ELE539A: Optimization of Communication Systems
Lecture 9: NUM and TCP Congestion Control

Professor M. Chiang
Electrical Engineering Department, Princeton University

February 24, 2006

Lecture Outline

- TCP congestion control protocol
- Network utility maximization
- Reverse engineering

Acknowledgement: Steven Low

Last Lecture

- How to derive subgradient: Danskin's Theorem
- How to choose step size: diminishing and constant step size rules

Network Utility Maximization

Basic NUM:

$$\begin{array}{ll} \text{maximize} & \sum_s U_s(x_s) \\ \text{subject to} & \mathbf{R}\mathbf{x} \preceq \mathbf{c} \\ & \mathbf{x} \succeq 0 \end{array}$$

- Extension of LP-based Network Flow Problem
- TCP congestion control protocols reverse engineered: they solve the basic NUM for different utility functions

Dual-based Distributed Algorithm

Basic NUM with concave smooth utility functions:

Convex optimization (Monotropic Programming) with zero duality gap

Lagrangian decomposition:

$$
\begin{aligned}
L(\mathbf{x}, \boldsymbol{\lambda}) &= \sum_s U_s(x_s) + \sum_l \lambda_l \left(c_l - \sum_{s:l \in L(s)} x_s \right) \\
&= \sum_s \left[U_s(x_s) - \left(\sum_{l \in L(s)} \lambda_l \right) x_s \right] + \sum_l c_l \lambda_l \\
&= \sum_s L_s(x_s, \lambda^s) + \sum_l c_l \lambda_l
\end{aligned}
$$

Dual problem:

$$
\begin{array}{ll}
\text{minimize} & g(\boldsymbol{\lambda}) = L(\mathbf{x}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda}) \\
\text{subject to} & \boldsymbol{\lambda} \succeq 0
\end{array}
$$

Dual-based Distributed Algorithm

Source algorithm:

$$x_s^*(\lambda^s) = \mathrm{argmax}\left[U_s(x_s) - \lambda^s x_s\right], \quad \forall s$$

- Selfish net utility maximization locally at source s

Link algorithm (gradient or subgradient based):

$$\lambda_l(t+1) = \left[\lambda_l(t) - \alpha(t)\left(c_l - \sum_{s:l\in L(s)} x_s(\lambda^s(t))\right)\right]^+, \quad \forall l$$

- Balancing supply and demand through pricing

Certain choices of step sizes $\alpha(t)$ of distributed algorithm guarantee convergence to globally optimal $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$

TCP Congestion Control

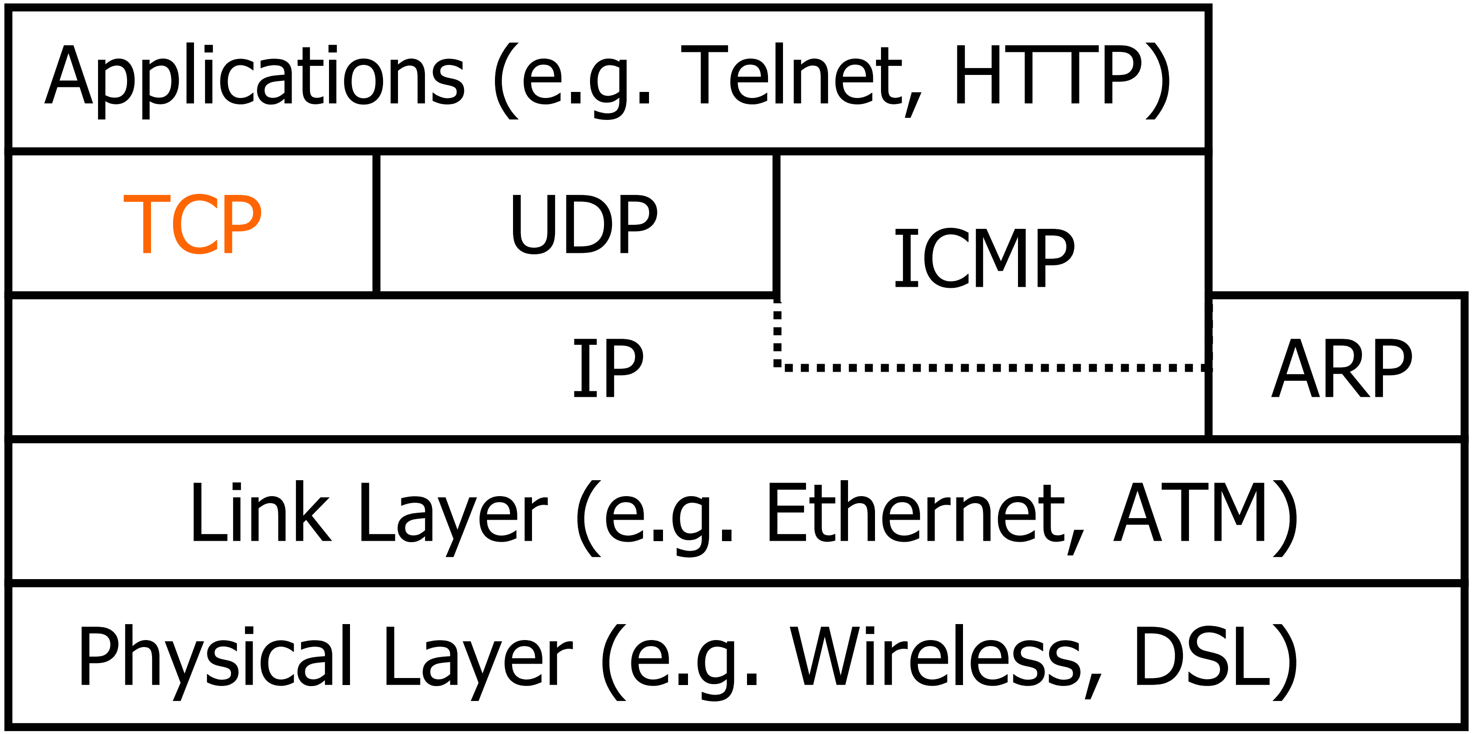
• Window-based end-to-end flow control, where destination sends ACK for correctly received packets and source updates window size (which is proportional to allowed transmission rate)

• Several versions of TCP congestion control distributively dissolve congestion in bottleneck link by reducing window sizes

