Network Transport Layer: Network Resource Allocation Framework

Qiao Xiang, Congming Gao

https://sngroup.org.cn/courses/cnns-xmuf23/index.shtml

11/23/2023

This deck of slides are heavily based on CPSC 433/533 at Yale University, by courtesy of Dr. Y. Richard Yang.

Outline

- Admin and recap
- TCP Congestion Control

Admin

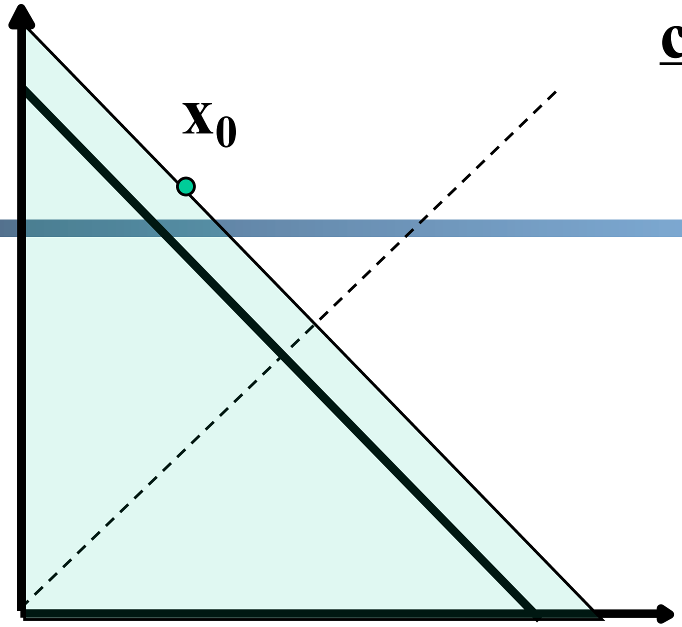
- Guest lectures (tentative schedule subject to change)
  - 11/28, Yutong Liu, SJTU, Internet of Things

Recap: Transport Design

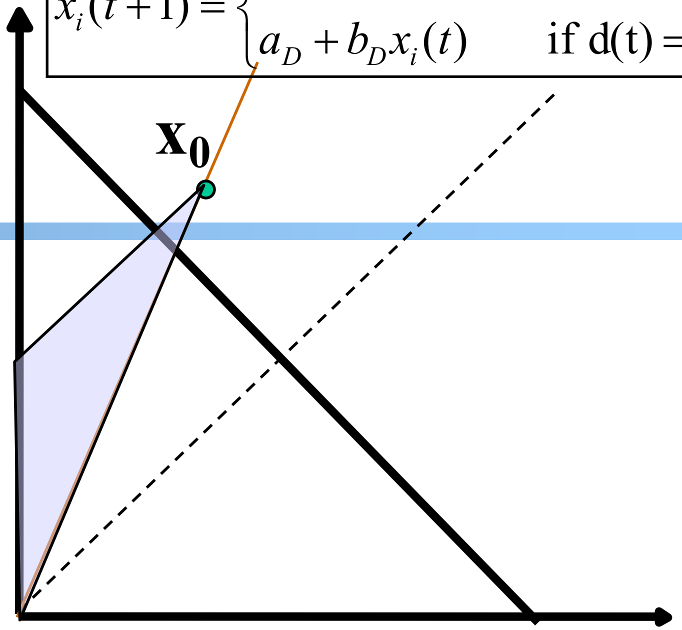
- Basic structure/reliability: sliding window protocols
- Determine the “right” parameters
  - Timeout
    - mean + variation
  - Sliding window size
    - Related w/ congestion control or more generally resource allocation
      - Bad congestion control can lead to congestion collapse (e.g., zombie packets)
    - Goals: distributed algorithm to achieve fairness and efficiency

congestion

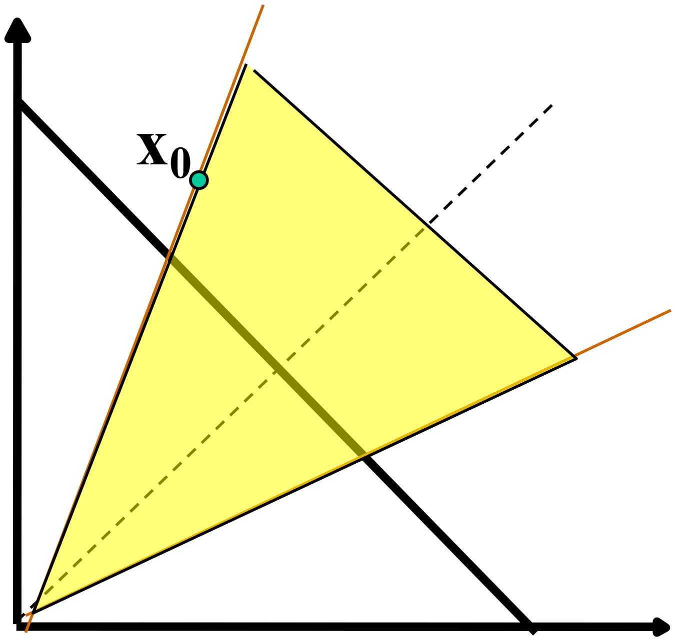
$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } \mathrm{d(t) = no\ cong.} \\ a_D + b_D x_i(t) & \text{if } \mathrm{d(t) = cong.} \end{cases}$$



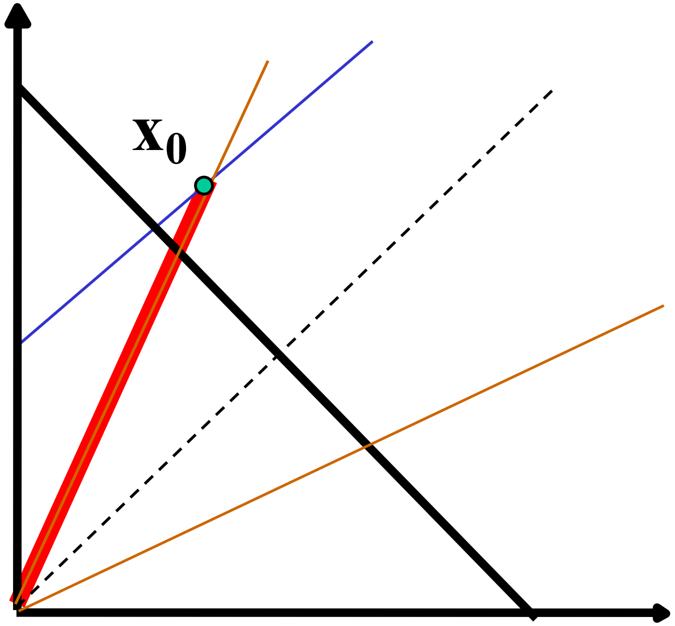
efficiency



efficiency: distributed linear rule



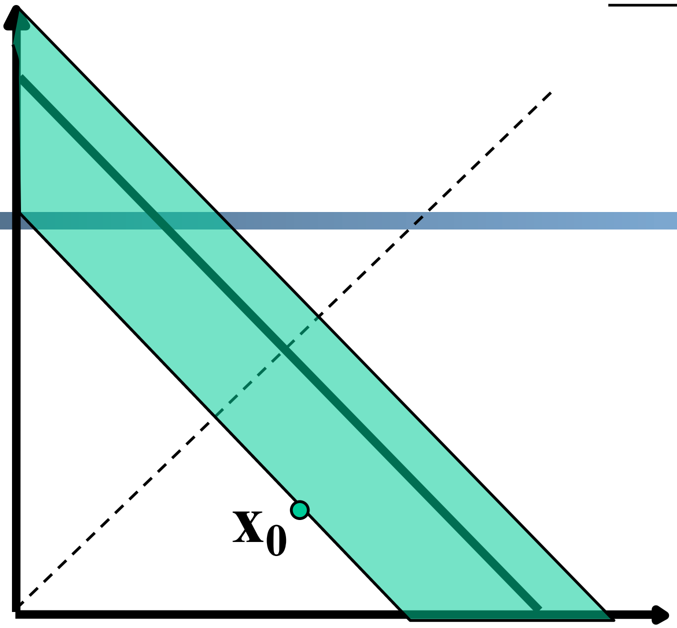
fairness



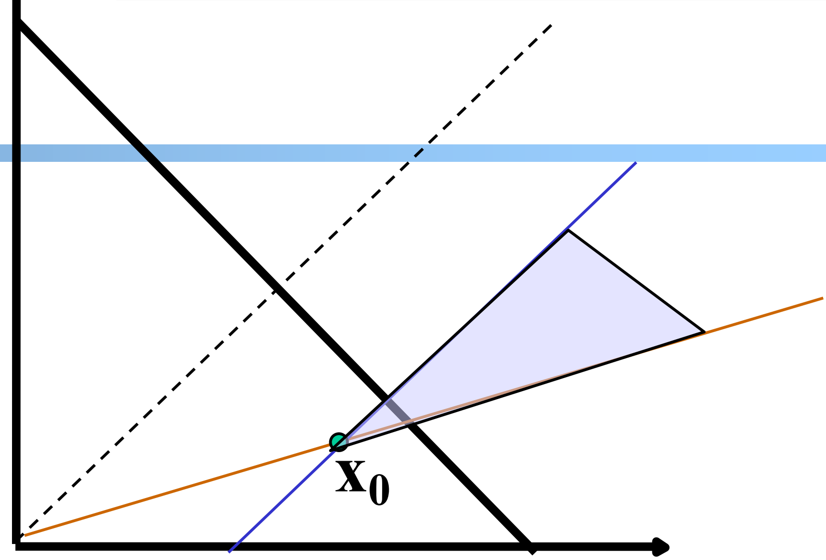
intersection

no-congestion

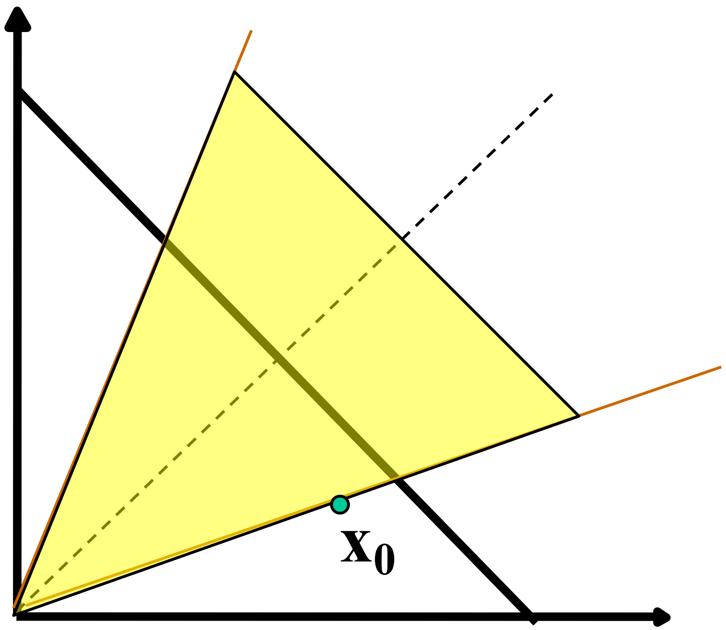
$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } \mathrm{d(t)} = \text{no cong.} \\ a_D + b_D x_i(t) & \text{if } \mathrm{d(t)} = \text{cong.} \end{cases}$$



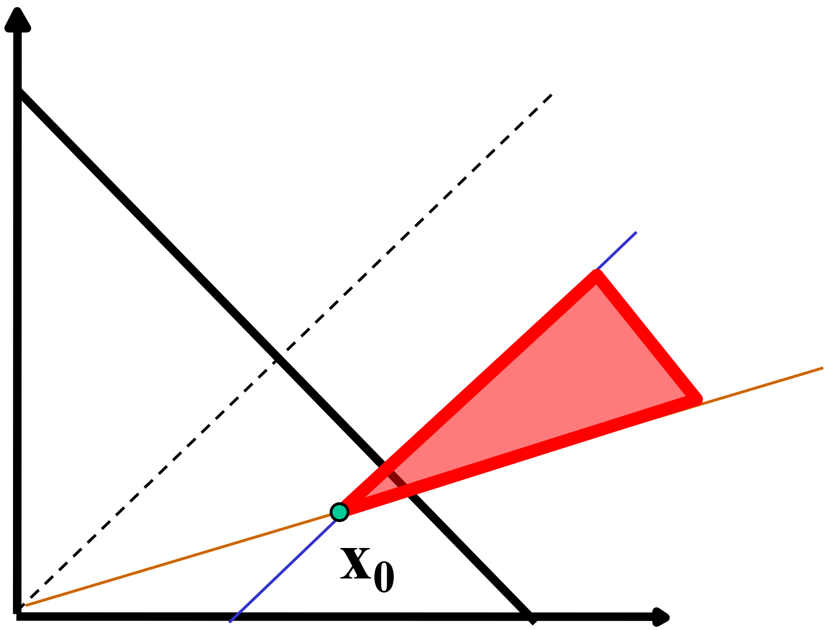
efficiency



efficiency: distributed linear rule



fairness



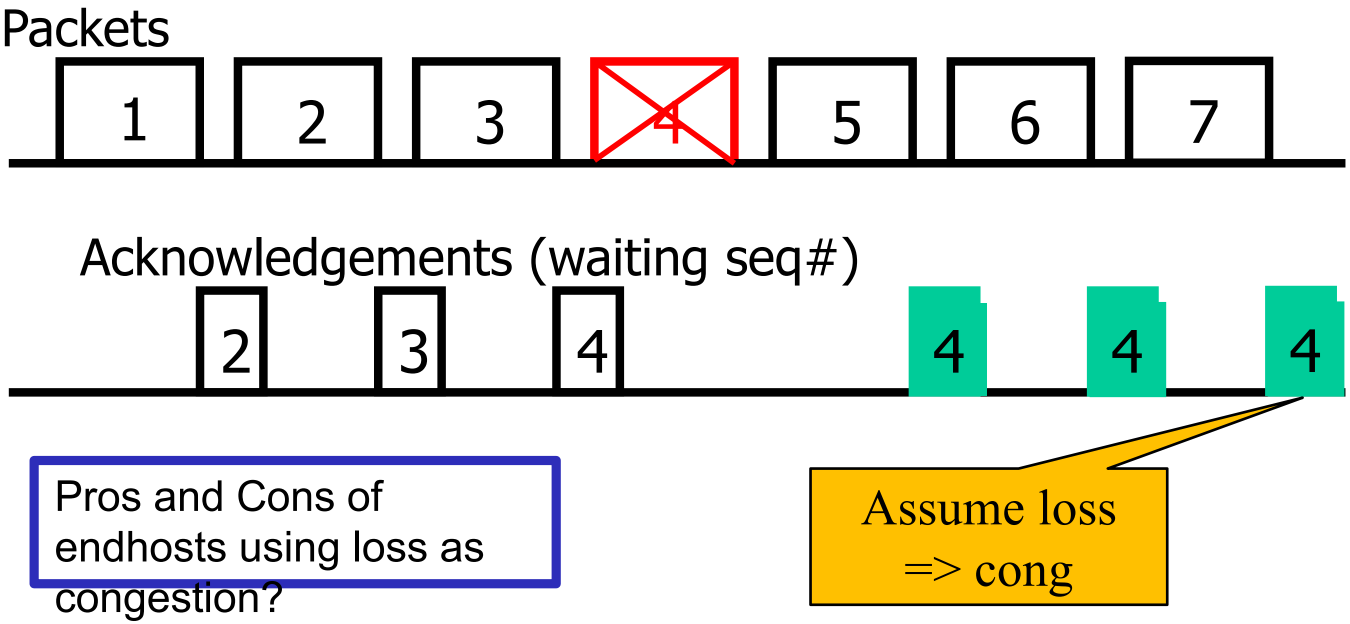
convergence

Mapping A(M)I-MD to Protocol

- Basic questions to look at:
  - How to obtain d(t)--the congestion signal?
  - What values do we choose for the formula?
  - How to map formula to code?

$$x_i(t+1) = \begin{cases} a_I + x_i(t) & \text{if } \mathrm{d(t)} = \text{no cong.} \\ b_D x_i(t) & \text{if } \mathrm{d(t)} = \text{cong.} \end{cases}$$

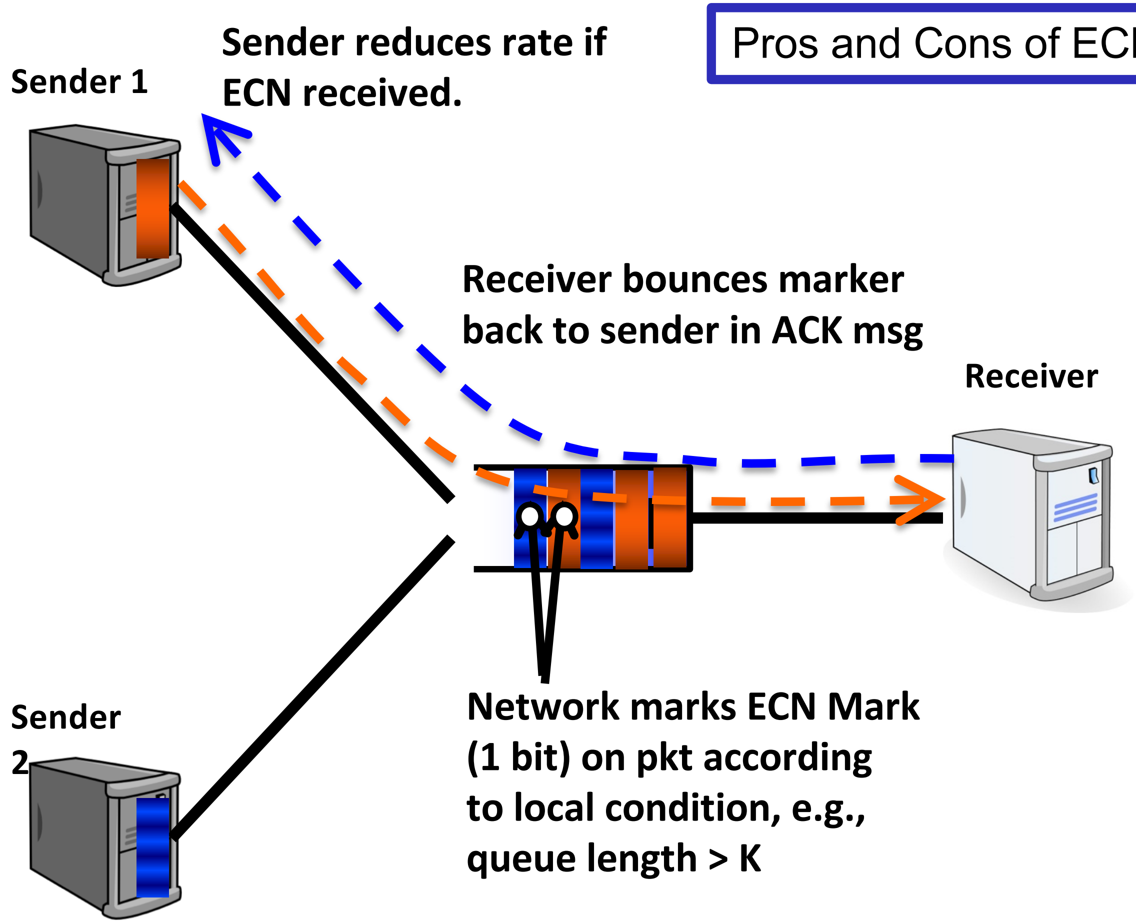
Obtain d(t) Approach 1: End Hosts Consider Loss as Congestion



Pros and Cons of endhosts using loss as congestion?

