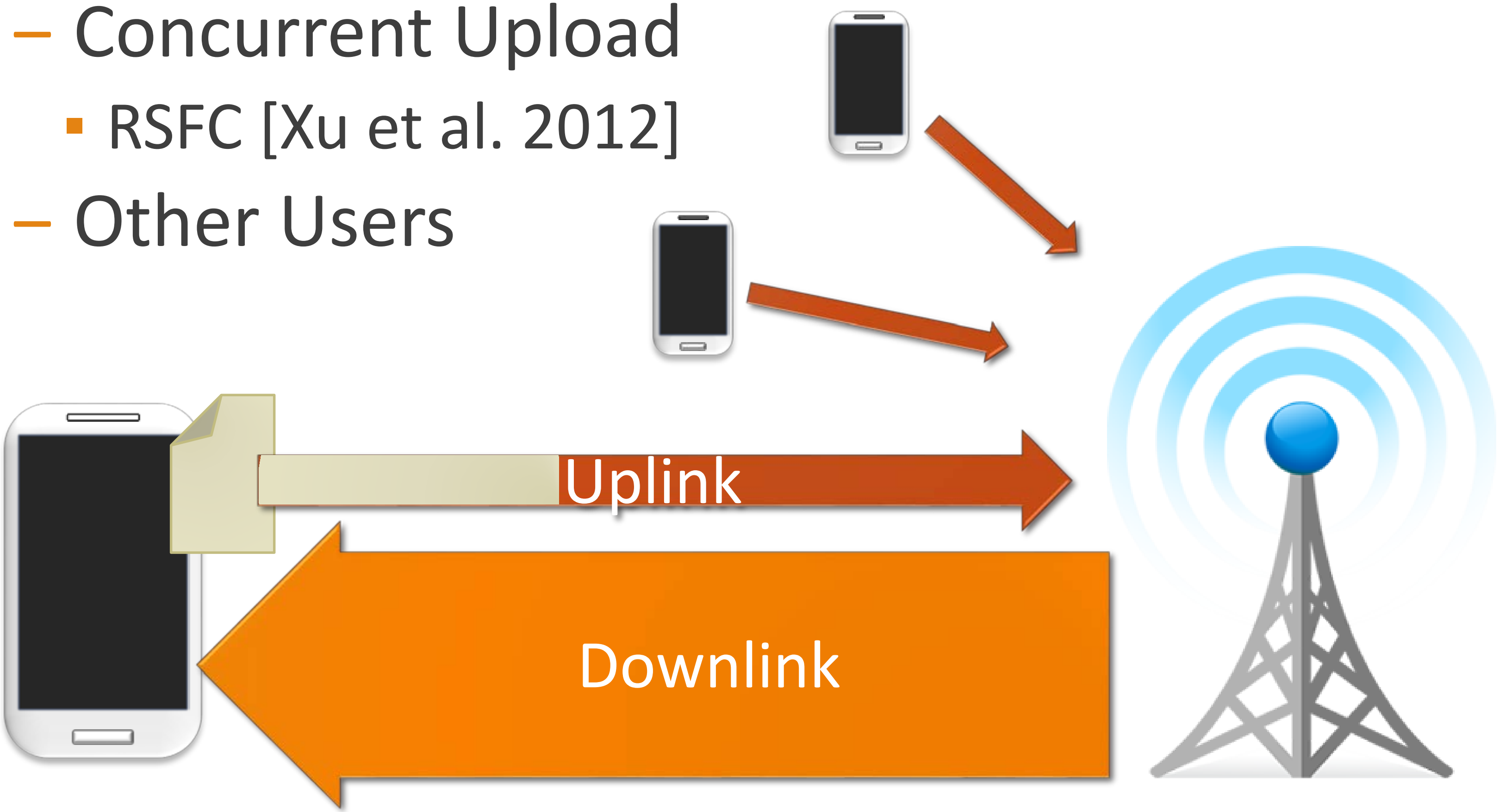
Mitigating Egregious ACK Delays in Cellular Data Networks by Eliminating TCP ACK Clocking

Wai Kay Leong, Yin Xu, Ben Leong, Zixiao Wang

National University of Singapore

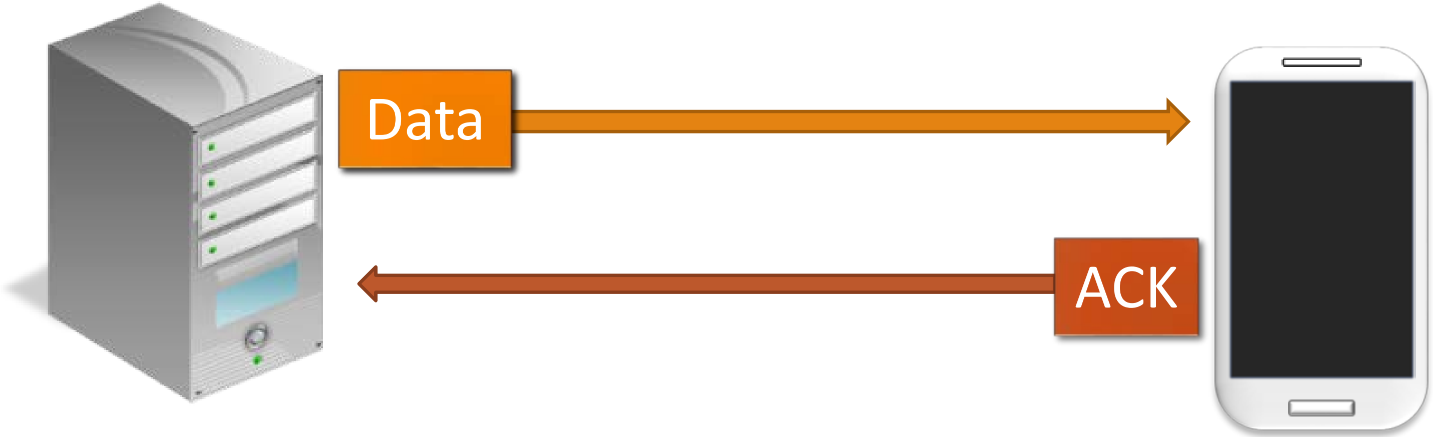
Asymmetry in Cellular Networks

- **Congestion in Uplink**
  - Concurrent Upload
    - RSFC [Xu et al. 2012]
  - Other Users



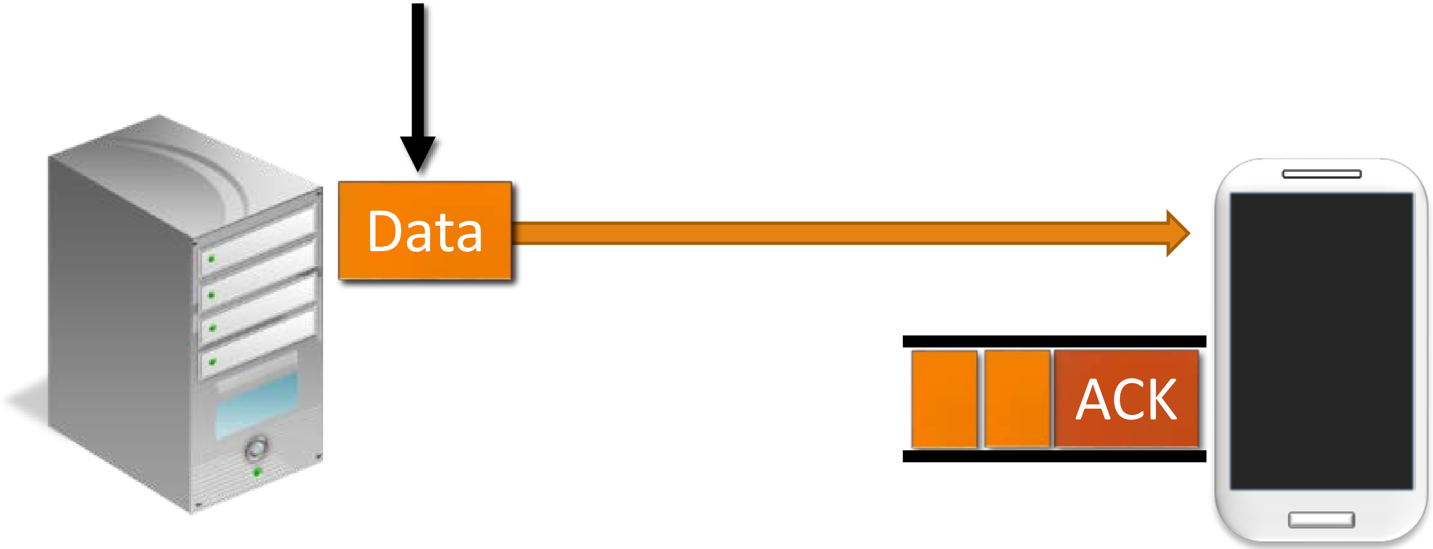
Egregious ACK Delays

- **TCP congestion control is ACK-clocked**



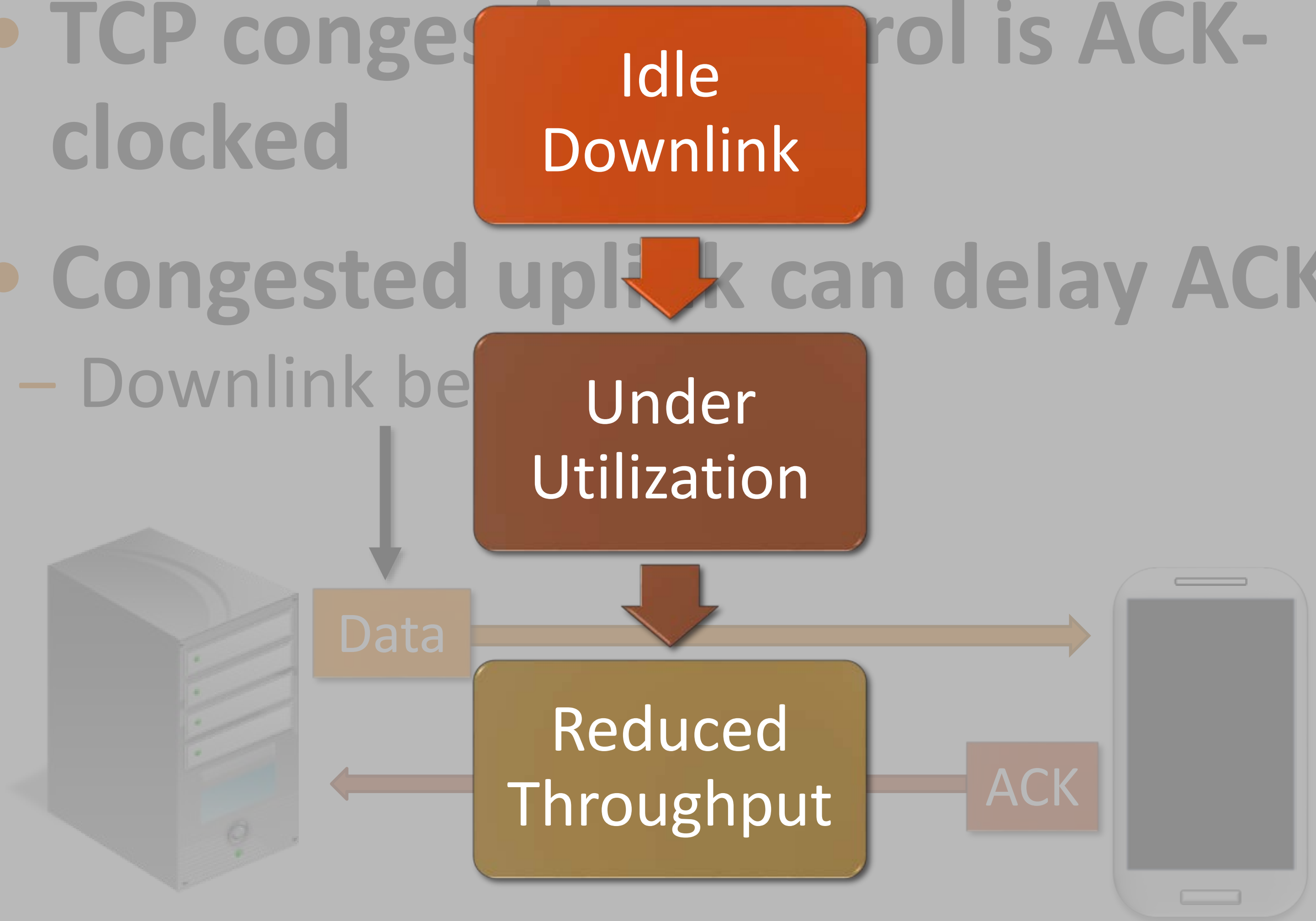
Egregious ACK Delays

- **TCP congestion control is ACK-clocked**
- **Congested uplink can delay ACKs**
  - Downlink becomes idle



