General AIMD Congestion Control

Y. Richard Yang and Simon S. Lam

Motivation for new congestion control protocols

- Many new apps (e.g. multimedia) use UDP instead of TCP because they do not require reliable delivery
- Reducing *cwnd* to half of its value after a loss indication is too severe a reduction for some real-time apps (e.g., interactive multimedia)
- Increasing use of UDP without congestion control would threaten stability of Internet

  -> Need new CC protocols for apps that prefer an alternative to TCP

TCP-friendly protocols

- Alternatives to TCP congestion control with smaller send rate fluctuations
  - Equation-based rate control [9, 21]
  - Datagram Congestion Control Protocol (DCCP)
  - GAIMD in this paper
- TCP-friendliness to better co-exist with TCP traffic
  - The send rate of a non-TCP flow should be approximately the same as that of a TCP flow under the same conditions of round-trip time and loss rate

GAIMD

- Consider a more general version of AIMD; let $\alpha > 0$ and $1 > \beta > 0$, b denote number of packets acknowledged by each ack

For each new ack received, $W \leftarrow W + \dfrac{\alpha}{bW}$

For a TD ack, $W \leftarrow \beta W$

For a timeout, $W \leftarrow 1$

- Other mechanisms (Slow Start, congestion indications, and round-trip time estimation) are the same as those of TCP Reno

GAIMD send rate

$$\text{send rate} = T_{\alpha,\beta}(p, RTT, T_0, b)$$

$$= \frac{1}{RTT\left(\sqrt{\frac{2b(1-\beta)p}{\alpha(1+\beta)}}\right) + \min\left(1, 3\sqrt{\frac{(1-\beta^2)bp}{2\alpha}}\right) p(1+32p^2)T_0}$$

- Same model and assumptions as Padhye et al.
  - p : loss rate
  - RTT: mean round-trip time
  - T_0: mean timeout value
- Reduces to previous formula with α = 1 and β = ½
- Send rate decreases with a larger RTT, larger T_0, or larger b
- Send rate increases as β increases to 1 or as α increases from 0

Interpreting the send rate formula

- Denominator is sum of the following 2 terms

$$TD_{\alpha,\beta}(p, RTT, b) = RTT\left(\sqrt{\frac{2b(1-\beta)p}{\alpha(1+\beta)}}\right)$$

$$TO_{\alpha,\beta}(p, T_0, b) = Qp(1+32p^2)T_0$$

$$\text{where } Q = \min\left(1, 3\sqrt{\frac{(1-\beta^2)bp}{2\alpha}}\right)$$

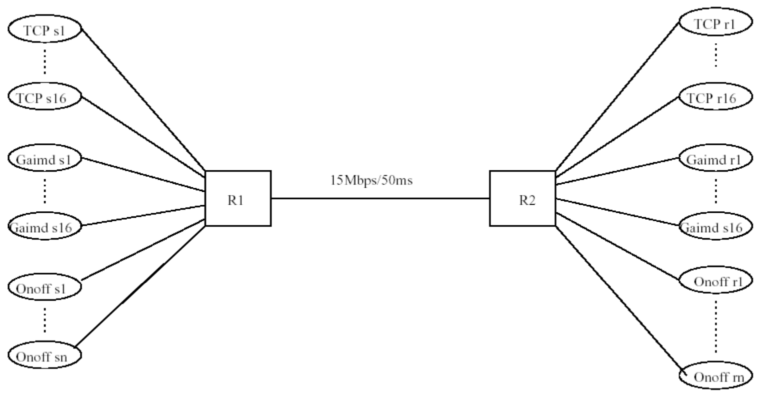
- Q, probability of a loss being a TO, increases toward 1 as p increases
- For a small p, $TD = O(p^{0.5})$ dominates $TO = O(p^{1.5})$

Formula validation

- Is the formula accurate? Over what range of loss rate *p* is it accurate?
- When do sending rate variations become significant?
- What is the general trend when the formula loses accuracy?

Simulation setup

16 TCP Reno flows, 16 GAIMD flows, and flows with ON/OFF times to model web-like traffic (UDP flows and short TCP flows)



- Mean ON time = 1 s, mean OFF time = 2 s, Pareto distribution
- During ON time, each source sends 500 Kbps

Prediction accuracy

- Measure of accuracy:
  - predicted sending rate/actual (ave.) sending rate
- Validity range of the formula
  - For each β, vary α from 0.1 to 1.0
  - For each (α, β), vary the number of ON/OFF flows from 10 to 70 to create a loss rate about 1% to 30%
- Impact of loss pattern on the accuracy of the formula
  - Used different kinds of routers: drop-tail and RED

