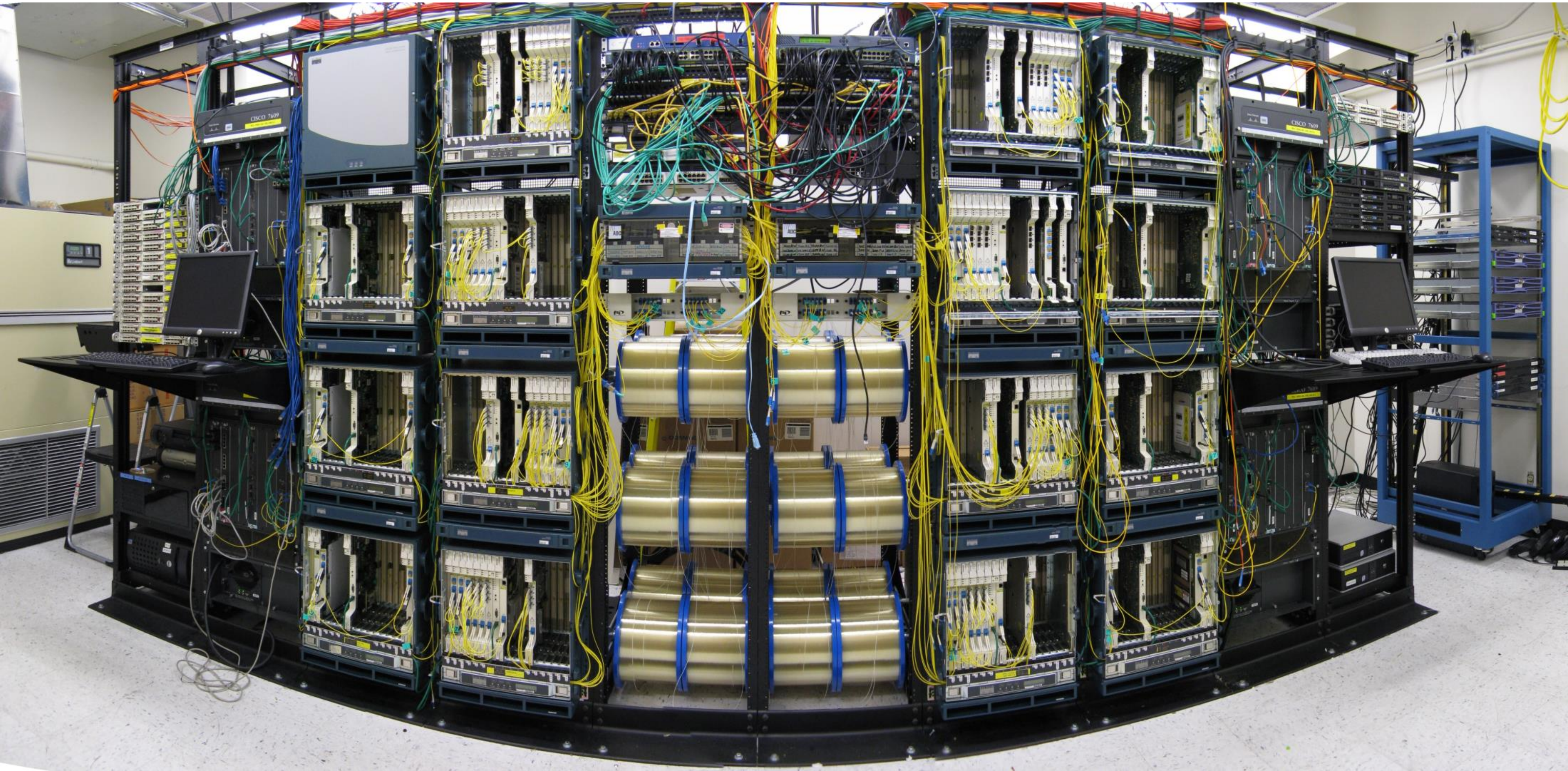
Congestion Control & Optimization

Steven Low
netlab.**CALTECH**.edu
Cambridge 2011



Acknowledgments

Caltech: L. Chen, J. Doyle, C. Jin, G. Lee, H. Newman, A. Tang, D. Wei, B. Wydrowski, Netlab Gen1

Uruguay: F. Paganini

Swinburne: L. Andrew

Princeton: M. Chiang

Goal of tutorial

Top-down summary of congestion control on Internet

Introduction to mathematical models of congestion control

Illustration of theory-guided CC algorithm design

Theory-guided design

Tight integration of theory, design, experiment

- Analysis done *at* design time, not after

Theory does not replace intuitions or heuristics

- Refines, validates/invalidates them

Theory provides structure and clarity

- Guides design
- Suggests ideas and experiments
- Explores boundaries that are hard to experiment

Theory-guided design

Integration of theory, design, experiment can be very powerful

- Each needs the other
- Combination much more than sum

Tremendous progress in the last decade

- Not as impossible as most feared
- Very difficult; but worth the effort
- Most critical: mindset

How to push theory-guided design approach further ?

Agenda

9:00	Congestion control protocols
10:00	break
10:15	Mathematical models
11:15	break
11:30	Advanced topics
12:30	lunch

Audience background

Know TCP/IP protocols?

Know congestion control?

Experiment with ns2? Linux kernel?

Know optimization theory? Control theory?

Know network utility maximization?

CONGESTION CONTROL PROTOCOLS

Congestion control protocols

Why congestion control?

Where is CC implemented?

Window control mechanism

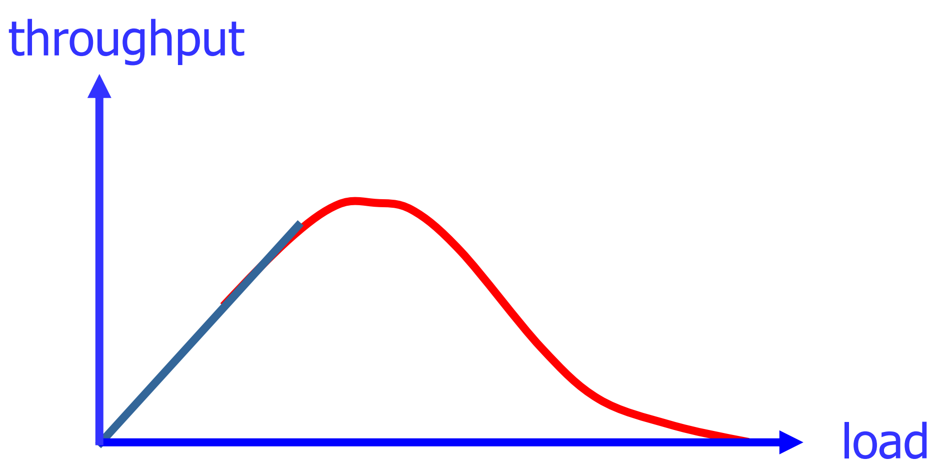
CC protocols and basic structure

Active queue management (AQM)

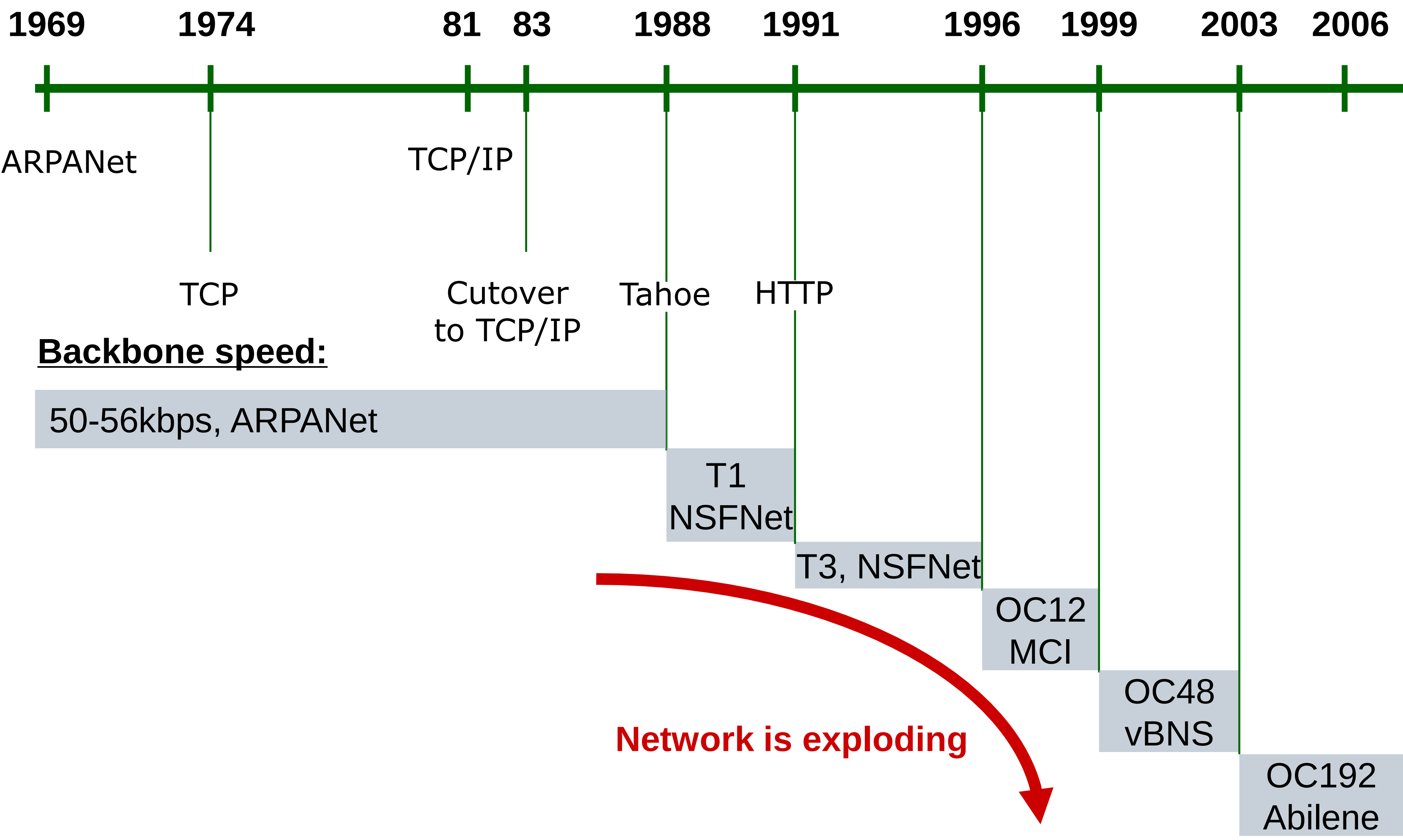
Congestion collapse

October 1986, the first congestion collapse on the Internet was detected

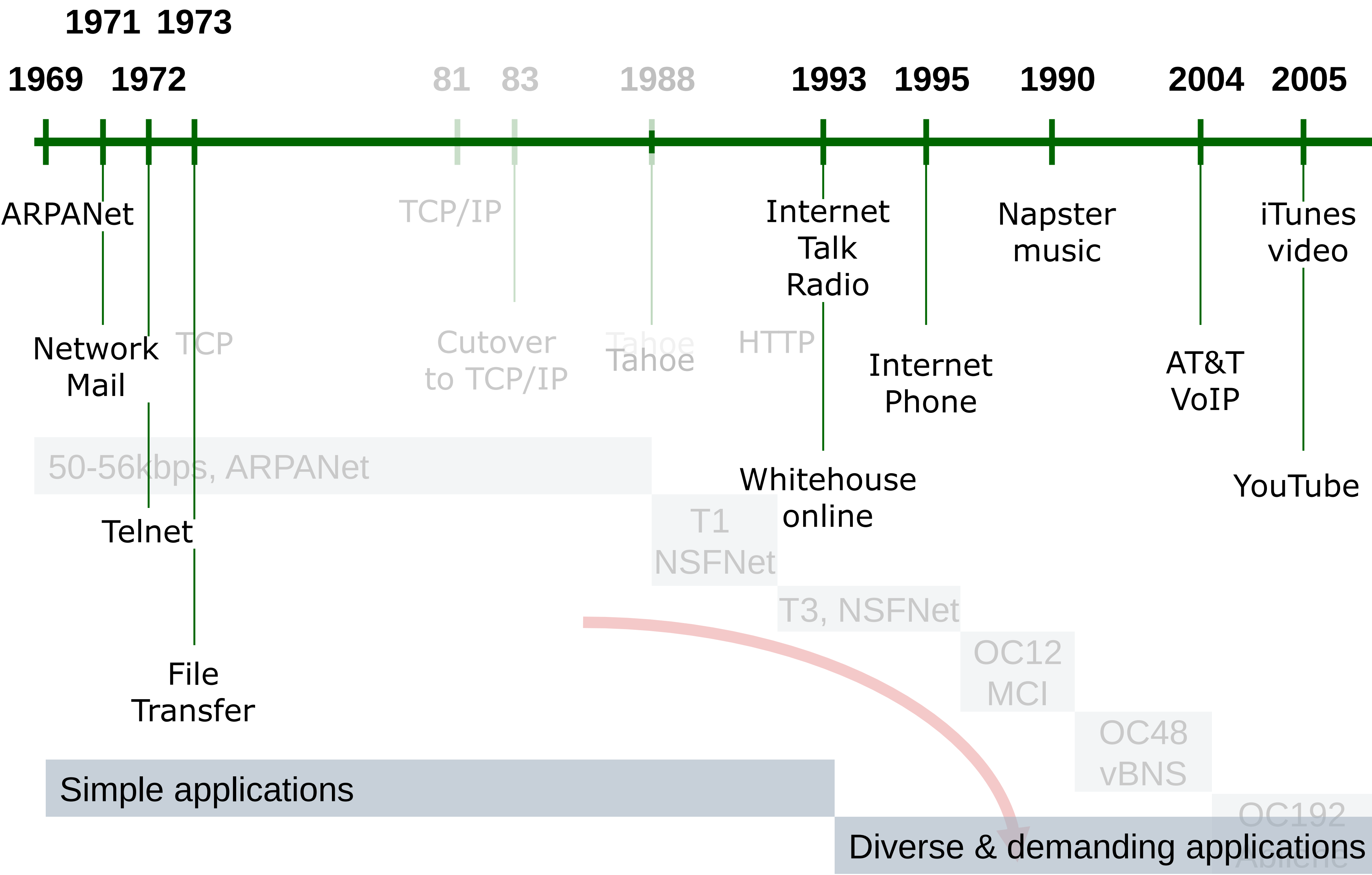
WHY ?



Network milestones



Application milestones



Network Mail (1971)

First Internet (ARPANet) application

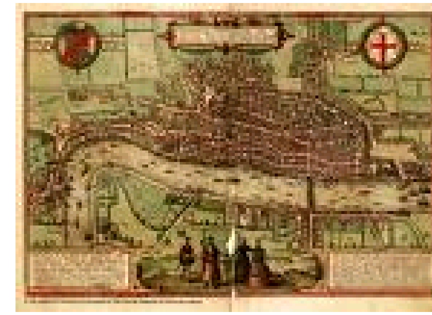
The first network email was sent by Ray Tomlinson between these two computers at BBN that are connected by the ARPANet.

Internet applications (2006)

