

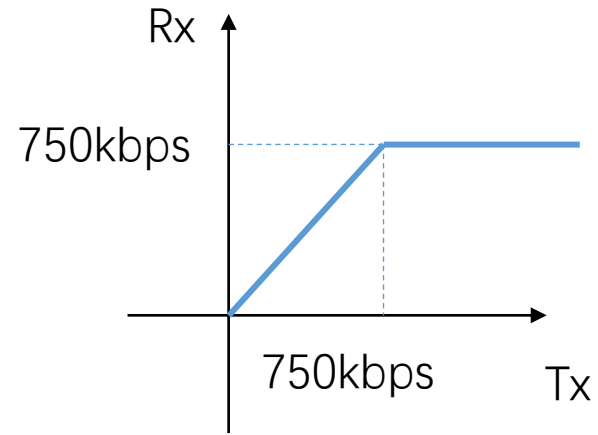


CS120: Computer Networks

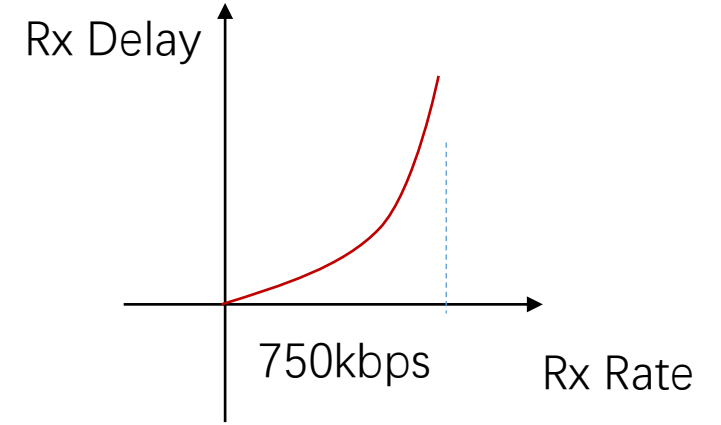
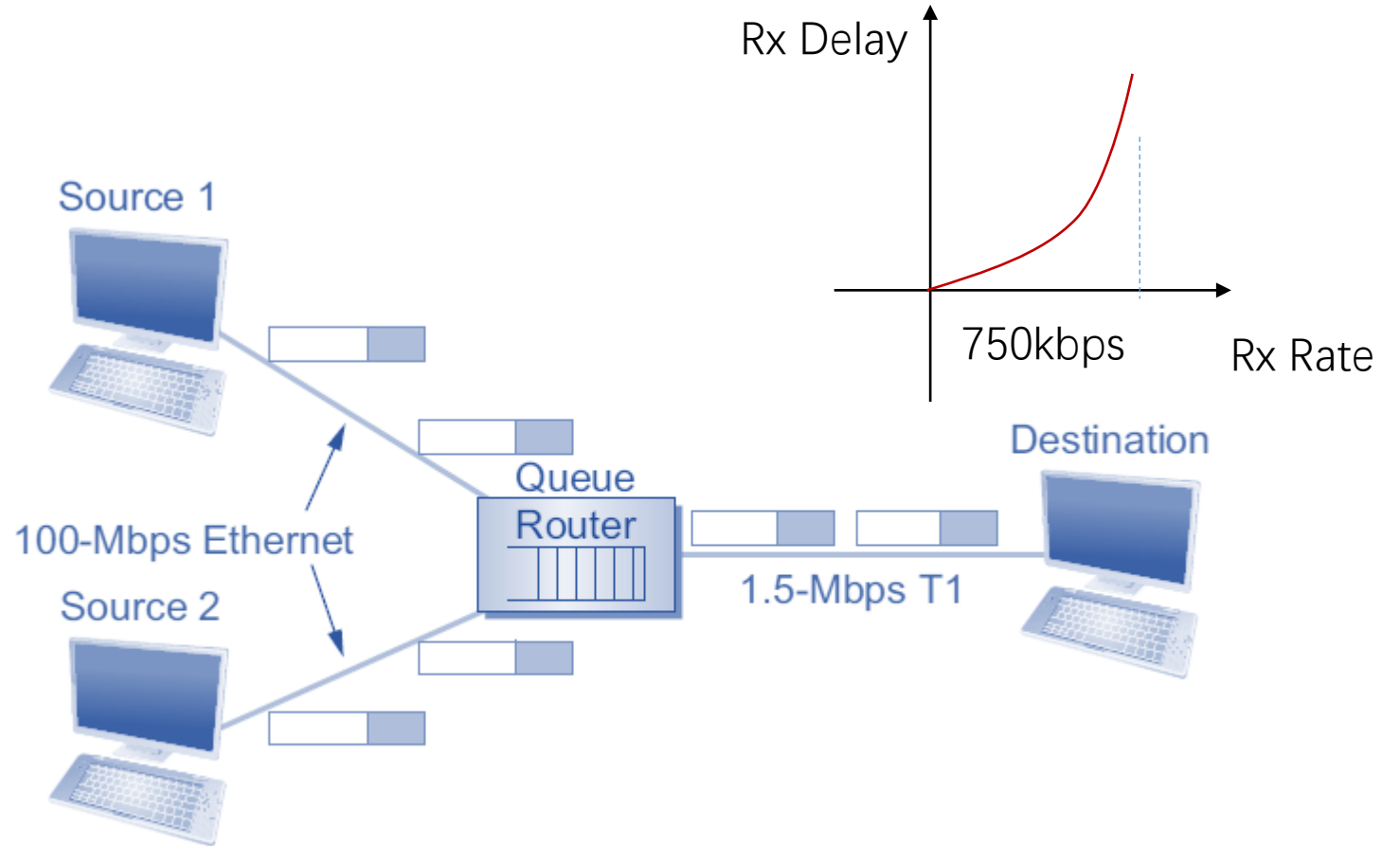
Lecture 17. Congestion Control 1