Accuracy (1)

prediction/measurement

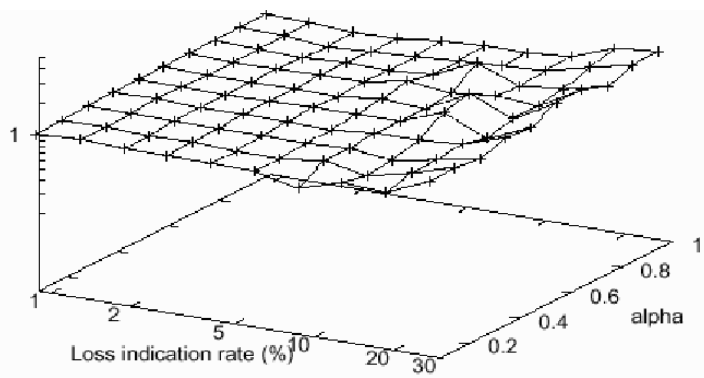


Figure 2: Accuracy for $\beta = 0.5$ and drop-tail

Accuracy (2)

prediction/measurement

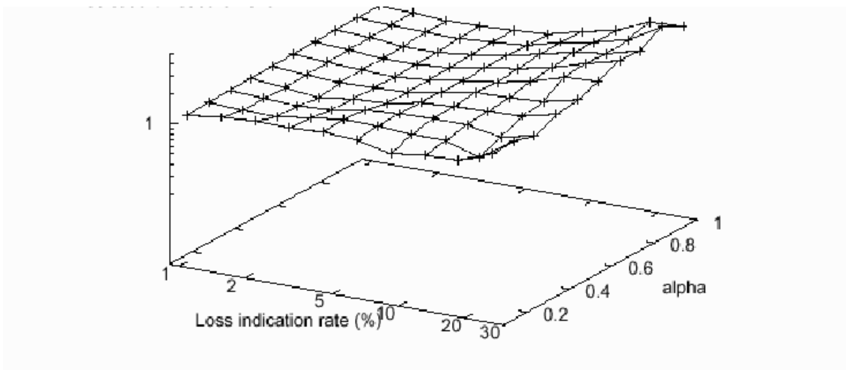


Figure 4: Accuracy for $\beta = 0.875$ and drop-tail

- ❐ Formula good for loss rate less than 20%

Accuracy (3)

prediction/measurement

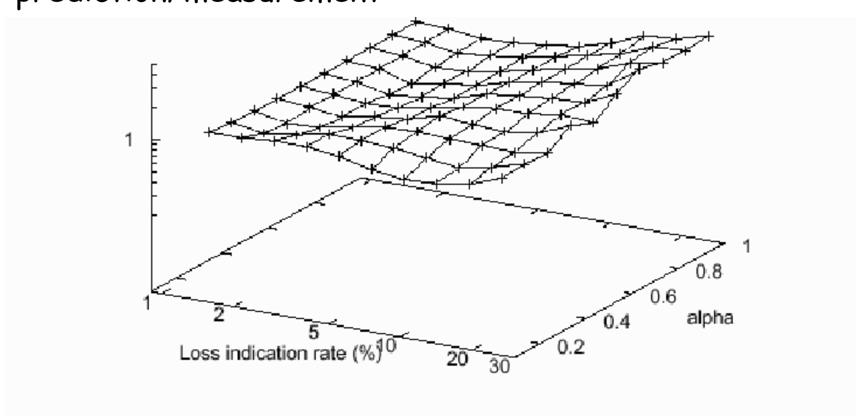
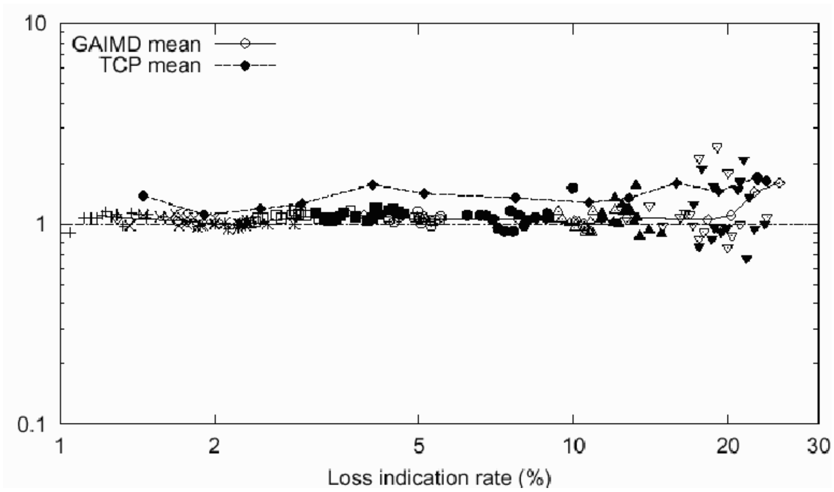


Figure 5: Accuracy for $\beta = 0.875$ and RED

RED router may not satisfy correlated loss assumption

Sending Rate Variation (1)