Obtain d(t) Approach 2: Network Feedback (ECN: Explicit Congestion Notification)

Pros and Cons of ECN?



Mapping A(M)I-MD to Protocol

- Basic questions to look at:
  - How to obtain d(t)--the congestion signal?
  - What values do we choose for the formula?
  - How to map formula to code?

$$x_i(t+1) = \begin{cases} a_I + x_i(t) & \text{if } \mathrm{d(t)} = \text{no cong.} \\ b_D x_i(t) & \text{if } \mathrm{d(t)} = \text{cong.} \end{cases}$$

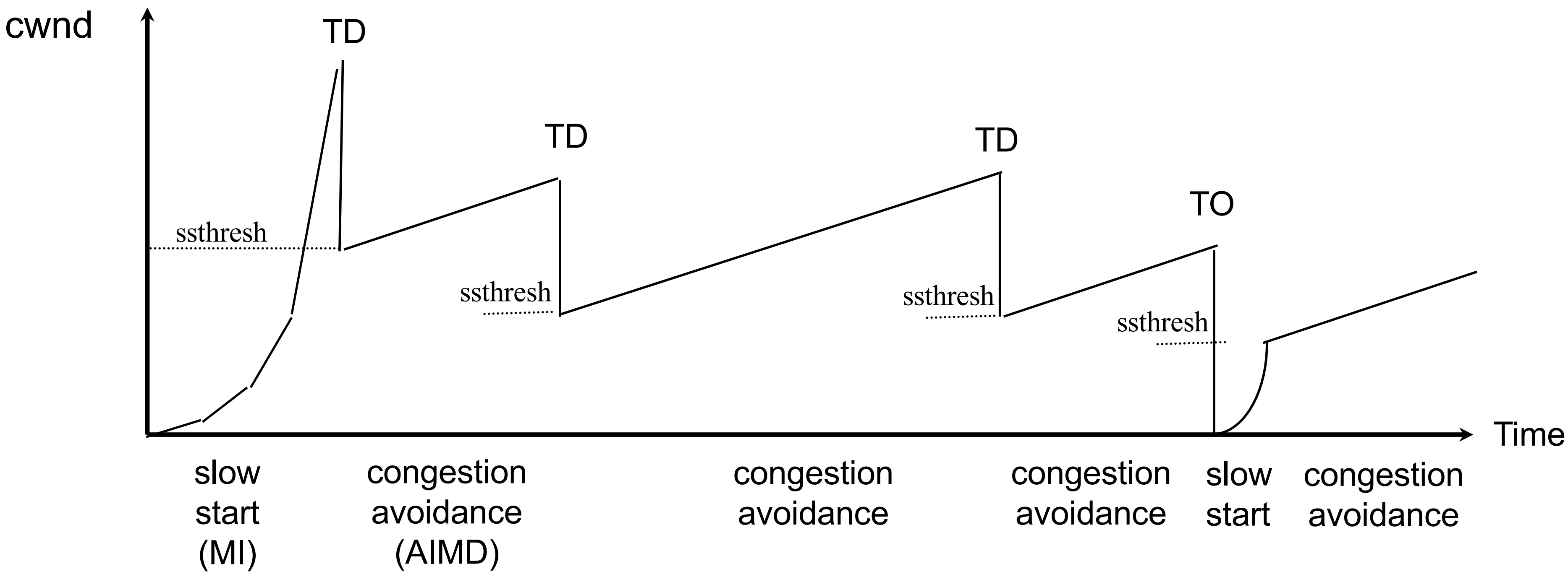
TCP/Reno Formulas

- Multiplicative Increase (MI)
  - double *the rate:* $x(t+1) = 2\, x(t)$
- Additive Increase (AI)
  - Linear increase *the rate:* $x(t+1) = x(t) + 1$
- Multiplicative decrease (MD)
  - half *the rate:* $x(t+1) = 1/2\, x(t)$

TCP/Reno Full Alg

```
Initially:
    cwnd = 1;
    ssthresh = infinite (e.g., 64K);
For each newly ACKed segment:
    if (cwnd < ssthresh)          // slow start: MI
        cwnd = cwnd + 1;
    else
                                  // congestion avoidance; AI
        cwnd += 1/cwnd;
Triple-duplicate ACKs:
                                  // MD
    cwnd = ssthresh = cwnd/2;
Timeout:
    ssthresh = cwnd/2;            // reset
    cwnd = 1;
(if already timed out, double timeout value; this is called exponential backoff)
```

TCP/Reno: Big Picture



TD: Triple duplicate acknowledgements
TO: Timeout

Outline

- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
    - design
    - *analysis*

Objective

- To understand
  - the throughput of TCP/Reno as a function of RTT (RTT), loss rate (p) and packet size
  - the underlying queue dynamics

- We will analyze TCP/Reno under two different setups

TCP/Reno Throughput Analysis

- Given mean packet loss rate p, mean round-trip time RTT, packet size S
- Consider only the congestion avoidance mode (long flows such as large files)
- Assume no timeout
- Assume mean window size is W_m segments, each with S bytes sent in one RTT:

$$\boxed{\text{Throughput} = \frac{W_m * S}{RTT} \text{ bytes/sec}}$$

Outline

- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
    - design
    - analysis
      - small fish in a big pond
        - loss rate given from the environment

TCP/Reno Throughput Modeling (Fixed, Given Loss Rate)

$$\Delta W = \begin{cases} \frac{1}{W} & \text{if the packet is not lost} \\ -\frac{W}{2} & \text{if packet is lost} \end{cases}$$

$$mean\ of\ \Delta W = (1-p)\tfrac{1}{W} + p(-\tfrac{W}{2}) = 0$$

$$\Rightarrow \quad mean\ of\ W = \sqrt{\frac{2(1-p)}{p}} \approx \frac{1.4}{\sqrt{p}}, \text{ when } p \text{ is small}$$

$$\Rightarrow \quad \boxed{throughput \approx \frac{1.4S}{RTT\sqrt{p}}, \text{ when } p \text{ is small}}$$

This is called the TCP throughput sqrt of loss rate law.

Exercise: Application of Analysis

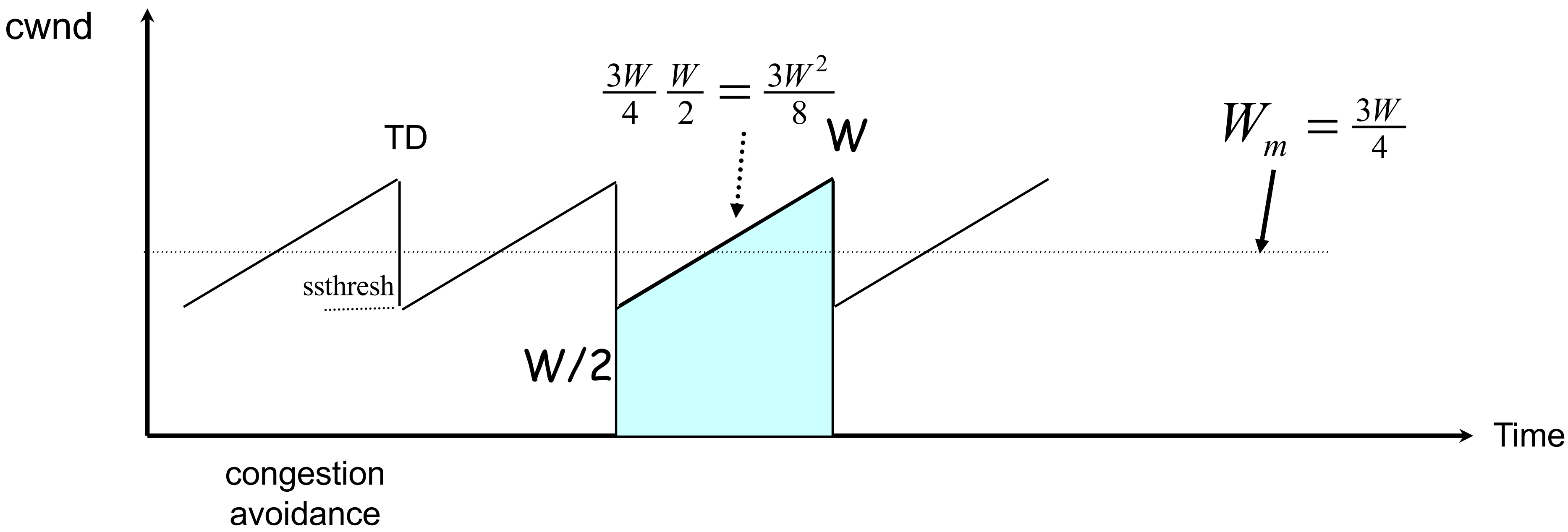
- State of art network link can reach 100 Gbps. Assume packet size 1250 bytes, RTT 100 ms, what is the highest packet loss rate to still reach 100 Gbps?

tcp-reno-tput.xlsx

Outline

- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
    - design
    - analysis
      - small fish in a big pond
      - big fish in small pond
        - growth causes losses

TCP/Reno Throughput Modeling: Relating W with Loss Rate p



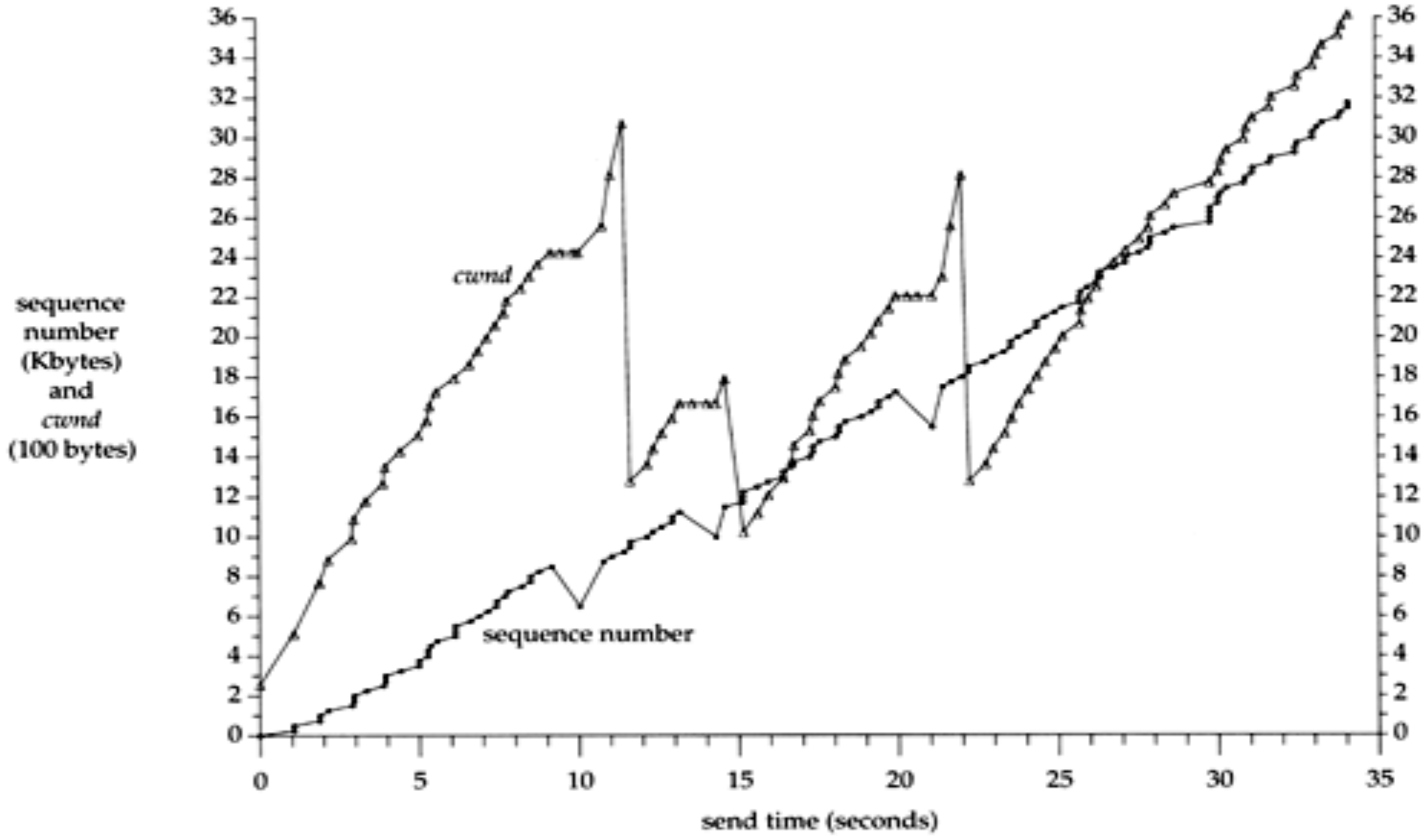
Total packets sent per cycle = (W/2 + W)/2 * W/2 = $3W^2/8$

Assume one loss per cycle => p = $1/(3W^2/8) = 8/(3W^2)$

$$\Rightarrow W = \frac{\sqrt{8/3}}{\sqrt{p}} = \frac{1.6}{\sqrt{p}}$$

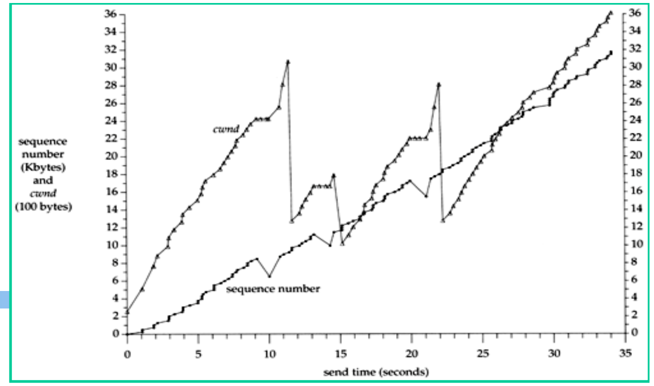
$$\Rightarrow throughput = \frac{S}{RTT}\frac{3}{4}\frac{1.6}{\sqrt{p}} = \boxed{\frac{1.2S}{RTT\sqrt{p}}}$$

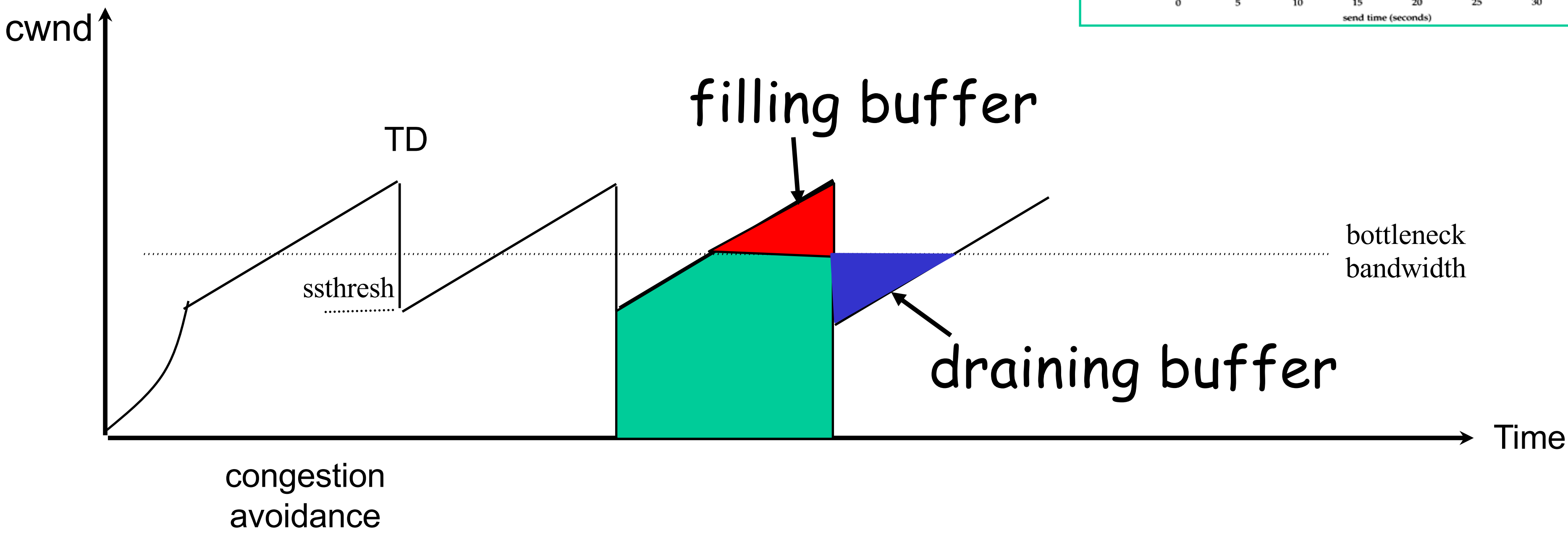
A Puzzle: cwnd and Rate of a TCP Session



Question: although cwnd fluctuates widely (i.e., cut to half), why can the sending rate stay relatively smooth?