Zhice Yang

Congestion in Network

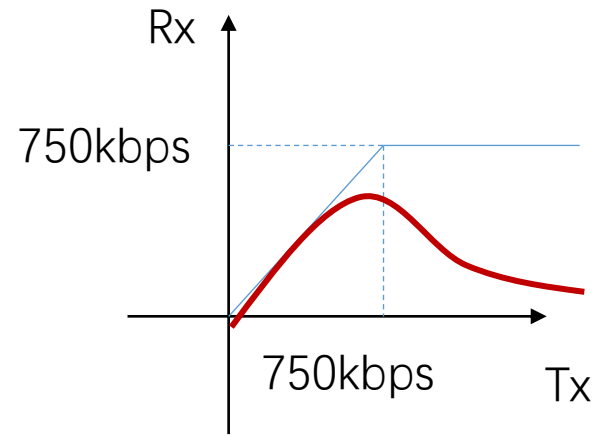


Ideal Case: Infinite Router buffer



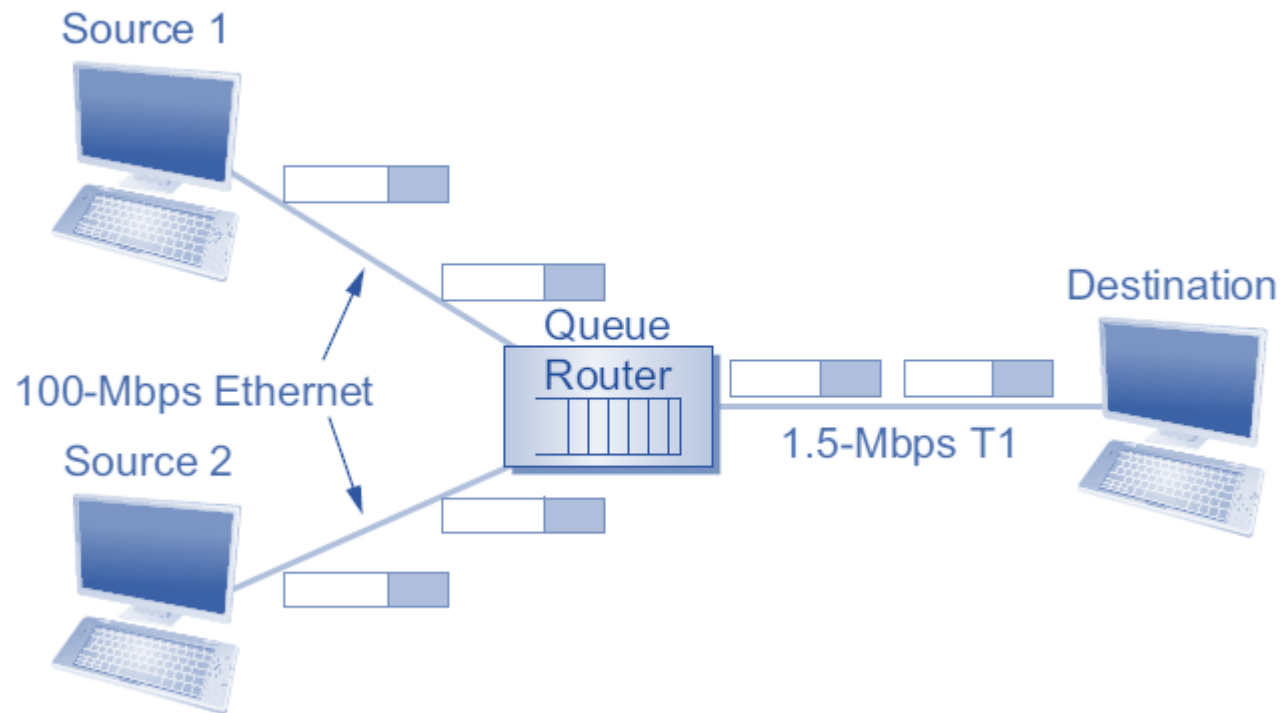
Impact: Network Delay

Congestion in Network



Actual Case: Finite Router buffer

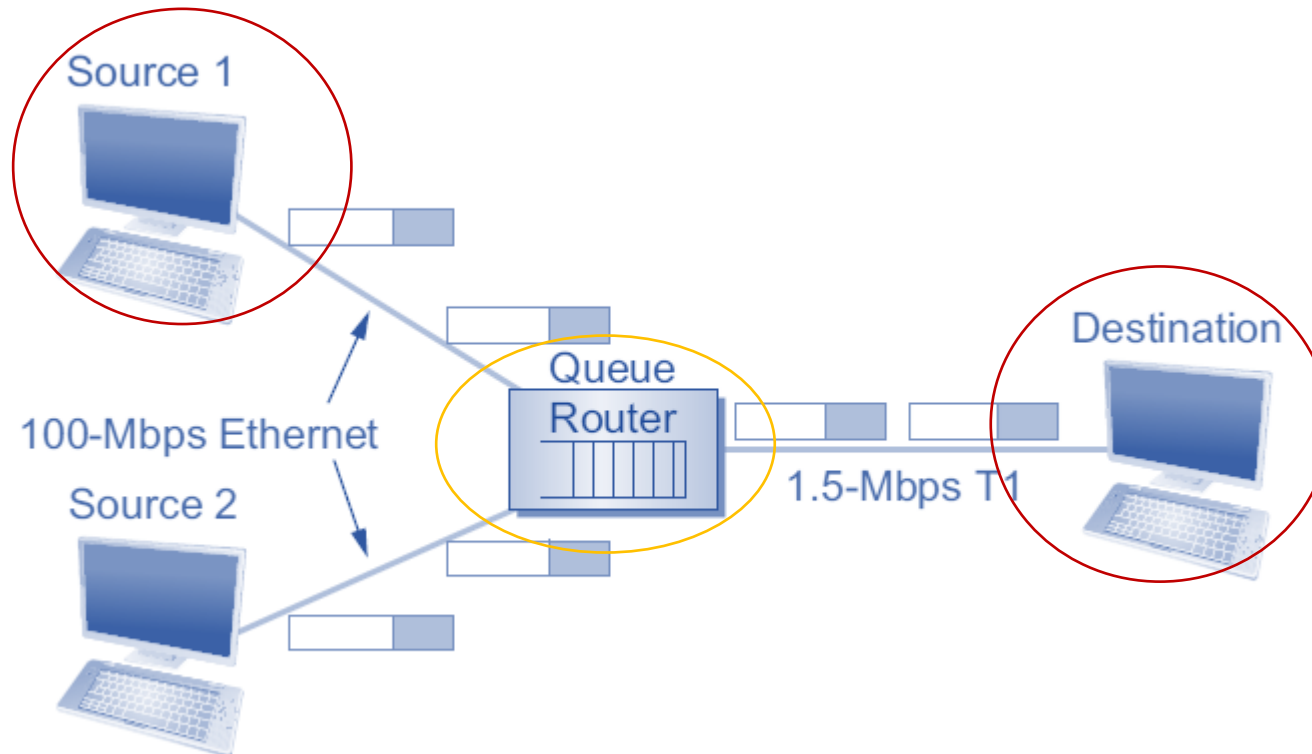
- Packets can be lost (dropped at router) due to full buffers
- Sender does not know when packet has been dropped, retransmissions might be unnecessary



Impact: Retransmissions waste network capacity

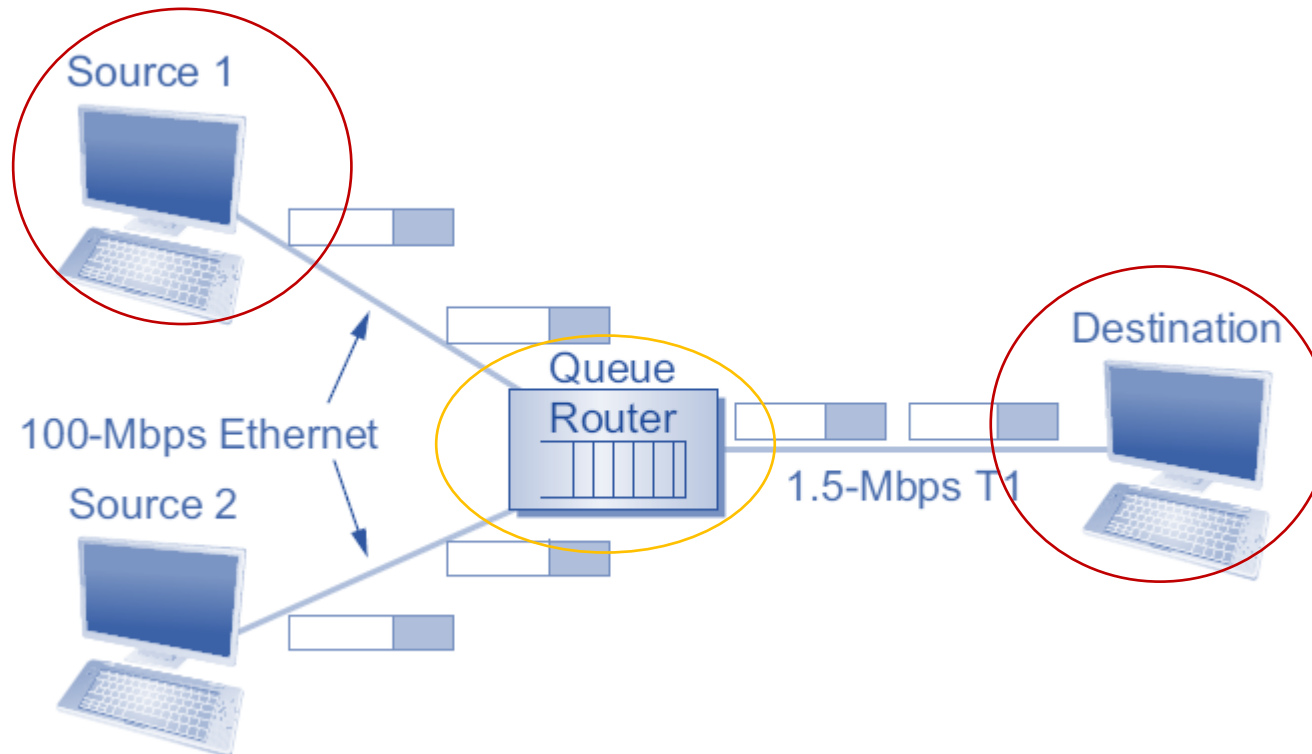
Two Places to Handle Network Congestion

- End hosts
- Routers



Two Places to Handle Network Congestion

- End hosts
- Routers



Congestion Control

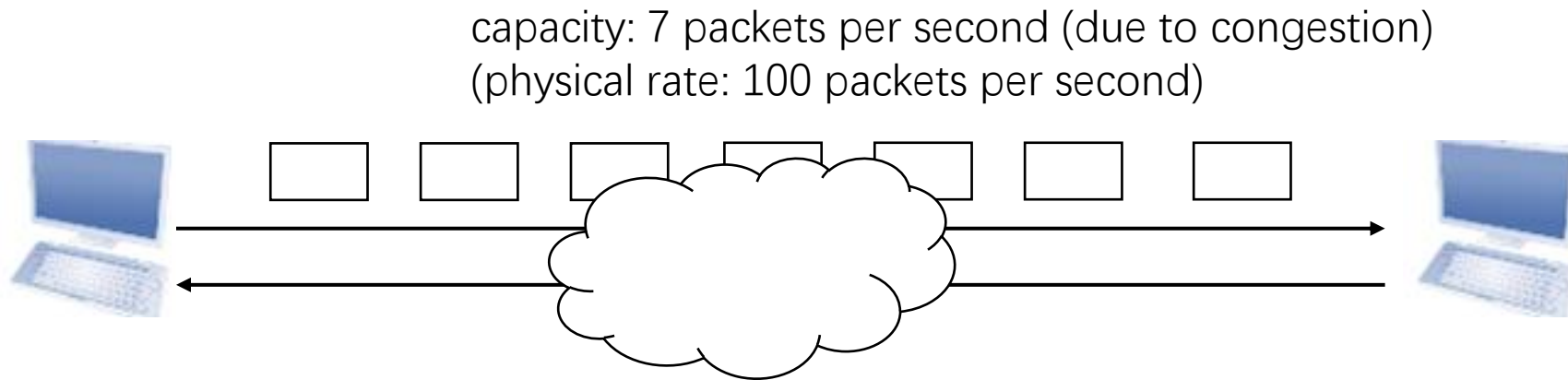
- Host-based Congestion Control
 - Packet Loss
 - AIMD
 - Slow Start
 - Fast Retransmission
 - Fast Recovery
 - Delay
- Router-based Congestion Control
 - Queuing Discipline

TCP Congestion Control

- Introduced by Van Jacobson through his Ph.D. dissertation work in late 1980s
 - 8 years after TCP became operational
- Basic ideas
 - Each host determines network capacity for itself
 - Leverage feedback
- Challenges
 - Determining the available capacity
 - Adjusting to changes in capacity

