Using IP-QOS for High Throughput TCP Transport Over Fat Long Pipes

Andrea Di Donato, Yee-Ting Li, Frank Saka, Peter Clarke
Centre of Excellence on Networked Systems
University College London, United Kingdom

PFLDnet2004

Issues/summary

- TCP's congestion control algorithm additive increase multiplicative decrease (AIMD) performs poorly over paths of high bandwidth-Round-Trip-Time product and the aggregate stability is even more precarious if such aggregate consists of only few flows as it is the case for the scientific community where few users require a reliable bulk data transfer at Gb/s speed.

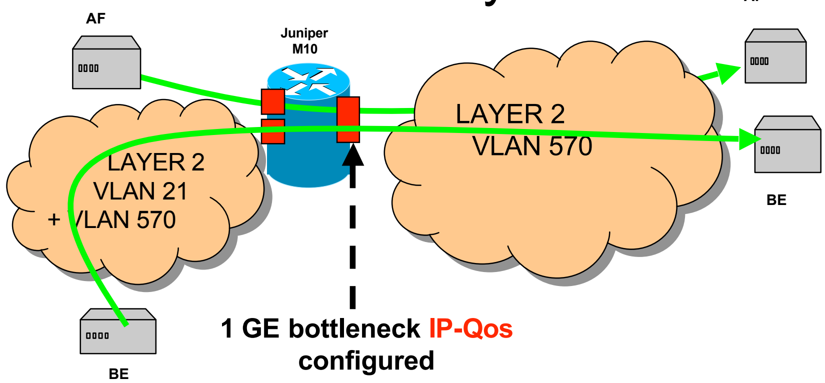
_[(c^3 _^3)/(4N^4)] * [(c_/2N)+1+1/(_c_)] < [π(1-_)^2] / √[4_^2 + π^2(1- _)^2]

where :

- _ = slope of marking probability in the router queues
- _ = weight that the router uses to estimate the average queue occupancy
- **c** = capacity
- **N** = # of flows
- _ = RTT

(*) linear stability of TCP/RED and a scalable control. Steven H. Low, Fernando Paganini, Jiantao Wang, John C. Doyle, May 6, 2003

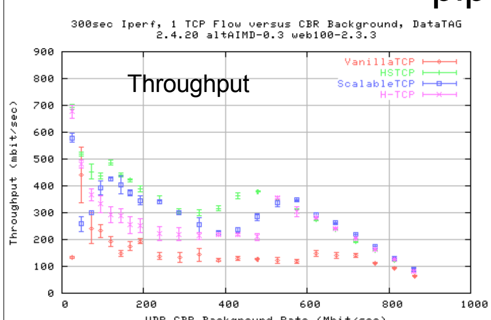
Testbed layout

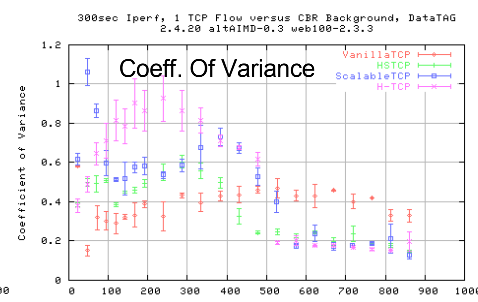


Path
- Bottleneck = 1Gbps
- Min RTT = 118 ms

PCs:
- Supermicro 6022P-6 Dual Intel® Xeon
- Linux Kernel: 2.4.20
- NIC: Sysconnect 98xx
- NIC driver : sk98lin 6.19

Standard Vs New TCP Stacks: the inadequacy of standard TCP in fat long pipes



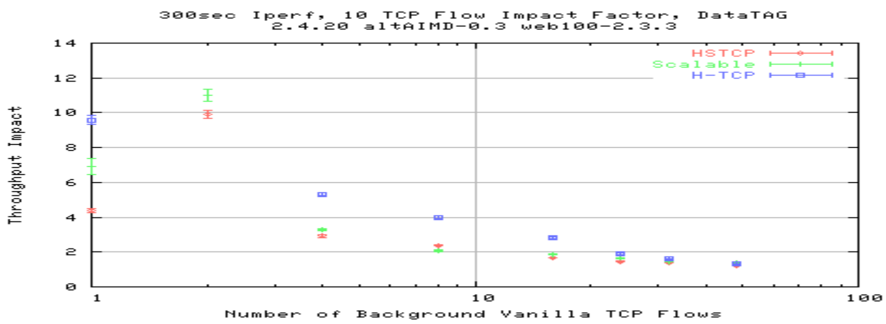


Standard TCP Th. always gets, on average, no more than 150M regardless of the amount of the CBR background injected. This indicates how weak and sensitive standard TCP is for high BW-delay product paths. Thus, **only in principle** there is availability of empty packet-switched gigabit paths. Standard TCP also shows a not negligible variability in throughput.

