TCP Westwood(+) Protocol Implementation in ns-3

Authors: Siddharth Fangadhar, Truc Anh N. Nguyenm Greeshma Umapathi, and James P.G. Sterbenz

CS577

Brett Levasseur

Outline

- Introduction
- TCP Variations
- ns-3 TCP Implementation
- ns-3 Westwood Implementation
- Evaluation
- Conclusions
- Remarks
- Questions

Introduction

WPI

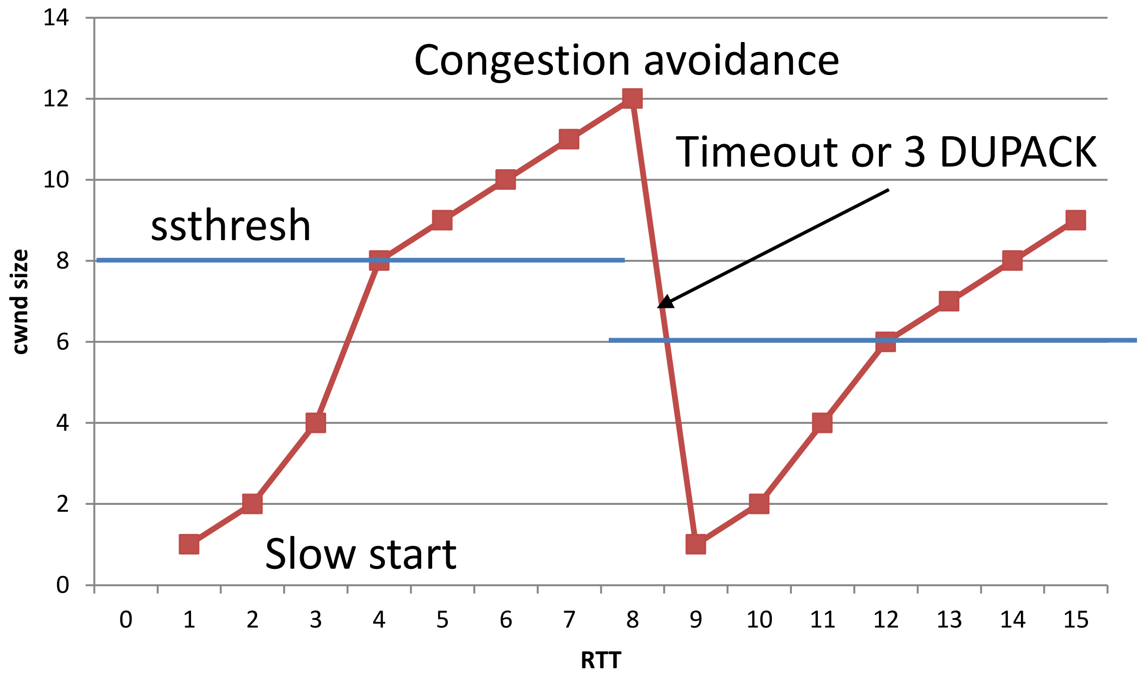
- ns-3 is a packet network simulator
  - Successor to ns-2
  - Improved design, better wireless support
  - Used by researchers around the world
  - Has TCP implementation
  - Lacks modern TCP variants
  - Tahoe, Reno, NewReno
- Authors present Westwood(+) for ns-3

TCP Tahoe

- Terms
  - cwnd: Congestion Window
  - ssthresh: Slow Start Threshold
- TCP States
  - Slow-start: cwnd exponential increase
  - Congestion Avoidance: cwnd linear increase
  - Fast Retransmit: Half ssthresh, reset cwnd to 1
- Timeouts and duplicate ACKs (DUPACK) considered congestion

TCP Tahoe

WPI



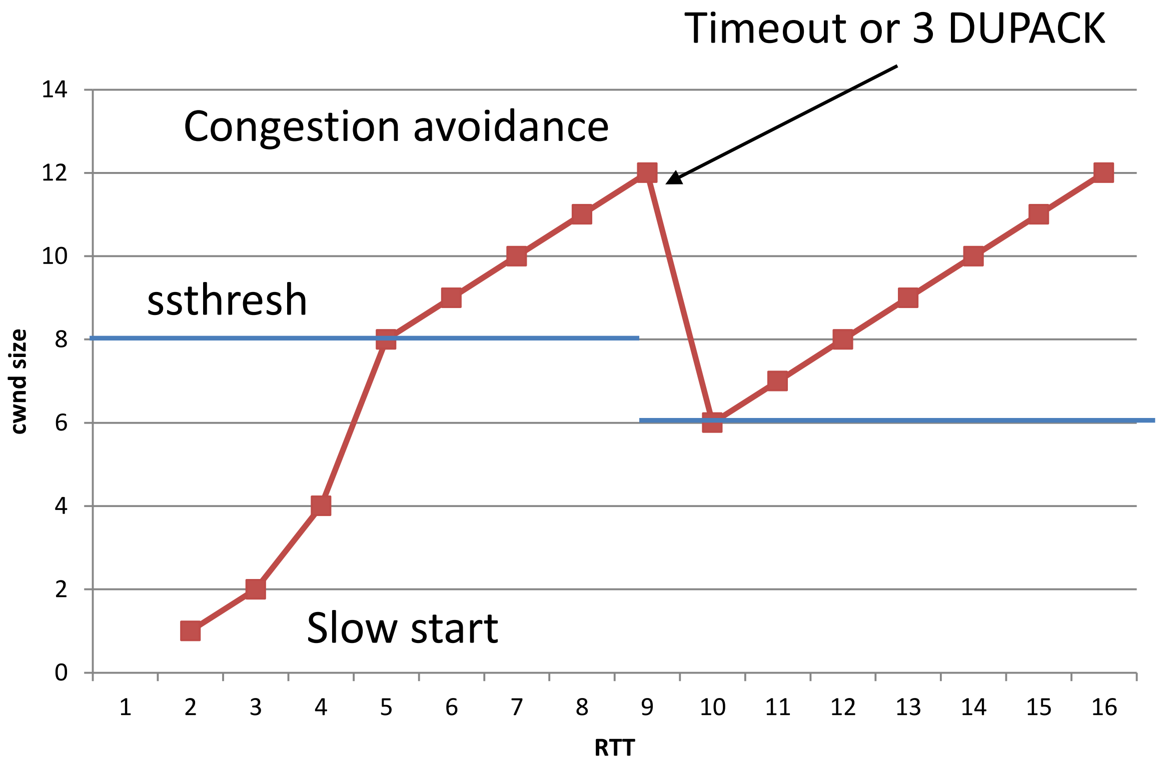
TCP Reno

WPI

- Tahoe vs Reno
  - Tahoe: 3 DUPACKs move to fast retransmit
  - Reno: 3 DUPACKs half ssthresh and cwnd, move to fast recovery
- Fast Recovery
  - Retransmit missing packet
  - Wait for ACK before congestion avoidance

TCP Reno

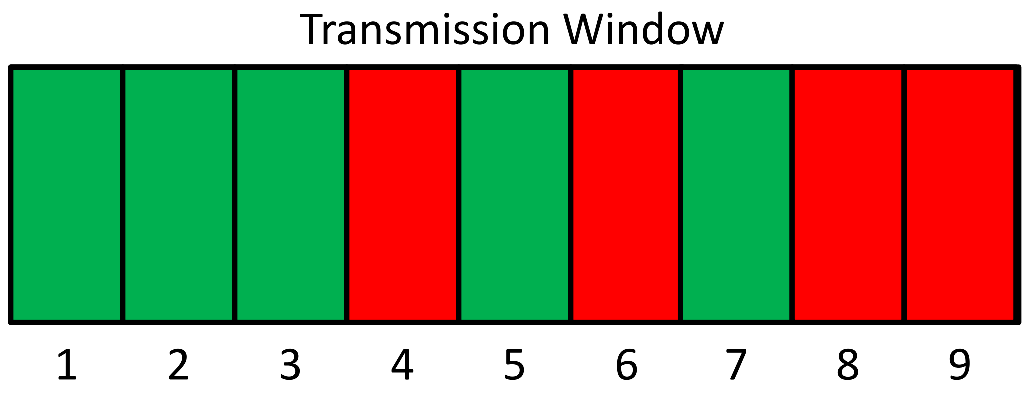
WPI


Timeout or 3 DUPACK
Congestion avoidance
ssthresh
Slow start
cwnd size
RTT
0
2
4
6
8
10
12
14
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