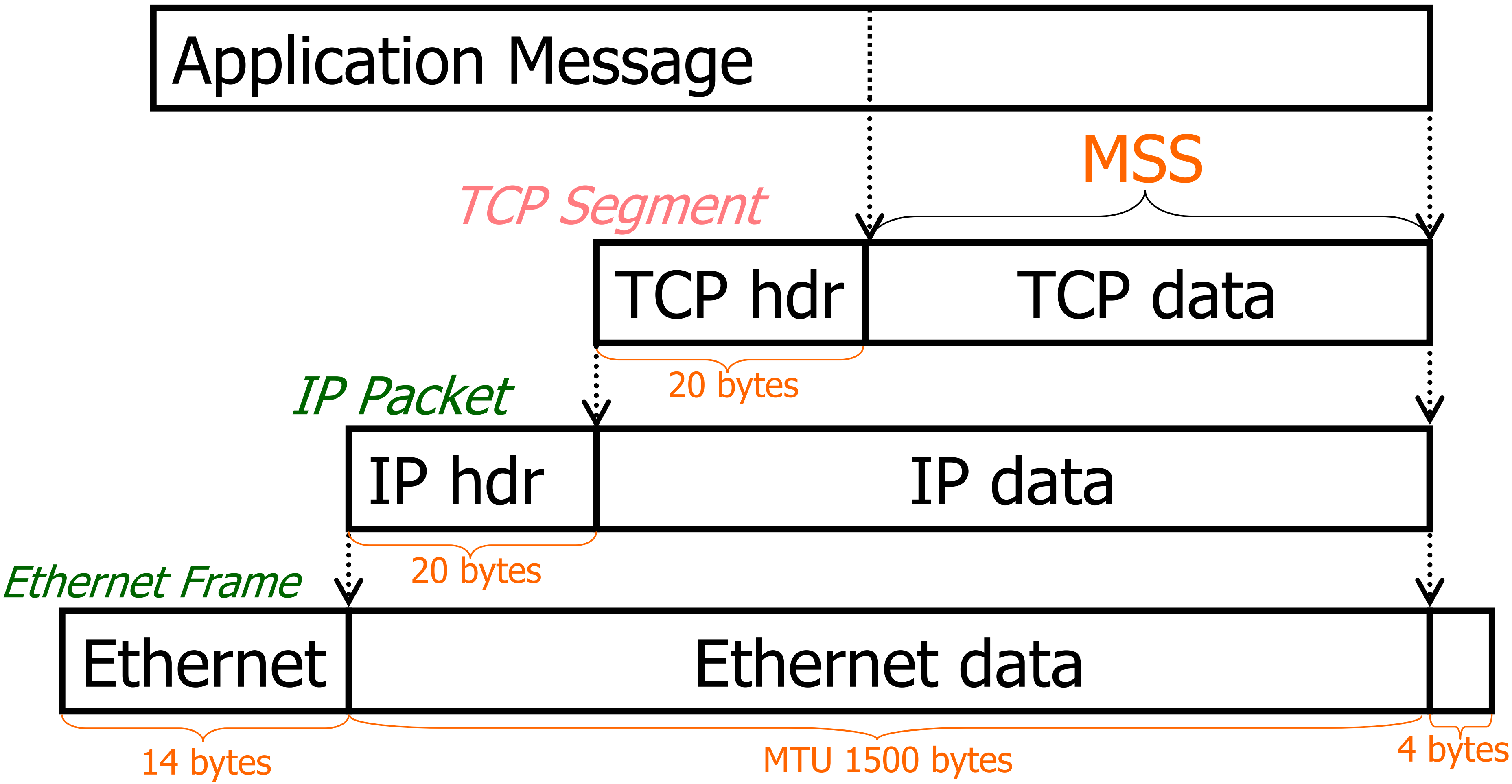
• Sources update window sizes and links update (sometimes implicitly) congestion measures that are feed back to sources using the link

Optimization-theoretic model: TCP congestion control carries out a distributed algorithm to solve an implicit, global convex optimization (network utility maximization), where source rates are primal variables updated at sources, and congestion measures are dual variables (shadow prices) updated at links

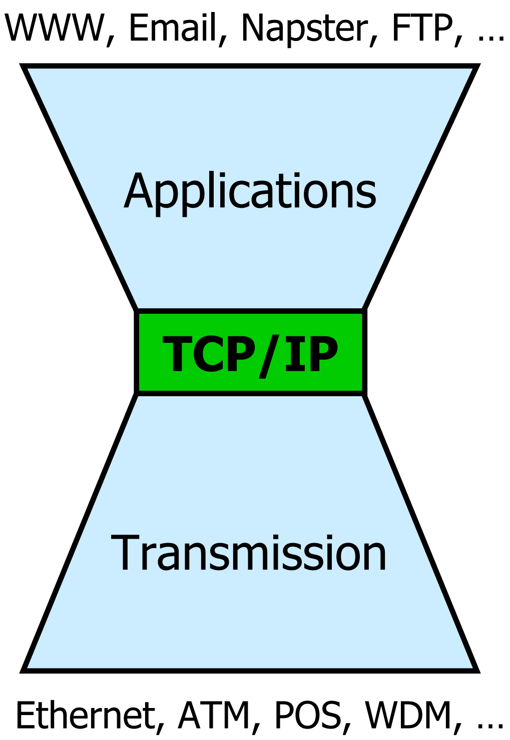
TCP/IP Protocol Stack



Packet Terminology



Success of TCP/IP



Simple/Robust

- Robustness against failure
- Robustness against technological evolutions
- Provides a service to applications
  - Doesn’t tell applications what to do

TCP Protocol

- End-to-end control
- Session initiation and termination
- In-order recovery of packets
- Flow control / congestion control
- …

Why Congestion Control

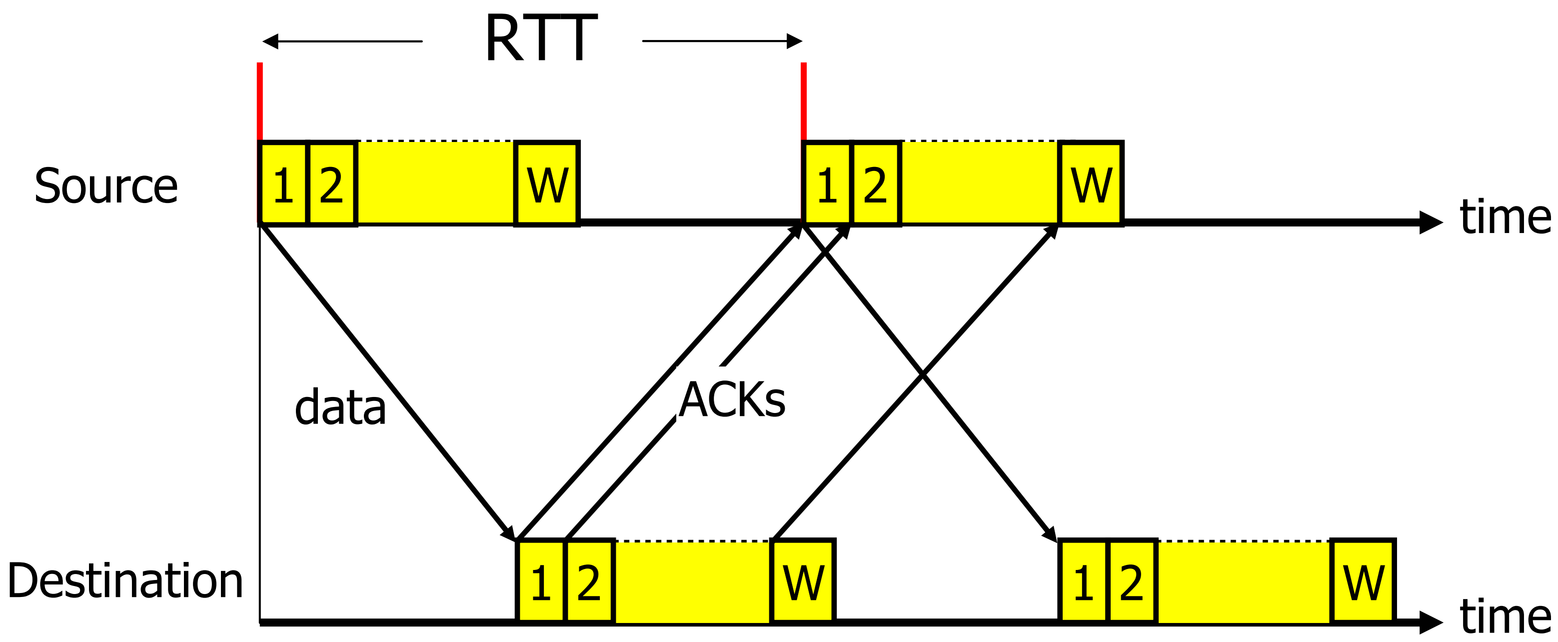
Oct. 1986, Internet had its first congestion collapse (LBL to UC Berkeley)