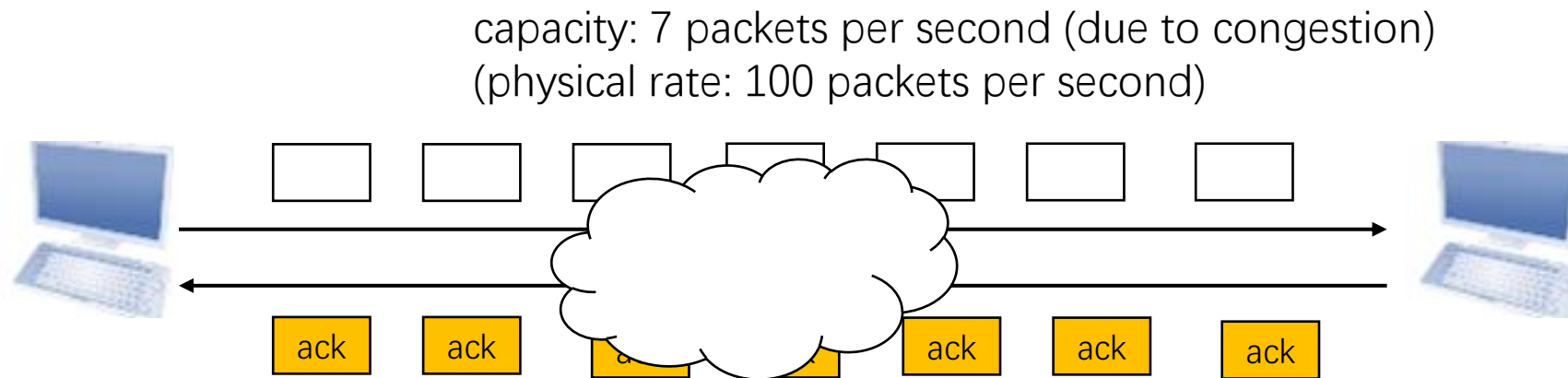
Simple Case – Steady Capacity

- In the steady state
 - How to measure the network capacity ?
 - How to pace the sender ?



Simple Case – Steady Capacity

- In the steady state
 - How to measure the network capacity ?
 - How to pace the sender ?



TCP uses ACKs to estimate the capacity and pace the sending,
i.e., self-clocking

TCP Congestion Control

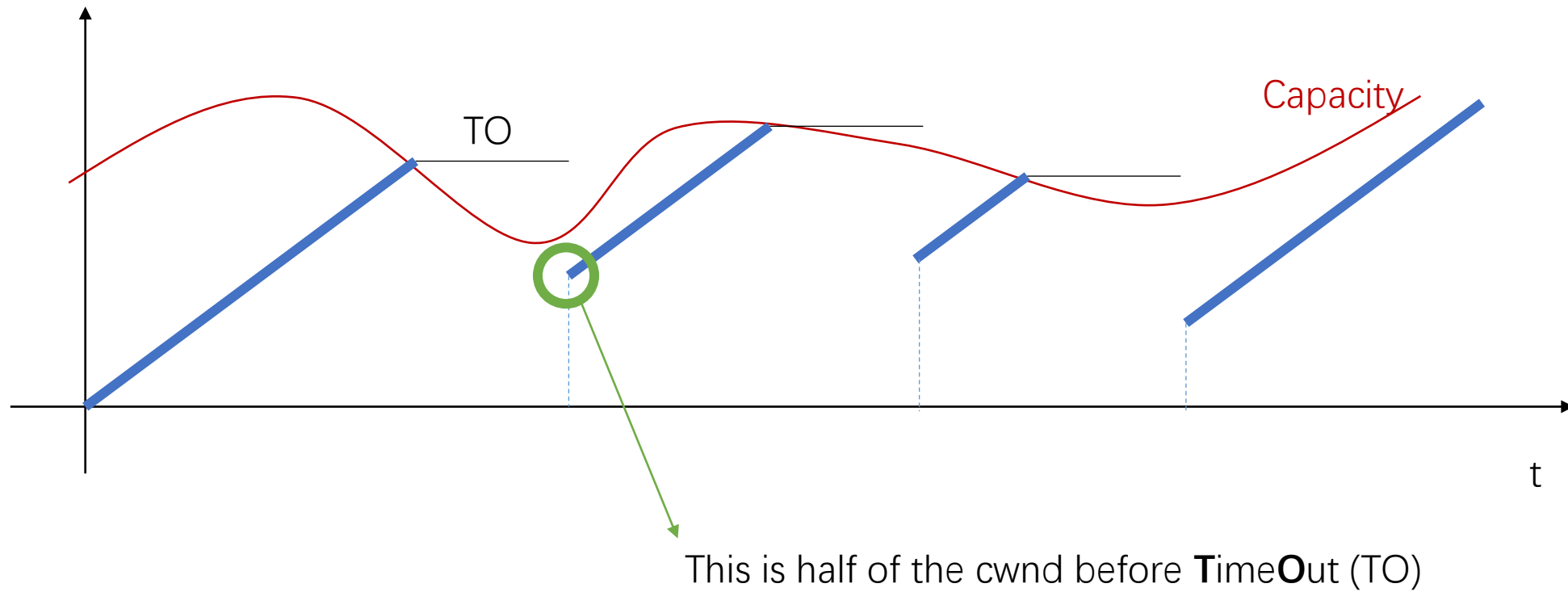
- Objective: Estimate and Adapt to (varying) Network Capacity
- Approach: Adjust Sliding Window according to ACKs
 - $\text{MaxWindow} = \text{MIN}(\text{CongestionWindow}, \text{AdvertisedWindow})$
 - Decrease **CongestionWindow** upon detecting congestion
 - Increase **CongestionWindow** upon lack of congestion
 - **CongestionWindow** abbr. cwnd (in unit of MSS)
- Basic Components
 - Additive Increase/Multiplicative Decrease (AIMD)
 - Slow Start
 - Fast Retransmission
 - Fast Recovery
- Other Variants

Additive Increase/Multiplicative Decrease (AIMD)

- Intuition: over-sized window is much worse than an under-sized window
 - Over-sized window: packets dropped and retransmitted
 - Under-sized window: somewhat lower throughput
- Additive Increase
 - If successfully received acks of the **last window** of data
 - $\text{cwnd} = \text{cwnd} + 1$
- Multiplicative Decrease
 - If packet loss
 - $\text{cwnd} = \text{cwnd} / 2$

AIMD

- TCP sawtooth pattern



Sliding Window in TCP: Adaptive Timeout

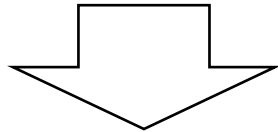
- Jacobson/Karels Algorithm Implementation

Difference = SampleRTT - EstimatedRTT

EstimatedRTT = EstimatedRTT + (δ *Difference)

Deviation = Deviation + δ *(|Difference| - Deviation)

TimeOut = μ * EstimatedRTT + φ * Deviation



```
SampleRTT -= (EstimatedRTT >> 3);
```

```
EstimatedRTT += SampleRTT;
```

```
if (SampleRTT < 0)
```

```
    SampleRTT = -SampleRTT;
```

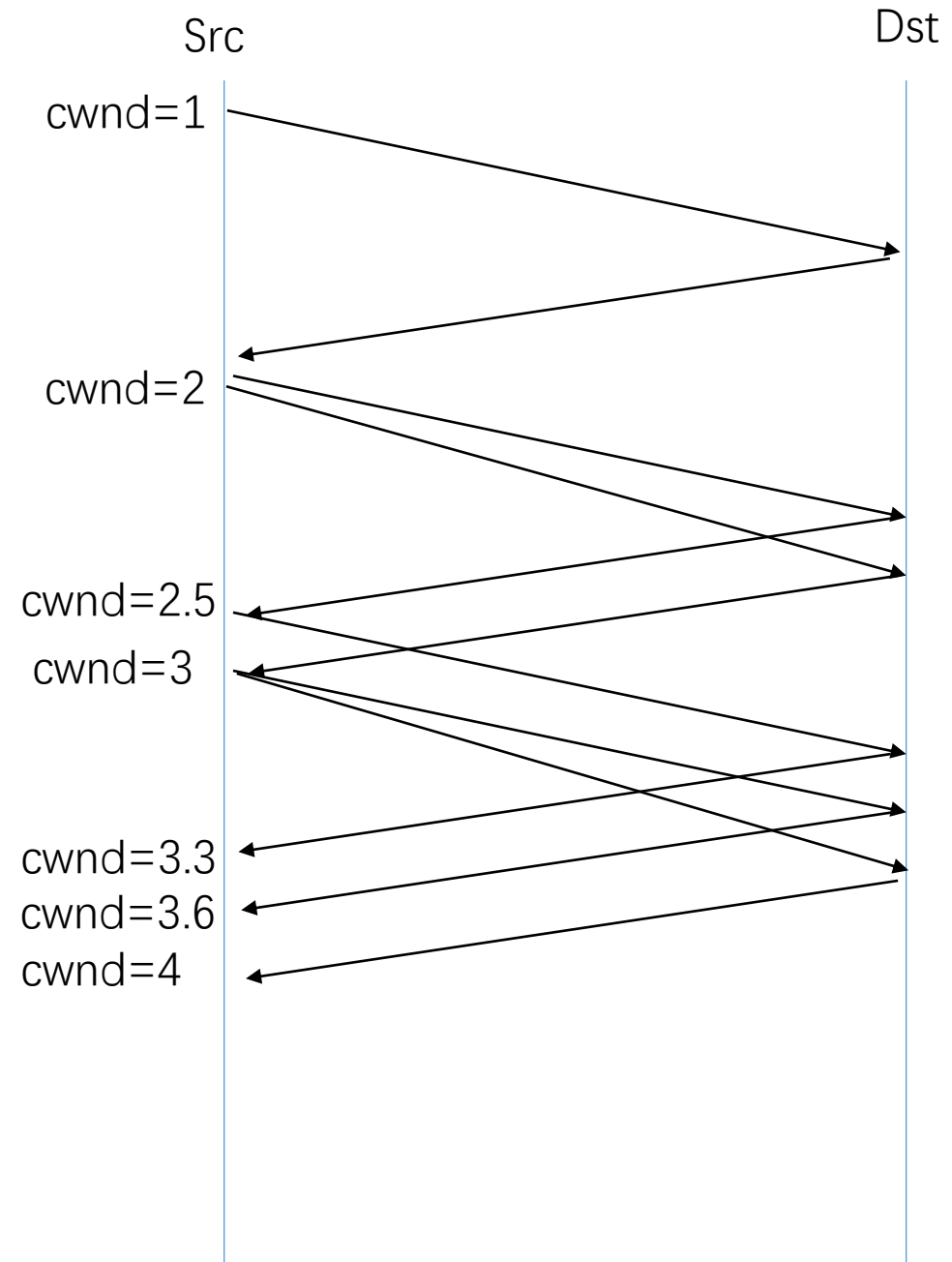
```
SampleRTT -= (Deviation >> 3);
```

```
Deviation += SampleRTT;
```

```
TimeOut = (EstimatedRTT >> 3) + (Deviation >> 1);
```