Egregious ACK Delays

- TCP conges rol is ACK-clocked
- Congested upli k can delay ACKs
  - Downlink be

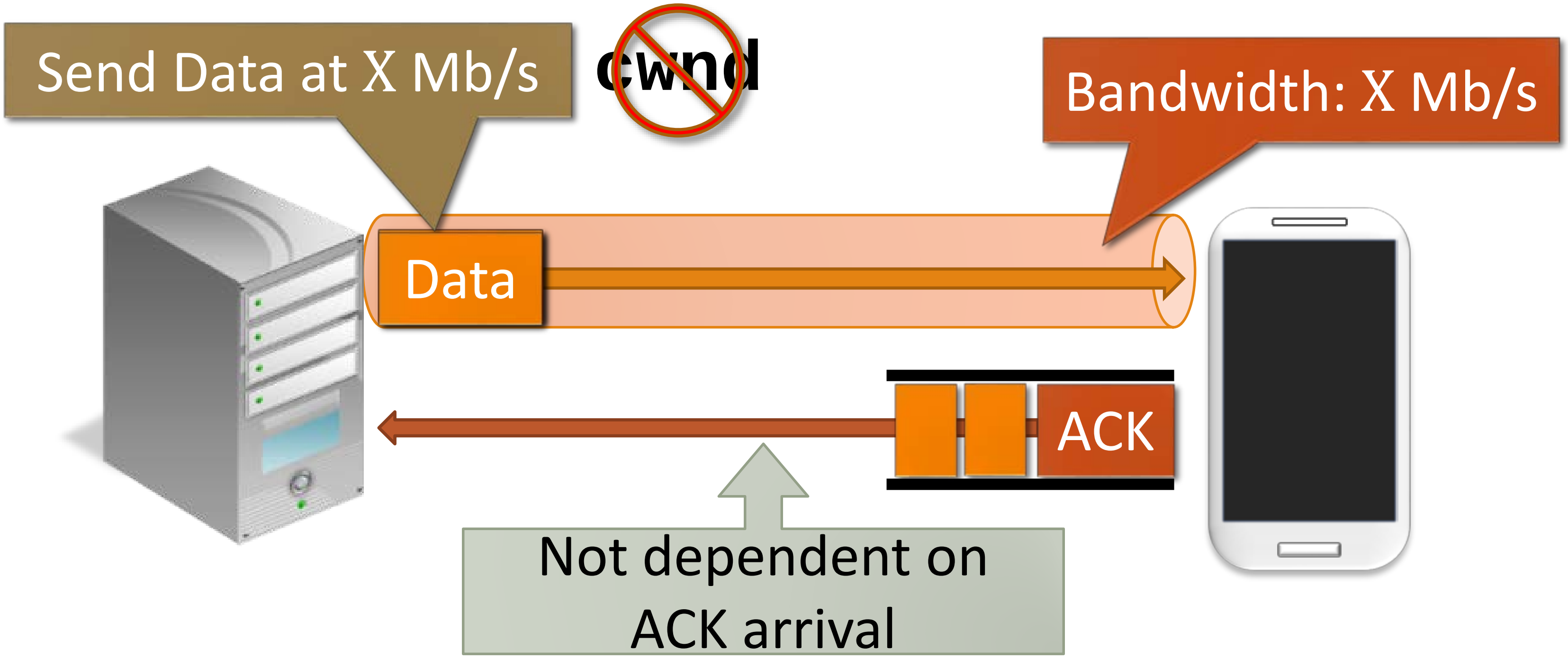

Idle Downlink

Under Utilization

Reduced Throughput

Solution: Eliminate ACK Clocking

Idea: If we know the bandwidth, we can send at maximum rate.



Send Data at X Mb/s

Bandwidth: X Mb/s

Not dependent on ACK arrival

Challenge 1: Estimating Bandwidth

Idea Bandwidth ≡ Receive Rate

- Use receiving rate as equivalent of available bandwidth

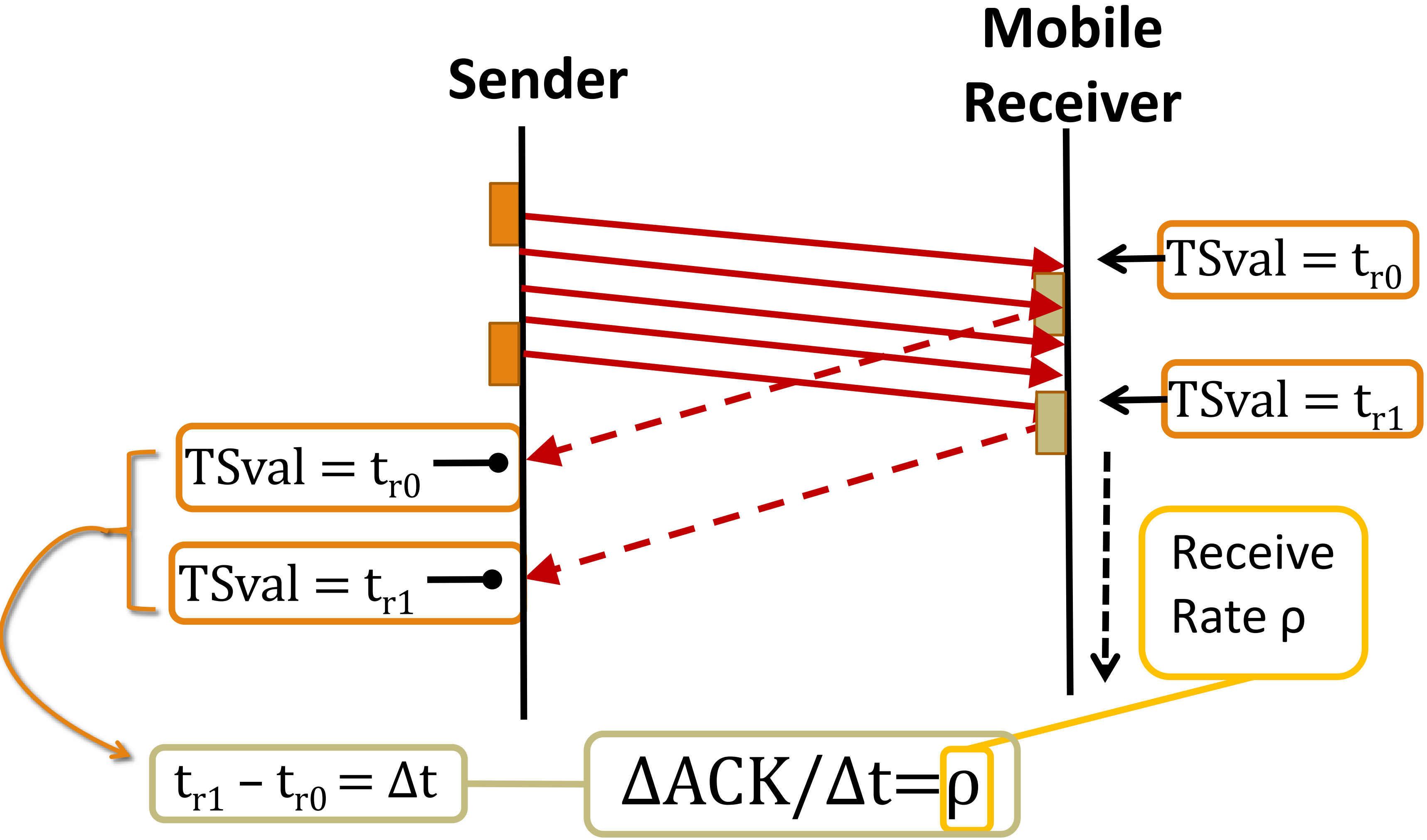
Condition Done Passively

- To avoid modifications at the receiver

Solution Use TCP Timestamps

- Enabled by default on Android and iPhones

Estimating Receive Rate


Sender
Mobile Receiver
$\mathrm{TSval} = t_{r0}$
$\mathrm{TSval} = t_{r1}$
$\mathrm{TSval} = t_{r0}$
$\mathrm{TSval} = t_{r1}$
Receive Rate ρ
$t_{r1} - t_{r0} = \Delta t$
$\Delta ACK/\Delta t = \rho$

Challenges

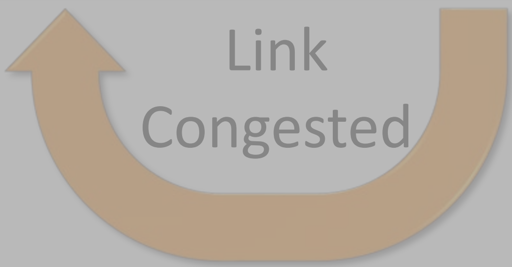
1. **Estimating Bandwidth**
2. **Timestamp Granularity too Coarse**
   - Cannot estimate with high accuracy
3. **Bandwidth variation**
   - Have to keep updating estimation

Solution

Self-oscillating Feedback Loop

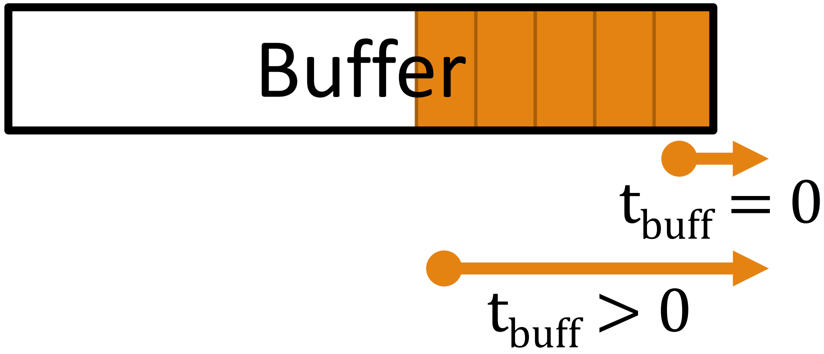
- Estimate Receive Rate ρ
- Send Rate σ

How to detect congestion?



