# *TFMCC Update*

Mark Handley
*mjh@aciri.org*

# *TFMCC Overview*

- **TFMCC = TCP Friendly Multicast Congestion Control**
  - sender-orientated single rate MCC
- **TFRC = TCP Friendly Rate Control**
  - unicast version of TFMCC
- **Use equation modelling TCP throughput as a function of loss rate and RTT directly in a congestion control mechanism.**
- **Hard parts:**
  - Good filter design for measuring loss.
  - Scalable way to measure RTT in multicast case.
  - Dynamics in response to changes in RTT

# *TFRC*

- **Unicast version basically complete.**
  - Extensive simulation, dummynet experiments, and real world experiments show it to work pretty well.
  - Especially since a few bugs have been fixed in the simulation code.
- **Sigcomm 2000 Paper, plus code:**
  - http://www.aciri.org/tfrc/
- **Still ongoing fine tuning, but no showstoppers.**
- **Probably submit a draft by next IETF, to become an Informational or Experimental RFC aimed at congestion control for unicast streaming media.**

# *TFMCC*

- **Multicast experiments really just beginning.**
- **Joerg Widmer (University of Mannheim) will be at ACIRI for next two months to conduct experiments.**
- **TFRC filter design for measuring loss should be directly applicable.**
- **RTT measurement needs experimentation.**

# *TFMCC for Nack-based Protocols*

As discussed in November:

- **Receivers start by assuming 500ms RTT in rate calculation.**
- **Use randomized timers to delay sending**
  - Weight timers based on calculated sending rate.
  - Low rate -> low timer, high rate -> high timer
  - Suppress if you see someone else reporting a lower rate than you.
- **Include timestamp from data message in response.**
- **Sender measures RTT and echoes it in a later data packet.**
  - This allows the receiver that responded to know its RTT.
- **Eventually all the congested receivers have measured their RTT, and TFMCC gets to go at a reasonable speed.**
- **Optimizations are possible if NTP is used for clock sync.**

# *TFMCC for Tree-based Protocols*

- **In TFRC receivers measure loss and RTT, calculate rate, and feed this back.**
- **There's a hybrid rate-window variant where the RTT measurement is implicit rate than explicit.**

# *TFMCC for Tree-based Protocols*

Rate version:

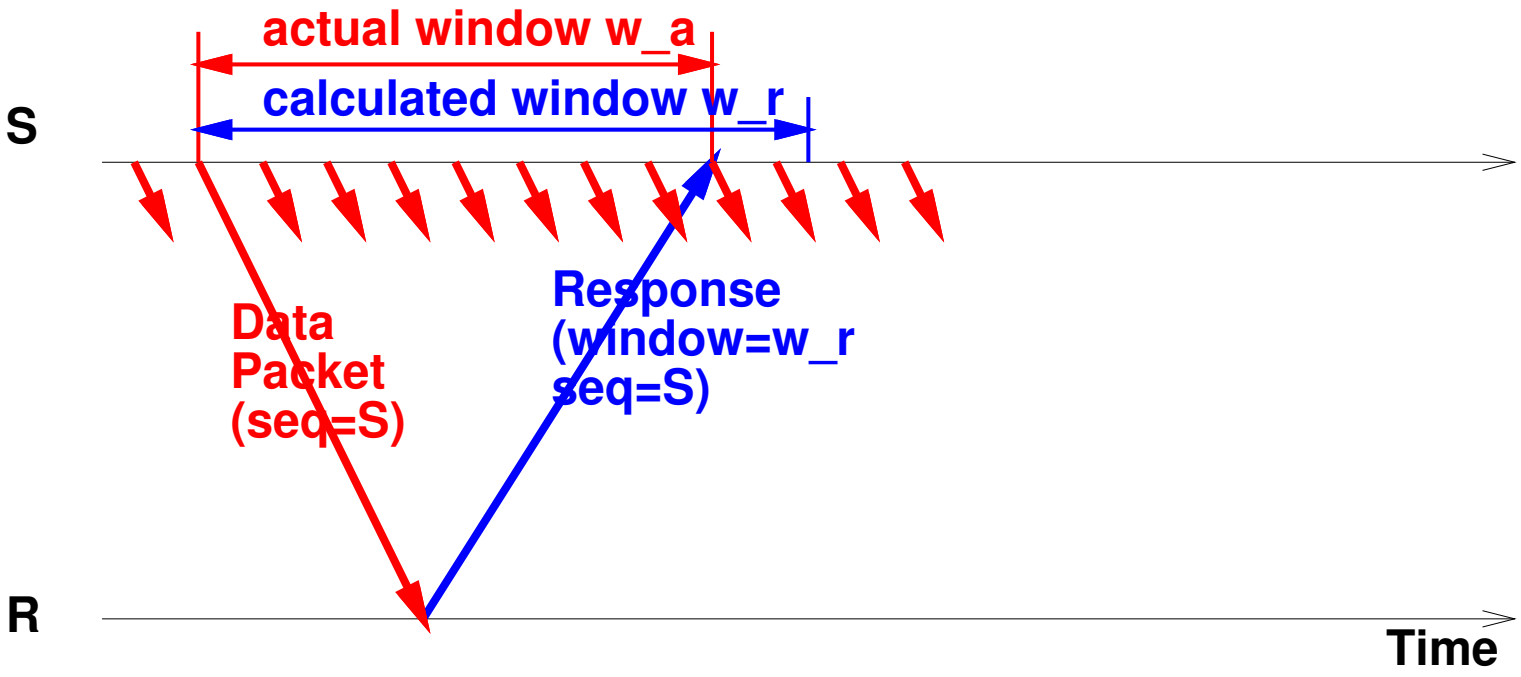
$$R = \frac{s}{t_{RTT}\sqrt{\frac{2bp}{3}} + 4t_{RTT}min\left(1,3\sqrt{\frac{3bp}{8}}\right)l(1+32p^2)}$$

