

Improving TCP Startup Performance using Active Measurements

Ningning Hu, Peter Steenkiste
Carnegie Mellon University

TCP Slow Start

- **Slow start exponentially increases the congestion window size**
 - Determine a proper congestion window size
 - Bootstrap the self-clocking behavior
- **Problems**
 - High instantaneous sending speed
 - Long startup time (ssthresh)
 - Large ssthresh → packet loss
 - Small ssthresh → large startup time
 - More serious on higher speed network
- **Idea: measure the available bandwidth!**



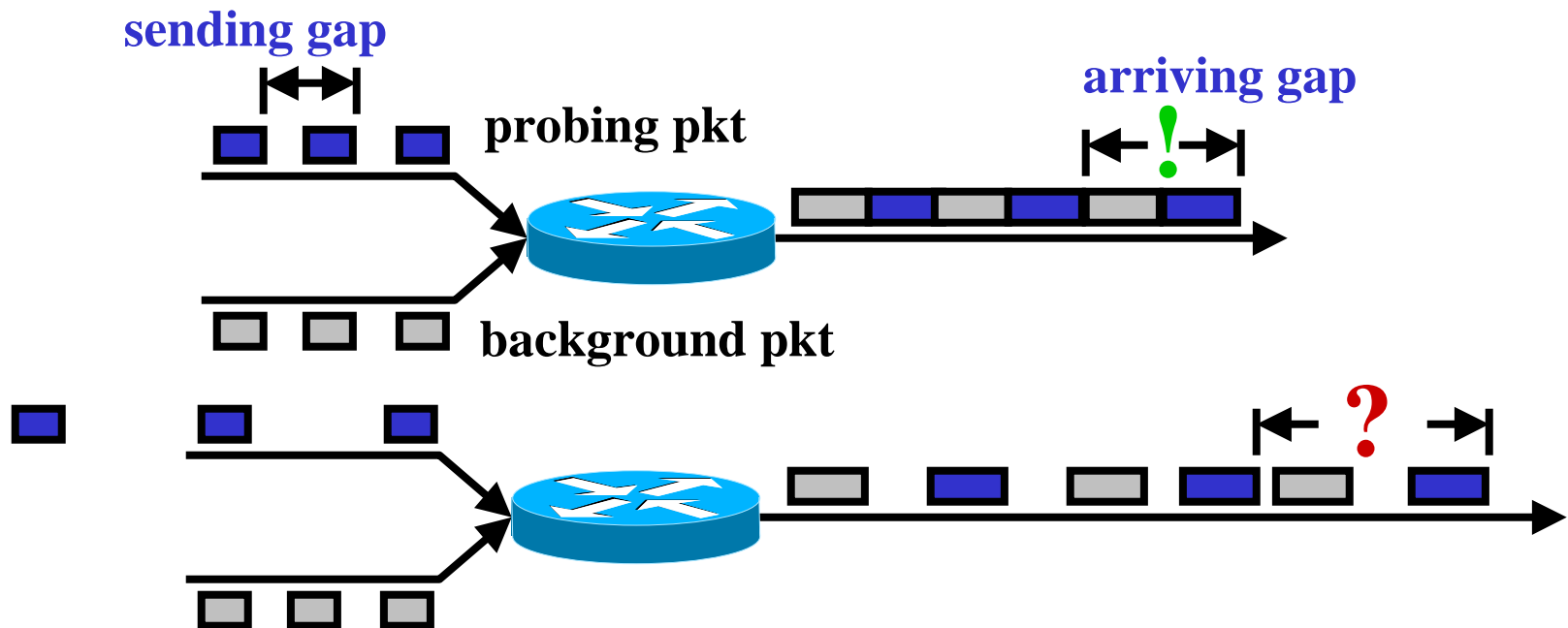
[only consider the slow start at the beginning of a TCP flow]