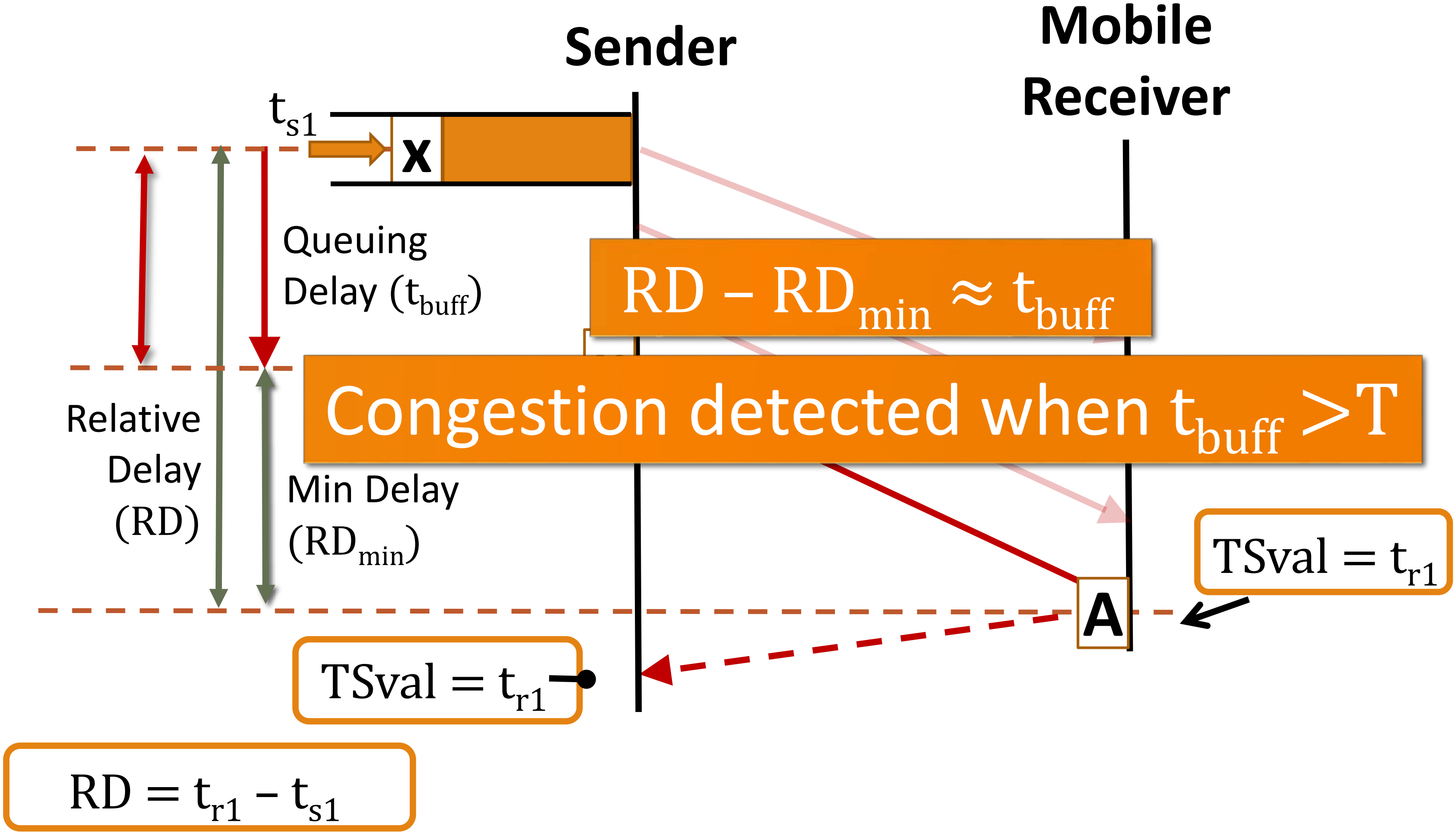
Detect Congestion

- **Idea: Monitor Queuing Delay**



- **How?**
  – TCP Timestamps
  – **Relative Difference** between sender and receiver

Detecting Congestion

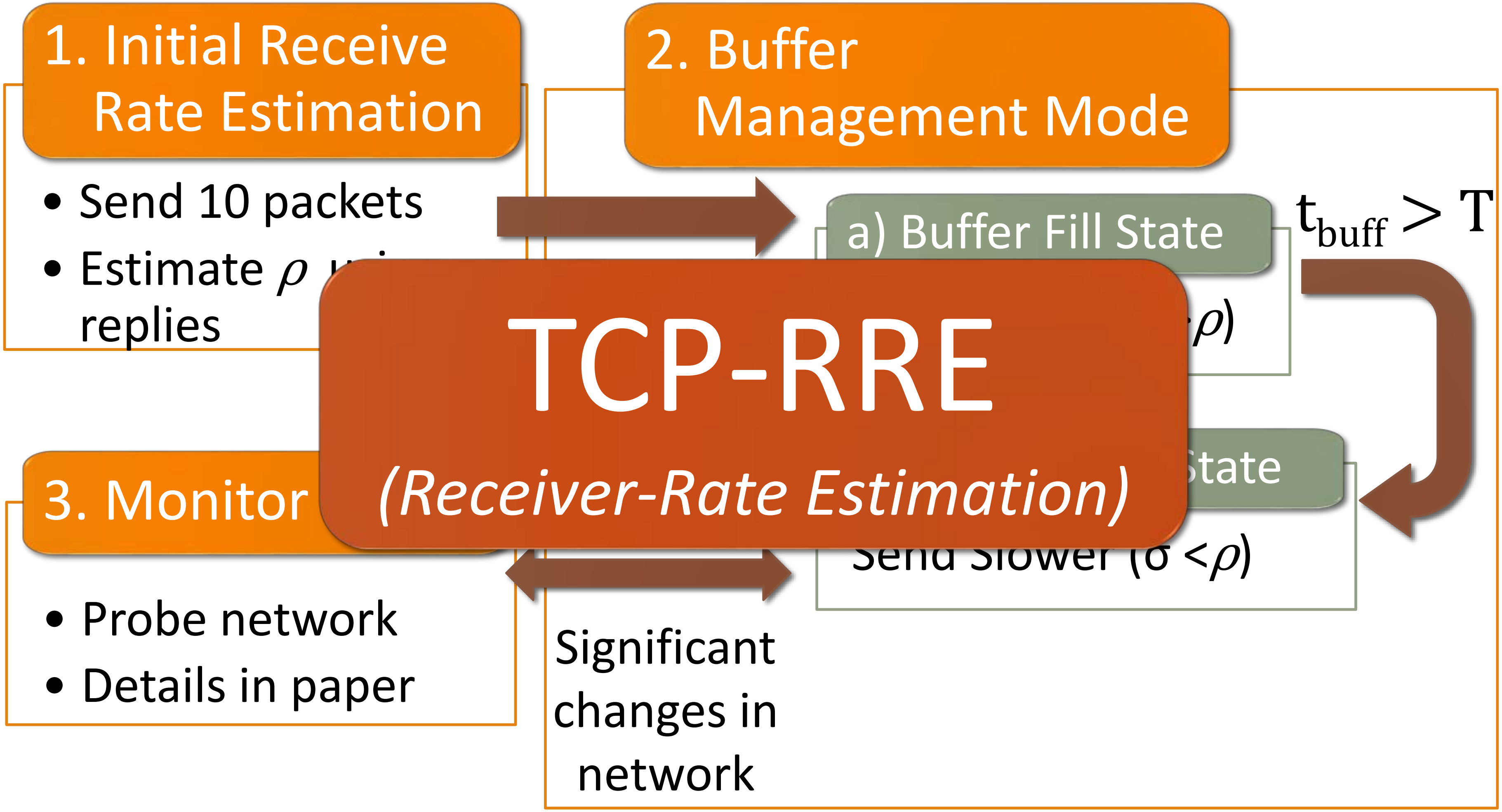


$RD – RD_{min} \approx t_{buff}$

Congestion detected when $t_{buff} > T$

$RD = t_{r1} – t_{s1}$

Summary of Algorithm



1. Initial Receive Rate Estimation

- Send 10 packets
- Estimate ρ u i replies

2. Buffer Management Mode

a) Buffer Fill State

ρ)

$t_{buff} > T$

State

Send Slower ($\sigma < \rho$)

3. Monitor

- Probe network
- Details in paper

Significant changes in network

TCP-RRE

(Receiver-Rate Estimation)

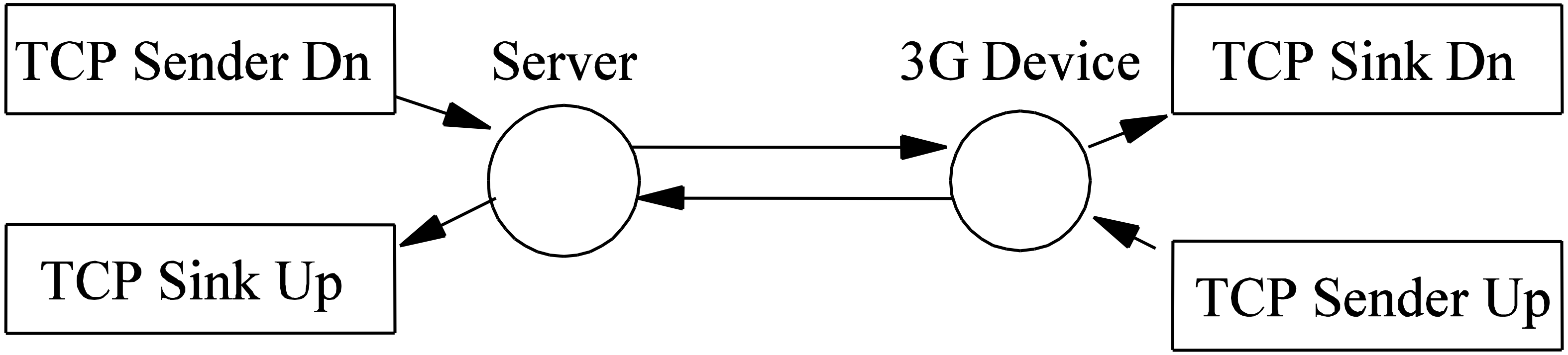
Parameters?

- **How much faster or slower to send?**
- **What threshold T to use?**
- **When to switch to monitor state?**