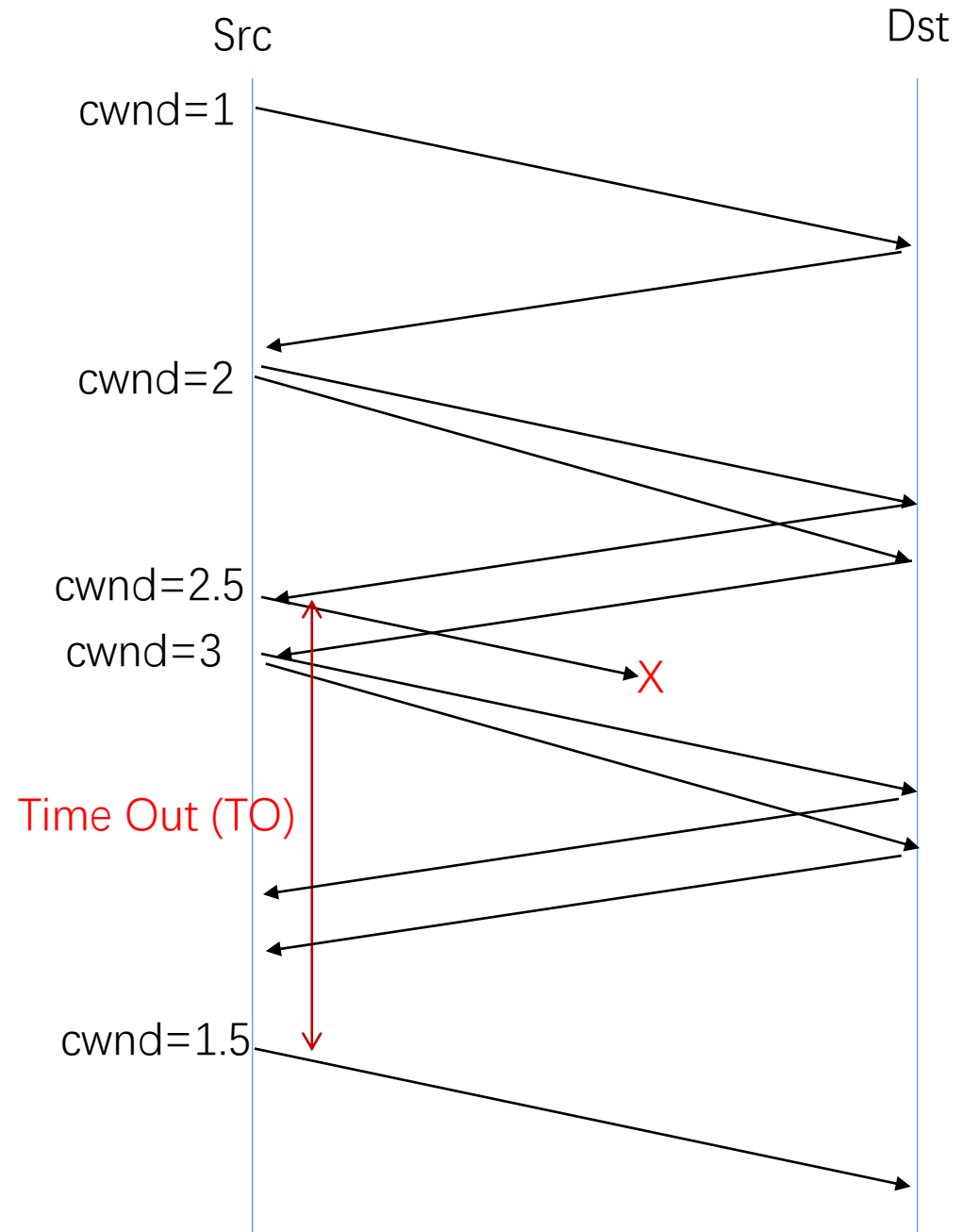
AIMD

- Additive Increase
 - $\text{Increment} = 1/\text{cwnd}$
 - $\text{cwnd} += \text{Increment}$



