Improving TCP Start-up over High Bandwidth Delay Paths

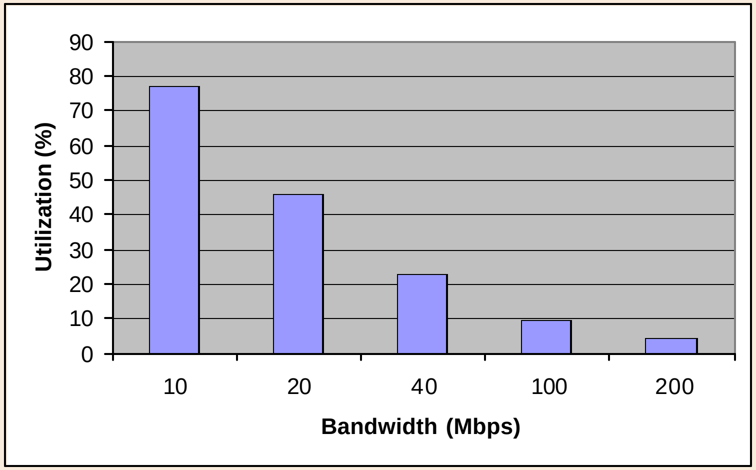
Ren Wang, Giovanni Pau, M.Y. Sanadidi and Mario Gerla

UCLA Computer Science Department

www.cs.ucla.edu/NRL

Motivation

- TCP Reno/Newreno Mechanism:
  - Slow-start: *cwnd* grows exponentially until hit *ssthresh*
  - Congestion-avoidance: *cwnd* grows linearly
- By setting initial *ssthresh* to an arbitrary value, TCP may suffer:
  - *ssthresh* too high: multiple loss and coarse timeout
  - *ssthresh* too low: premature exit of slow-start and low utilization
- Majority of TCP flows are short-lived
- The Bandwidth Delay Product(BDP) has been growing fast, resulting in poor utilization for short-lived connections with current TCP implementation:



Utilization during the first 20 seconds (RTT=100ms)

Related Work

- Larger initial *cwnd*
  - Good for transfers with a few packets
- Fast Start
  - Cached information may be stale
- Smooth Start
  - Assuming initial *ssthresh* is large enough
- Shared passive network discovery(SPANK)
  - Needs leaky bucket pacing
- TCP Vegas
- Hoe's Method

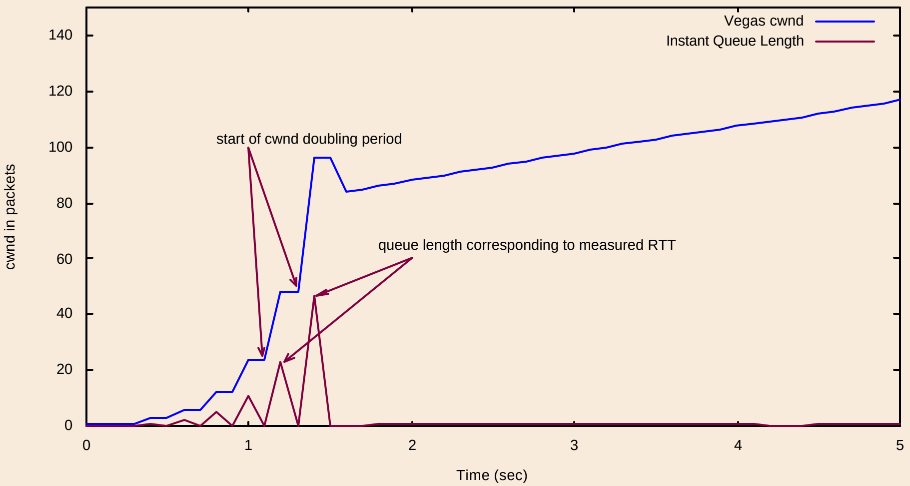
TCP Vegas(1)

- Sender watches for sign that router's queue is building up and congestion will happen; e.g.,
  - RTT grows
  - sending rate flattens
- Sender lowers sending rate to avoid buffer overflow
- During Slow-start, Vegas doubles *cwnd* every other RTT
- Until *diff* (between expected and achieved rate) exceeds a threshold:

