Extending Equation-based Congestion Control to Multicast Applications

Joerg Widmer
University of Mannheim
widmer@informatik.uni-mannheim.de

Mark Handley
AT&T Center for Internet Research at ICSI
mjh@aciri.org

ACM SIGCOMM 2001

Overview of TFMCC

From TFRC to TCP-friendly Multicast Congestion Control (TFMCC)

- Equation-based

  model TCP throughput based on RTT and loss rate

- TCP-friendliness

  no greater medium-term throughput than TCP to *any* of the receivers

- Single-rate congestion control scheme for single-source multicast

  adapt the rate of the sender to the slowest receiver

- No router support required

So it should work in today's Internet.

TFRC in a Nutshell

- Receiver measures RTT and loss rate ...
- ... and calculates a TCP-friendly rate using

$$T_{TCP} = \frac{s}{t_{RTT}\left(\sqrt{\frac{2p}{3}} + \left(12\sqrt{\frac{3p}{8}}\right) p\left(1+32p^2\right)\right)}$$

with s = packet size, t_{RTT} = RTT, p = loss event rate

- Receiver reports rate to sender who in turn adjusts its sending rate

The measurement of the two parameters RTT and loss rate is critical.

Challenges for Multicast

Each TFMCC receiver has to individually determine a TCP-friendly rate.

Challenges:

- Scalable RTT measurements to a large number of receivers (without synchronized clocks)
- Scalable feedback mechanism
  - Prevent feedback implosion
  - Get feedback from receiver(s) with the lowest rate

But:

- Adjusting the sending rate is fairly staight-forward
- The loss measurement mechanism can be directly taken from TFRC

Adjusting the Sending Rate

Sending rate determined by the receiver that is assumed to have the lowest calculated rate

- Whenever lower rate feedback is received the sending rate is adjusted accordingly

How does the rate increase?

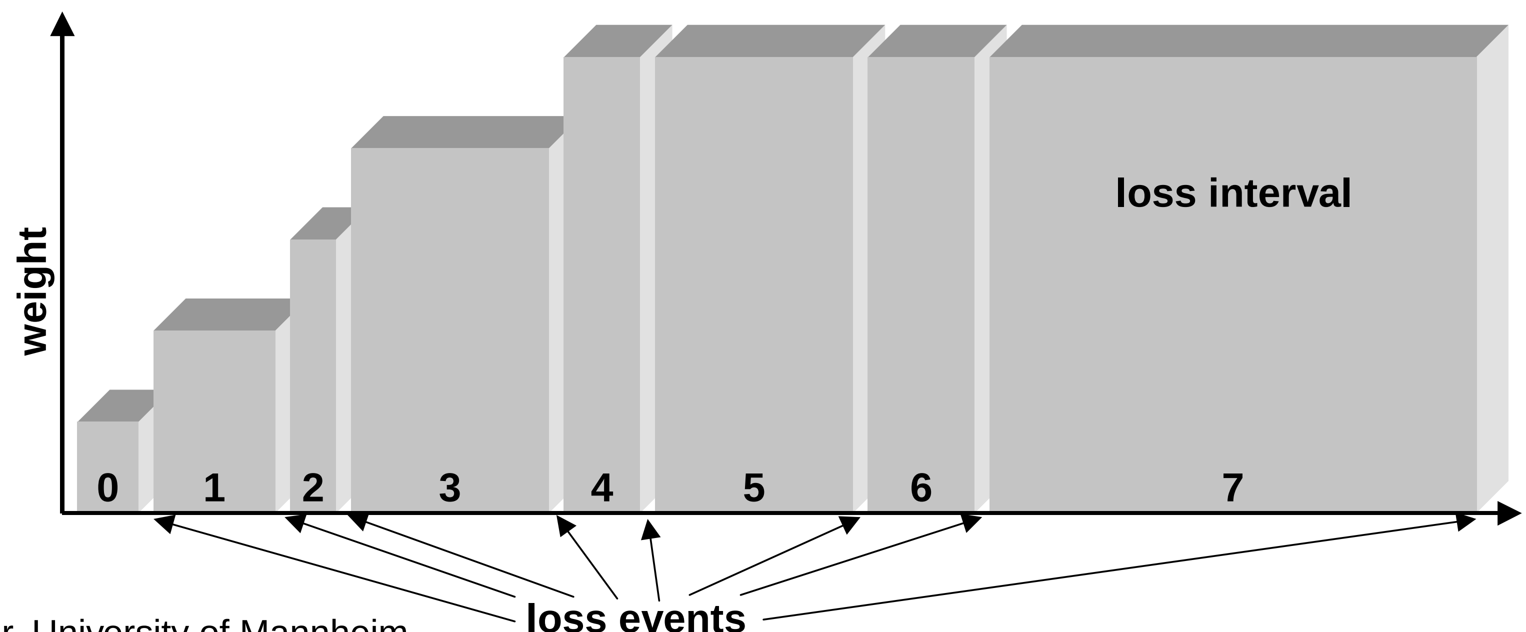
- Concept of the *current limiting receiver* (CLR)
- CLR always gives feedback irrespective of the rate $\Longrightarrow$ CLR can cause a rate increase
- Time out CLR if no feedback was received for some time