TCP/Reno Queueing Dynamics





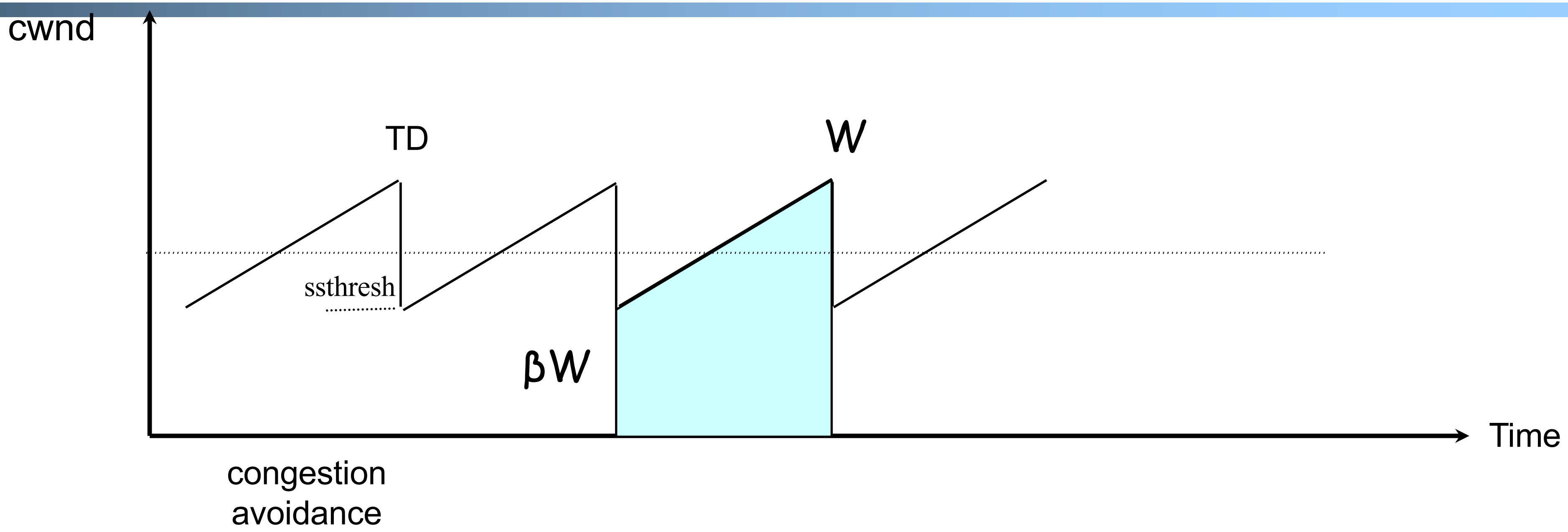
If the buffer at the bottleneck is large enough, the buffer is never empty (not idle), during the cut-to-half to "grow-back" process.

Exercise: How big should the buffer be to achieve full utilization?

Design

- Assume a generic AIMD alg:
  - increase to W + α after each successful RTT
  - reduce to β W after each loss event
- Q: What value β gives higher utilization (assume small/zero buffer)?
- Q: Assume picking a high value β, how to make the alg TCP friendly (same throughput as α=1, β=0.5)?

Generic AIMD and TCP Friendliness



Total packets sent per cycle = $\frac{\beta W+W}{2}\;\frac{(1-\beta)W}{\alpha} = \frac{(1-\beta)(1+\beta)}{2\alpha}\mathrm{w}^2$

Assume one loss per cycle $\mathrm{p} = \frac{2\alpha}{(1-\beta)(1+\beta)\mathrm{w}^2}$ $\quad \mathrm{w} = \sqrt{\frac{2\alpha}{(1-\beta)(1+\beta)\mathrm{p}}}$

$$\mathrm{tput} = \frac{W_m S}{RTT} = \frac{S}{RTT}\;\frac{(1+\beta)W}{2} = \frac{S}{RTT}\sqrt{\frac{\alpha(1+\beta)}{2(1-\beta)\mathrm{p}}}$$

TCP friendly => $\quad \alpha = 3\,\frac{1-\beta}{1+\beta}$

Outline

- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic

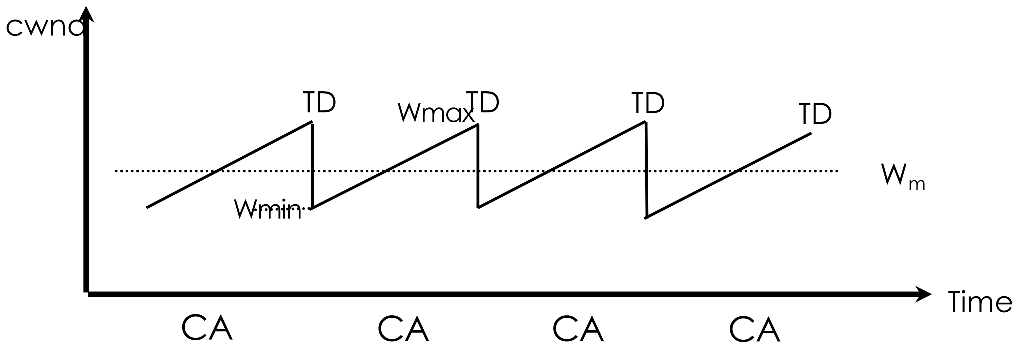
TCP Cubic

- Designed in 2008
- Default for Linux
- Most sockets in MAC appear to use cubic as well
  - sw_vers
  - sysctl -a

TCP Cubic Goals

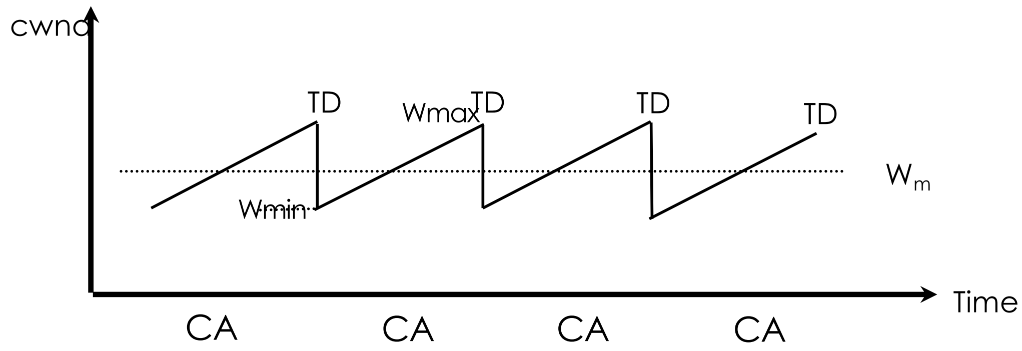
❑ Improve TCP efficiency over fast, long-distance links	Smaller reduction, longer stay at BDP, faster than linear increase---cubic function
❑ TCP friendliness	Follows TCP if TCP gives higher rate
❑ Fairness of flows w/ different RTTs	Window growth depends on real-time (from congestion-epoch through synchronized losse

TCP BIC Algorithm



- Setting
  - W_{max} = cwnd size before reduction
    - Too big
  - $W_{min} = \beta * W_{max}$ – just after reduction, where β is multiplicative decrease factor
    - Small
- Basic idea
  - binary search between W_{max} and W_{min}

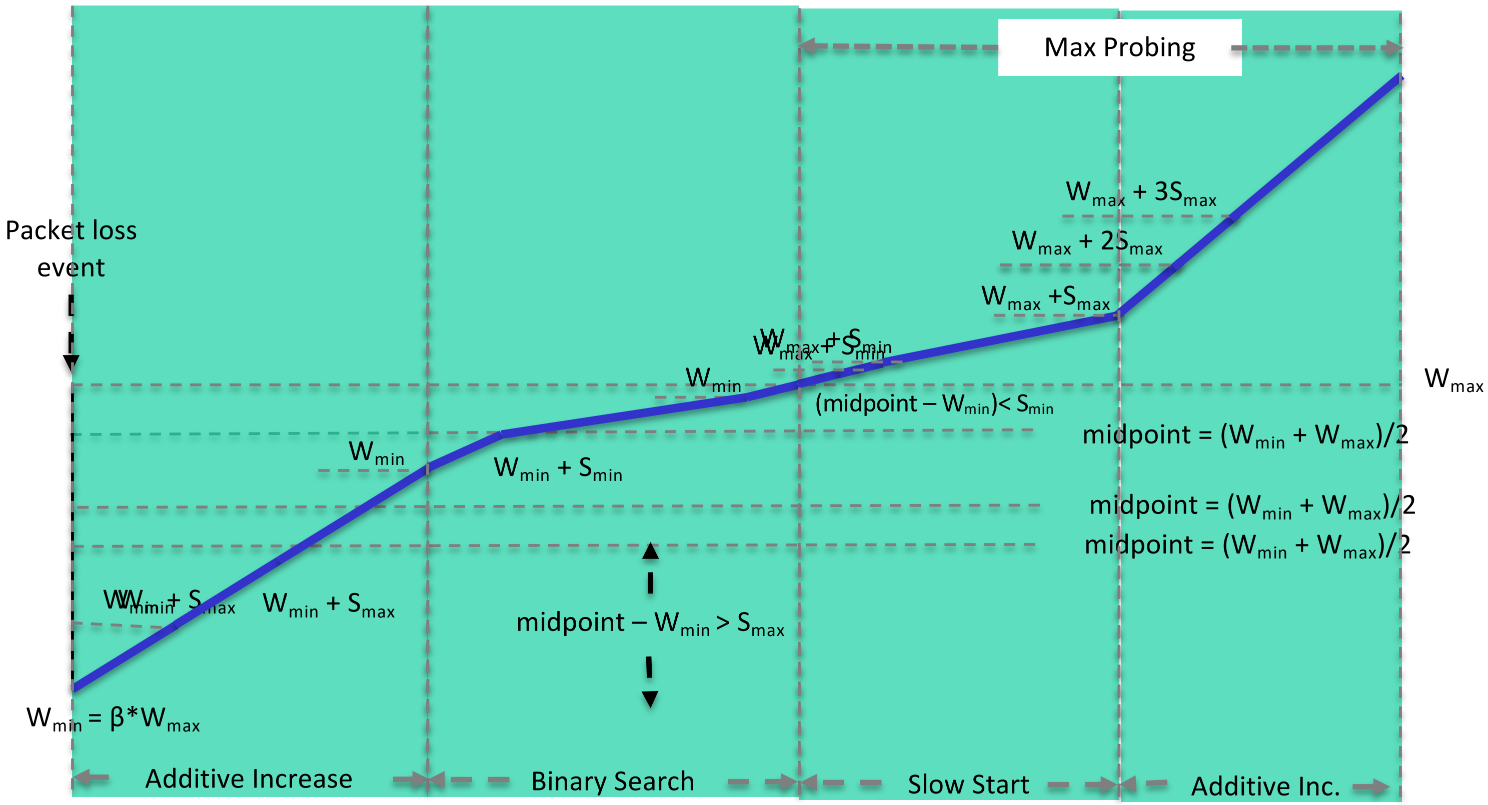
TCP BIC Algorithm: Issues



- Pure binary search (jump from W_{min} to (W_{max} and W_{min})/2) may be too aggressive
  - Use a large step size Smax

- What if you grow above W_{max}?
  - Use binary growth (slow start) to probe more

TCP BIC Algorithm



TCP BIC Algorithm

```
while (cwnd < Wmax) {
     if ( (midpoint - Wmin) > Smax )
          cwnd = cwnd + Smax
      else
          if ((midpoint - Wmin) < Smin)
                cwnd = Wmax
          else
                cwnd = midpoint
     if (no packet loss)
          Wmin = cwnd
     else
          Wmin = β*cwnd
          Wmax = cwnd
     midpoint = (Wmax + Wmin)/2
 }
```

Additive Increase

Binary Search

TCP BIC Algorithm: Probe

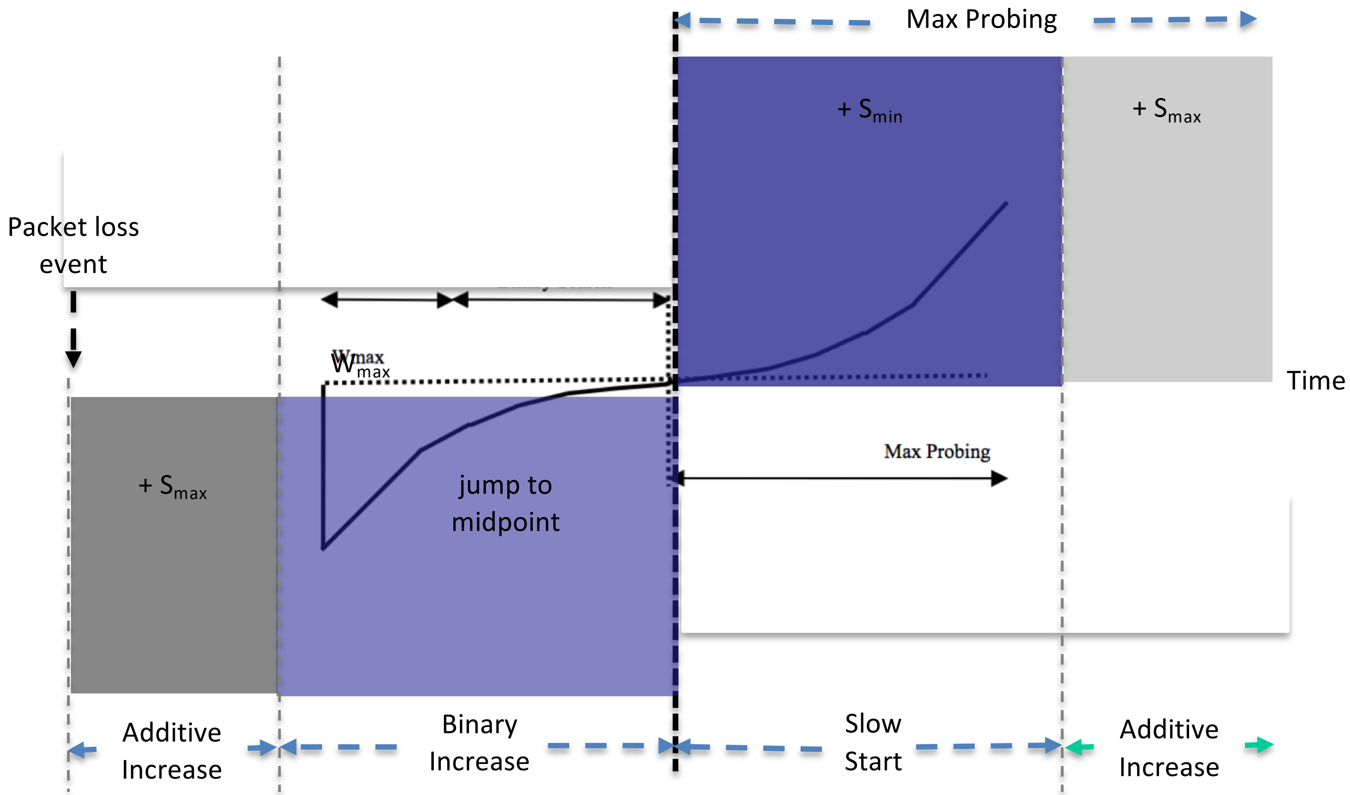
```
while (cwnd >= W_max){
  if (cwnd < W_max + S_max)
        cwnd = cwnd + S_min
   else
        cwnd = cwnd + S_max

   if (packet loss)
        W_min = β*cwnd
        W_max = cwnd
}
```

Slow growth

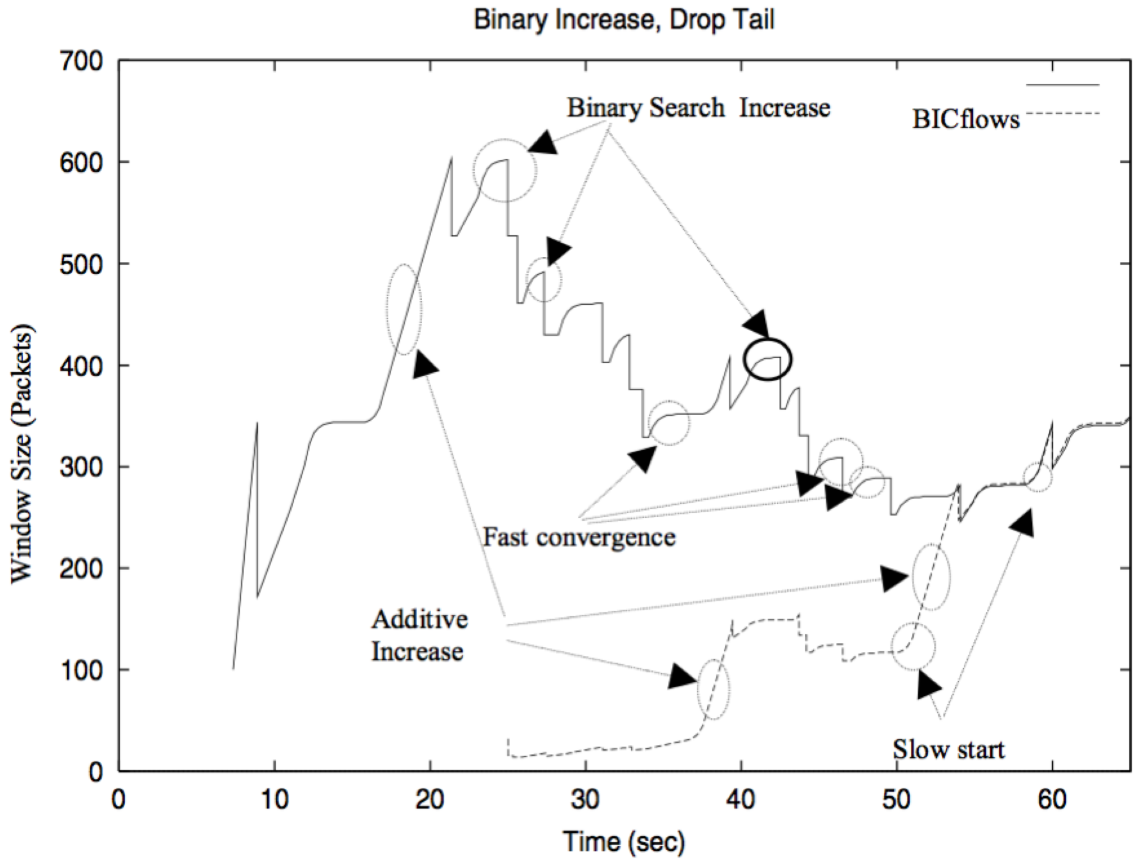
Fast growth

Max Probing

TCP BIC - Summary



TCP BIC in Action



TCP BIC Analysis

- Advantages
  - *Faster convergence at large gap*
  - *Slower growth at convergence to avoid timeout*

