Updating TCP to support Rate-Limited Traffic

draft-ietf-tcpm-newcwv-02

G. Fairhurst, A. Sathiaseelan, R. Secchi

IETF 87 - 30 July 2013 - Berlin

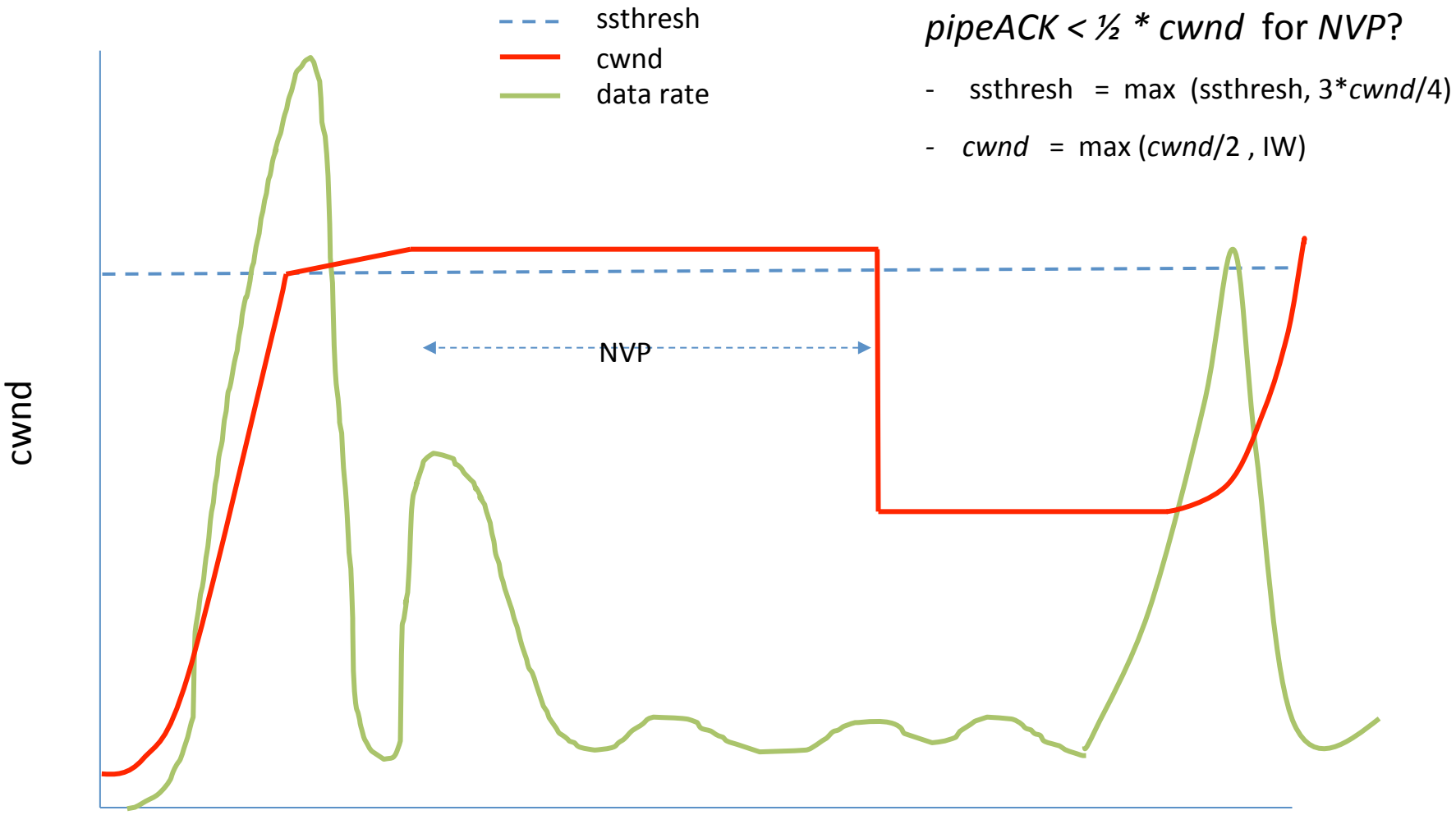
NewCWV (key concepts)

- New-CWV is a replacement for RFC 2861 Congestion Window Validation for CC of rate-limited traffic

- **PipeACK:** Recently used capacity
  - PipeACK is a lower bound for the TCP sustainable rate during data-limited periods
  - PipeACK is calculated from ACKs that acknowledge new data (not the *FlightSize*)

- Basic mechanism
  - While ***cwnd/2 < pipeack < cwnd,*** the cwnd is "validated"
    - Cwnd is increased using normal TCP rules
  - While ***pipeack <= cwnd/2,*** the cwnd is non-validated
    - Cwnd is frozen
  - Different cwnd reduction upon loss in NVP
  - cwnd is halved after 5min of low path utilization