- 400 yards, 3 hops, 32 kbps
- throughput dropped by a factor of 1000 to 40 bps

1988, Van Jacobson proposed TCP congestion control

- Window based with ACK mechanism
- End-to-end

Window Flow Control



- ~ W packets per RTT
- Lost packet detected by missing ACK

TCP Congestion Control

- Tahoe (Jacobson 1988)
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
- Reno (Jacobson 1990)
  - Fast Recovery
- Vegas (Brakmo & Peterson 1994)
  - New Congestion Avoidance
- RED (Floyd & Jacobson 1993)
  - Probabilistic marking
- REM (Athuraliya & Low 2000)
  - Clear buffer, match rate
- Others…

Window-based Congestion Control

Limit number of packets in network to window size W

Source rate allowed (bps) $= \frac{W \times \text{Message Size}}{\text{RTT}}$

Too small W: under-utilization of link capacities

Too large W: link congestion occurs

Effects of congestion:

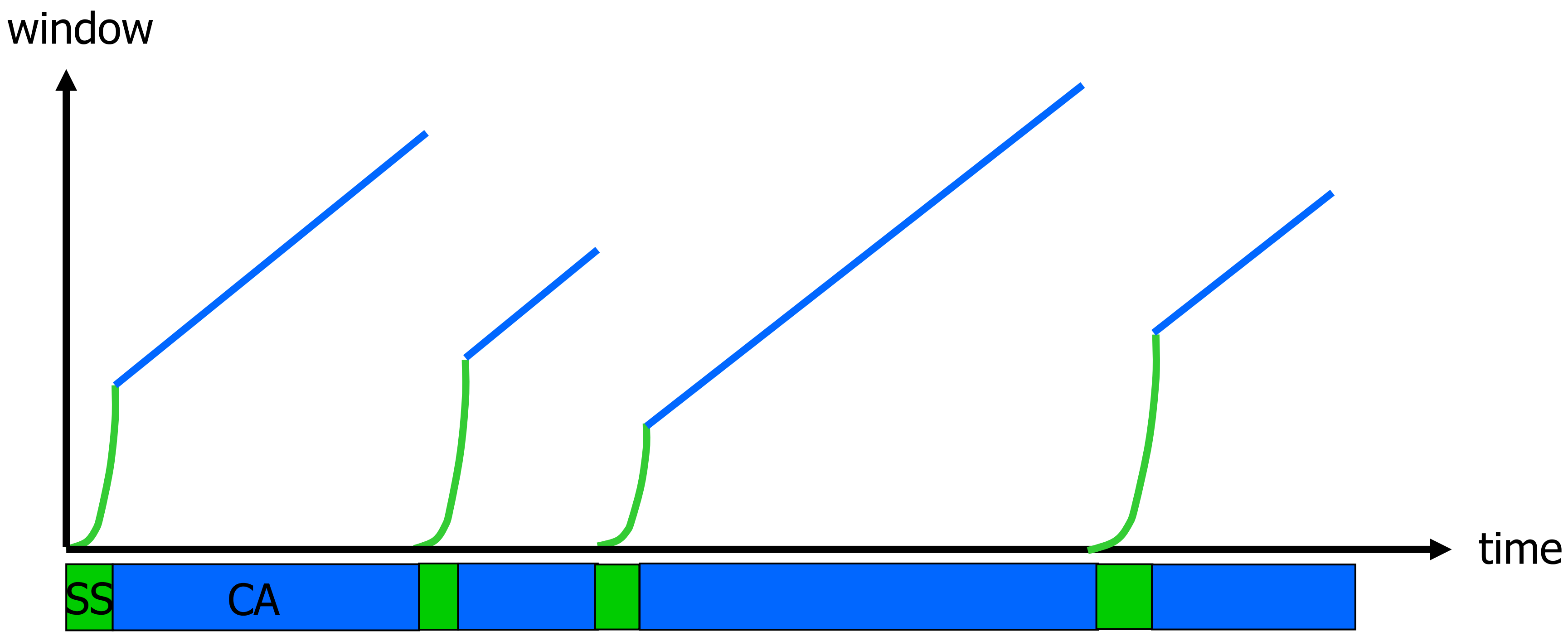
- Packet loss
- Retransmission and reduced throughput
- Congestion may continue after the overload

Basics of Congestion Control

- Goals: achieve high utilization without congestion or unfair sharing
- Receiver control (awnd): set by receiver to avoid overloading receiver buffer
- Network control (cwnd): set by sender to avoid overloading network
- $W = \min(\text{cwnd}, \text{awnd})$
- Congestion window cwnd usually the bottleneck

Different algorithms for short and long-lived flows

TCP Tahoe (Jacobson 1988)