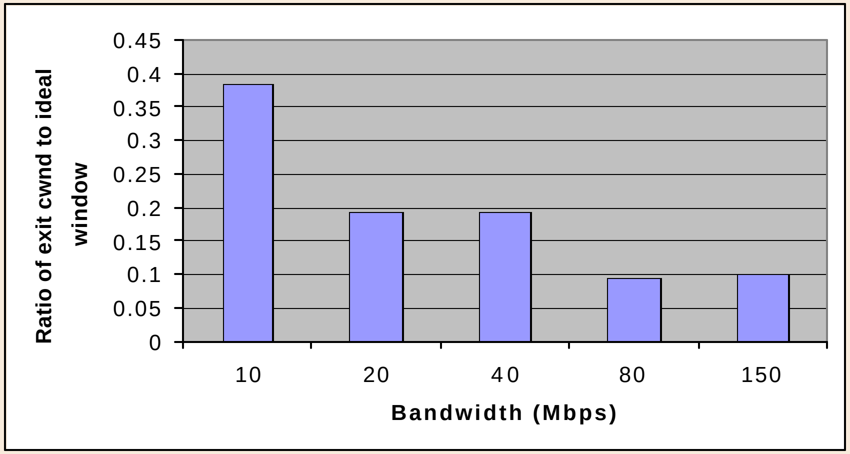
$$diff = \frac{cwnd}{baseRTT} - \frac{cwnd}{RTT_n}$$

TCP Vegas(2)

- Problem: premature exit due to RTT over-estimation, caused by temporary queue buildup:



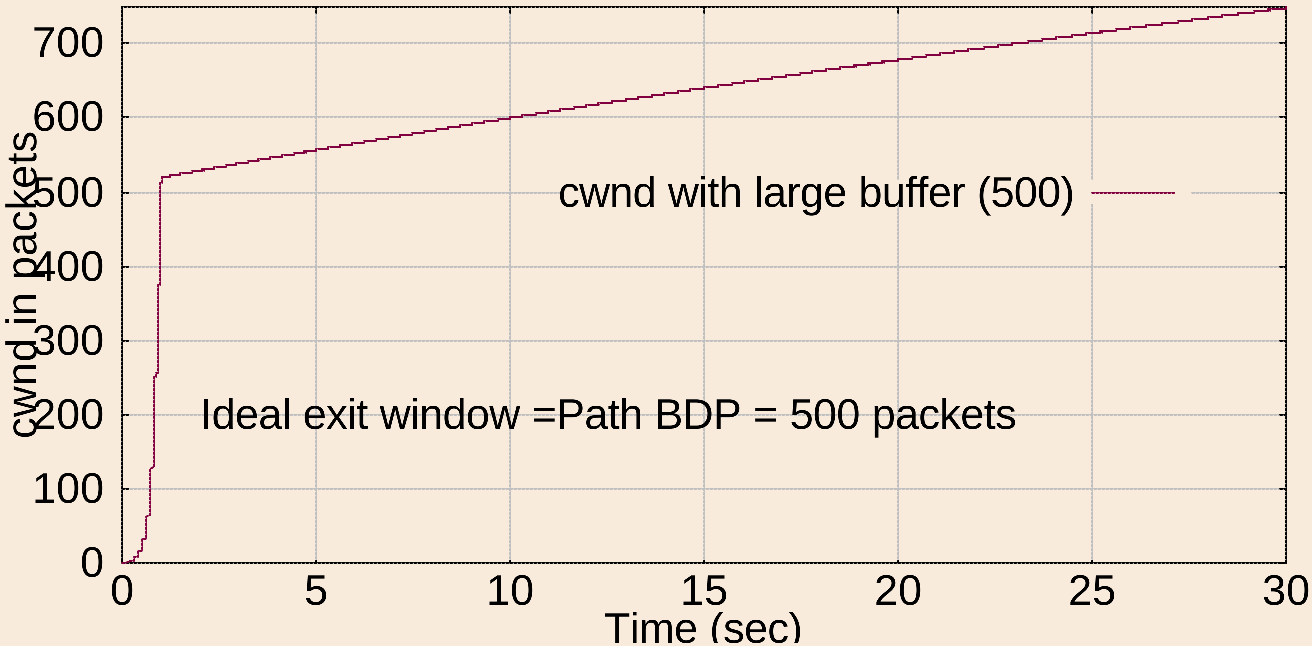
- Under-utilization becomes severe when the bandwidth delay product increases:
  - Exit *cwnd*: the congestion window when a connection exits slow start