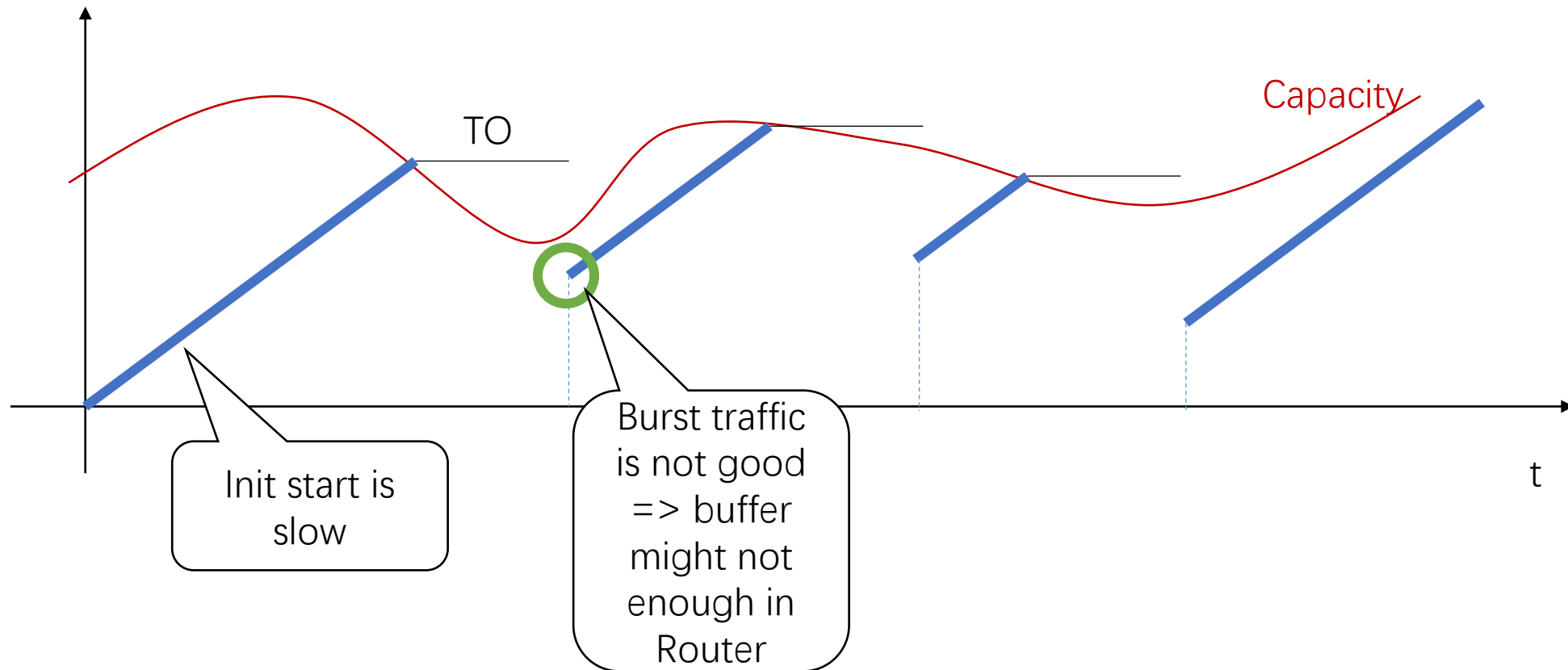
AIMD

- Multiplicative Decrease
 - $\text{cwnd} = \text{cwnd} / 2$



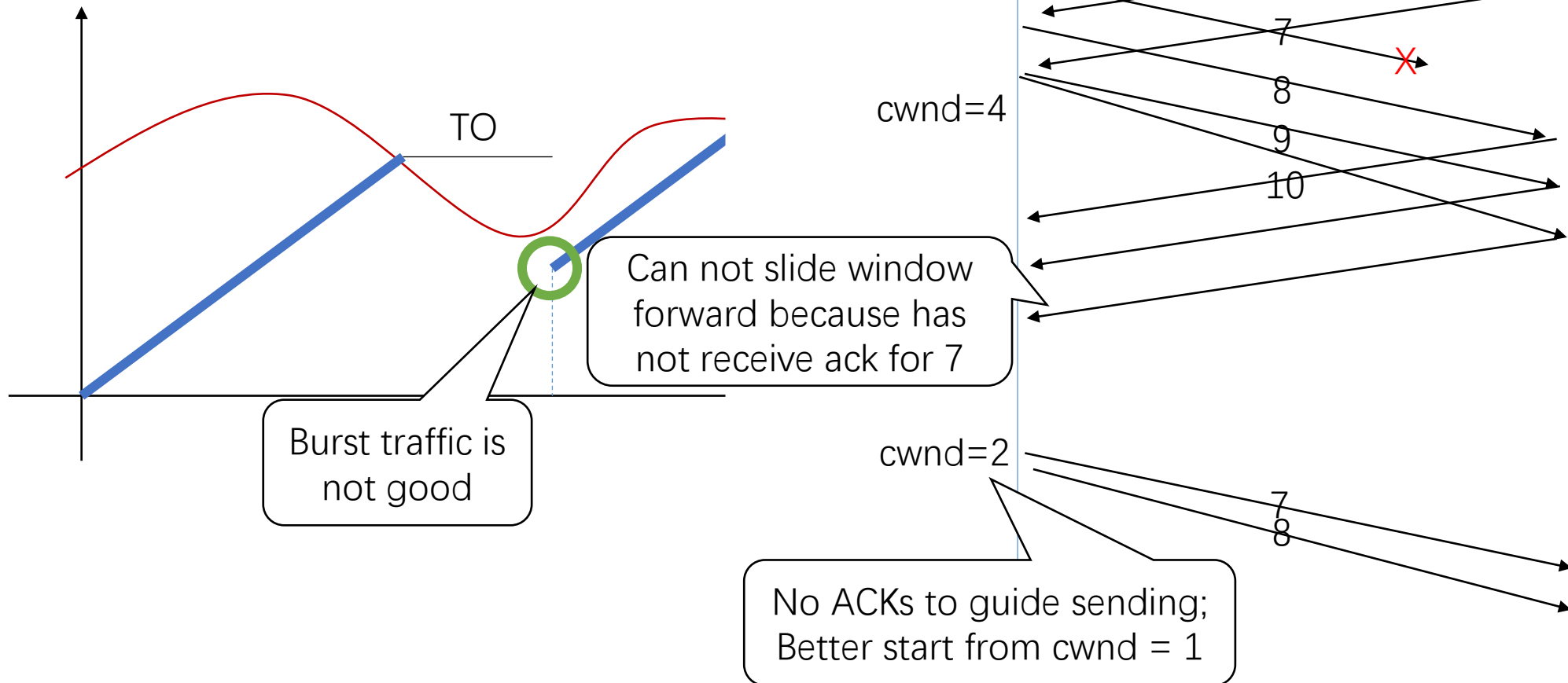
AIMD

- TCP sawtooth pattern



AIMD

- TCP sawtooth pattern

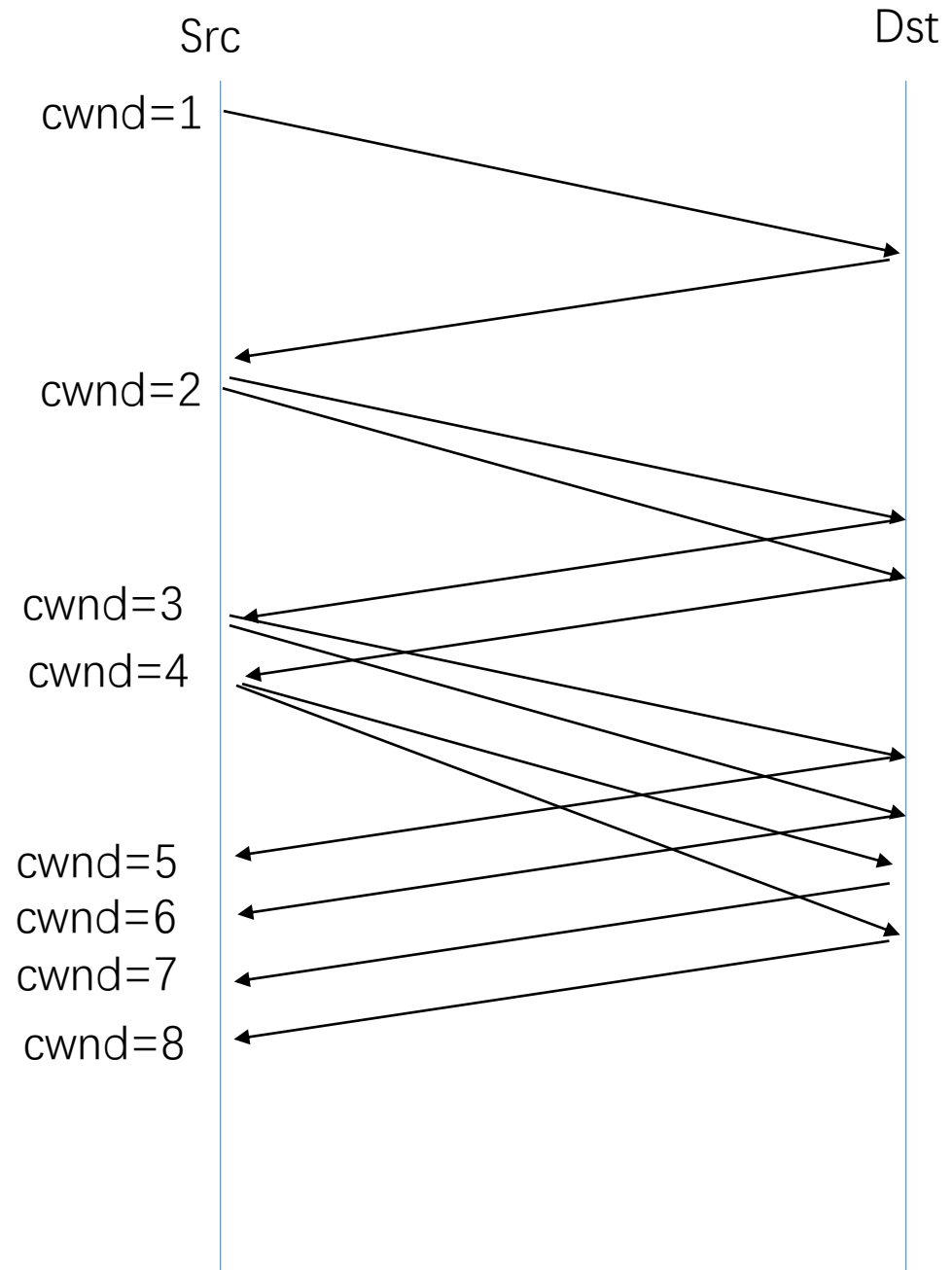


Slow Start

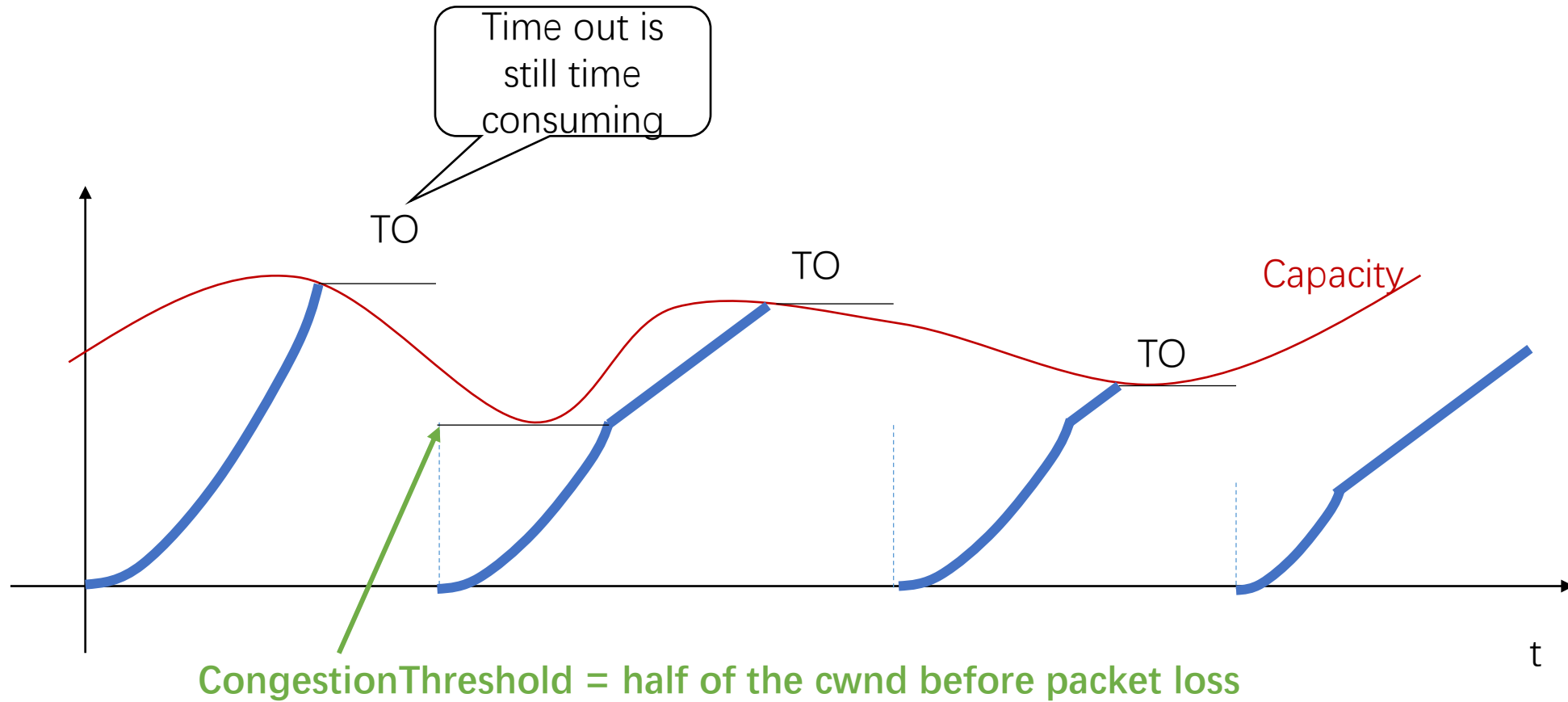
- Intuition: speed up additive Increase when TCP start
- Why “Slow Start”
 - “Slow Start” is not slow compared with additive Increase
 - “Slow Start” is slow compared with sending a whole window’s worth of data
- **Double** CongestionWindow per round-trip time
 - If successfully received **one ack**
 - $\text{cwnd} = \text{cwnd} + 1$
 - Until $\text{cwnd} == \text{CongestionThreshold}$
 - $\text{CongestionThreshold} = \text{half of the cwnd before packet loss}$
 - Then do Additive Increase

