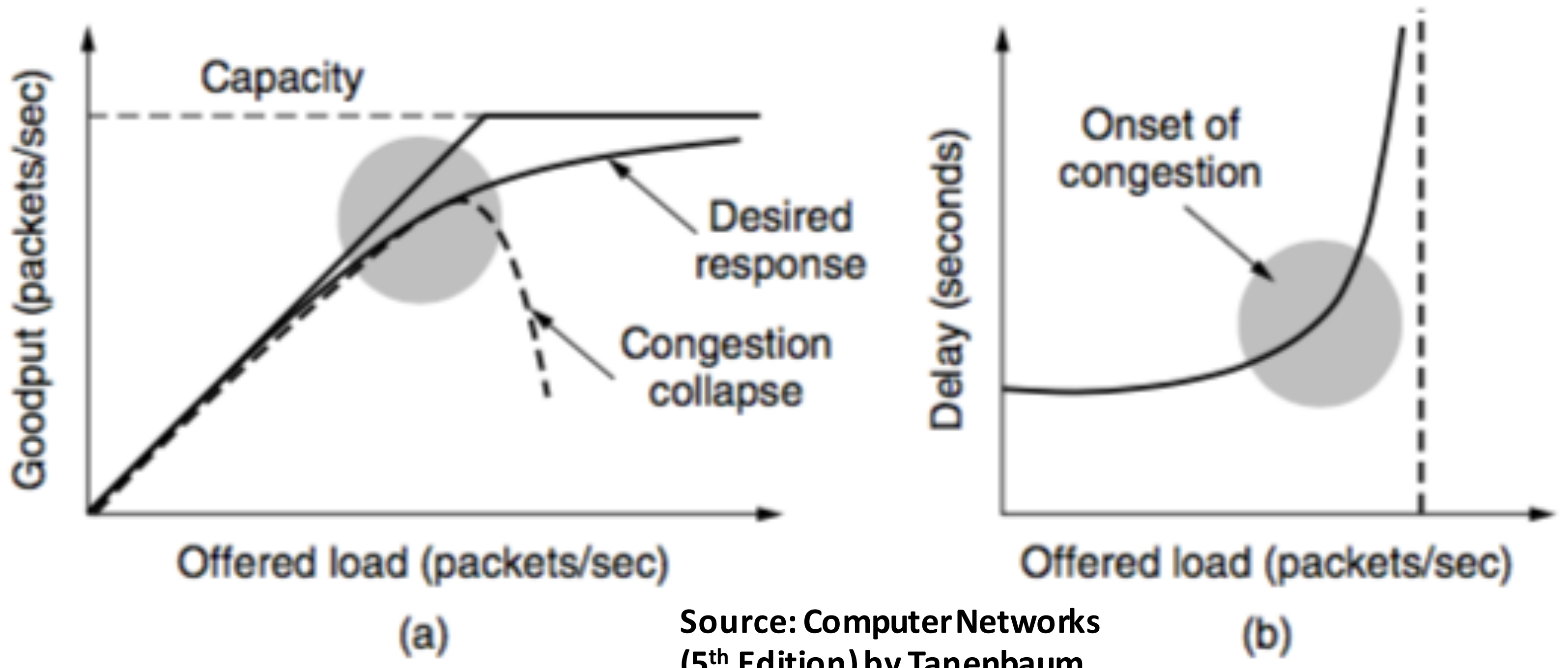
- **Congestion avoidance:** Regulate the sending rate based on what the network can support

Sending Rate = minimum (network rate, Receiver rate)

Gradually increase the network rate and observe the effect on flow rates (packet loss)

Comes from flow control – receiver advertised window size for a sliding window flow control