Outline

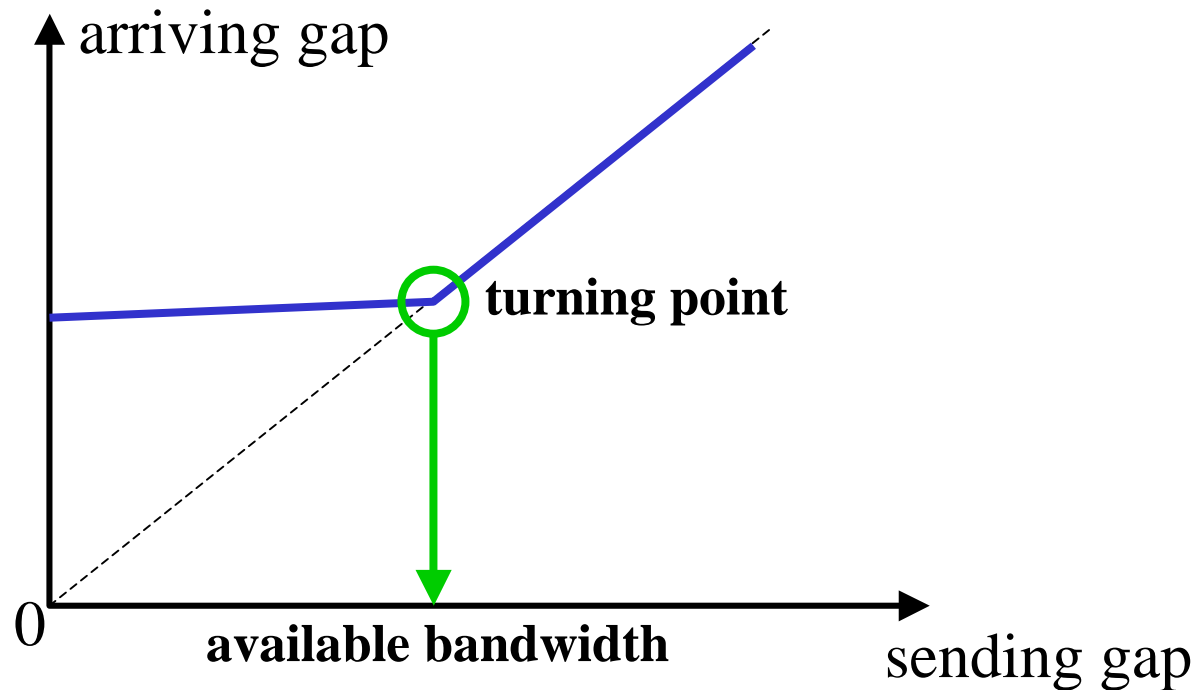
- **Available bandwidth measurement**
- **Algorithm — Paced Start (PaSt)**
- **Evaluation results**

Available Bandwidth Measurement

- PTR (Packet Transmission Rate)
 - A type of packet train probing technique
 - Focus on the probing packet gap
- Observations



Algorithm: PTR



- **Sample the different sending rate**
- **Monitor the difference between the sending rate and the arriving rate**
- **Measure the available bandwidth at the turning point**

Slow Start → Paced Start (PaSt)