Telephony

Music

TV & home theatre

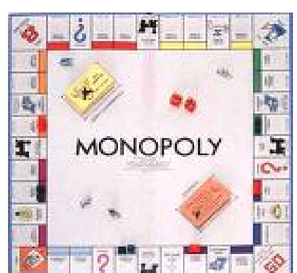


Finding your way

Mail

Friends

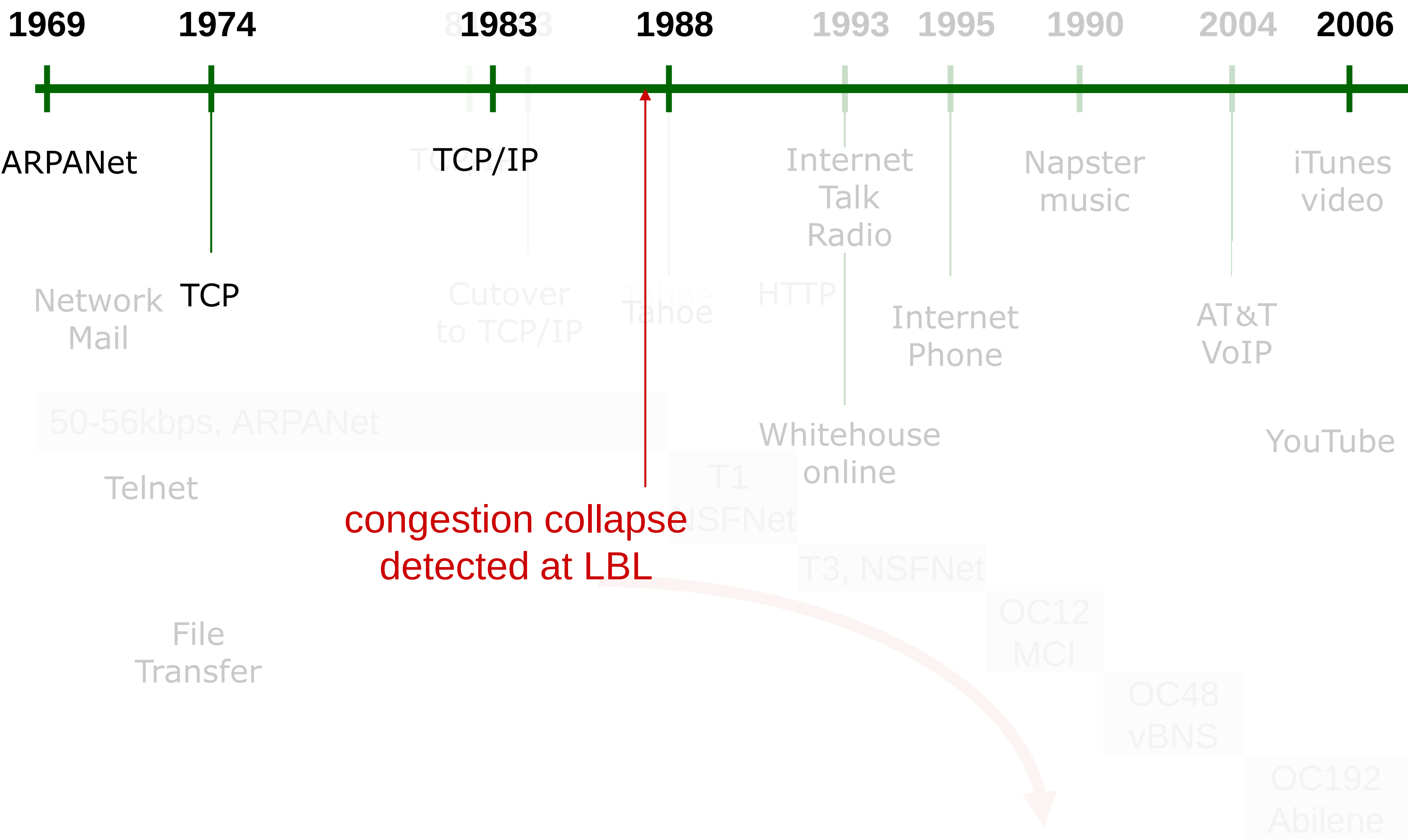
Library at your finger tip



Games

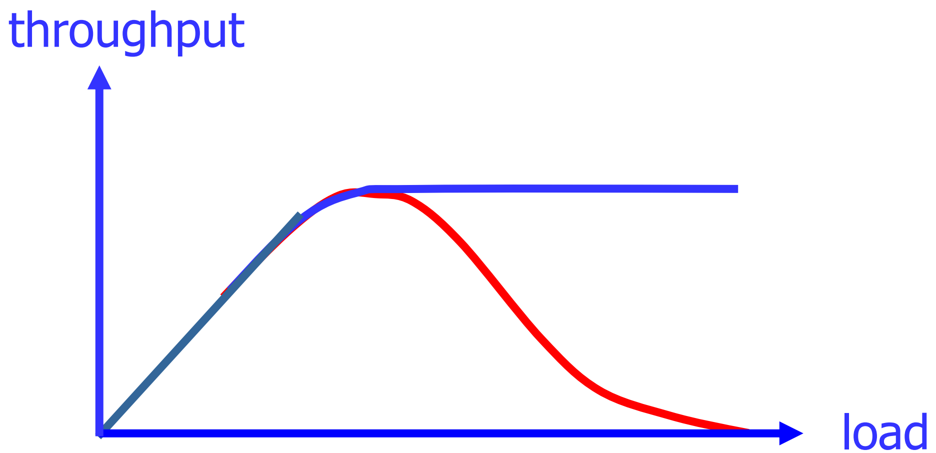
Cloud computing

Congestion collapse

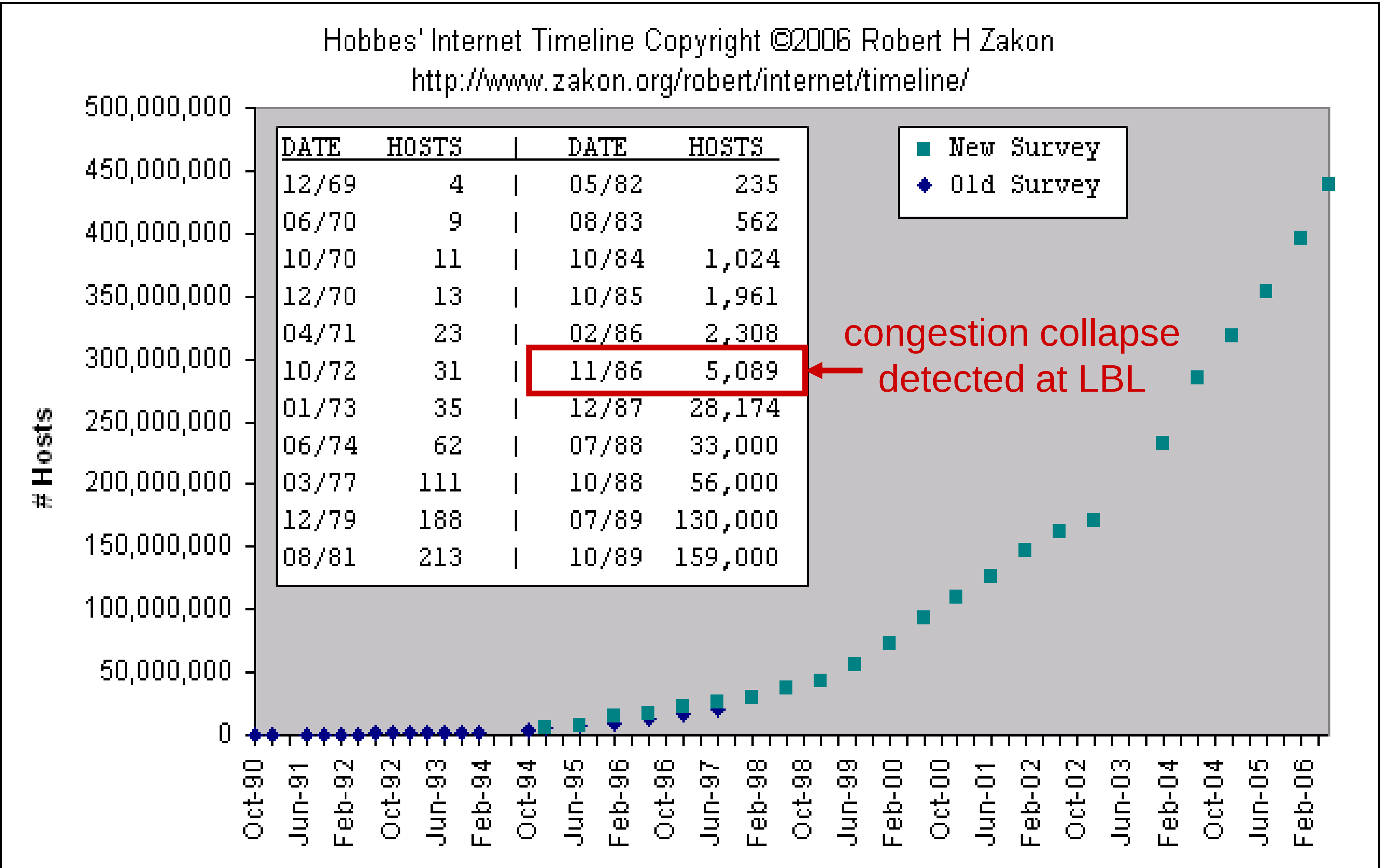


Congestion collapse

- October 1986, the first congestion collapse on the Internet was detected
- Link between UC Berkeley and LBL
  - 400 yards, 3 hops, 32 Kbps
  - throughput dropped to 40 bps
  - factor of ~1000 drop!
- 1988, Van Jacobson proposed TCP congestion control



Why the 1986 collapse



DATE	HOSTS	DATE	HOSTS
12/69	4	05/82	235
06/70	9	08/83	562
10/70	11	10/84	1,024
12/70	13	10/85	1,961
04/71	23	02/86	2,308
10/72	31	11/86	5,089
01/73	35	12/87	28,174
06/74	62	07/88	33,000
03/77	111	10/88	56,000
12/79	188	07/89	130,000
08/81	213	10/89	159,000

Why the 1986 collapse

- 5,089 hosts on Internet (Nov 1986)
- Backbone speed: 50 – 56 kbps
- Control mechanism focused only on receiver congestion, not network congestion

- Large number of hosts sharing a slow (and small) network
  - <u>Network</u> became the bottleneck, as opposed to receivers
  - But TCP flow control <u>only</u> prevented overwhelming receivers

Jacobson introduced feedback control to deal with network congestion in 1988

Tahoe and its variants (1988)

Jacobson, Sigcomm 1988

\+ Avoid overwhelming network

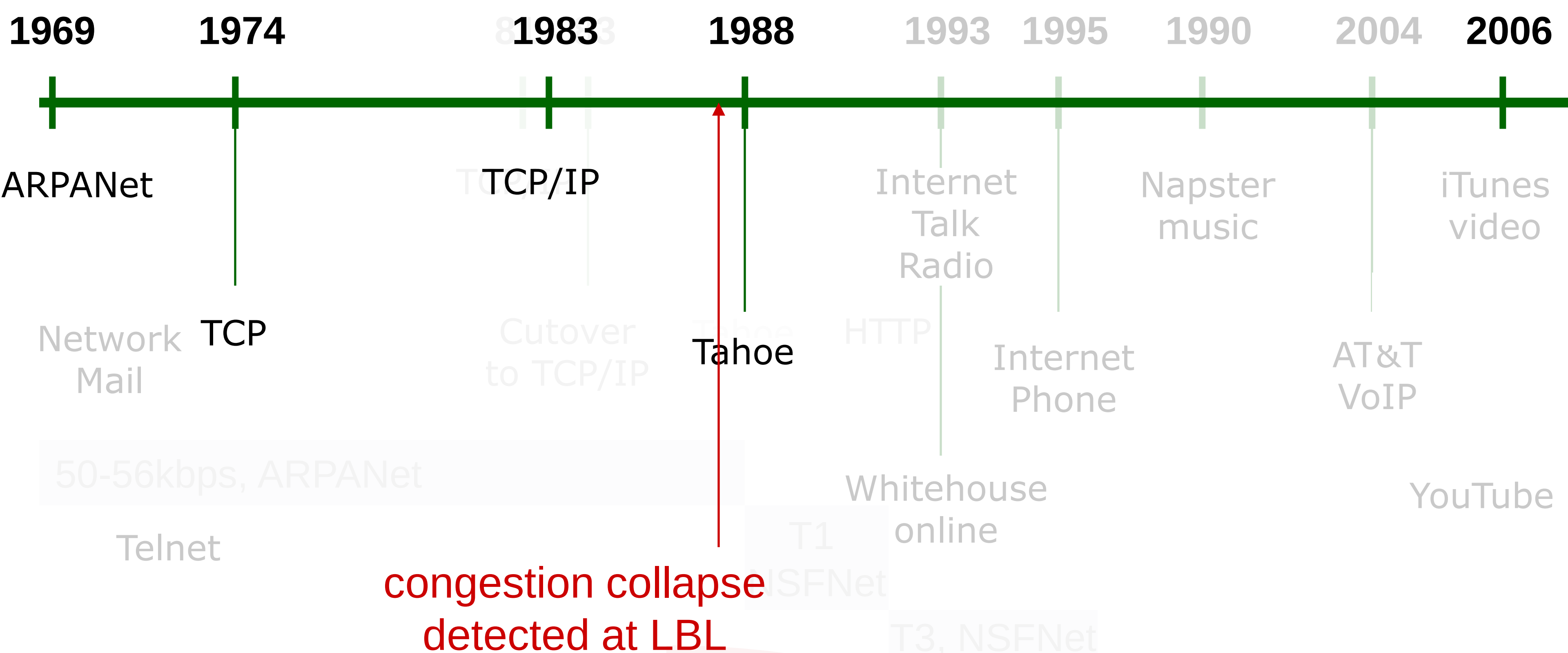
\+ Window control mechanisms

- Dynamically adjust sender window based on congestion (as well as receiver window)
- Loss-based AIMD
- Based on idea of Chiu, Jain, Ramakrishnan

"… important considering that TCP spans a range from 800 Mbps Cray channels to 1200 bps packet radio links"

-- Jacobson, 1988

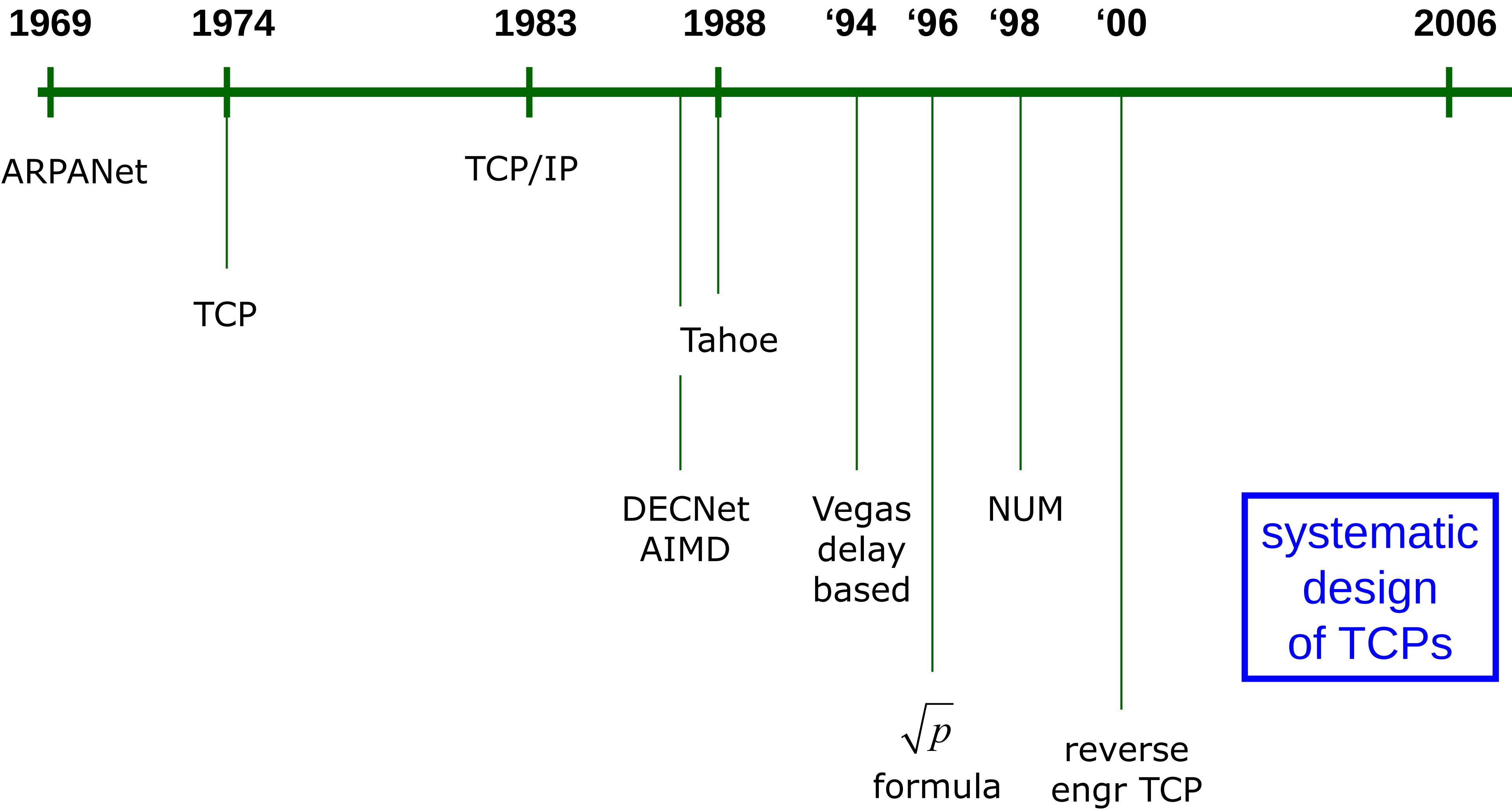
TCP congestion control



Flow control:
Prevent overwhelming receiver

+ Congestion control:
Prevent overwhelming network

Transport milestones



Congestion control protocols

Why congestion control?

Where is CC implemented?

Window control mechanism

CC protocols and basic structure

Active queue management (AQM)