Window version:

$$W = \frac{1}{\sqrt{\frac{2bp}{3}} + 4min\left(1,3\sqrt{\frac{3bp}{8}}\right)l(1+32p^2)}$$

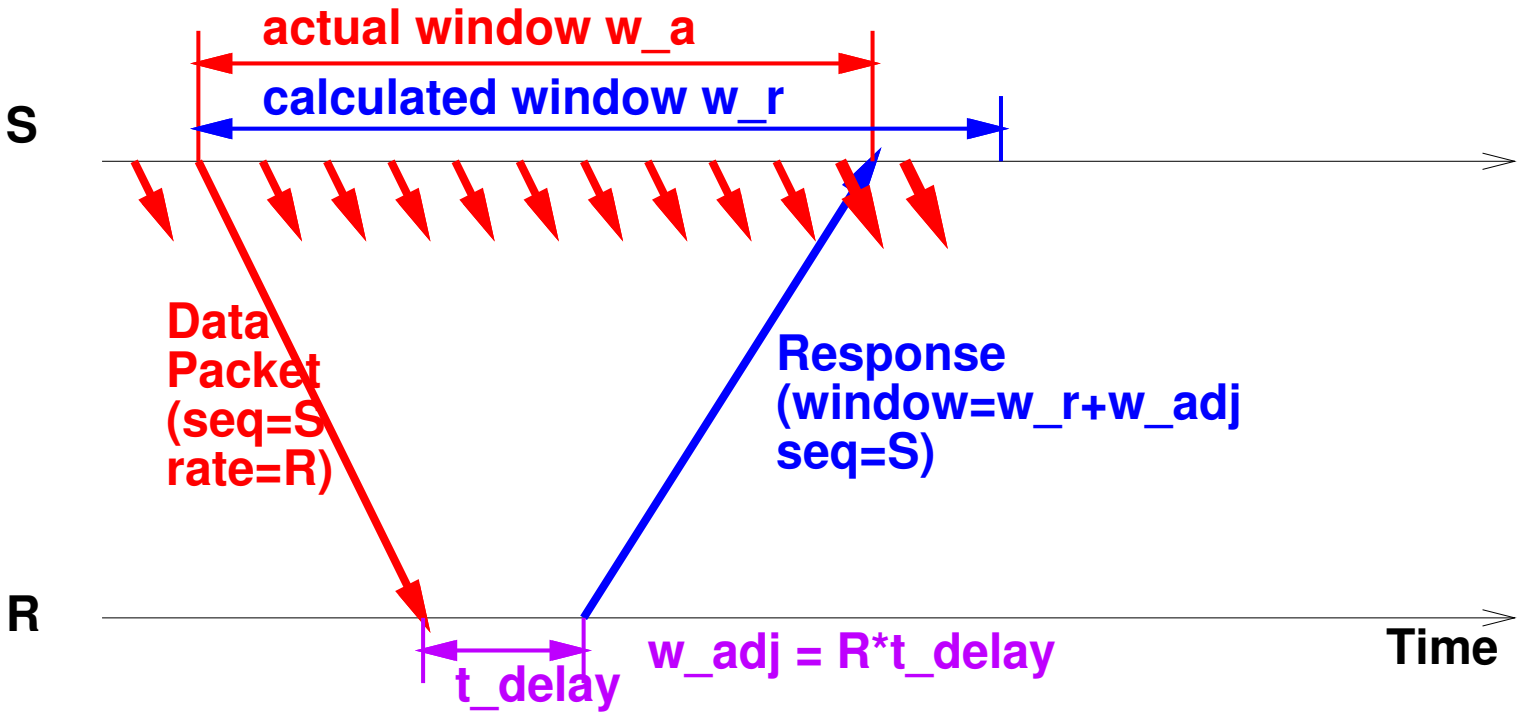
R=rate, b=pkts/ack, p=loss probability, s=pkt size