- *Issues*
  - Still depend on RTT
  - Complex growth function

More details: http://www.land.ufrj.br/~classes/coppe-redes-2007/projeto/BIC-TCP-infocom-04.pdf

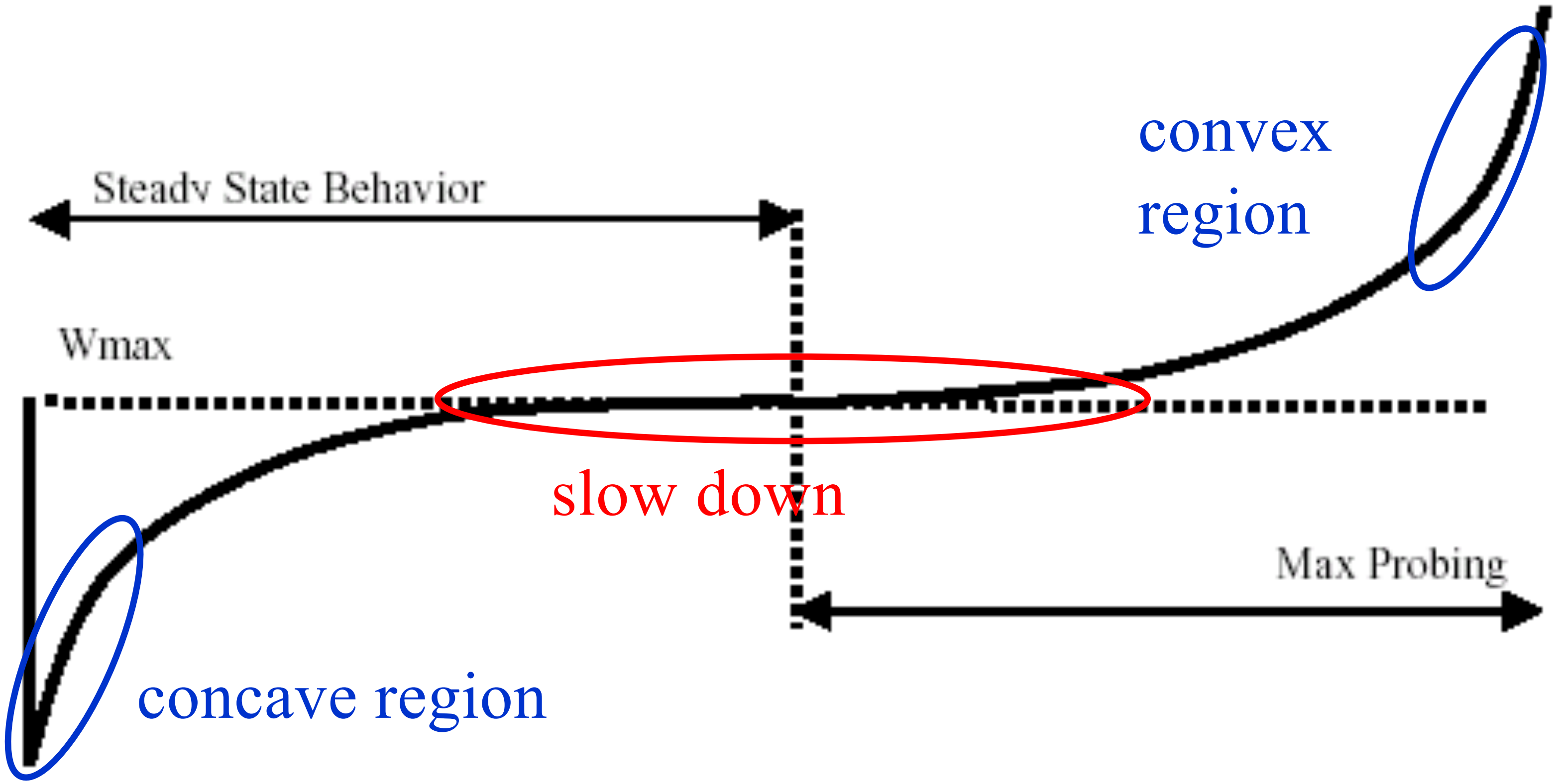
Cubic High-Level Structure

- If (received ACK && state == cong avoid)
  - Compute $W_{cubic}(t+RTT)$.
  - If cwnd < W_{TCP}
    - Cubic in TCP mode
  - If cwnd < Wmax
    - Cubic in concave region
  - If cwnd > Wmax
    - Cubic in convex region

The Cubic function

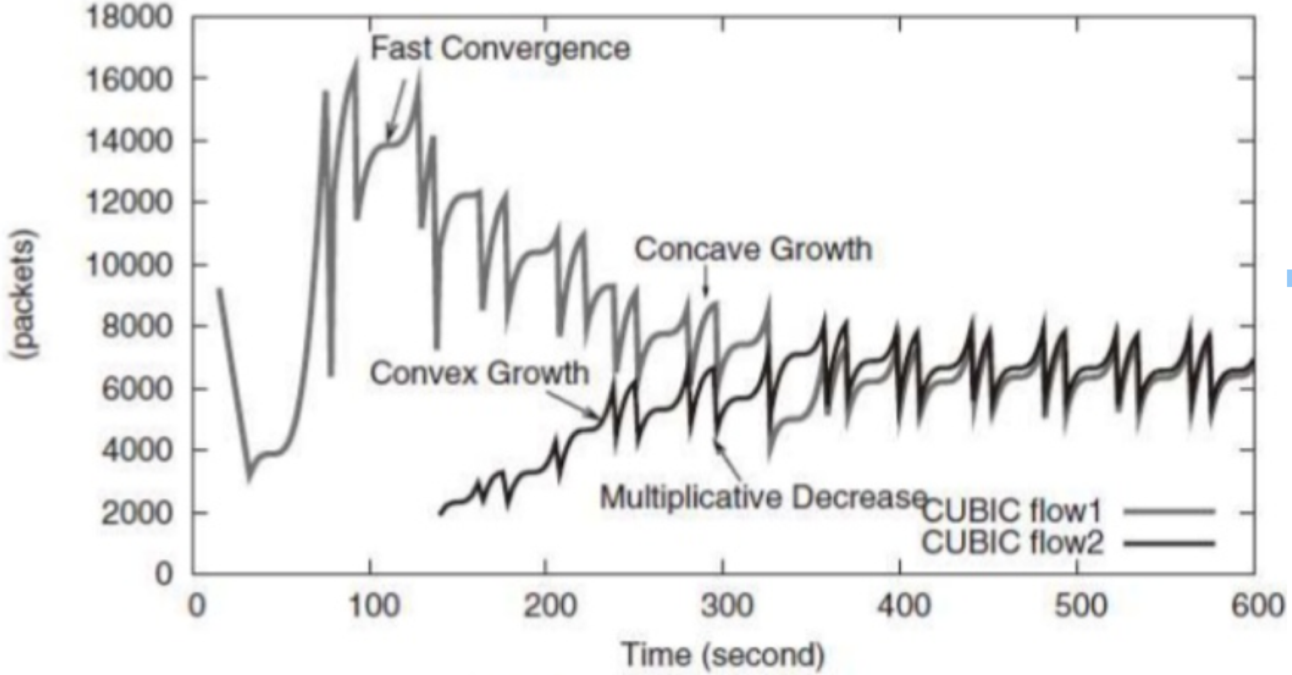
$$\beta' = 1 - \beta$$

$$W_{tcp(t)} = W max\,\beta' + 3\,\frac{1-\beta'}{1+\beta'}\,\frac{t}{RTT}$$

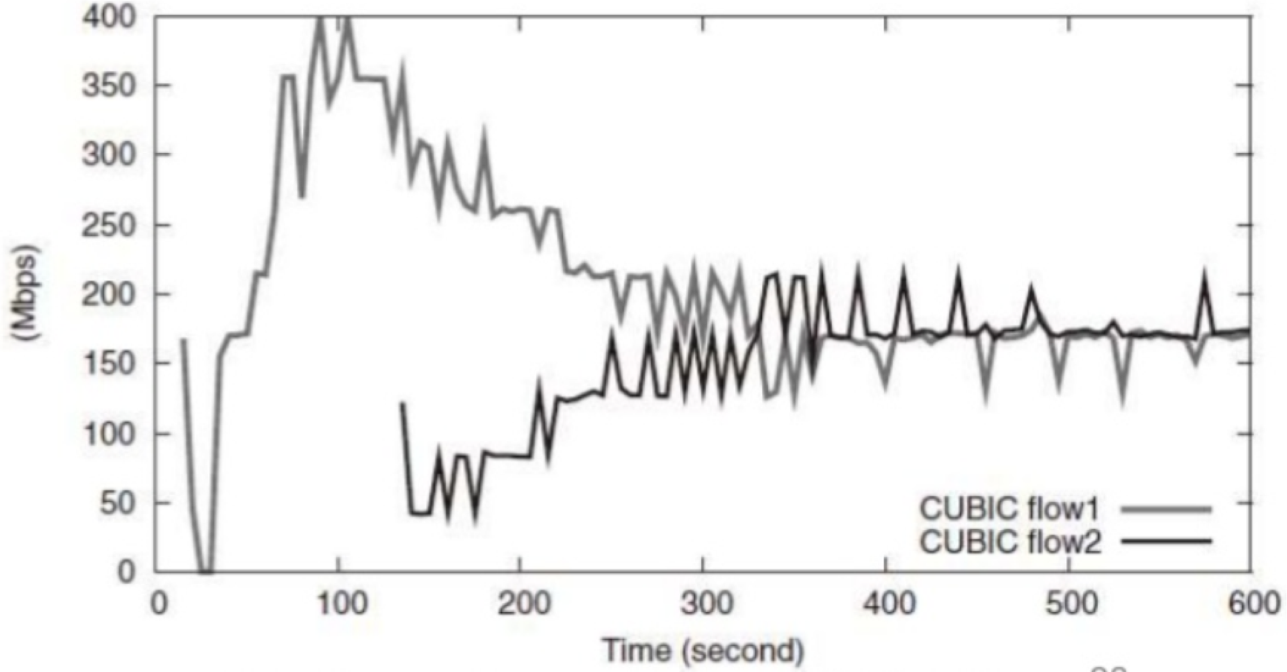


$$W_{cubic} = C(t-K)^3 + W_{\max} \qquad K = \sqrt[3]{W_{\max}\beta/C}$$

where ***C*** is a scaling factor, ***t*** is the elapsed time from the last window reduction, and ***β*** is a constant multiplication decrease factor



(a) CUBIC window curves.

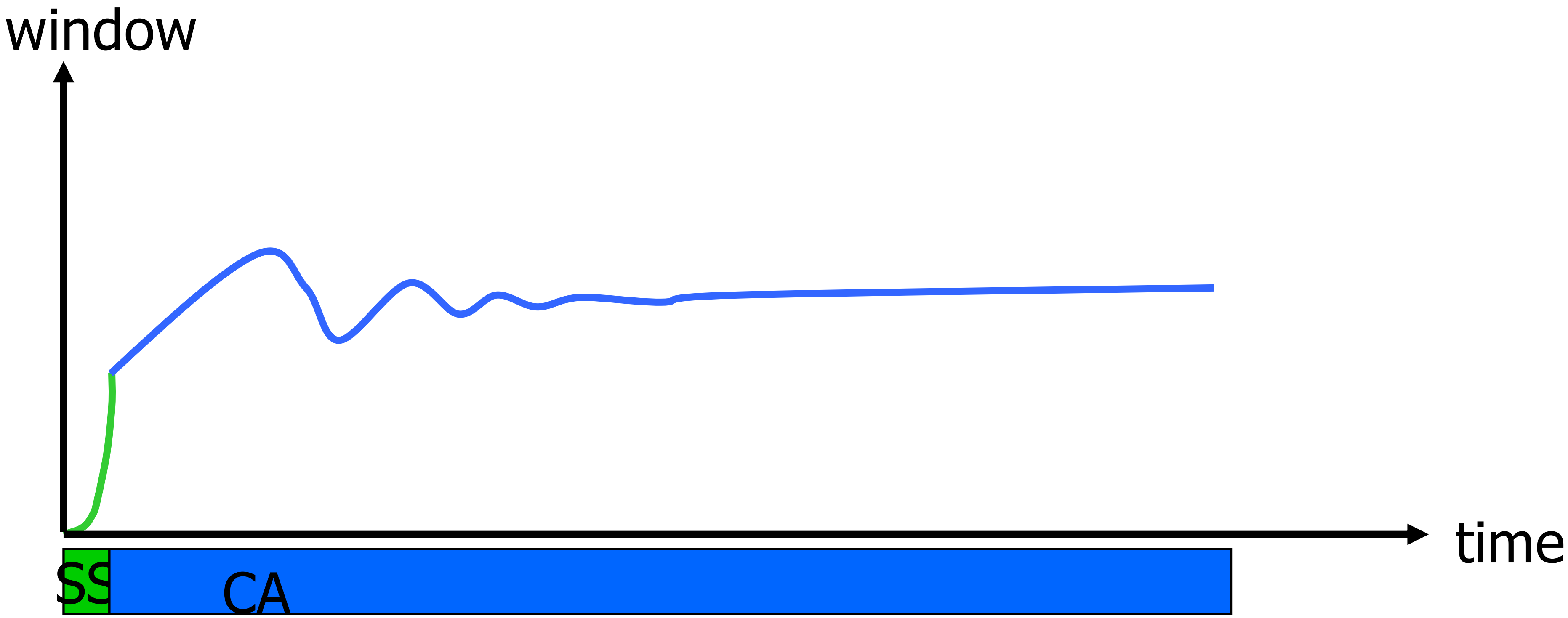


(b) Throughput of two CUBIC flows.

Outline

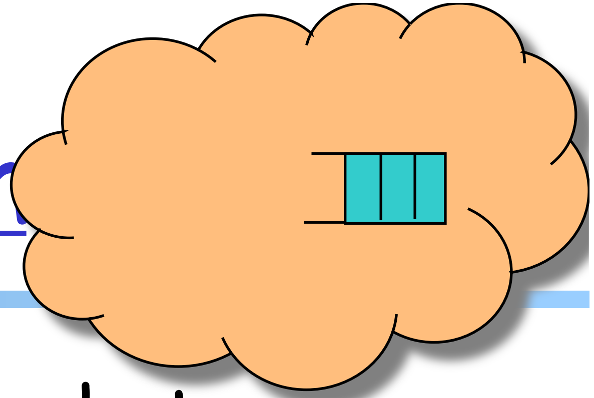
- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
  - TCP/Vegas

TCP/Vegas (Brakmo & Peterson 1994)

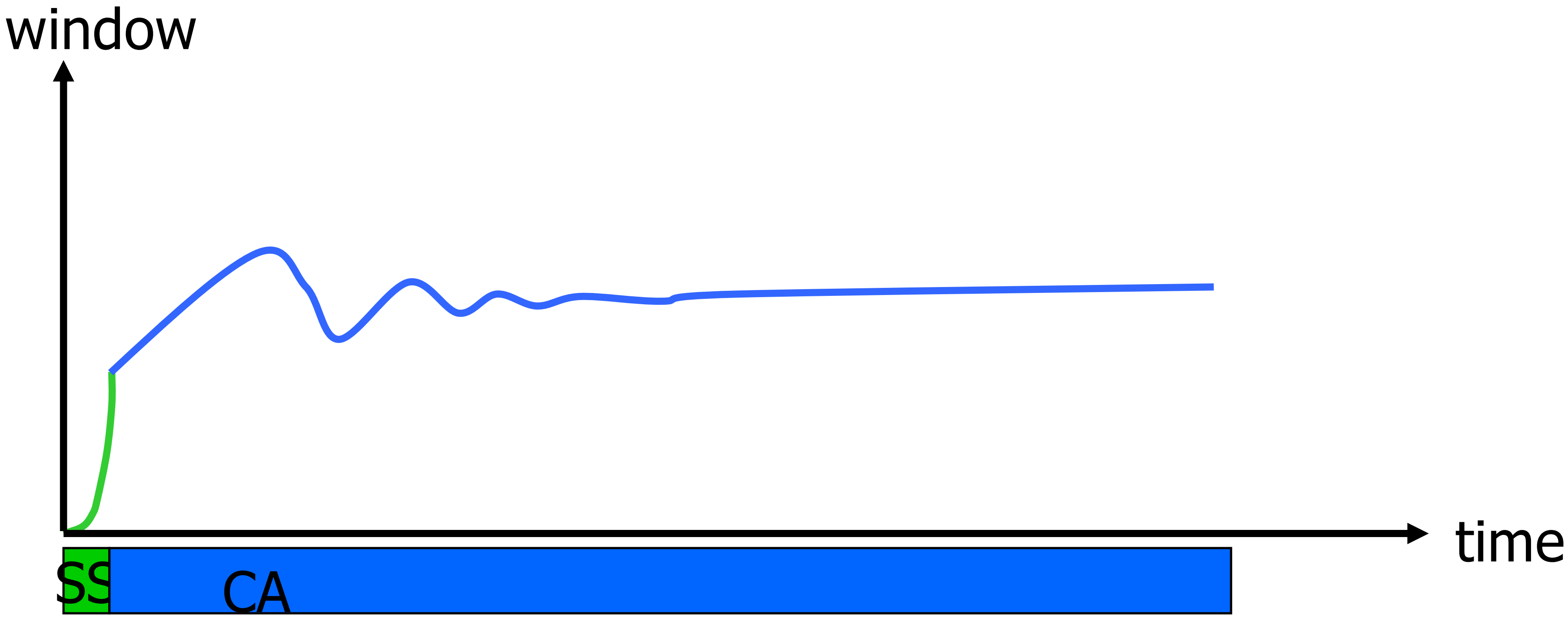


- Idea: try to detect congestion by delay before loss
- Objective: not to overflow the buffer; instead, try to maintain a *constant* number of packets in the bottleneck queue

TCP/Vegas: Key Question

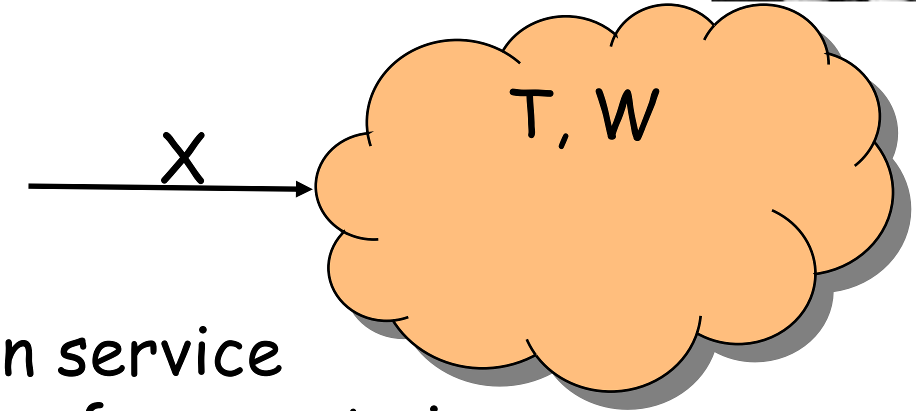


- How to estimate the number of packets queued in the bottleneck queue?