SS: Slow Start

CA: Congestion Avoidance

Slow Start

- Start with cwnd = 1 (slow start)
- On each successful ACK increment cwnd

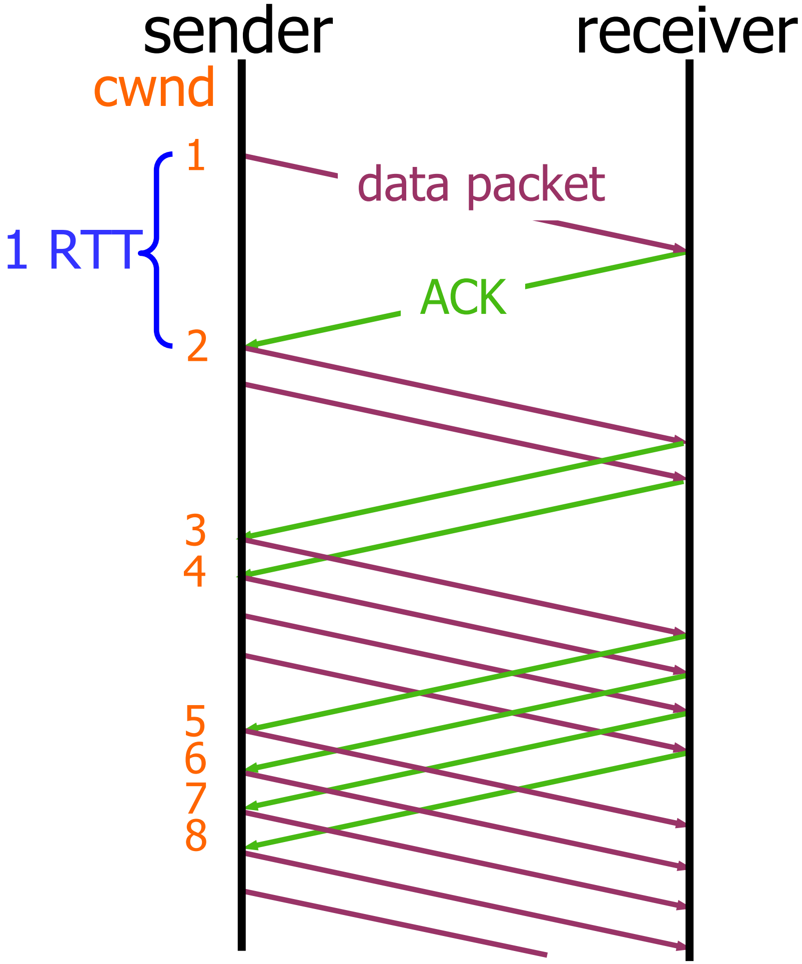
  $$cwnd \leftarrow cnwd + 1$$

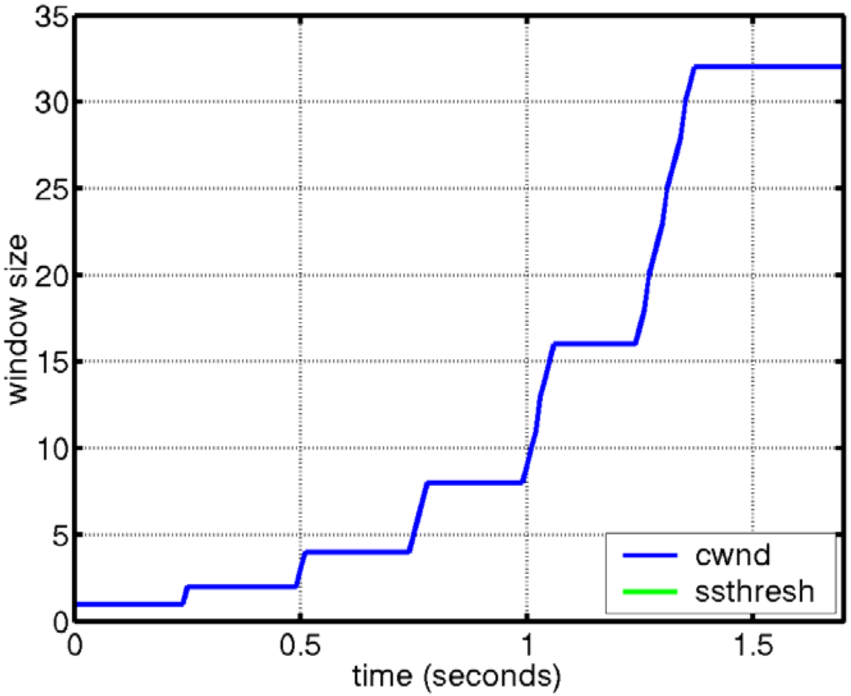
- Exponential growth of cwnd

  each RTT: $cwnd \leftarrow 2 \times cwnd$

- Enter CA when cwnd >= ssthresh

Slow Start





cwnd ← cwnd + 1 (for each ACK)

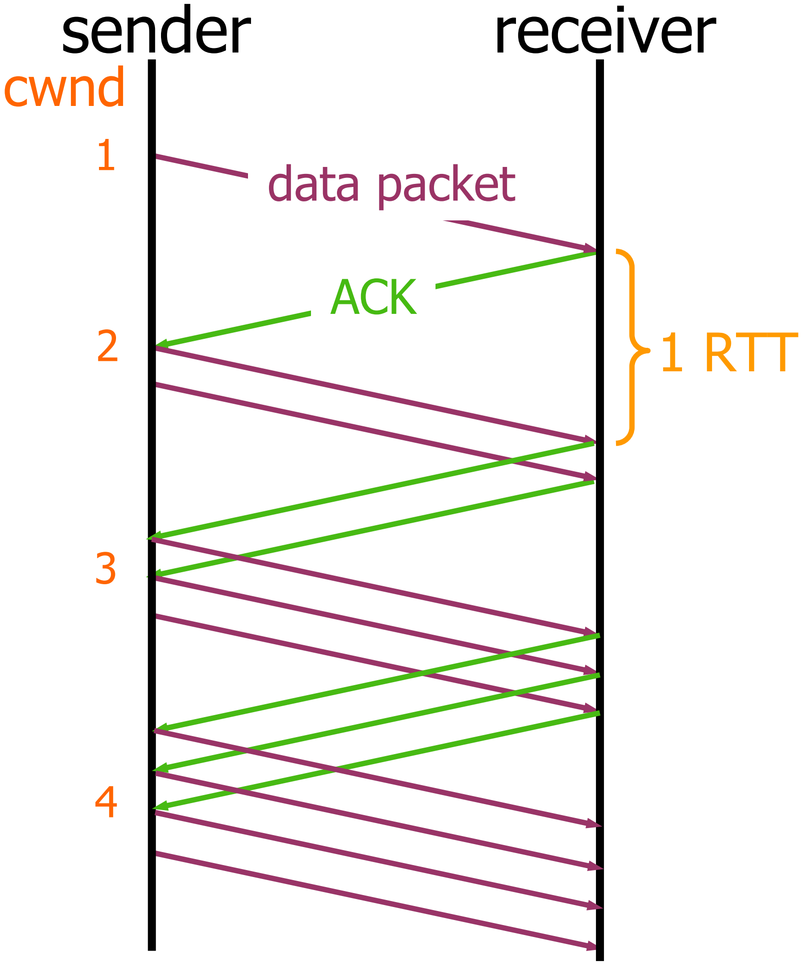
Congestion Avoidance

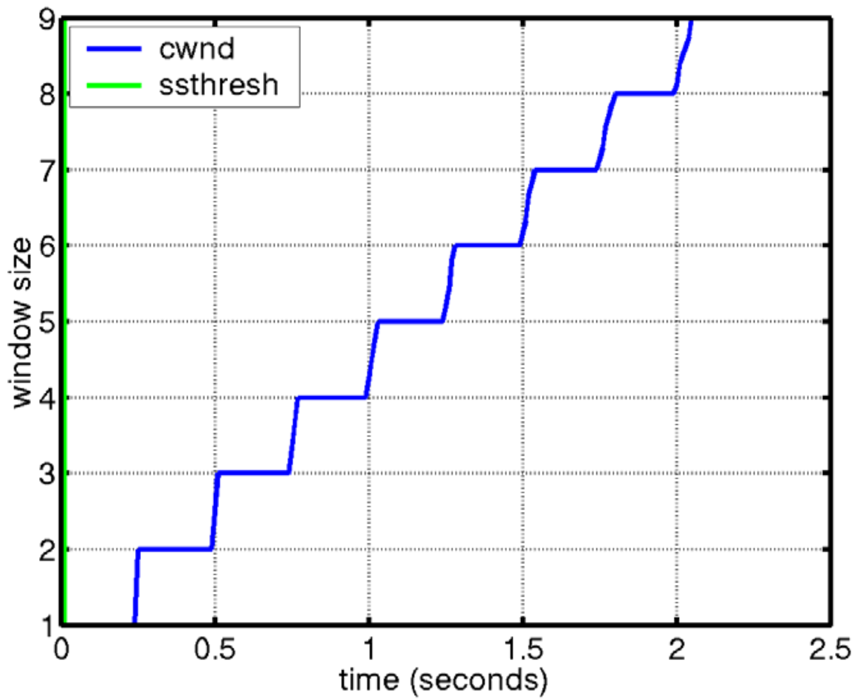
- Starts when $\text{cwnd} \geq \text{ssthresh}$
- On each successful ACK:

