Advanced Computer Networks

TCP-Friendly Congestion Control

Jianping Pan
Summer 2007

Review: congestion control

- Loss-based congestion control
  - e.g., TCP Tahoe, Reno, NewReno, etc
  - slow-start, congestion avoidance
  - timeout retransmit
  - fast retransmit, fast recovery
- Delay-based congestion control
  - e.g., TCP Vegas
  - more aggressive retransmission
  - less aggressive congestion avoidance
  - less aggressive slow start

TCP congestion control principles

- Packet conservation with ACK self-clocking
  - Q: why ACK self-clocking?
  - Q: when ACK self-clocking not working well?
  - Q: traffic with no ACK?
    - e.g., UDP-transported CBR (constant bit rate) flow
- Additive increase multiplicative decrease
  - Q: why AIMD?
    - alternatives: AIAD, MIAD, MIMD, etc
  - Q: the consequence of TCP AIMD
    - TCP: increase by one, reduce by half
    - or (1, 0.5)-AIMD

TCP-friendly congestion control

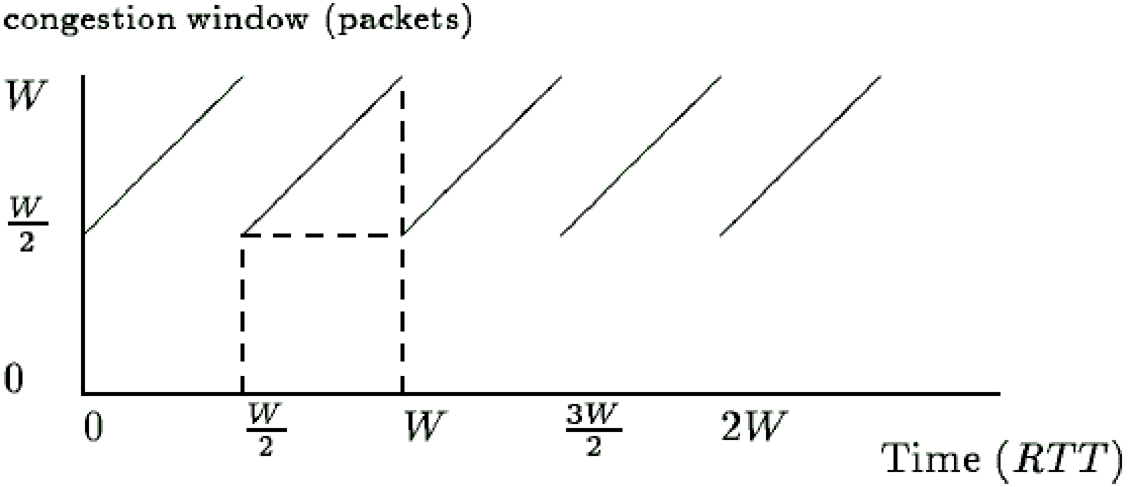
- For non-TCP traffic
  - particularly for multimedia traffic
    - no TCP-like per-packet acknowledgment
    - performance degrades severely due to rate-halving
  - to maintain friendliness with TCP
    - achieve the average throughput no more than a TCP flow can do under the same condition over a long time period
- Goal
  - allow TCP and non-TCP traffic to coexist
    - TCP traffic not adversely affected by non-TCP one
    - and vice versa

TCPFCC approaches

- Rate-based TCP-friendly congestion control
  - obtain the average throughput for TCP
    - Q: how to know the throughput of TCP
  - under the same network condition
    - e.g., packet loss ratio, round-trip time, etc
  - and set sending rate properly
- AIMD-based TCP-friendly congestion control
  - follow the same AIMD principle as TCP
  - with different sets of AIMD parameters
    - e.g., avoid rate-halving, etc
  - to maintain TCP friendliness

TCP throughput [MSMO97]