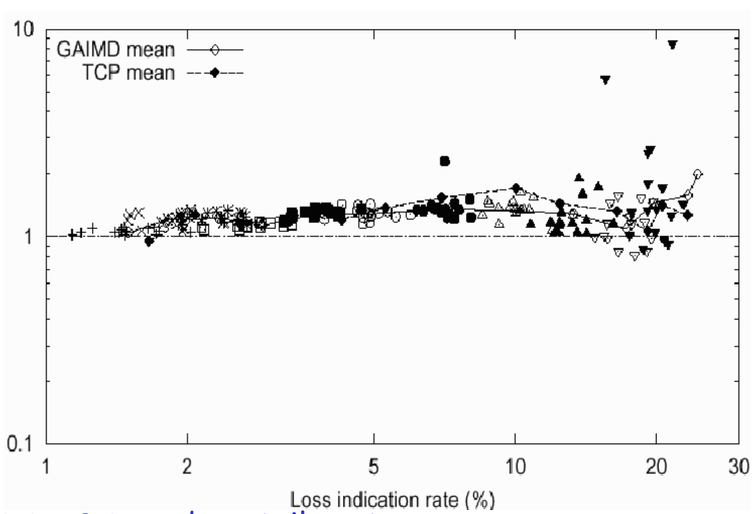
accuracy for individual GAIMD flows and TCP flows



α=0.5, β=0.5, drop-tail router

Sending Rate Variation (2)

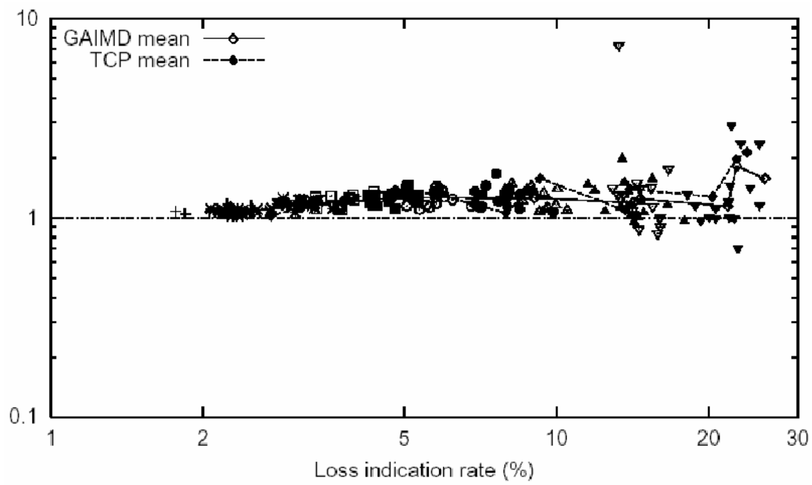
accuracy for individual GAIMD flows and TCP flows



α=0.4, β=0.75, drop-tail router

Sending Rate Variation (3)

accuracy for individual GAIMD flows and TCP flows



α=0.4, β=0.75, RED router

Summary of Validation Tests

- Accurate for loss rate p < 20%
- Loss patterns (RED vs. drop-tail) do not have a large impact on accuracy
- Sending rate variance is small for a loss rate of up to 10%
- Trend: rate formulas tend to overestimate when loss rate is high or when α, β are *aggressive*
  - Overestimates are similar for both TCP and GAIMD (most experiments)

TCP-friendly GAIMD

- Choose α and β values such that

$$\text{send rate} = T_{\alpha,\beta}(p, RTT, T_0, b)$$

$$= \frac{1}{RTT\left(\sqrt{\frac{2b(1-\beta)p}{\alpha(1+\beta)}}\right) + \min\left(1, 3\sqrt{\frac{(1-\beta^2)bp}{2\alpha}}\right) p(1+32p^2)T_0}$$

$$= T_{1,\frac{1}{2}}(p, RTT, T_0, b)$$

- For all p, only solution is $\alpha = 1$ and $\beta = 1/2$

TD TCP-friendly curve

$$TD_{\alpha,\beta}(p, RTT, b) = TD_{1,\frac{1}{2}}(p, RTT, b)$$

$$RTT\left(\sqrt{\frac{2b(1-\beta)p}{\alpha(1+\beta)}}\right) = RTT\left(\sqrt{\frac{2b(1-1/2)p}{(1+1/2)}}\right)$$

$$\boxed{\alpha = \frac{3(1-\beta)}{(1+\beta)}}$$

TO TCP-friendly curve

$$TO_{\alpha,\beta}(p,T_0,b) = TO_{1,\frac{1}{2}}(p,T_0,b)$$

$$\min\left(1, 3\sqrt{\frac{(1-\beta^2)bp}{2\alpha}}\right) p(1+32p^2)T_0 = \min\left(1, 3\sqrt{\frac{(1-1/4)bp}{2}}\right) p(1+32p^2)T_0$$