  $\text{cwnd} \leftarrow \text{cwnd} + 1/\text{cwnd}$
- <u>Linear growth</u> of cwnd

  each RTT: $\text{cwnd} \leftarrow \text{cwnd} + 1$

Congestion Avoidance

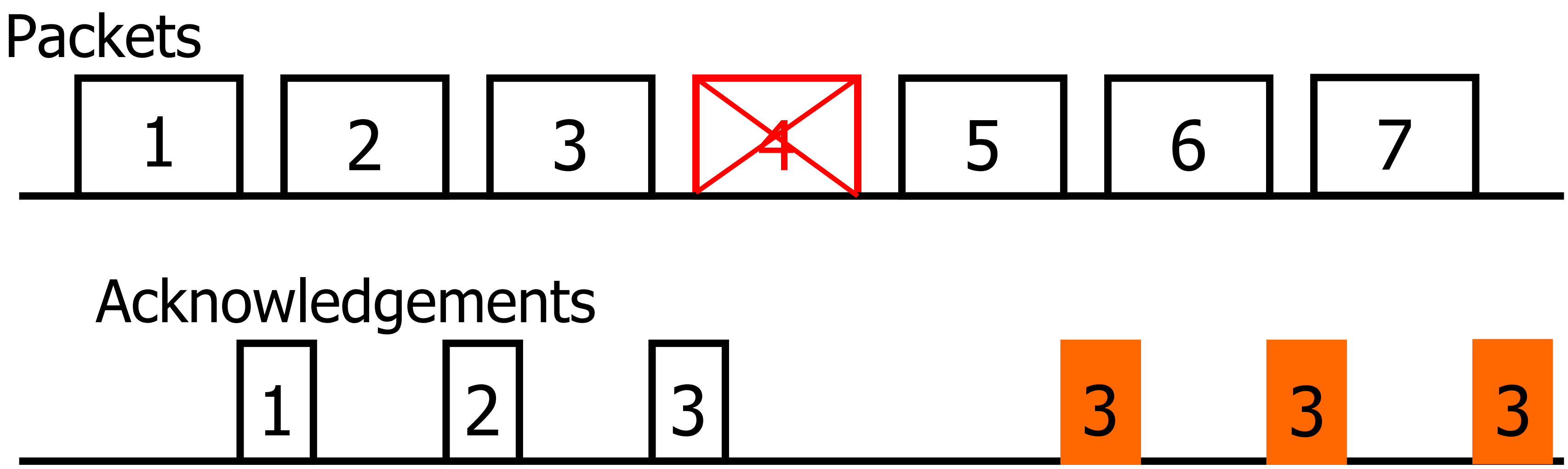




cwnd ← cwnd + 1 (for each cwnd ACKS)

Packet Loss

- Assumption: loss indicates congestion
- Packet loss detected by
  - Retransmission TimeOuts (RTO timer)
  - Duplicate ACKs (at least 3)



Fast Retransmit

- Wait for a timeout is quite long
- Immediately retransmits after 3 dupACKs without waiting for timeout
- Adjusts ssthresh

  flightsize = min(awnd, cwnd)

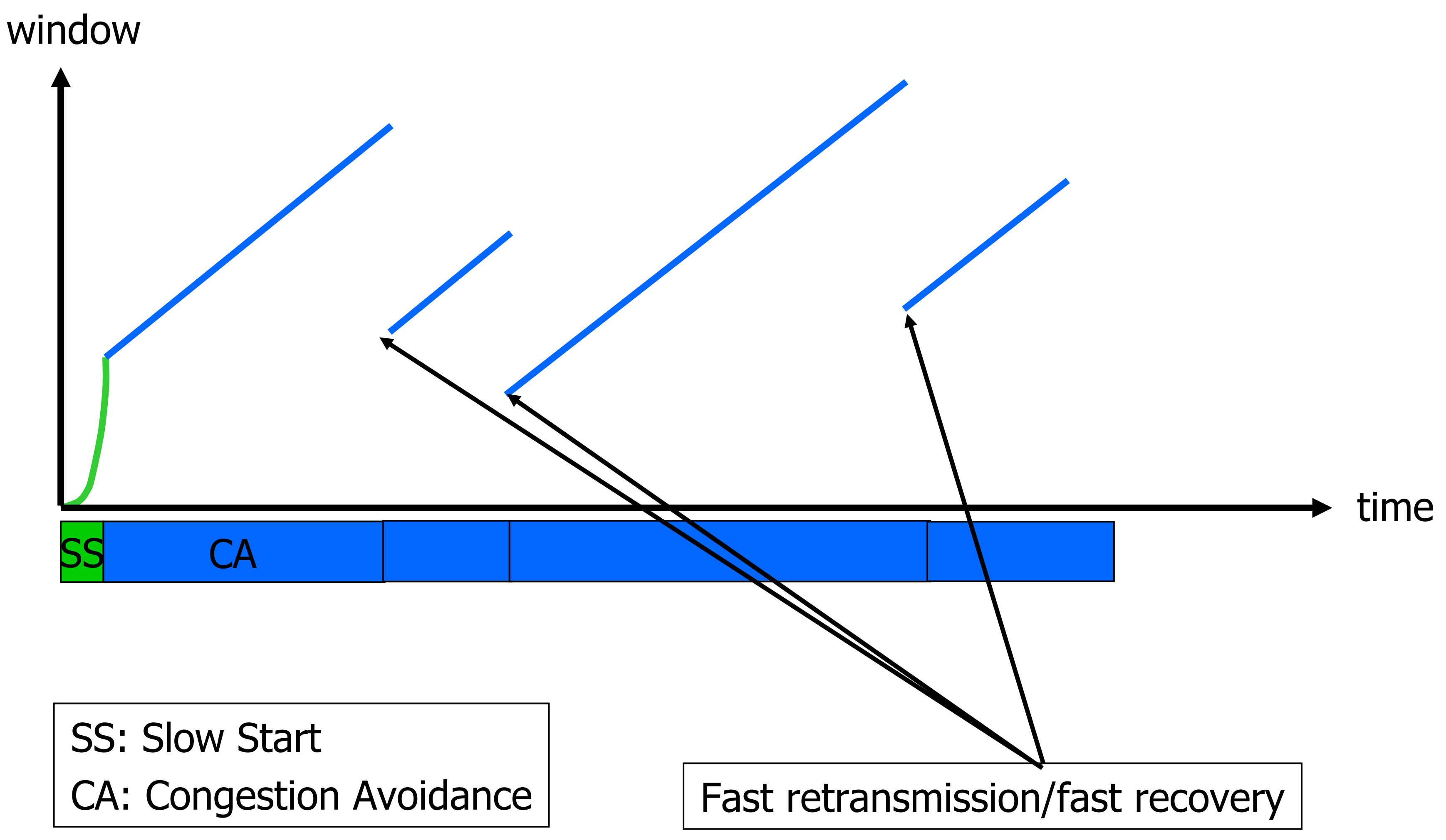
  ssthresh $\leftarrow$ max(flightsize/2, 2)
- Enter Slow Start (cwnd = 1)