Packet networks

Packet-switched as opposed to circuit-switched

- No dedicated resources
- Simple & robust: states in packets

More efficient sharing of resources

- Multiplexing gain

Less guarantee on performance

- Best effort

Network mechanisms

Transmit bits across a link

- encoding/decoding, mod/dem, synchronization

Medium access

- who transmits when for how long

Routing

- choose path from source to destination

Loss recovery

- recover packet loss due to congestion, error, interference

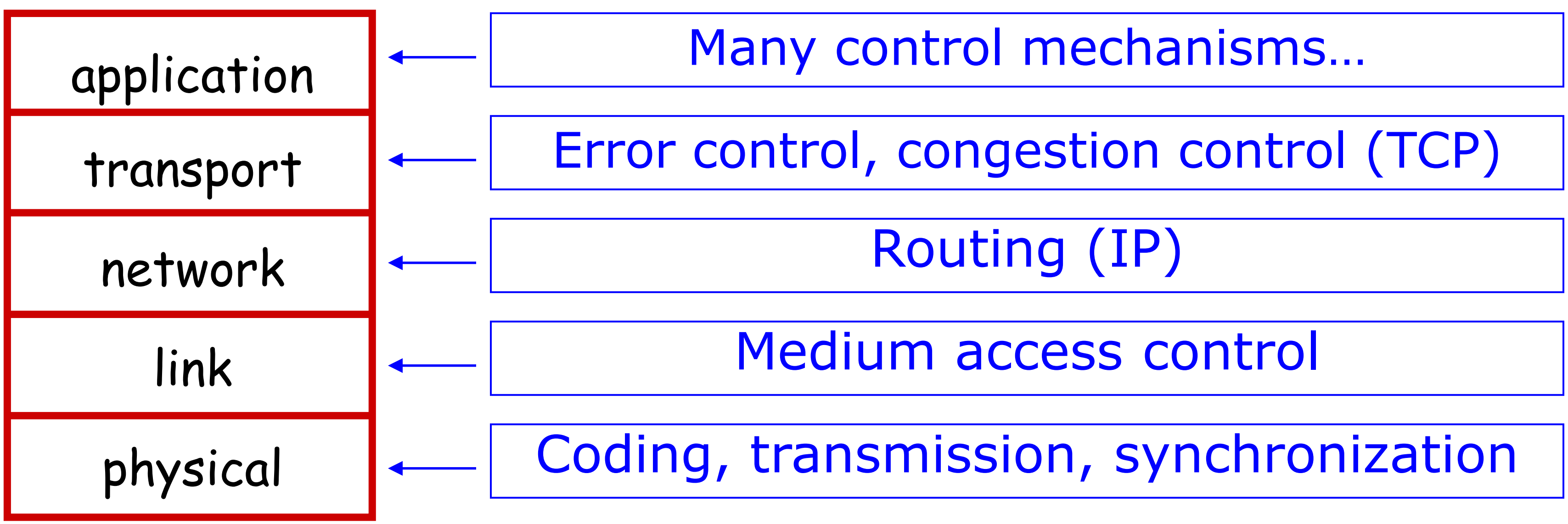
Flow/congestion control

- efficient use of bandwidth/buffer without overwhelming receiver/network

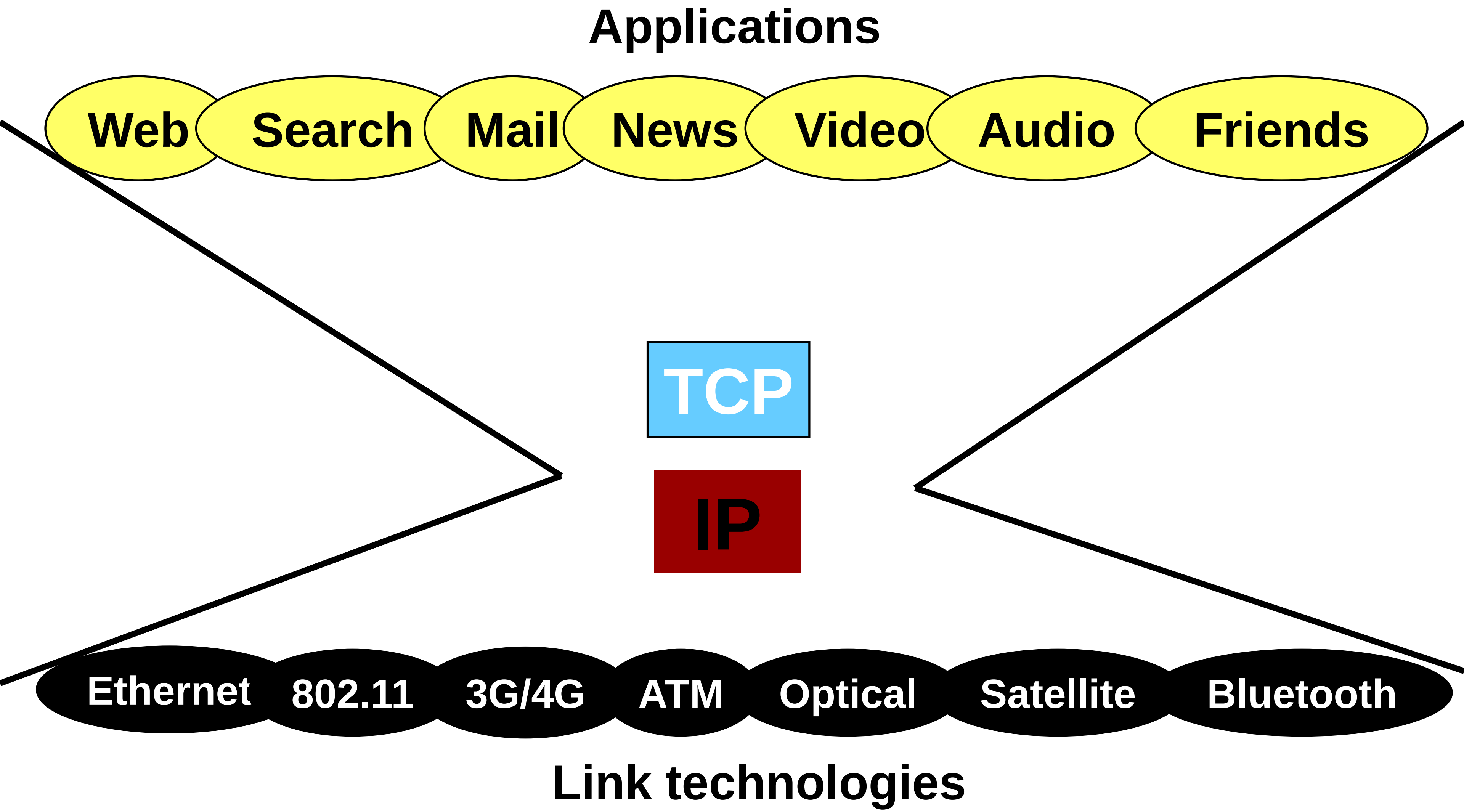
Protocol stack

Network mechanisms implemented as protocol stack

Each layer designed separately, evolves asynchronously



The Internet hourglass



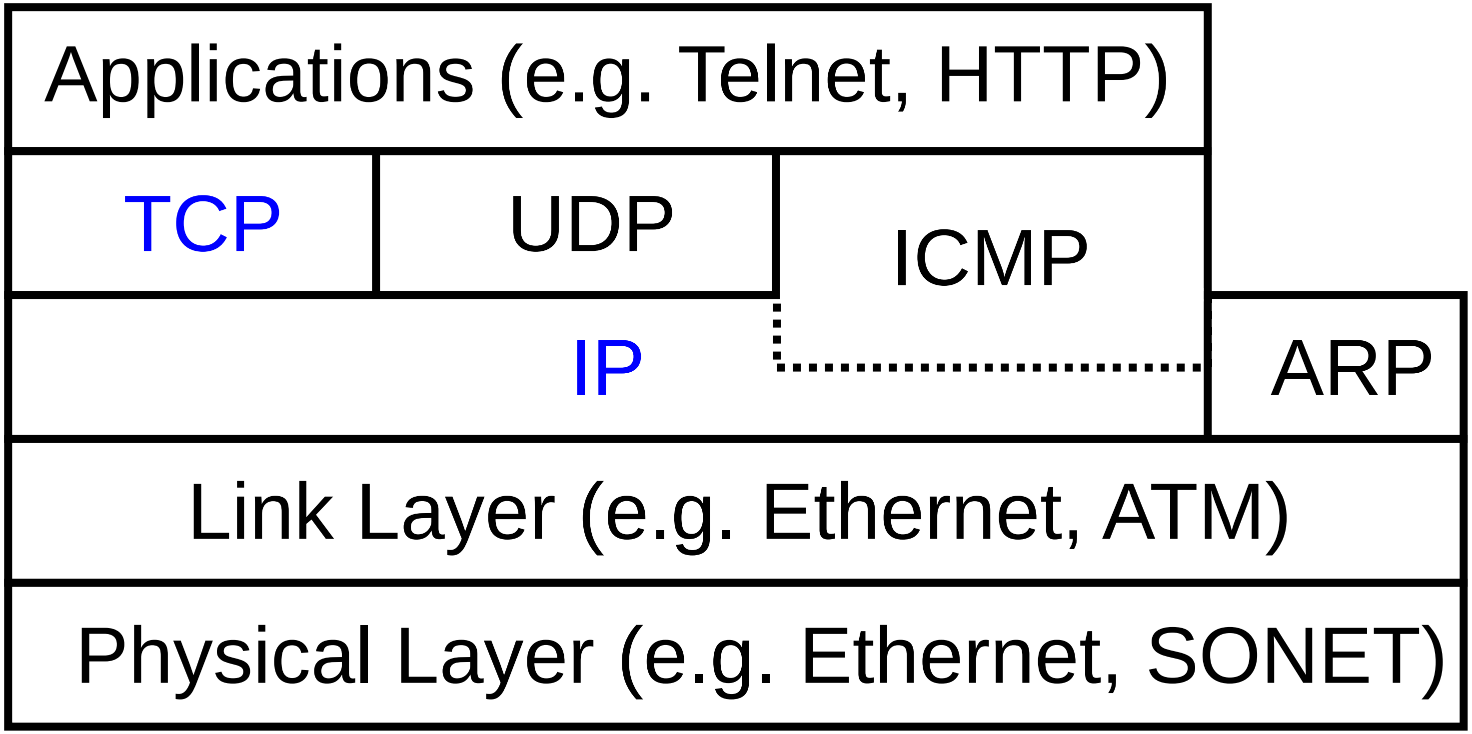
IP layer

- Routing from source to destination
  - Distributed computation of routing decisions
  - Implemented as routing table at each router
  - Shortest-path (Dijkstra) algorithm within an autonomous system
  - BGP across autonomous systems
- Datagram service
  - Best effort
  - Unreliable: lost, error, out-of-order
- Simple and robust
  - Robust against failures
  - Robust against, and enables, rapid technological evolution above & below IP

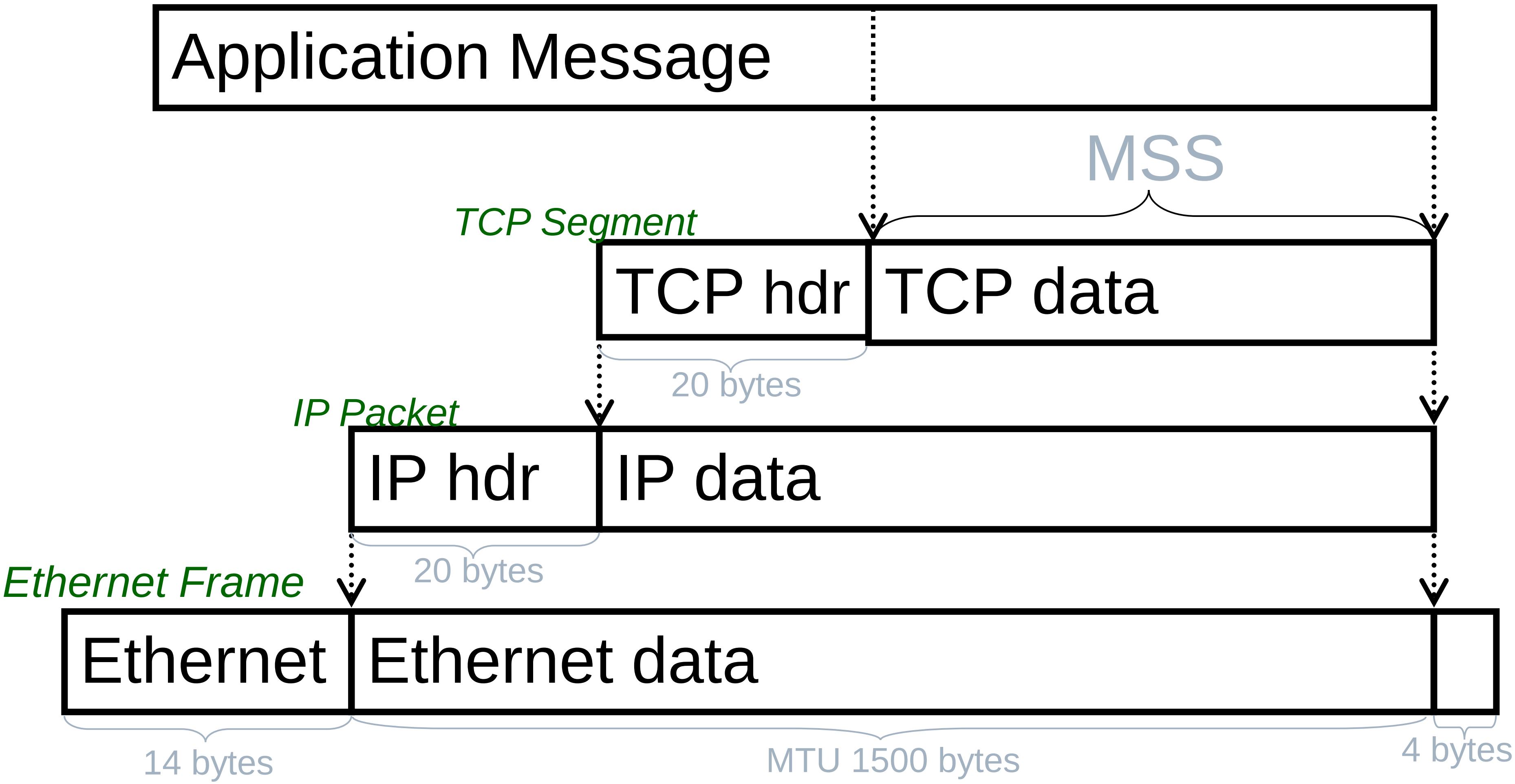
TCP layer

- End-to-end reliable byte stream
  - On top of unreliable datagram service
  - Correct, in-order, without loss or duplication
- Connection setup and tear down
  - 3-way handshake
- Loss and error recovery
  - CRC to detect bit error
  - Sequence number to detect packet loss/duplication
  - Retransmit packets lost or contain errors
- Congestion control
  - Source-based distributed control

Protocol data format



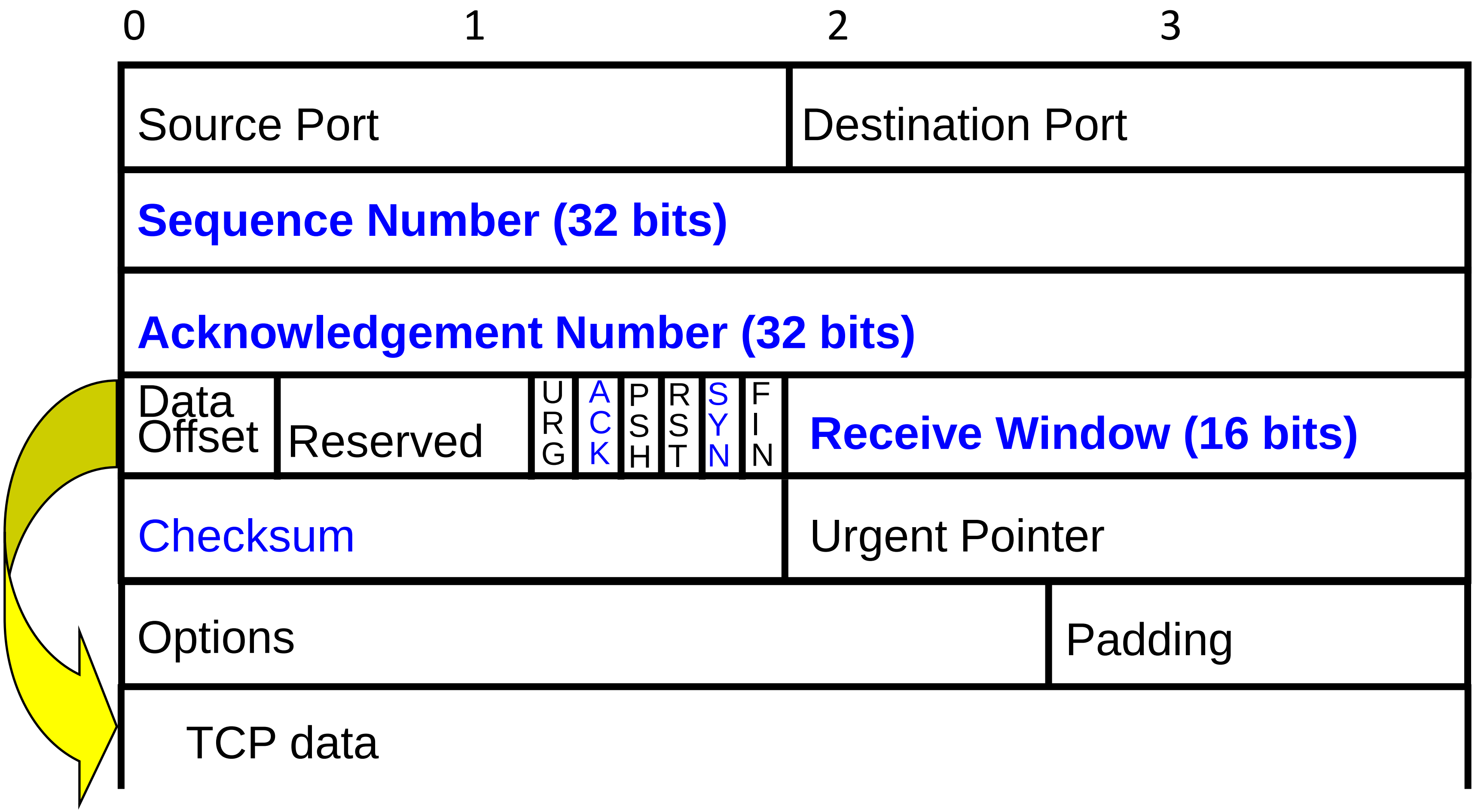
Protocol data format



IP Header

<table>
<tr><th colspan="2">0</th><th colspan="2">1</th><th colspan="2">2</th><th colspan="2">3</th></tr>
<tr><td>Vers(4)</td><td>H len</td><td colspan="2">Type of Service</td><td colspan="4">Total Length (16 bits)</td></tr>
<tr><td colspan="4">Identification</td><td>Flags</td><td colspan="3">Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td colspan="2">Protocol (TCP=6)</td><td colspan="4">Header Checksum</td></tr>
<tr><td colspan="8">Source IP Address</td></tr>
<tr><td colspan="8">Destination IP Address</td></tr>
<tr><td colspan="6">Options</td><td colspan="2">Padding</td></tr>
<tr><td colspan="8">IP data</td></tr>
</table>

TCP Header



0
1
2
3
Source Port
Destination Port
Sequence Number (32 bits)
Acknowledgement Number (32 bits)
Data Offset
Reserved
URG
ACK
PSH
RST
SYN
FIN
Receive Window (16 bits)
Checksum
Urgent Pointer
Options
Padding
TCP data

Congestion control protocols

Why congestion control?

