TCP SIAD: Congestion Control supporting Low Latency and High Speed

Mirja Kühlewind <mirja.kuehelwind@tik.ee.ethz.ch>

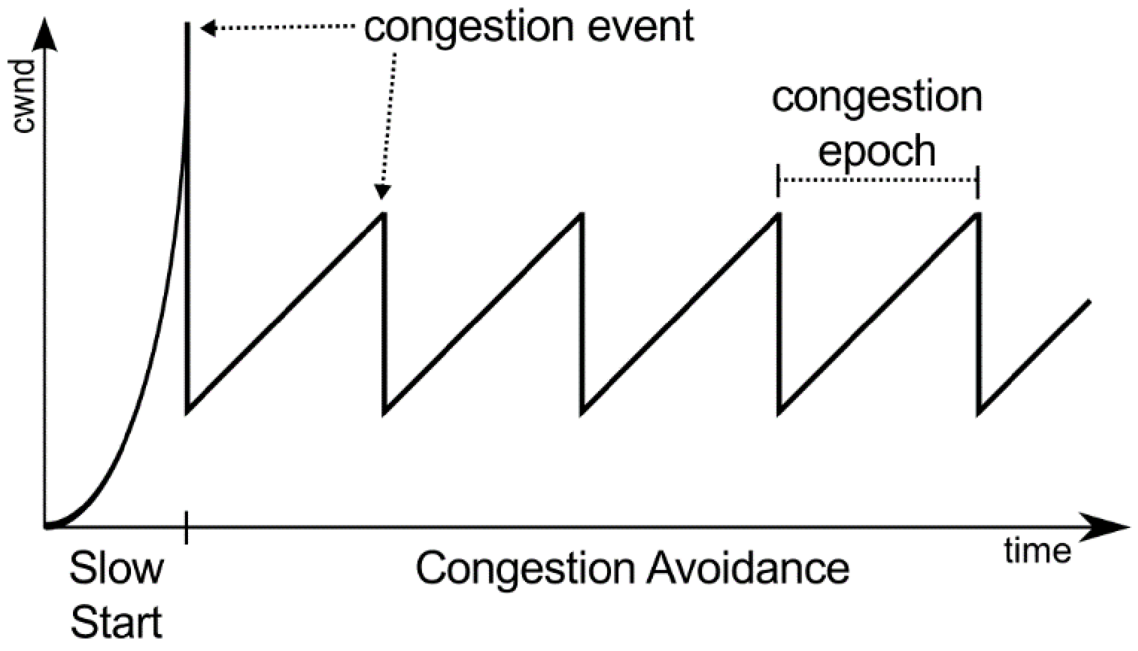
IETF91 Honolulu ICCRG

Nov 11, 2014

Outline

- Current Research Challenges
  - Scalability in large BDP networks
  - Low Latency Support In the Internet
  - Per-User Fairness based on Congestion Policing
- TCP SIAD: Algorithm Design
  - Scalable Increase
  - Adaptive Decrease
- Evaluation
  - Comparison in Single Flow Scenario
  - Capacity Sharing
- Conclusion and Outlook

Scalability



TCP NewReno

- is limited by theoretical limits of the network bit error rate

$$B(p) = \frac{1}{RTT}\sqrt{\frac{3}{2p}}$$

- needs long time to allocate new capacity

  e.g. to raise from 5 to 10 Gbit/s with RTT 100ms and 1500 bytes packets → more than 1h!

→ Most proposed schemes scale much better but still depend on the BDP!

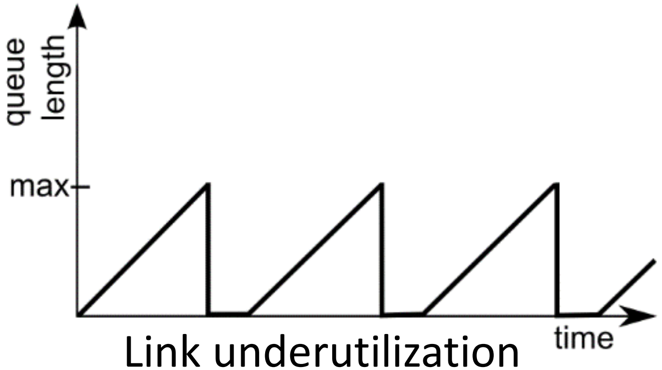
Congestion control should

- provide a fixed feedback rate independent of the link BDP
- allocate quickly newly available bandwidth (under changing network conditions)

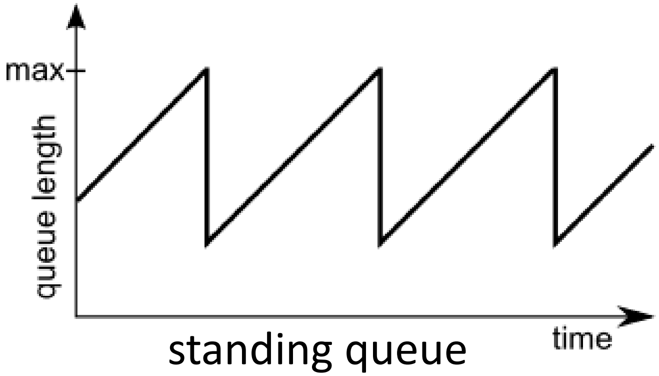
Low Latency Support

Today's Internet is mainly optimized for high through-put and low loss rates

→ Large buffers needed to provide sufficient space for TCP congestion control (worst case: BDP)



Link underutilization



standing queue

→Enable operators to configure small buffers and keep utilization high!