- Additionally limit rate increase to 1 packet/RTT^2

Measuring the Loss Event Rate

Possible to reuse TFRC's loss measurement mechanism without any modifications

- Loss interval: number of packets between loss events
- Compute weighted average of *n* loss intervals
- Inverse of this average serves as an estimate of the loss event rate



RTT Measurements

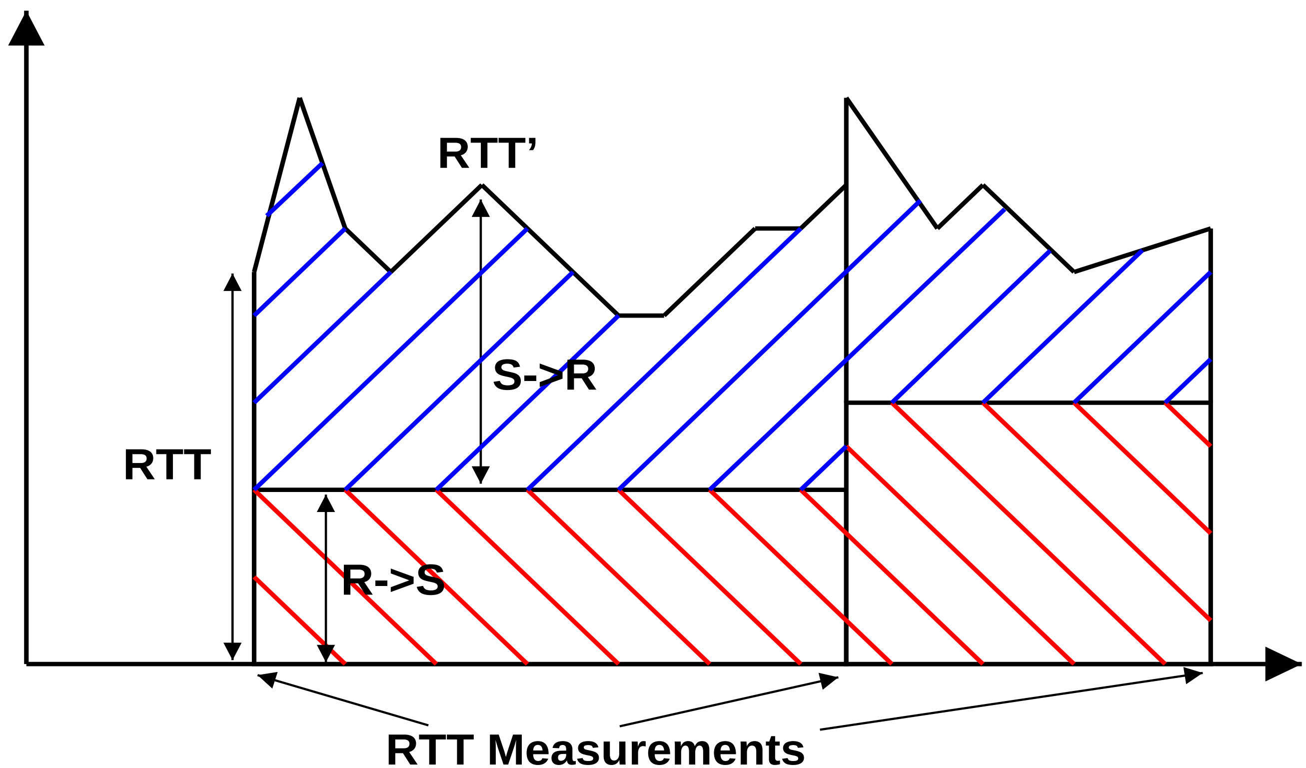
- Well-known RTT measurement mechanism of echoing timestamps