(*) Coefficient of variance = Stdev Throughput / Mean Throughput

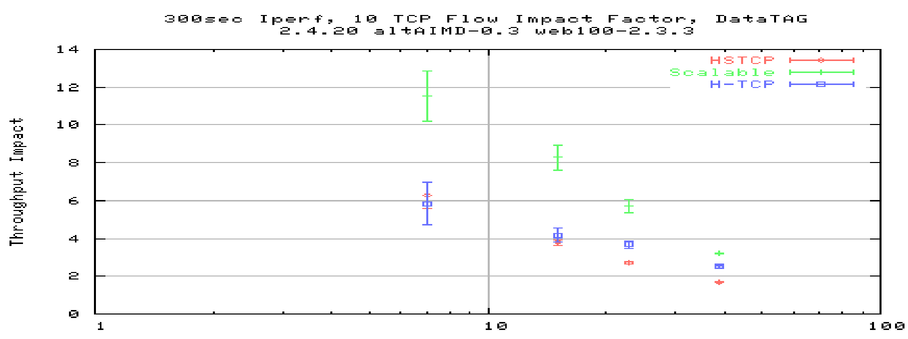
New stacks: **U**ser **I**mpact **F**actors investigation(1)

1/UIF where **UIF =** $\dfrac{NVanillaTCPflowsThroughputWhenWith1NewTCP * \frac{1}{N}}{(N+1)VanillaStandaloneThroughput * \frac{1}{(N+1)}}$



When N < 30The throughput of 1 vanilla flow when part of an aggregate with one new TCP stack flow is smaller than the throughput it achieves when part of an aggregate with NO New TCP stack flows.

New stacks: **U**ser **I**mpact **F**actors investigation(2)



The same as for the previous plot but for **10 New TCP stacks:**
The UIPs are even higher than 1 with respect to the 1 new TCP flow scenario and this corroborates the results even more as this is a much more realistic scenario.

The need for QoS….

The results obtained so far suggest we need a mechanism for segregating traffic sharing the same packet-switched path. This is in order to

1. Protect traditional BE traffic from the possible aggression of the new TCP stacks
2. Guarantee a level of end-to-end service predictability for TCP flows which is sufficient to enforce a network resources reservation system through the use of some sort of middleware (GRID)

IP-QOS (a la Diff. Serv.)

- Packets are dscp-marked at the sender host
- Two classes are defined:
  - BE
  - AF
- Packet are dscp-classified at the router input ports.
- Each class is put in a physically different IP-level router output queue. Each of them is assigned a minimum BW guaranteed (WRR).
  - Low priority for BE
  - High priority for AF

Juniper M10 BW scheduler "priority"

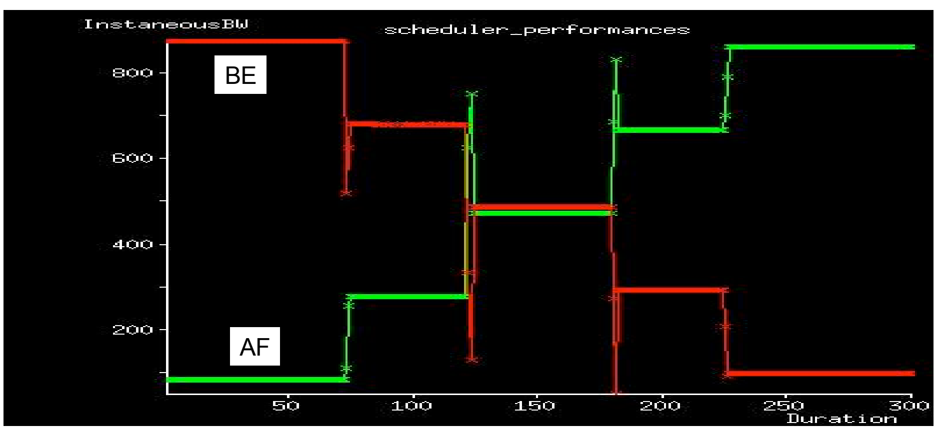
- The queue weight ensures the queue is provided a given minimum amount of bandwidth which is proportional to the weight. As long as this minimum has not been served, the queue is said to have a "positive credit". Once this minimum amount is reached, the queue has a "negative credit".

- The credits are decreased immediately when a packet is sent. They are increased frequently (by the weight assigned).

- For each packet, the WRR algorithm strictly follows this queue service order:

  1. High priority, positive credit queues
  2. Low priority, positive credit queues
  3. High priority, negative credit queues
  4. Low priority, negative credit queues

Router BW scheduler benchmarking



JUNIPER M10

• Cards used: 1x G/E, 1000 BASE-SX REV 01
• IOS Version used: "Junos 5.3R2.4"

Router config

```
classifiers {
    dscp UCL-classifier {
        forwarding-class AF {
            loss-priority low code-points 001000;
        }
        forwarding-class BE {
            loss-priority high code-points 000000;
        }

    }
}
forwarding-classes {
    queue 0 BE;
    queue 1 AF;
}
schedulers {
    sch-AF {
        transmit-rate percent X;
        buffer-size percent X;
        priority high;
        drop-profile-map loss-priority low protocol any drop-
profile wred1
    }
    sch-BE {
        transmit-rate percent Y;
        buffer-size percent Y;
        priority low;
    }
  }
drop-profiles {
        wred1
         {
     fill-level 30 drop-probability 15;
    fill-level 50 drop-probability 30;
    fill-level 80 drop-probability 40;
    fill-level 100 drop-probability 60;
        }
scheduler-maps {
        scheduler-map-UCL {
            forwarding-class BE scheduler sch-BE;
            forwarding-class AF scheduler sch-AF ;
        }
    }

      interfaces {
        input_port_name {
            unit 0 {
                classifiers {
                    dscp UCL-classifier;
                }
            }
        }
input_port_name {
            unit 0 {
                classifiers {
                    dscp UCL-classifier;
                }
            }
        }
output_port_name {
            scheduler-map scheduler-map-UCL;
            unit 0 {
                classifiers {
                    dscp UCL-classifier;
        }
```