Ratio of Slow-start termination cwnd to the ideal window (=BDP) (RTT =100ms)

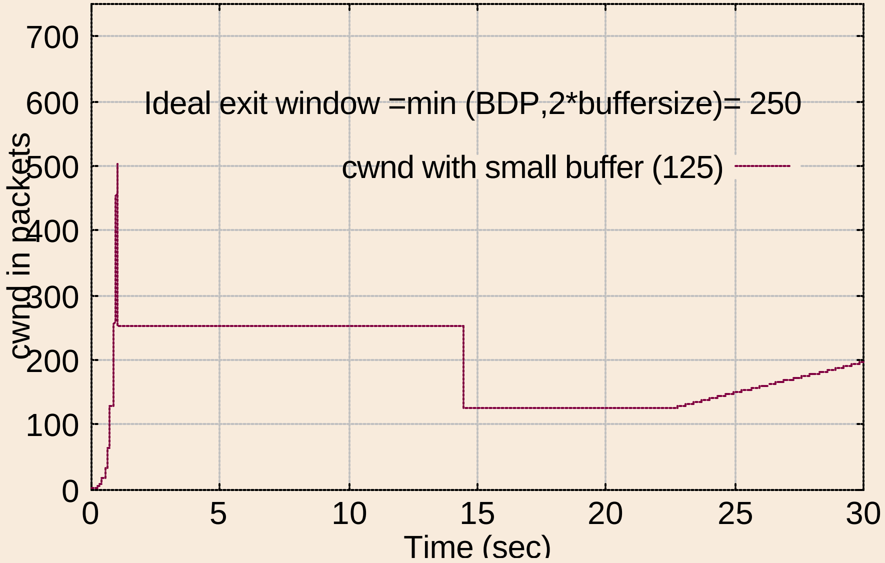
Hoe's method(1)

- Setting initial *ssthresh* to estimated BDP
  - Bandwidth: packet pair bandwidth estimate
  - RTT: measured RTT of the first segment transmitted
- May achieve high utilization, but not robust to buffer variation and dynamic load during slow start phase

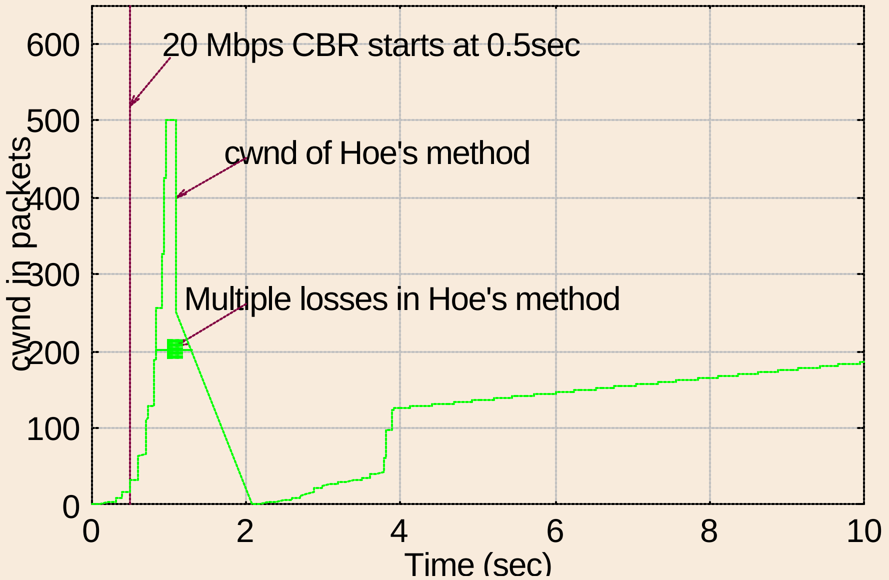


Hoe's method(2)

- Problem:
  - ➢ The bottleneck buffer is small compared to BDP
  - ➢ Other large volume traffic join in during Slow-start phase
- ➔ Multiple losses, timeout, and low utilization