TCP NewReno

WPI

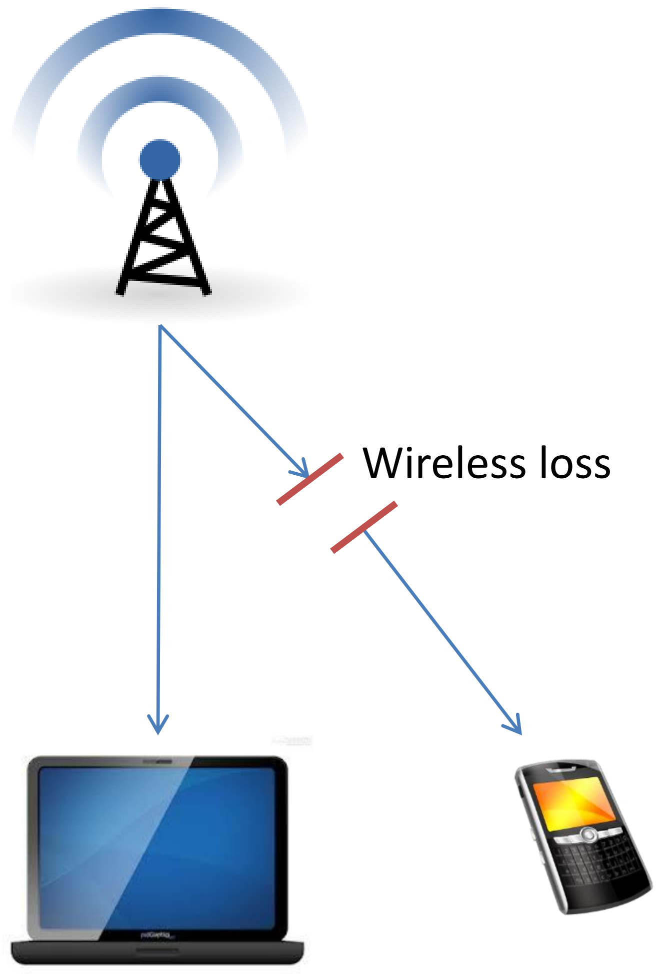
- Adds partial and full ACKs
  - Partial ACK remain in fast recovery
  - Full ACK continues congestion avoidance



Partial ACK = 4, 6, 8
Full ACK = 9

TCP Packet Corruption

WPI



- Lost packets considered congestion
- Wireless has bursty errors
- High wireless bit error rate confused as congestion
- TCP lowers cwnd needlessly

TCP Westwood

WPI

- Made for wireless networks
- Estimates bandwidth
  - Set cwnd based on estimate
  - Set ssthresh based on estimate
  - Rate of ACK and DUPACK arrivals used

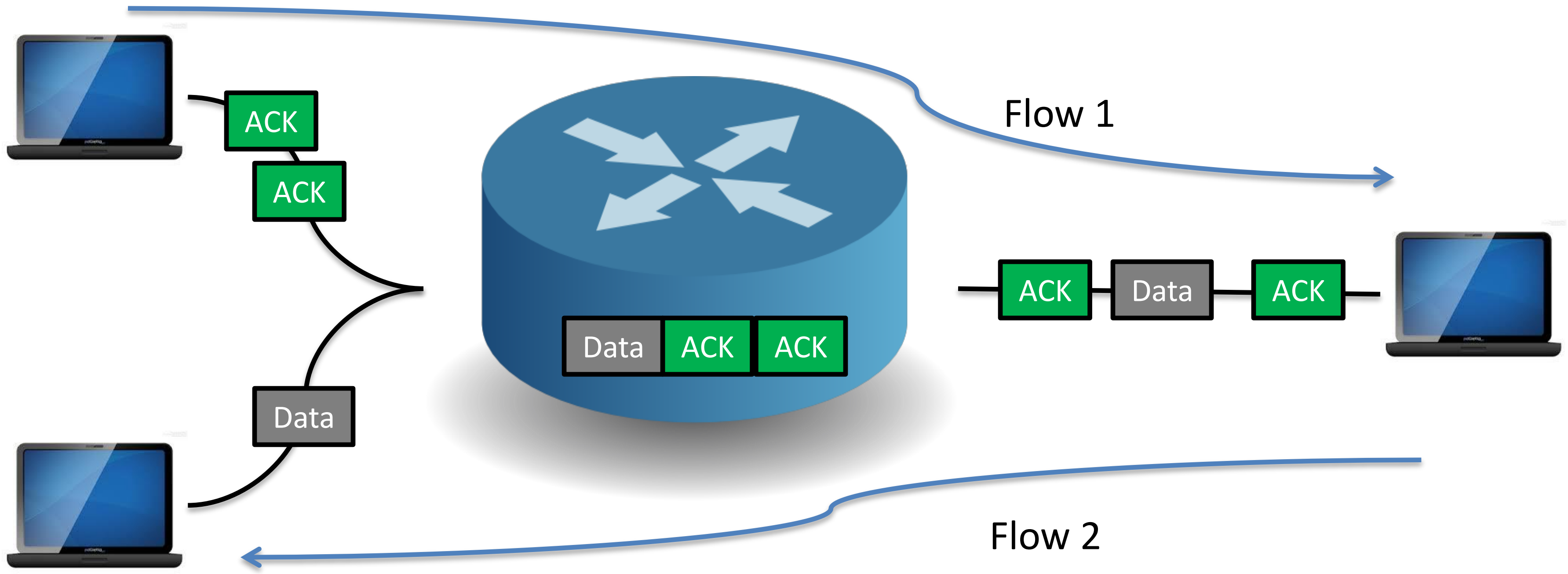
BWE	ˆ*b*
BWA	*b*
Weight	*a = 0.9*
Time	*k*

$$\hat{b}_k = a\hat{b}_{k-1} + \frac{1-a}{2}[b_k + b_{k-1}]$$

TCP Westwood+

WPI

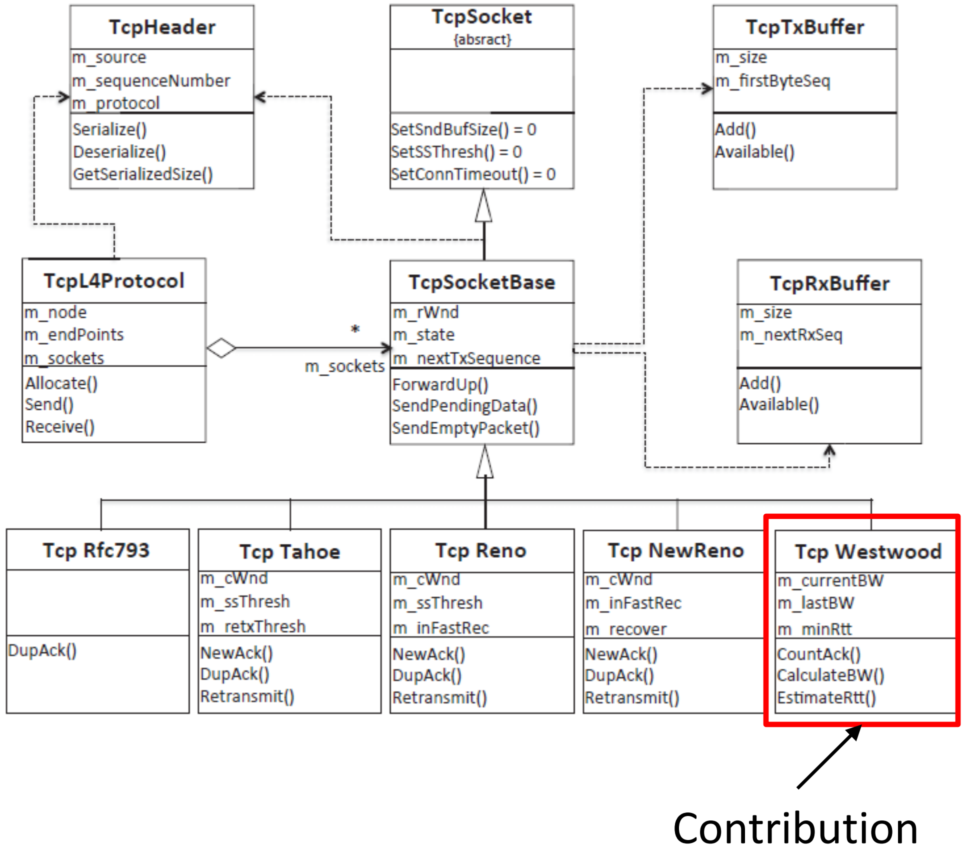
- ACK compression hurts Westwood estimation
- Westwood+ compensates
  - Samples every RTT instead of every ACK