QoS test result summary table

AF-BE / TCP	10-90 %	30-70 %	50-50 %	70-30 %	90-10 %	Clamping at 90-10 %
Scalable	AF-ts=100 AF-s=100 BE-s=100	AF-ts=99 AF-s=100 BE-s=100	AF-ts=89.4 AF-s=90.8 BE-s=109.3	AF-ts=81 AF-s=80.4 BE-s=146.6	AF-ts=72.99 AF-s=84.4 BE-s=258.6	AF-ts=96.5 AF-s=94.7 BE-s=105
Standard	AF-ts=100 AF-s=100 BE-s=100	AF-ts=100 AF-s=100 BE-s=100	AF-ts=99.53 AF-s=99.7 BE-s=100	AF-ts=99.1 AF-s=99.6 BE-s=100	AF-ts=94.15 AF-s=96.5 BE-s=132.1	AF-ts=97.5 AF-s=96.3 BE-s=102
Hs-TCP	AF-ts=100 AF-s=100 BE-s=100	AF-ts=100 AF-s=100 BE-s=100	AF-ts=98.5 AF-s=99.2 BE-s=101.4	AF-ts=93.1 AF-s=96 BE-s=107.7	AF-ts=83.7 AF-s=97.9 BE-s=131.2	AF-ts=97.2 AF-s=96.1 BE-s=102

AF-QosEfficiencyFactors = AF-QEFs= AF-TCP_Throughput / AF_QoS_pipe_allocated

BE-QosEfficiencyFactors=BE-QEFs= BE_CBR_Throughput / BE_QoS_pipe_allocated

S = Steady state test

Ts= Transient and steady state test

BE/AF **Q**oS **E**fficiency **F**actors (1)

- 50MB socket buffersize
- Congestion moderation on
- txqueuelen and backlog = 50000
- TCP with an offset of 30 sec.

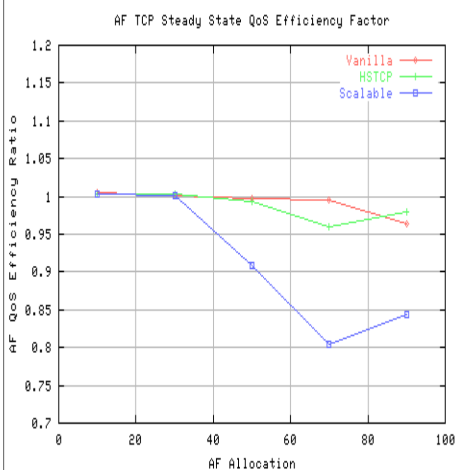


Figure 1 AFs-QEF in steady state

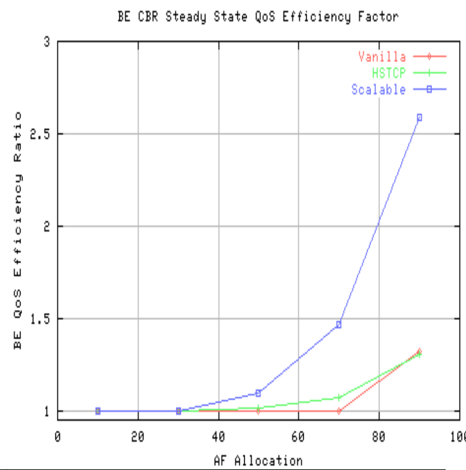
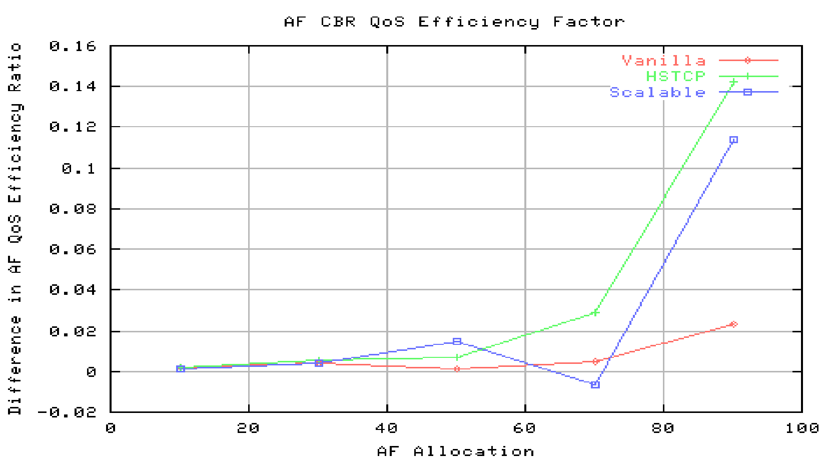