Slow Start

- If successfully received one ack
 - $\text{cwnd} = \text{cwnd} + 1$



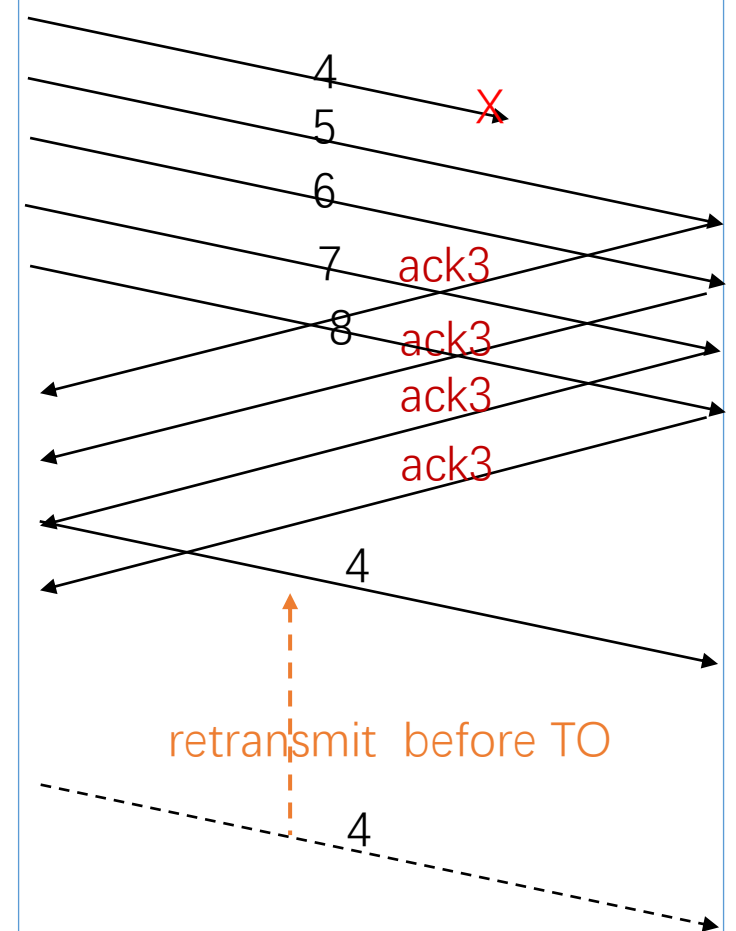
Slow Start



Fast Retransmission

- Intuition: use **duplicate ACK** to indicate packet loss
- Approach:
 - Receiver replies every TCP segment with acknum = next byte expected
 - Transmitter resends a segment after 3 duplicate acks
 - 3 duplicate acks => possible packet loss
- Throughput Gain: 20%

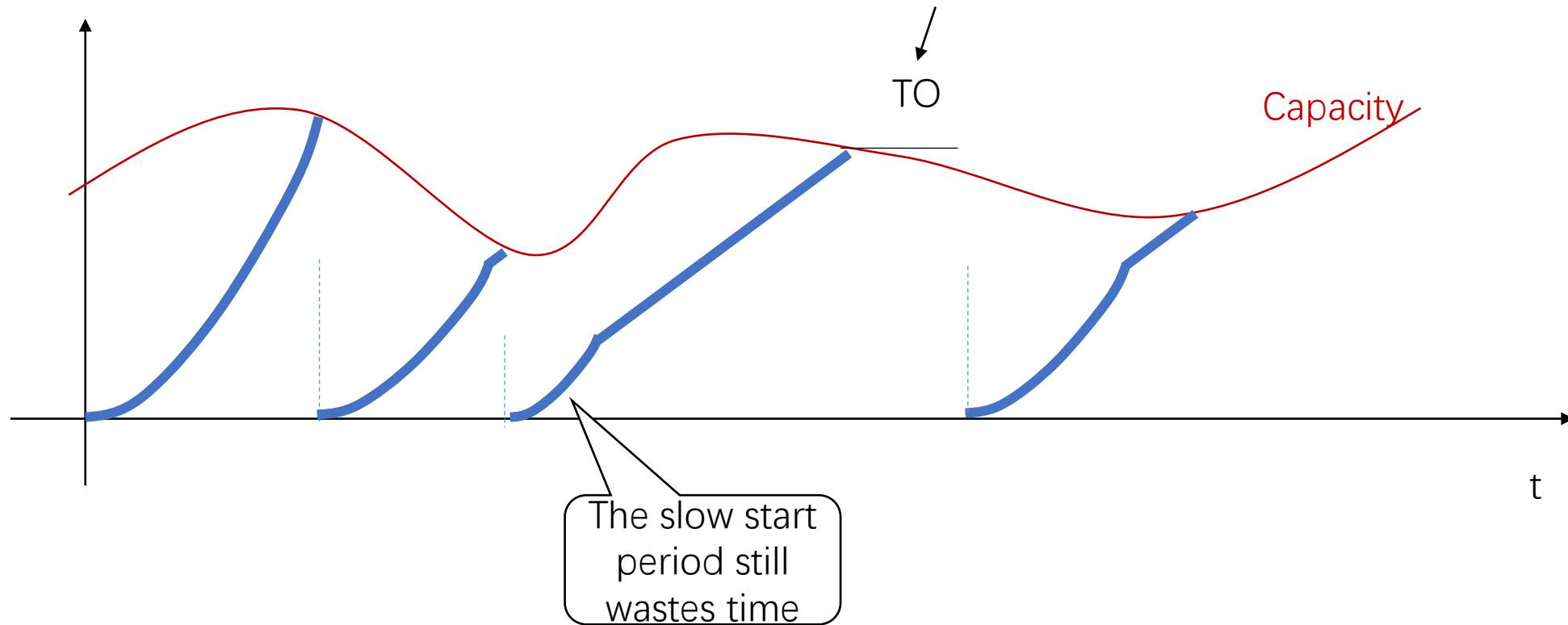
Time
out



Fast Retransmission

Timeout still exists

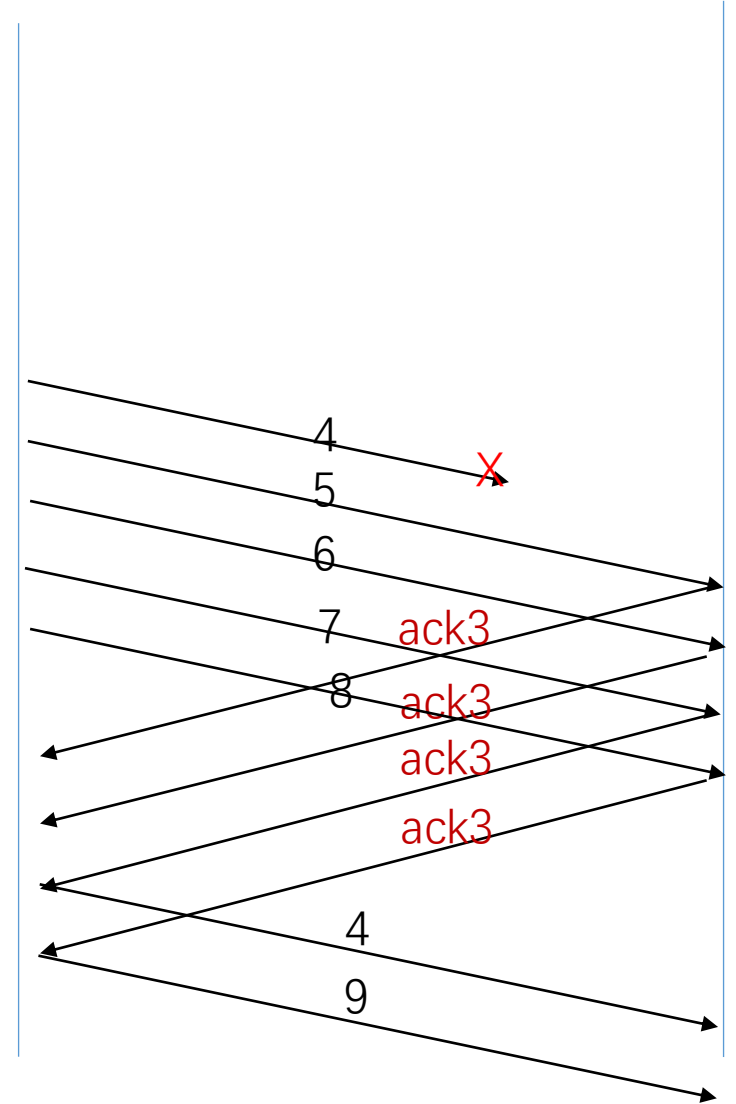
- Too many packet loss
- Window may be too small to generate enough duplicate acks