Recall: Little's Law

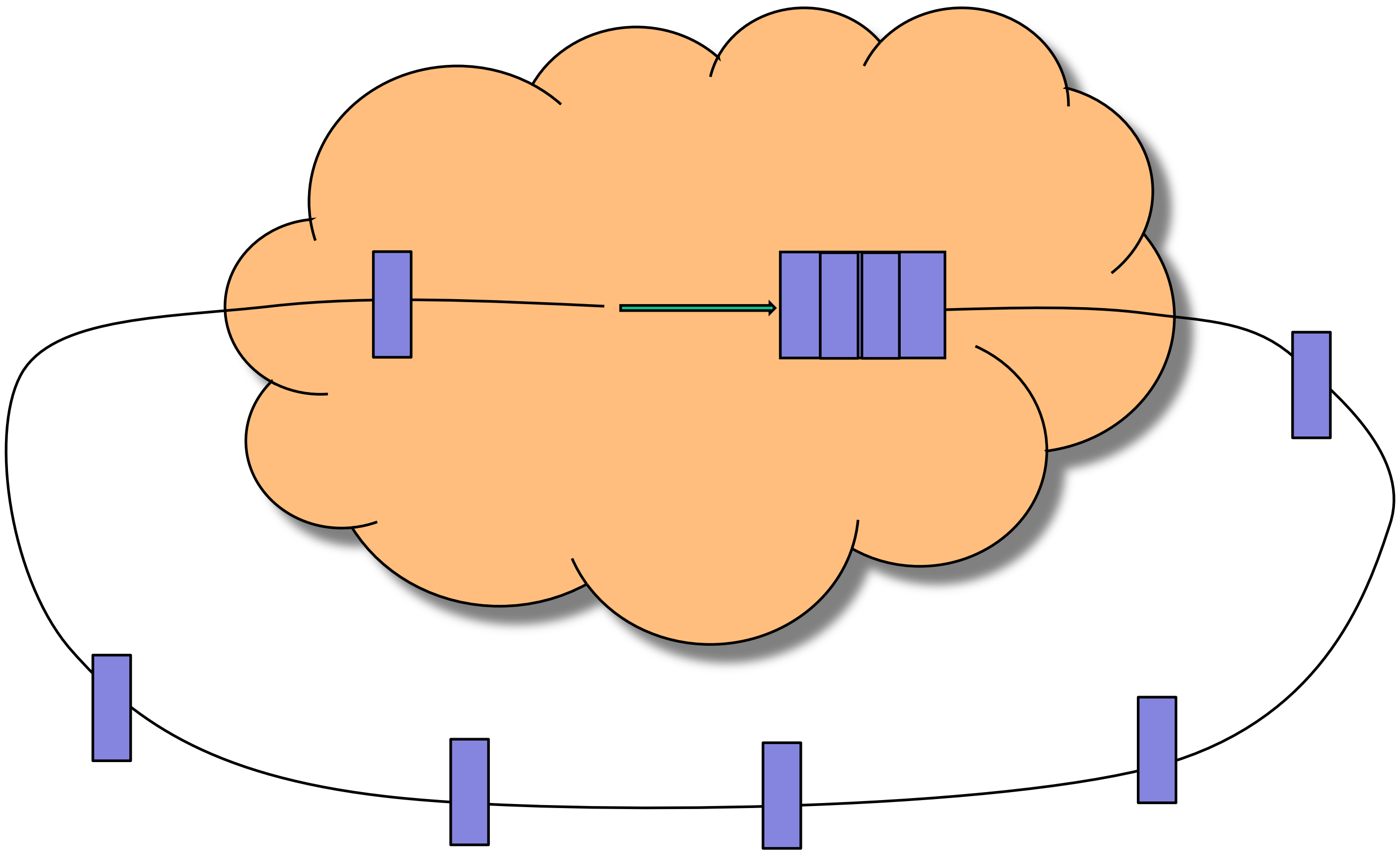
- For any system with no or (low) loss.
- Assume
  - mean arrival rate X, mean service time T, and mean number of requests in the system W



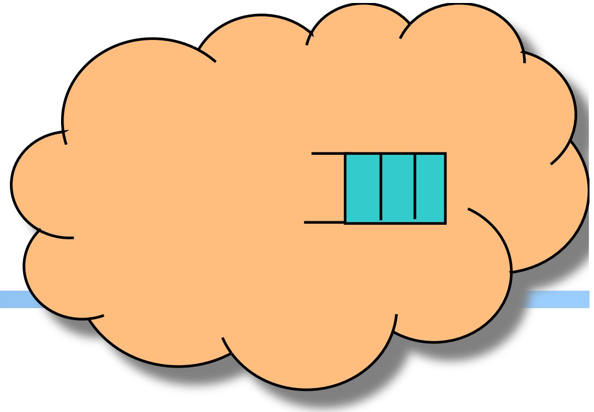
- Then relationship between W, X, and T:

$$W = XT$$

Estimating Number of Packets in the Queue



TCP/Vegas CA algorithm



$$T = T_{prop} + T_{queueing}$$

Applying Little's Law:

$$x_{vegas}\, T = x_{vegas}\, T_{prop} + x_{vegas}\, T_{queueing},$$

where $x_{vegas} = W / T$ is the sending rate

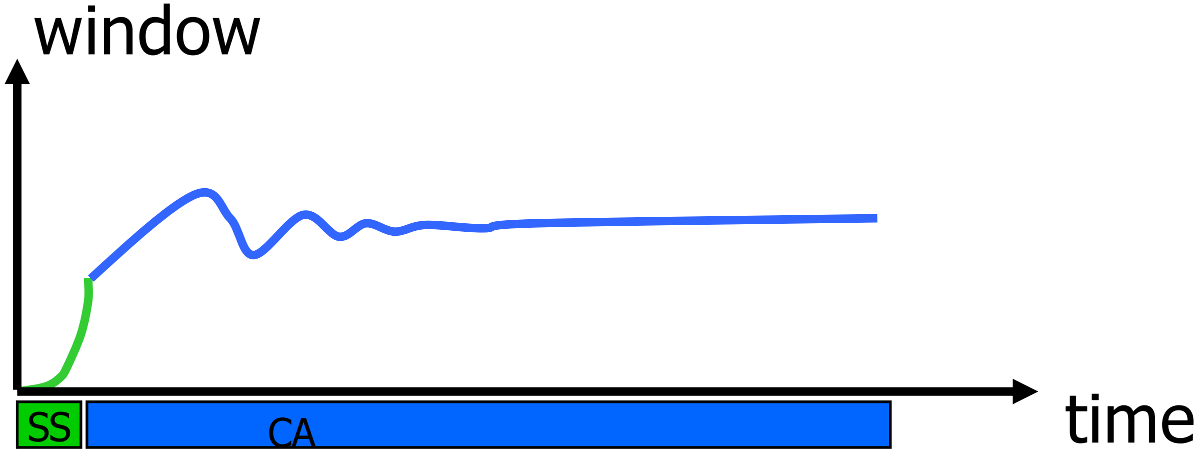
Then number of packets in the queue is

$$x_{vegas}\, T_{queueing} = x_{vegas}\, T - x_{vegas}\, T_{prop}$$

$$= W - W/T\, T_{prop}$$

TCP/Vegas CA algorithm

maintain a *constant* number of packets in the bottleneck buffer



for every RTT

{ if $W - W/RTT\ RTT_{min} < \alpha$ then W ++

if $W - W/RTT\ RTT_{min} > \alpha$ then W --

}

queue size

for every loss

W := W/2

Discussions

- If two flows, one TCP Vegas and one TCP reno run together, how may bandwidth partitioned among them?

- Issues that limit Vegas deployment?

Outline

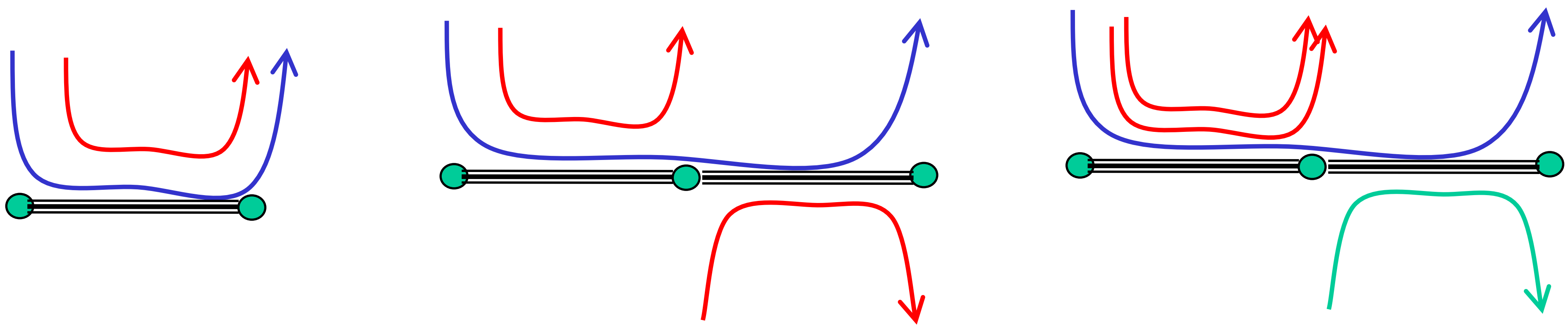
- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
  - TCP/Vegas
  - network wide resource allocation
    - general framework

Motivation

- So far our discussion is implicitly on a network with a single bottleneck link; this simplifies design and analysis:
  - efficiency/optimality (high utilization)
    - fully utilize the bandwidth of the link
  - fairness (resource sharing)
    - each flow receives an ***equal*** share of the link's bandwidth

Network Resource Allocation

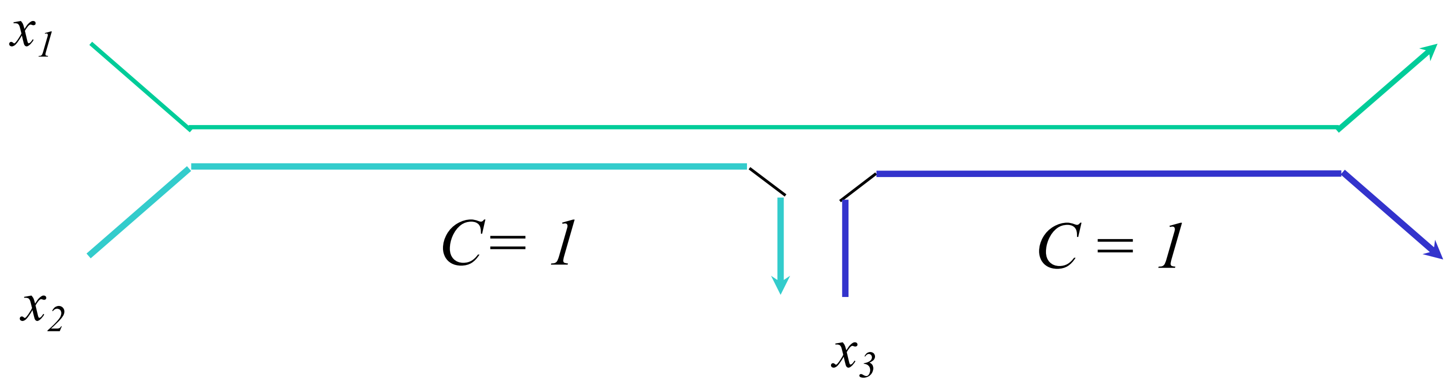
- It is important to understand and design protocols for a general network topology
  - how will TCP allocate resource in a general topology?
  - how should resource be allocated in a general topology?



Example: TCP/Reno Rates

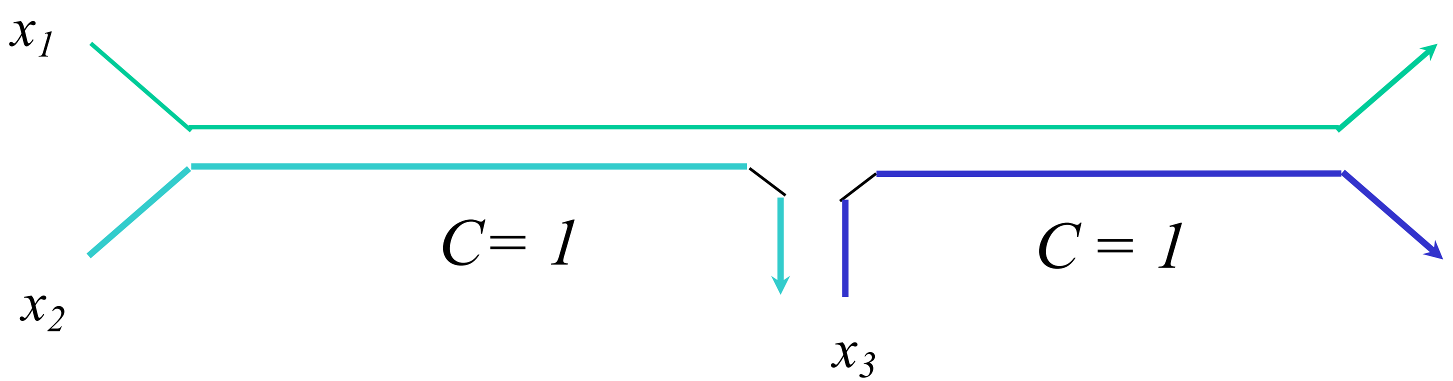
- Rates: $x_1 = \frac{1}{1+2\sqrt{2}} = 0.26$

  $x_2 = x_3 = 0.74$



Example: TCP/Vegas Rates

- Rates : $x_1 = 1/3$
  $x_2 = x_3 = 2/3$



Example: Max-min Fairness

- Max-min fairness: maximizes the throughput of the flow receiving the minimum (of resources)
    - Justification: John Rawls, *A Theory of Justice* (1971)
        - http://en.wikipedia.org/wiki/John_Rawls
    - This is a resource allocation scheme used in ATM and some other network resource allocation proposals

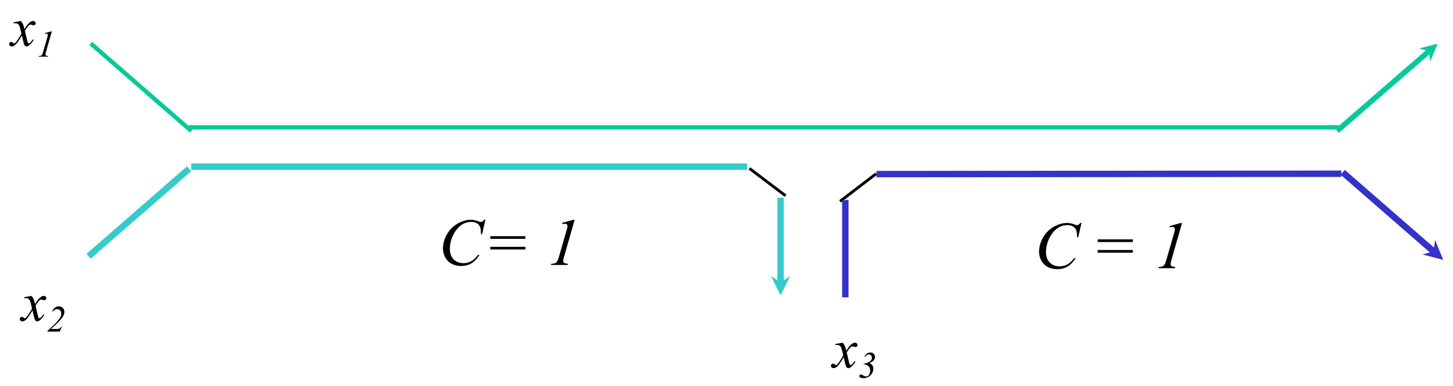
Example: Max-Min

$$\max_{x_f \geq 0} \quad \min\{x_f\}$$

$$\text{subject to} \quad x_1 + x_2 \leq 1$$

$$x_1 + x_3 \leq 1$$

- Rates: $x_1 = x_2 = x_3 = 1/2$



Framework: Network Resource Allocation Using Utility Functions

- A set of flows F
- Let x_f be the rate of flow f, and the utility to flow f is $U_f(x_f)$.
- Maximize aggregate utility, subject to capacity constraints

$$
\begin{array}{ll}
\text{max} & \sum_{f \in F} U_f\left(x_f\right) \\
\text{subject to} & \sum_{f : f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\
\text{over} & x \geq 0
\end{array}
$$