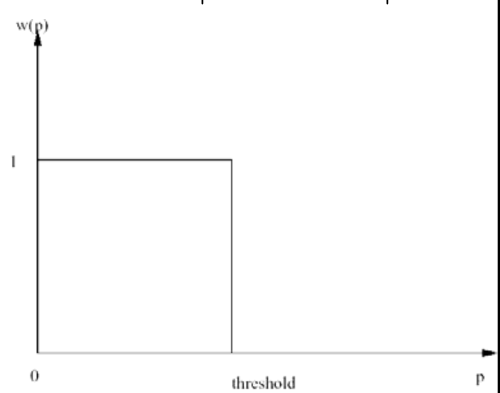
$$\frac{(1-\beta^2)}{2\alpha} = \frac{3}{8}$$

$$\boxed{\alpha = \frac{4(1-\beta^2)}{3}}$$

Minimizing error over a range of *p* values

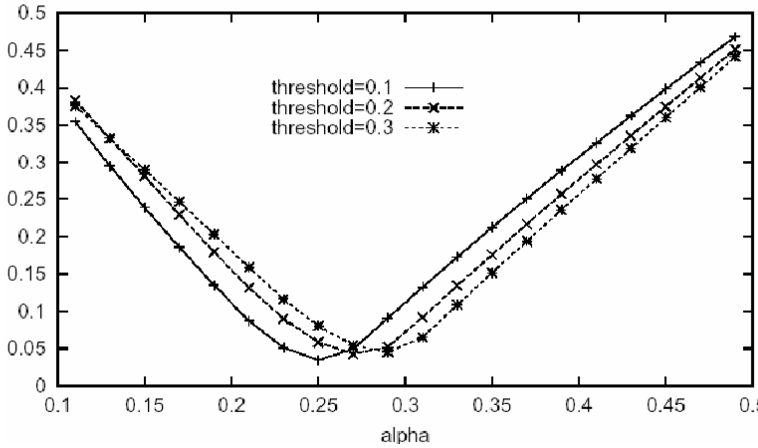
- Error function $$E_\beta(\alpha) = \int_0^1 w(p)\left|\frac{T_{\alpha,\beta}(p)}{T_{1,\frac{1}{2}}(p)} - 1\right| dp$$

  where *w*(*p*) allocates weight over *p* between 0 and 1



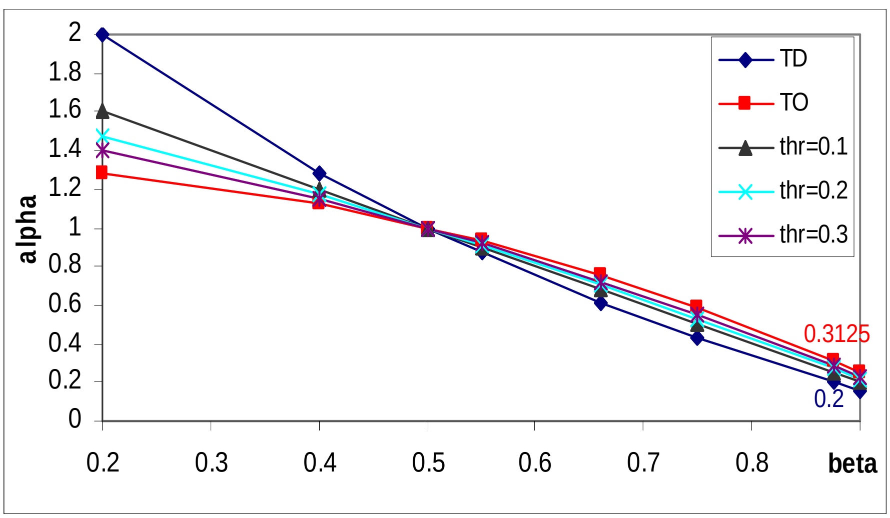
- For a given β, minimize error to get the best α

Error as a function of α



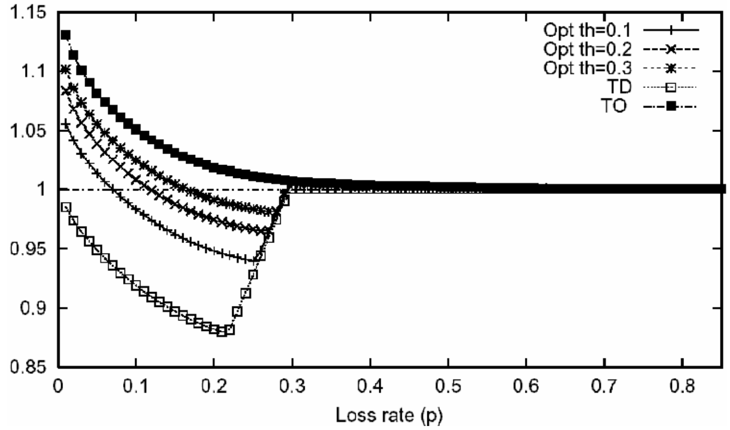
- β = 0.875 T_0 = 4(RTT)
- Optimal value of α increases as threshold increases