(a)Small buffer cause multiple losses



(b) Traffic interference cause multiple losses

Adaptive Start (Astart) Approach

- Take advantage of Eligible Rate Estimation (ERE) in TCP Westwood (TCPW)
  - Adaptively and repeatedly reset *ssthresh* to estimated bandwidth share or Eligible Rate Estimate, if appropriate
- Key idea in TCPW and ERE
  - Enhance congestion control via the ***Eligible Rate Estimate***
    - ERE is estimated at the sender by *sampling* and *exponential filtering* measures from ACK stream
    - Samples are determined from ACK *inter-arrival times* and info about *bytes delivered*
  - after packet loss (ie, 3 DUPACKs, or Timeout), ERE is used by sender to set *cwnd*, *ssthresh=ERE x RTTmin*

TCP Westwood: the Control Algorithm

- TCPW Algorithm Outline:
  - *When three duplicate ACKs are detected*:
    - set `ssthresh=ERE*RTTmin` (instead of `ssthresh=cwin/2` as in Reno and NewReno)
    - if `(cwin > ssthresh)` set `cwin=ssthresh`
  - *When a TIMEOUT expires*:
    - set `ssthresh=ERE*RTTmin` (instead of `ssthresh=cwnd/2` as in Reno) and `cwin=1`

    Note: RTTmin = min round trip delay experienced by the connection

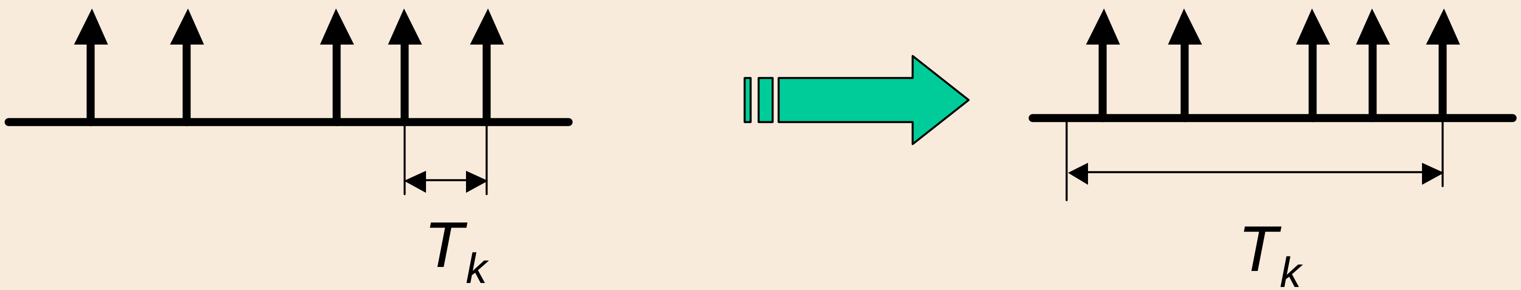
ERE in TCPW

BE Sampling: $S_k = d_k / (t_k - t_{k-1})$

Packet pair, may overestimate (e.g. in Congestion), effective in random loss

RE Sampling: $S_k = \dfrac{\sum_{t_j > t_k - RTT} d_j}{RTT}$

Packet train, fair estimate in congestion, may underestimate (e.g. in random loss)



No Congestion *Congestion*

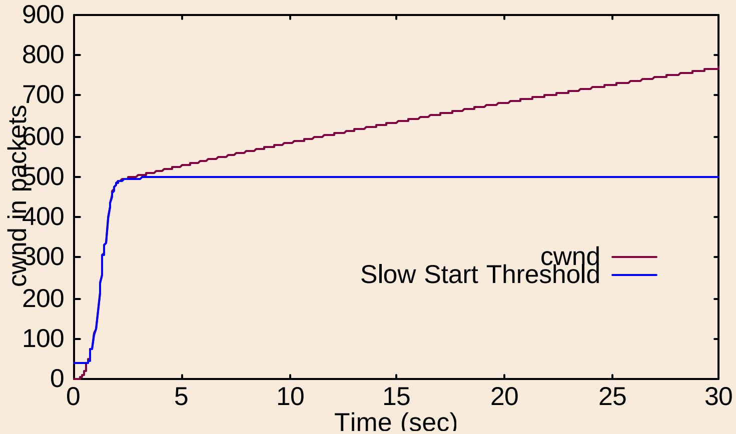
- ERE estimate: adapt the sample interval T_k according to current measured congestion level (Vegas measure of congestion level)
- T_k ranges from *one ACK interarrival interval* → *RTT*
- *RE* <= *ERE* <= *BE*

