Doing away with TCP-friendliness: is there a realistic option?

Panelists:

Michael Welzl (UiO, Norway)
Matt Mathis (PSC, USA)
Bob Briscoe (BT, GB)
Kevin Mills (NIST, USA)
Michio Honda (Keio University, Japan)

Should / could we find a new metric?

Should: I think so

- TCP-friendliness quite easy to understand
  - Requirement "make your protocol TCP-friendly" influenced the research landscape: papers saying "our protocol has some advantage over others (e.g. a smoother rate), and it's even more (or quite, …) TCP-friendly"
  - Several TCP-friendly protocols were proposed: RAP, TFRC, TEAR, LDA+, …
- Why was this possible?
  - I think that the equation helped
  - Easy to check TCP-friendliness

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p\left(1+32p^2\right)}$$

s: packet size
R: rtt
t_{RTO}: TCP retransmit timeout
p: steady-state loss event rate

- Now, we have a more relaxed requirement:
  - Be more aggressive when packet loss is low
  - TCP-friendly otherwise ⇒ congestion collapse won't happen
- Protocols such as HighSpeed TCP, BIC, CUBIC, CTCP … behave this way
  - how do they interact? (didn't look good in our PfldnetŠ07 paper)
  - is this vague fairness notion enough?

Could: I think so

- Consider *n* TCP‘s:
  - low packet loss: more aggressive than a single TCP
  - high packet loss: they all back off
  - we know that *n* TCP‘s don‘t kill the Internet
- So why not prescribe *n*-TCP-friendliness?
- But how to choose the value of *n*?
  - 100?
  - Maybe ∞ ?
- Could we have an equation, just like the original steady-state-model, but for *n*-TCP-friendliness?

Good news: we made such an equation

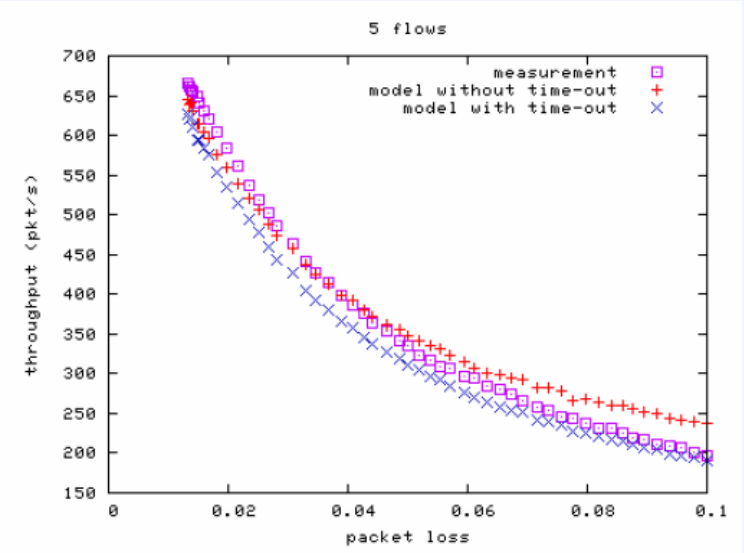
- Dragana Damjanovic, Michael Welzl, Miklos Telek, Werner Heiss: "Extending the TCP Steady-State Throughput Equation for Parallel TCP Flows", University of Innsbruck, Institute of Computer Science, DPS NSG Technical Report 2, August 2008.
  - available from http://www.welzl.at/research/publications/all.html
  - also SIGCOMM'07 poster, 2-page text available from the same page
  - referenced and applied in our upcoming CCR paper on MulTFRC

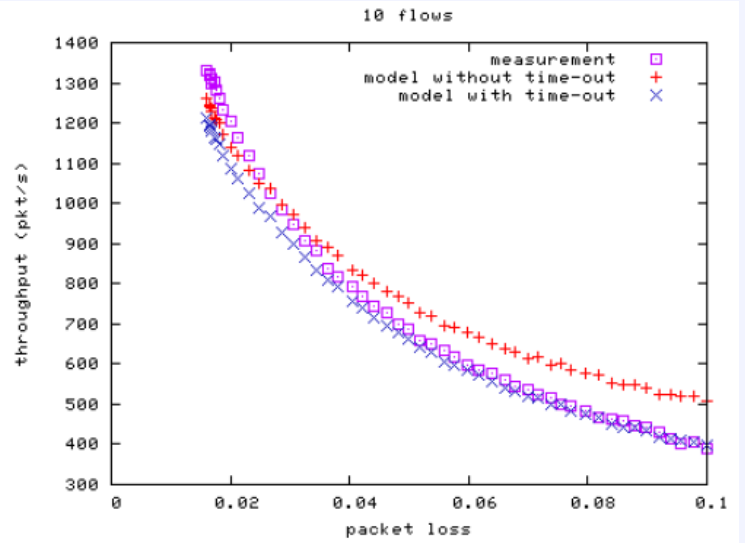
- Best readable in algorithm form
  - **n** - number of flows
  - **b** – no. of packets ACKed by one ACK
  - **RTT** - round-trip time
  - **T** - initial period of time (in TO phase) after which the sender retransmits unacknowledged packets
  - $\mathbf{p_e}$ - loss event probability of the cumulative flow
  - $\mathbf{p_r}$ - probability that a packet is lost