- Double the congestion window size every RTT ✓

- Packet sending in startup

~~Self-clocking~~

- ~~• Each ACK triggers 2 data packets~~

– Self-controlled

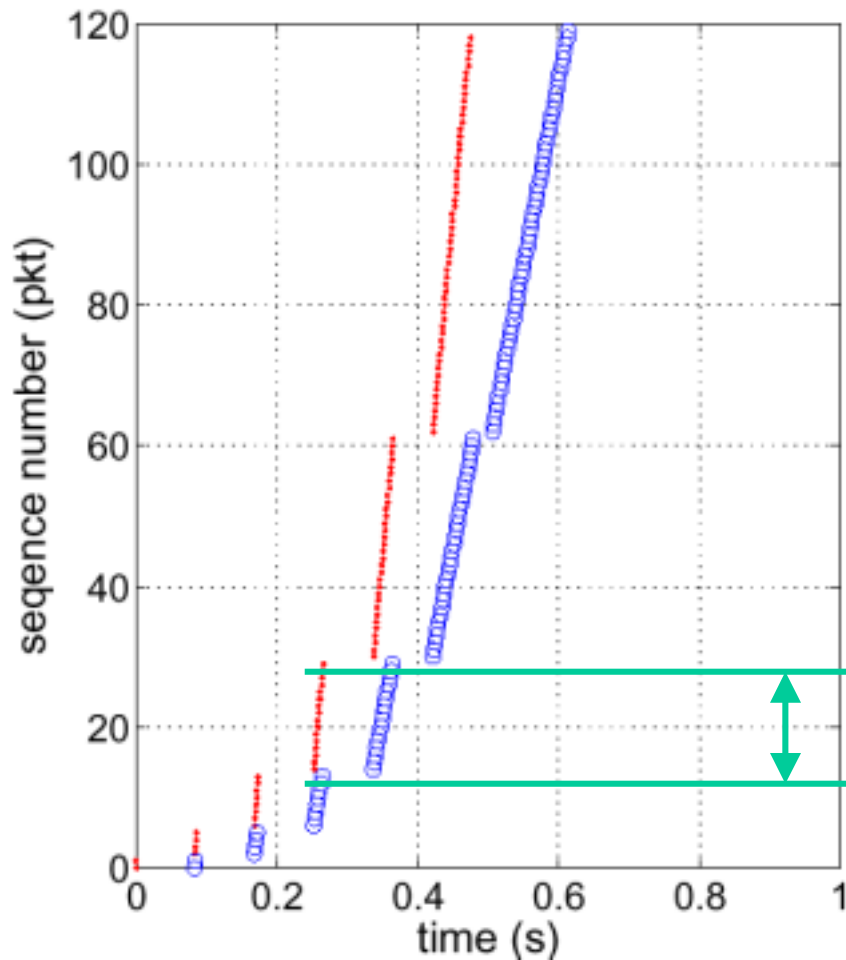
- Sending speed & sending time

- Switch to AIMD

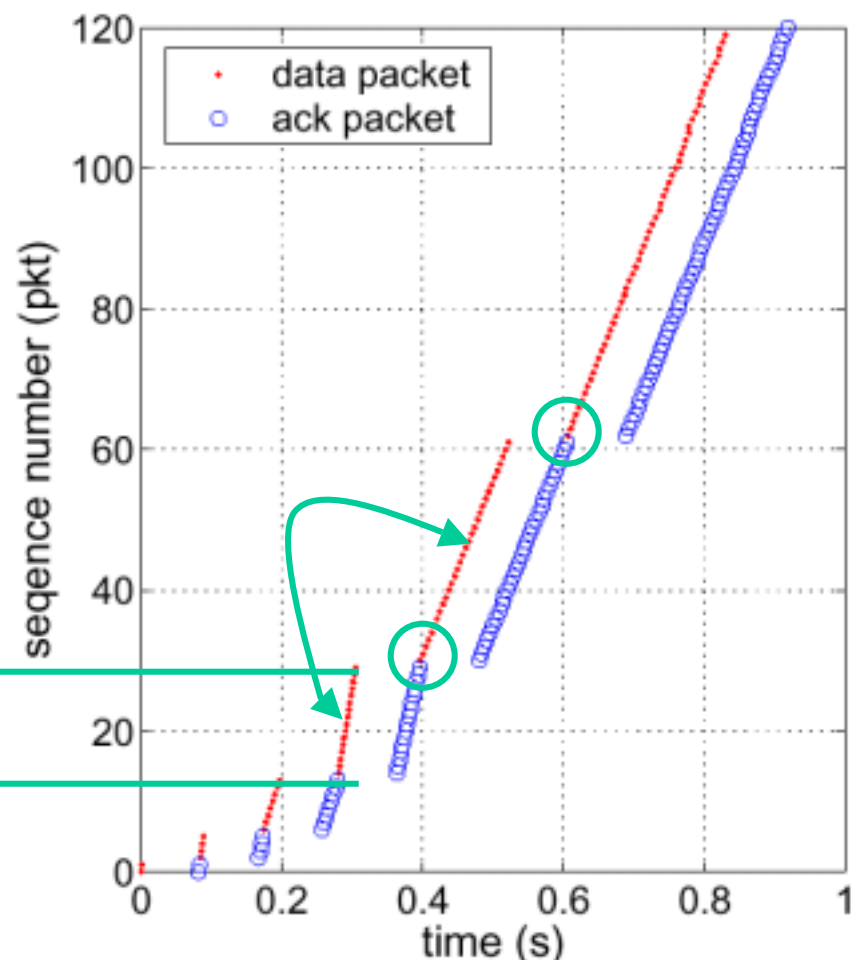
~~$cwnd = ssthresh \mid \text{packet loss} \mid \text{timeout}$~~