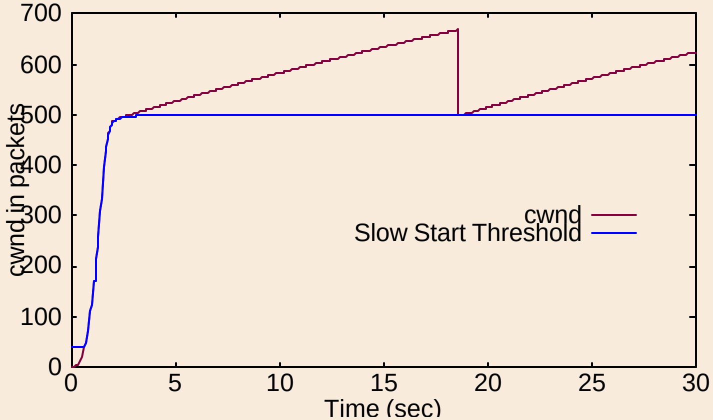
Adaptive Start (Astart) (1)

Astart uses ERE to adaptively and repeatedly reset *ssthresh* during the startup phase (connection startup and after a coarse timeout):

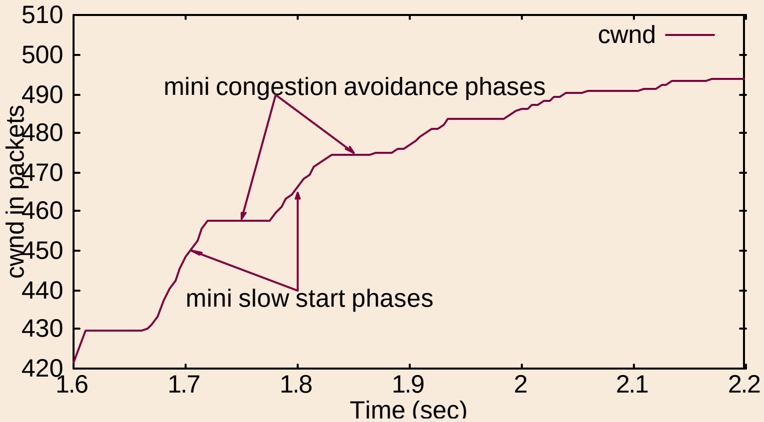
```
if ( 3 DUPACKS are received)
    switch to congestion avoidance phase;
else (ACK is received)
    if (ssthresh < (ERE*RTTmin)/seg_size)
            ssthresh =(ERE*RTTmin)/seg_size;
    endif
    if (cwnd >ssthresh)  /*mini congestion avoid. phase*/
        increase cwnd by 1/cwnd;
   else if (cwnd <ssthresh)  /*mini slow start phase*/
        increase cwnd by 1;
   endif
endif
```

Astart (2)





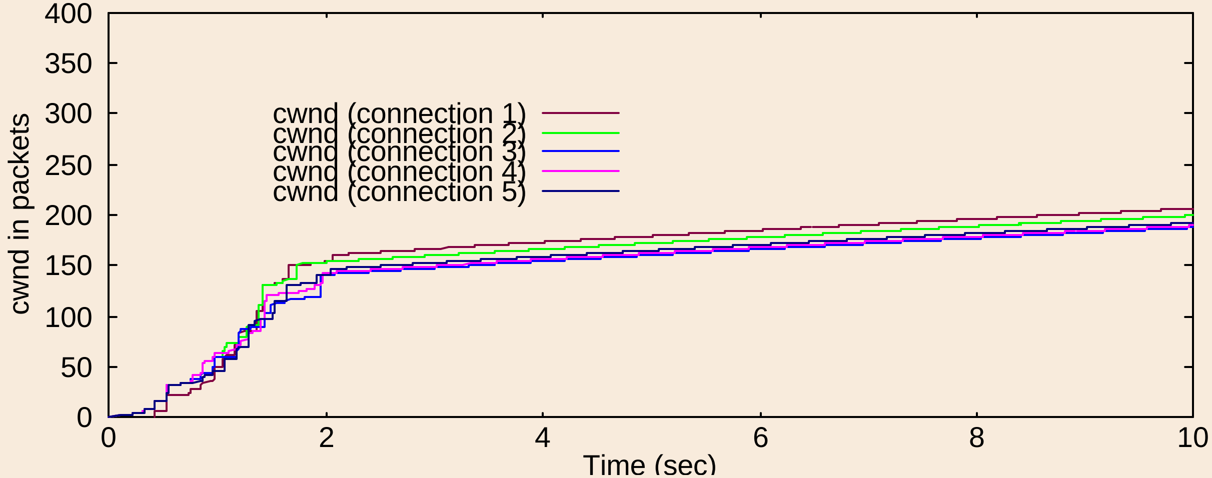
(a)Big buffer (500 packets)   BDP = 500 packets   (b) Small buffer(125 packets)