See details in paper

ns-2 Evaluation

Measured real networks to get simulation parameters



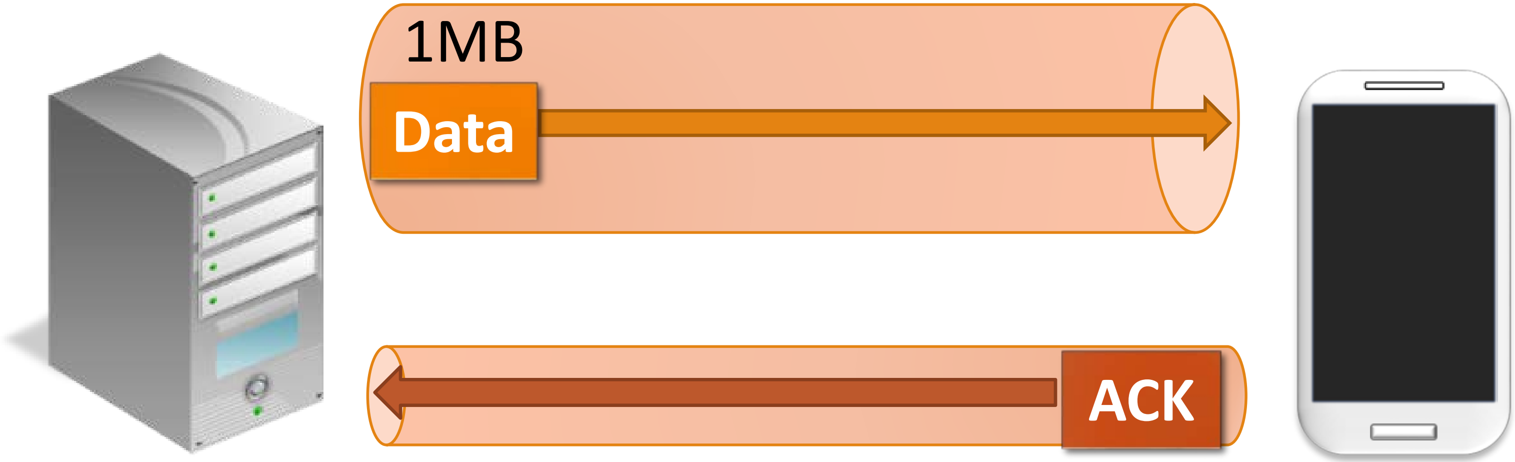
ns-2 Evaluation

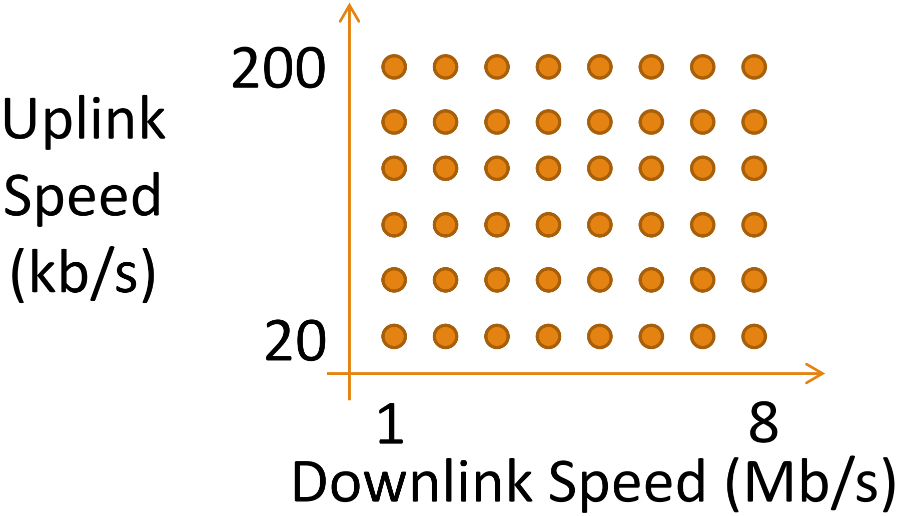
1. **Single Download with Slow Uplink**
2. **Single Download under Normal Conditions**
3. **Download with Concurrent Upload**
4. **Handling Network Fluctuation**
5. **TCP Friendliness**

ns-2 Evaluation

1. **Single Download with Slow Uplink**
2. **Single Download under Normal Conditions**
3. **Download with Concurrent Upload**
4. **Handling Network Fluctuation**
5. **TCP Friendliness**