Figure 2 BEs QEF in steady state

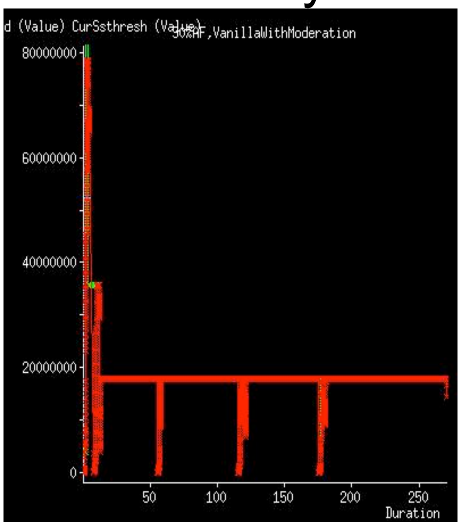
BE/AF **Q**oS **E**fficiency **F**actors (2)

AFs-QEF – AFts-QEF as an indication of the convergence speed



This is when HS and Standard TCP differ

Standard TCP AF90%:typical dynamic in time



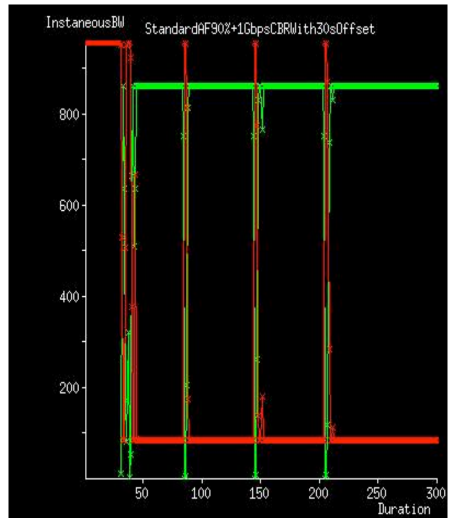
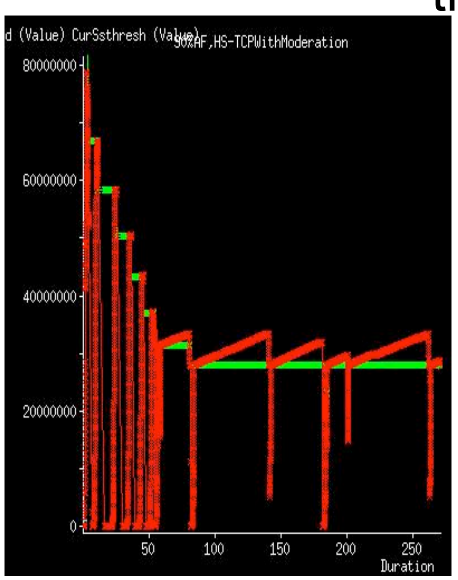


Figure 6: Vanilla 90% cwnd	Figure 7: Vanilla 90% Rx. Throughput

Hs-TCP 90%: typical dynamic in time



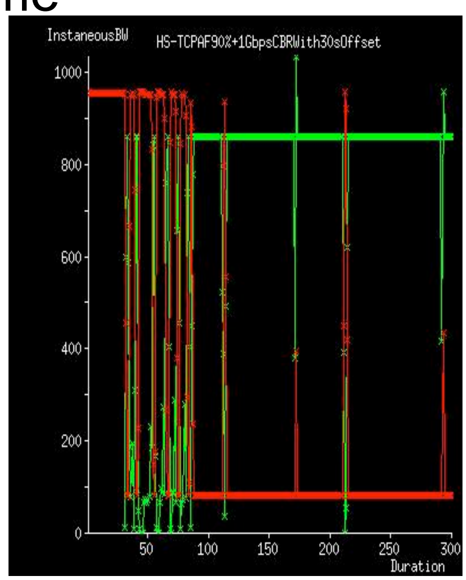
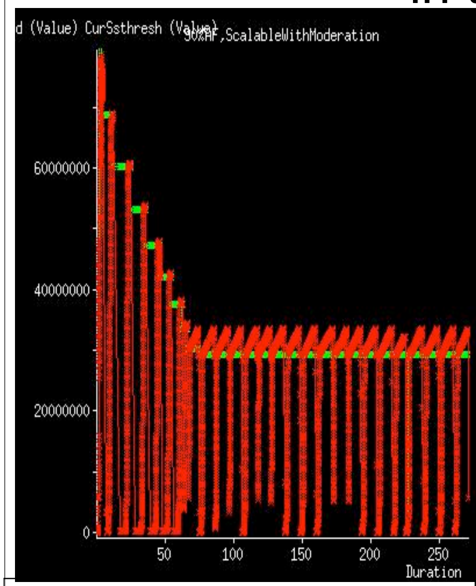