(α, β) curves for the three approaches



Comparing the three approaches

$$\frac{T_{\alpha,\beta}(p)}{T_{1,\frac{1}{2}}(p)}$$

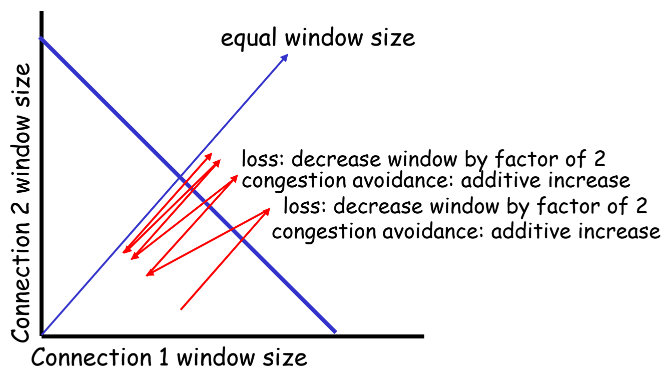


- β = 0.875
- As to be shown, TCP is more aggressive at higher loss rates than the model's prediction. Therefore, it is okay to choose the TO approach

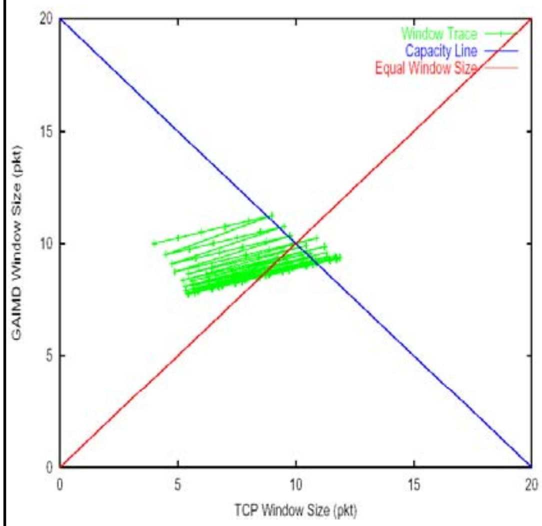
Chiu and Jain model

Two competing TCP Reno flows:

- Additive increase gives slope of 1, as window size increases
- Multiplicative decrease reduces window size proportionally



Evolution of Window Sizes



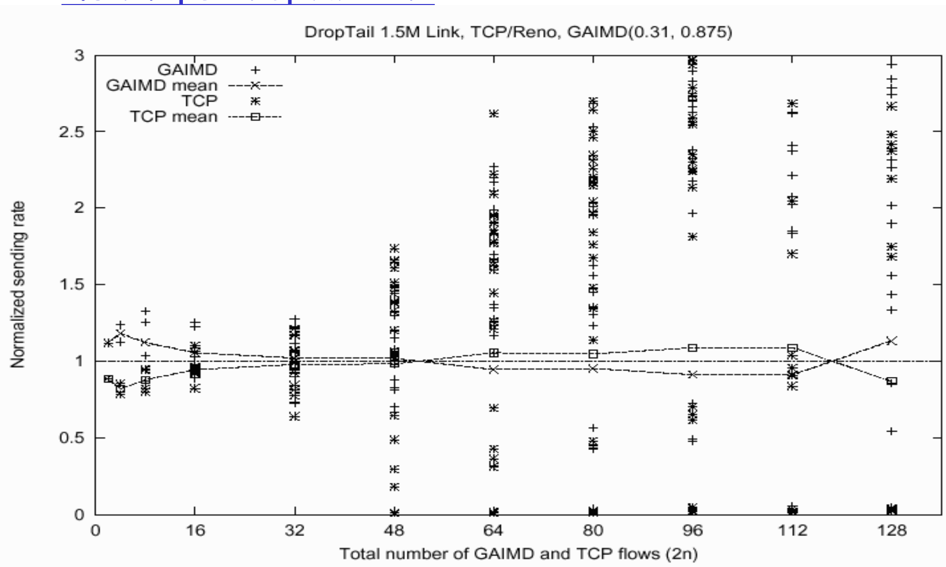
- Apply Chiu and Jain [5] model to a TCP flow and a GAIMD flow (no timeout, *same* RTT)
- GAIMD with α=0.31 and β=0.875
- Windows of the two flows do not converge to equal window size curve, but zigzag across it
- GAIMD has smaller window size oscillations

Experiments on TCP friendliness

- TCP Reno/SACK flows compete with GAIMD(0.31, 0.875) flows, *n* flows each, same simulation topology
- Drop-tail or RED bottleneck link
- Each run for 120 seconds of simulated time
- Vary *n* from 1 to 64
- Loss rate controlled by n value and link bandwidth