Congestion control (cannot change the buffer configuration but) should

- keep the link utilization high even if small buffers are configured
- avoid a standing queue by emptying the buffers at every decrease

Per-User Congestion Policing

- TCP-friendliness should not be a requirement for congestion control
- Fairness should be enforce on a long-term per-user (not instantaneous per-flow) basis
  - E.g. based on (Ingress) congestion policing using Congestion Exposure (ConEx)
  - It’s okay to grab a larger share of the capacity (for a limited time) if needed

Congestion control should

- provide an configuration knob to influence the amount of congestion

TCP SIAD: Design Goals

High Link Utilization independent of network buffer sizes

Avoid Standing Queue/Minimize Average Delay (however, still loss-based)

Speed-up for Bandwidth Allocation (under changing network conditions)

Fixed Feedback Rate independent of bandwidth (when self-congested)

Configurable Aggressiveness (when competing with other traffic)

can be used by a higher layer control loop to impact the capacity share at the cost of higher congestion, e.g. for applications that need a minimum rate

Additive Increase Multiplicative Decrease

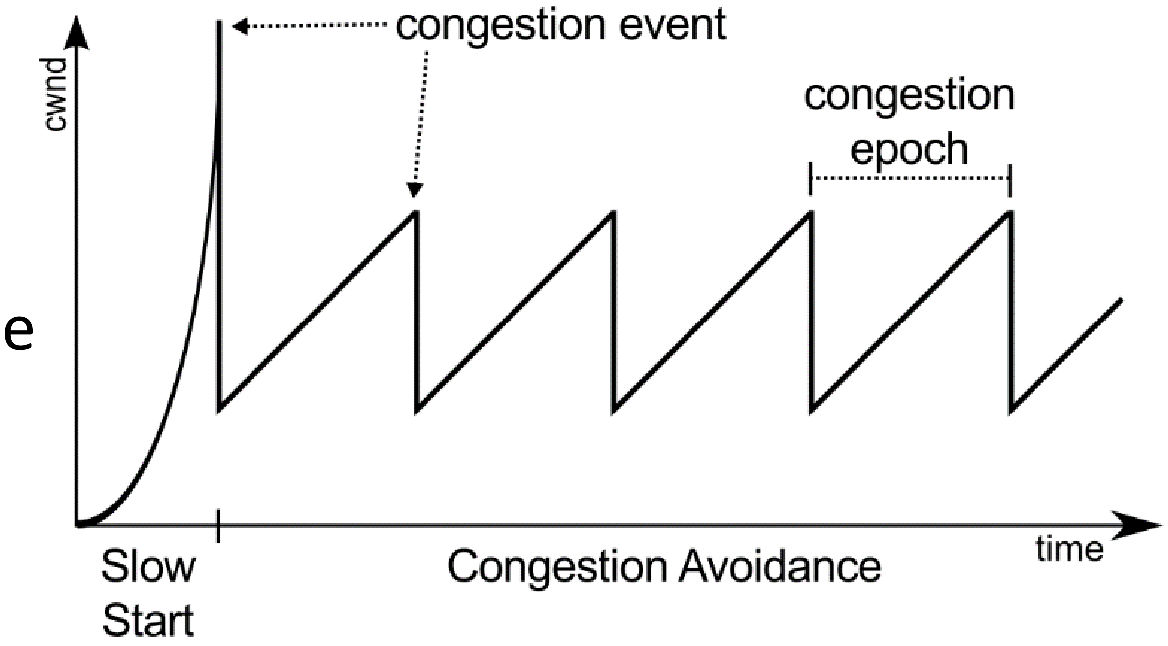
Exponential Increase (Slow Start)

$cwnd = cwnd + 1$ [per ACK]

Additive Increase (Congestion Avoidance

$cwnd = cwnd + \frac{\alpha}{cwnd}$ [per ACK]

for TPC NewReno: $\alpha = 1$



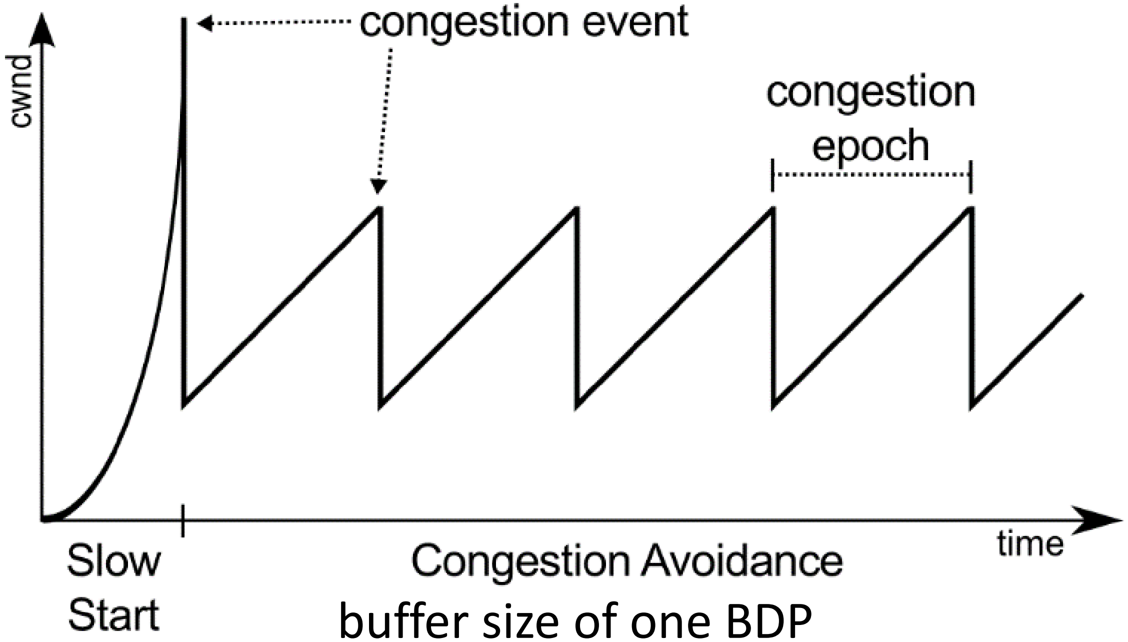
congestion window: $cwnd$ [pkts]