– sending rate \approx arriving rate

Slow Start vs. Paced Start



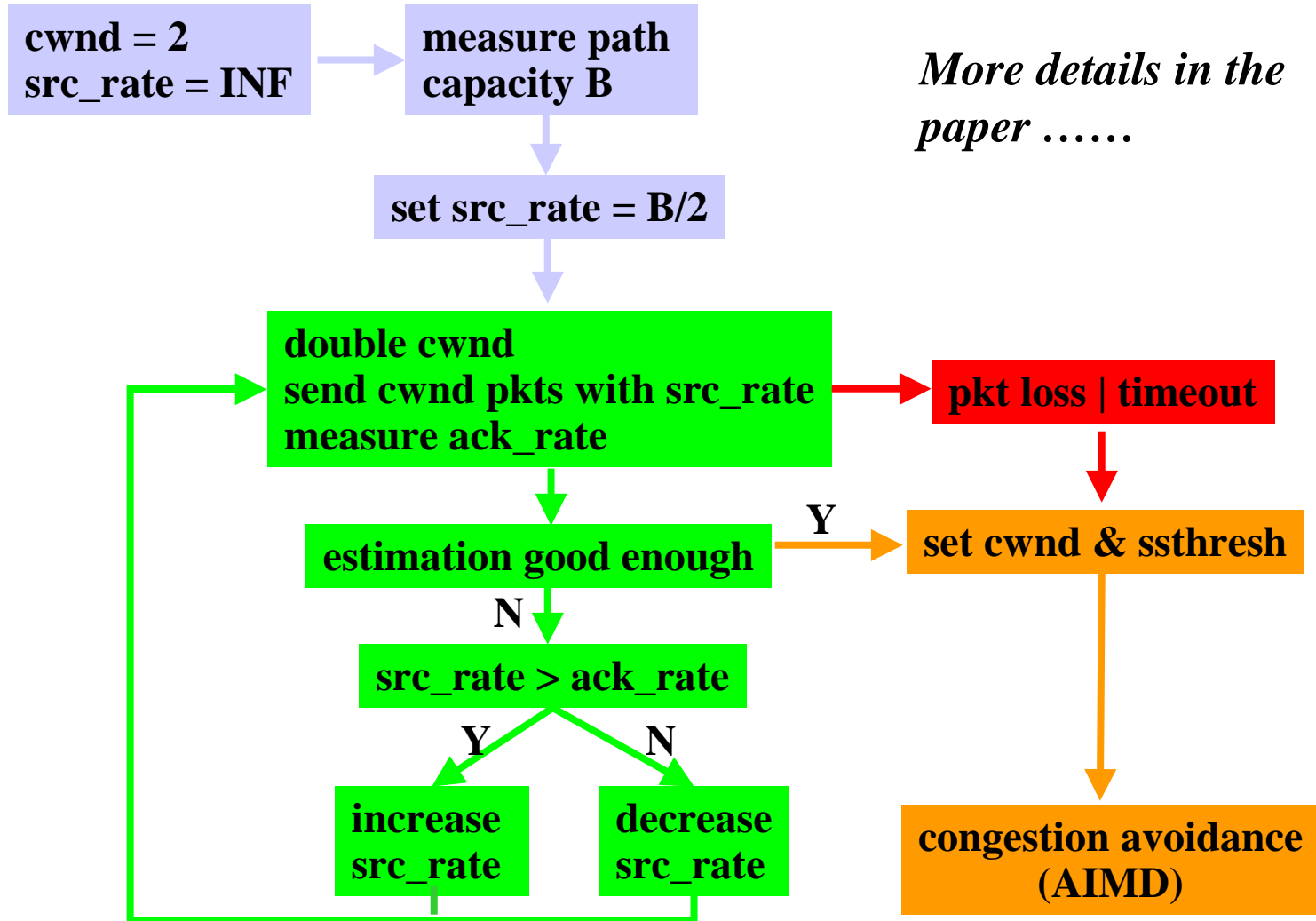
SACK



PaSt

use the ACKing rate to approximate the data packet arriving rate

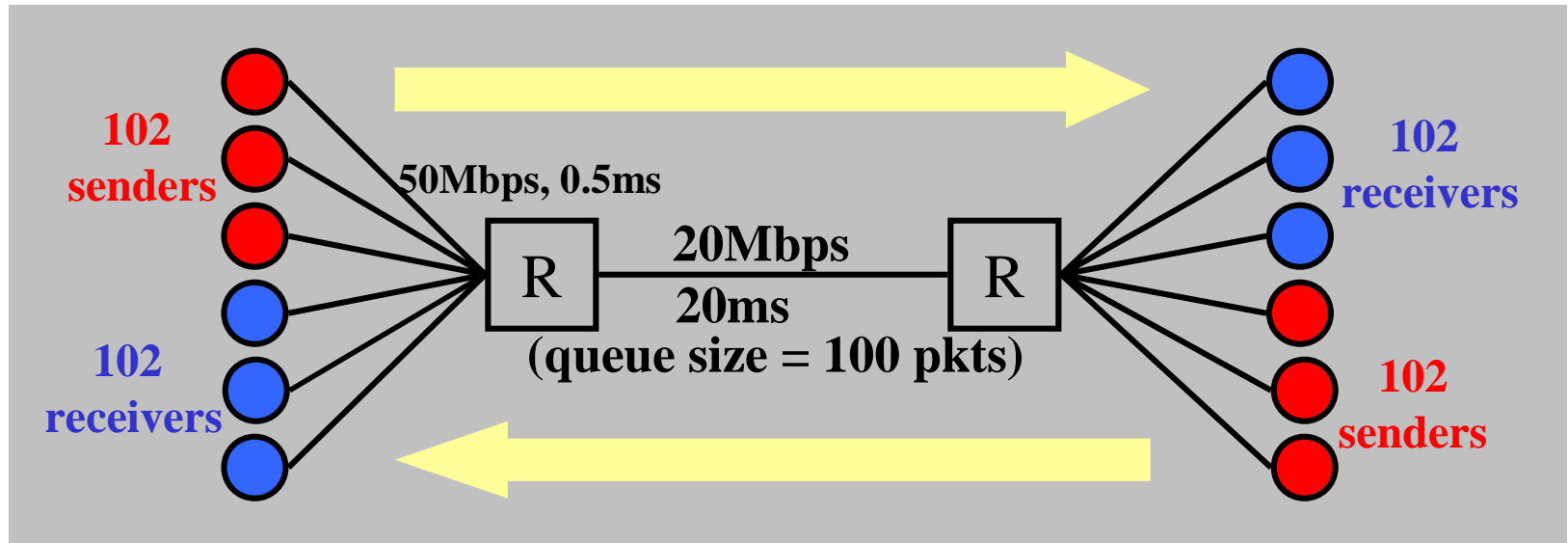
Algorithm — Paced Start



Evaluation