GAIMD competing with Reno

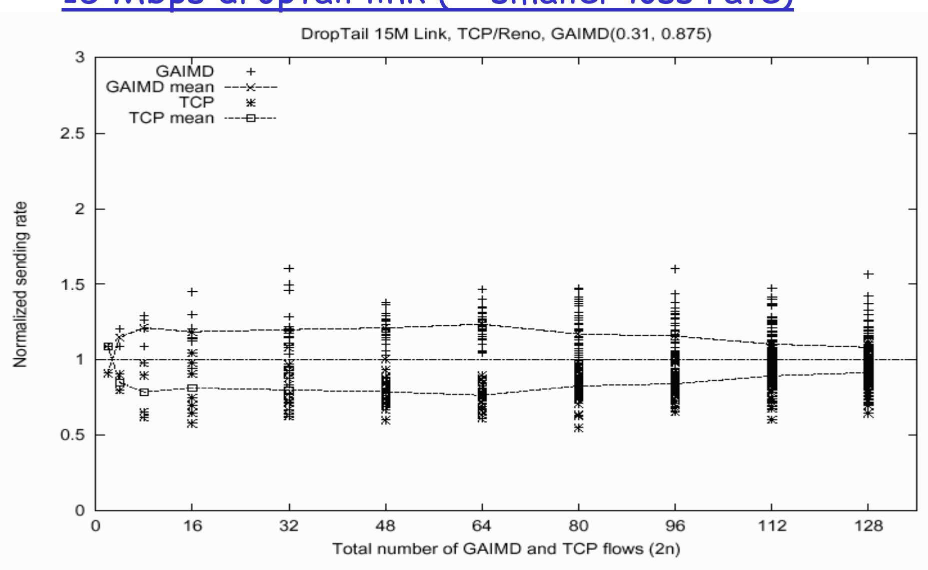
1.5 Mbps droptail link



GAIMD (Simon Lam) 27

GAIMD competing with Reno

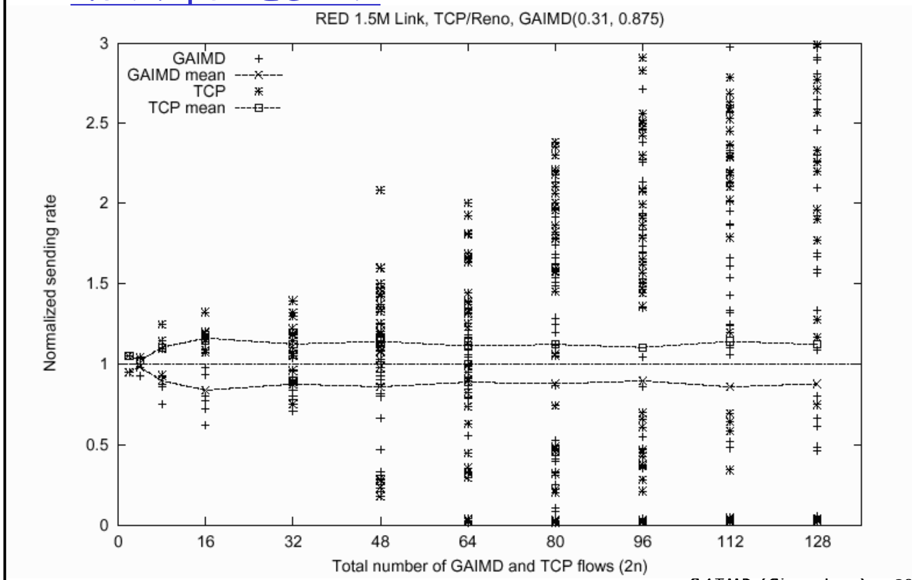
15 Mbps droptail link (-> smaller loss rate)