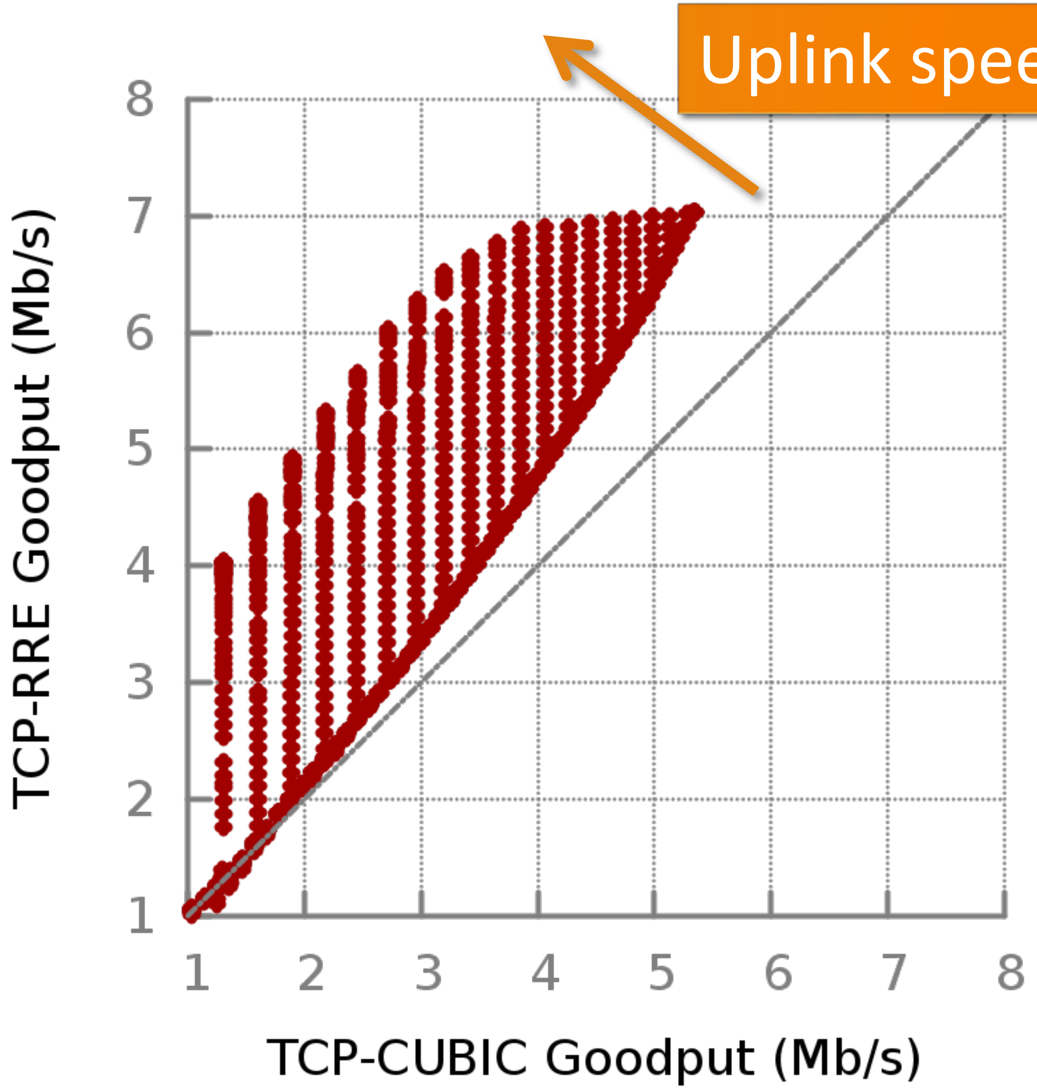
Download with Slow Uplink





ns-2

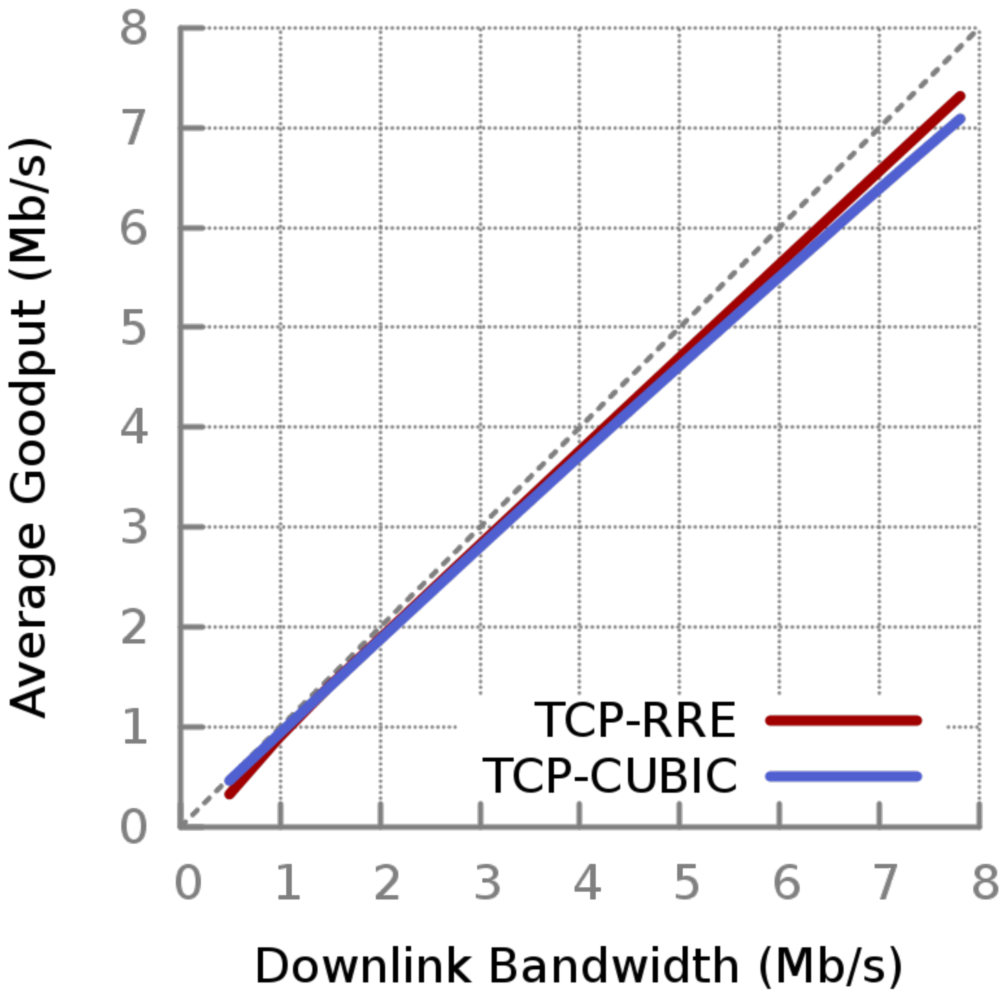
Download with Slow Uplink



Uplink speed decreases

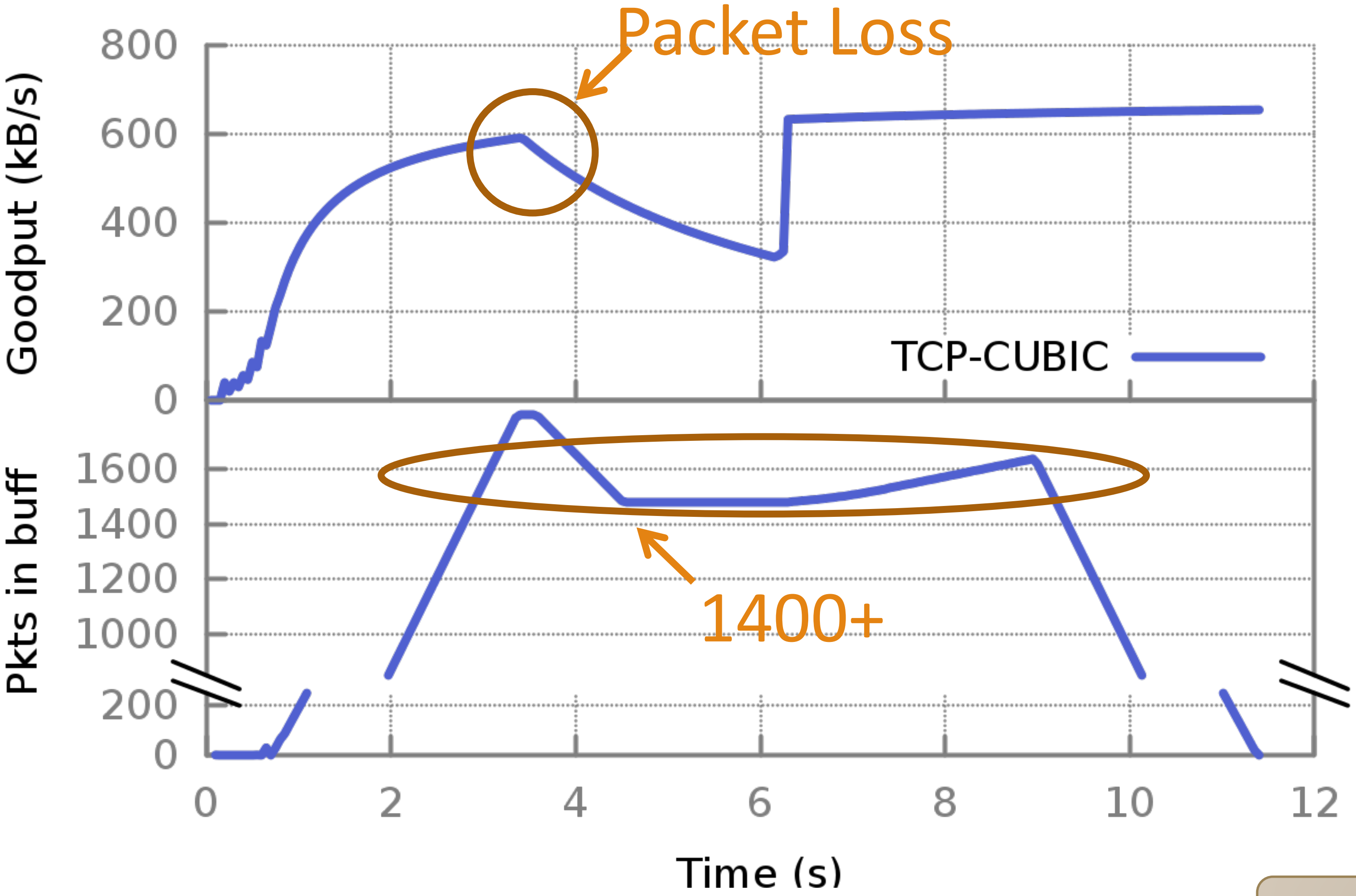
ns-2

Download under Normal Conditions



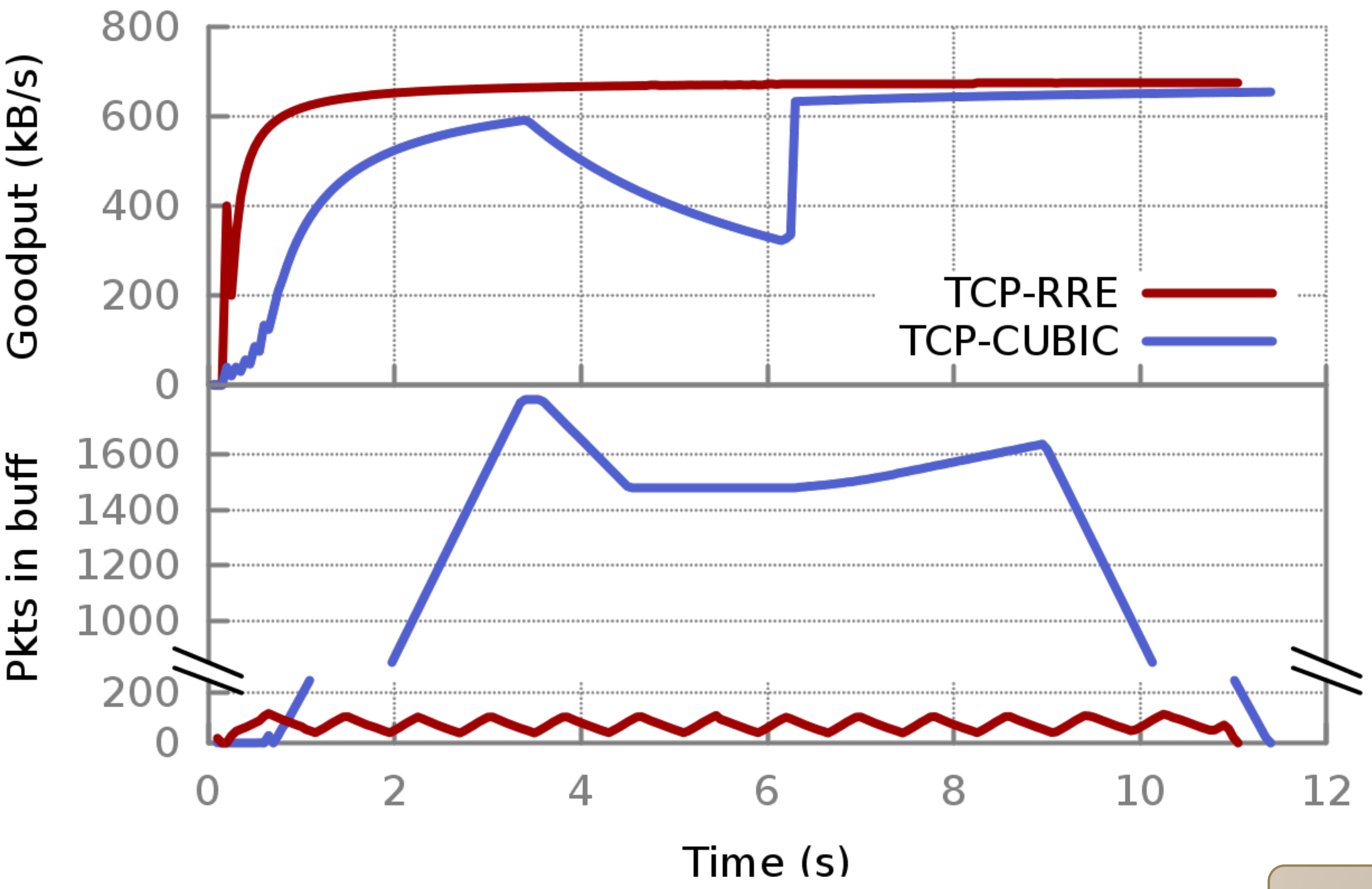
ns-2

Download under Normal Conditions



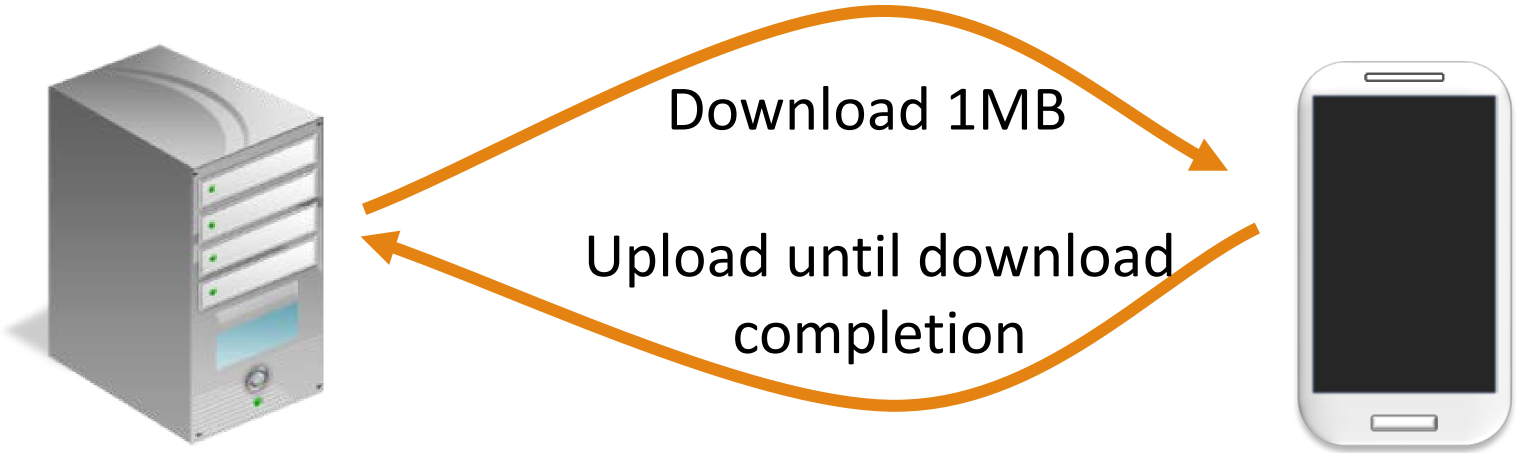
ns-2

Download under Normal Conditions



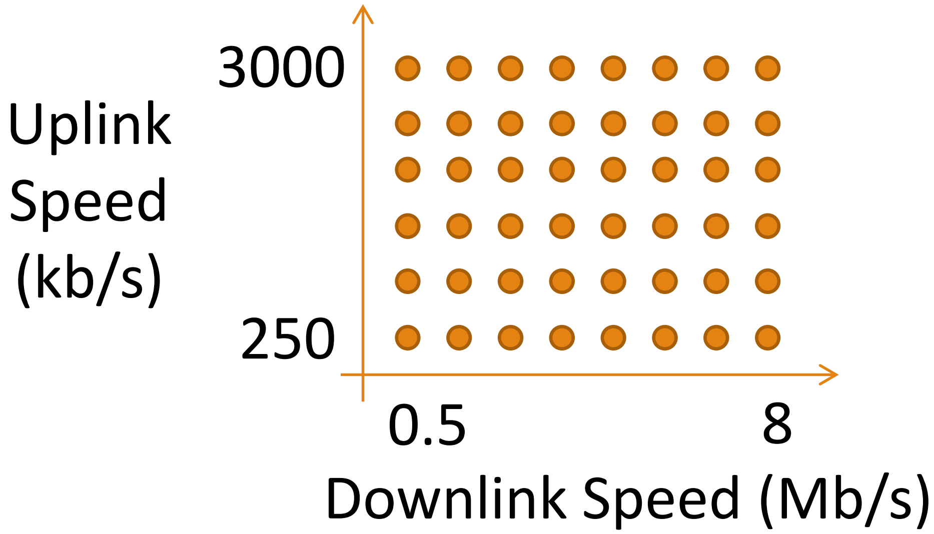
ns-2