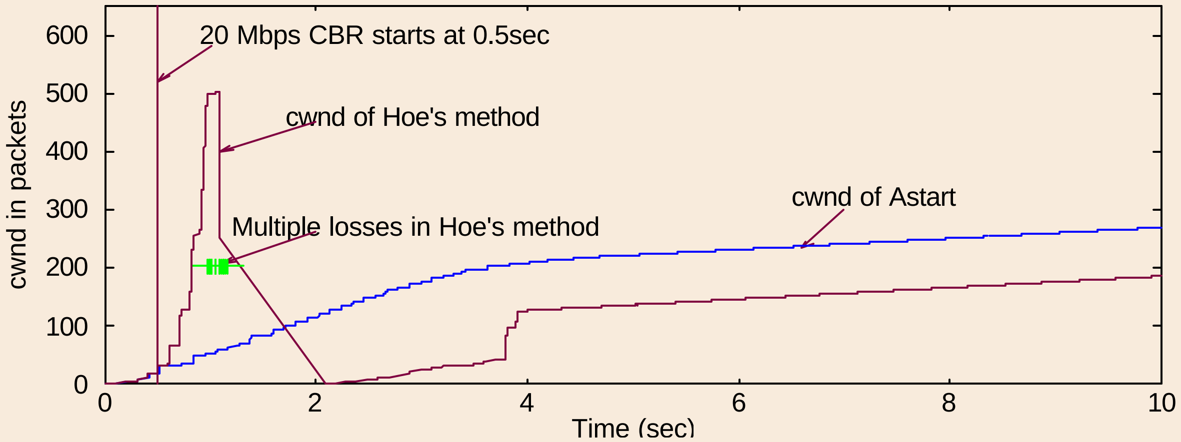
(c)A closer look at Astart *cwnd* dynamic

- Astart continuously uses ERE
- Contains mini slow-start and mini congestion-avoidance phases
- *cwnd* grows slower as it approaches BDP

Astart (3)



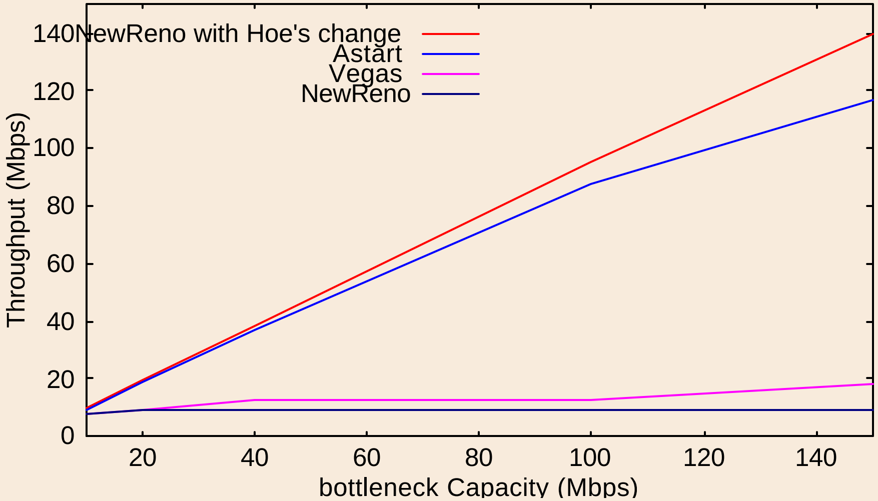
Astart *cwnd* dynamic with 5 connections startup simultaneously