GAIMD (Simon Lam) 28

GAIMD competing with Reno

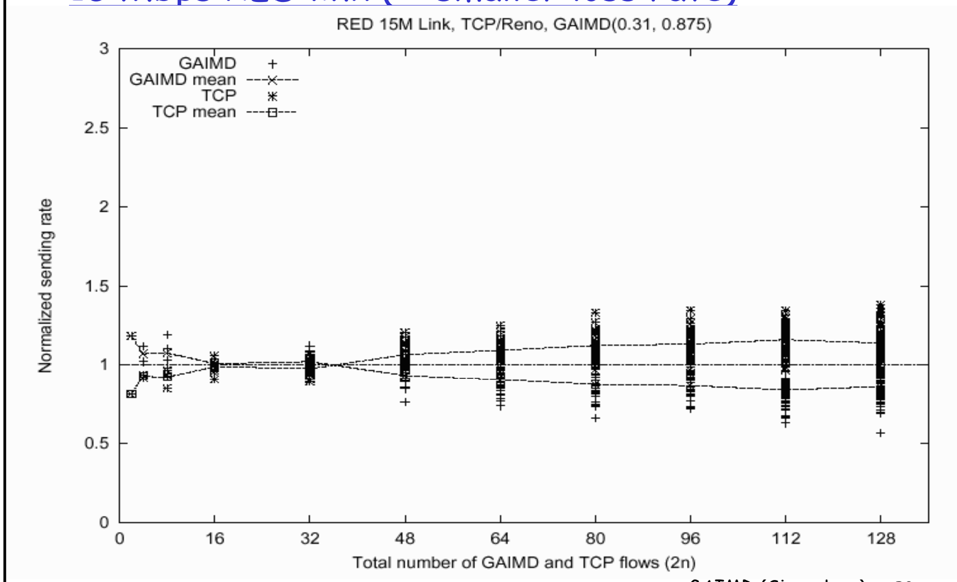
1.5 Mbps RED link



GAIMD (Simon Lam) 29

GAIMD competing with Reno

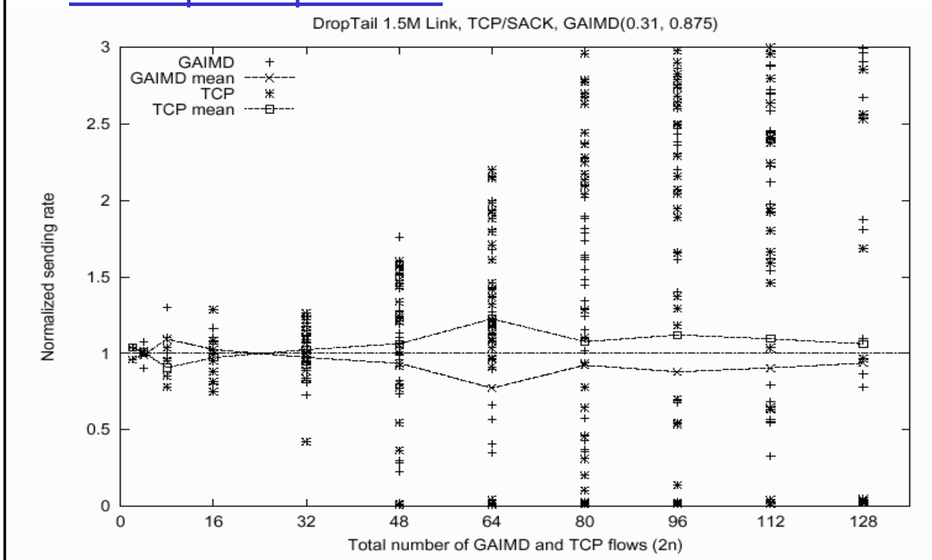
15 Mbps RED link (-> smaller loss rate)