$$t_{RTT} = t_{now} - t_{echo}$$

- Priority list of which timestamps to echo in data packets
  1. CLR directly after change of CLR
  2. Receivers without valid RTT measurement
  3. Non-CLR receivers
  4. CLR
- Additional smoothing (EWMA) to be insensitive to short-term RTT variations
- Assume a high initial RTT until the first measurement is made (e.g. 500ms)

RTT Measurements (cont.)

Infrequent RTT measurements for non-CLR receivers
$\Longrightarrow$ continuously adjust RTT using one-way delay measurements

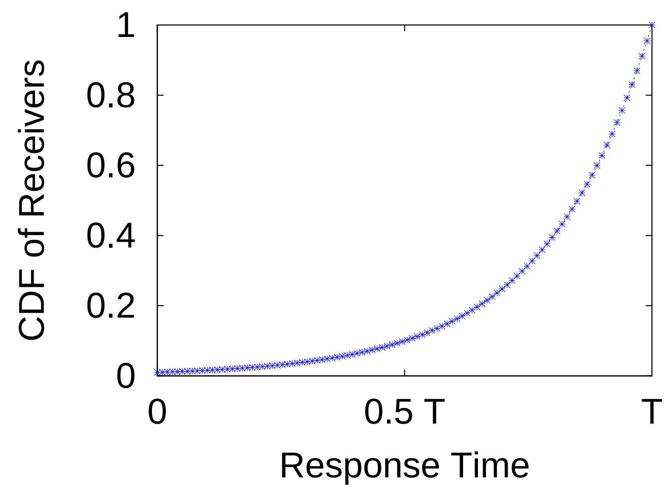


- RTT' can then be used to detect changes in the RTT

Feedback Control

- Only receivers with a lower than the CLR's rate get to send feedback
- Use exponentially distributed feedback timers

$$t = \max\left(T(1 + \log_N x), 0\right)$$



with T = max. feedback delay, N = upper bound on the number of receivers, x = uniformly distribued random variable