TCP in ns-3

WPI

- Object oriented design
- Generic TCP defined
- TCP variants are extended from base
- TCP headers and buffers provided



Global Variables

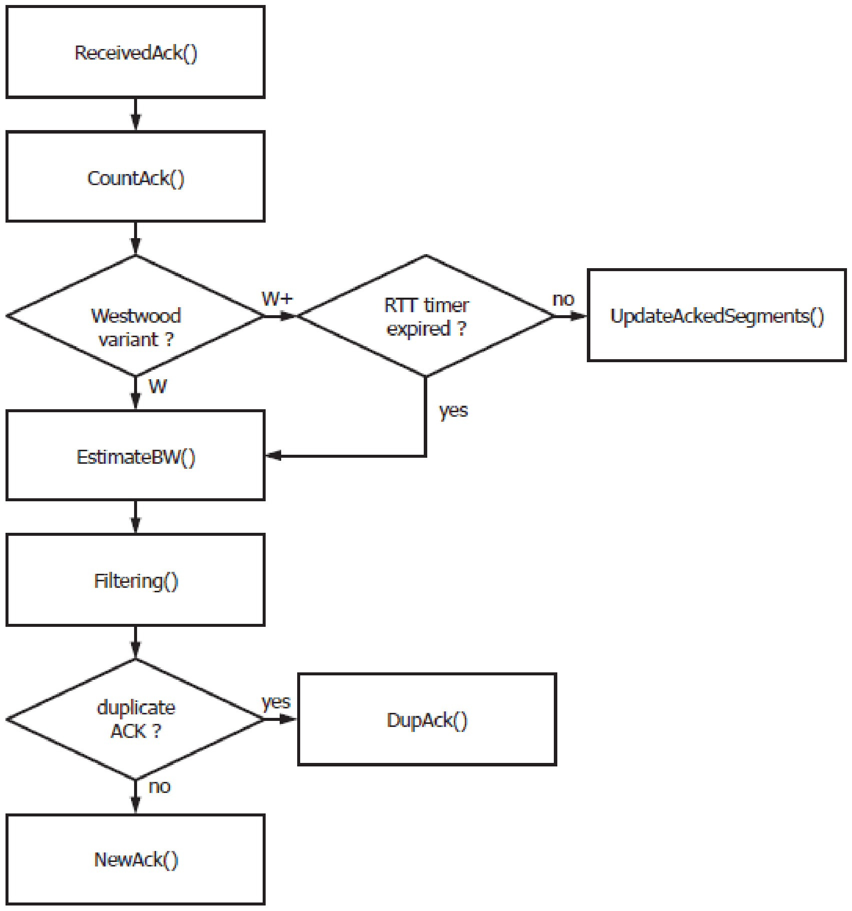
WPI

m_cWnd	Congestion window
m_ssThresh	Slow start threshold
m_initialCWnd	Initial value of m_cWnd
m_inFastRec	Fast recovery flag
m_prevAckNo	Last received ACK
m_accountedFor	Track number of DUPACKs during loss
m_lastAck	Arrival time of previous ACK
m_currentBW	Current bandwidth estimate
m_minRTT	Minimum round trip time
m_lastBW	Last bandwidth estimate
m_lastSampleBW	Total measured bandwidth
m_ackedSegments	Total ACKed segments for current RTT
m_IsCount	Flag to count for m_ackedSegments
m_bwEstimateEvent	Bandwidth sampling event

Execution

WPI

- ACK arrives at sender
- ACKs counted
- Bandwidth is estimated
  - Immediate in Westwood
  - After RTT timeout in +
- Optional use of Tustin filter (user choice)
  - Off: Measured BW
  - On: Estimate BW



Count ACK

- Need total number of bytes sent
  - Count TCP segments received
  - cumul_ack = Current ACK number – m_prevAckNo

ACK = 10

m_prevAckNo = 6
cumul_ack = 10 - 6
4 Packets received

Count ACK

- Take DUPACKs into account
  - If cumul_ack = 0 then current ACK is a duplicate

ACK = 6

m_prevAckNo = 6
cumul_ack = 6 - 6
DUPACK

  - Else check m_accountedFor for number of DUPACKs

ACK = 9

m_prevAckNo = 6
cumul_ack = 9 - 6
m_accountedFor = 1
ACKed 3 > 1 DUPACK
3 – 1 = 2 received

ACK = 7

m_prevAckNo = 6
cumul_ack = 7 - 6
m_accountedFor = 2
ACKed 1 < 2 DUPACK
cumul_ack = 1

Estimate Bandwidth

- Westwood

Bytes since last ACK

$$\text{m\_currentBW} = \frac{\text{cumul\_ack} \times \text{m\_segmentSize}}{\textbf{Simulator::Now}() - \text{m\_lastAck}}$$

Time since last ACK

- Westwood+

$$\text{m\_currentBW} = \frac{\text{m\_ackedSegments} \times \text{m\_segmentSize}}{\text{m\_lastRtt}}$$

Last known RTT

Tustin Filtering

WPI

- Off – Measure bandwidth assumed current
- On – Estimate current bandwidth

$$\text{sample\_bwe} = \text{w1} \times \text{w2}$$

$$\text{w1} = \text{m\_currentBW} \times \alpha$$

$$\text{w2} = \frac{1-\alpha}{2} \times (\text{sample\_bwe} + \text{m\_lastSampleBW})$$

$$\hat{b}_k = \underset{\text{w1}}{\underset{\uparrow}{a\hat{b}_{k-1}}} + \underbrace{\frac{1-a}{2}[b_k + b_{k-1}]}_{\text{w2}}$$

Tustin Filtering

$$\text{sample\_bwe} = \text{w1} \times \text{w2}$$

?

$$\hat{b}_k = \underbrace{a\hat{b}_{k-1}}_{\text{w1}} + \underbrace{\frac{1-a}{2}[b_k + b_{k-1}]}_{\text{w2}}$$

From ns-3 source code (version 3.18):

```
(alpha * m_lastBW) + ((1 - alpha) * ((sample_bwe + m_lastSampleBW) / 2));
```

w1

w2

- Source code and Westwood equation use addition
- Equation 4 uses multiplication so probably a typo