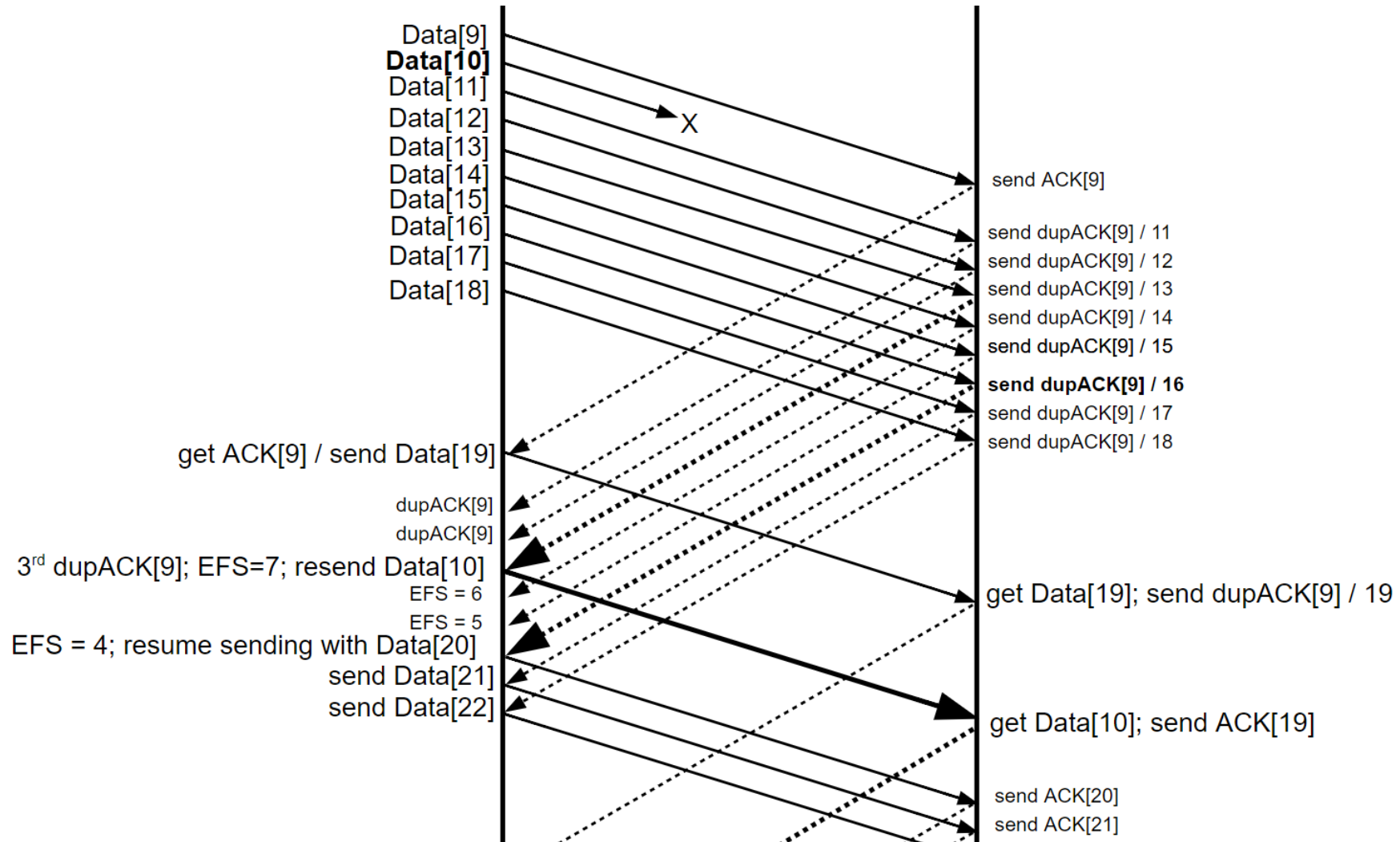
TCP Tahoe

Fast Recovery

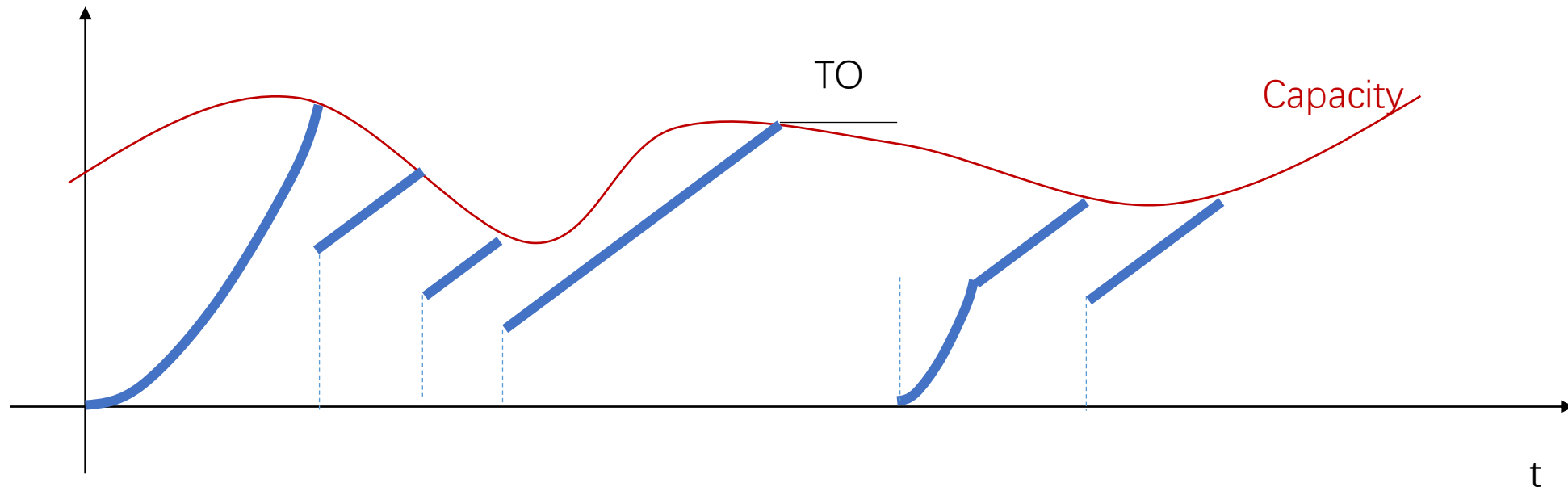
- Intuition:
 - There might be flying acks
 - No need to start from window size 1