GAIMD (Simon Lam) 30

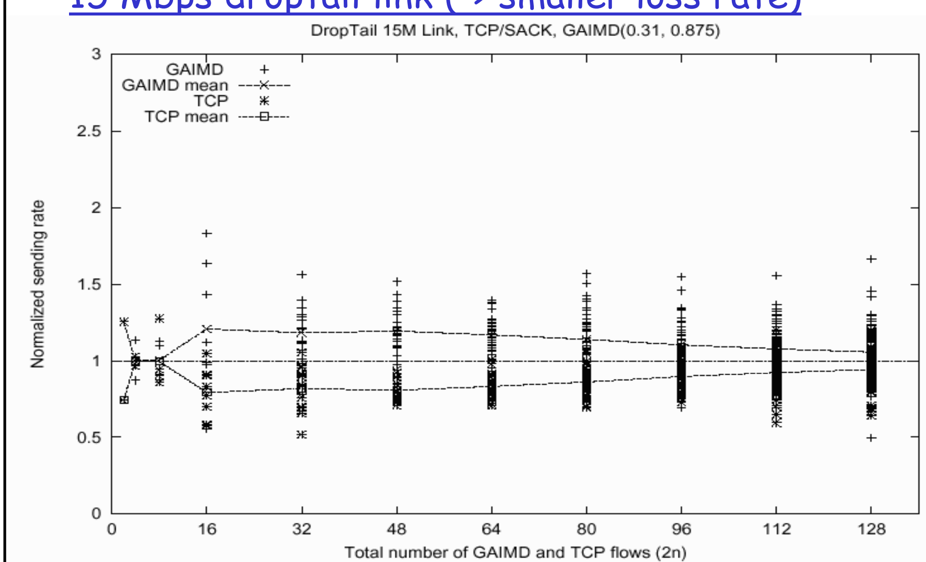
GAIMD competing with SACK

1.5 Mbps droptail link



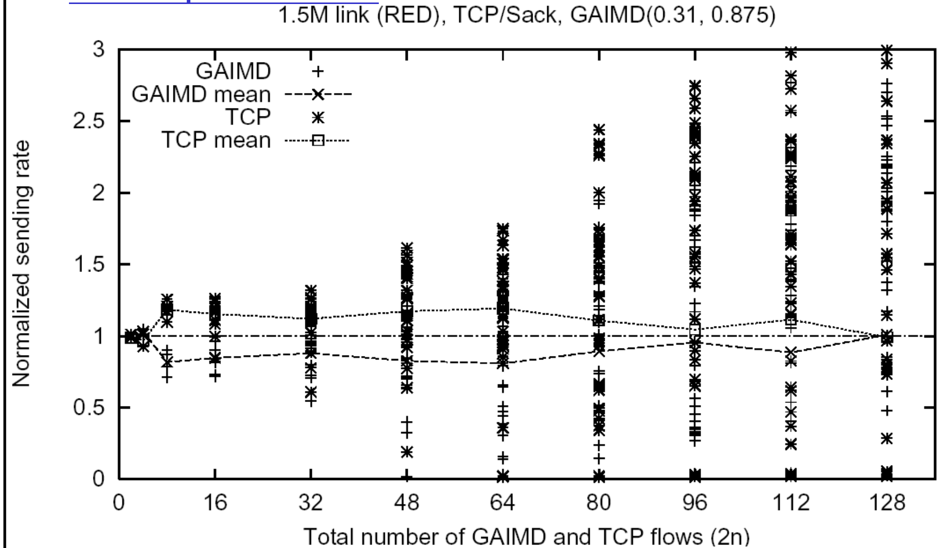
GAIMD competing with SACK

15 Mbps droptail link (-> smaller loss rate)



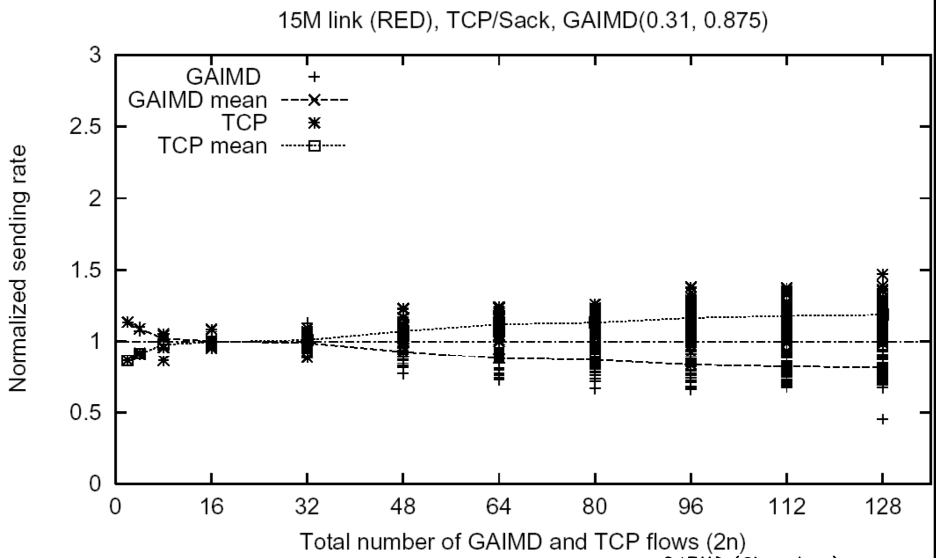
GAIMD competing with SACK

1.5 Mbps RED link



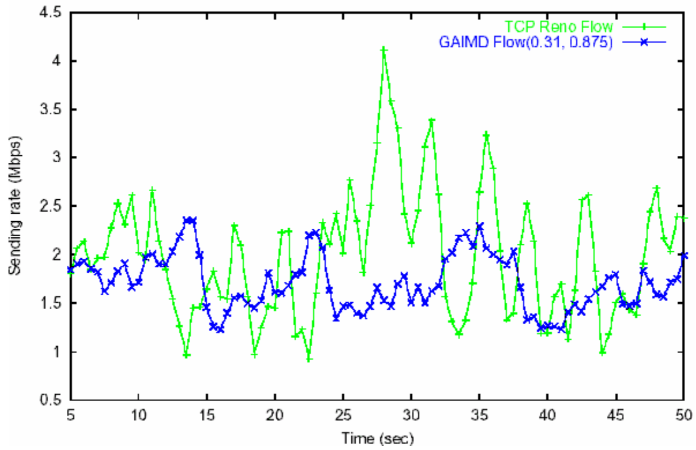
GAIMD competing with SACK

15 Mbps RED link (-> smaller loss rate)



Rate Fluctuations

4 GAIMD(0.31, 0.875) flows & 4 TCP Reno flows share



- 15 Mbps RED link
- Each point in a trace obtained by averaging over 150 ms, about 2-3 times RTT, of 1 flow
- From [33] we know that the CoV of GAIMD(0.31, 0.875) send rate is about half the CoV of TCP send rate

Conclusions

- A general version of AIMD with α and β parameter values
  - A formula for the (mean) send rate of a GAIMD flow as a functions of α, β, p, b, RTT, and T_0 and it is accurate for p up to 20%
- Relationship between α and β for GAIMD to be TCP-friendly
  - Simulation results from experiments show that GAIMD(0.31, 0.875) flows compete with TCP Reno or SACK flows, at a drop-tail or RED bottleneck link, in a friendly manner
  - GAIMD(0.31, 0.875) has reduced rate fluctuatons