Multiplicative Decrease (Fast Recovery)

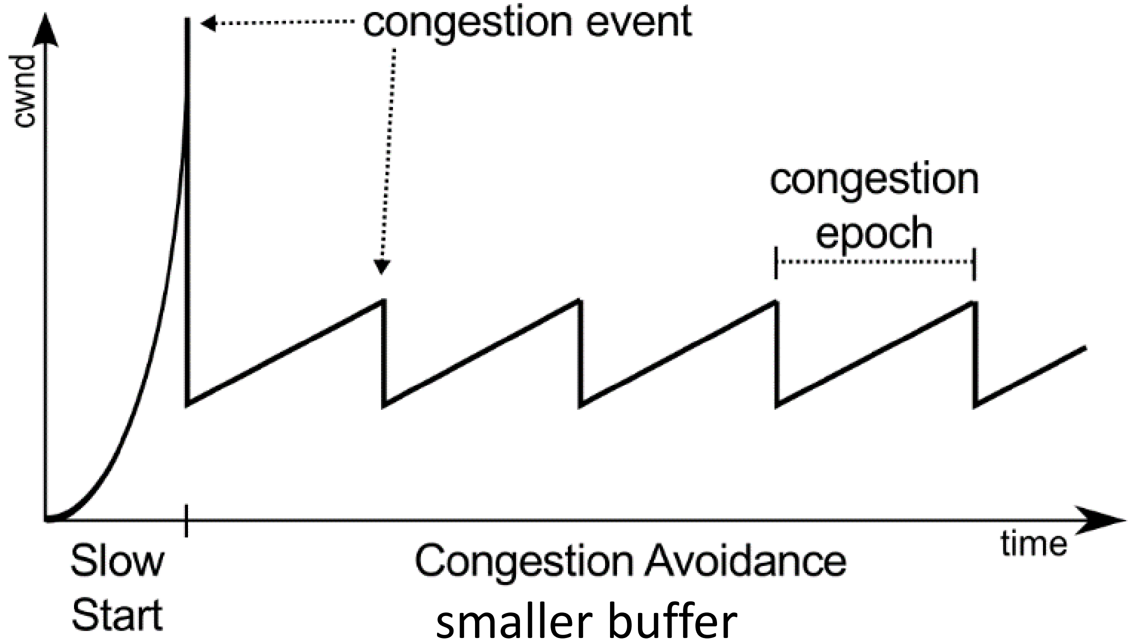
$cwnd = \beta\ cwnd$ [on congestion event]

for TCP NewReno: $\beta = 0.5$

Scalable Increase Adaptive Decrease (SIAD)



buffer size of one BDP



smaller buffer

- Adaptive Decrease adapts decrease factor β to queue size
- Scalable Increase recalculates α to realize fixed feedback rate

Algorithm Design: SIAD

Scalable Increase (SI)

to receive the congestion feedback with a constant rate of Num_{RTT} RTTs

$$\alpha = \frac{incthresh - ssthresh}{Num_{RTT}}, \qquad 1 < \alpha < ssthresh$$

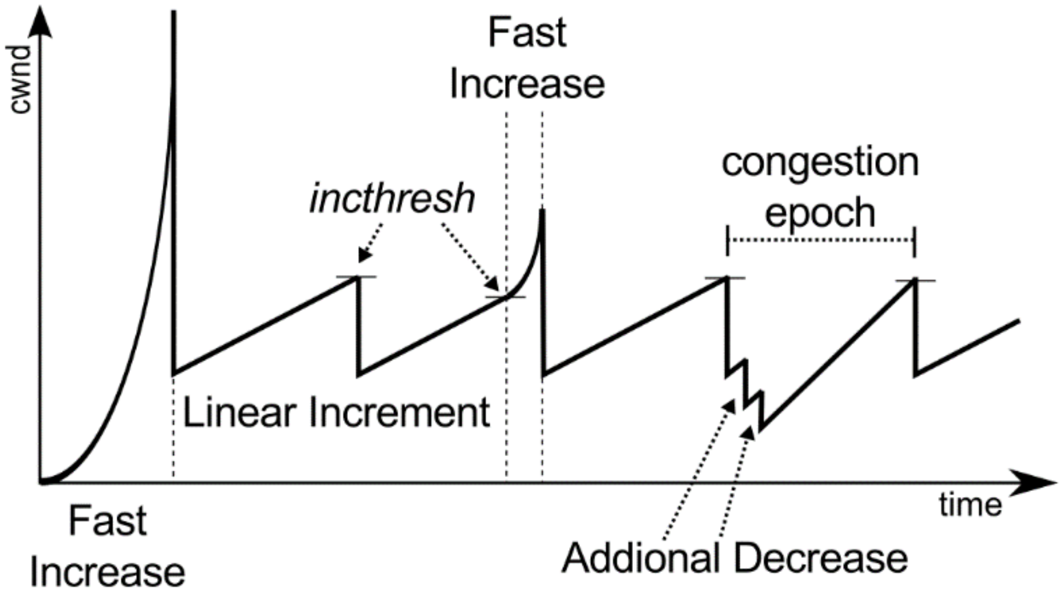
Adaptive Decrease (AD)

to empty queue without causing underutilization or a standing queue

$$cwnd \leftarrow \frac{RTT_{min}}{RTT_{Curr}} cwnd_{max} - 1, \qquad cwnd \geq 2 \quad \text{[on congestion notification]}$$