Download with Concurrent Upload



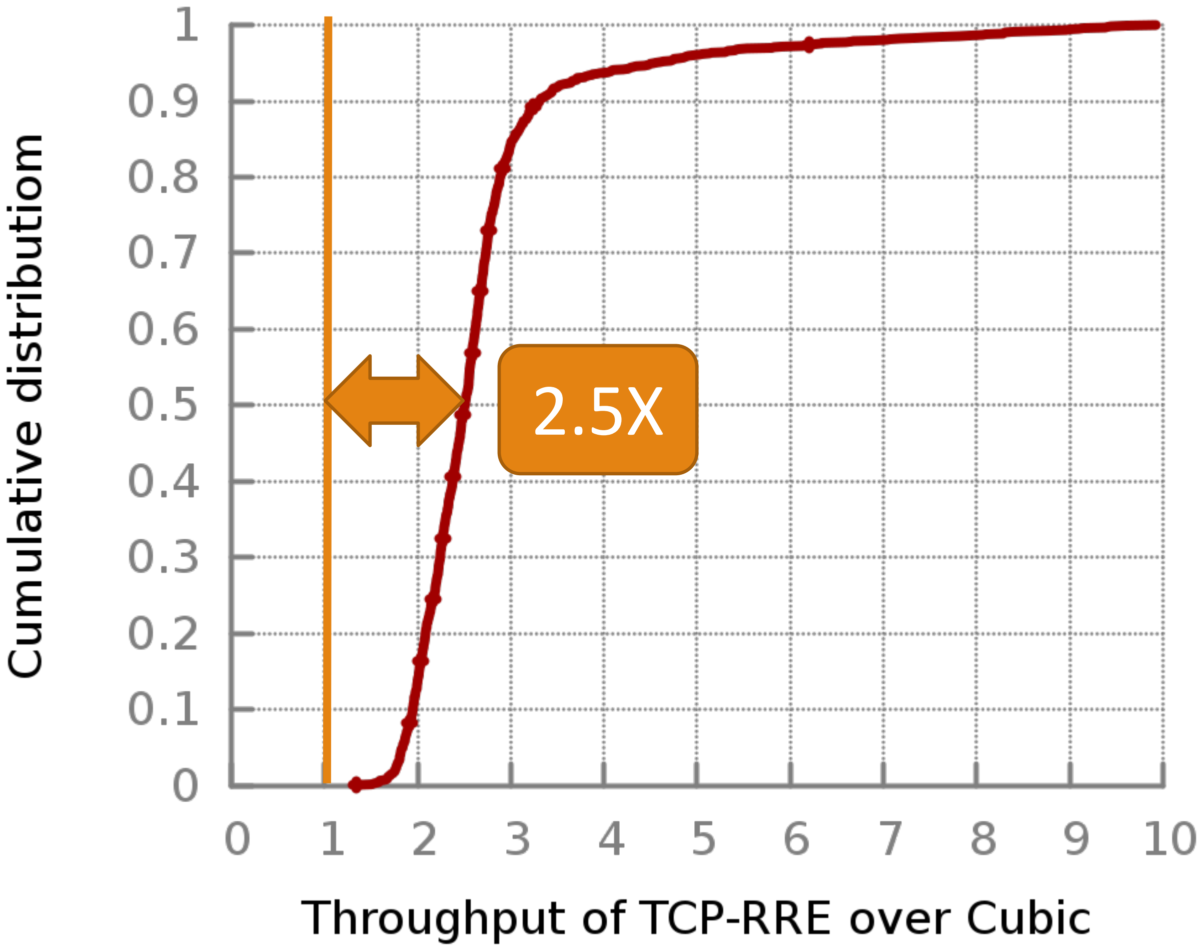
Download 1MB

Upload until download completion



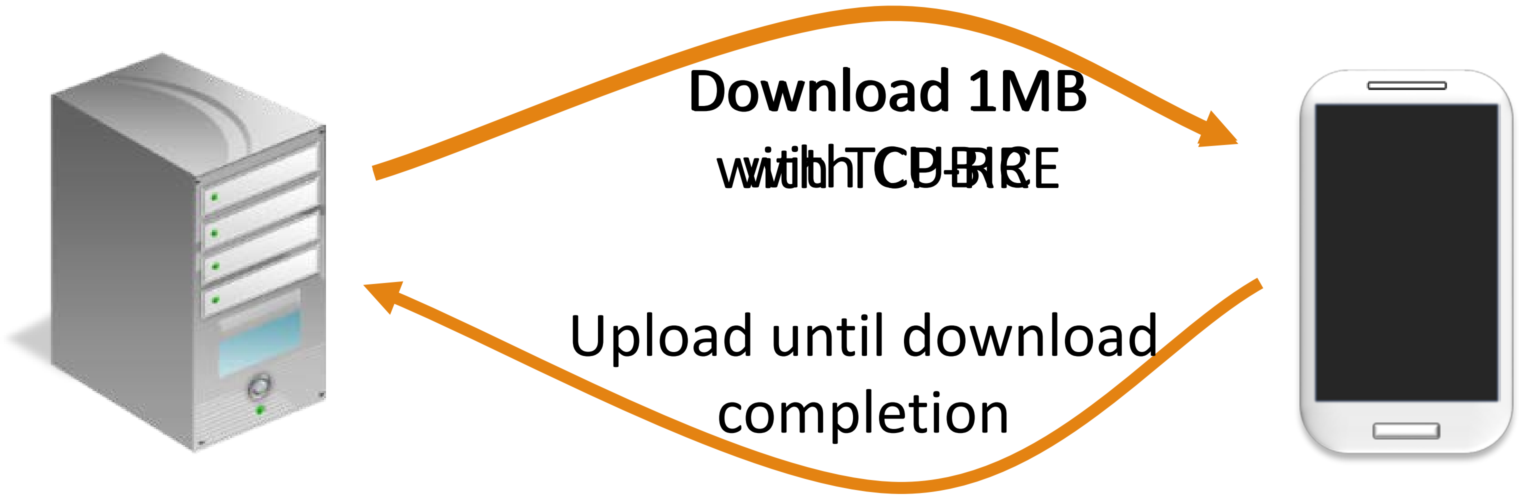
ns-2

Download with Concurrent Upload



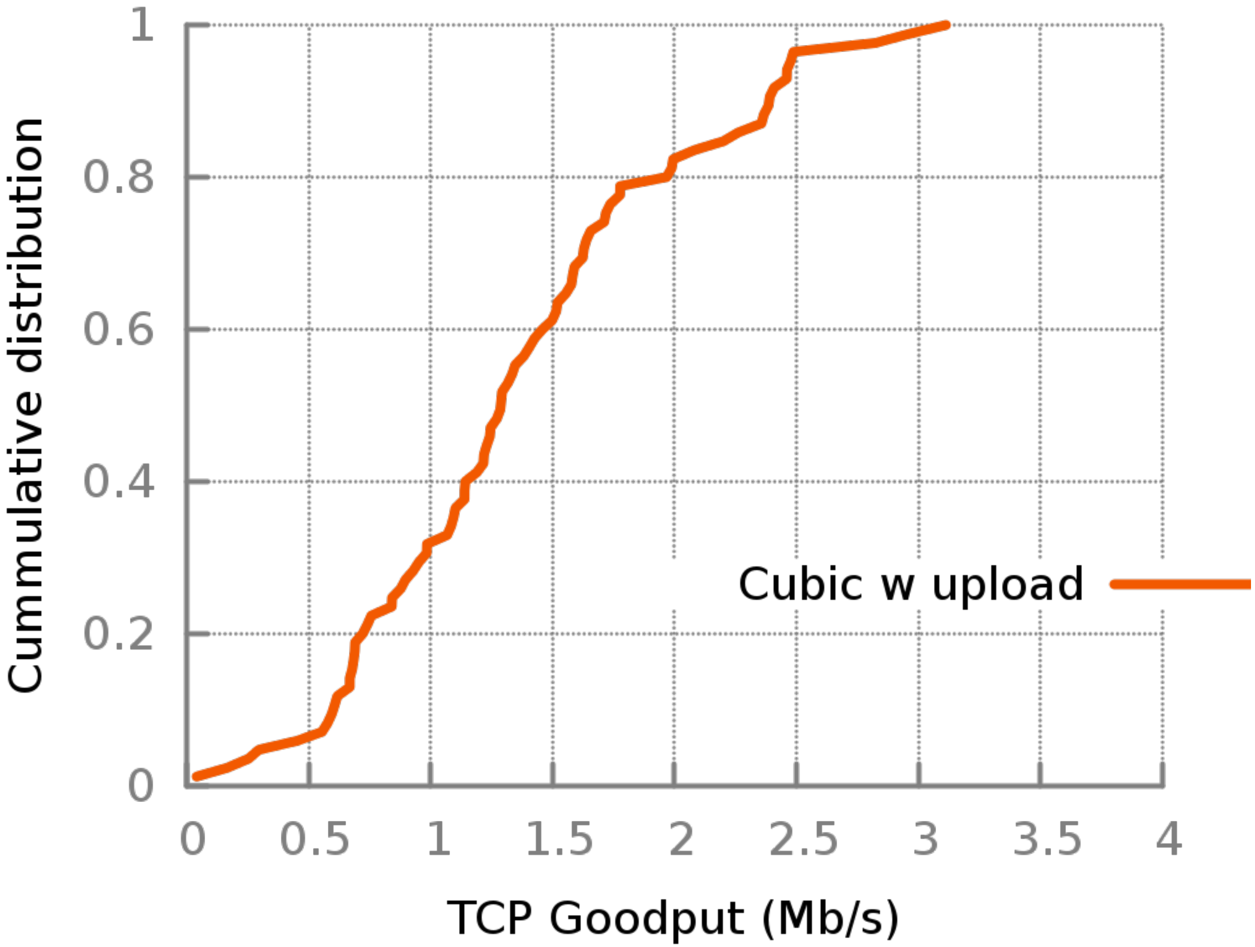
ns - 2

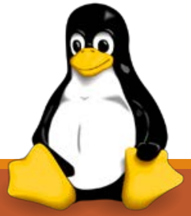
Evaluation in Linux



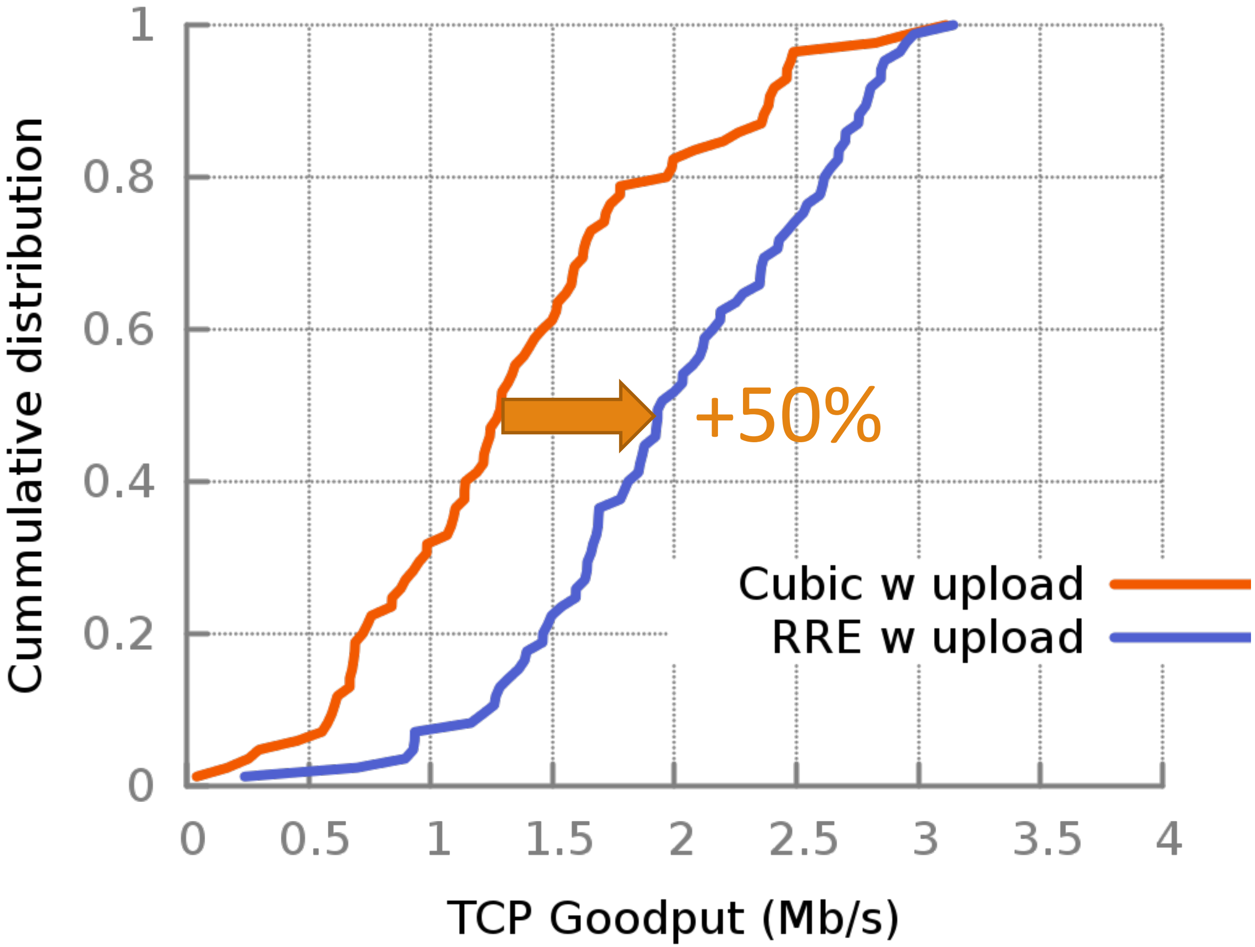
- **Several Places, different ISPs**
- **Multiple times**
- **CDF of all experiments at each place for each ISP**