Figure 8: Hs-tcp 90% cwnd	Figure 9: Hs-tcp 90% Rx. Throughput

Scalable TCP 90% typical dynamic in time



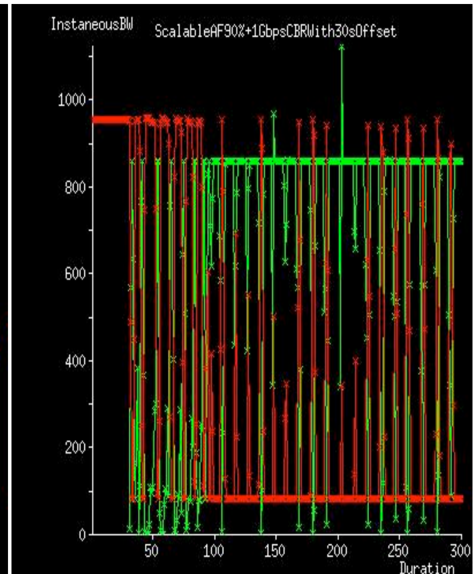
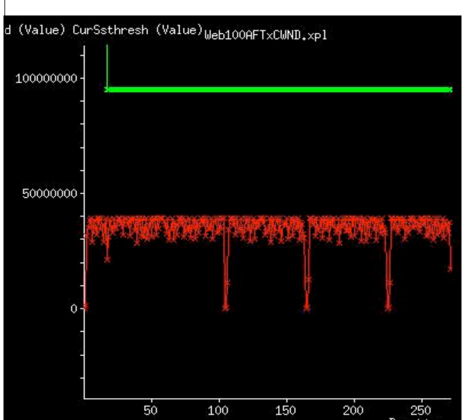
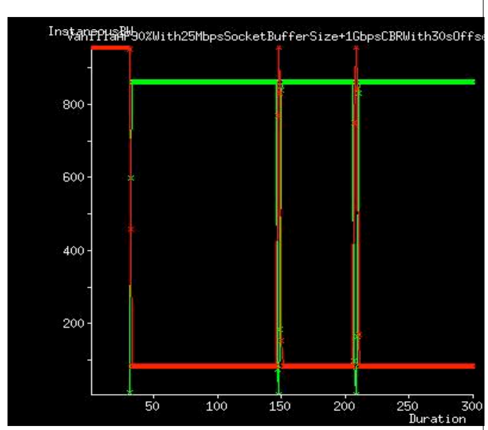


Figure 4: scalable 90% cwnd	**Figure 5: scalable 90% Rx.throughput**

Clamping for 90%AF(40MB Socket Buffer Size)

Same performances regardless of the TCP stack since it is “NOT” TCP.

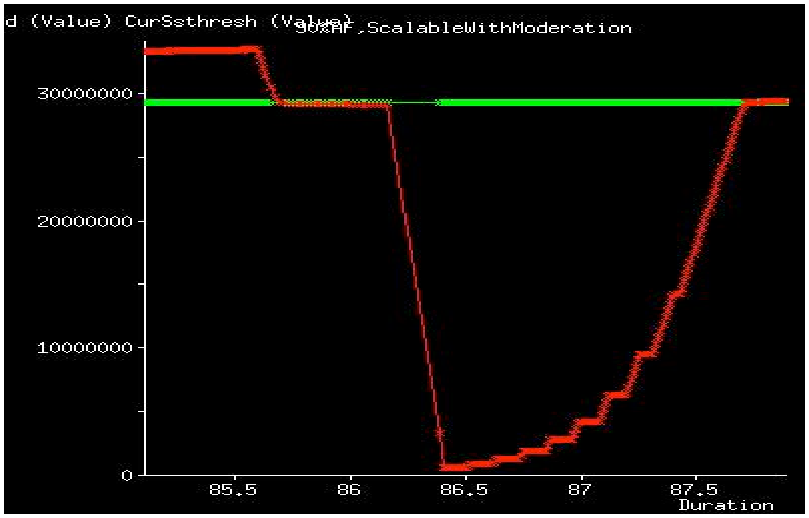




CWND = 40MB = 1Gbps * 320ms ….320 ms is the RTT under interface congestion with 900Mbps pipe allocated for AF

Clamping: CWND	**Clamping:Throughput**

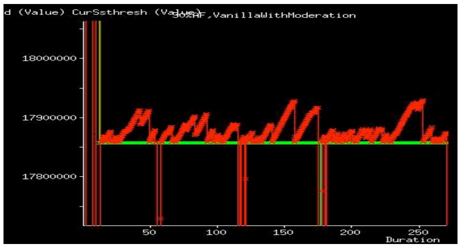
Zoom on the "period of silence" experienced from all the TCPs senders but with different magnitude

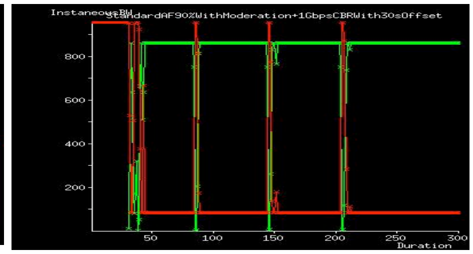


Rationale for turning Cong. Moderation off

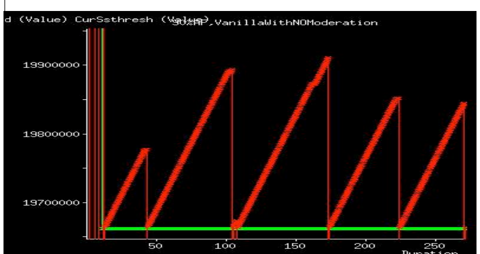
- All protocols experienced (with different magnitudes) "unusual" slow start periods.
- "Unusual" as there were not any timeouts experienced.
- One thing that may have caused this is "Congestion Moderation". One more reason to turn it off is that is not an IETF standard.
- Congestion moderation is triggered by a dubious event which is typically the acknowledge of more than 3 packets. The action induced is that the CWND is set to the # of packets are thought to be in flight.