Summary: Tahoe

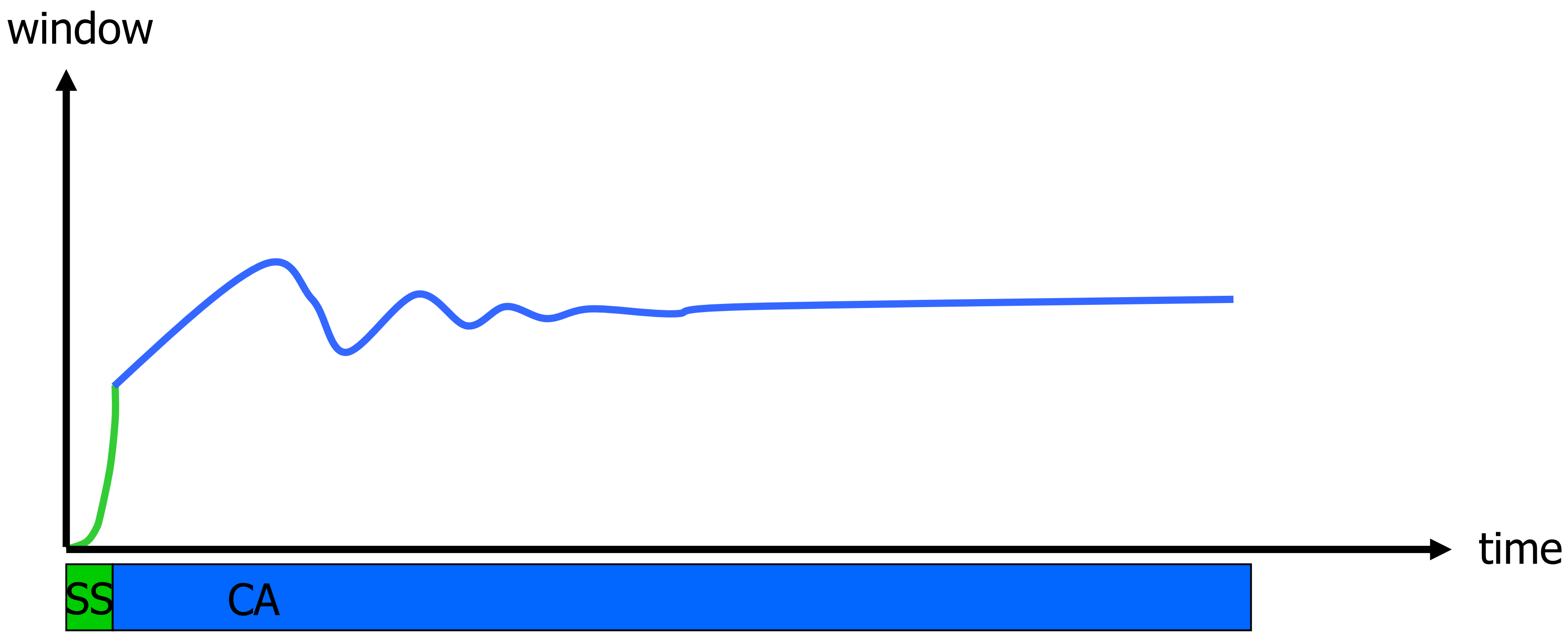
- Basic ideas
  - Gently probe network for spare capacity
  - Drastically reduce rate on congestion
  - Windowing: self-clocking
  - Other functions: round trip time estimation, error recovery

```
for every ACK {
    if (W < ssthresh) then W++      (SS)
    else  W += 1/W                  (CA)
}
for every loss {
    ssthresh = W/2
          W  = 1
}
```

TCP Reno (Jacobson 1990)



TCP Vegas (Brakmo & Peterson 1994)



- Converges, no retransmission
- … provided buffer is large enough

Vegas CA algorithm

```
for every RTT
{   if W/RTTmin – W/RTT < α then W ++
    if W/RTTmin – W/RTT > β then W --    }
for every loss
        W := W/2
```

queue size

Queue Buffer Processes

At intermediate links:

- FIFO buffer process updates queuing delay as measure of congestion for Vegas and feeds back to sources

- Drop tail updates packet loss as measure of congestion for Reno and feeds back to sources

- RED: instead of dropping only at full buffer, drops packets with a probability that increases with (exponentially weighted) average queue length (example of Active Queue Management)

Analytic Model

Communication network with L links, each with fixed capacity c_l packets per second, shared by N sources, each using a set L_i of links

R: $0-1$ routing matrix with $R_{li}=1$ iff $l \in L_i$

Deterministic flow model: $x_i(t)$ at each source i at discrete time t

Aggregate flow on link l:

$$y_l(t) = \sum_i R_{li} x_i(t - \tau_{li}^f)$$

where τ_{li}^f is forward transmission delay

Each link updates congestion measure (shadow price) $p_l(t)$. Each source has access to aggregate price along its route (end-to-end):

$$q_i(t) = \sum_l R_{li} p_l(t - \tau_{li}^b)$$

where τ_{li}^b is backward delay in feedback path

Generic Source and Link Algorithms

Each source updates rate (z_i is a local state variable):

$$\begin{aligned} z_i(t+1) &= F_i(z_i(t), q_i(t), x_i(t)) \\ x_i(t+1) &= G_i(z_i(t), q_i(t), x_i(t)) \end{aligned}$$

Often $x_i(t+1) = G_i(q_i(t), x_i(t))$

Each link updates congestion measure:

$$\begin{aligned} v_l(t+1) &= H_l(y_l(t), v_l(t), p_l(t)) \\ p_l(t+1) &= K_l(y_l(t), v_l(t), p_l(t)) \end{aligned}$$

Notice access only to local information (distributed)

Goals and Limitations

Goals: To characterize

- Throughput, delay, loss
- Fairness
- Reverse engineering: start with a given protocol
- Forward engineering: start with a desired equilibrium

Limitations:

- Congestion avoidance phase only (long-lived flows)
- Deterministic fluid model
- Average behavior
- Equilibrium properties

Utility Maximization and Equilibrium Properties

At equilibrium: $y^* = Rx^*, q^* = R^T p^*$. τ_{li}^b and τ_{li}^f set to 0.

Let $x_i^* = f_i(q_i^*)$ where f_i is a positive, decreasing function

Construct source utility function U_i such that $U_i'(x_i) = f_i^{-1}(x_i)$ (thus increasing and concave)

Equilibrium solves the local profit maximization problem over x_i:

$$\text{maximize} \;\; [U_i(x_i) - x_i q_i^*]$$

Need local profit-seeking to align with social welfare

Use duality-based pricing to solve basic NUM (last lecture)

Link Algorithm

One possibility: apply gradient algorithm to the Lagrangian:

$$L(x,p) = \sum_i [U_i(x_i) - q_i x_i] + \sum_l p_l c_l$$

Congestion price updates by gradient methods:

$$\lambda_l(t+1) = \left[\lambda_l(t) - \alpha(t)\left(c_l - \sum_{s:l\in L(s)} x_s(\lambda^s(t))\right)\right]^+, \ \forall l$$

In general: complementary slackness condition for dual optimality

$p_l^* > 0$ indicates $y_l^* = c_l$ (link saturation) and

$y_l^* < c_l$ indicates $p_l^* = 0$ (buffer clearance)