Fast Recovery

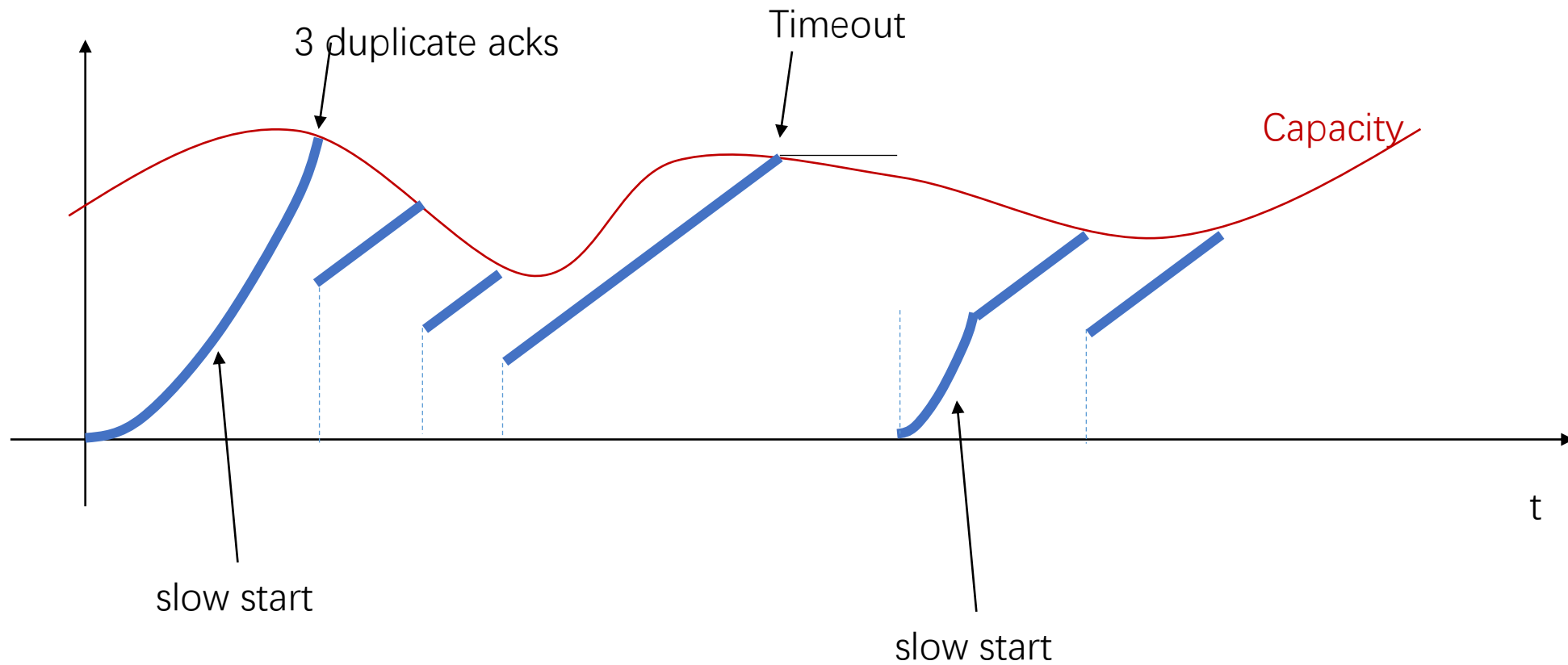


Fast Recovery

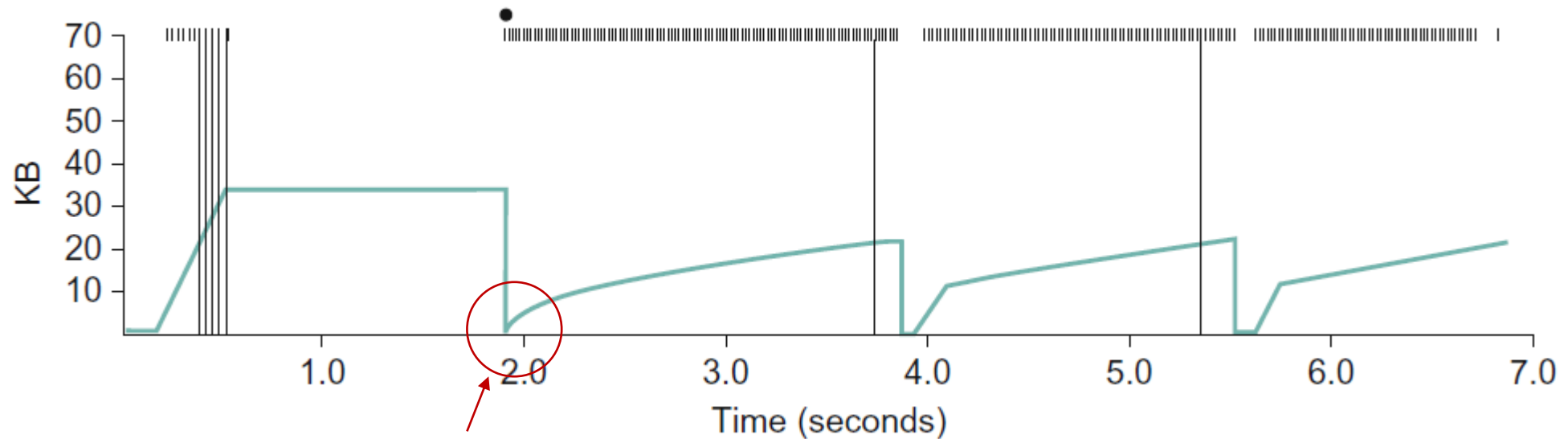


TCP Reno

TCP Reno



Figures in Textbook



Hard to explain, should be slow start

No Fast Recovery in the Simulation, i.e., TCP Tahoe

TCP Congestion Control

- Objective: Estimate and adapt to (varying) network capacity
- Approach: Adjust Sliding Window
 - $\text{MaxWindow} = \text{MIN}(\text{CongestionWindow}, \text{AdvertisedWindow})$
 - Decrease **CongestionWindow** upon detecting congestion
 - Increase **CongestionWindow** upon lack of congestion
- Basic Components
 - Additive Increase/Multiplicative Decrease (AIMD)
 - Slow Start
 - Fast Retransmission
 - Fast Recovery
- Other Variants

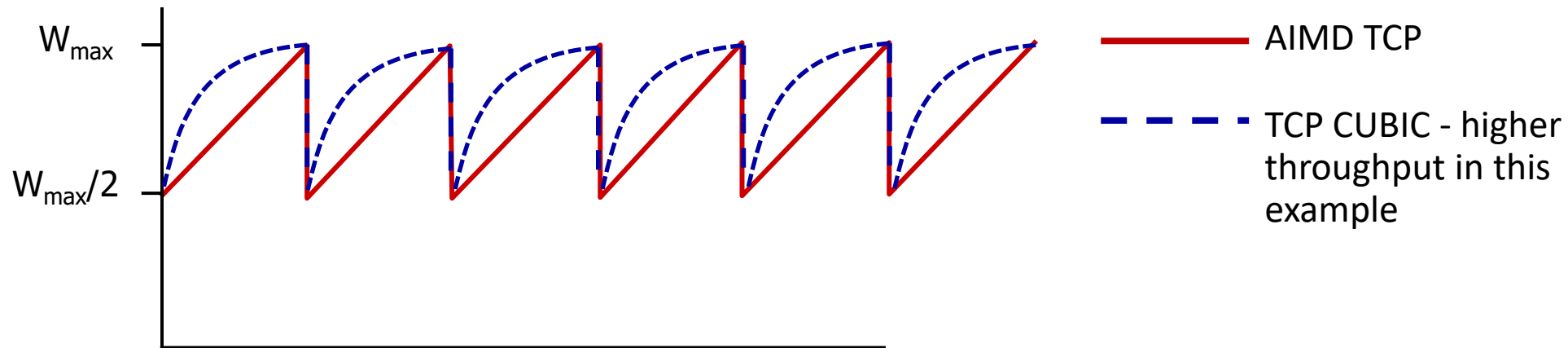
TCP Congestion Control Algorithms

- ref: https://en.wikipedia.org/wiki/TCP_congestion_control

Variant	Feedback	Required changes	Benefits	Fairness
(New)Reno	Loss	-	-	Delay
Vegas	Delay	Sender	Less loss	Proportional
High Speed	Loss	Sender	High bandwidth	
BIC	Loss	Sender	High bandwidth	
CUBIC	Loss	Sender	High bandwidth	
H-TCP	Loss	Sender	High bandwidth	
FAST	Delay	Sender	High bandwidth	Proportional
Compound TCP	Loss/Delay	Sender	High bandwidth	Proportional
Westwood	Loss/Delay	Sender	L	
Jersey	Loss/Delay	Sender	L	
BBR ^[11]	Delay	Sender	BLVC, Bufferbloat	
CLAMP	Multi-bit signal	Receiver, Routers	V	Max-min
TFRC	Loss	Sender, Receiver	No Retransmission	Minimum delay
XCP	Multi-bit signal	Sender, Receiver, Router	BLFC	Max-min
VCP	2-bit signal	Sender, Receiver, Router	BLF	Proportional
MaxNet	Multi-bit signal	Sender, Receiver, Router	BLFSC	Max-min
JetMax	Multi-bit signal	Sender, Receiver, Router	High bandwidth	Max-min
RFD	Loss	Router	Smaller delay	

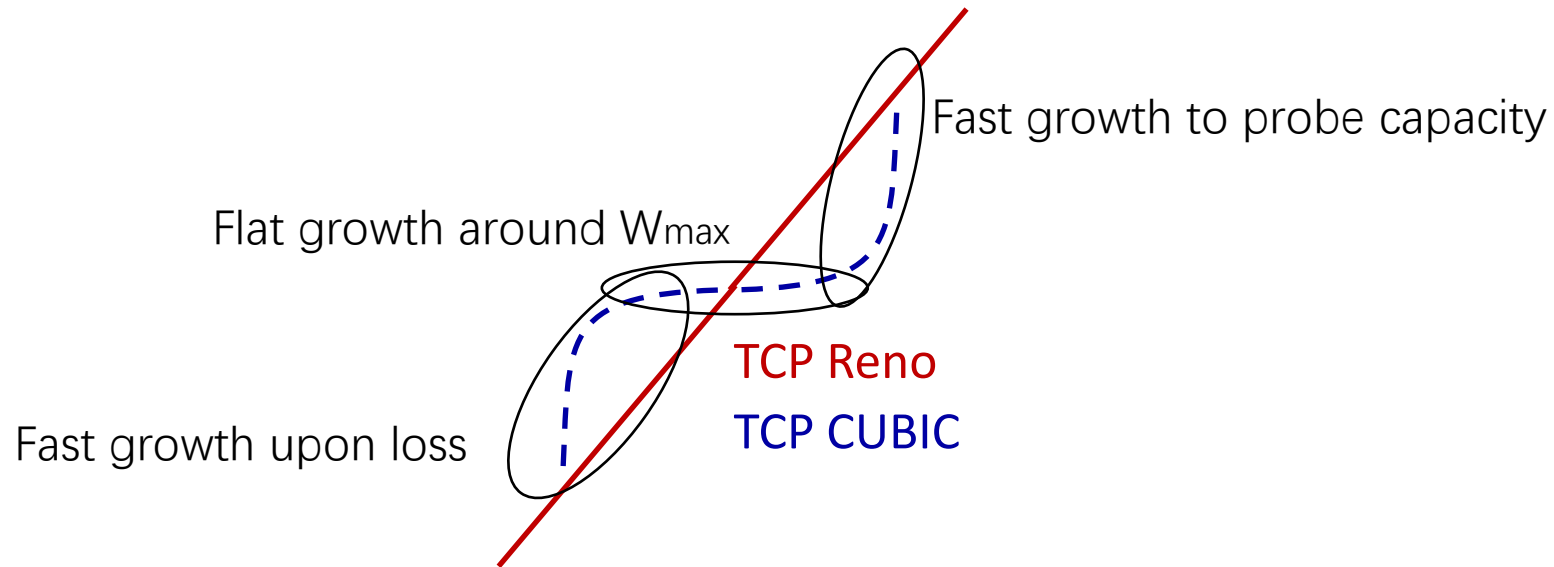
TCP CUBIC

- A better way than AIMD to “probe” for usable bandwidth
 - Intuition: after cutting rate/window in half on loss, initially ramp to W_{\max} faster, but then approach W_{\max} more slowly



TCP CUBIC

- The name “CUBIC” ?
 - Increase cwnd as a function of the cube of the distance between current time and the estimated time reaching the capacity



Demo

- https://wps.pearsoned.com/ecs_kurose_compnetw_6/216/55463/14198700.cw/index.html

Demo

- Sliding Window code location
/net/ipv4/
<https://elixir.bootlin.com/linux/latest/source/net/ipv4>
- Switching Sliding Window Scheme
 - Show current schemes
`cat /proc/sys/net/ipv4/tcp_congestion_control`
 - Switch congestion control scheme
`sysctl net.ipv4.tcp_available_congestion_control[=XX]`

Reference

- Textbook 6.3
- <http://intronetworks.cs.luc.edu/current/html/reno.html>