- A simple model
  - steady state
  - dupack only
  - fast recovery only
- Sawtooth cwnd
  - packets sent
  - W(p)
  - throughput



$$\left(\frac{W}{2}\right)^2 + \frac{1}{2}\left(\frac{W}{2}\right)^2 = \frac{3}{8}W^2$$

$$W = \sqrt{\frac{8}{3p}}$$

$$\frac{MSS * \frac{3}{8}W^2}{RTT * \frac{W}{2}} = \frac{MSS/p}{RTT\sqrt{\frac{2}{3p}}} = \frac{MSS}{RTT}\frac{C}{\sqrt{p}}$$

Limitations

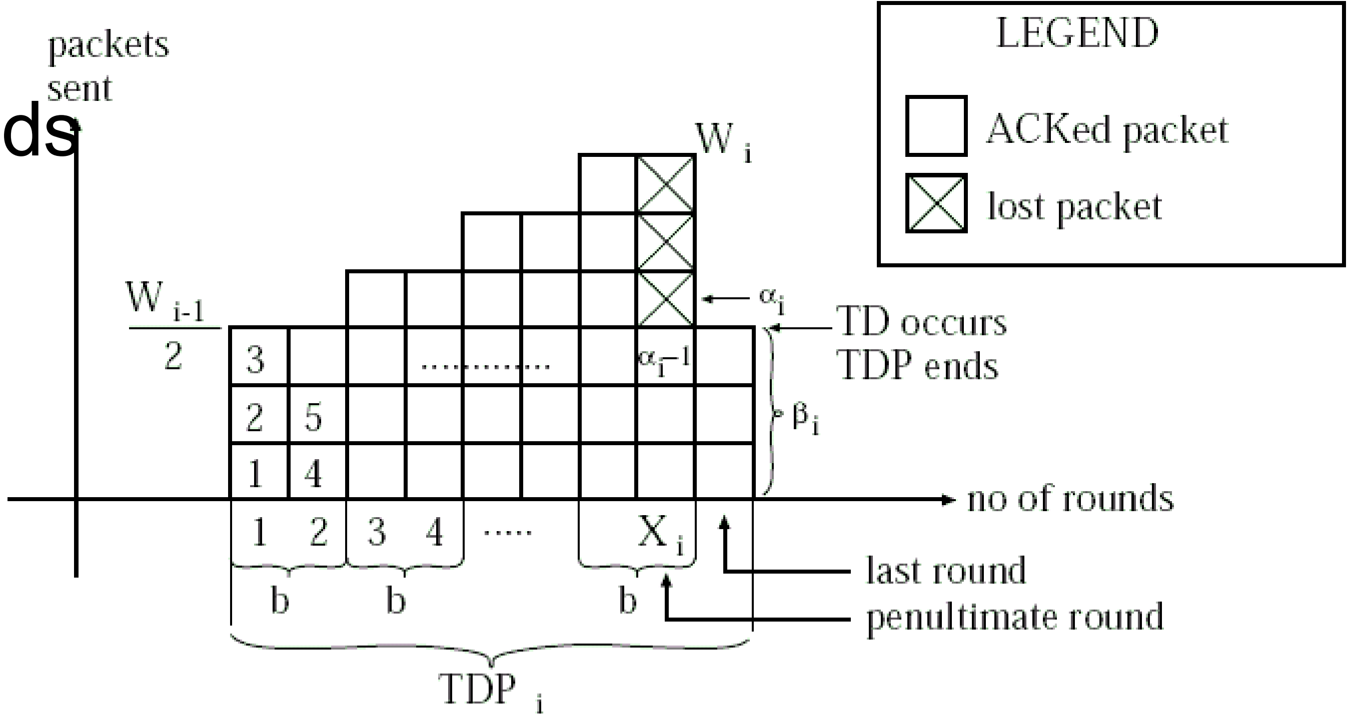
- Limitations
  - sender's window = min {rwin, buffer, cwnd}
  - sender is not persistent
  - timeout not considered
  - slow-start not considered
  - short connections
  - periodic loss
  - some other TCP implementation details
- Upper bound
  - TCP throughput $BW < \left(\frac{MSS}{RTT}\right)\frac{1}{\sqrt{p}}$

TCP throughput [PFTK98]