Algorithm Design: Further Components

- **Additional Decreases** during congestion epoch
  to drain the queue
- **Fast Increase phase** above Linear Increment threshold $incthresh$
  to quickly allocate new bandwidth
- **Trend** calculation of $cwnd_{max}$
  to improve convergence



Scalable Increase (1)

1. Adapt $cwnd$ as congestion event occurred about one RTT ago

$$cwnd_{max} = cwnd - \begin{cases} \frac{cwnd}{2} & \text{if } \alpha < cwnd \cup cwnd \leq ssthresh \\ \frac{cwnd}{3} & \text{if } cwnd \geq insthresh \cap \alpha = \frac{cwnd}{2} \\ \frac{incthresh - ssthresh}{Num_{RTT}} & \text{if } cwnd \geq insthresh \cap \alpha = 1 \\ \frac{\alpha}{2} & \text{if } cwnd > incthresh \\ \alpha & \text{else} \end{cases}$$

2. **Trend** calculation

$$trend = cwnd_{max} - prev\_cwnd_{max}$$

$prev\_cwnd_{max}$: maximum congestion window of previous congestion event

Scalable Increase (2)

3. Linear Increment threshold

$$incthresh = cwnd_{max} + trend, \qquad incthresh \geq ssthresh$$

4. Adaption of $\boldsymbol{\alpha}$

$$\alpha = \frac{incthresh - ssthresh}{Num_{RTT}}, \qquad 1 < \alpha < ssthresh$$

$ssthresh$:	Slow Start threshold (= congestion window after reduction)
$incthresh$:	Linear Increment threshold (see previous slide)
Num_{RTT}:	configuration parameter for number of RTT between two congestion events

Scalable Increase (3)

Linear Increment phase

$$cwnd = cwnd + \frac{\alpha}{cwnd} \quad \text{[per ACK]}$$

Fast Increase phase (if $cwnd \geq incthresh$)

1. Reset increase factor α to 1
2. Double increase factor α per RTT

$$\alpha = \alpha + \frac{\alpha}{cwnd}, \qquad \alpha \geq \frac{cwnd}{2} \quad \text{[per ACK]}$$

Adaptive Decrease (1)

Backlogged packets in queue (see Vegas, Compound, H-TCP...)

$$q = \frac{RTT(t) - RTT_{min}}{RTT(t)} \, cwnd$$

Adaptive Decrease

$$\beta = \frac{RTT_{min}}{RTT_{curr}}$$

$$cwnd \leftarrow \beta \, cwnd_{max} - 1, \quad cwnd \geq 2 \qquad \text{[on congestion]}$$

→ only drains queue if all competing flow are synchronized

Adaptive Decrease (2)

Additional Decrease (if minimum RTT cannot be observed after one RTT)

1.
$$cwnd = \frac{RTT_{min}}{RTT_{curr}}\ ssthresh\ - 1$$

2.
$$cwnd\ \leftarrow cwnd\ - \max(red, \alpha_{new})$$

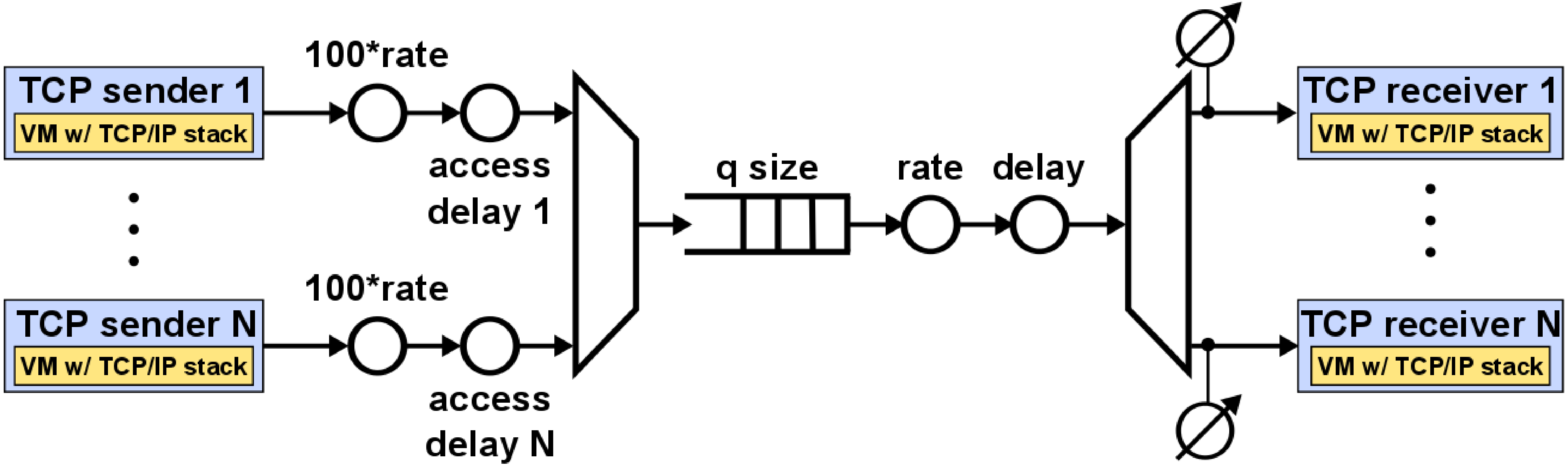
$$red = cwnd\ \frac{1}{Num_{RTT} - dec\_cnt}$$