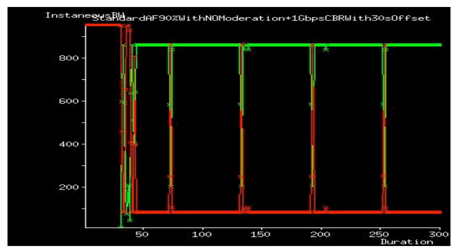
No Congestion Moderation: Standard TCP(1)





90% with Moderation: CWND	90% With Moderation: Rx.Throughput





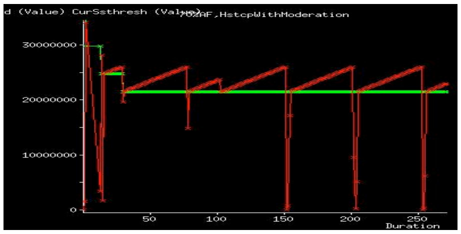
90% with NO Moderation: CWND	90% with NO Moderation: Rx. Throughput

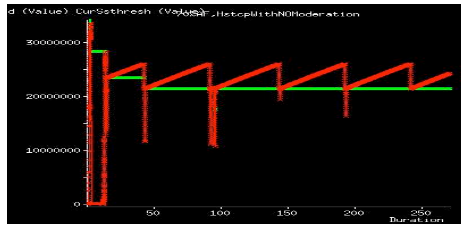
No “Congestion Moderation”: Standard TCP(2)

- There is no significant difference in the CWND dynamic with or without congestion moderation for all the AF allocations but 90%.
- Although the 90% allocation shows difference in CWND dynamics, all the unusual drops experienced are **mainly** caused by UNDOs and not by Congestion Moderation. The UNDOs seem to have a **positive effect** in that they prevent the CWND to grow while preserving the throughput performance.
- The CWND dynamic without Congestion Modification is smoother and sweeps a wider values range. The absolute CWND range is also bigger.
- We are not in a position to say which is better.

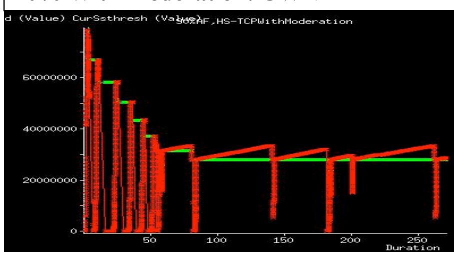
UNDOs = ss-thresh and cwnd reduce as loss is thought to have happened but the sudden reception of THE ack makes both cwnd and ss-tresh to reset to the previous value

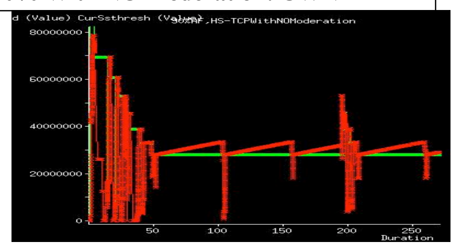
No Congestion Moderation: HighSpeed-TCP(1)





70% With Moderation: CWND	70% With NO Moderation: CWND



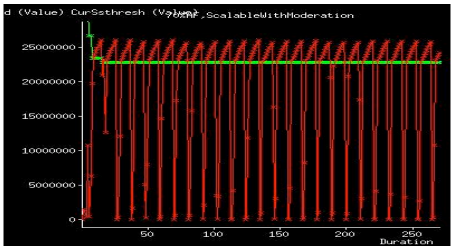


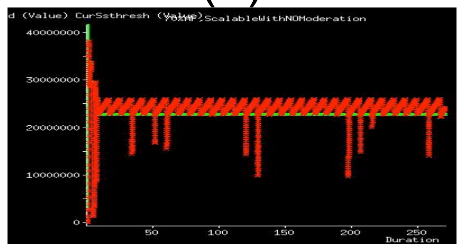
90% With Moderation: CWND	90% With NO Moderation: CWND

No Congestion Moderation: HighSpeed-TCP(2)

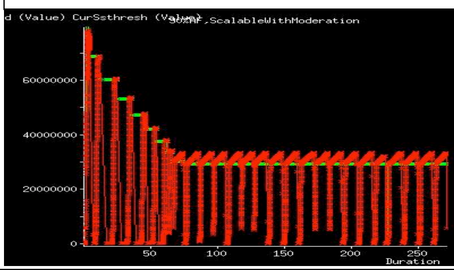
- Congestion moderation turned off is **highly** beneficial only when a 700Mbps pipe is allocated to the AF flow
- Congestion moderation turned off is only **partially beneficial** when 900 Mbps pipe is allocated to the AF flow