- A newer model
  - consider timeout
    - measurement indicates timeout is quite often
  - consider small receiver window
- Modeling approach
  - based on “rounds”
  - round: from the back-to-back transmission of W packets (cwnd size) till their first acknowledgment
  - RTT is independent of W
  - transmission time << RTT
  - packet loss: tail-drop

TD-only

- TDP: TD-period
  - initial cwnd: $W_{i-1}/2$
  - increased by 1/b MSS per round
    - b=2 for delayed ack
  - i.e., increased by 1 MSS per b rounds
- TCP throughput



$$B = \frac{E[Y]}{E[A]}$$

$$E[Y] = \frac{1-p}{p} + E[W]$$

$$E[A] = (E[X]+1)E[r]$$

$$B(p) = \frac{1}{RTT}\sqrt{\frac{3}{2bp}} + o(1/\sqrt{p})$$

TD and TO

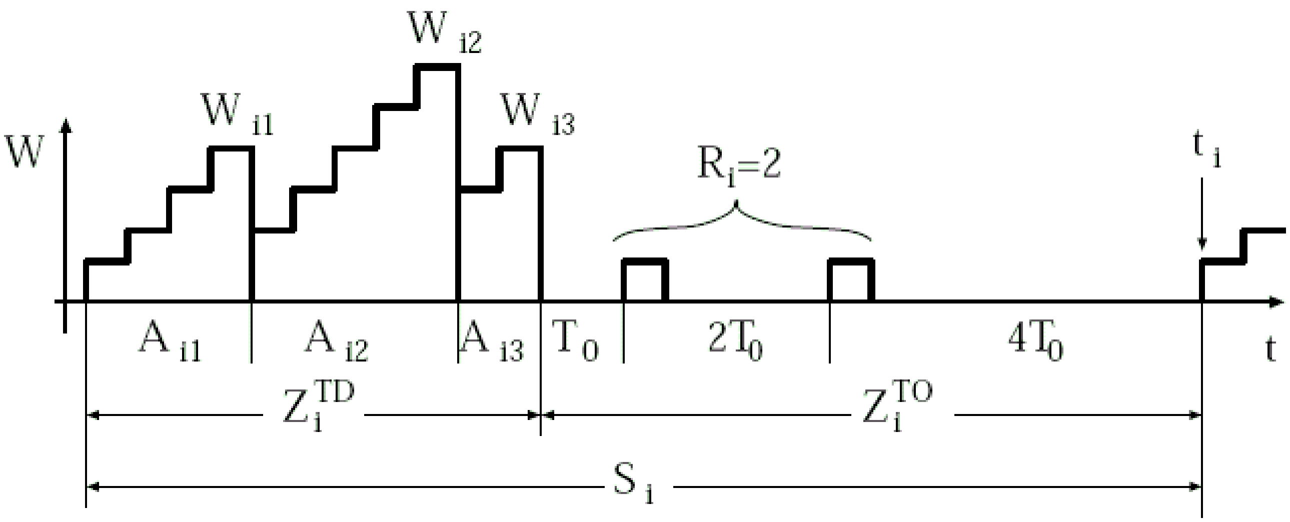
- Example
  - timeout after T_0
  - cwnd reset to 1 MSS
  - timeout again after $2T_0$
    - timer backoff
- TCP throughput