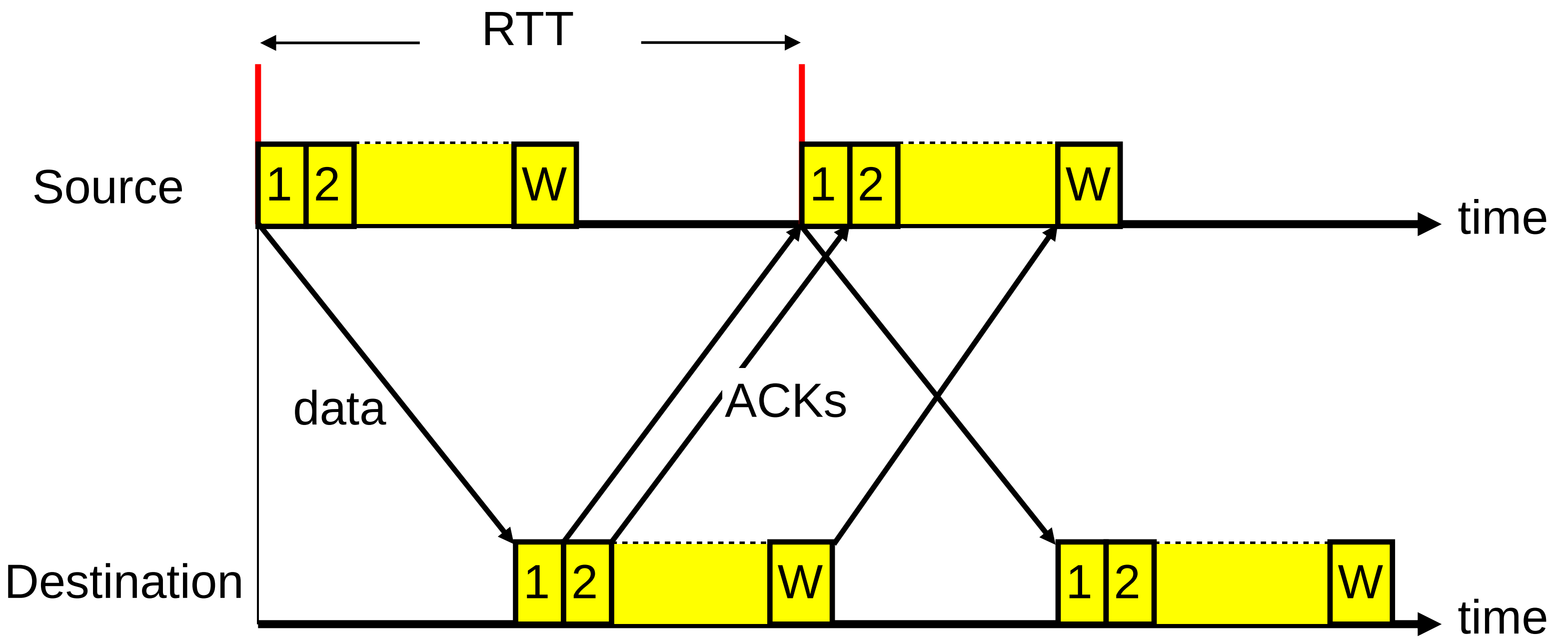
Where is CC implemented?

Window control mechanism

CC protocols and basic structure

Active queue management (AQM)

Window control



- ~ W packets per RTT
- Lost packet detected by missing ACK
- Self-clocking: regulates flow

Source rate

Limit the number of packets in the network to window W

$$\text{Source rate} = \frac{W \times MSS}{RTT} \text{ bps}$$

If W too small then rate < capacity

else rate > capacity (➔ congestion)

How to decide W?

Early TCP

- Pre 1988
- Go-back-N ARQ
  - Detects loss from timeout
  - Retransmits from lost packet onward
- Receiver window flow control
  - Prevents overflow at receive buffer
  - Receiver sets awnd in TCP header of each ACK
    - Closes when data received and ack'ed
    - Opens when data delivered to application
  - Sender sets W = awnd
- Self-clocking

TCP congestion control

Post 1988

ARQ, awnd from ACK, self-clocking

In addition:

Source calculates cwnd from indication of network congestion

- Packet loss
- Packet delay
- Marks, explicit congestion notification

Source sets W = min (cwnd, awnd)

Algorithms to calculate cwnd

- Reno, Vegas, FAST, CUBIC, CTCP, …

Congestion control protocols

Why congestion control?

Where is CC implemented?

Window control mechanism

CC protocols and basic structure

Active queue management (AQM)

Key references

TCP/IP spec

- RFC 791 Internet Protocol
- RFC 793 Transmission Control Protocol

AIMD idea: Chiu, Jain, Ramakrishnan 1988-90

Tahoe/Reno: Jacobson 1988

Vegas: Brakmo and Peterson 1995

FAST: Jin, Wei, Low 2004

CUBIC: Ha, Rhee, Xu 2008

CTCP: Kun et al 2006

RED: Floyd and Jacobson 1993

REM: Athuraliya, Low, Li, Yin 2001

There are many many other proposals and references

TCP Congestion Control

Has four main parts

- Slow Start (SS)
- Congestion Avoidance (CA)
- Fast Retransmit
- Fast Recovery

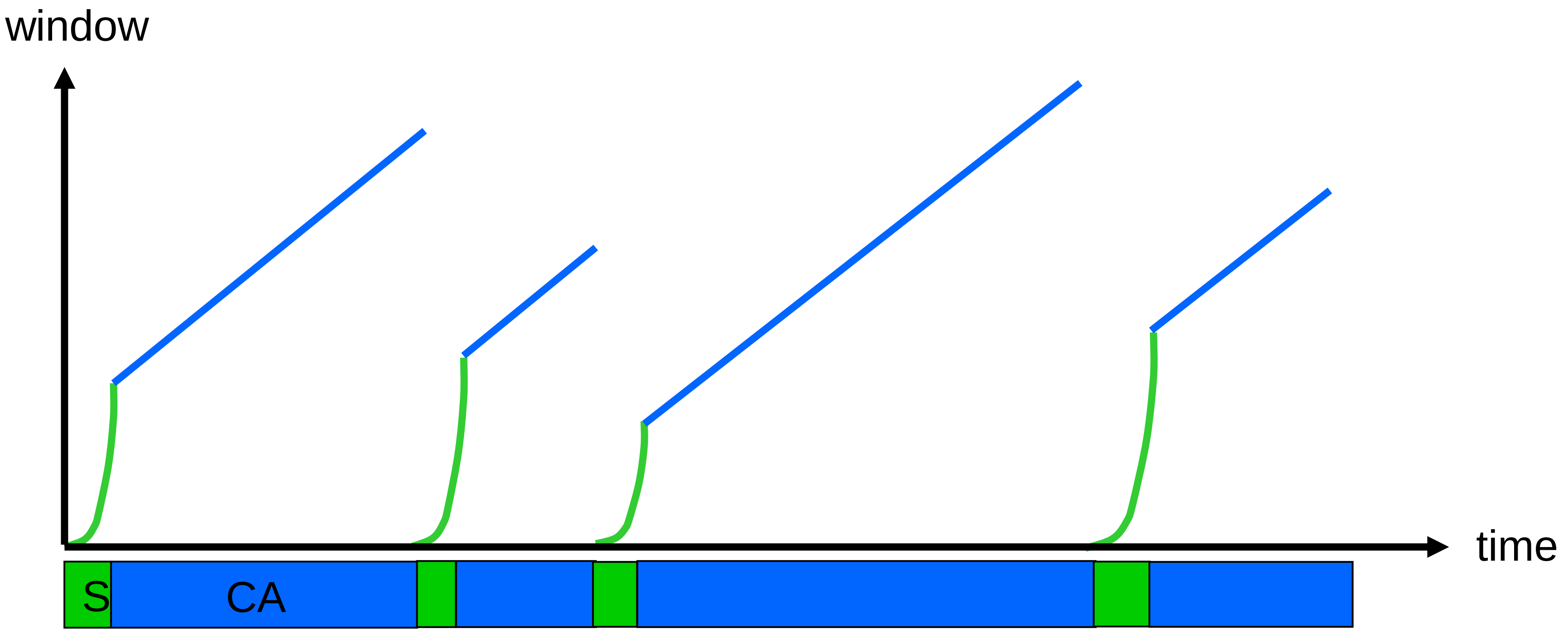
Tahoe

Reno

ssthresh: slow start threshold determines whether to use SS or CA

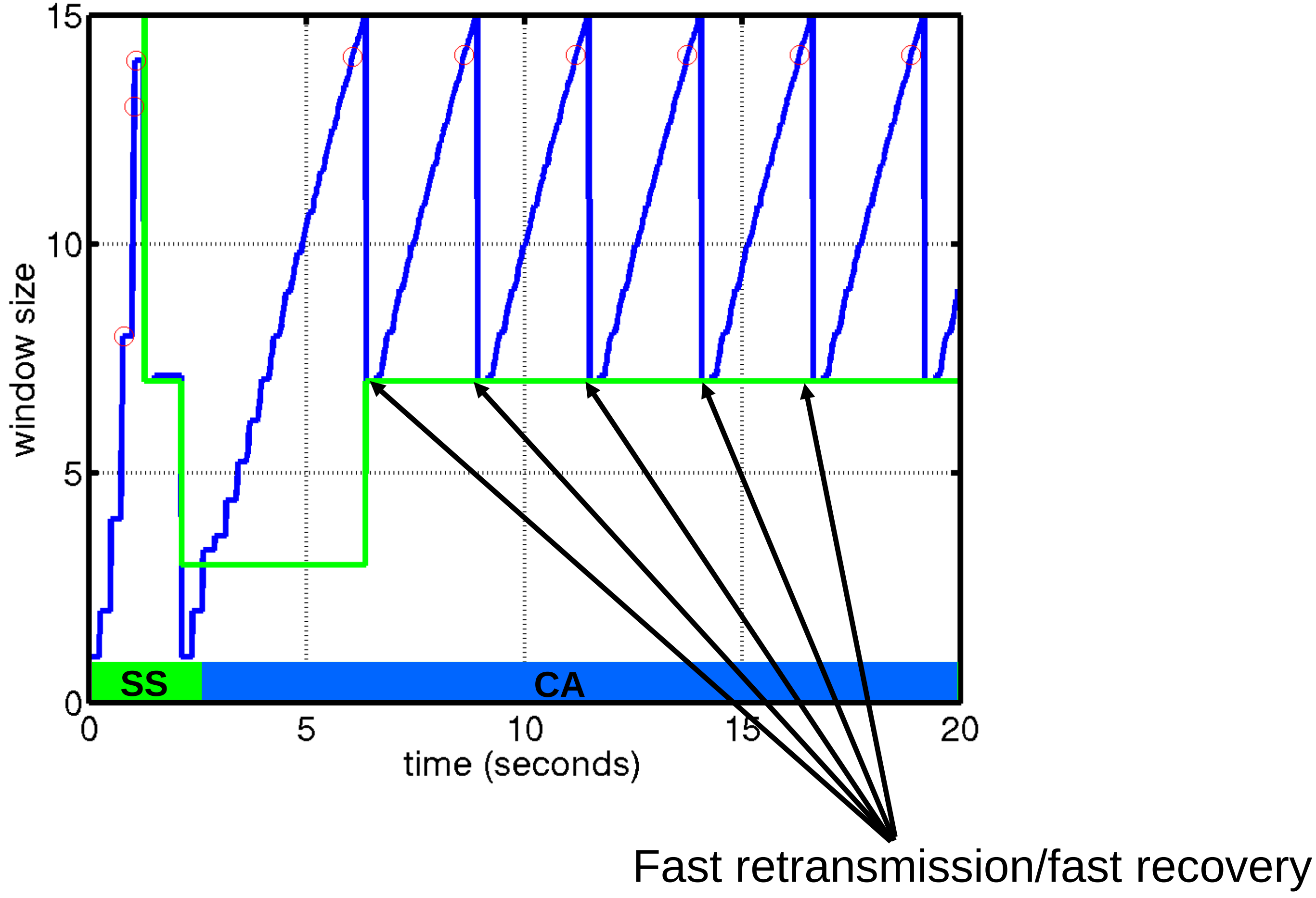
Assumption: packet losses are caused by buffer overflow (congestion)

TCP Tahoe (Jacobson 1988)



SS: Slow Start
CA: Congestion Avoidance

TCP Reno (Jacobson 1990)



Slow Start

Start with $cwnd = 1$ (slow start)

On each successful ACK increment cwnd

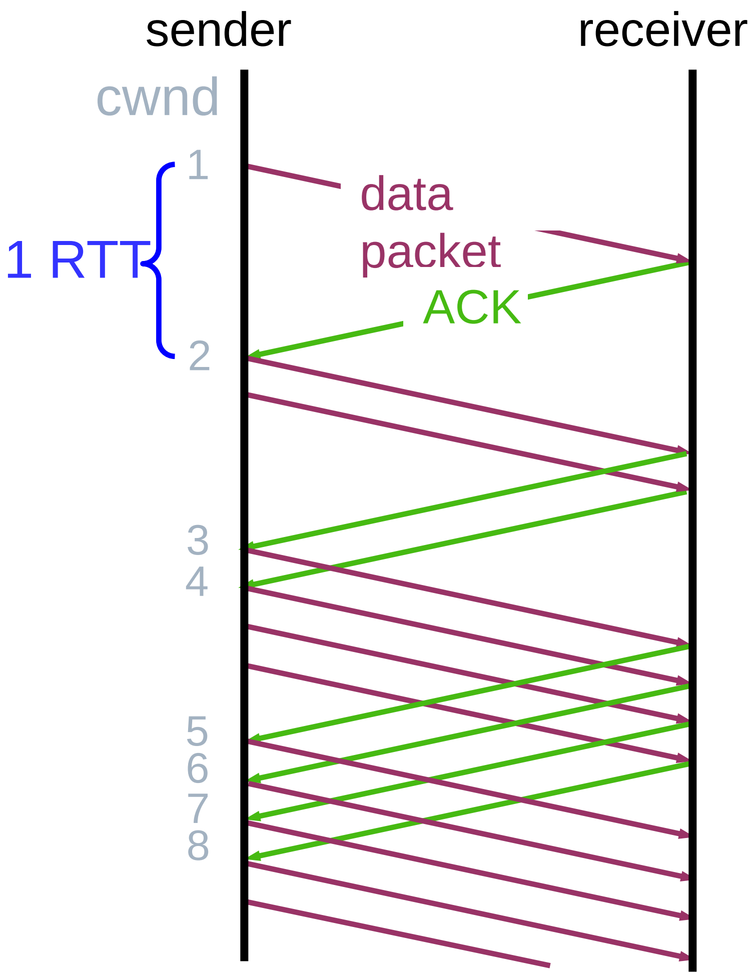
$$cwnd \leftarrow cnwd + 1$$

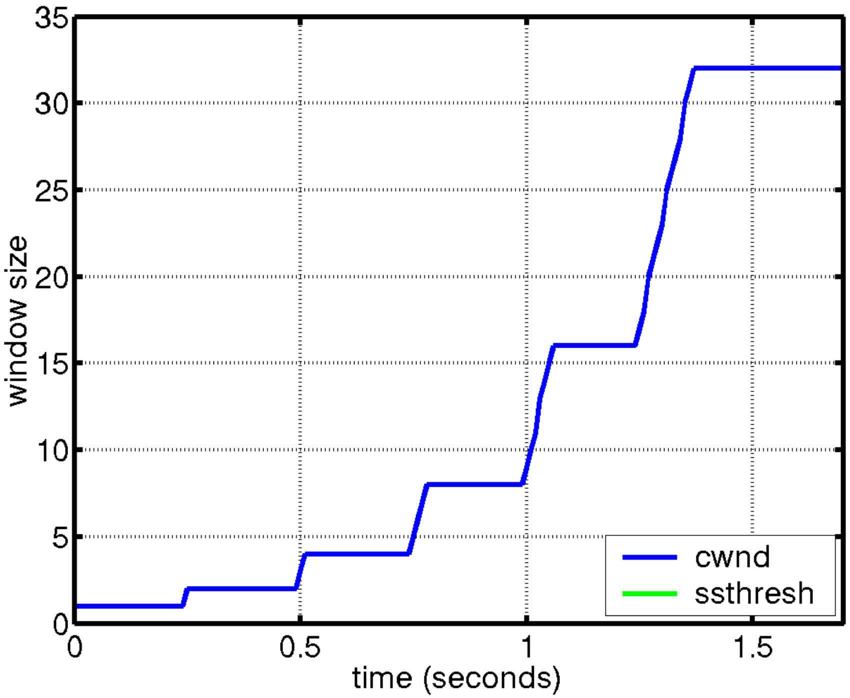
Exponential growth of cwnd

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

Enter CA when $cwnd >= ssthresh$

Slow Start





cwnd $\leftarrow$ cwnd + 1 (for each ACK)

Congestion Avoidance

Starts when cwnd $\geq$ ssthresh

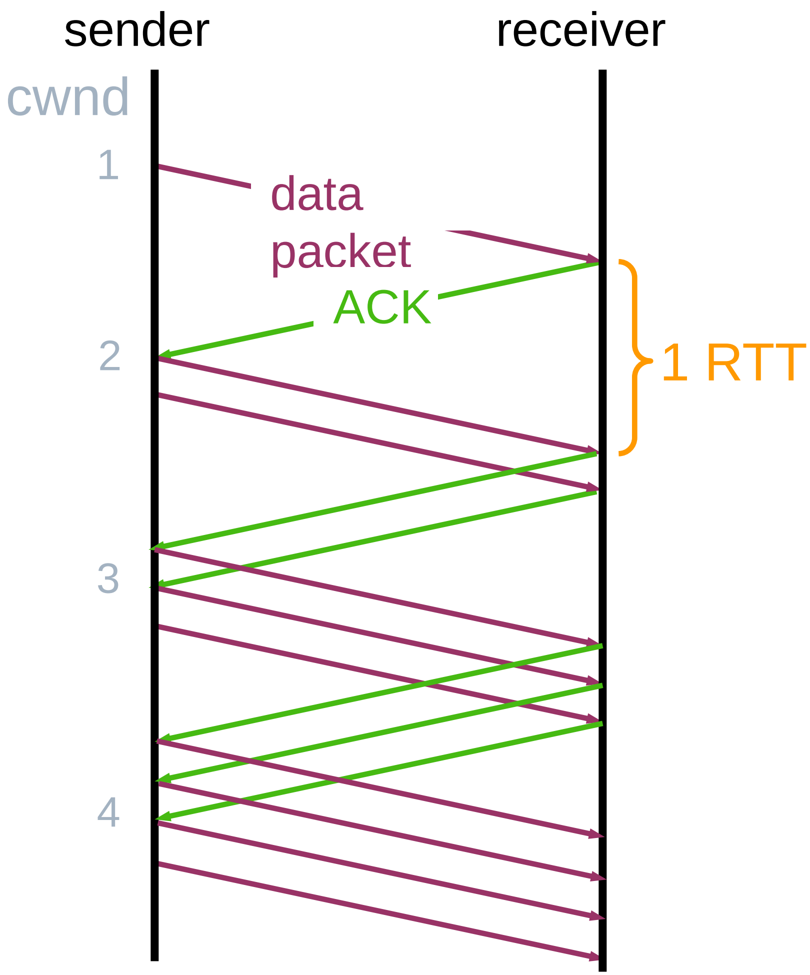
On each successful ACK:

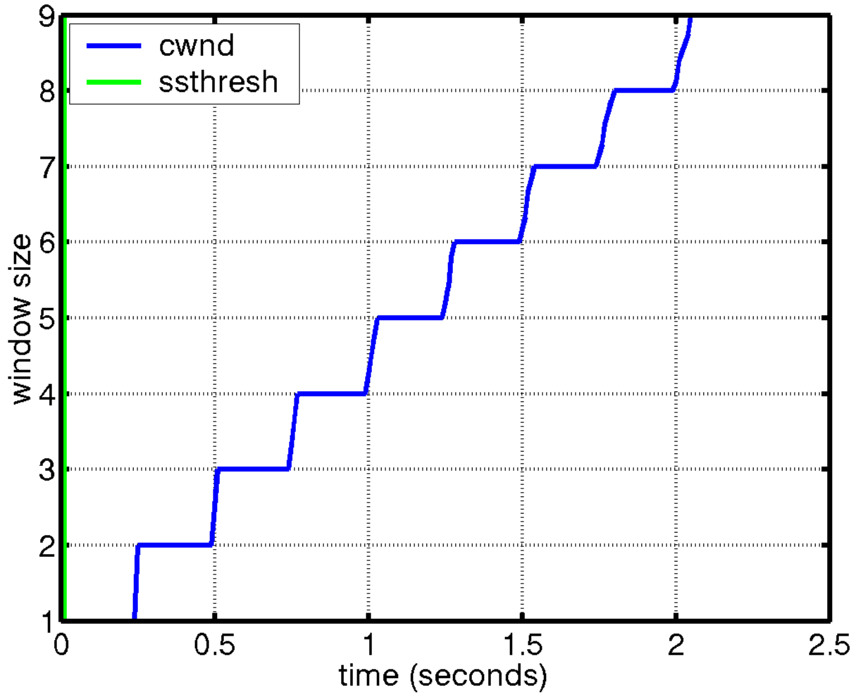
$\text{cwnd} \leftarrow \text{cwnd} + 1/\text{cwnd}$

Linear growth of cwnd

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

Congestion Avoidance

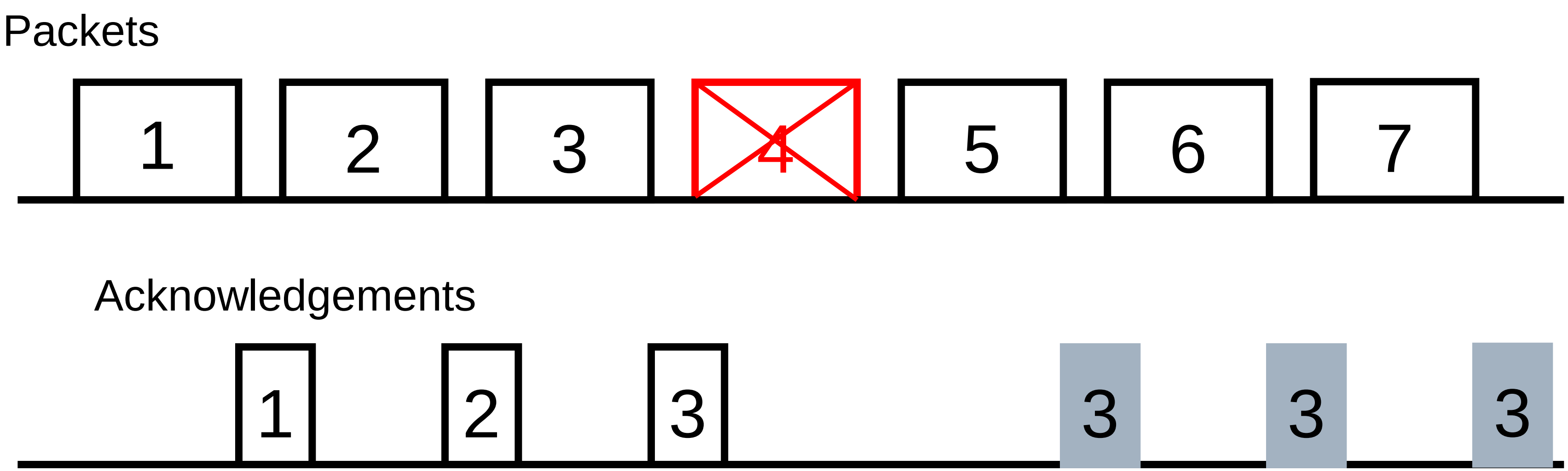




cwnd ← cwnd + 1 (for cwnd worth of ACKs)

Packet Loss

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



Fast Retransmit/Fast Recovery

Motivation

- Waiting for timeout is too long
- Prevent `pipe' from emptying during recovery

Idea

- 3 dupACKs indicate packet loss
- Each dupACK also indicates a packet having left the pipe (successfully received)!

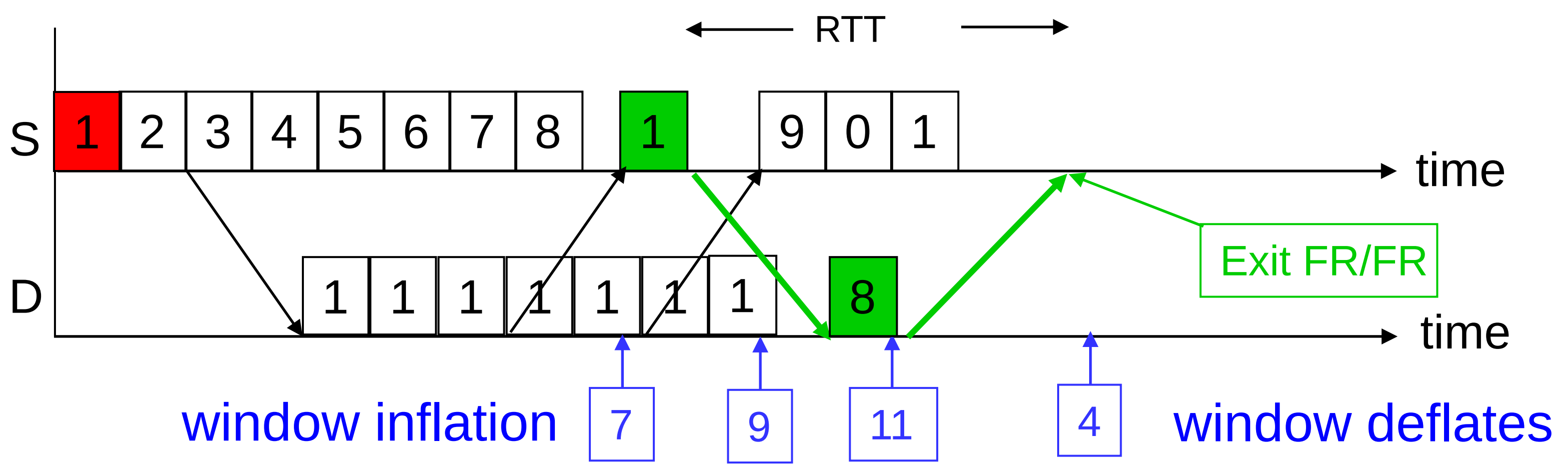
Fast Retransmit/Fast Recovery

Enter FR/FR after 3 dupACKs

- Set ssthresh $\leftarrow$ max(flightsize/2, 2)
- Retransmit lost packet
- Set cwnd $\leftarrow$ ssthresh + ndup (window inflation)
- Wait till W=min(awnd, cwnd) is large enough; transmit new packet(s)
- On non-dup ACK (1 RTT later), set cwnd $\leftarrow$ ssthresh (window deflation)

Enter CA (unless timeout)

Example: FR/FR



Fast retransmit

- Retransmit on 3 dupACKs

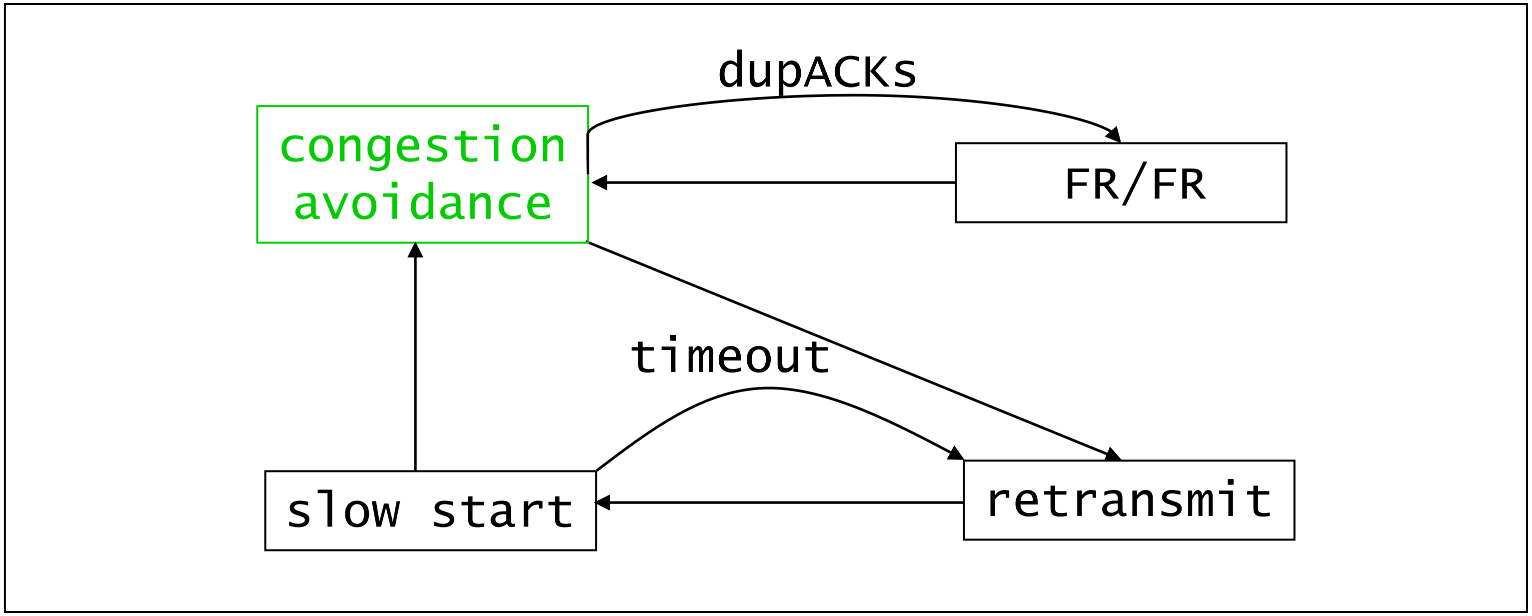
Fast recovery

- Inflate window while repairing loss to fill pipe

Summary: Reno

Basic idea

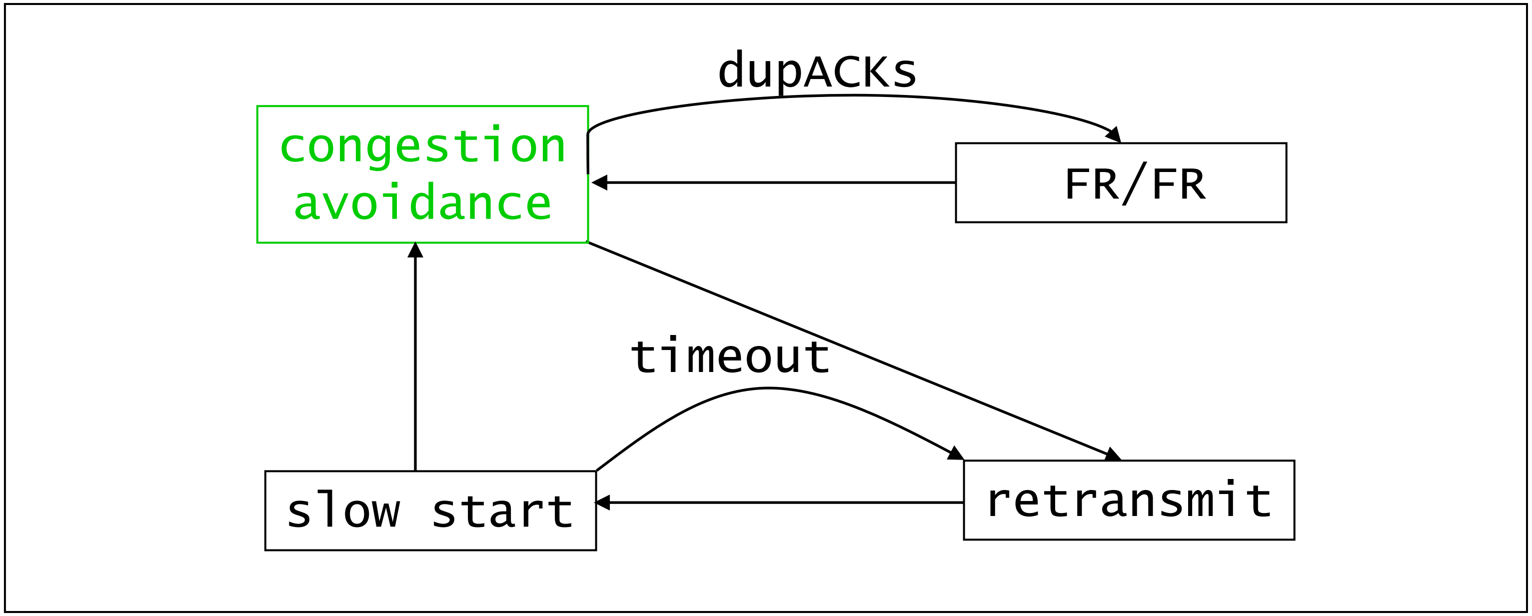
- AIMD probes available bandwidth
- Fast recovery avoids slow start
- dupACKs: fast retransmit + fast recovery
- Timeout: fast retransmit + slow start



TCP CC variants

Differ mainly in Congestion Avoidance

- Vegas: delay-based
- FAST: delay-based, scalable
- CUBIC: time since last congestion
- CTCP: use both loss & delay



Congestion avoidance

Reno

Jacobson
1988

```
for every ACK {
    W += 1/W                (AI)
}
for every loss {
    W = W/2                 (MD)
}
```

Vegas

Brakmo
Peterson
1995

for every ACK {

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

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

}

for every loss {

  W = W/2

}

Congestion avoidance

FAST

Jin, Wei, Low

2004

```
periodically
{
```

$$W = \frac{\text{baseRTT}}{\text{RTT}} W + a$$

```
}
```

Congestion control protocols

Why congestion control?

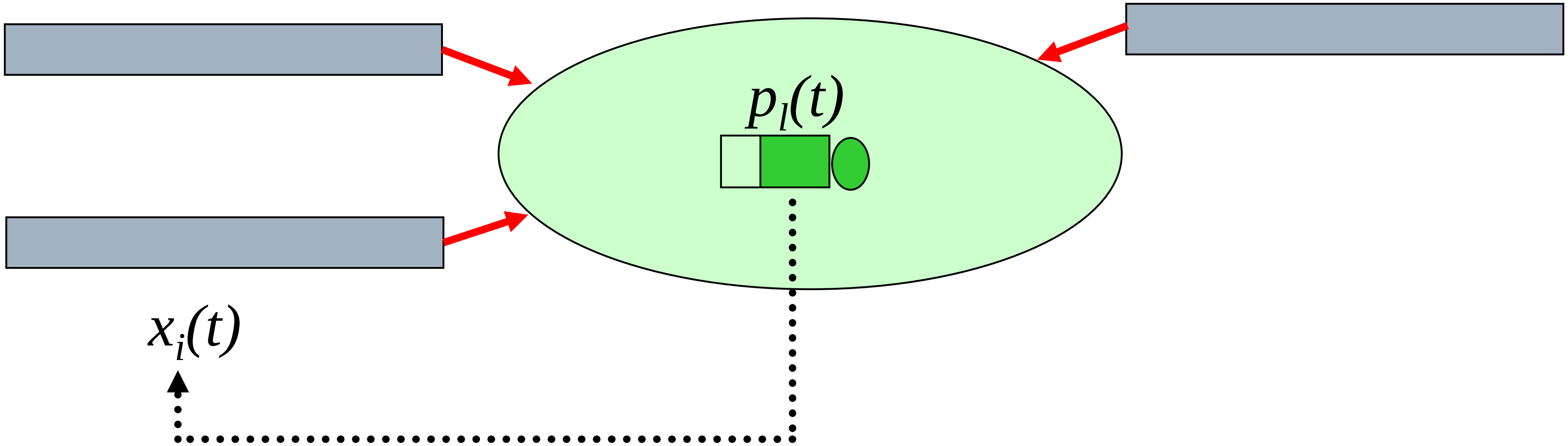
Where is CC implemented?