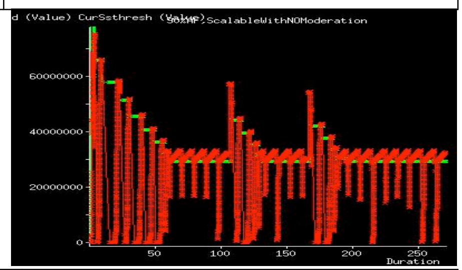
No Congestion Moderation:Scalable(1)





70% With moderation: CWND	70% With NO Moderation: CWND





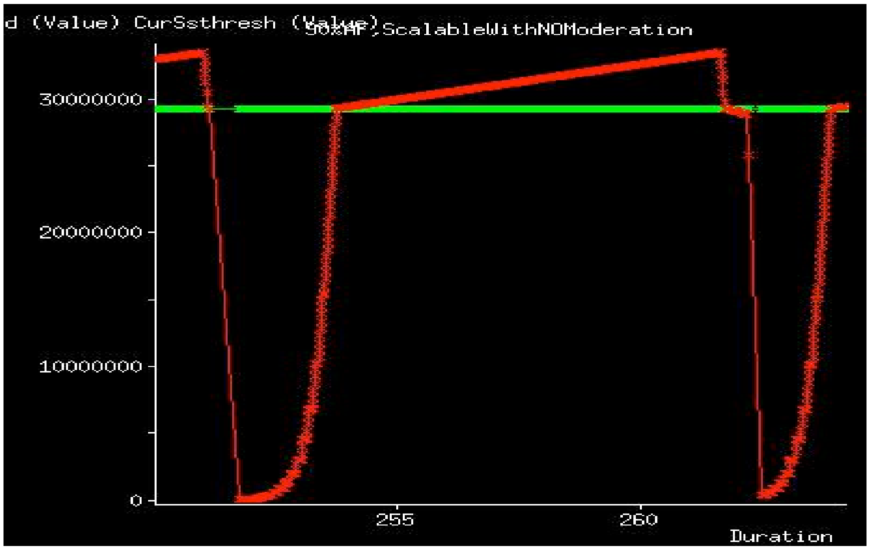
90% With Moderation: CWND	90% With NO Moderation: CWND

No Congestion Moderation:Scalable(2)

- Congestion moderation turned off is **extremely beneficial** for a 700 Mbps pipe allocated to the AF flow.

- Congestion moderation turned off **is only partially beneficial** for a 900 Mbps pipe allocated to the AF flow.

No Cong. Mod.- Scalable(3):remaining silence



No Cong. Mod.-Scalable(3): Analysis

<u>We know the current SACK processing is not optimal yet.</u>

It is likely that the SACKs coming back might cause the sender host to:

- Hang **a bit (shorter periods of silences)** when AF is allocated 70% and below, this causing a bad (lower) estimation of the # of packets in flight.

- Hang **much more deeply (longer periods of silence)** when 90% is used, this way making the Congestion moderation bad estimation effect ineffective as it is dominated and therefore mostly neutralised by the host resetting everything an thus restarting from slow start after a period of silence.

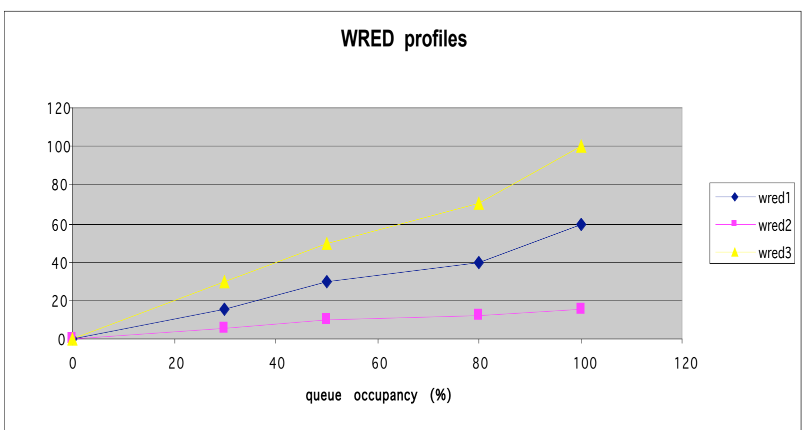
(*)Tom Kelly SACKs patch is used.

WRED(1): *test rationale*

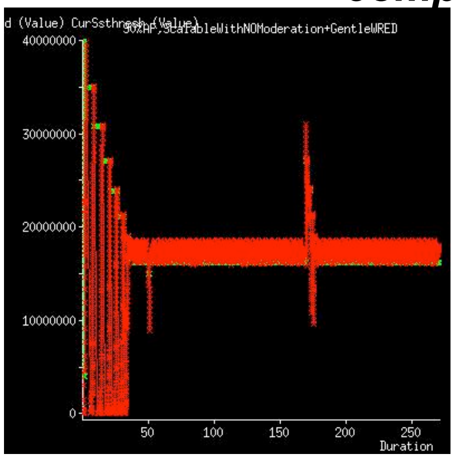
- WRED is applied on Scalable TCP as it showed the worst result but it could have also been done for Hs-TCP
- Only 900 and 700 Mbps pipes are involved in the tests as they showed the worst performance.
- Congestion moderation is kept **turned off** based on the good results that the aforementioned tests showed
- Dual approach rationale:
  – **Flow-level perspective**:
    - The oscillations are the effect of protocol instability. This being the case, a gentle (small _) Weighted Random Early Detection (WRED) drop profile should be able to facilitate the validation of the inequality expression governing the stability condition for the TCP flow [low]
  – **Segment-level perspective**
    - A smoother distribution of the loss pattern all along the AF scheduler buffer would help in lowering the rate of bursts of SACKS coming back and therefore would help in not stalling the sender.