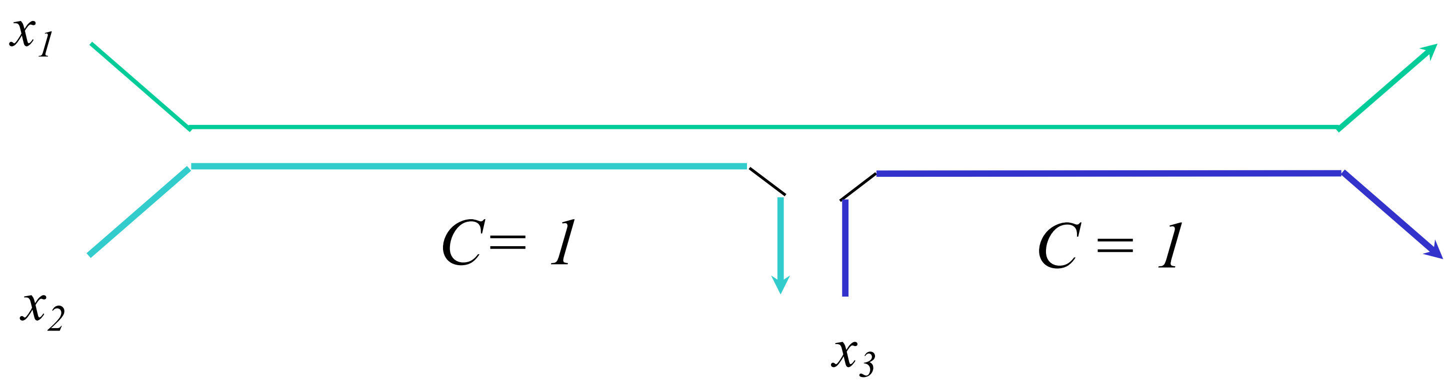
Example: Maximize Throughput

$$\max_{x_f \geq 0} \quad \sum_f x_f \qquad U_f(x_f) = xf$$

$$\text{subject to} \quad x_1 + x_2 \leq 1$$

$$x_1 + x_3 \leq 1$$

- Optimal: $x_1 = 0$
  $x_2 = x_3 = 1$



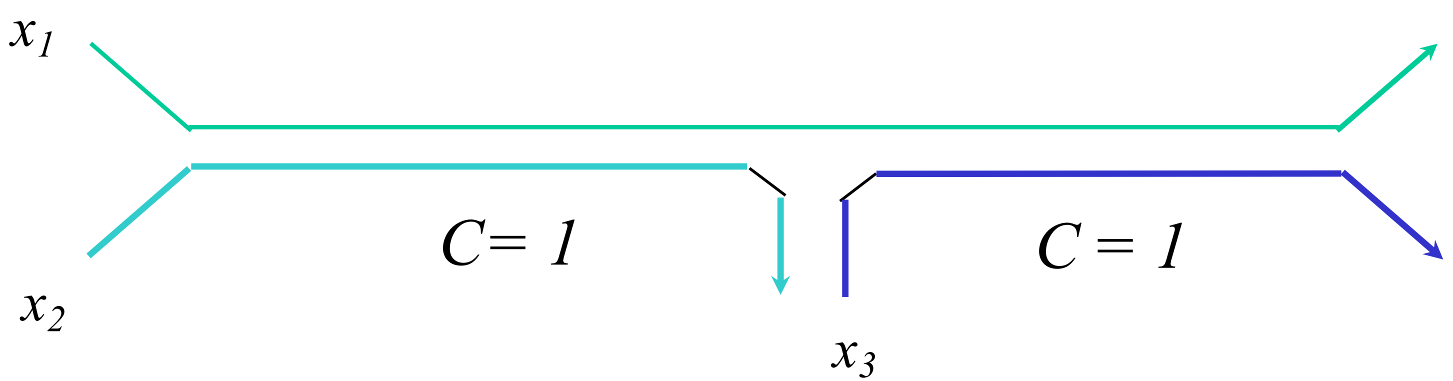
Example: Proportional Fairness

$$\max_{x_f \geq 0} \quad \sum_f \log x_f \qquad U_f\left(x_f\right) = \log\left(x_f\right)$$

$$\text{subject to} \quad x_1 + x_2 \leq 1$$

$$x_1 + x_3 \leq 1$$

- Optimal: $x_1 = 1/3$
  $x_2 = x_3 = 2/3$



Example 3: a "Funny" Utility Function

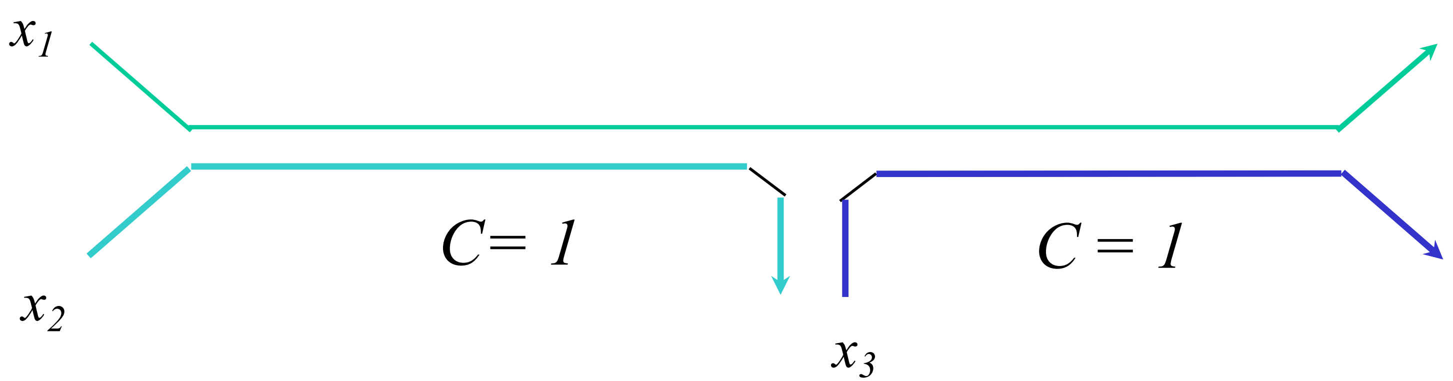
$$\max_{x_f \geq 0} \quad -\frac{1}{4x_1} - \frac{1}{x_2} - \frac{1}{x_3}$$

$$\text{subject to} \quad x_1 + x_2 \leq 1$$

$$x_1 + x_3 \leq 1$$

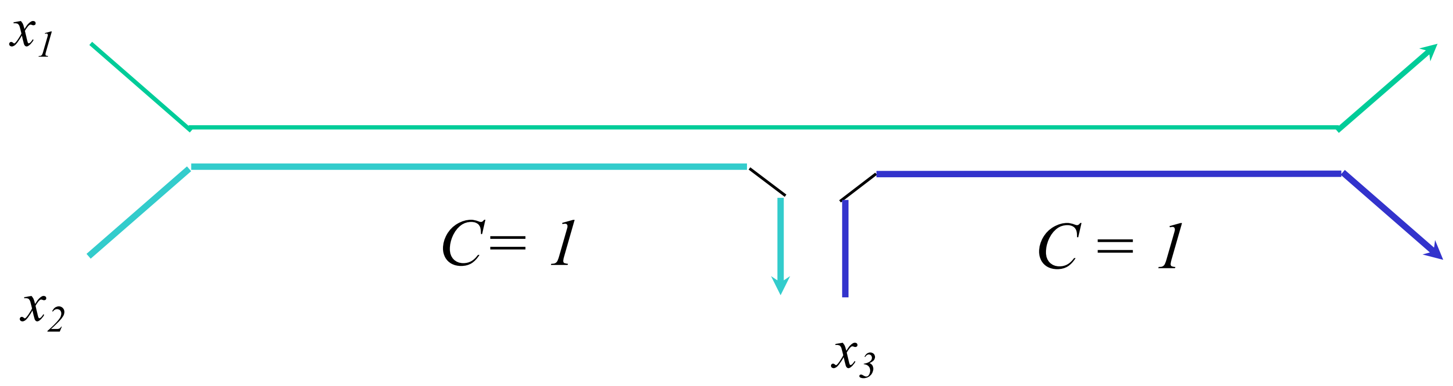
$$U_f(x_f) = -\frac{1}{RTT^2 x_f}$$

- Optimal: $x_1 = \frac{1}{1+2\sqrt{2}} = 0.26$
  $x_2 = x_3 = 0.74$



Summary: Allocations

Objective	Allocation (x1, x2, x3)		
TCP/Reno	0.26	0.74	0.74
TCP/Vegas	1/3	2/3	2/3
Max Throughput	0	1	1
Max-min	½	½	½
Max sum log(x)	1/3	2/3	2/3
Max sum of $-1/(RTT^2\ x)$	0.26	0.74	0.74



Questions

$$\begin{array}{ll} \max & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \le c_l \text{ for any link } l \\ \text{over} & x \ge 0 \end{array}$$

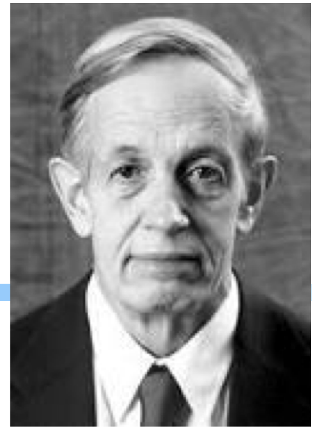
- **Forward engineering:** systematically
  - design objective function
  - design distributed alg to achieve objective
- **Science/reverse engineering:** what do TCP/Reno, TCP/Vegas achieve?

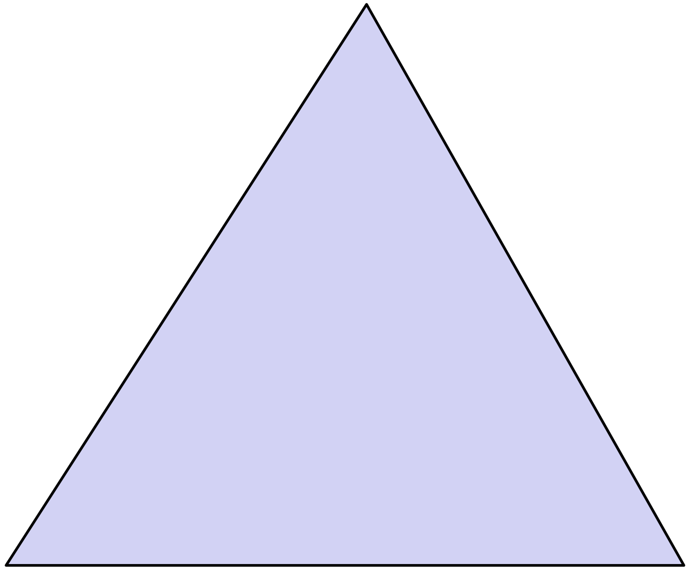
Objective	Allocation (x1, x2, x3)		
TCP/Reno	0.26	0.74	0.74
TCP/Vegas	1/3	2/3	2/3
Max throughput	0	1	1
Max-min	½	½	½
Max sum log(x)	1/3	2/3	2/3
Max sum of $-1/(RTT^2 x)$	0.26	0.74	0.74

Outline

- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
  - TCP/Vegas
  - network wide resource allocation
    - general framework
    - objective function: an example of an axiom derivation of network-wide objective function

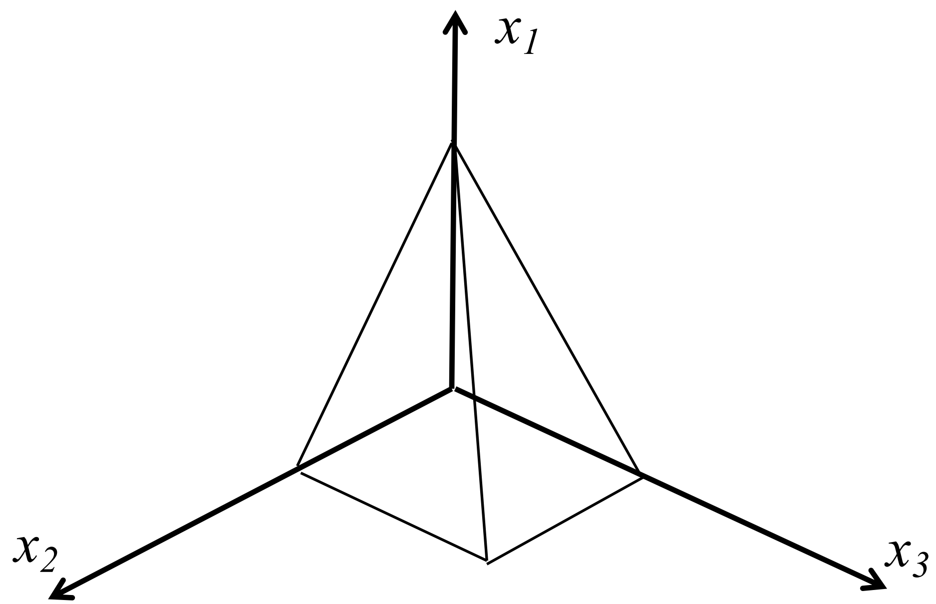
Network Bandwidth Allocation Using Nash Bargain Solution (NBS)

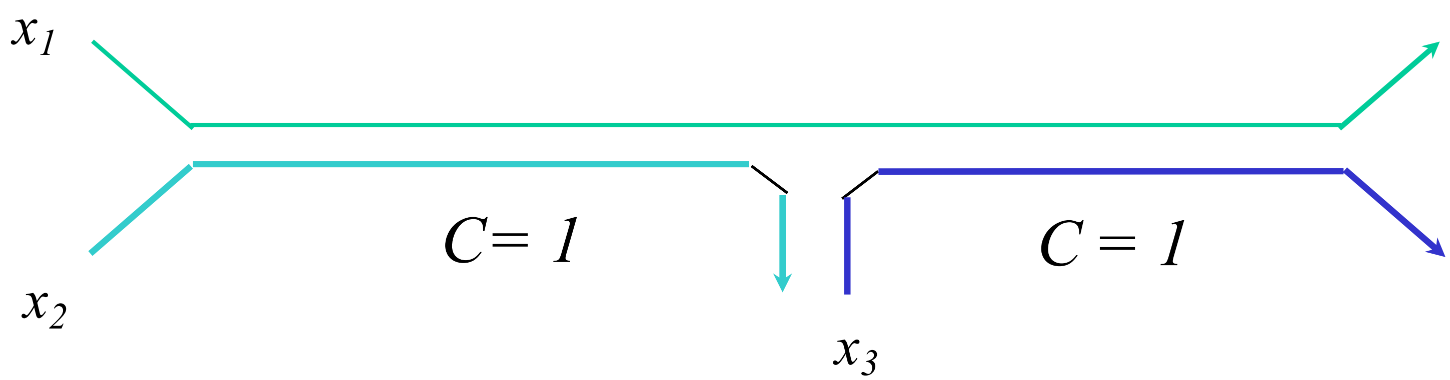




- High level picture
  - given the feasible set of bandwidth allocation, we want to pick an allocation point that is efficient and fair
- The determination of the allocation point should be based on "first principles" (axioms)

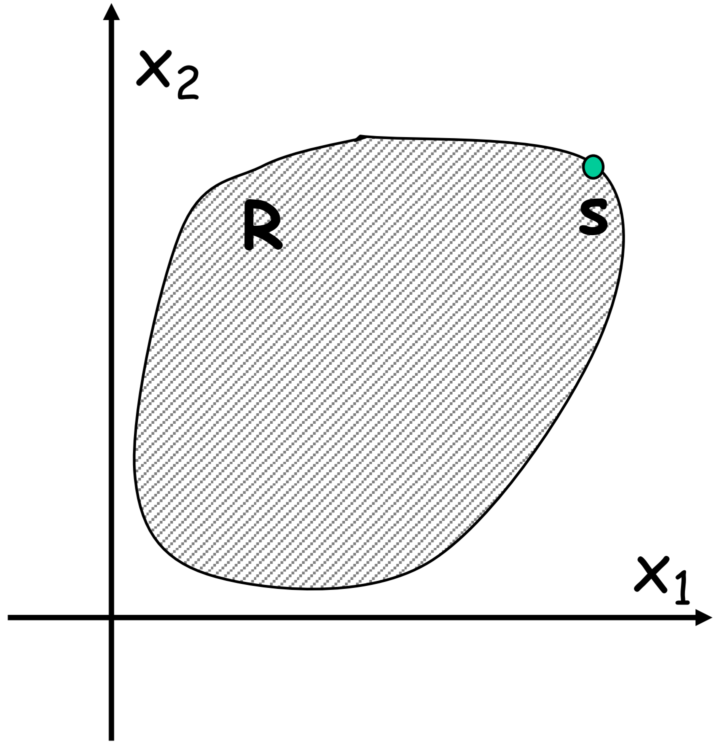
Network Bandwidth Allocation: Feasible Region





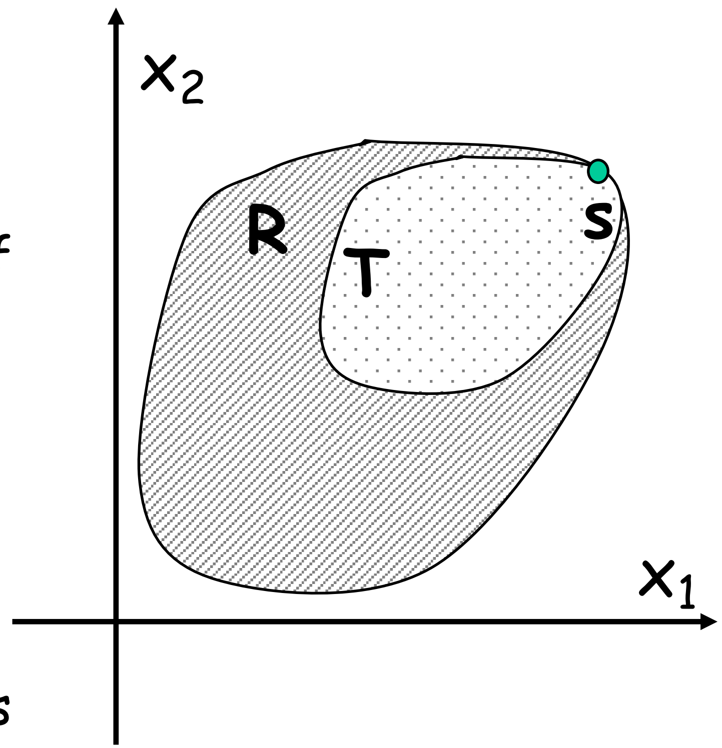
Nash Bargain Solution (NBS)