$$B = \frac{E[M]}{E[S]}$$

$$E[M] = E[\sum_{j=1}^{n_i} Y_{ij}] + E[R],$$

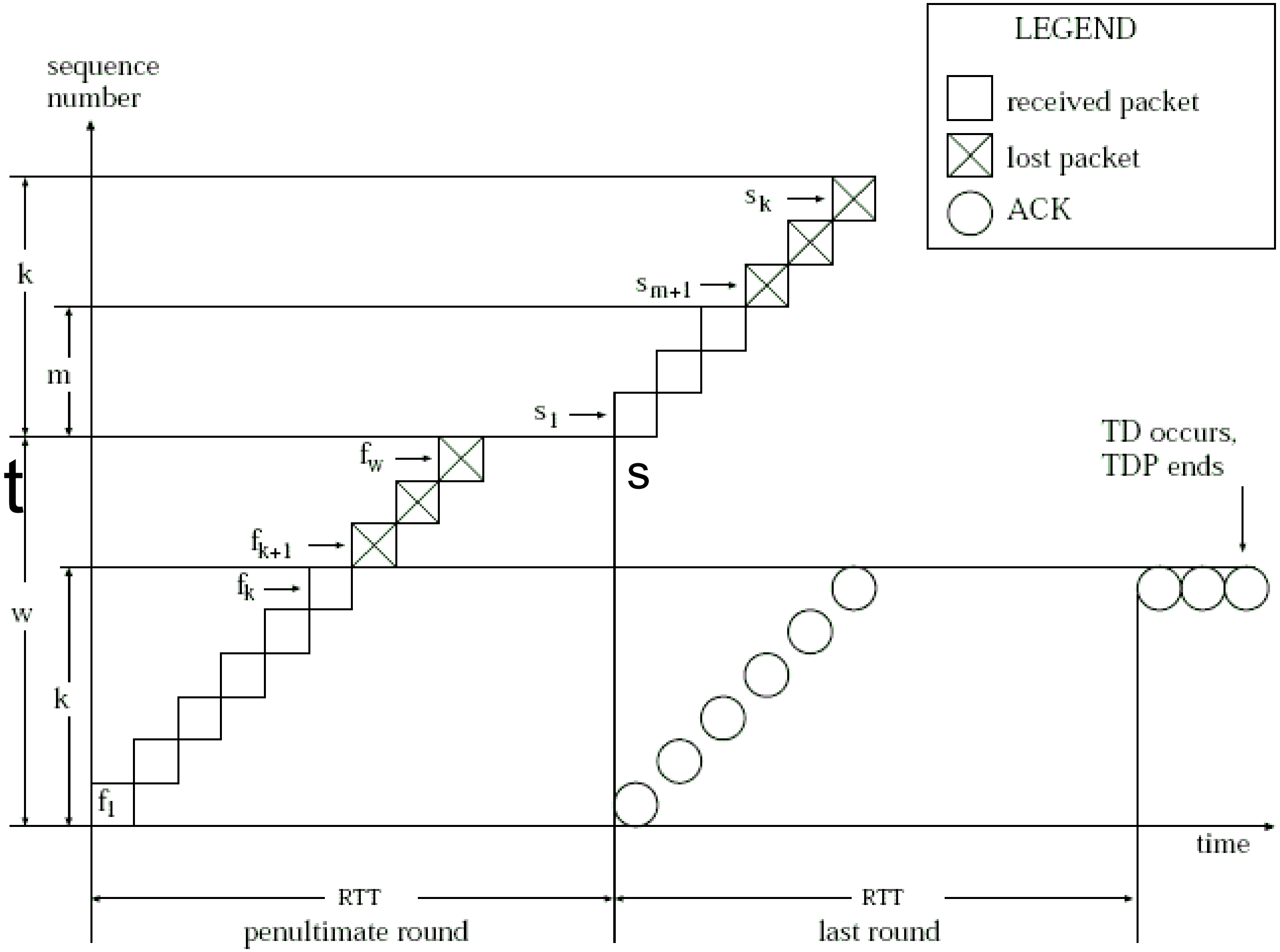
$$E[S] = E[\sum_{j=1}^{n_i} A_{ij}] + E[Z^{TO}]$$

$$B = \frac{E[Y] + Q * E[R]}{E[A] + Q * E[Z^{TO}]} \qquad \check{Q} = 1/E[n]$$



How to determine Q

- The 2nd last round
  - w packets sent
  - k acknowledged
- The last round
  - k more packets sent
  - m duplicate acknowledgments for f_{k+1}
  - either TD or TO happens



$$\hat{Q}(w) = \begin{cases} 1 & \text{if } w \le 3 \\ \sum_{k=0}^{2} A(w,k) + \sum_{k=3}^{w} A(w,k) \sum_{m=0}^{2} C(k,m) & \text{otherwise} \end{cases}$$

b=1 in this example

TCP throughput with TD and TO

- So far

$$B = \frac{E[Y] + Q * E[R]}{E[A] + Q * E[Z^{TO}]}$$

- How to determine E[R]

$$E[R] = \sum_{k=1}^{\infty} kP[R = k] = \frac{1}{1-p}$$

- How to determine E[Z^{TO}]
  - TCP timer backoff
    - 2, 4, 8, 16, 32, 64, 64, 64, ...
  - give up after a certain number of retries