Window control mechanism

CC protocols and basic structure

Active queue management (AQM)

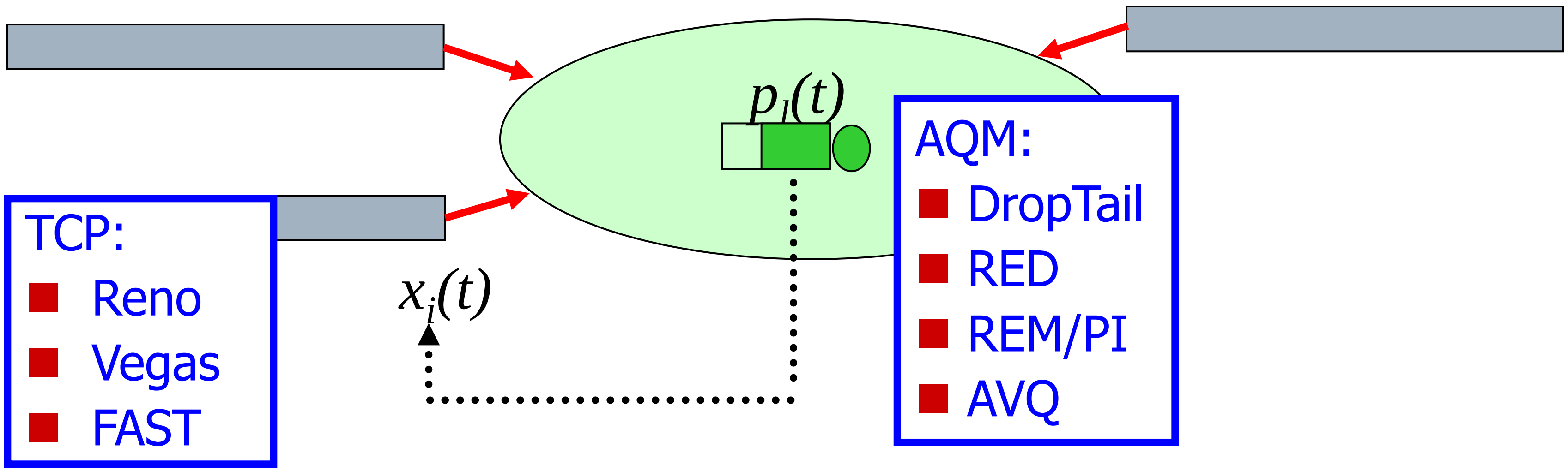
Feedback control



Example congestion measure $p_l(t)$

- Loss (Reno)
- Queueing delay (Vegas)

TCP/AQM



Congestion control is a distributed asynchronous algorithm to share bandwidth

It has two components

- TCP: adapts sending rate (window) to congestion
- AQM: adjusts & feeds back congestion information

They form a distributed feedback control system

- Equilibrium & stability depends on both TCP and AQM
- And on delay, capacity, routing, #connections

Implicit feedback

Drop-tail

- FIFO queue
- Drop packet that arrives at a full buffer

Implicit feedback

- Queueing process implicitly computes and feeds back congestion measure
- Delay: simple dynamics
- Loss: no convenient model

Active queue management

Explicit feedback

- Provide congestion information by probabilistically marking packets
- 2 ECN bit in IP header allocated for AQM

Supported by all new routers but usually turned off in the field

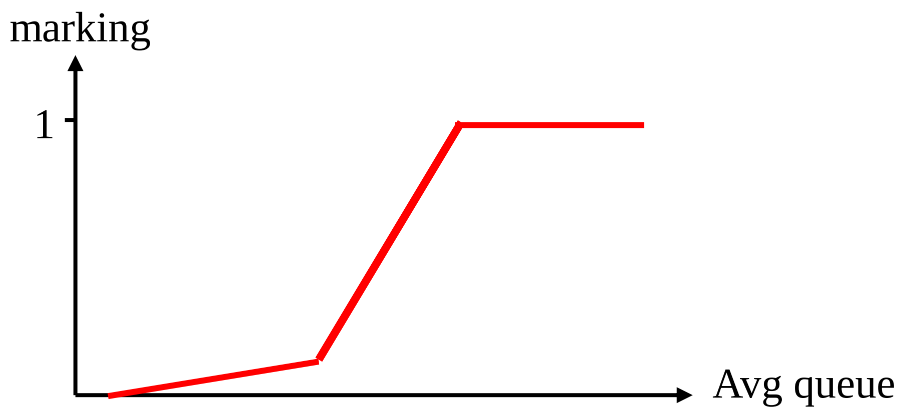
RED (Floyd & Jacobson 1993)

Congestion measure: average queue length

$$b_l(t+1) \;=\; [b_l(t) + y_l(t) - c_l]^+$$

$$r_l(t+1) \;=\; (1-\alpha)\, r_l(t) + \alpha\, b_l(t)$$

Embedding: p-linear probability function



Feedback: dropping or ECN marking

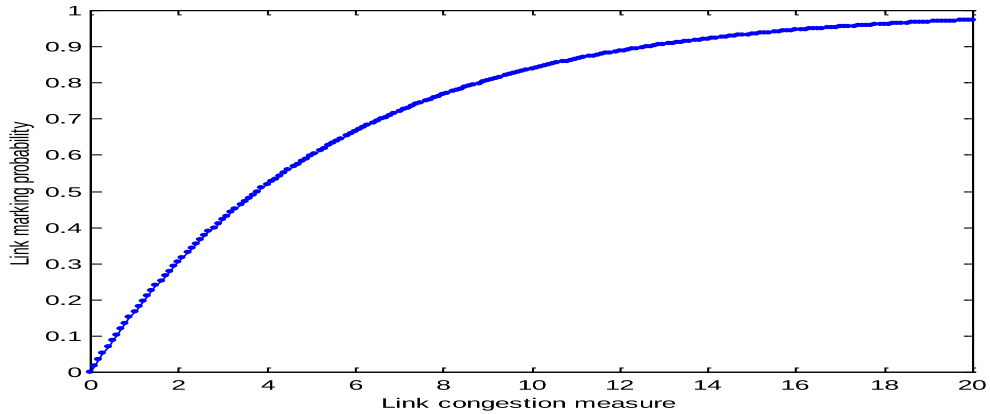
REM (Athuraliya & Low 2000)

Congestion measure: price

$$b_l(t+1) = [b_l(t) + y_l(t) - c_l]^+$$

$$p_l(t+1) = [p_l(t) + \gamma(\alpha_l\, b_l(t) + x^l(t) - c_l)]^+$$

Embedding: exponential probability function



Feedback: dropping or ECN marking

REM

Clear buffer and match rate

$$p_l(t+1) = [p_l(t) + \gamma(\underbrace{\alpha_l b_l(t)}_{\text{Clear buffer}} + \underbrace{\hat{x}^l(t) - c_l}_{\text{Match rate}})]^+$$

Sum prices

$$1 - \phi^{-p_l(t)} \;\Rightarrow\; 1 - \phi^{-p^s(t)}$$

Theorem (Paganini 2000)

Global asymptotic stability for general utility function (in the absence of delay)

Summary: CC protocols

End-to-end CC implemented in TCP

- Basic window mechanism
- TCP performs connection setup, error recovery, and congestion control,
- CC dynamically computes cwnd that limits max #pkts enroute

Distributed feedback control algorithm

- TCP: adapts congestion window
- AQM: adapts congestion measure

Agenda

9:00	Congestion control protocols
10:00	break
10:15	Mathematical models
11:15	break
11:30	Advanced topics
12:30	lunch

MATHEMATICAL MODELS

Mathematical models

Why mathematical models?

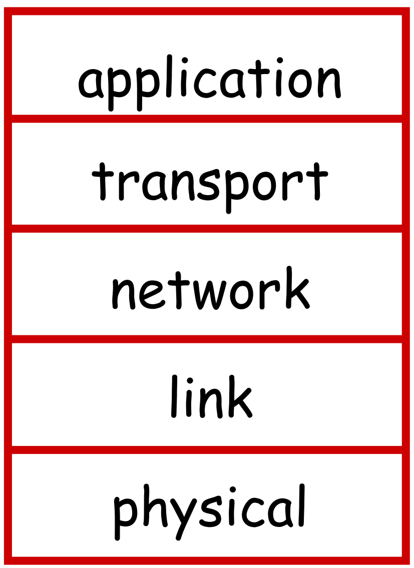
Dynamical systems model of CC

Convex optimization primer

Reverse engr: equilibrium properties

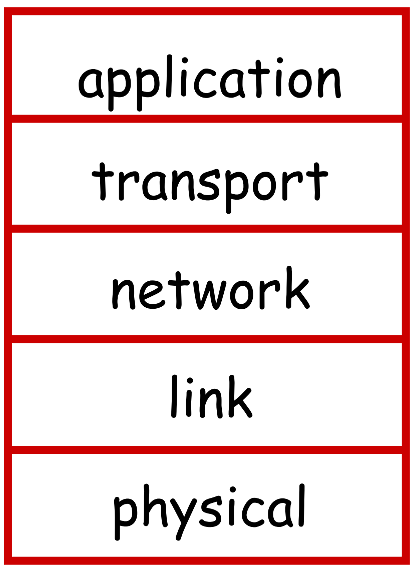
Forward engr: FAST TCP

Why mathematical models



- Protocols are critical, yet difficult, to understand and optimize
- Local algorithms, distributed spatially and vertically → global behavior
- Designed separately, deployed asynchronously, evolves independently

Why mathematical models



Need systematic way to understand, design, and optimize

- Their interactions
- Resultant global behavior

Why mathematical models

Not to replace intuitions, expts, heuristics

Provides structure and clarity

- Refines intuition
- Guides design
- Suggests ideas
- Explores boundaries
- Understands structural properties

Risk

- "All models are wrong"
- "… some are useful"
- Validate with simulations & experiments

Structural properties

Equilibrium properties

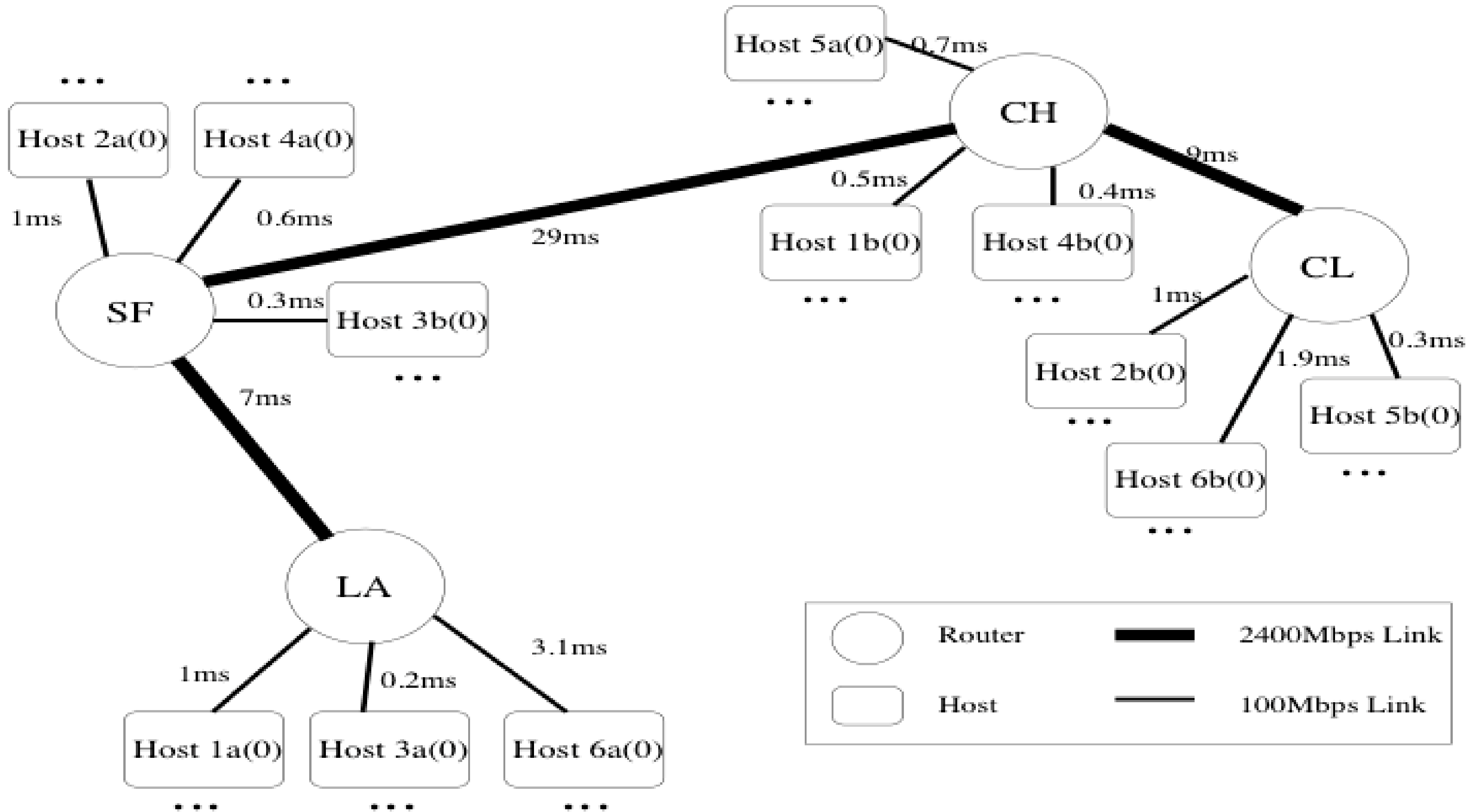
- Throughput, delay, loss, fairness

Dynamic properties

- Stability
- Robustness
- Responsiveness

Scalability properties

- Information scaling (decentralization)
- Computation scaling
- Performance scaling



L., Peterson, Wang, JACM 2002

Source Class	1a		2a		3a		4a		5a		6a	
	M	S	M	S	M	S	M	S	M	S	M	S
baseRTT (ms)	75.17	75.17	80.17	80.17	15.17	15.17	60.17	60.17	20.17	20.17	100.18	100.18
RTT w/ queueing (ms)	76.96	76.64	81.96	81.62	15.89	15.77	61.23	61	20.89	20.76	102.69	102.24
Sending rate (KB/s)	1382	1363	1382	1378	3618	3625	2236	2237	3618	3601	1000	968
Congestion window (pkts)	106.35	105.4	113.27	113.8	57.5	57.2	136.93	138.1	75.6	75.3	102.69	100.7

Queue (pkts)	LA		SF		CH	
	M	S	M	S	M	S
	166.0	160.7	268.5	259.2	166.0	159.6

Limitations of basic model

Static and deterministic network
- Fixed set of flows, link capacities, routing
- Real networks are time-varying and random

Homogeneous protocols
- All flows use the same congestion measure

Fluid approximation
- Ignore packet level effects, e.g. burstiness
- Inaccurate buffering process

basic model has been generalized
to address these issues to various degrees

Mathematical models

Why mathematical models?

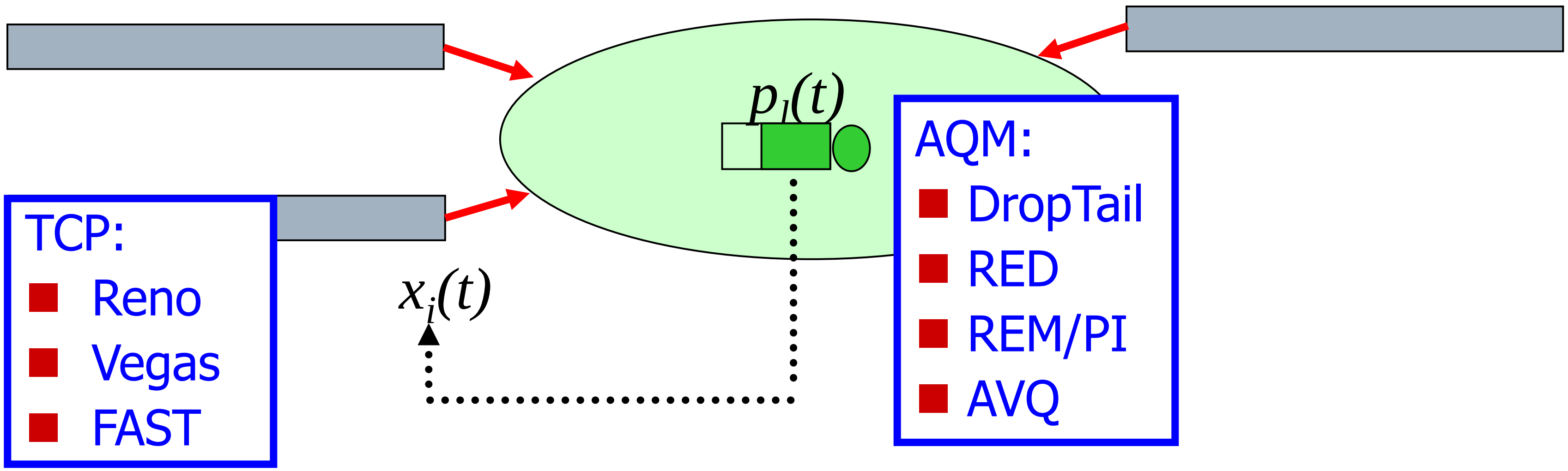
Dynamical systems model of CC

Convex optimization primer

Reverse engr: equilibrium properties

Forward engr: FAST TCP

TCP/AQM



Congestion control is a distributed asynchronous algorithm to share bandwidth

It has two components

- TCP: adapts sending rate (window) to congestion
- AQM: adjusts & feeds back congestion information

They form a distributed feedback control system

- Equilibrium & stability depends on both TCP and AQM
- And on delay, capacity, routing, #connections

Network model

Network

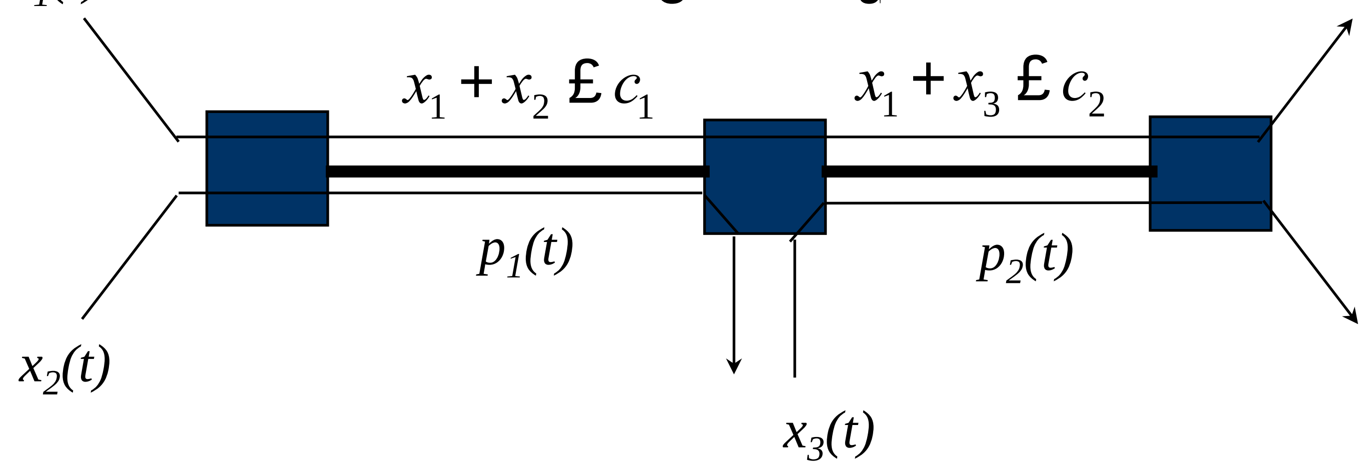
- Links l of capacities c_l and congestion measure $p_l(t)$