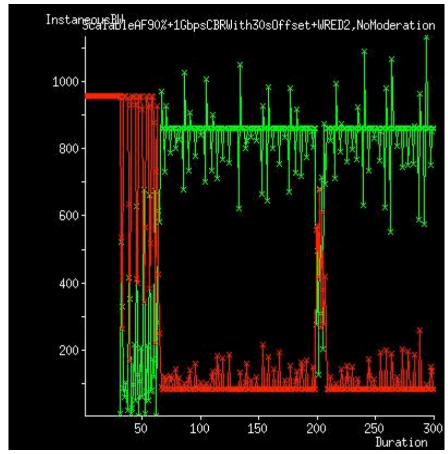
[low] linear stability of TCP/RED and a scalable control. Steven H. Low, Fernando Paganini, Jiantao Wang, John C. Doyle, May 6, 2003

WRED profiles used (2)



Wred(3): Scalable, main results and comparison

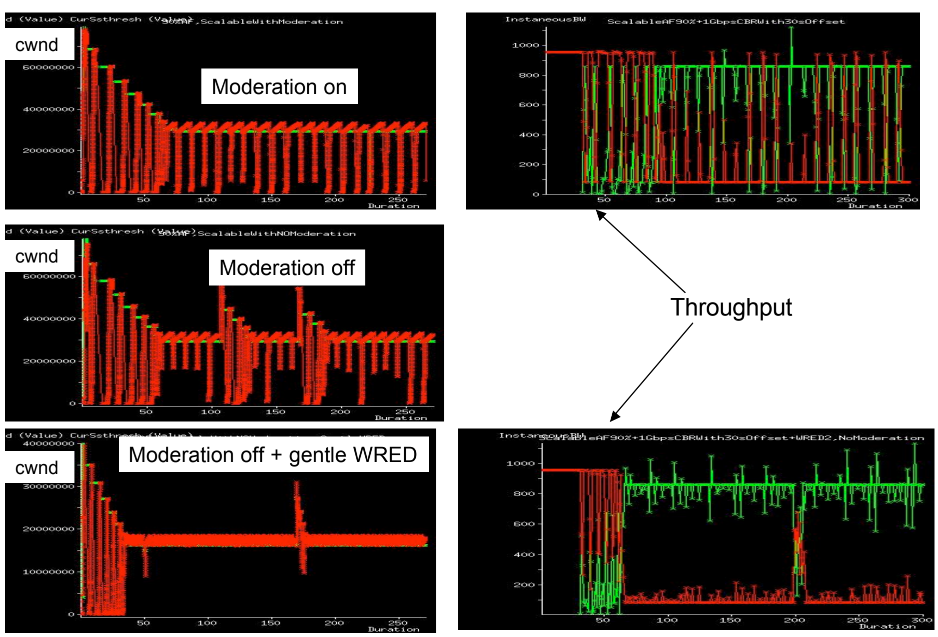




90% With No Moderation and with a gentle WRED (3): CWND	**90% With No Moderation and with a gentle WRED (3): Rx. Throughput**

As expected, only a gentle WRED works. Other profiles are found to be too aggressive, this way inducing too much loss.

Wred(3): Scalable TCP 90% summary



Conclusions(1): summary/results

- With pretty empty fat long pipes, TCP as it is now would invariably fail in terms of high performance throughput.
- New TCP stacks are therefore felt as necessary (Scalable, Hs-TCP,….)

………BUT

- New TCP stacks can affect the performance of traditional BE traffic: IP-QOS is therefore tried out as a solution for traffic protection as well as for service predictability
- In a 1Gbps static IP-QOS-enabled (minimum BW guaranteed) fat long path and with the current state of the TCP Linux kernel/CPU speed, **Standard TCP performs surprisingly better** than HS-TCP and scalable with HS-TCP much closer to standard rather than to Scalable TCP.

Conclusions(2): results

- For high ranges of the CWND (AF-70%), the steeper the CWND slope is (Scalable), the higher is the sack rate and the higher is the probability that the then-stalled sender host negatively affects the job of some part of the code as when Congestion Moderation is on.
- For even higher CWND ranges (AF-90%) the sender <u>**deeply**</u> hangs and seems to reset all to slow start after a period of silence that occurs every time it “hits” the QOS pipe allocated.

Conclusions(3):results

- For high BW allocations there is an inherent higher risk, for aggressive transport protocol like Scalable, of stalling the sender with the effect of reducing drastically the stability and thus the throughput performance.
  - A proven remedy is that of switching off code whose bad performance is worst than not having it in place at all as for with Congestion Moderation.
- For even higher BW allocations, then, the network itself needs to smooth out the loss pattern, this way reducing the sacks rate.
  - the use of a gentle WRED has been proven of extreme efficacy.

THANK YOU……….