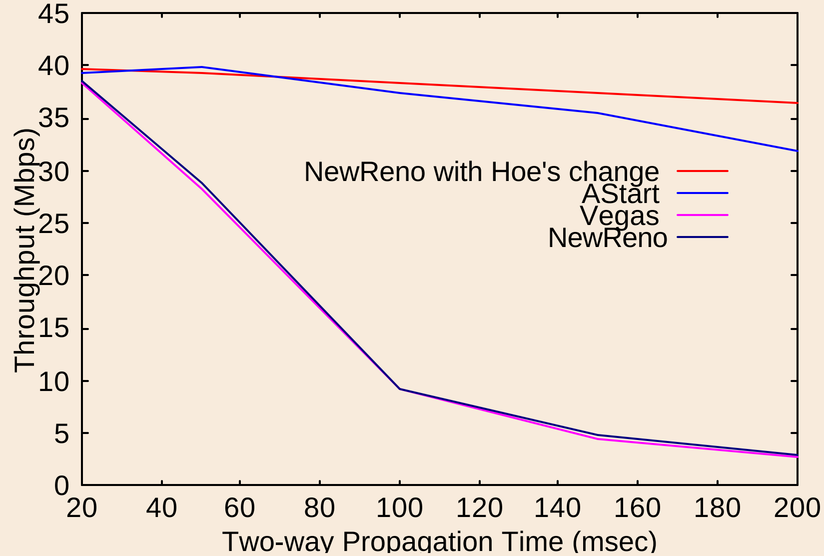
cwnd dynamic with UDP traffic joins in during startup phase (compare Astart and Hoe's method)

Throughput Comparison

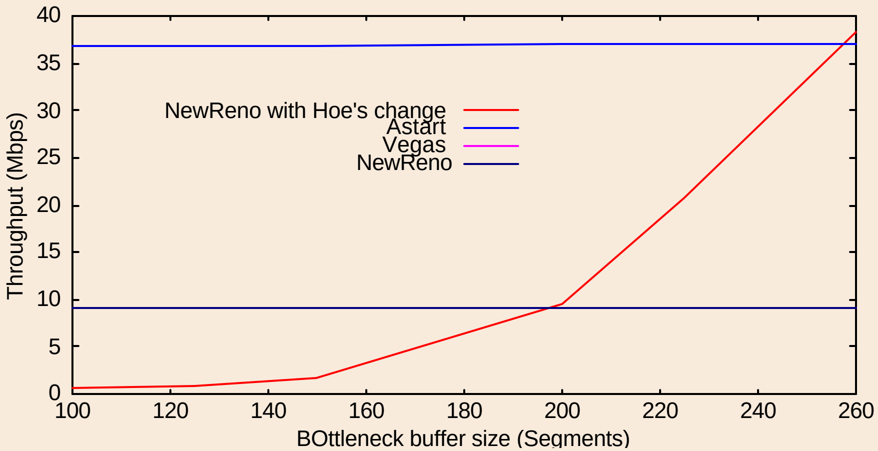
Throughput vs. Bottleneck Capacity (During first 20 seconds) (RTT =100ms, Buffer size =BDP)



Throughput vs. Two-way Propagation Time (During first 20 seconds) (Bottleneck capacity = 40 Mbps, Buffer size =BDP)



Throughput vs. Bottleneck Buffer size (During first 20 seconds) (RTT =100ms, Bottleneck =40 Mbps)