NewCWV behaviour



Changes in draft-ietf-tcpm-newcwv-02

- Clarified the pipeACK calculation
  - ***pipeACK sample*** is the acknowledged data for an RTT – stored for the pipeACK Sampling Period (PSP)
  - ***pipeACK variable*** is measured from one or more pipeACK samples and used to determine the non-validated phase

- Response to congestion takes into account pipeACK
  - When congestion is detected:
    - *cwnd = Min (cwnd/2 , Max (pipeACK variable, LossFlightSize))*
    - Avoids reducing cwnd to very small values when few packets happen to be in flight during recovery
  - At end of the recovery phase:
    - *cwnd = (LossFlightSize - R) /2*
    - *"R" is the number of retransmission (known at the end of recovery)*
    - *Inspired by Almann's Jump-Start*

Status of Implementation & Testing

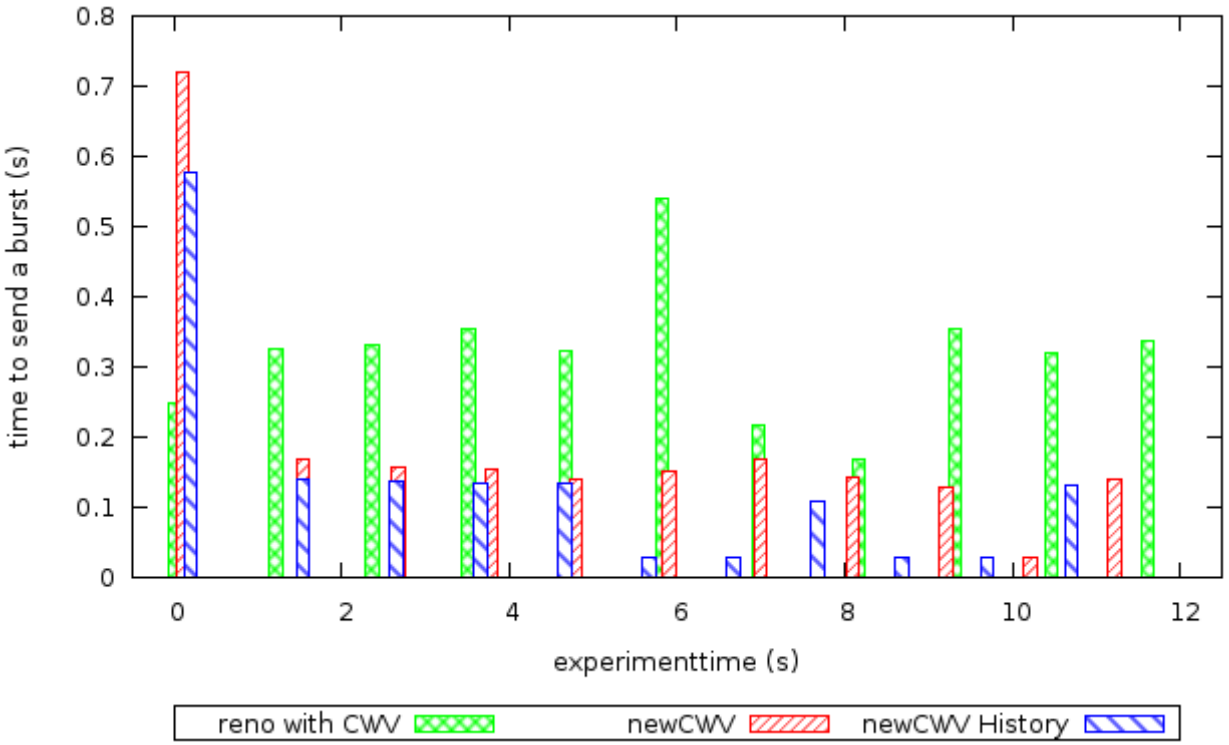
- Implemented NewCWV in a Link Loadable Kernel Module (LKM): *tcp_newcwv.ko*
  - *Uses Linux framework for testing CC algorithms*
  - *Should be compatible with a range of Linux kernels*
  - *Tested with kernel 2.6 and 3.8*

- Implementation proposes a strategy to compute pipeACK
  - Selects the maximum observed pipeACK sample in recent history
  - Updated draft to include example of how to implement