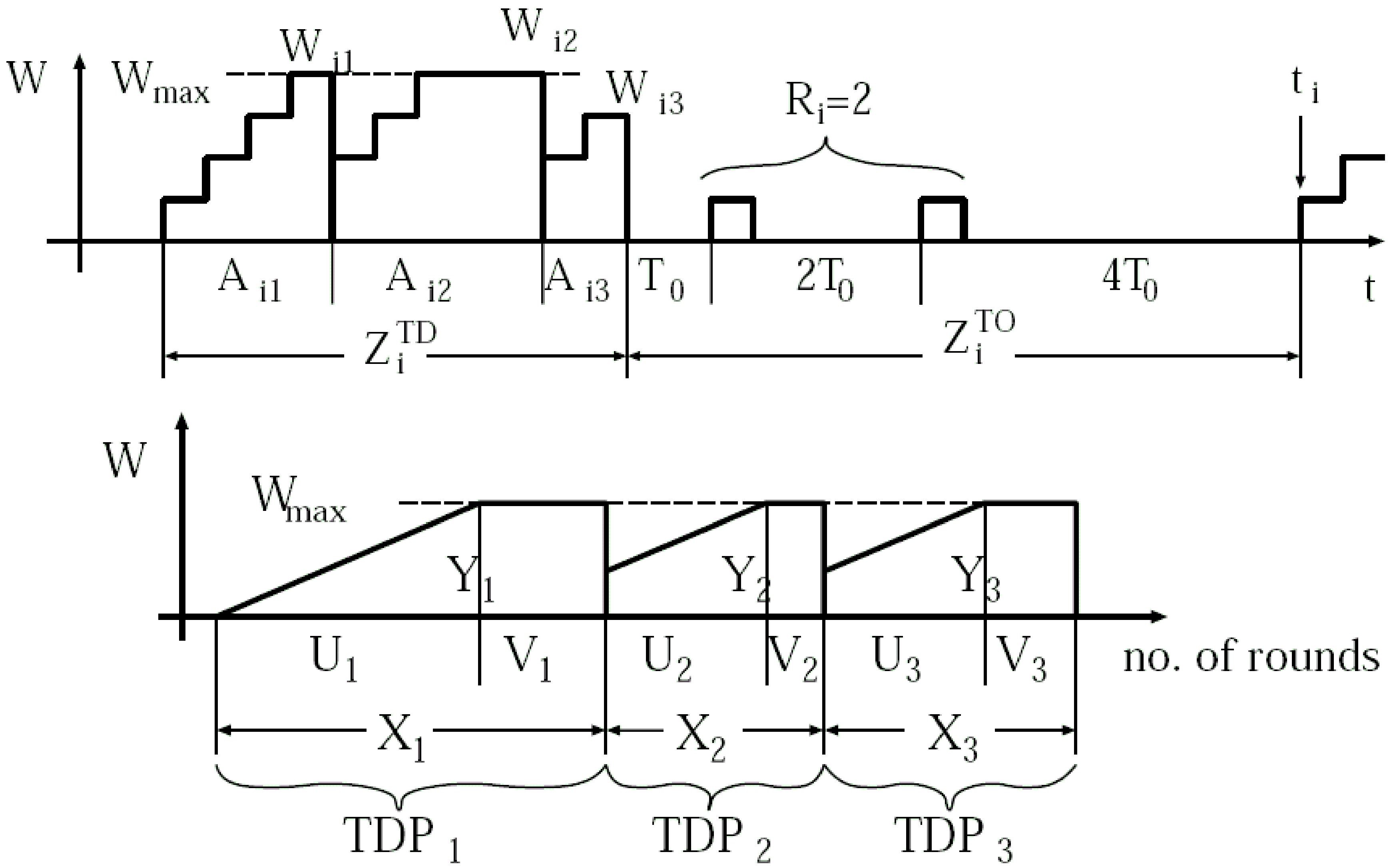
$$L_k = \begin{cases} (2^k - 1)T_0 & \text{for } k \le 6 \\ (63 + 64(k-6))T_0 & \text{for } k \ge 7 \end{cases}$$

$$\begin{aligned} E[Z^{TO}] &= \sum_{k=1}^{\infty} L_k P[R = k] \\ &= T_0 \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1-p} \end{aligned}$$