Any link algorithm satisfying complementary slackness defines an equilibrium, but may not converge

TCP Reno

RTT τ_i assumed constant. Loss probabilities assumed to be small:

$$q_i(t) \approx \sum_{l \in L_i} p_l(t)$$

Since $(1 - q_i(t))x_i(t)$ of ACK are positive, each incrementing window size by $1/w_i(t)$, and $q_i(t)x_i(t)$ are negative, each halfing the window, using $x_i(t) = w_i(t)/\tau_i$, we have

$$\dot{x}_i = \frac{1 - q_i(t)}{\tau_i^2} - \frac{1}{2} q_i(t) x_i^2(t)$$

At equilibrium:

$$q_i^* = \frac{2}{2 + (\tau_i x_i^*)^2}$$

Using $U_i'(x_i^*) = q_i^*$, utility function for TCP Reno

$$U_i(x_i) = \frac{\sqrt{2}}{\tau_i} \tan^{-1}\left(\frac{\tau_i x_i}{\sqrt{2}}\right)$$

TCP Vegas

Window size w_s

Propagation delay d_s. Expected rate $\frac{w_s(t)}{d_s}$

Queuing delay q_s and total delay D_s. Actual rate $\frac{w_s(t)}{D_s}$

Source algorithm:

$$w_s(t+1) = \begin{cases} w_s(t) + \frac{1}{D_s(t)} & \text{if } \frac{w_s(t)}{d_s} - \frac{w_s(t)}{D_s(t)} < \alpha_s \\ w_s(t) - \frac{1}{D_s(t)} & \text{if } \frac{w_s(t)}{d_s} - \frac{w_s(t)}{D_s(t)} > \alpha_s \\ w_s(t) & \text{else.} \end{cases}$$

Equilibrium round-trip time and window size satisfy:

$$\frac{w_s^*}{d_s} - \frac{w_s^*}{D_s^*} = \alpha_s$$

TCP Vegas: Log Utility Function

$$U_s(x_s) = \alpha_s d_s \log x_s$$

Complementary slackness satisfied. For KKT, need to also check

$$U_s'(x_s^*) = \frac{\alpha_s d_s}{x_s^*} = \sum_{l \in s} p_l^*$$

Let b_l^* be equilibrium backlog at link l. Window size equals bandwidth-delay product plus total backlog:

$$w_s^* = x_s^* d_s + \sum_{l \in s} \frac{x_s^*}{c_l} b_l^*$$

Using $x_s = w_s / D_s$, we have

$$\alpha_s = \frac{w_s^*}{d_s} - \frac{w_s^*}{D_s^*} = \frac{1}{d_s}(w_s^* - x_s^* d_s) = \frac{1}{d_s}\left(\sum_{l \in s} \frac{x_s^*}{c_l} b_l^*\right)$$

since $p_l^* = \frac{b_l^*}{c_l}$ (dual variable is queuing delay)

TCP Vegas: Source-Link Algorithms

Primal variable is source rate, updated by source algorithm:

$$
\begin{aligned}
w_s(t+1) &= [w_s(t) + v_s(t)]^+ \\
v_s(t) &= \frac{1}{d_s + q_s(t)} \left[\mathbf{1}(x_s(t)q_s(t) < \alpha_s d_s) - \mathbf{1}(x_s(t)q_s(t) > \alpha_s d_s)\right] \\
x_s(t) &= \frac{w_s(t)}{d_s + q_s(t)}
\end{aligned}
$$

Dual variable is queuing delay, updated by link algorithm:

$$
p_l(t+1) = \left[p_l(t) + \frac{1}{c_l}(y_l(t) - c_l)\right]^+
$$

Equilibrium: $x_s^* = \frac{\alpha_s d_s}{q_s^*}$

TCP Reno and Vegas

TCP Reno (with Drop Tail or RED):

- Source utility: arctan
- Link price: packet loss

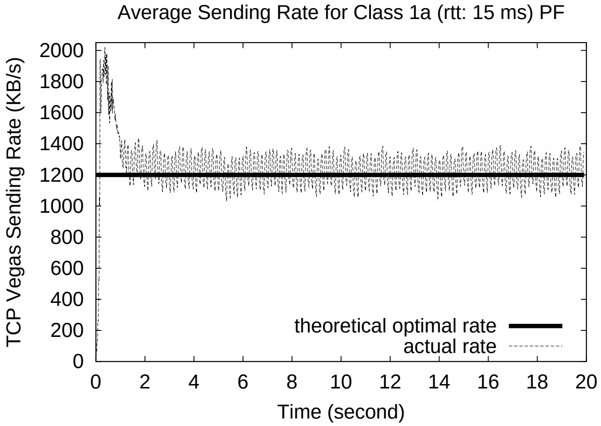
TCP Vegas (with FIFO)

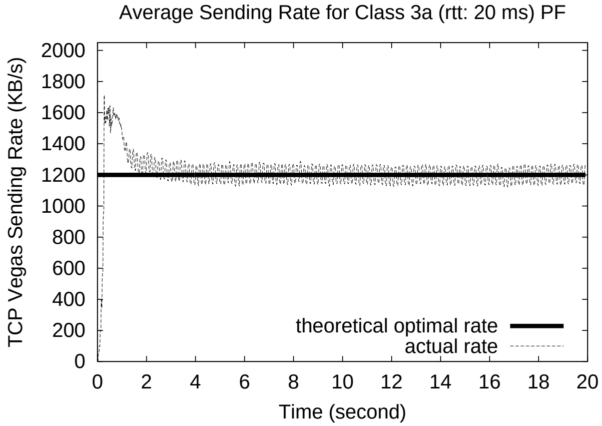
- Source utility: weighted log
- Link price: queuing delay

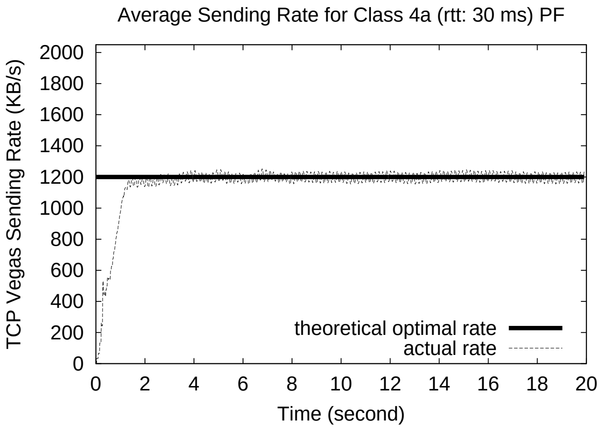
Implications: Delay, Loss, Fairness

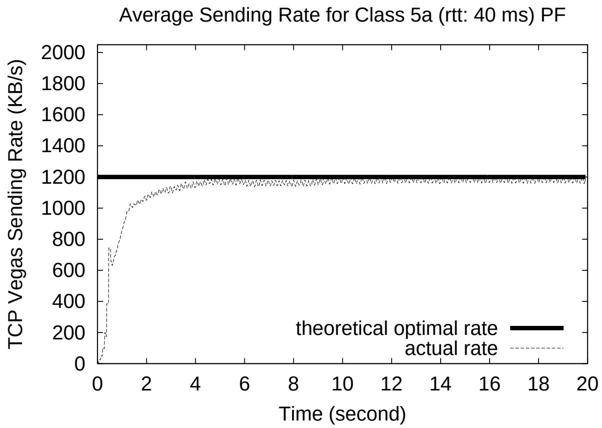
- TCP Reno: equilibrium loss probability is independent of link algorithms and buffer sizes. Increasing buffer sizes alone does not decrease equilibrium loss probability (buffer just fills up)
- TCP Reno: discriminates against connections with large propagation delays
- Desirable to decouple link pricing from loss
- TCP Vegas: bandwidth-queuing delay product equals number of packets buffered in the network $x_s^* q_s^* = \alpha_s d_s$
- TCP Vegas: achieves proportional fairness
- TCP Vegas: gradient method for updating dual variable. Converges with the right scaling (γ small enough)
- Persistent congestion, TCP-friendly protocols ...