- Cancel timers of receivers that are notified of other receivers' feedback

Improving Feedback

- Biased feedback timers
- Modifi ed suppression mechanism

Feedback Bias

Bias feedback timers such that low rate feedback is sent earlier

$$t = \max(T(1 + \log_N x), 0)$$

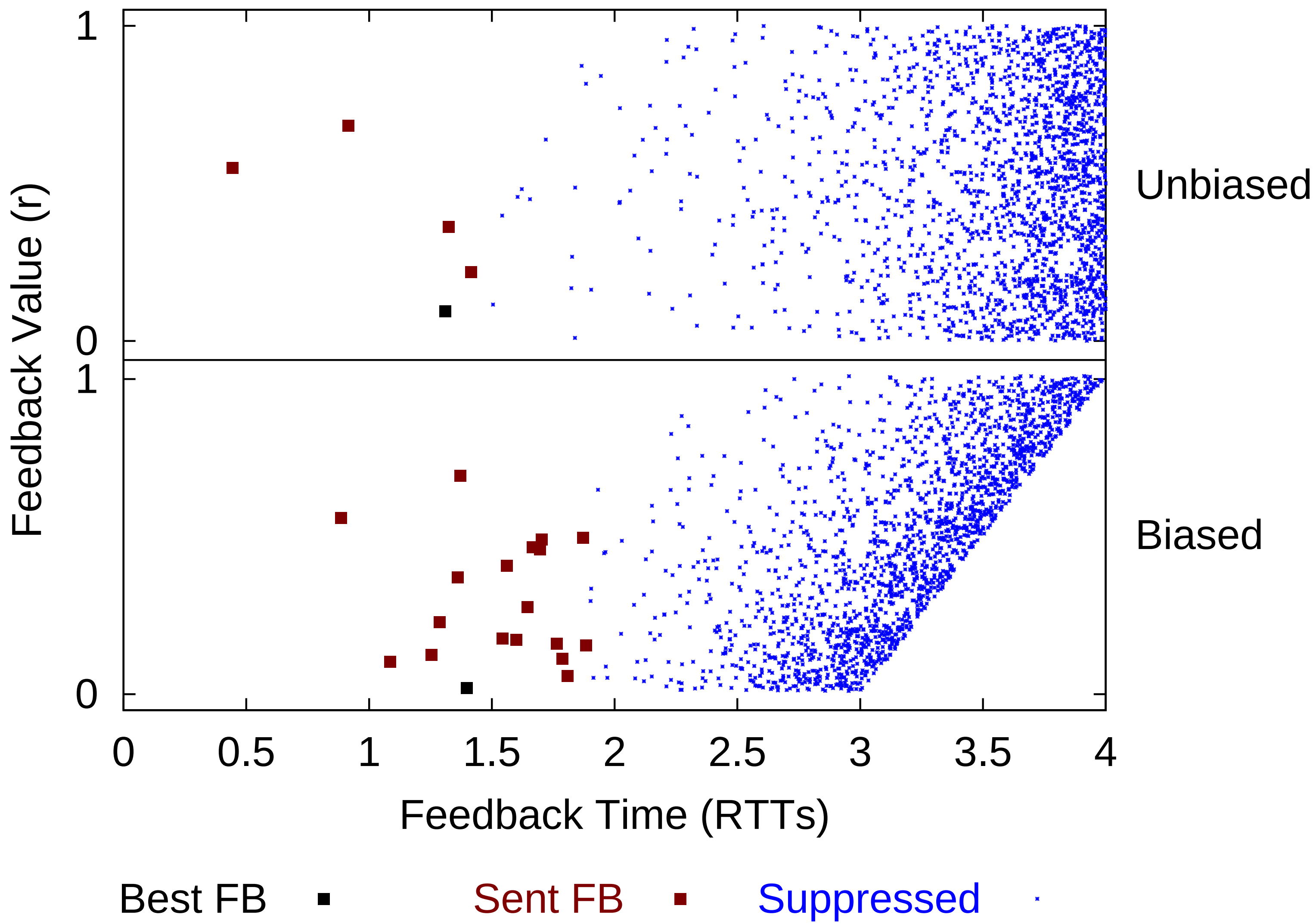
$$\Downarrow$$

$$t = \gamma \max(T(1 + \log_N x), 0) + (1 - \gamma) T r$$

where

- r is the calculated rate relative to the CLR's rate
- γ is the fraction of T now used for suppression