$$\alpha_{new} = \frac{incthresh\ - cwnd}{Num_{RTT}\ -\ dec_{cnt} -\ 1}$$

dec_cnt: number of additional decrease that have already been performed

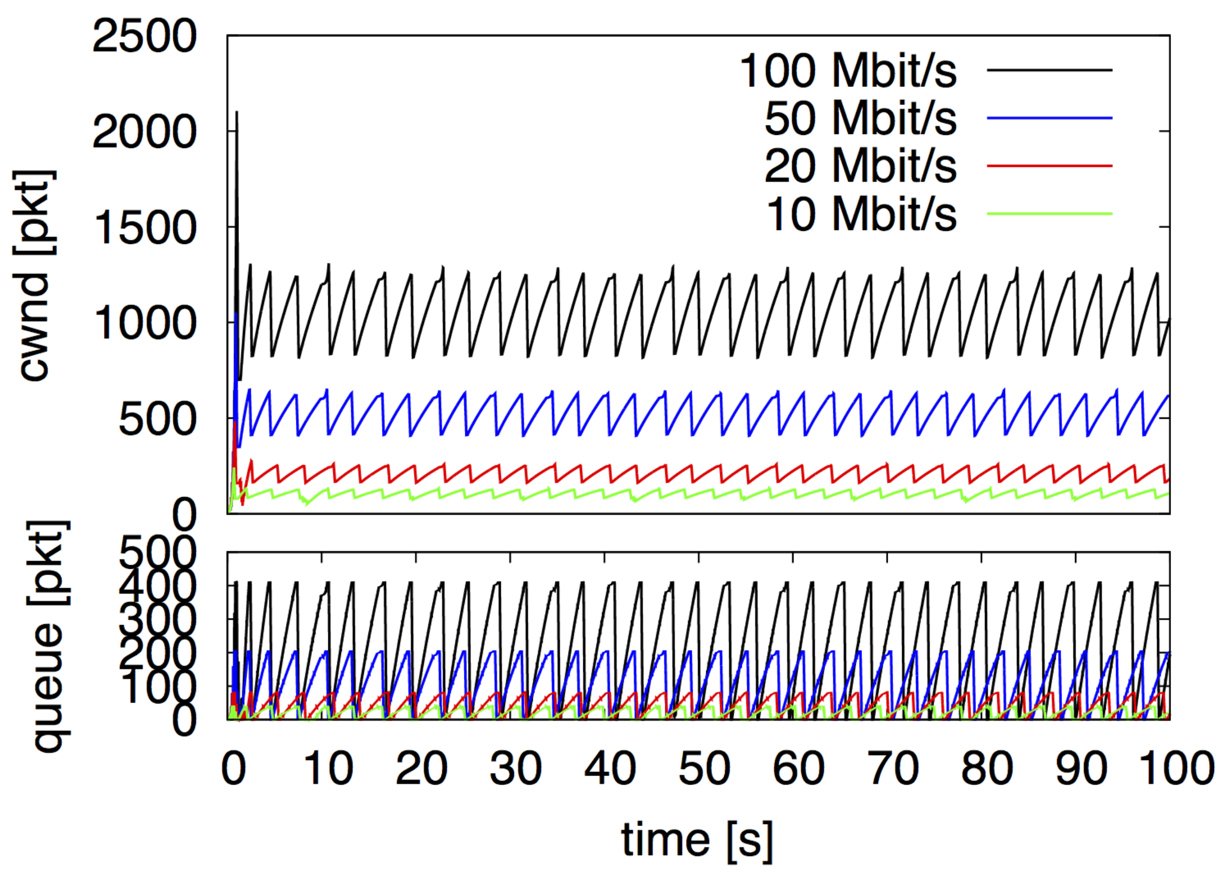
Simulation Setup

- Event-driven network simulator IKR SimLib with integration for virtual machines IKR VMSimInt (http://www.ikr.uni-stuttgart.de/IKRSimLib/Download/)
- Implementation in Linux kernel 3.5.7 (default $Num_{RTT} = 20$)