Numercal Example: Single Bottleneck

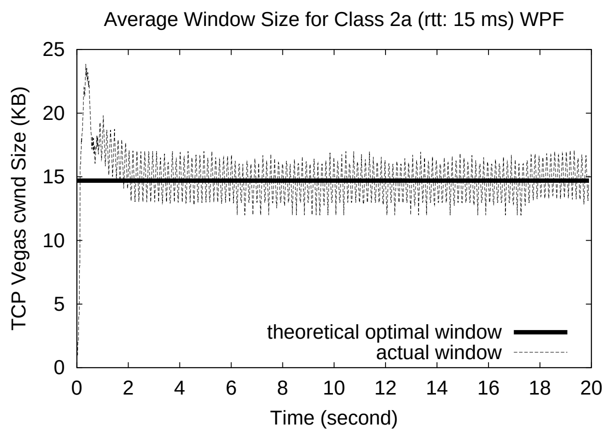


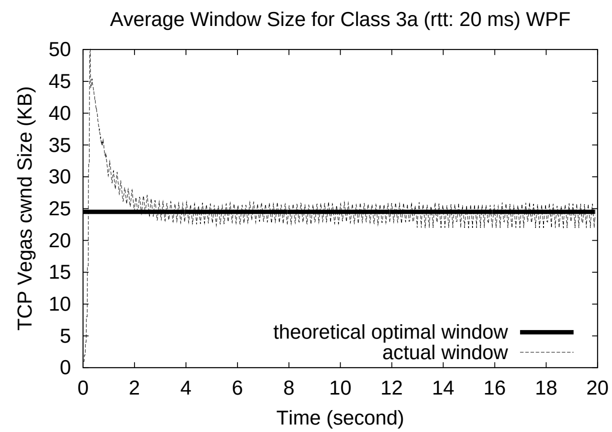


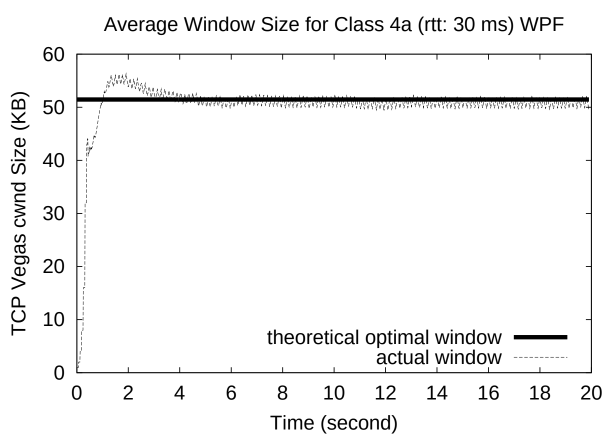


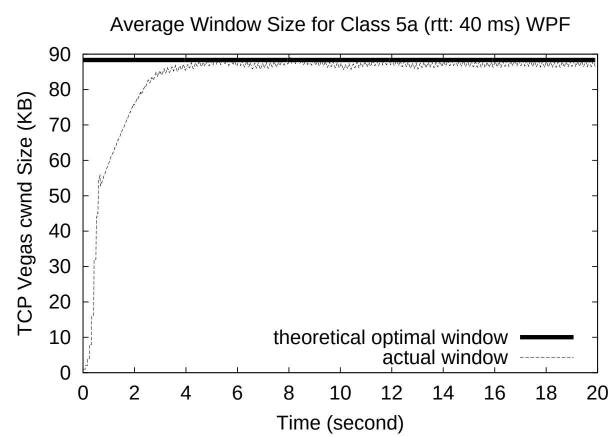


Numercal Example: General Cases



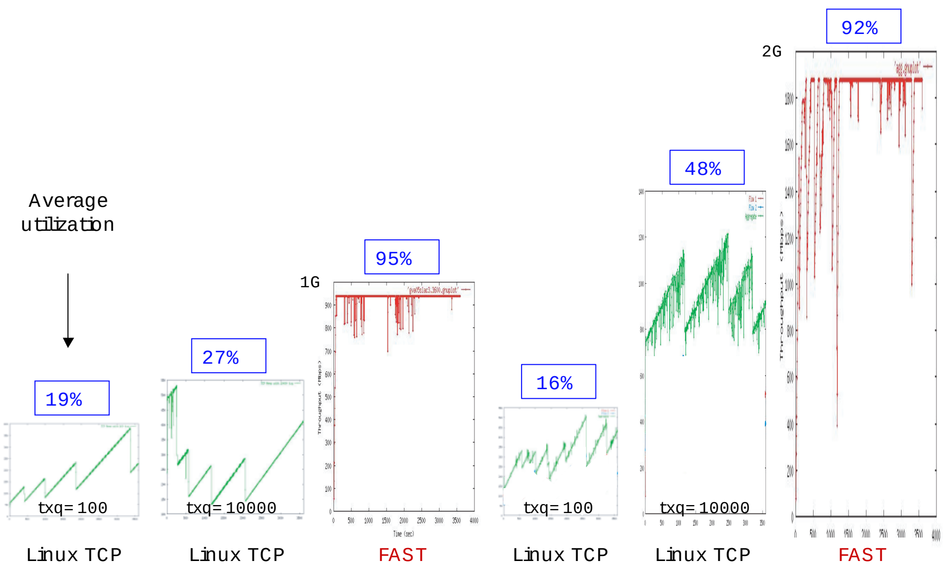






Stability and Dynamics

- Optimization theoretic analysis has focused on equilibrium state
- TCP congestion control may oscillates
- Use control theoretic ideas to stabilize TCP
- FAST TCP implemented in real networks, increasing bandwidth utilization efficiency from 20% to 90%



Three Meanings of the Course Title

- Forward engineering: Formulate a communication systems problem as an optimization problem and solve it

Example: information theory, physical layer signal processing, routing ...

- Reverse engineering: Given a network protocol, interpret it as a distributed algorithm solving an implicit optimization problem

Example: TCP congestion control (can also go the reverse direction: start with U_s and find out F_i)

- Extension: Extend the underlying theory by generalizing using optimization theory

Example: detection and estimation

Lecture Summary

- TCP congestion control is implicitly maximizing network utility over linear flow constraints, where each source updates source rate (primal variable) and each link updates congestion measure (dual variable)
- NUM solution algorithm approximated by TCP window update and queue management
- Models, understands and improves TCP congestion control protocols

Readings: S. H. Low, F. Paganini, J. C. Doyle, "Internet congestion control," *IEEE Control Systems Magazine,* Feb. 2002.

S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Trans. Networking,* August 2003.