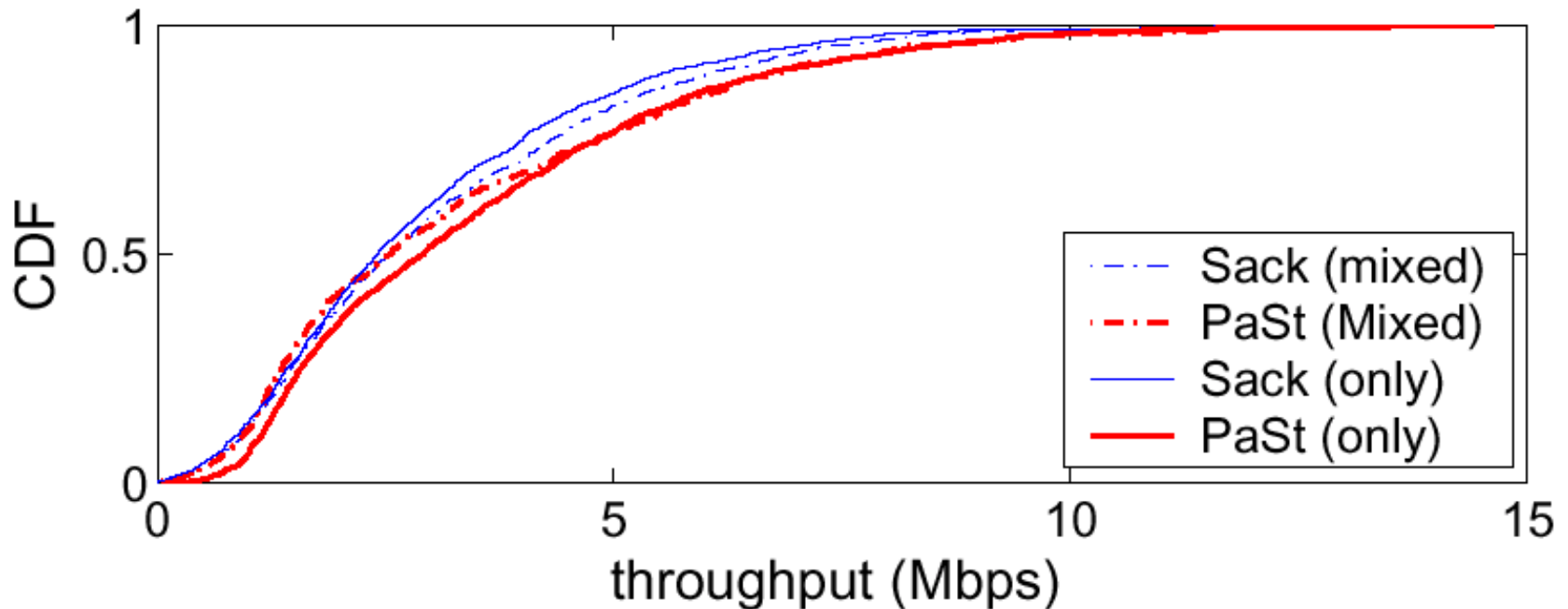
NS2 Simulation			Real system Experiment
• Flow level analysis			Internet experiment (user level TCP)
	PaSt & Sack	PaSt & NewReno & Vegas	
• Network level analysis			Improvement on Apache (Kernel implementation)
	Dumbbell	Parking-lot	