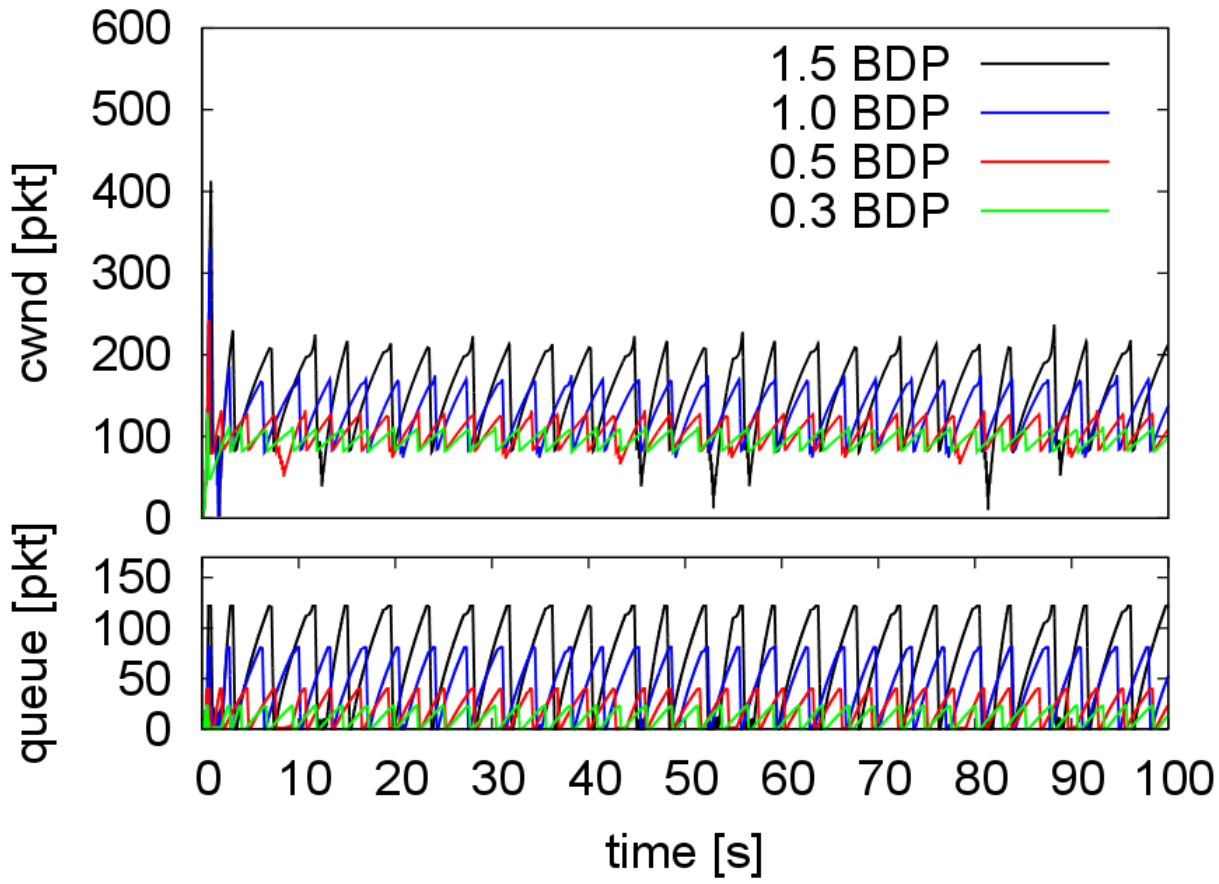


TCP SIAD's Congestion Window Development

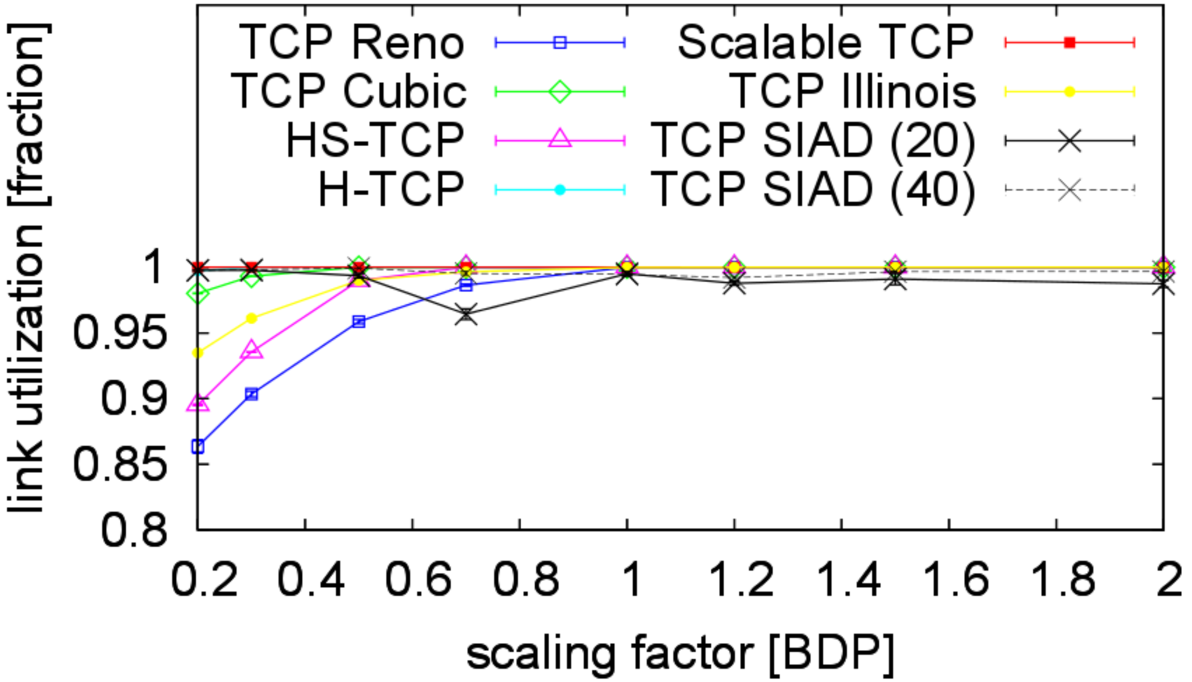
0.5 BDP buffering

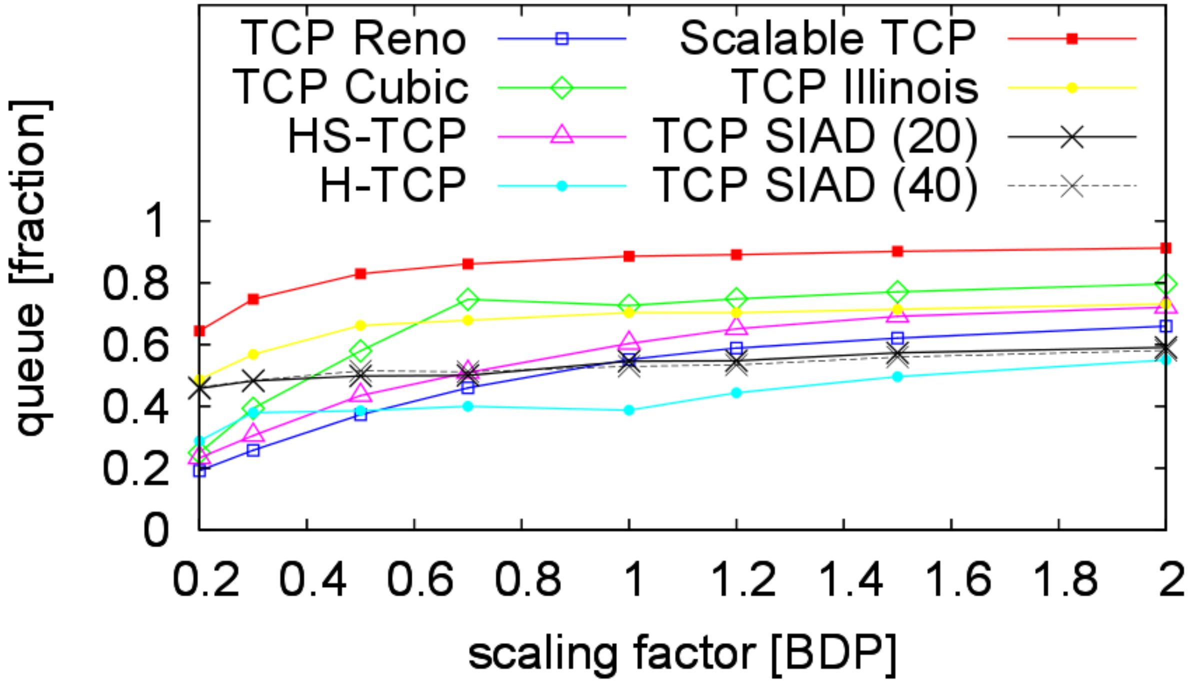


10 Mbit/s



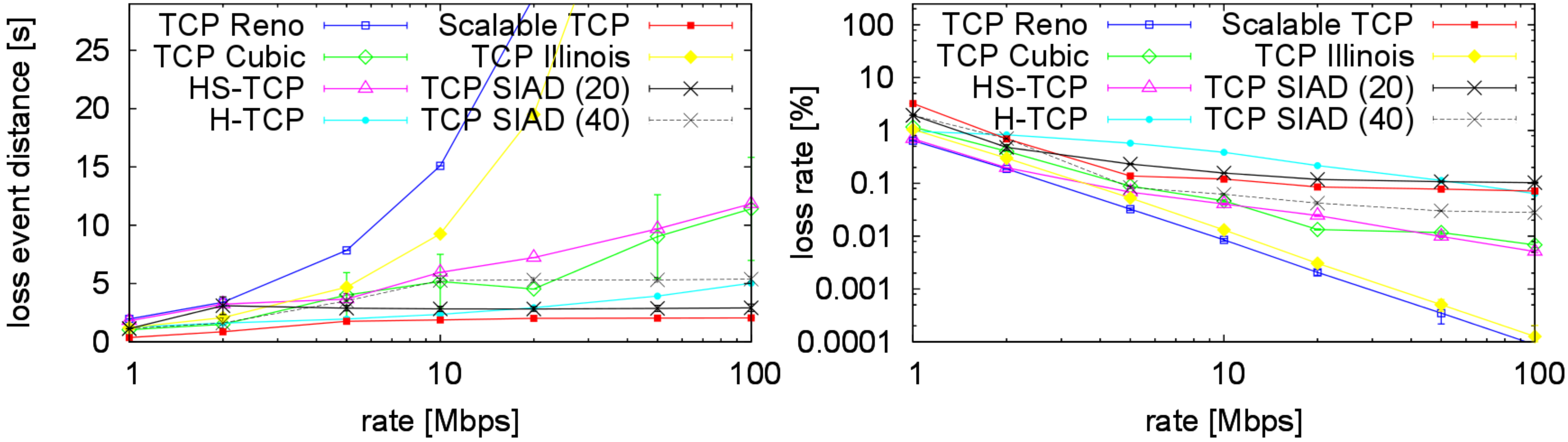
Single Flow on 10 Mbit/s link





→TCP SIAD always fully utilizes link + has average queue fill of 0.5