- TCP throughput

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1 + 32p^2)}$$

The impact of window limitation



Limitations

- Discussion

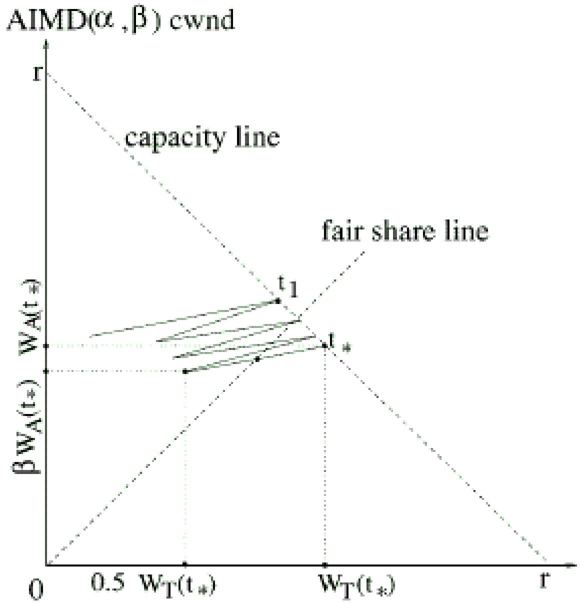
AIMD-based congestion control

- Follow the same AIMD principle as TCP
  - with parameters other than (1, 0.5)
- Example
  - one TCP and one AIMD
  - fluid model when underload: AI

$$W_A(t+\Delta t) = W_A(t) + \alpha \cdot \Delta t \qquad \frac{W_A(t+\Delta t) - W_A(t)}{W_T(t+\Delta t) - W_T(t)} = \alpha$$
$$W_T(t+\Delta t) = W_T(t) + 1 \cdot \Delta t.$$

  - fluid model when overload: MD
    - r: bottleneck capacity

$$W_A(t_i) + W_T(t_i) = r$$
$$W_A(t_i^+) = \beta W_A(t_i)$$
$$W_T(t_i^+) = 0.5 W_T(t_i)$$



TCP-friendly AIMD parameters

- Converged window size in overload state

$$W_A(t_*) = \frac{r \cdot \alpha}{2(1-\beta)+\alpha}$$

$$W_T(t_*) = \frac{r \cdot 2(1-\beta)}{2(1-\beta)+\alpha}$$

- Average window size

$$\overline{W_A} = \frac{(1+\beta)}{2} W_A(t_*) = \frac{(1+\beta)\alpha r}{4(1-\beta)+2\alpha}$$

$$\overline{W_T} = \frac{(1+0.5)}{2} W_T(t_*) = \frac{3(1-\beta) r}{4(1-\beta)+2\alpha}$$

- TCP-friendly condition: $\overline{W_A} = \overline{W_T}$.

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

- For two AIMD flows: $\frac{\alpha_1}{\alpha_2} = \frac{(1+\beta_2)(1-\beta_1)}{(1-\beta_2)(1+\beta_1)}$

This lecture

- TCP-friendly congestion control
  - for non-TCP traffic
    - ack self-clocking issue
    - rate-halving problem
  - two approaches
    - rate-based (or equation-based)
    - AIMD-based
- Explore further
  - http://www.icir.org/padhye/tcp-model.html
  - http://www.psc.edu/networking/tcp_friendly.html

Next lecture

- Explicit congestion control
  - [KDR02] Dina Katabi, Mark Handley, and Chalrie Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. In the proceedings on ACM Sigcomm 2002. [XCP]
- Student presentations are back
  - presenters are notified one week in advance