Network Level Analysis



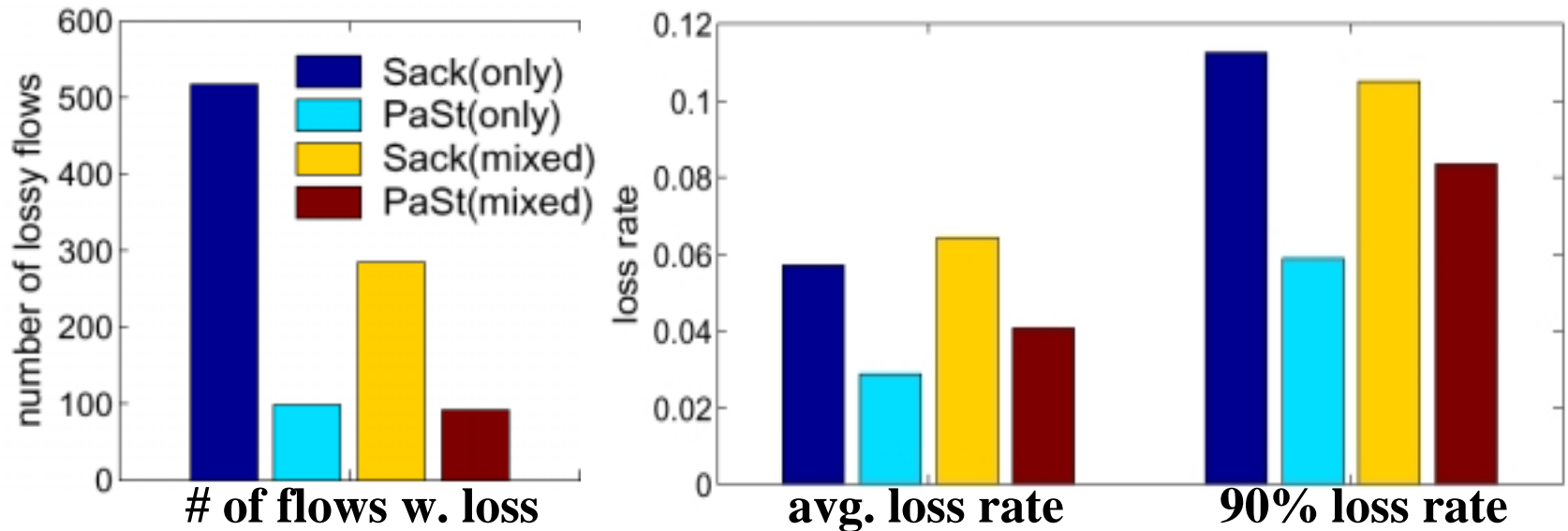
- Exponential flow arriving rate: 2 flows/sec (mean)
- Exponential flow length: 200 packets (mean)
- Three flow scenarios: Sack (only), PaSt (only), Mixed (PaSt & Sack)
- Study 2000 flows in stable status