Sources i

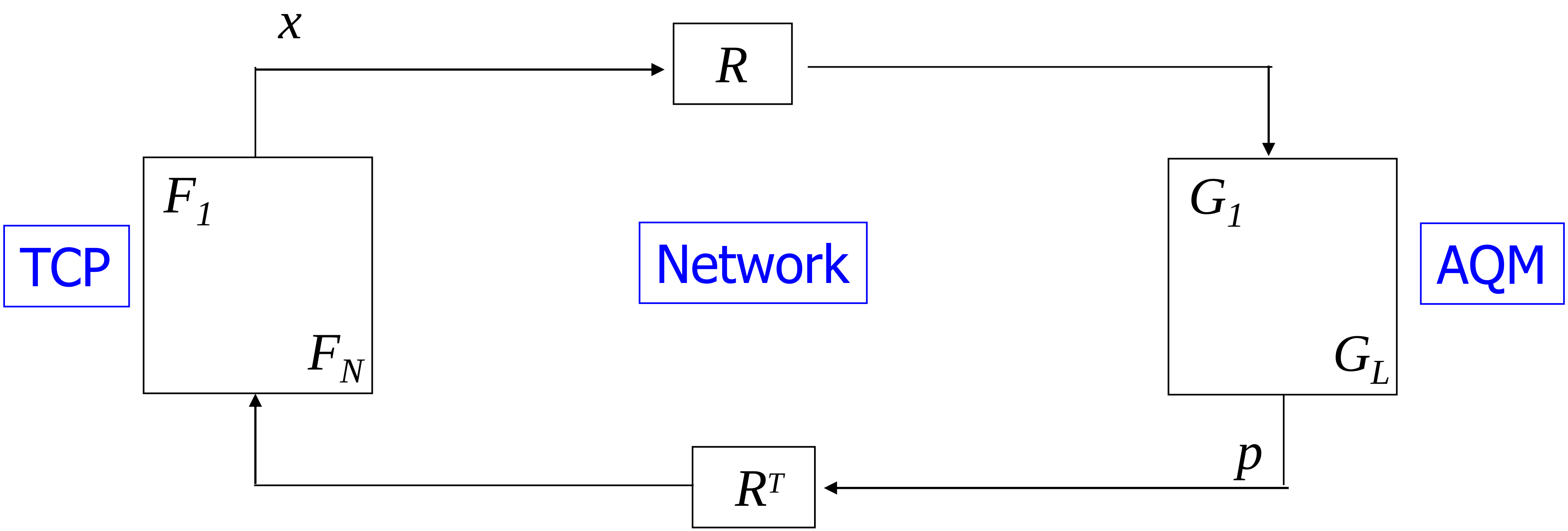
- Source rates $x_i(t)$

Routing matrix R

$$R = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$



Network model



TCP CC model consists of
specs for F_i and G_l

Examples

Derive (F_i, G_l) model for

- Reno/RED
- Vegas/Droptail
- FAST/Droptail

Focus on Congestion Avoidance

Model: Reno

```
for every ack (ca)
{   W += 1/W   }
for every loss
{   W := W/2   }
```

$$\Delta w_i(t) \quad =$$

Model: Reno

```
for every ack (ca)
{   W += 1/W   }
for every loss
{   W := W/2   }
```

$$\Delta w_i(t) = \frac{x_i(t)(1-q_i(t))}{w_i(t)}$$

throughput

window size

$$q_i(t) = \sum_l R_{li} p_l(t)$$

round-trip
loss probability

link loss
probability

Model: Reno

```
for every ack (ca)
{   W += 1/W    }
for every loss
{   W := W/2    }
```

$$\Delta w_i(t) = \frac{x_i(t)(1-q_i(t))}{w_i(t)} - \frac{w_i(t)}{2} x_i(t) q_i(t)$$

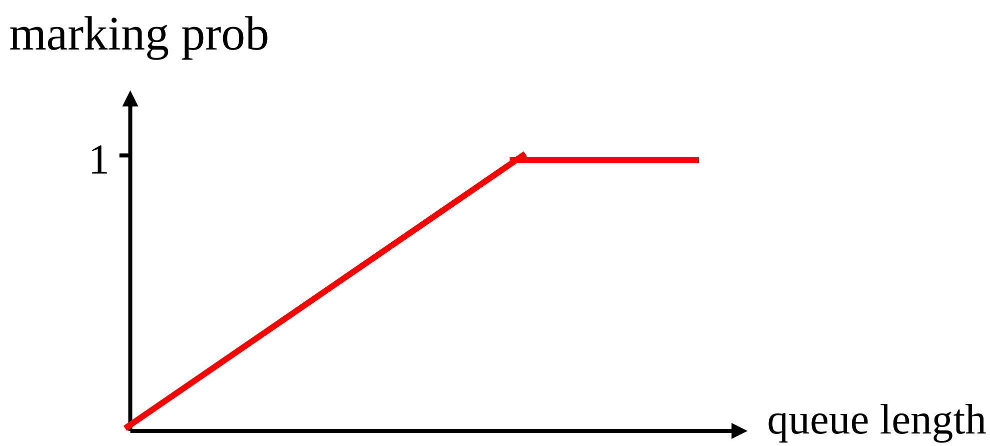
$$x_i(t+1) = \underbrace{x_i(t) + \frac{1}{T_i^2} - \frac{x_i^2}{2} q_i(t)}_{F_i\left(x_i(t), q_i(t)\right)}$$

Uses:

$$x_i(t) = \frac{w_i(t)}{T_i}$$

$$q_i(t) \approx 0$$

Model: RED



$$\boxed{y_l(t) = \sum_i R_{li} x_i(t)}$$

aggregate link rate

source rate

$$b_l(t+1) = \left[b_l(t) + y_l(t) - c_l \right]^+$$

$$\underbrace{p_l(t) = \min\left\{ \alpha b_l(t), 1 \right\}}_{p_l(t) = G_l\left(y_l(t), p_l(t)\right)}$$

Model: Reno/RED

$$\underbrace{x_i(t+1) = x_i(t) + \frac{1}{T_i^2} - \frac{x_i^2}{2} q_i(t)}_{x_i(t+1) = F_i\left(x_i(t), q_i(t)\right)}$$

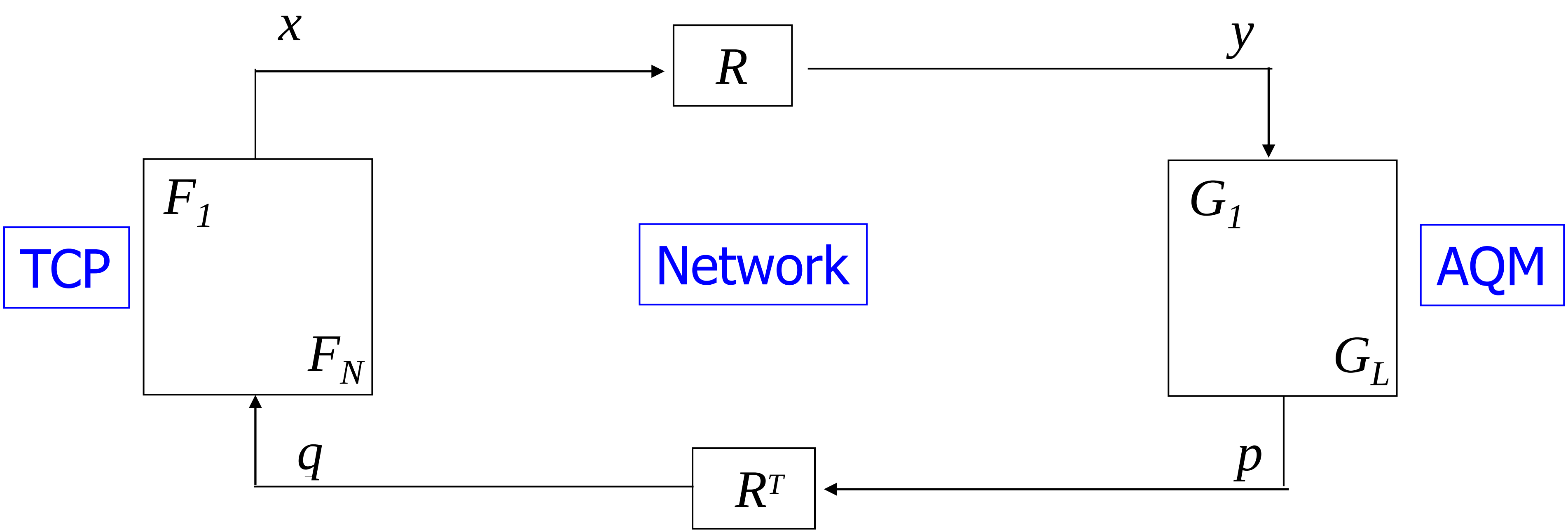
$$q_i(t) = \sum_l R_{li} p_l(t)$$

$$y_l(t) = \sum_i R_{li} x_i(t)$$

$$b_l(t+1) = \left[b_l(t) + y_l(t) - c_l \right]^+$$

$$\underbrace{p_l(t) = \max\{ \alpha b_l(t), 1 \}}_{p_l(t) = G_l\left(y_l(t), p_l(t)\right)}$$

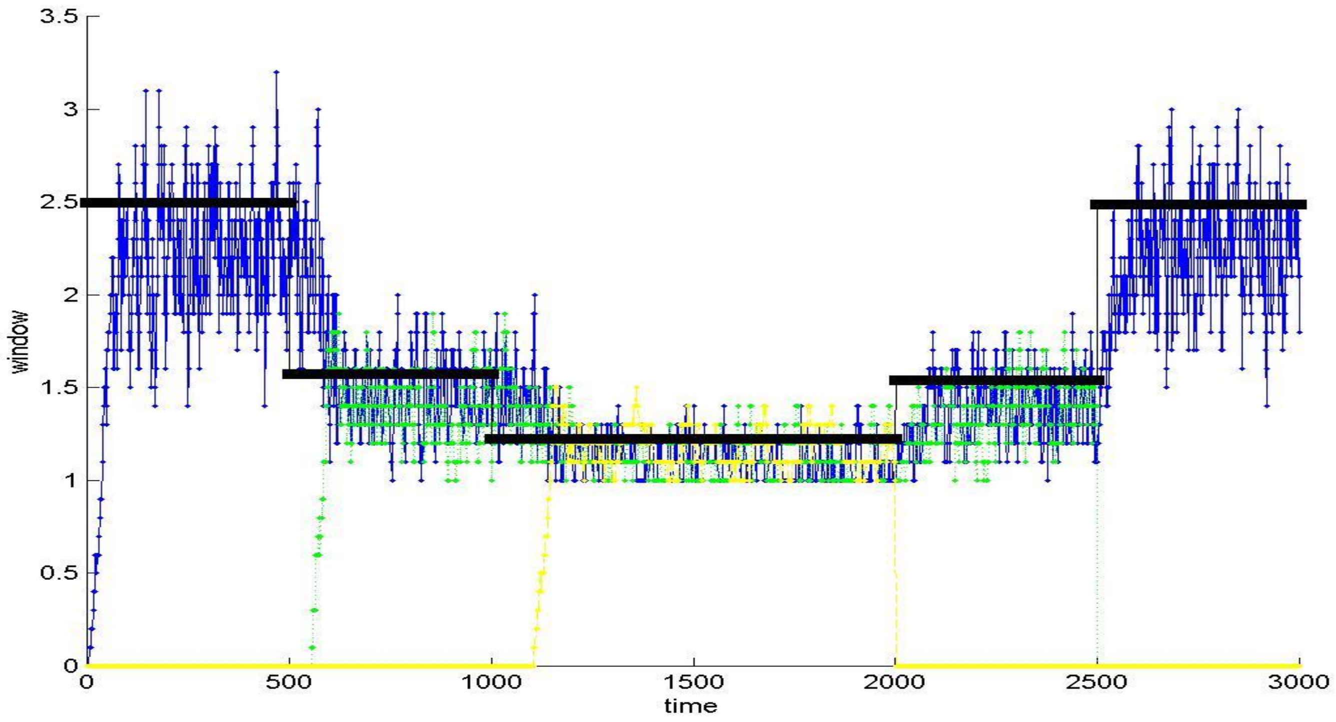
Decentralization structure



$$x(t+1) \;=\; F(x(t),\; q(t))$$
$$p(t+1) \;=\; G(y(t),\; p(t))$$

$$q_i(t) = \sum_l R_{li}\, p_l(t)$$
$$y_l(t) = \sum_i R_{li}\, x_i(t)$$

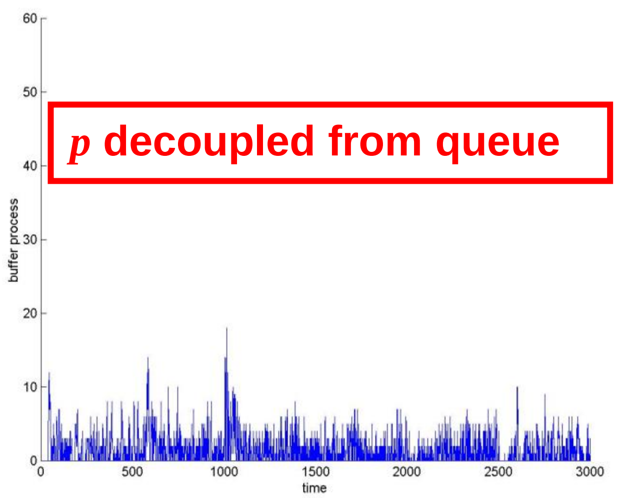
Validation – Reno/REM

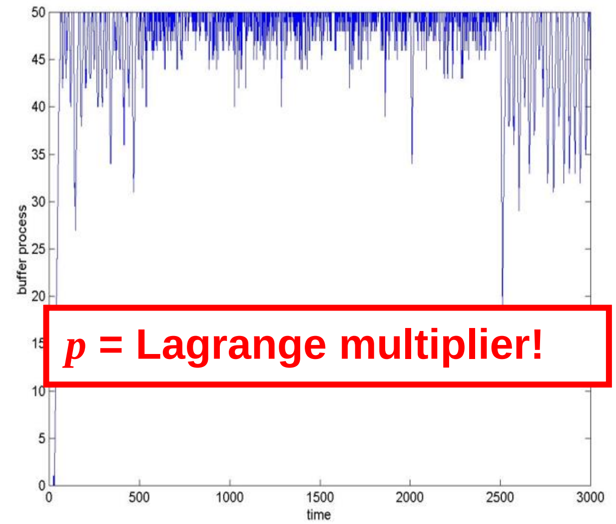


- 30 sources, 3 groups with RTT = 3, 5, 7 ms
- Link capacity = 64 Mbps, buffer = 50 kB
- Smaller window due to small RTT (~0 queueing delay)

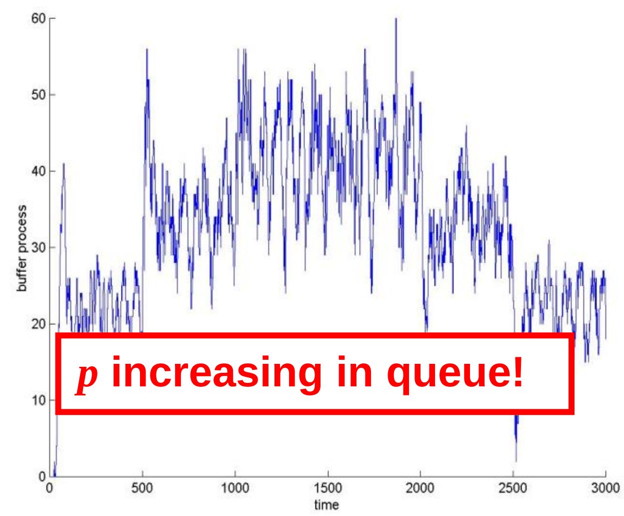
Queue

REM
queue = 1.5 pkts
utilization = 92%
$\gamma = 0.05$, $\alpha = 0.4$, $\phi = 1.15$





DropTail
queue = 94%



RED
min_th = 10 pkts
max_th = 40 pkts
max_p = 0.1

Model: Vegas/Droptail

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

queue size

F_i:

$$x_i(t+1) = \begin{cases} x_i(t) + \dfrac{1}{T_i^2(t)} & \text{if } \; w_i(t) - d_i x_i(t) < \alpha_i d_i \\[2ex] x_i(t) - \dfrac{1}{T_i^2(t)} & \text{if } \; w_i(t) - d_i x_i(t) > \alpha_i d_i \\[2ex] x_i(t) & \text{else} \end{cases}$$