Westwood Cont

WPI

- For new ACK adjust variables same as Reno
- After receiving set number of DUPACKs
  - Adjust slow start threshold

$$\text{m\_ssThresh} = \text{m\_currentBW} \times \text{m\_minRtt}$$

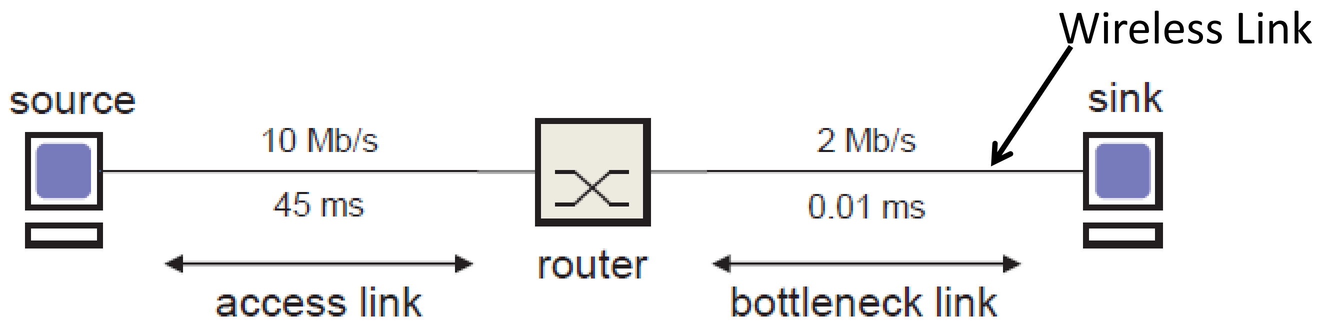
If m_cWnd > m_ssThresh Then m_cWnd = m_ssThresh

- If retransmit timeout
  - Adjust slow start threshold the same as previous
  - Cwnd set to one TCP segment size

Westwood Evaluation

WPI

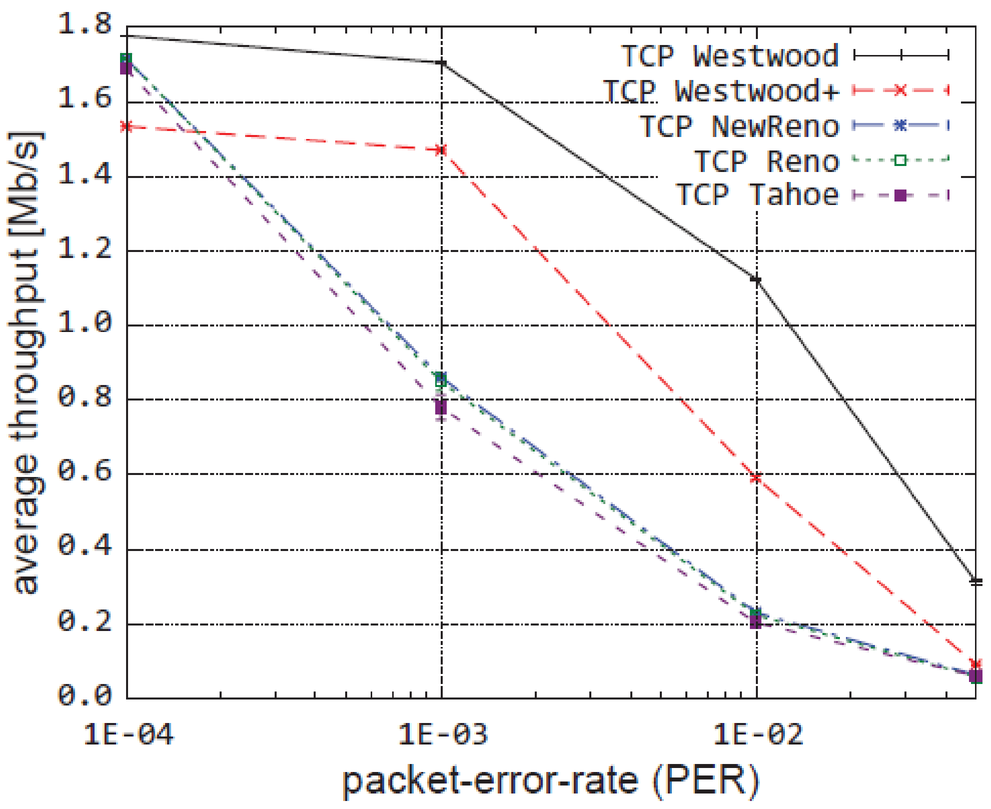
- Simulate original TCP Westwood study



Parameter	Values
Access link bandwidth	10 Mb/s
Bottleneck link bandwidth	2 Mb/s
Access link propagation delay	45 ms
Bottleneck link propagation delay	0.01 ms
Packet MTU size	400 B
Delayed ACK count	2 segments
Delayed ACK timeout	200 ms
Error model	Uniform error model
Error rate	0.005
Application type	Bulk send application
Simulation time	600 s

Packet Error Rate

WPI

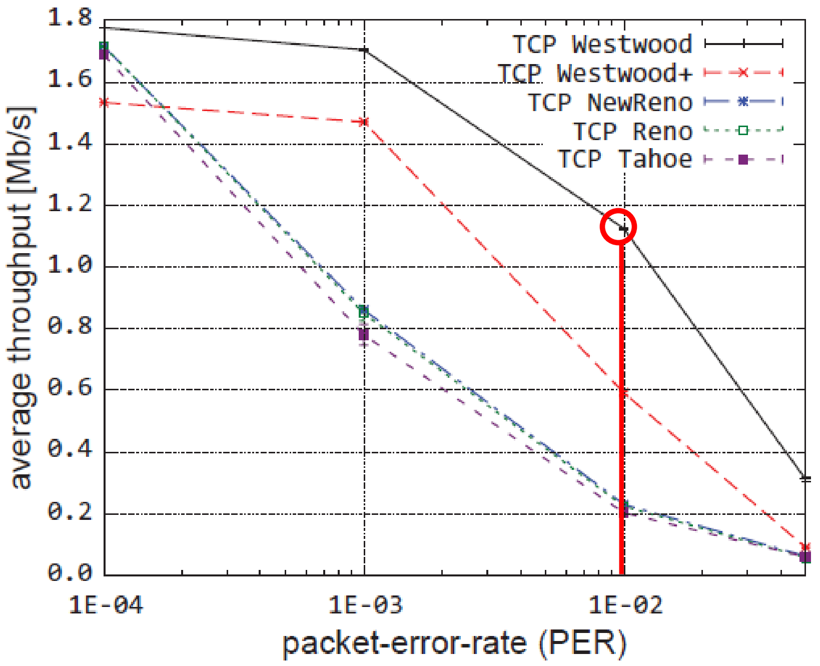


- Westwood samples bandwidth on every ACK
- Westwood+ samples every RTT
- Westwood+ takes longer to stabilize
- As error rate increases Westwood+ performs worse