Network Congestion – Impact over Goodput and Delay

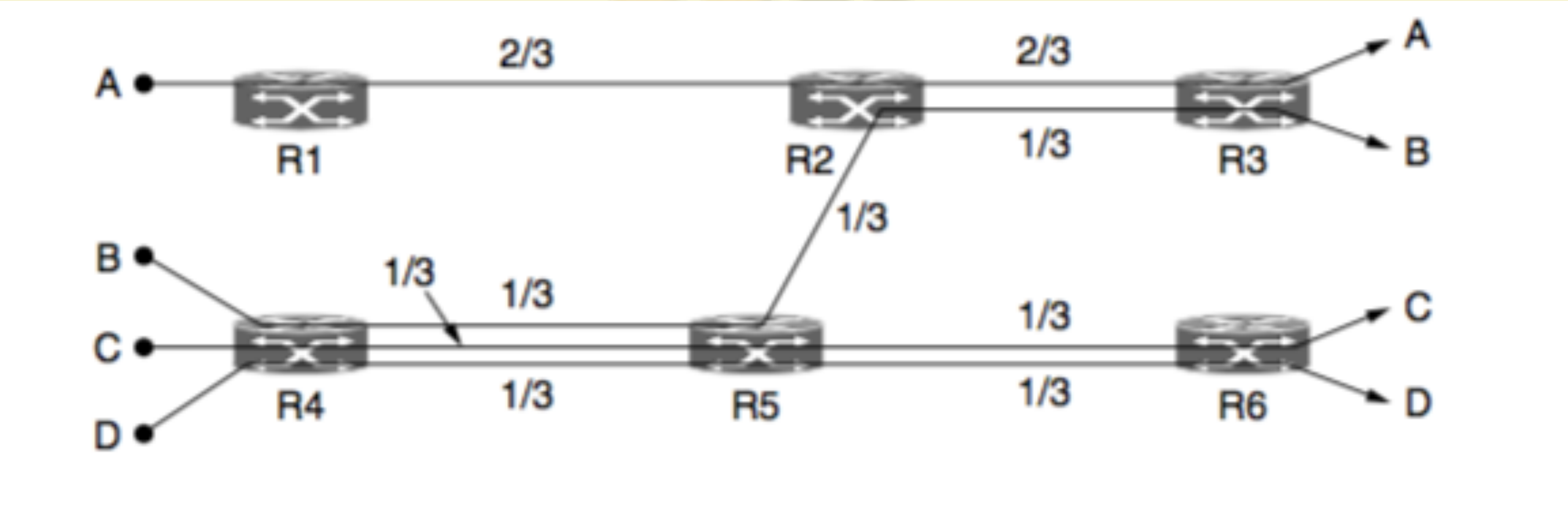


Source: Computer Networks (5^{th} Edition) by Tanenbaum, Wetherell

Congestion Control and Fairness

- Ensure that the rate of all the flows in the network is controlled in a **fair way**
- A bad congestion control algorithm may affect fairness - Some flows can get starved
- Hard fairness in a decentralized network is difficult to implement
- **Max-Min Fairness**: An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation.

Max-Min Fairness – An Example



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

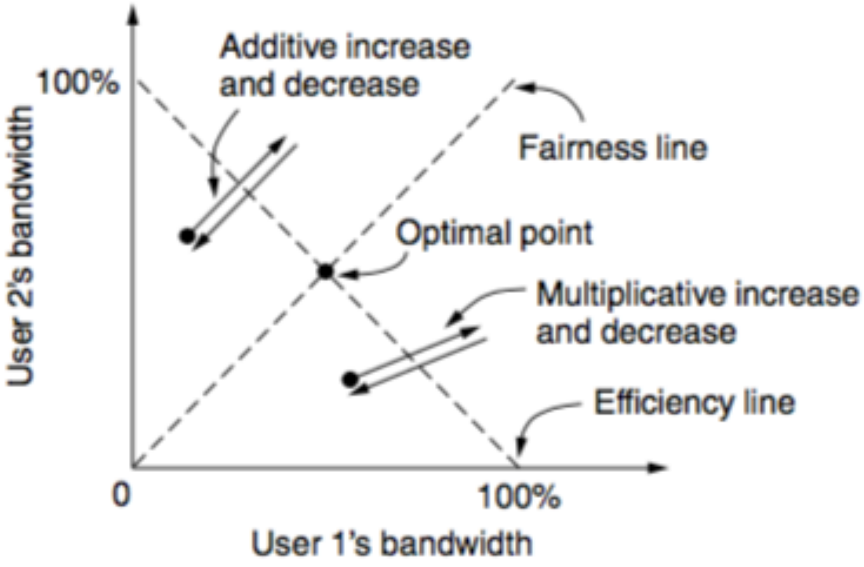
AIMD – Efficient and Fair Operating Point for Congestion Control

- **Additive Increase Multiplicative Decrease (AIMD)** – Chiu and Jain (1989)

- Let *w(t)* be the sending rate. *a (a > 0)* is the additive increase factor, and *b (0<b<1)* is the multiplicative decrease factor

$$w(t+1) = \begin{cases} w(t) + a & \text{if congestion is not detected} \\ w(t) \times b & \text{if congestion is detected} \end{cases}$$

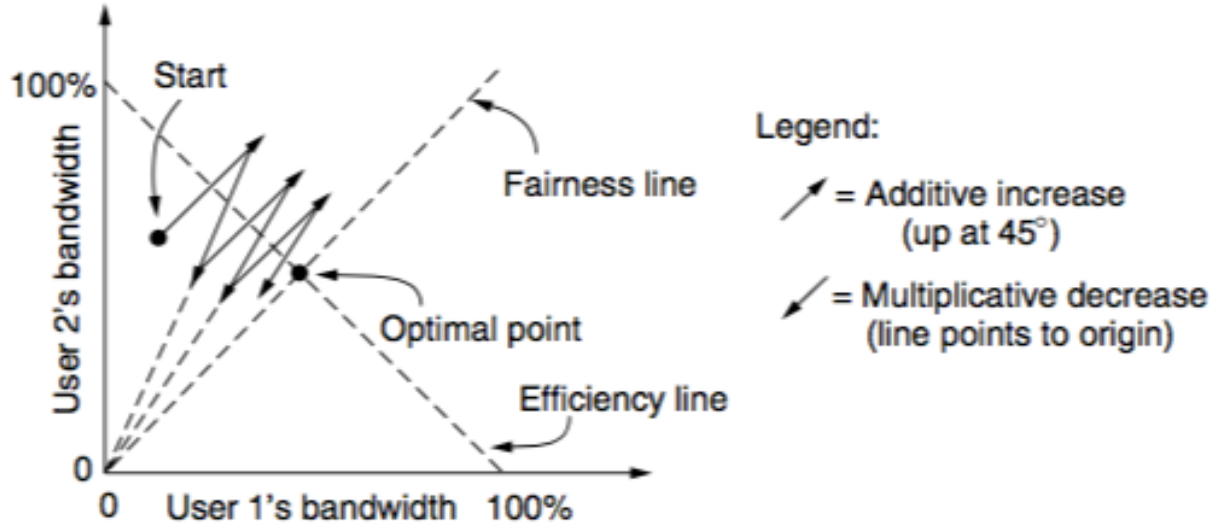
AIMD – Design Rationale (Two Flows Example)



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

- **AIAD** – Oscillate across the efficiency line
- **MIMD** – Oscillate across the efficiency line (different slope from AIAD)

AIMD – Design Rationale (Two Flows Example)



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

- The path converges towards the optimal point
- Used by TCP - Adjust the size of the sliding window to control the rates

thank you!