- Assume a finite, convex feasible set in the first quadrant
- Axioms



Nash Bargain Solution (NBS)

- Assume a finite, convex feasible set in the first quadrant
- Axioms
  - Pareto optimality
    - impossibility of increasing the rate of one user without decreasing the rate of another
  - symmetry
    - a symmetric feasible set yields a symmetric outcome
  - invariance of linear transformation
    - the allocation must be invariant to linear transformations of users' rates
  - independence of irrelevant alternatives
    - assume s is an allocation when feasible set is R, $s \in T \subset R$, then s is also an allocation when the feasible set is T



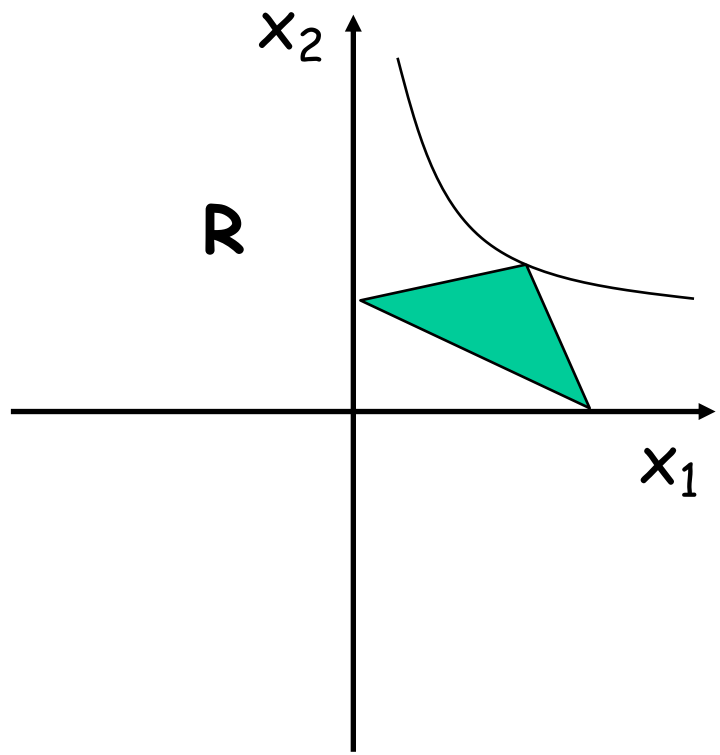
Nash Bargain Solution (NBS)

- Surprising result by John Nash (1951)
  - the rate allocation point is the feasible point which maximizes

$$x_1 x_2 \cdots x_F$$

- This is equivalent to maximize

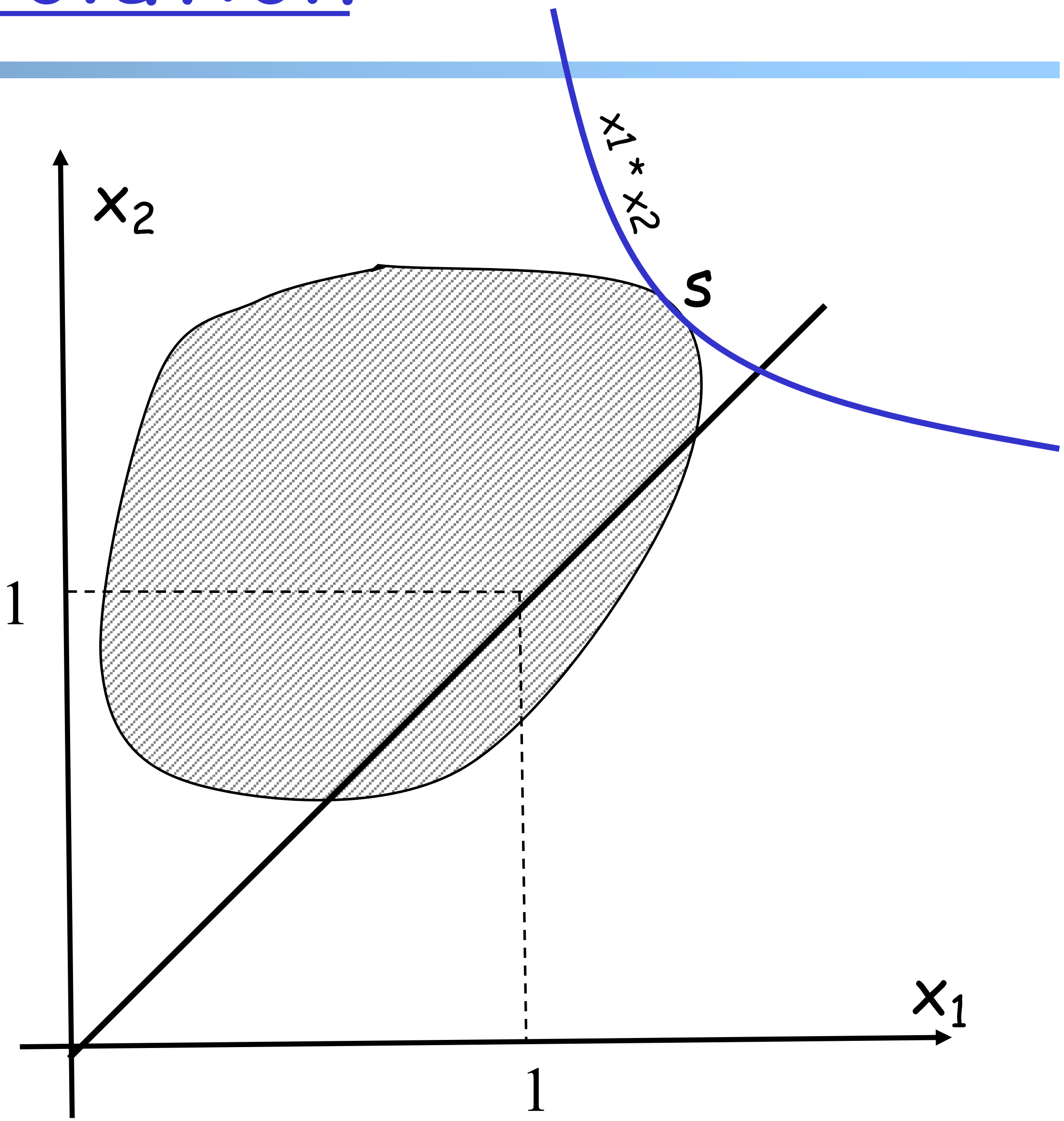
$$\sum_f \log(x_f)$$



- In other words, assume each flow f has utility function $\log(x_f)$
- I will give a proof for F = 2
  - think about F > 2

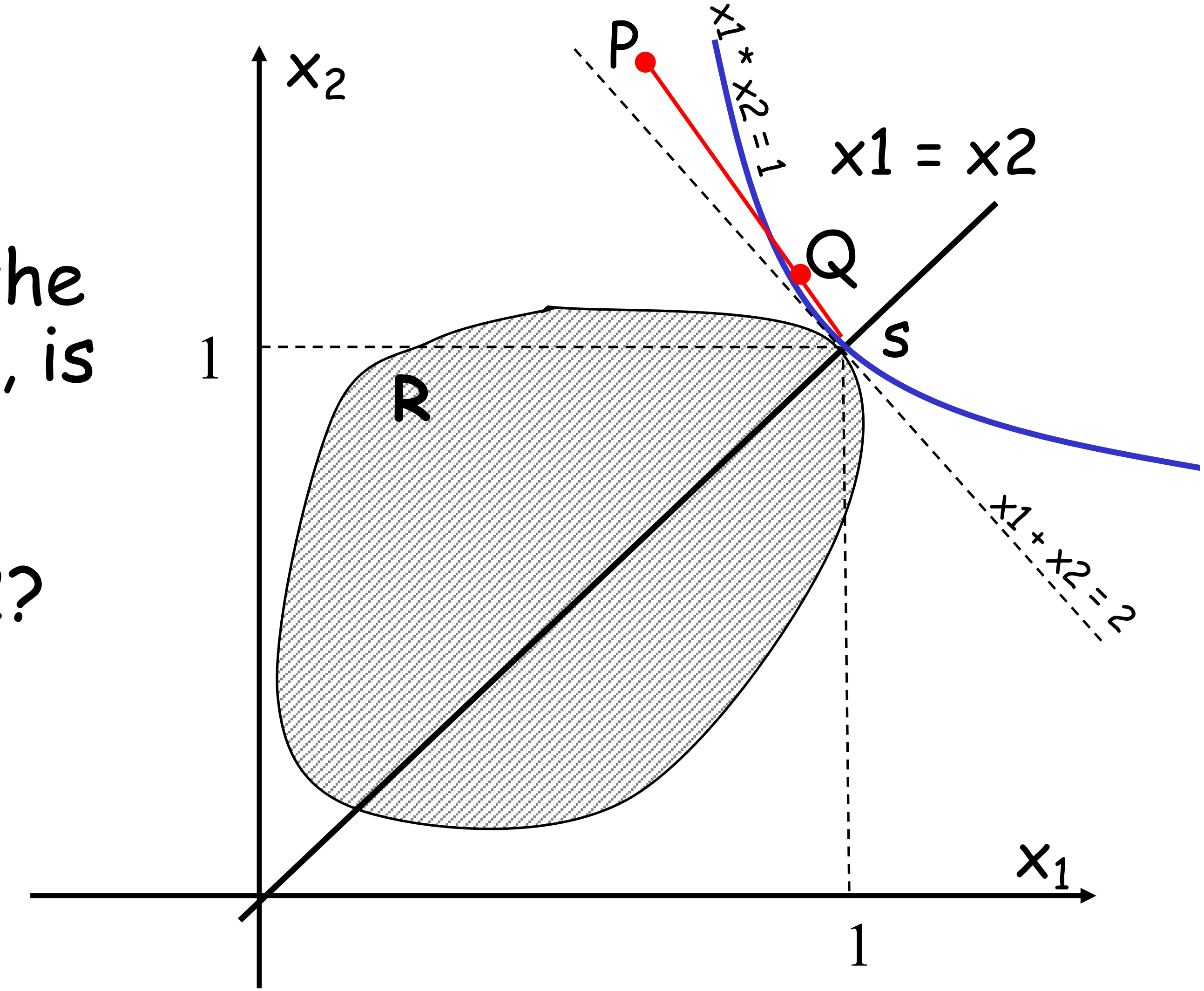
Nash Bargain Solution

- Assume s is the feasible point which maximizes x1 * x2
- Scale the feasible set so that s is at (1, 1)
  - how?



Nash Bargain Solution

Question: after the transformation, is there any feasible point with x1 + x2 > 2?

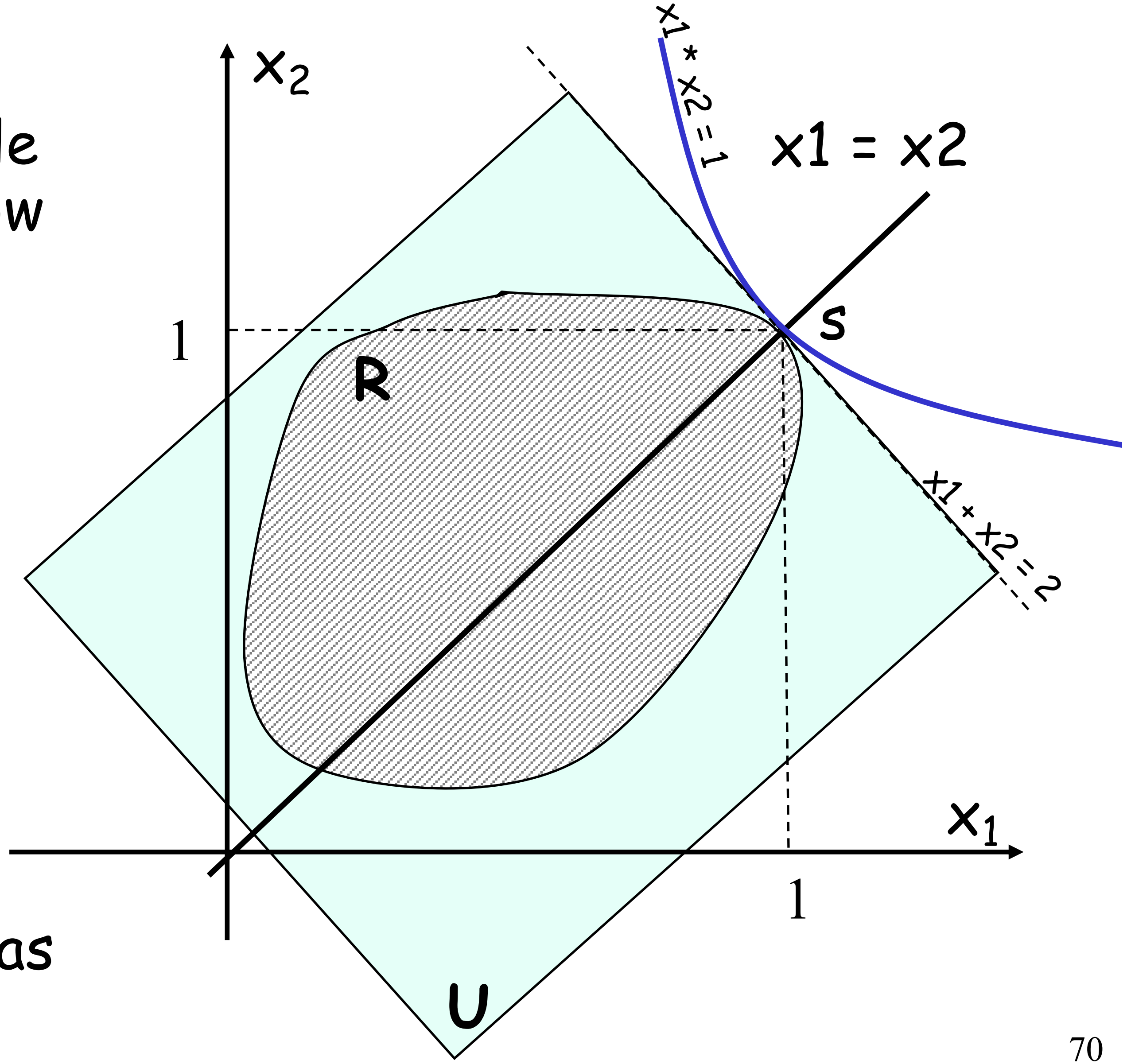


Nash Bargain Solution

- Consider the symmetric rectangle U containing the now feasible set

-> According to symmetry and Pareto, s is the allocation when feasible set is U

- According to independence of irrelevant alternatives, the allocation of R is s as well.



NBS ⇔ Proportional Fairness

- Allocation is proportionally fair if for any other allocation, aggregate of proportional changes is non-positive, e.g. if x_f is a proportional-fair allocation, and y_f is any other feasible allocation, then require

$$\sum_f \frac{y_f - x_f}{x_f} \le 0$$

Questions to Think

- Vary the axioms and see if you can derive any objective functions

Outline

- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
  - TCP/Vegas
  - network wide resource allocation
    - general framework
    - objective function: an example axiom derivation of network-wide objective function
    - algorithm: general distributed algorithm framework
    - application: TCP/Reno TCP/Vegas revisited

Recall: Resource Allocation Framework

- The Resource-Allocation Problem:

$$
\begin{array}{ll}
\max & \sum_{f \in F} U_f\left(x_f\right) \\
\text{subject to} & \sum_{f : f \text{ uses link } l} x_f \le c_l \text{ for any link } l \\
\text{over} & x \ge 0
\end{array}
$$

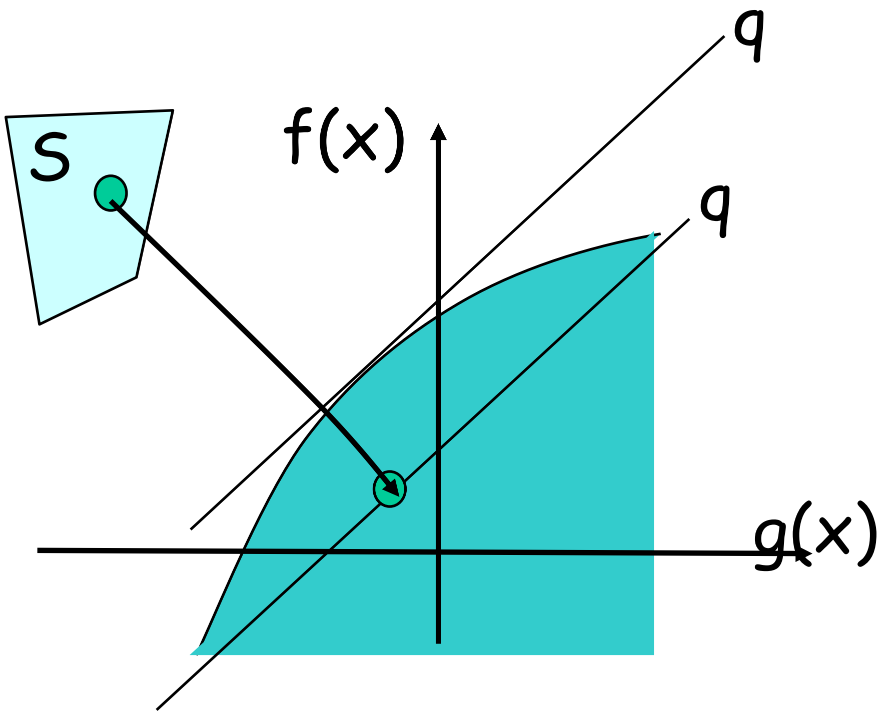
- Goal: Design a distributed alg to solve the problem.
- Discussion:
  - What are typical approaches to solve optimization, e.g.,?
    max U(x)
  - Why is the Resource-Allocation problem hard to solve by a distributed algorithm?