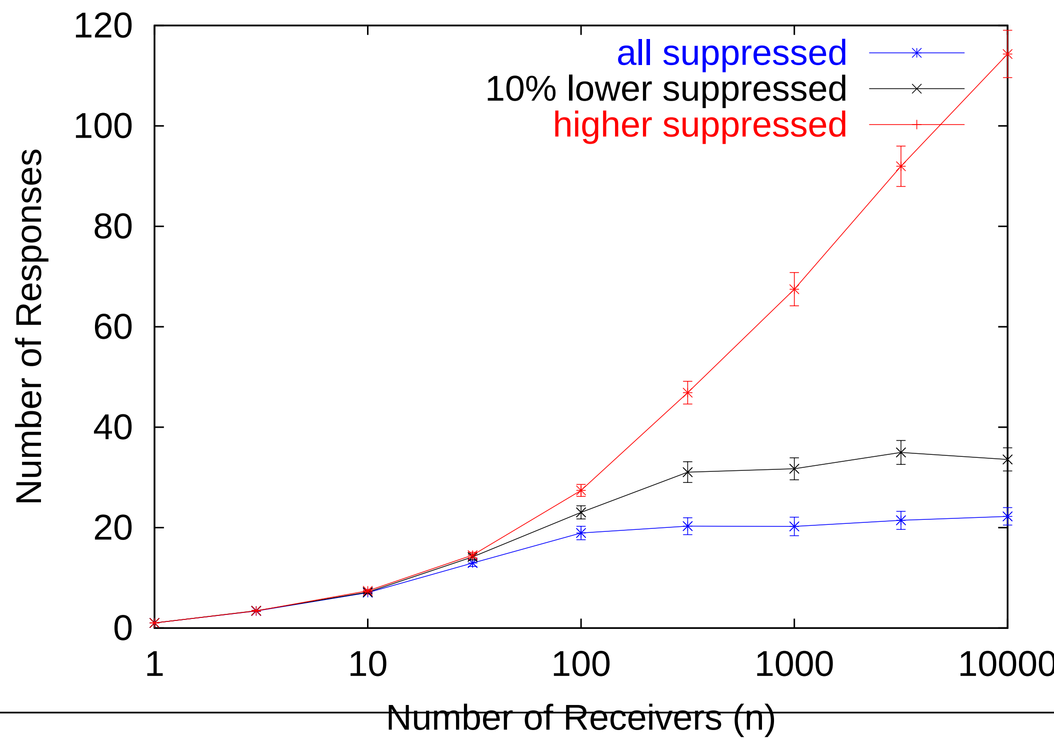
Feedback Bias



Feedback Suppression

Options what feedback to cancel:

- Cancel timer if any feedback was received
- Cancel timer if “better” feedback was received
- Cancel timer if $T_{fb} - T_{TCP} < \theta\, T_{fb}$