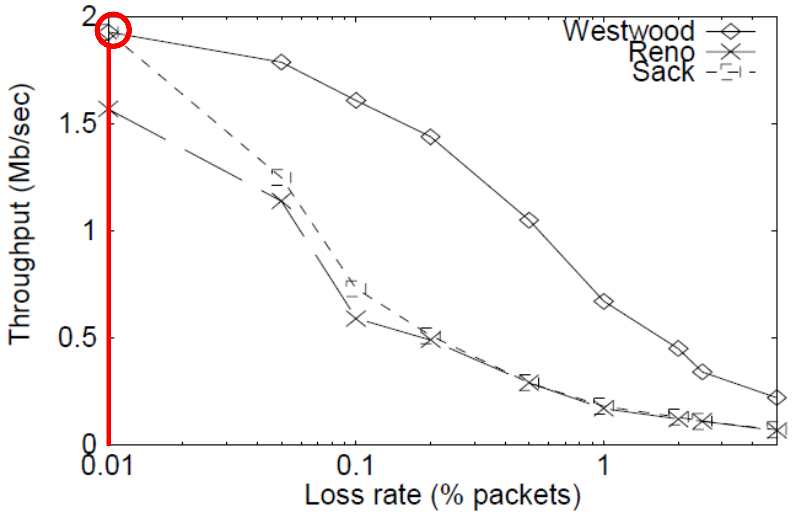
Packet Error Rate

WPI

ns-3 Simulation



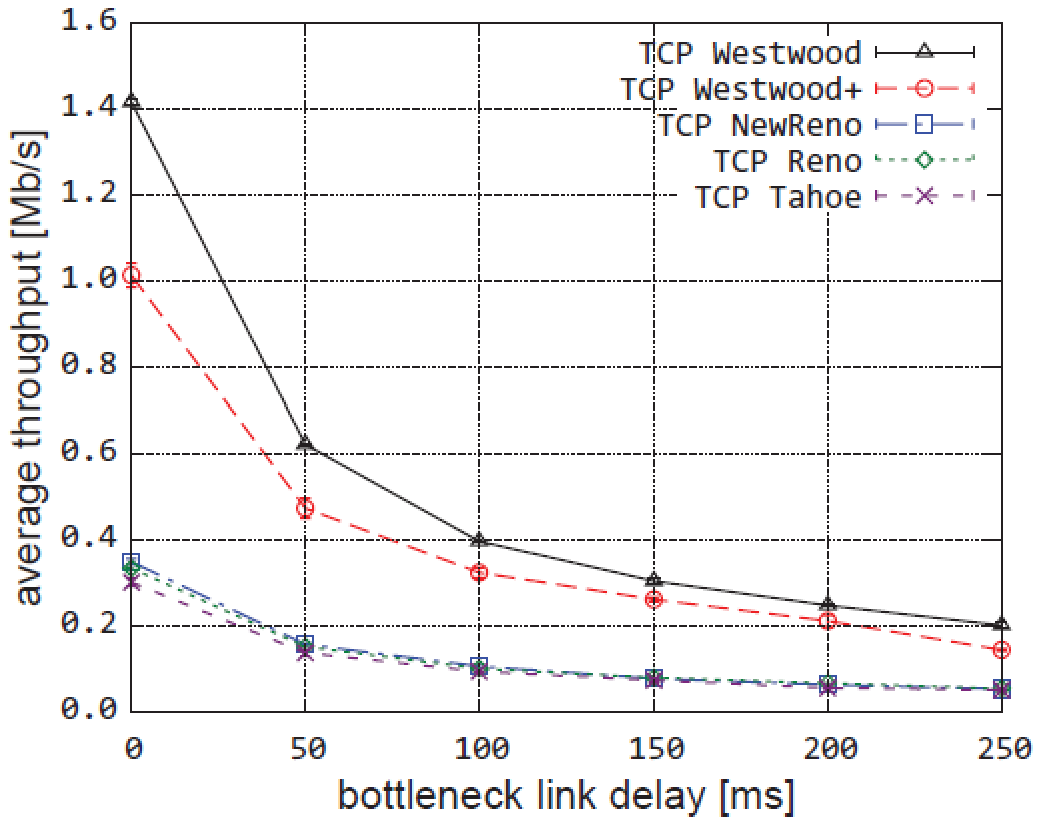
Westwood Paper



Authors claim this is validation of their work

Bottleneck Delay

WPI

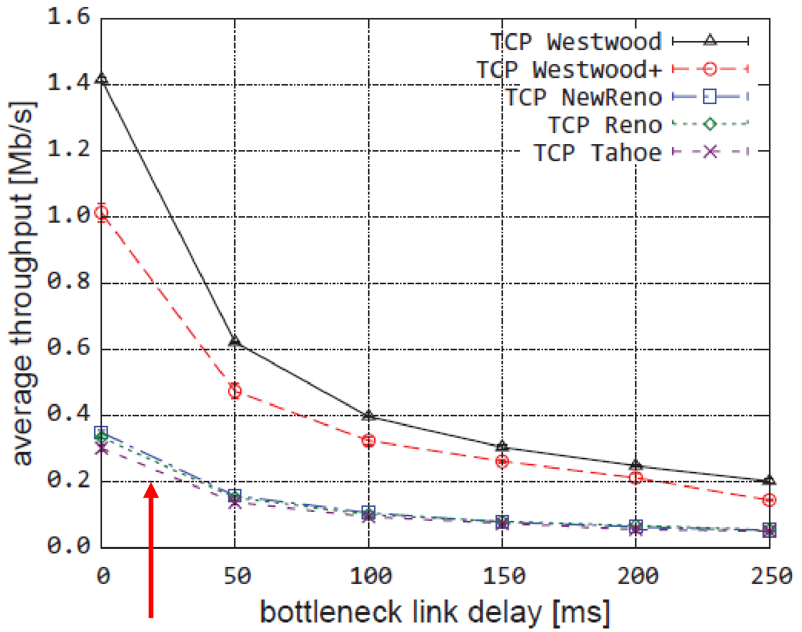


- PER = 0.005
- Westwood(+) attempt to fill the pipe
- Other variants conservative