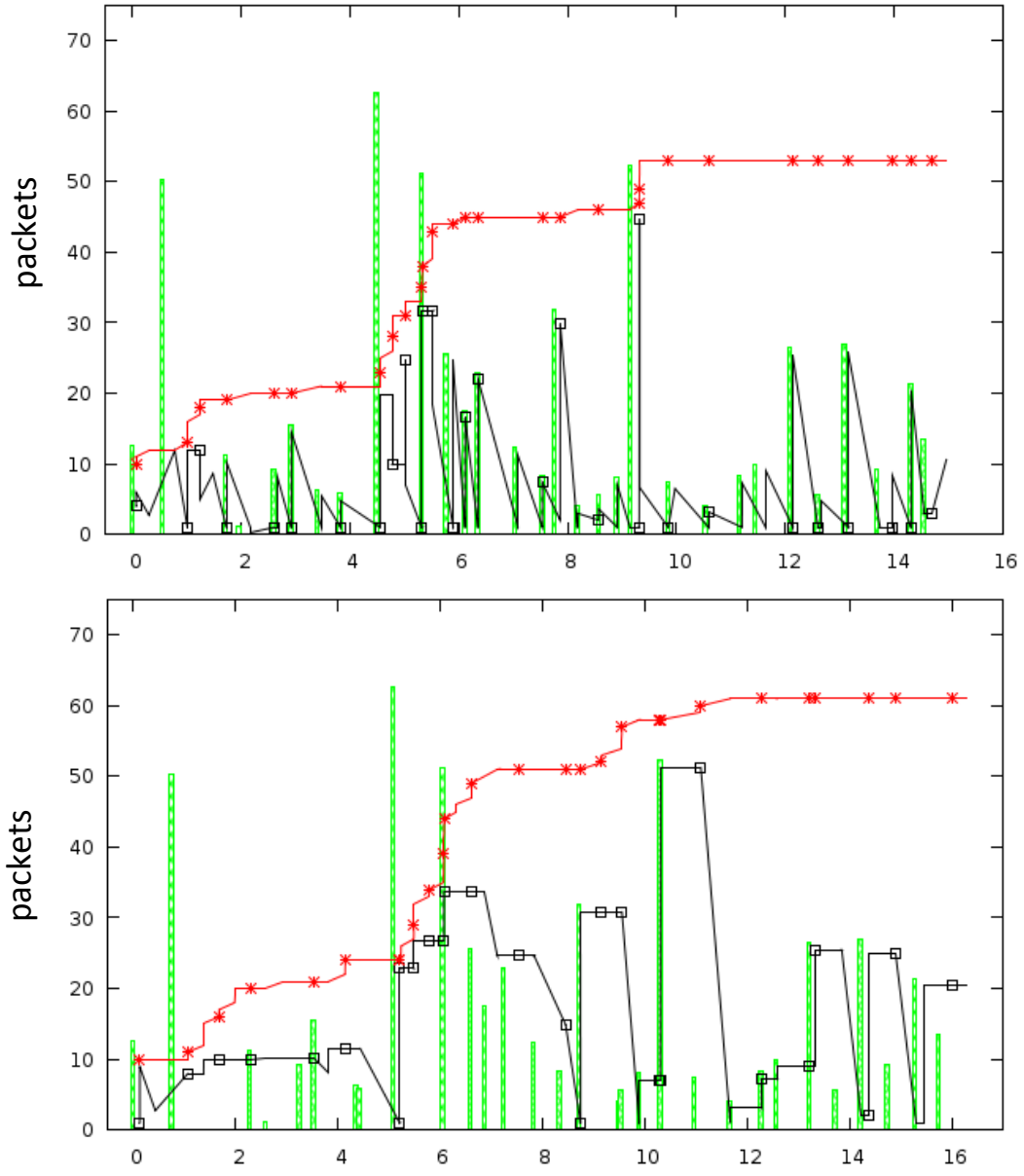
- Maximum Filter
  - Reduces jitter in Flightsize for bursty traffic
  - pipeACK SP determined based on RTT: *max (3 * RTT, 1 sec)*
  - Implementation can be simplified using multiple bins

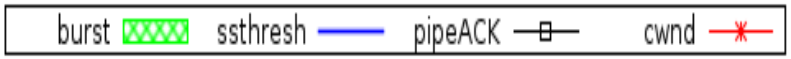
NewCWV benefits applications

- NewCWV tested using virtual hosts in Linux
  - Internet path emulated using Netem (RTT=20ms)
  - Application sends a sequence of bursts (50kB, burst inter-arrival=1sec)
- Bursts using RFC 2861 take longer to transmit in general; the newCWV variants finishes earlier because cwnd not reduced



Stabilizing pipeACK with maximum filter





- *pipeACK variable* set to *pipeACK sample*
- Measured almost every RTT
- For bursty application, highly fluctuating *pipeACK variable*

- *pipeACK variable* calculated from multiple *pipeACK samples* (using maximum filter)
- Retains peak value for PSP
- Stabilizes *pipeACK variable*

Plan to revise draft

- Michael Scharf: After finishing recovery,

  cwnd = (LossFlightSize - R) / 2

  - Can 'R' be larger than LossFlightSize?
    - Not encountered such a case
    - However happy to add that cwnd is always >= 1 MSS.

  - Why is cwnd set differently at start and end of recovery?
    - Beginning and end of recovery represent upper and lower bounds of safe values
    - We'll try simulations to see if we can improve calculation.

- End of burst losses
  - Can *max (pipeACKsample , LossFlightSize)* resolve this?

- Now looking for people to experiment with!