Throughput



- **PaSt > Sack (5% - 10% improvement)**
- **PaSt co-exists well with Sack**
 - PaSt (mixed) = PaSt (only)
 - Sack (mixed) > Sack (only)

Packet Loss

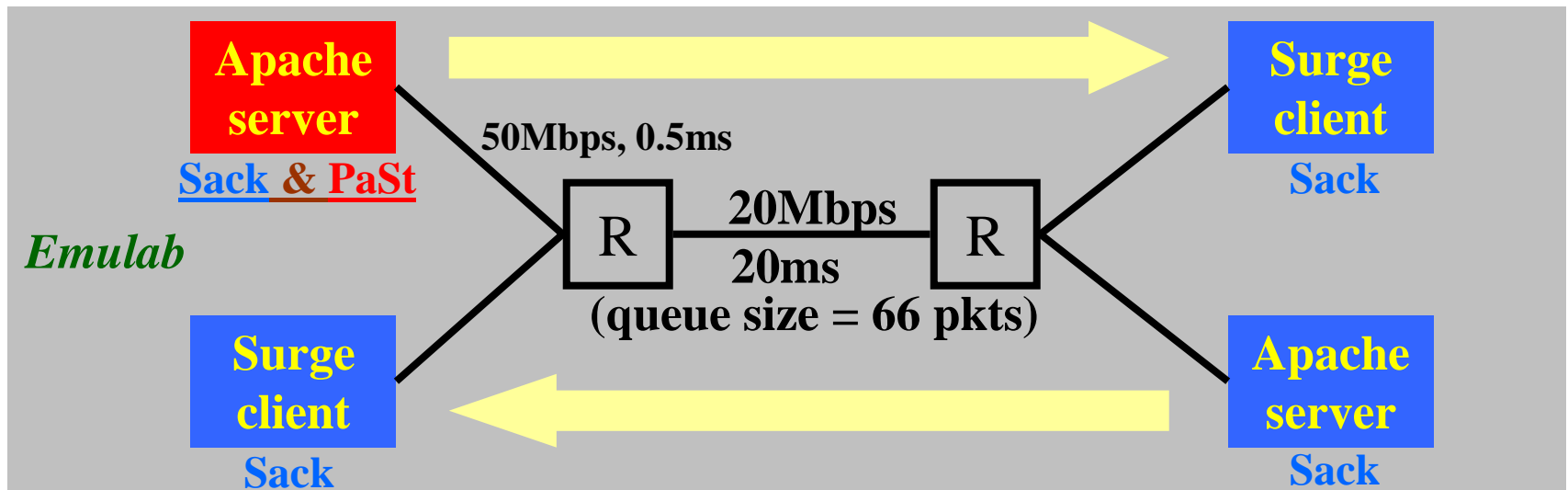


- **PaSt \ll Sack**
- **Slow Start is the main reason for the large number of packet loss in Sack**