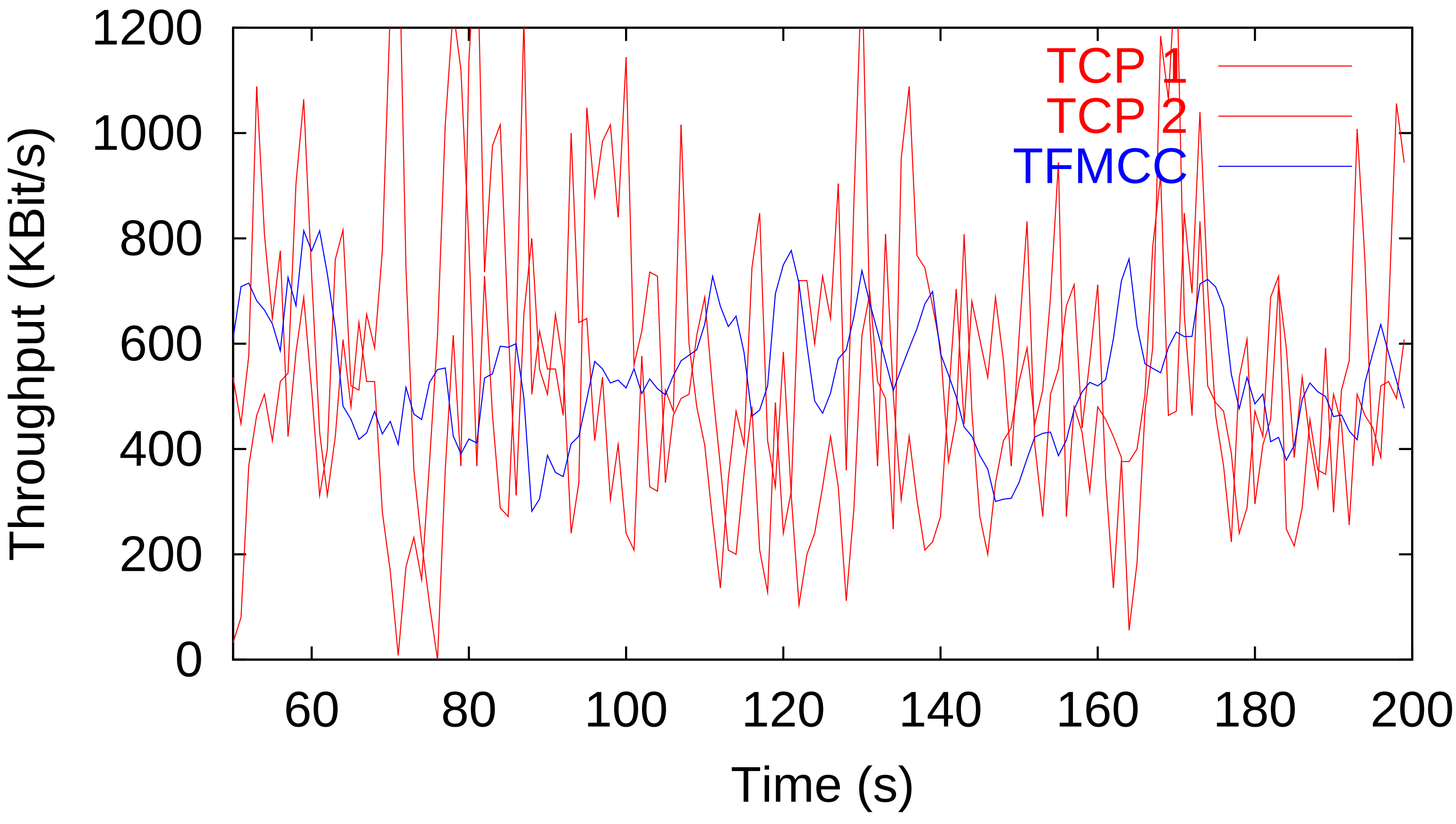


Simulations

Some examples of TFMCC simulations

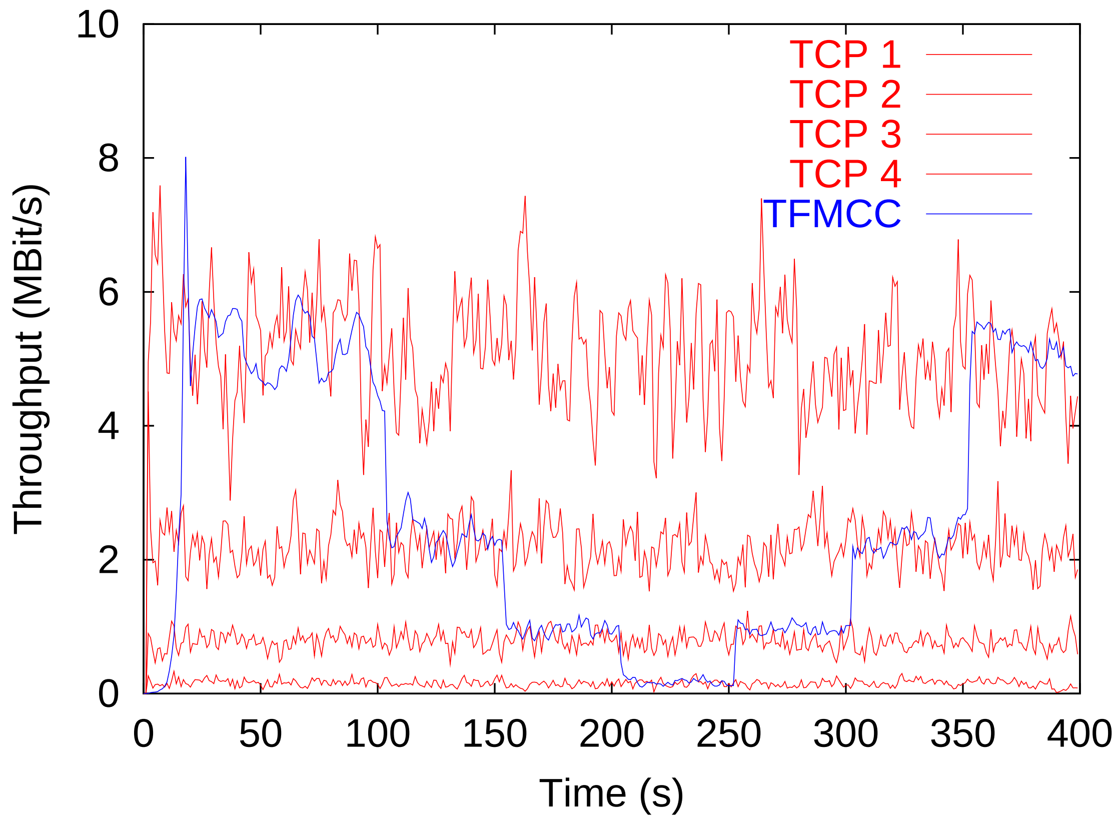
Fairness

One TFMCC flow and 15 TCP flows over a single 8 