Evaluation in Linux

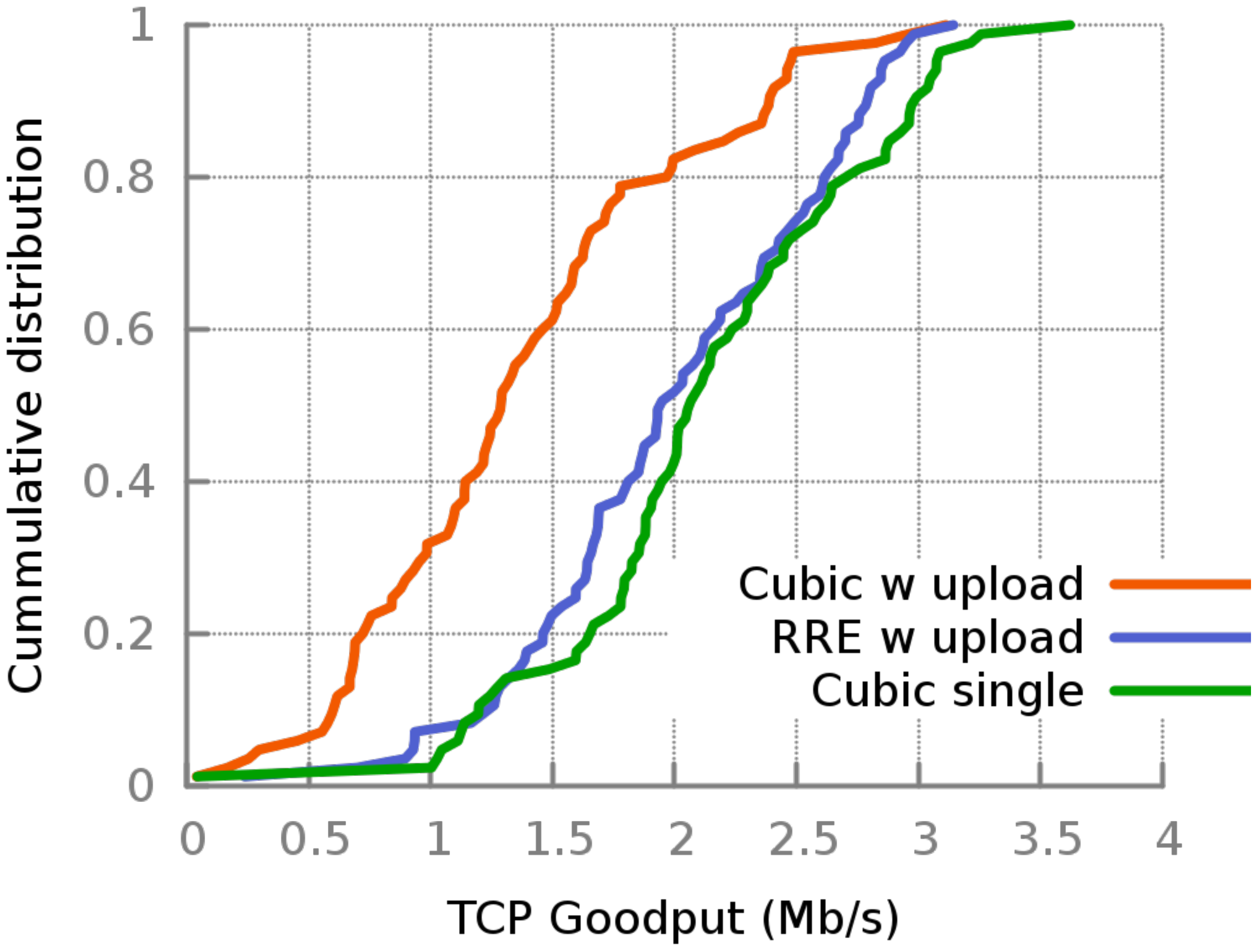




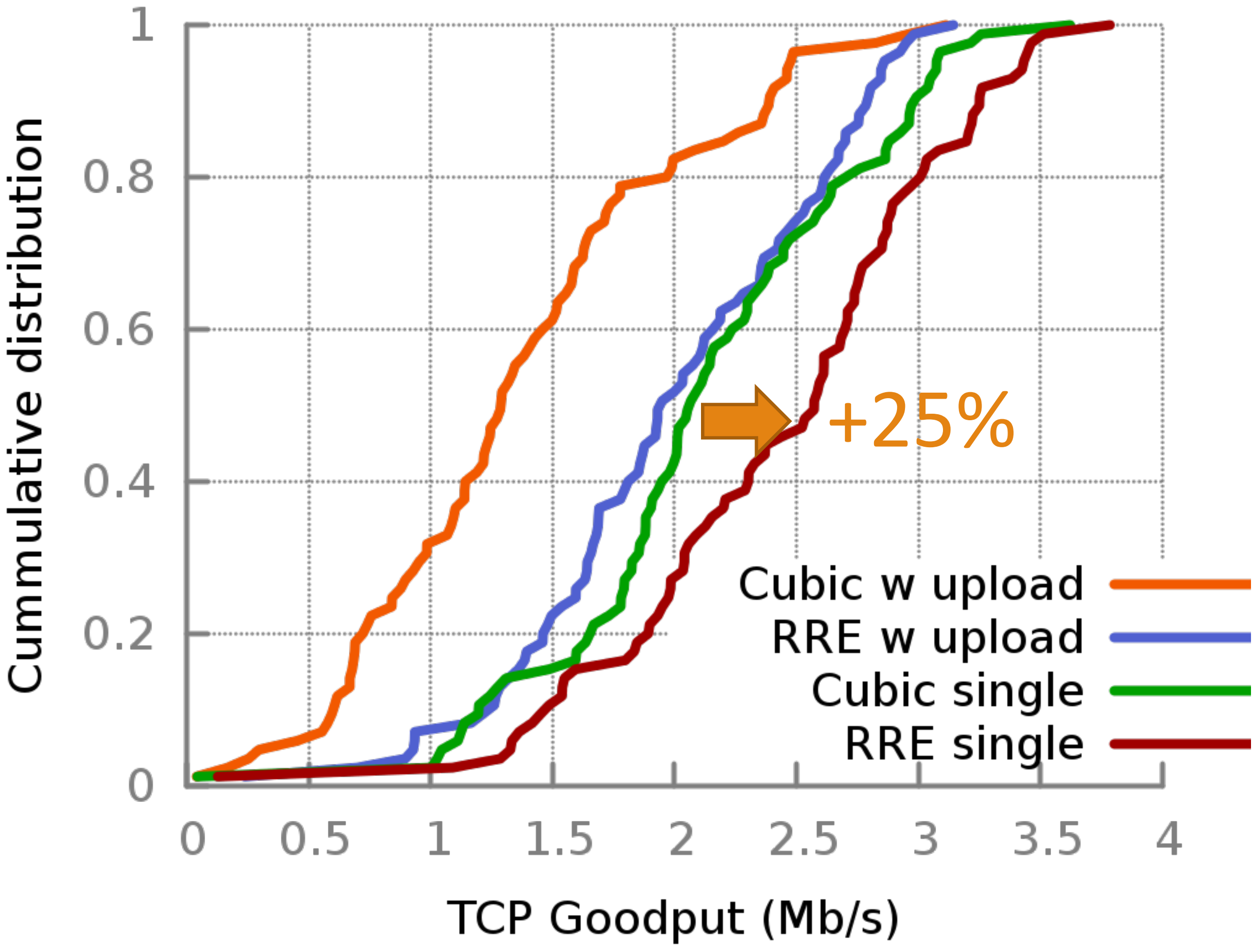
Evaluation in Linux



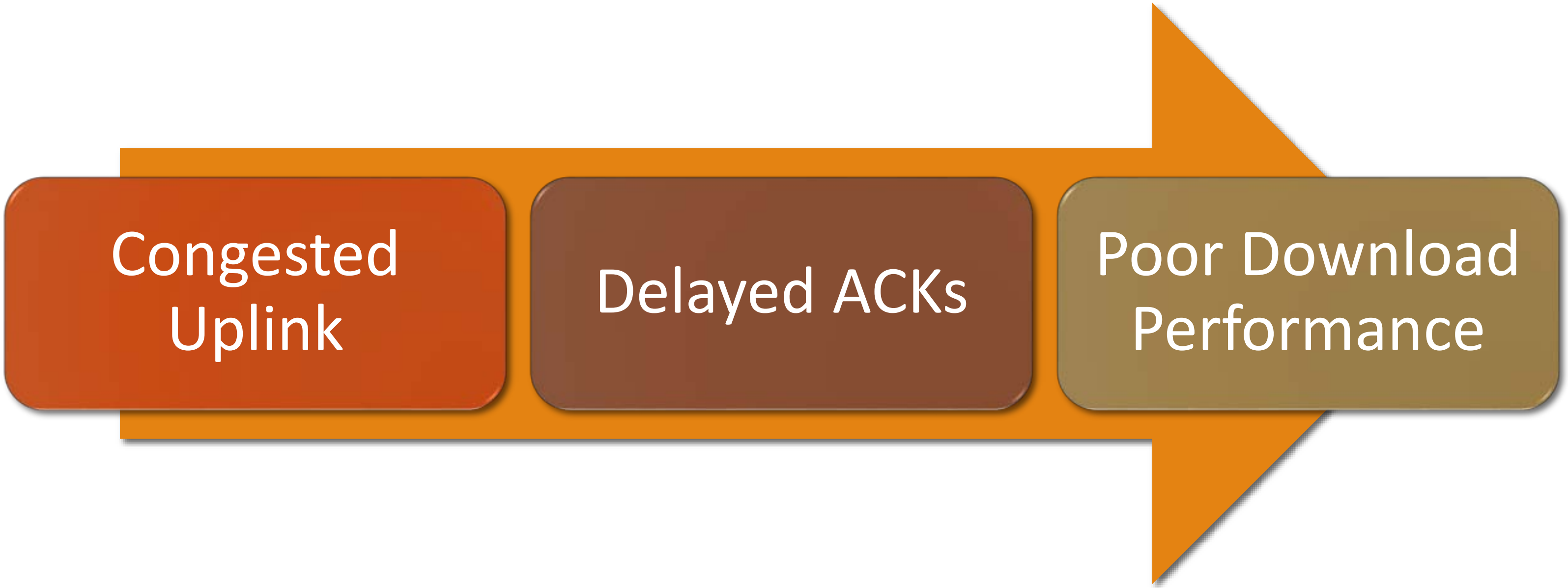
Evaluation in Linux



Evaluation in Linux



Conclusion



Congested Uplink

Delayed ACKs

Poor Download Performance

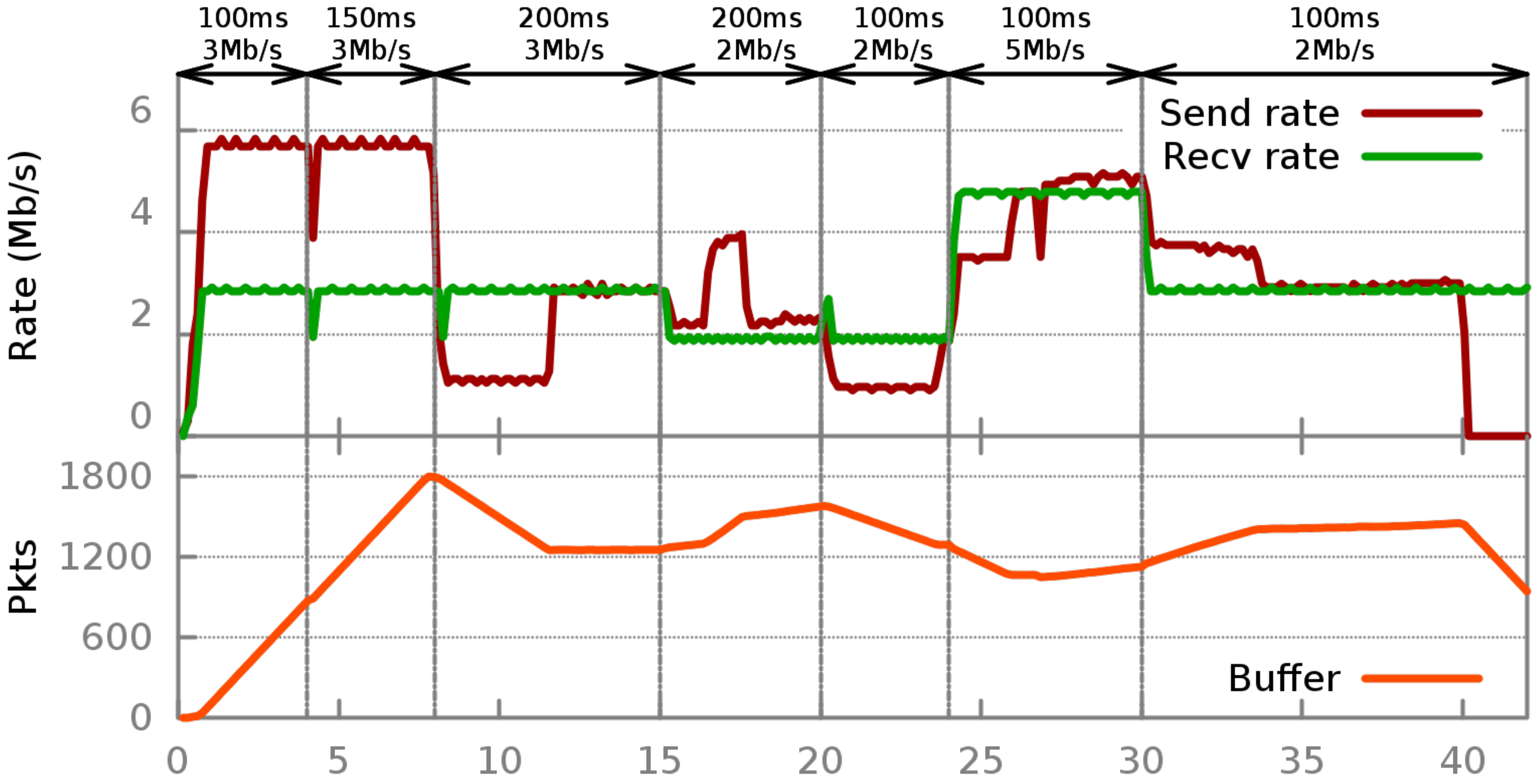
Conclusion

- **TCP-RRE**
  - ~~ACK Clocking~~
  - Rate Control with Feedback Loop
- **Use TCP Timestamp**
  - Estimate Receive Rate
  - Detect Congestion
- **Improves TCP**
  - Uplink is Slow