Evaluation

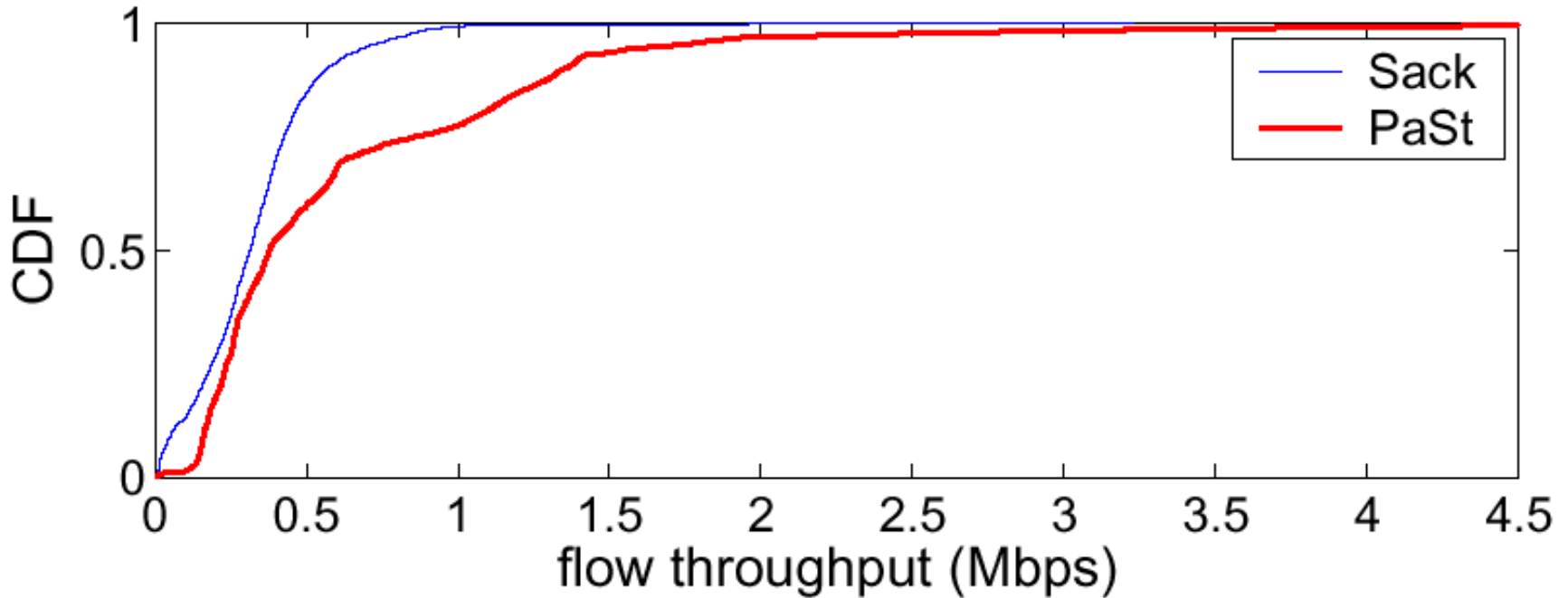
NS2 Simulation			Real system Experiment
• Flow level analysis			Internet experiment (user level TCP)
PaSt & Sack	PaSt & NewReno & Vegas		
• Network level analysis			Improvement on Apache (Kernel implementation)
Dumbbell	Parking-lot	Flow length	

Testbed on *Emulab*



- **Sack kernel: Linux 2.4.18**
- **PaSt kernel:**
 - **Implement PaSt algorithm in Linux 2.4.18**

Performance of Apache



- **Analyze 2000 HTTP flows**
- **Throughput: PaSt > Sack**
- **Loss: PaSt (1168 pkt loss) << Sack (94186 pkt loss)**

Related Work

- **TCP NewReno**
- **TCP Vegas**
- **Congestion Manager**

More in the paper

Conclusion

- **The design of Paced Start algorithm**
- **The performance of Paced Start**
 - **Less aggressive during startup**
 - **Less packet loss**
 - **Smaller startup time**
 - **Better network level performance**

Questions?