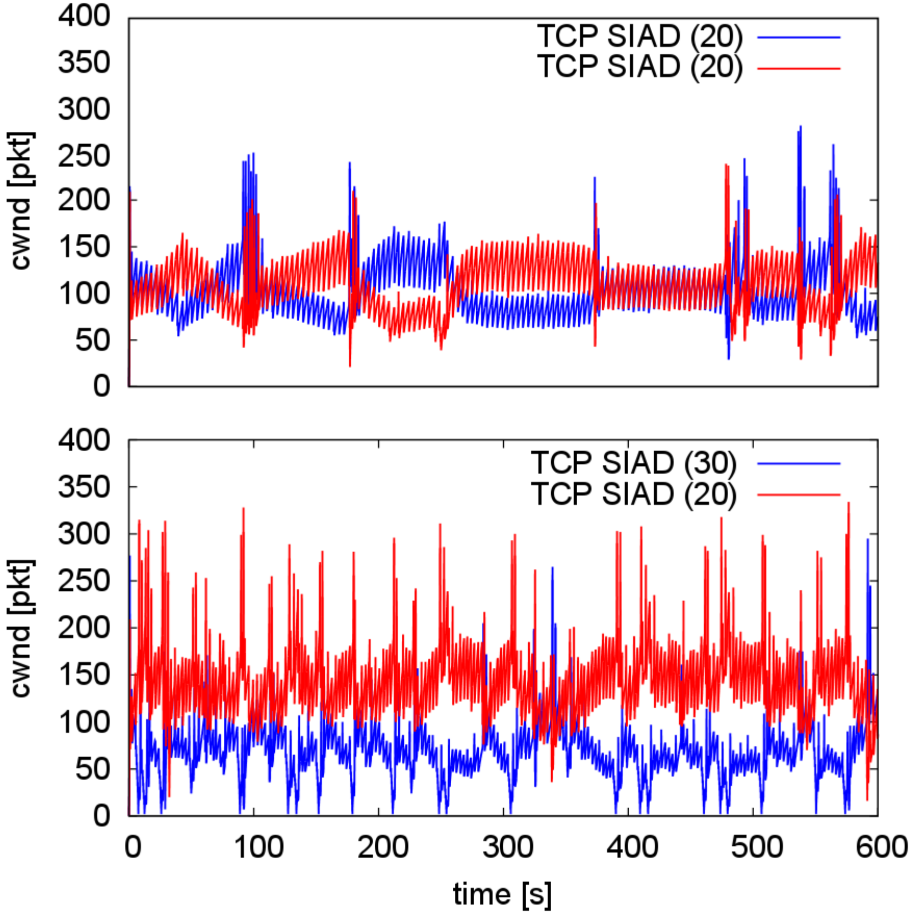
Single Flow with 0.5 BDP buffering



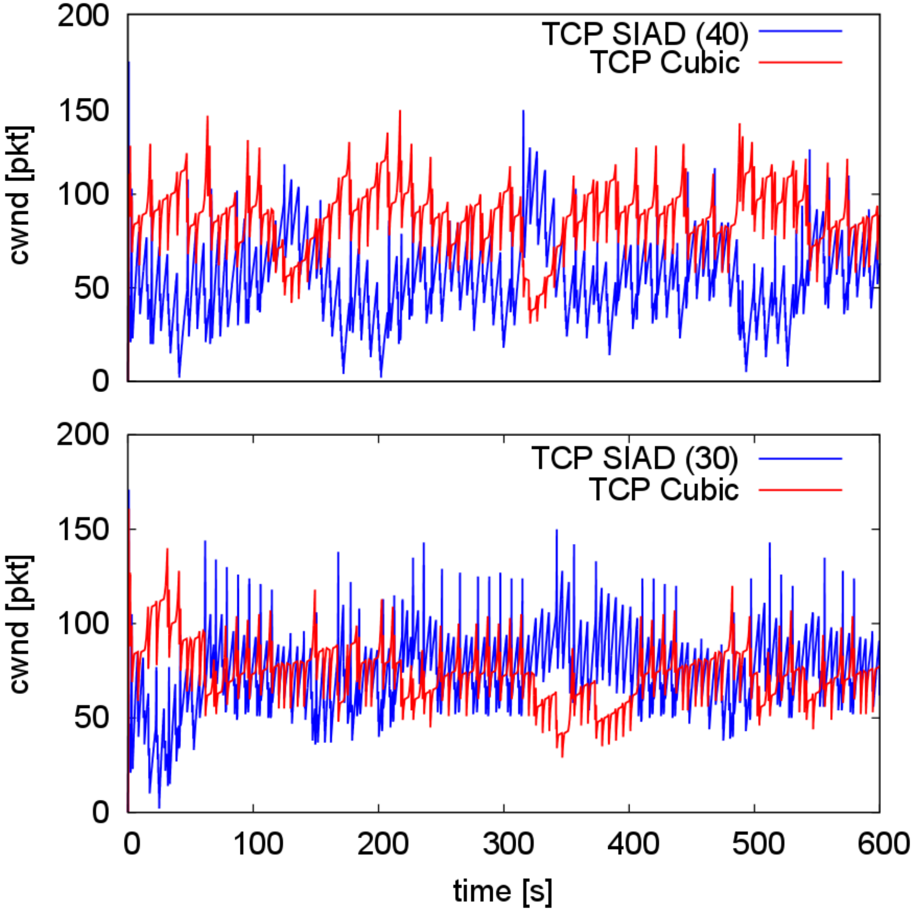
→TCP SIAD induces fixed feedback rate independent of link bandwidth

TCP SIAD‘s Capacity Sharing

20 Mbit/s, 0.5 BDP buffer



10 Mbit/s, 1 BDP buffer



Conclusion and Outlook

TCP SIAD provides

- high link utilization with small buffer and standing queue avoidance
- configurable fixed feedback rate

→ Allows network operators to configure small buffers (or low marking thresholds) and maintain high utilization!

Next

- TCP SIAD and ECN
- SimpleSIAD
- Higher layer control loop for e.g. real-time video