# *Rate/Window-based Unicast TFRC*



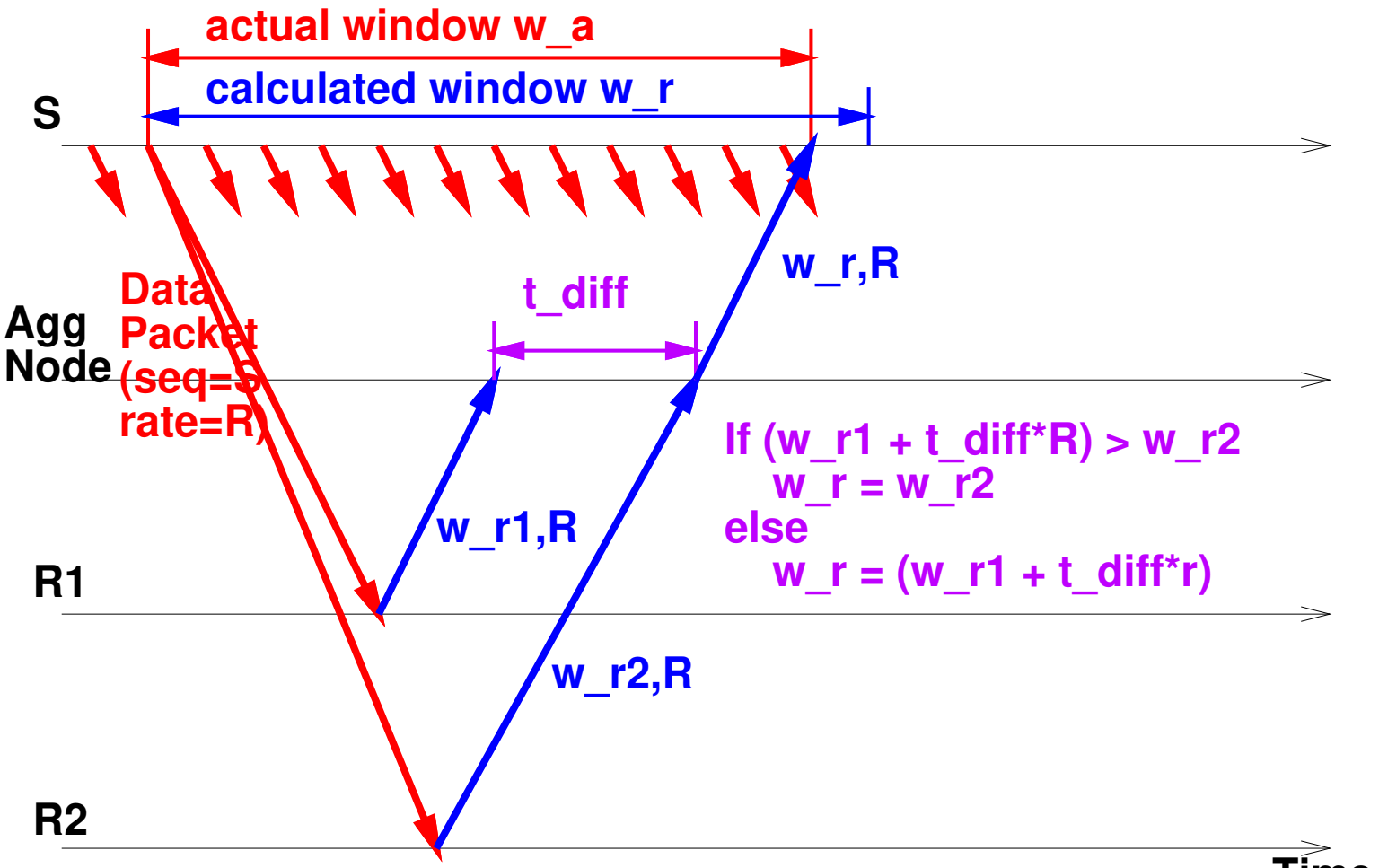

**w_a < w_r : sender can send faster**
**w_a > w_r : sender must send slower**

# Rate/Window-based Unicast TFRC (2)




w_a < w_r+w_adj : sender can send faster
w_a > w_r+w_adj : sender must send slower

# Rate/Window-based TFMCC




w_a < w_r : sender can send faster
w_a > w_r : sender must send slower

# *Summary*

- **TFRC is ready for standardization.**
- **TFMCC isn’t yet ready for standardization.**
- **Experiments this summer should confirm whether the design has any unforeseen problems.**
  - Tree-based TFMCC is easier than NAK based TFMCC.
  - We plan to investigate both.
- **Re-appraise situation before December IETF.**