MBit/s bottleneck with 60ms RTT



Responsiveness

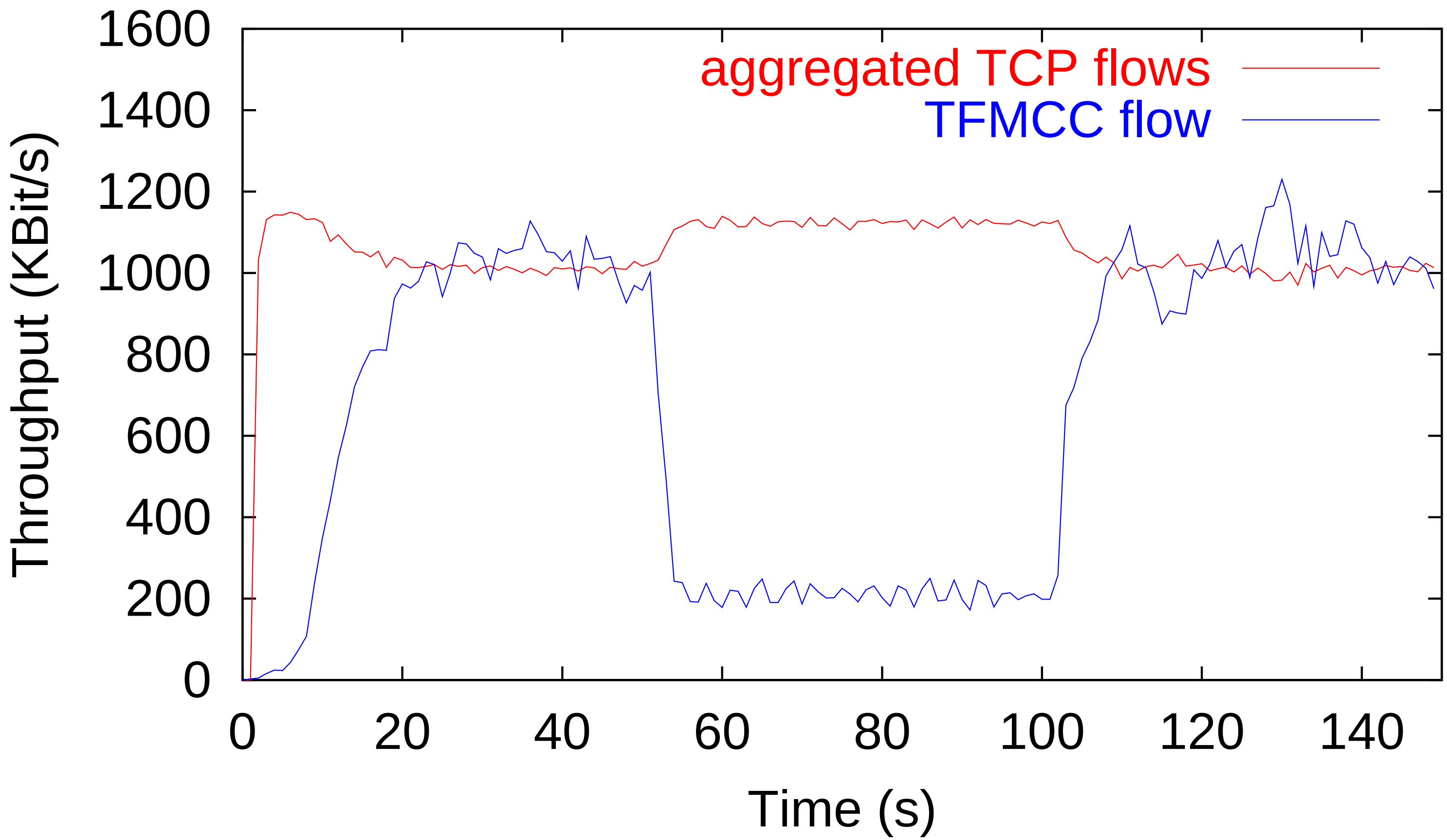
Responsiveness to changes in the loss rate
(60ms RTT and loss rates of 0.1%, 0.5%, 2.5%, and 12.5%)