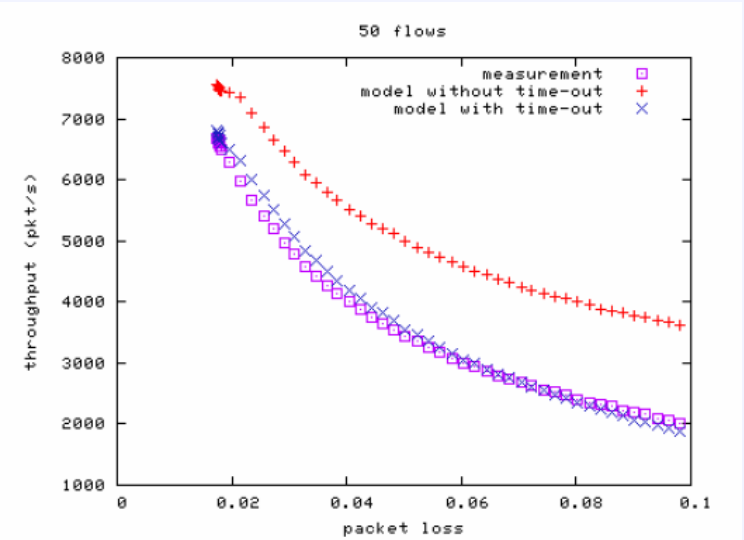
Require: n, p_e, p_r, b, RTT, T.
Ensure: The throughput B.

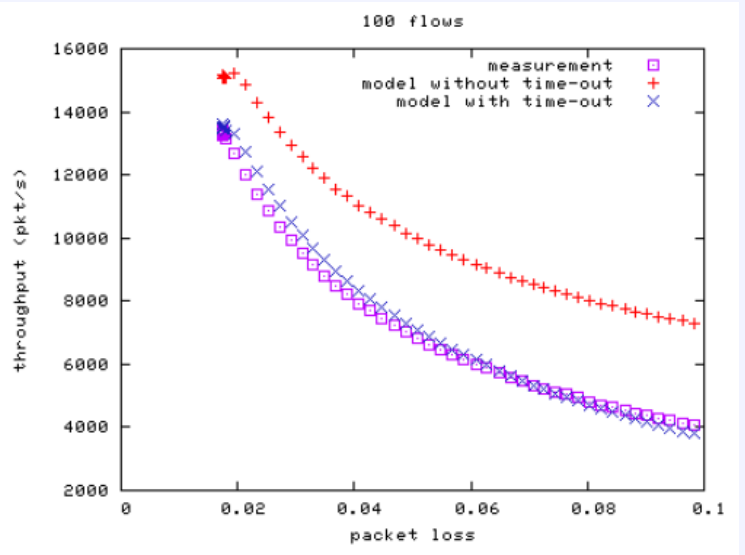
$j = p_r/p_e$
$j1 = j$
if $j1 > n$ **then**
  $j1 = n$
end if
$a = sqrt(p_e*b*j1*(24*n*n+p_e*b*j1*(n*n-4*n*j1+4*j1*j1))$
$x = (j1 * p_e * b * (2 * j1 - n) + a)/(6 * n * n * p_e)$
$w = n * x/(2 * b)(1 + 3 * n/j1)$
$z = T * (1 + 32 * p_e * p_e)/(1 - p_e)$
$q1 = j * n/w$
if $q1 > 1$ **then**
  $q1 = 1$
end if
if $q1 * z/(x * RTT) \geq n$ **then**
  $q = n$
else
  $q = q1 * z/(x * RTT)$
end if
return $(1 - q/n)/(p_e * x * RTT) + q/(z * (1 - p_e))$

Validation / how do *n* TCPs perform?









Bad news: can‘t do $n \to \infty$

- Just doesn‘t work
- Our equation was designed to be simple
  - There are more sophisticated models; maybe one of them becomes simple and useful when $n \to \infty$?
- Diagrams show little difference between 50 flows and 100 flows
  - So should we use ours, and just pick a large value for n?
  - I don‘t like this…
- How to use it?
  - really as an upper limit, like TCP-friendliness?
  - then, it would make sense to check if current TCPs are below our line
- What do you think?