A Two-Slide Summary of Constrained Convex Optimization Theory

$$\begin{array}{ll} \max & \sum_{f \in F} U_f\left(x_f\right) \\ \text{subject to} & Ax \leq C \\ \text{over} & x \geq 0 \end{array}$$

$$\begin{array}{ll} \max & f(x) \\ \text{subject to} & g(x) \leq 0 \\ \text{over} & x \in S \end{array}$$

f(x) concave
g(x) linear
S is a convex set



-Map each x in S, to [g(x), f(x)]

-Top contour of map is concave

-Easy to read solution from contour

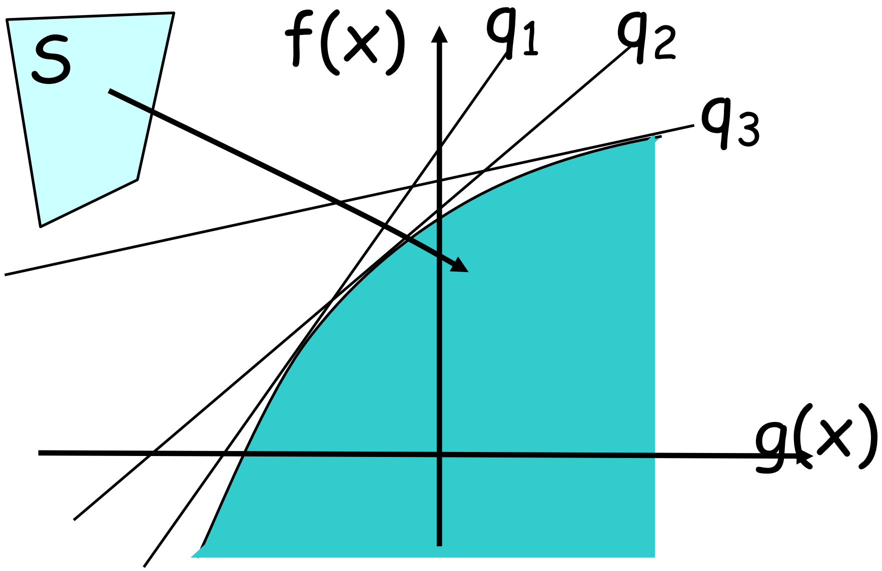
-For each slope q (>=0), computes f(x) – q g(x) of all mapped [f(x), g(x)]

$$D(q) = \max_{x \in S}\big(f(x) - qg(x)\big)$$

A Two-Slide Summary of Constrained Convex Optimization Theory

max	$f(x)$
subject to	$g(x) \leq 0$
over	$x \in S$

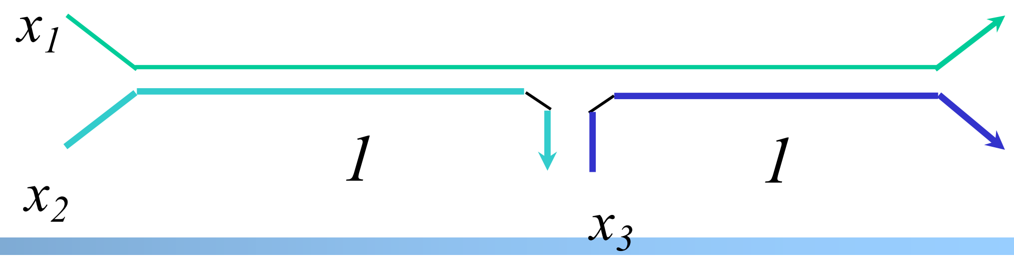
f(x) concave
g(x) linear
S is a convex set



$$D(q) = \max_{x \in S}\left(f(x) - qg(x)\right)$$

-D(q) is called the dual;
q (>= 0) are called prices in economics
-D(q) provides an upper bound on obj.
- According to optimization theory:
when D(q) achieves minimum over all q (>= 0), then the optimization objective is achieved.

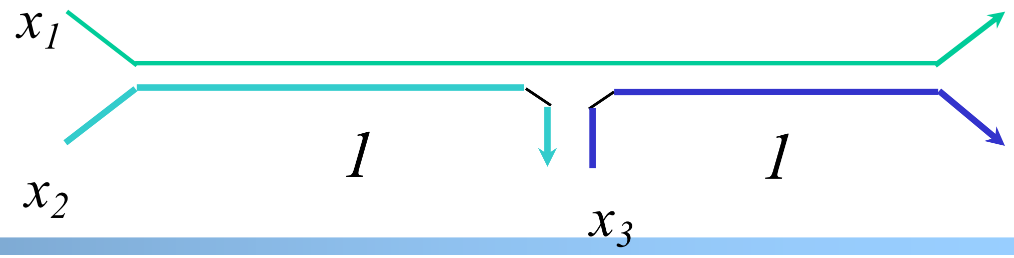
Dual of the Primal



max	$\sum_{f \in F} U_f\left(x_f\right)$
subject to	$\sum_{f : f \text{ uses link } l} x_f \le c_l$ for any link l
over	$x \ge 0$

$$D(q) = \max_{x_f \ge 0} \left(\sum_f U_f(x_f) - \sum_l q_l \Big(\sum_{f :\text{uses } l} x_f - c_l \Big) \right)$$

Dual of the Primal



$$D(q) = \max_{x_f \geq 0}\left(\sum_f U_f(x_f) - \sum_l q_l \Big(\sum_{f:\text{uses } l} x_f - c_l \Big) \right)$$

$$= \max_{x_f \geq 0} \sum_f \left(U_f(x_f) - x_f \sum_{l:\text{f uses } l} q_l \right) + \sum_l q_l c_l$$

$$= \sum_f \max_{x_f \geq 0} \left(U_f(x_f) - x_f \sum_{l:\text{f uses } l} q_l \right) + \sum_l q_l c_l$$

Distributed Optimization: User Problem

- Given p_f (=sum of dual var q_l along the path) flow f chooses rate x_f to maximize:

$$\max_{x_f} \quad U_f(x_f) - x_f p_f$$

$$\text{over} \quad x_f \geq 0$$

- Using the price signals, the optimization problem of each user is independent of each other!

Distributed Optimization: User Problem

$$\boxed{\begin{array}{ll}\max\limits_{x_f} & U_f\left(x_f\right) - x_f p_f \\ \text{over} & x_f \geq 0\end{array}}$$

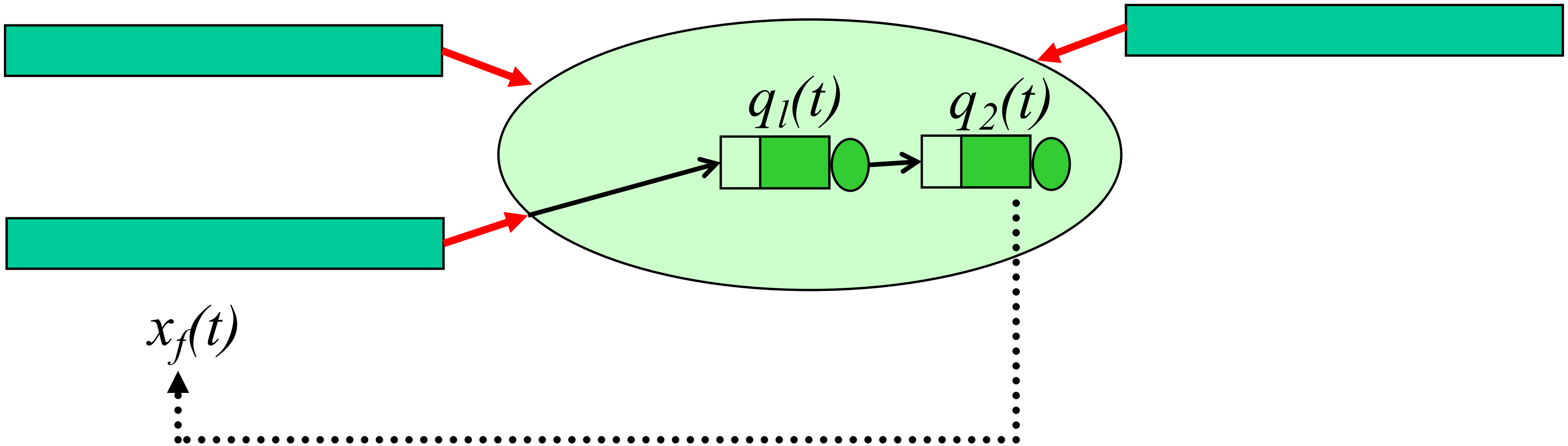
How should flow f adjust x_f locally?

$$\boxed{\Delta x_f \propto U'_f\left(x_f\right) - p_f}$$

At equilibrium (i.e., at optimal), x_f satisfies:

$$U'_f\left(x_f\right) - p_f = 0$$

Interpreting Congestion Measure



$$\boxed{p_f = \sum_{\text{f uses l}} q_l}$$

$$\boxed{\Delta x_f \propto U'_f\left(x_f\right) - p_f}$$

Distributed Optimization: Network Problem

$$D(q) = \sum_f \max_{x_f \geq 0} \left(U_f(x_f) - x_f \sum_{l:\text{f uses } l} q_l \right) + \sum_l q_l c_l$$

The network (i.e., link l) adjusts the link signals q_l (assume after all flows have picked their optimal rates given congestion signal)

$$\min_{q \geq 0} \widetilde{D}(q) = \sum_l q_l (c_l - \sum_{f:\text{f uses } l} x_f)$$

Distributed Optimization: Network Problem

$$\min_{q\geq 0} D(q) = \sum_{l} q_l (c_l - \sum_{f:\text{ f uses } l} x_f)$$

how should link l adjust q_l locally?

$$\Delta q_l \propto -\frac{\partial D(q)}{q_l}$$

$$\frac{\partial}{\partial q_l} D(q) = c_l - \sum_{f:\text{uses } l} x_f$$

$$\Delta q_l \propto \sum_{f:\text{uses } l} x_f - c_l$$

System Architecture

- SYSTEM(U):

$$\begin{array}{ll} \text{max} & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \le c_l \text{ for any link } l \\ \text{over} & x \ge 0 \end{array}$$

- USER$_f$:

$$\Delta x_f \propto U'_f(x_f) - p_f$$

$$\begin{array}{ll} \max_{x_f} & U_f(x_f) - x_f p_f \\ \text{over} & x_f \ge 0 \end{array}$$

- NETWORK:

$$\Delta q_l \propto -\frac{\partial D(q)}{q_l}$$

$$\min_{q \ge 0} \widetilde{D}(q) = \sum_l q_l (c_l - \sum_{f: \text{f uses } l} x_f)$$

Decomposition Theorem

- There exist vectors **p** , **w** and **x** such that
  1. $w_f = p_f x_f$ for $f \in F$
  2. w_f solves $USER_f(U_f; p_f)$
  3. x solves NETWORK(w)

- The vector x then also solves SYSTEM(U).

Outline

- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
  - TCP/Vegas
  - network wide resource allocation
    - general framework
    - objective function: an example axiom derivation of network-wide objective function
    - algorithm: a general distributed algorithm framework
    - application: TCP/Reno and TCP/Vegas revisited

TCP/Reno Dynamics

$$\Delta x_f \propto U'_f(x_f) - p_f$$

$$\Delta W_{pkt} = (1-p)\frac{1}{W} - p\frac{W}{2}$$

$$\Delta W_{RTT} = \Delta W_{pkt}\, W = (1-p) - p\frac{W^2}{2} \cong 1 - p\frac{W^2}{2}$$

$$\Delta x = \frac{\Delta W_{RTT}}{RTT} = \frac{1}{RTT} - \frac{RTT}{2}\, p\, x^2$$

$$= \frac{RTT}{2}\, x^2 \left(\frac{2}{x^2 RTT^2} - p\right)$$

TCP/Reno Dynamics

$$\boxed{\Delta x_f \propto U'_f(x_f) - p_f}$$

$$\Delta x = \frac{RTT}{2} x^2 \boxed{\left(\frac{2}{x^2 RTT^2} - p\right)}$$

$$U'_f(x_f) - p_f$$

$$\Rightarrow U'_f(x_f) = \left(\frac{\sqrt{2}}{x_f RTT}\right)^2 \quad \Rightarrow U_f(x_f) = -\frac{2}{RTT^2 x_f}$$

TCP/Vegas Dynamics

$$\Delta x_f \propto U'_f\left(x_f\right) - p_f$$

$$\Delta \mathrm{w}_{\mathrm{RTT}} \approx -(w - xRTT_{\min} - \alpha)$$

$$\Delta x \;=\; \frac{\Delta WRT_T}{RTT} = -\left(\frac{w}{RTT} - \frac{x}{RTT}RTT_{min} - \frac{\alpha}{RTT}\right)$$

$$= -\frac{w}{RTT} + \frac{x}{RTT}RTT_{min} + \frac{\alpha}{RTT}$$

$$= -x + \frac{x}{RTT}RTT_{min} + \frac{\alpha}{RTT}$$

$$= \frac{x}{RTT}\left(-RTT + \mathrm{RTT}_{\min} + \frac{\alpha}{x}\right)$$

$$= \frac{x}{RTT}\left(\frac{\alpha}{x} - (RTT - \mathrm{RTTmin})\right)$$

$$\begin{aligned}\Delta W &\simeq \alpha - \left(W - \frac{RTT_{min}}{RTT}W\right)\\ &\simeq \alpha - \left(W - \frac{RTT_{min}}{RTT}xRTT\right)\\ &\simeq -(W - xRTT_{min} - \alpha)\end{aligned}$$

TCP/Vegas Dynamics

$$\boxed{\Delta x_f \propto U'_f\left(x_f\right) - p_f}$$

$$\Delta x = \frac{x}{RTT}\left(\boxed{\frac{\alpha}{x} - (RTT - \mathrm{RTTmin}})\right)$$

$$U'_f\left(x_f\right) - p_f$$

$$\Rightarrow U'_f(x_f) = \tfrac{\alpha}{x} \qquad \Rightarrow U_f(x_f) = \alpha \log(x_f)$$

Summary: TCP/Vegas and TCP/Reno

- Pricing signal is queueing delay $T_{queueing}$
- Pricing signal is loss rate p

$$x_f = \frac{\alpha}{T_{queueing}}$$

$$U'_f(x_f) = T_{queueing}$$

$$\Rightarrow U'_f(x_f) = \frac{\alpha}{x_f}$$

$$\Rightarrow U_f(x_f) = \alpha \log(x_f)$$

$$x_f = \frac{\alpha}{RTT\sqrt{p}}$$

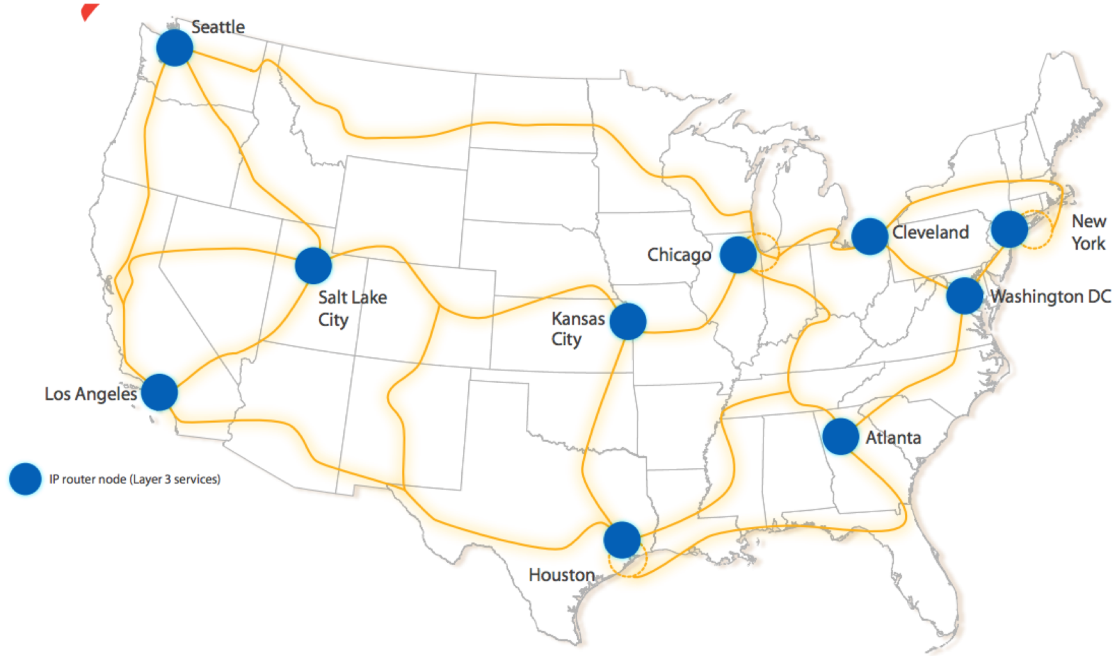
$$U'_f(x_f) = p$$

$$\Rightarrow U'_f(x_f) = \left(\frac{\alpha}{x_f RTT}\right)^2$$

$$\Rightarrow U_f(x_f) = -\frac{\alpha'}{RTT^2 x_f}$$

Discussion

- Assume that you are given a set of flows deployed at a given network topology.
- What is a simple way to predict TCP rate allocation?



Summary: Resource Allocation Frameworks

- Forward (design) engineering:
  - how to determine objective functions
  - given objective, how to design effective alg

$$
\begin{array}{ll}
\max & \sum_{f \in F} U_f\left(x_f\right) \\
\text{subject to} & Ax \leq C \\
\text{over} & x \geq 0
\end{array}
$$

- Reverse (understand) engineering:
  - understand current protocols (what are the objectives of TCP/Reno, TCP/Vegas?)
- Additional pointers:
  - http://www.statslab.cam.ac.uk/~frank/pf/