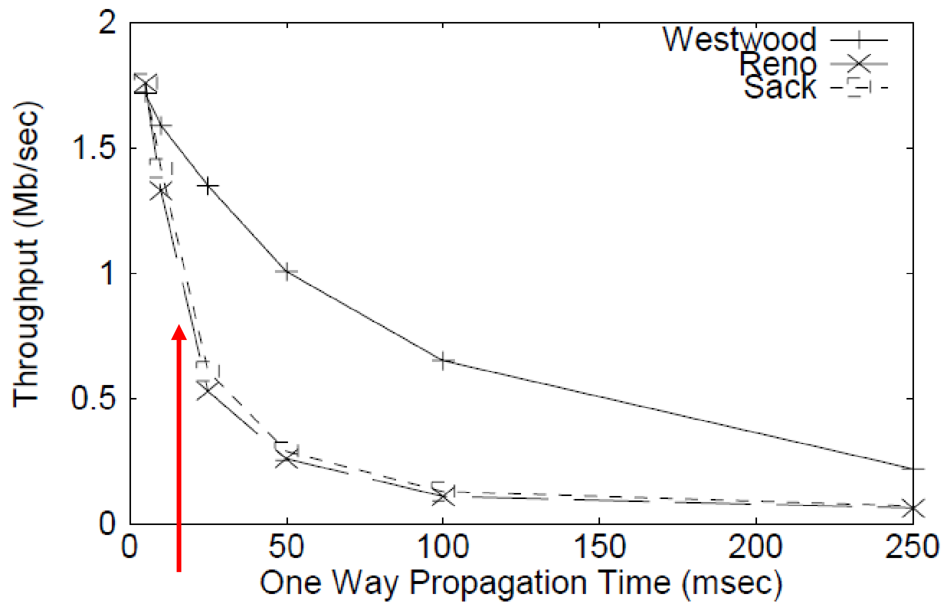
Bottleneck Delay

WPI

ns-3 Simulation

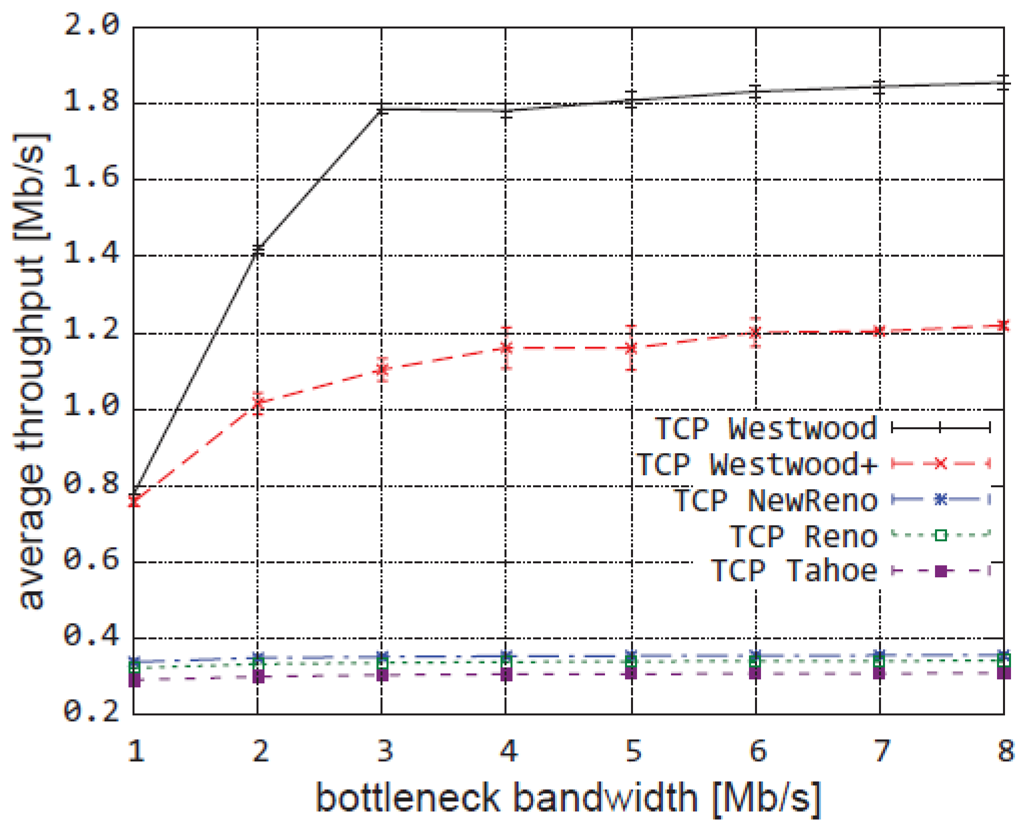


Westwood Paper



TCP Reno appears to behave differently in ns-3 vs ns-2

Bottleneck Bandwidth

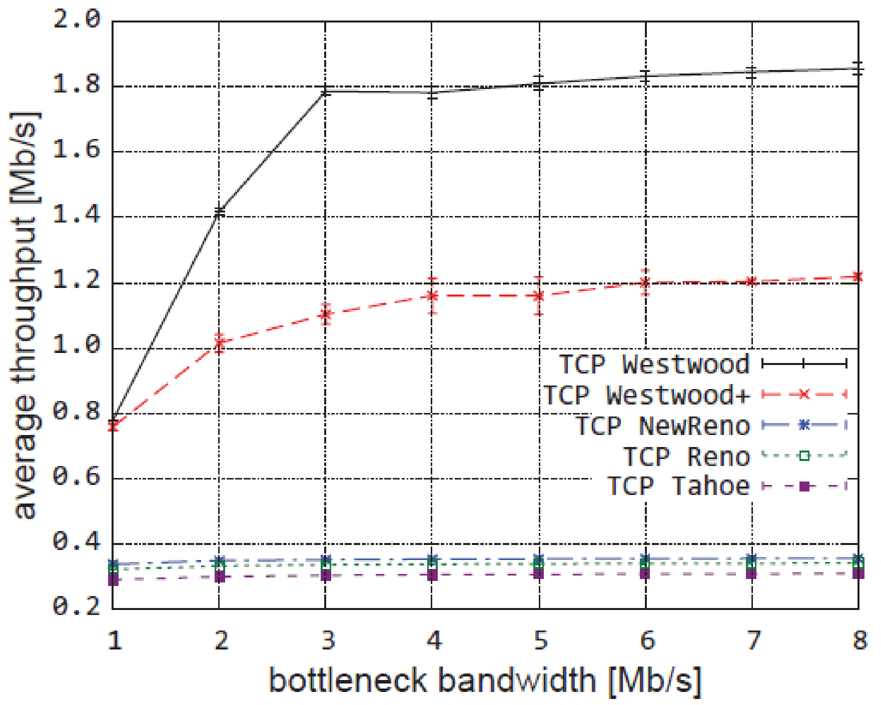


- PER = 0.005
- Delay = 0.01ms
- Westwood(+) attempt to fill the pipe
- Other variants conservative