Correct CLR chosen after ca. 500ms

Late-Join of Low-Rate Receiver

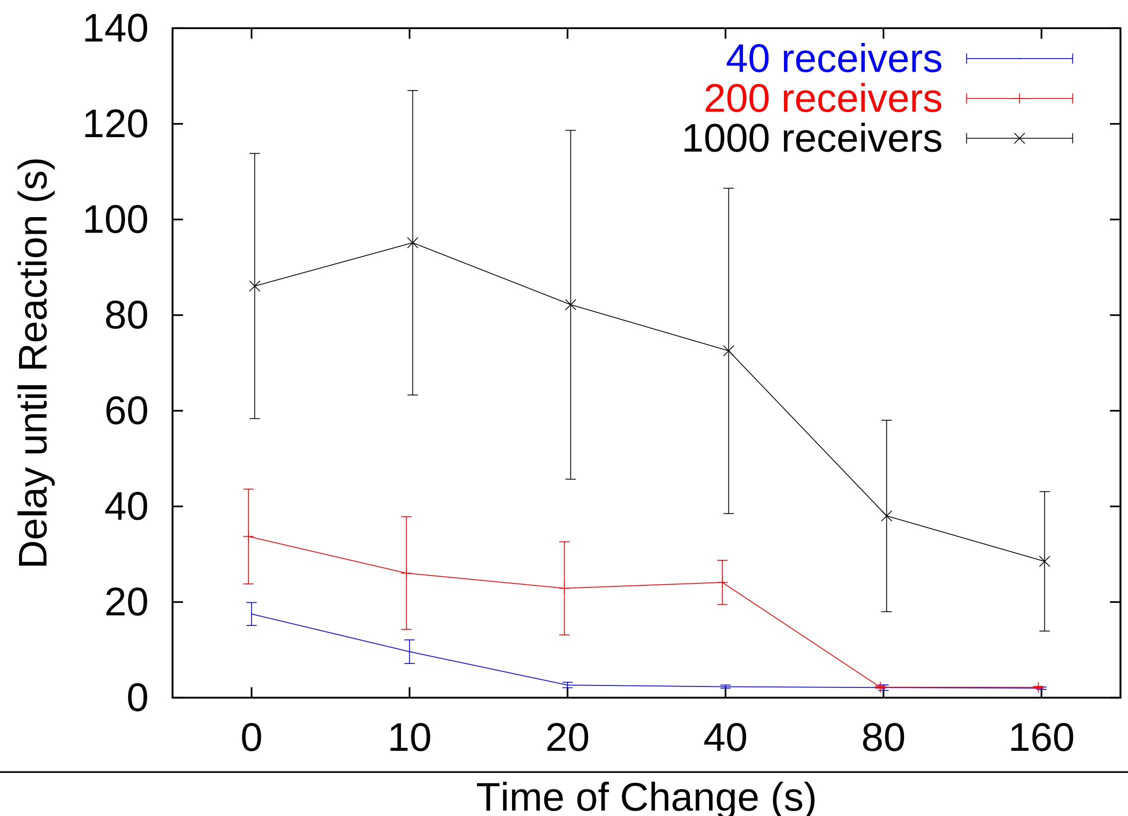
- TFMCC competing with 7 TCPs on 8MBit/s link
- TFMCC receiver 200KBit/s link joins for 50 seconds



RTT Responsiveness

Responsiveness to changes in the RTT (worst case analysis)

- How long does it take to find a single high RTT receiver among a large number of low RTT receivers?



Conclusions

Results look quite promising so far. We've got a working implementation of TFMCC in the *ns* simulator that performs well under a wide range of network conditions.

Future Work:

- Need implementation (currently being done) as well as real-world tests
- Work on variable packet size TFRC/TFMCC
- Ongoing work on feedback control
  - Estimate distribution of calculated rates
    $\Longrightarrow$ Support application level access control
  - Track changes in the distribution