  - Uplink is Congested
- **Keep the Delay Low**
- **Fair to Other TCP Flows**

Thank You

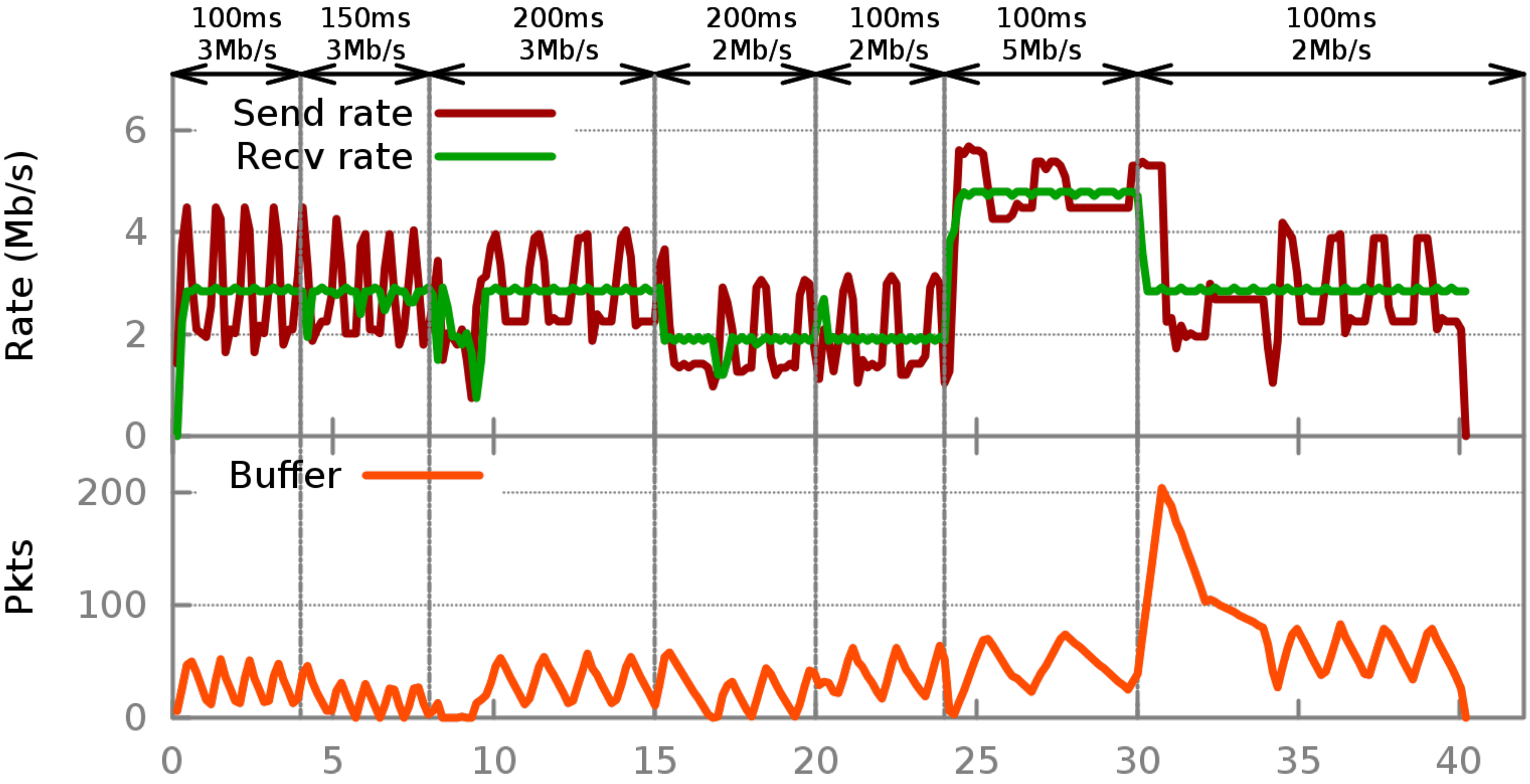
QUESTIONS

Handling Network Fluctuations



CUBIC

Handling Network Fluctuations



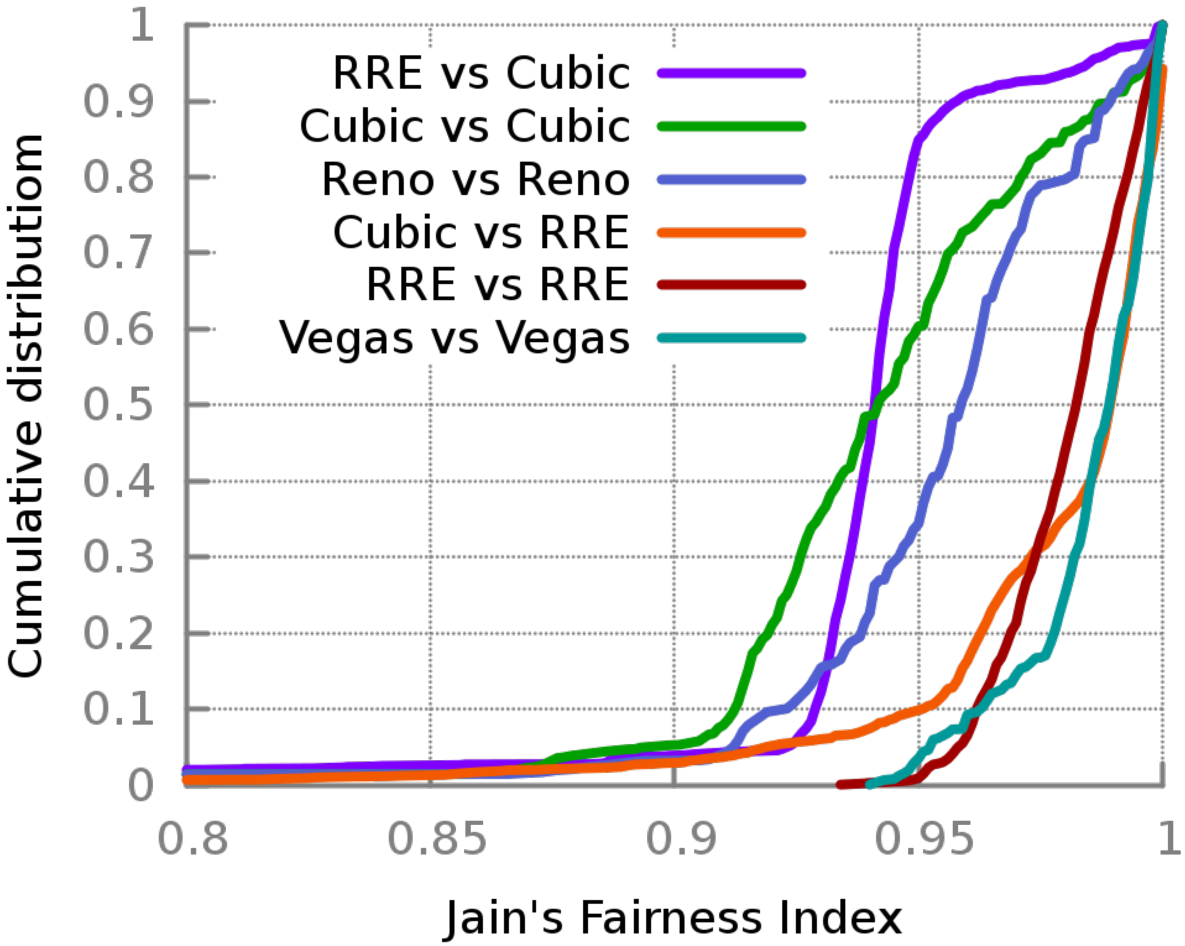
TCP-RRE

TCP Friendliness

- **Run two RSFC uploads concurrently**
- **Calculate Jain fairness index:**

$$(R_1 + R_2)^2 / (2(R_1^2 + R_2^2))$$

TCP Friendliness


Cumulative distributiom
RRE vs Cubic
Cubic vs Cubic
Reno vs Reno
Cubic vs RRE
RRE vs RRE
Vegas vs Vegas
Jain's Fairness Index