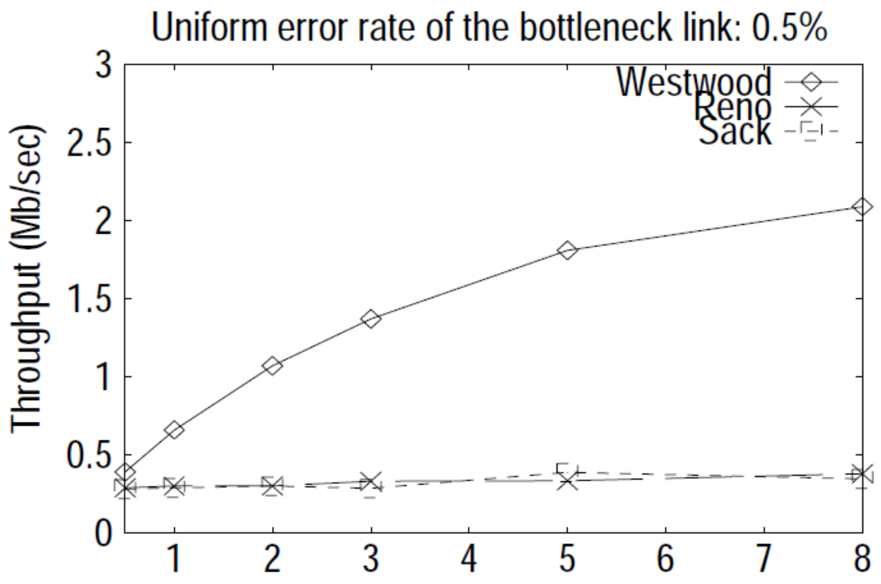
Bottleneck Bandwidth

WPI

ns-3 Simulation

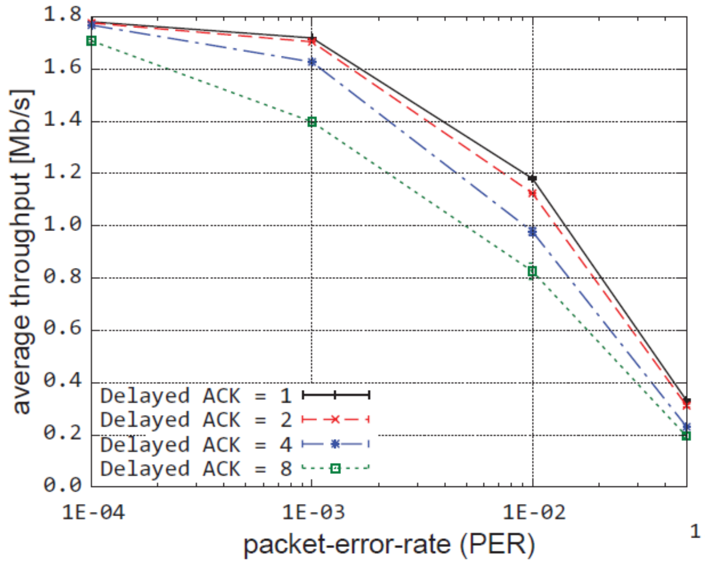


Westwood Paper

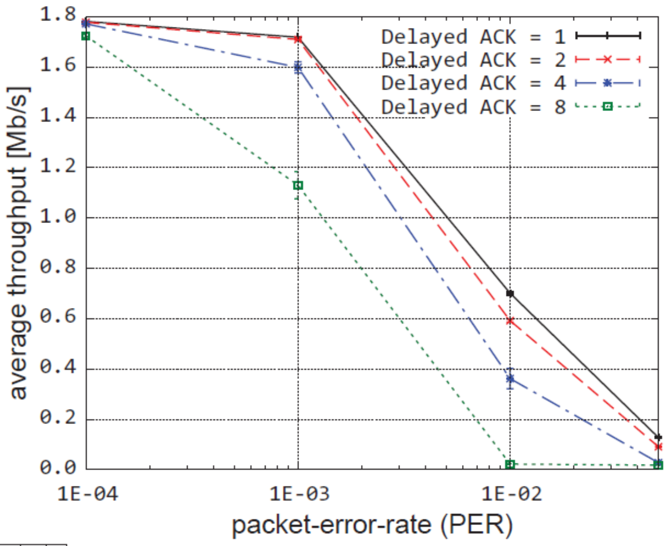


Delayed ACK

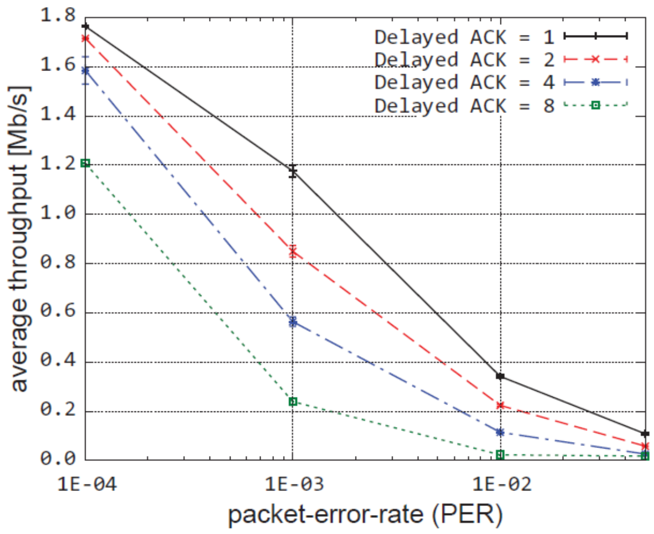
WPI



(a) Westwood



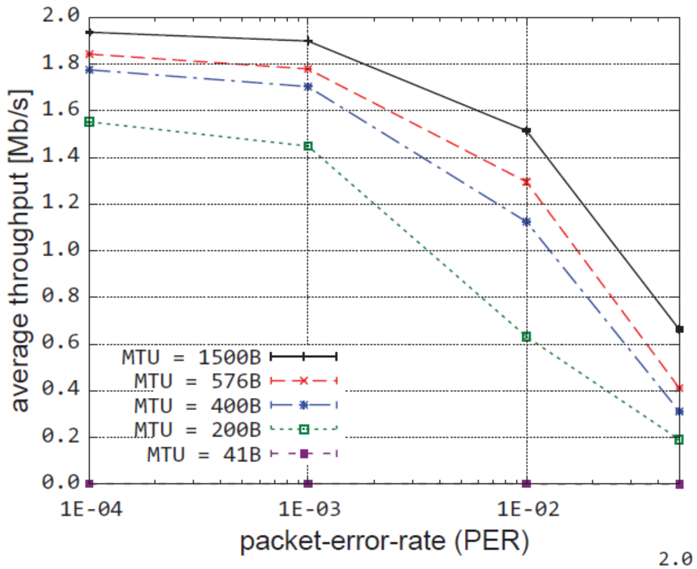
(b) Westwood+



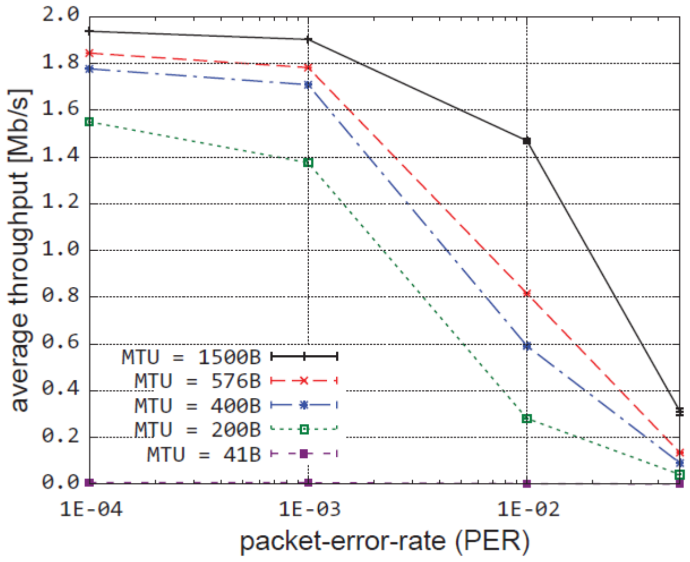
(c) Reno

Delayed ACK Timeout = 200ms

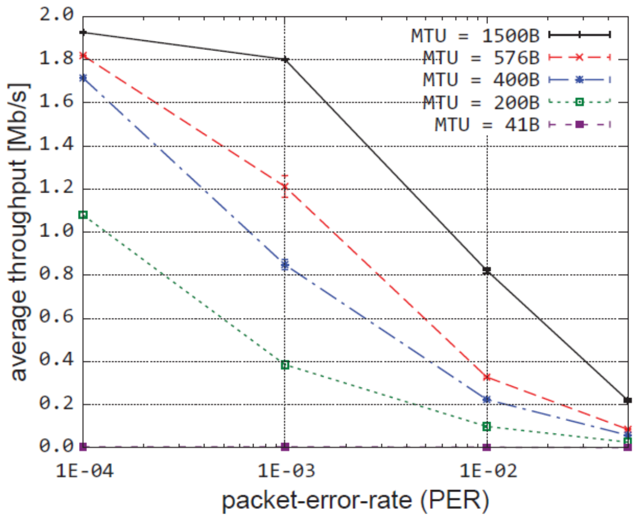
MTU Size



(a) Westwood



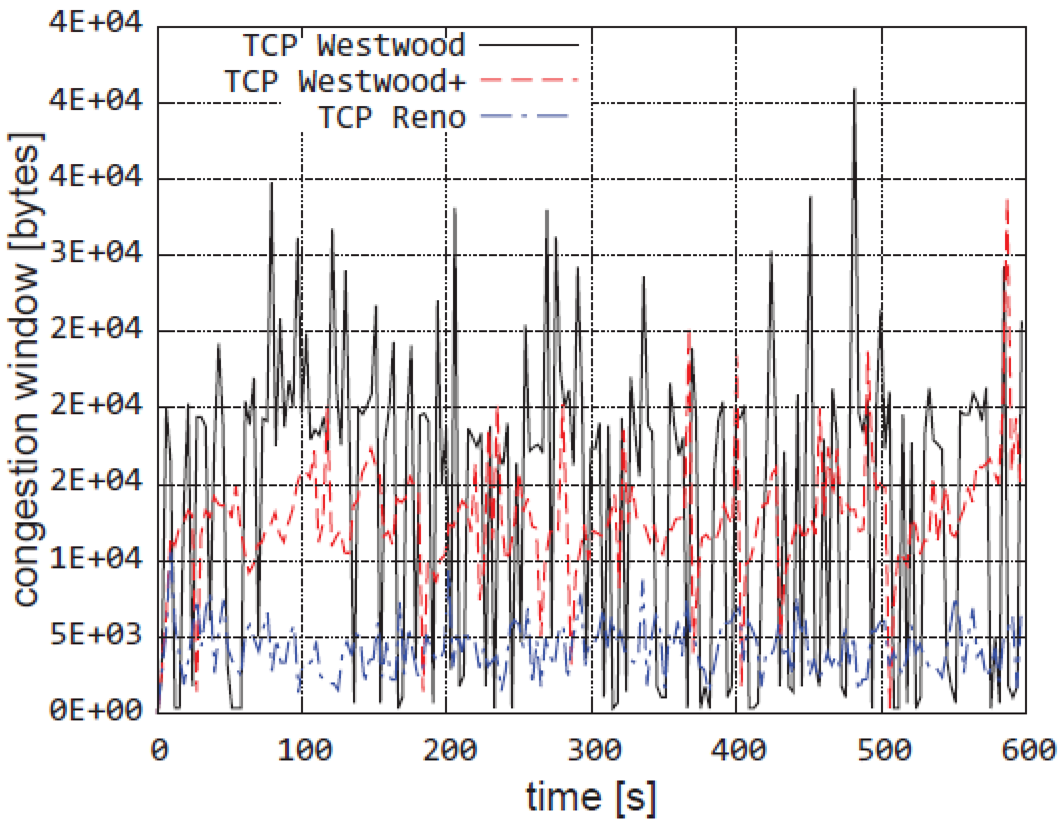
(b) Westwood+



(c) Reno

Cwnd Size

WPI

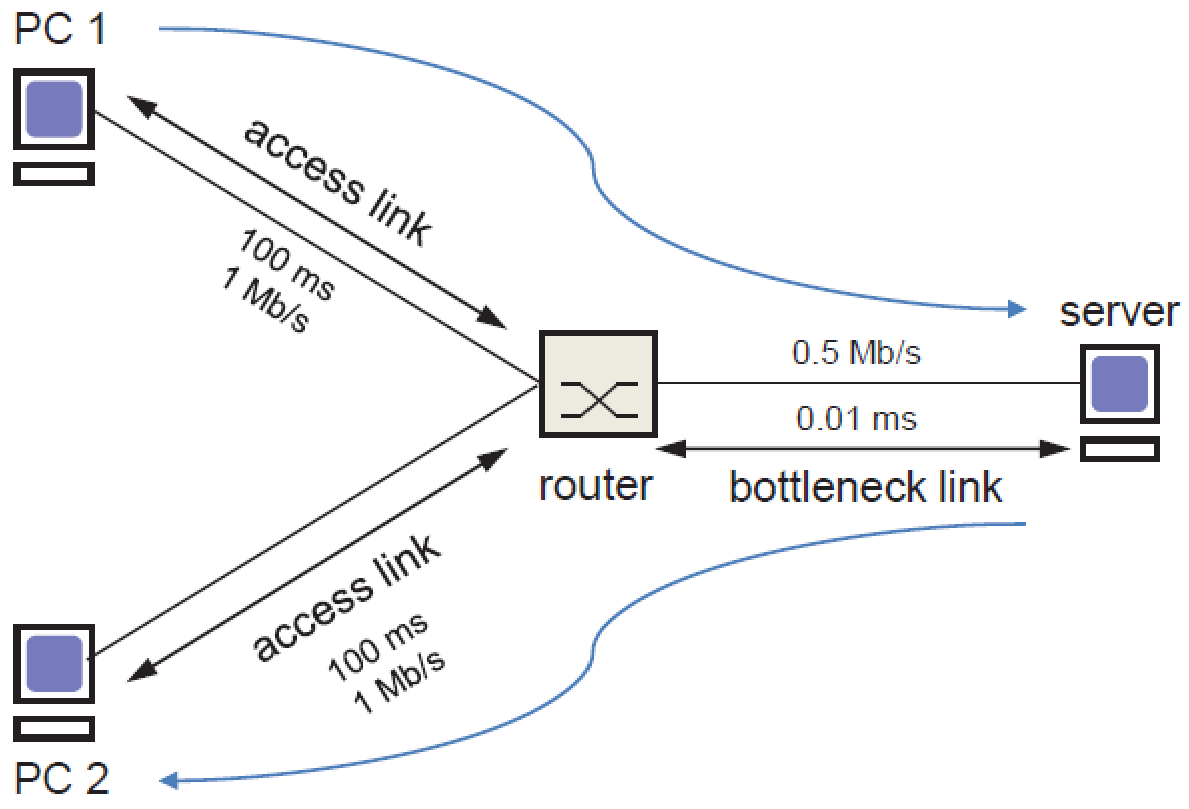


- PER = 0.005
- Samples every 3sec

Westwood+ Evaluation

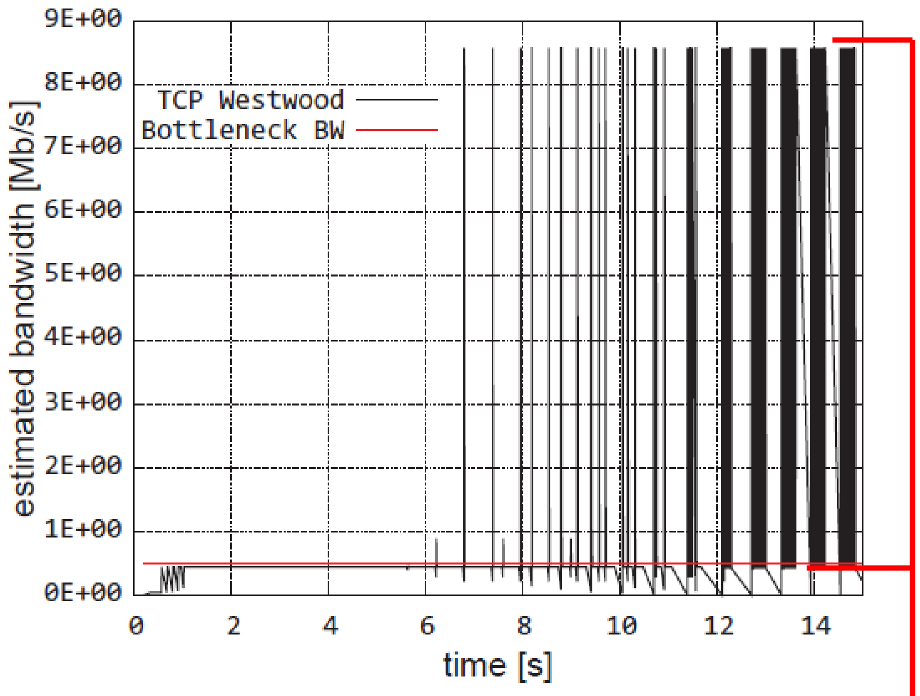
WPI

- Simulation designed to create ACK compression

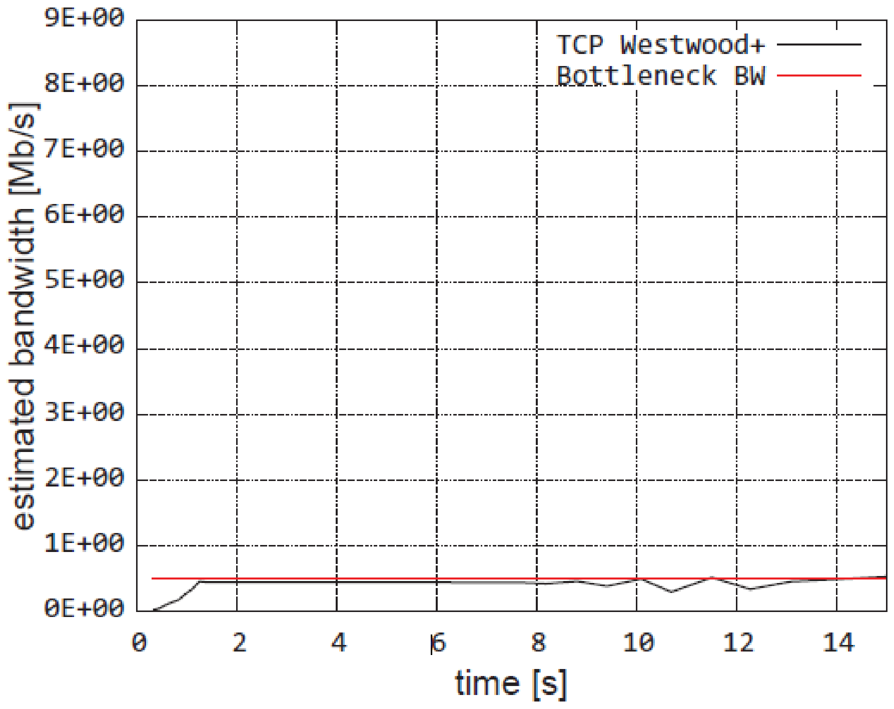


ACK Compression

WPI



(a) Westwood



(b) Westwood+

Westwood overestimates bandwidth

Conclusions

WPI

- Created Westwood(+) for ns-3
- Validated similar to original Westwood work
- Westwood+ better when ACK compression present
- Working on TCP SACK and Vegas implementations

Remarks

WPI

- Inconsistency in Reno implementation
- Tests did not emphasize Westwood+ strengths
- Comparison to original Westwood work is not as conclusive as author’s suggest
- Typo in the Westwood equation

Questions

WPI

References

WPI

- S. Gangadhar, T. Nguyen, G. Umapathi, and J. Sterbenz. TCP Westwood(+) protocol implementation in ns-3. In *ICST 2013*, pages 167-175.
- S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. In *MOBICOM 2001*, pages 287–297.