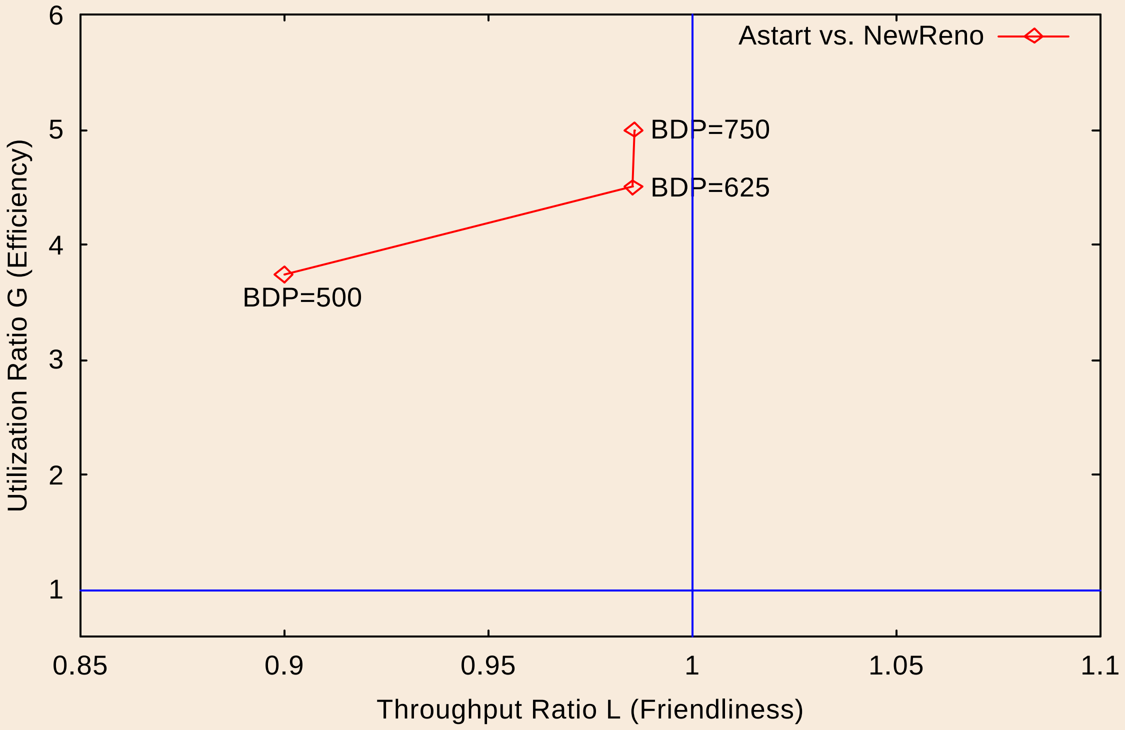


Results summary

- Scaling with Capacity
- Robust to buffer size
- Scaling with Propagation time

Efficiency/Friendliness

- Use Efficiency/Friendliness Tradeoff Graph
  - X-axis represents friendliness
  - Y-axis represents efficiency



Simulation setup:

- Bottleneck: 40 Mbps
- BDP: Varies with RTT
- Two connections start up at the same time
- Record the throughput during first 5 seconds
- Calculate Utilization ratio and throughput ratio

Summary

- Reviewed and evaluated Vegas, Hoe's method
- Presented Astart, a new approach based on TCPW ERE
- Compared throughput, scaling with BDP, and robustness to buffer and load variations
- Hoe's method provides high throughput, but Astart comes very close, AND is robust to buffer size and dynamic load
- Astart is another illustration of the benefits of pursuing estimates of bandwidth measures to improve congestion control in TCP

References

- The papers about TCP Westwood, TCP Westwood CRB and ABSE can be found in the papers section of the TCP Westwood Web Page: http://www.cs.ucla.edu/NRL/hpi/tcpw/
- TCP Vegas: New Techniques for Congestion Detection and Avoidance. Lawrence Brakmo, Sean O'Malley, and Larry Peterson. In *ACM SIGCOMM*, pages 24-35, August 1994
- Hoe's Method: J. C. Hoe, "Improving the Start-up Behavior of A Congestion Control Scheme for TCP", Proc. ACM SIGCOMM '96.
- Fast Start: V.N. Padmamabhan and R.H. Katz, "TCP Fast Start: A Technique for Speeding Up Web Transfers", Proceedings of IEEE globecom'98, Sydney, Australia, Nov. 1998.
- Smooth Start: H. Wang, H. Xin, D.S. Reeves and K.G. Shin "A Simple Refinement of Slow Start of TCP Congestion Control", In proceedings of ISCC'00, Antibes, France, 2000
- Large initial window: M. Allman, S. Floyd and C. Patridge, "Increasing TCP's initial Window", INTERNET DRAFT, April 1998
- SPANK: Y. Zhang, L. Qiu and S. Keshav, "Optimizing TCP Start-up Performance", Cornell CSD Technical Report, February, 1999