G_l:

$$p_l(t+1) = [p_l(t) + y_l(t)/c_l - 1]^+$$

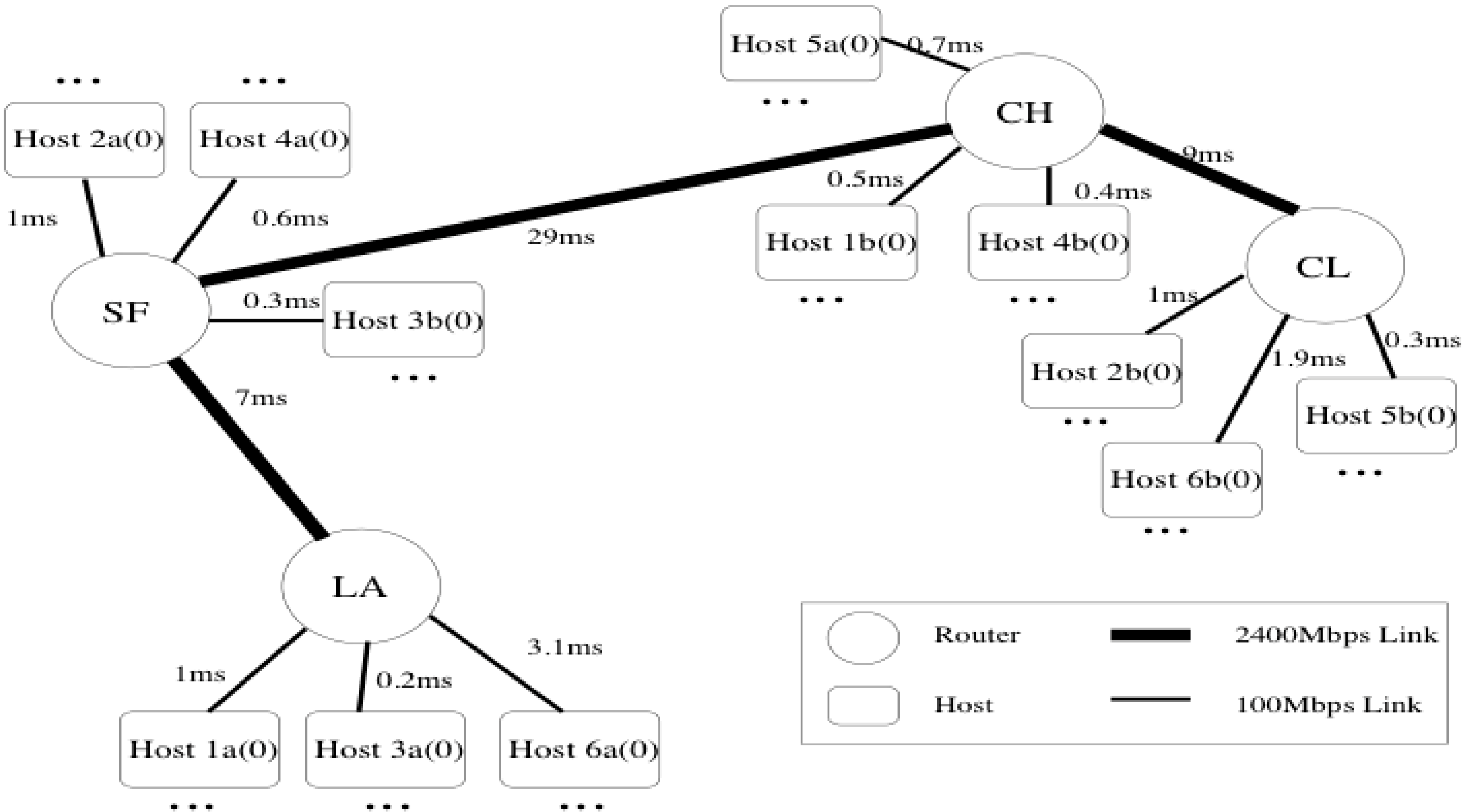
$$\boxed{T_i(t) = d_i + q_i(t)}$$

Model: FAST/Droptail

periodically
{
$$W := \frac{\text{baseRTT}}{\text{RTT}} W + \alpha$$
}

$$x_i(t+1) = x_i(t) + \frac{\gamma_i}{T_i(t)}\left(\alpha_i - x_i(t) q_i(t)\right)$$

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



L., Peterson, Wang, JACM 2002

Source	*1a*		*2a*		*3a*		*4a*		*5a*		*6a*	
Class	M	S	M	S	M	S	M	S	M	S	M	S
baseRTT (ms)	75.17	75.17	80.17	80.17	15.17	15.17	60.17	60.17	20.17	20.17	100.18	100.18
RTT w/ queueing (ms)	76.96	76.64	81.96	81.62	15.89	15.77	61.23	61	20.89	20.76	102.69	102.24
Sending rate (KB/s)	1382	1363	1382	1378	3618	3625	2236	2237	3618	3601	1000	968
Congestion window (pkts)	106.35	105.4	113.27	113.8	57.5	57.2	136.93	138.1	75.6	75.3	102.69	100.7

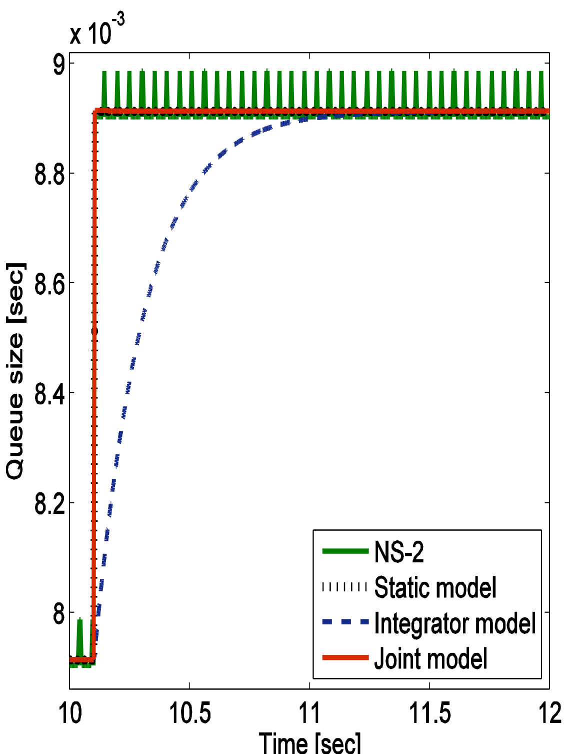
Queue (pkts)	LA		SF		CH	
	M	S	M	S	M	S
	166.0	160.7	268.5	259.2	166.0	159.6

Validation: matching transients

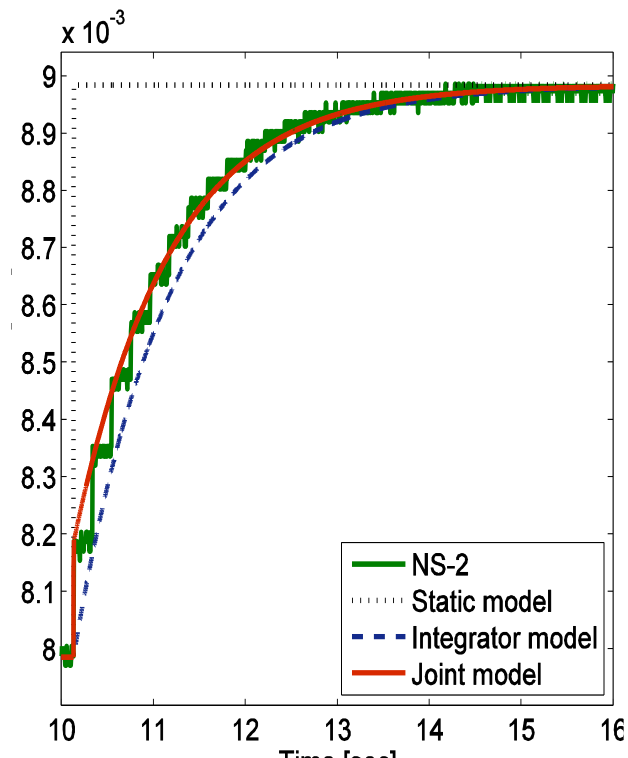
$$\dot{p} = \frac{1}{c}\left[\left(\sum_i \frac{w_i(t-\tau_i^f)}{d_i + p(t)} + \dot{w}_i(t-\tau_i^f)\right) + x_0(t) - c\right]$$

[Jacobsson et al 2009]

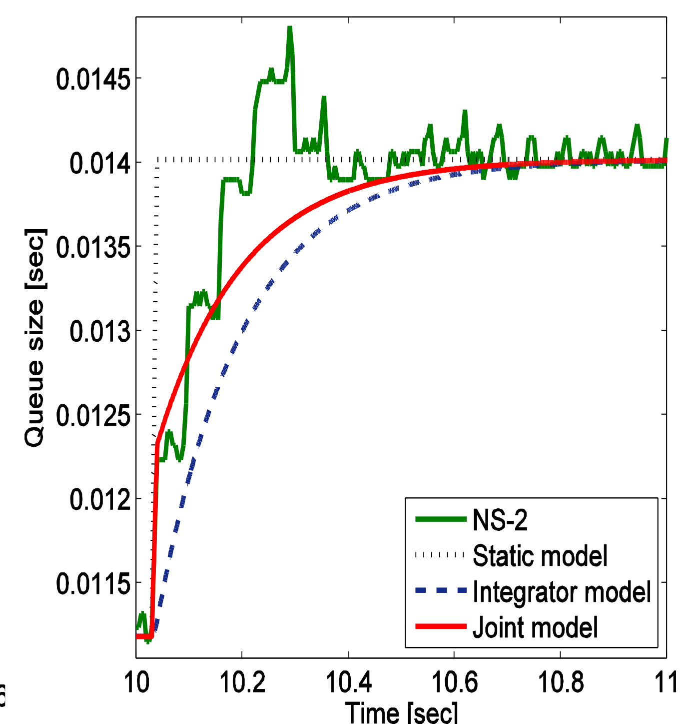
Same RTT, no cross traffic



Same RTT, cross traffic



Different RTTs, no cross traffic



Recap

Protocol (Reno, Vegas, FAST, Droptail, RED…)

$$x(t+1) = F\ (x(t),\ q(t))$$
$$p(t+1) = G\ (y(t),\ p(t))$$

Equilibrium
- Performance
  - Throughput, loss, delay
- Fairness
- Utility

Dynamics
- Local stability
- Global stability

Mathematical models

Why mathematical models?

Dynamical systems model of CC

Convex optimization primer

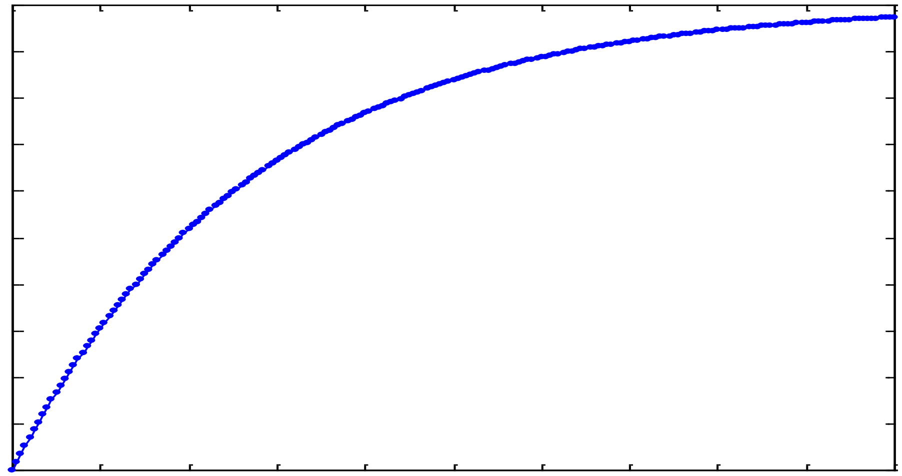
Reverse engr: equilibrium properties

Forward engr: FAST TCP

Background: optimization

$$\max_{x \geq 0} \sum U_i(x_i) \qquad \text{subject to} \quad Rx \leq c$$

Called convex program if U_i are concave functions



Background: optimization

$$\max_{x \geq 0} \sum U_i(x_i) \qquad \text{subject to} \quad Rx \leq c$$

Called convex program if U_i are concave functions

Local optimum is globally optimal

- First order optimality (KKT) condition is necessary <u>and</u> sufficient

Convex programs are polynomial-time solvable

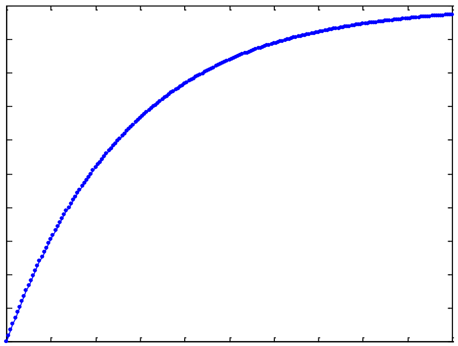
- Whereas nonconvex programs are generally NP hard

Background: optimization

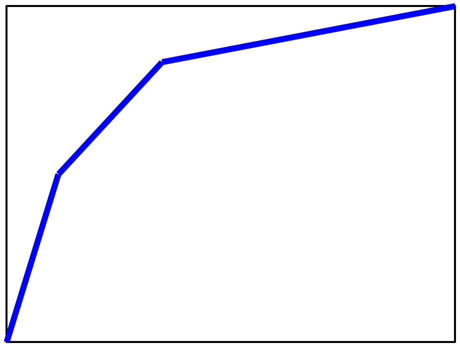
$$\max_{x \geq 0} \sum U_i(x_i) \qquad \text{subject to} \quad Rx \leq c$$

Theorem

- Optimal solution x^* exists
- It is unique if U_i are strictly concave



strictly concave



not

Background: optimization

$$\max_{x \geq 0} \sum U_i(x_i) \qquad \text{subject to} \quad Rx \leq c$$

Theorem

x^* is optimal if and only if there exists $p^* \geq 0$ such that

$$U_i'\left(x_i^*\right) = q_i^* := \sum_l R_{li} p_l^*$$

Lagrange multiplier

$$y_l^* := \sum_i R_{li} x_i^* \begin{cases} \leq c_l \\ = c_l & \text{if} \quad p_l^* > 0 \end{cases}$$

Complementary slackness: all bottlenecks are fully utilized

Background: optimization

$$\max_{x \geq 0} \sum U_i(x_i) \quad \text{subject to} \quad Rx \leq c$$

<u>Theorem</u>

p^* can be interpreted as prices

- Optimal x_i^* maximizes its own benefit

$$\max_{x_i} \; U_i(x_i) - x_i \sum_l R_{li} p_l^*$$

incentive compatible

Background: optimization

$$\max_{x \ge 0} \sum U_i(x_i) \qquad \text{subject to} \quad Rx \le c$$

Theorem

Gradient decent algorithm to solve the dual problem is decentralized

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

law of supply & demand

$$x_i(t) = U_i'^{-1}\left(q_i(t) \right)$$

$$q_i(t) = \sum_l R_{li} p_l(t)$$

$$y_l(t) = \sum_i R_{li} x_i(t)$$

Background: optimization

$$\max_{x \geq 0} \sum U_i(x_i) \qquad \text{subject to} \quad Rx \leq c$$

Theorem

Gradient decent algorithm to solve the dual problem is decentralized

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

$$x_i(t) = U_i'^{-1}\left(q_i(t) \right)$$

Gradient-like algorithm to solve NUM defines TCP CC algorithm !

➔ reverse/forward engineer TCP

Mathematical models

Why mathematical models?

Dynamical systems model of CC

Convex optimization primer

Reverse engr: equilibrium properties

Forward engr: FAST TCP

Duality model of TCP/AQM

TCP/AQM
$$x^* = F(x^*, R^T p^*)$$
$$p^* = G(Rx^*, p^*)$$

Equilibrium (x^*, p^*) primal-dual optimal:

$$\max_{x \geq 0} \sum U_i(x_i) \quad \text{subject to} \quad Rx \leq c$$

- F determines utility function U
- G guarantees complementary slackness
- p^* are Lagrange multipliers

Kelly, Maloo, Tan 1998
Low, Lapsley 1999

Uniqueness of equilibrium

- x^* is unique when U is strictly concave
- p^* is unique when R has full row rank

Duality model of TCP/AQM

TCP/AQM

$$x^* = F(x^*, R^T p^*)$$

$$p^* = G(Rx^*, p^*)$$

Equilibrium (x^*, p^*) primal-dual optimal:

$$\max_{x \geq 0} \sum U_i(x_i) \quad \text{subject to} \quad Rx \leq c$$

- F determines utility function U
- G guarantees complementary slackness
- p^* are Lagrange multipliers

Kelly, Maloo, Tan 1998
Low, Lapsley 1999

The underlying convex program also
leads to simple dynamic behavior

Duality model of TCP/AQM

Equilibrium (x^*, p^*) primal-dual optimal:

$$\max_{x \geq 0} \sum U_i(x_i) \qquad \text{subject to} \quad Rx \leq c$$

Mo & Walrand 2000:

$$U_i(x_i) = \begin{cases} \log x_i & \text{if } \alpha = 1 \\ (1-\alpha)^{-1} x_i^{1-\alpha} & \text{if } \alpha \neq 1 \end{cases}$$

- $\alpha = 1$: Vegas, FAST, STCP
- $\alpha = 1.2$: HSTCP
- $\alpha = 2$: Reno
- $\alpha = \infty$: XCP (single link only)

Low 2003

Duality model of TCP/AQM

Equilibrium (x^*, p^*) primal-dual optimal:

$$\max_{x\geq 0} \sum U_i(x_i) \qquad \text{subject to} \quad Rx \leq c$$

Mo & Walrand 2000:

$$U_i(x_i) = \begin{cases} \log x_i & \text{if } \alpha = 1 \\ (1-\alpha)^{-1} x_i^{1-\alpha} & \text{if } \alpha \neq 1 \end{cases}$$

- $\alpha = 0$: maximum throughput
- $\alpha = 1$: proportional fairness
- $\alpha = 2$: min delay fairness
- $\alpha = \infty$: maxmin fairness

Low 2003

Some implications

Equilibrium

- Always exists, unique if R is full rank
- Bandwidth allocation independent of AQM or arrival
- Can predict macroscopic behavior of large scale networks

Counter-intuitive throughput behavior

- Fair allocation is not always inefficient
- Increasing link capacities do not always raise aggregate throughput

[Tang, Wang, Low, ToN 2006]

Forward engineering: FAST TCP

- Design, analysis, experiments

[Wei, Jin, Low, Hegde, ToN 2006]

Equilibrium throughput

$$\text{Reno} \qquad x_i = \frac{1}{T_i} \times \frac{\alpha}{q_i^{0.5}}$$

$$\text{HSTCP} \qquad x_i = \frac{1}{T_i} \times \frac{\alpha}{q_i^{0.84}}$$

$$\text{Vegas, FAST} \qquad x_i = \frac{\alpha}{q_i}$$

α = 1.225 (Reno), 0.120 (HSTCP)

- Reno penalizes long flows
- Reno's square-root-p throughput formula
- Vegas, FAST: equilibrium cond = Little's Law

Vegas/FAST: effect of RTT error

Persistent congestion can arise due to

- Error in propagation delay estimation

Consequences

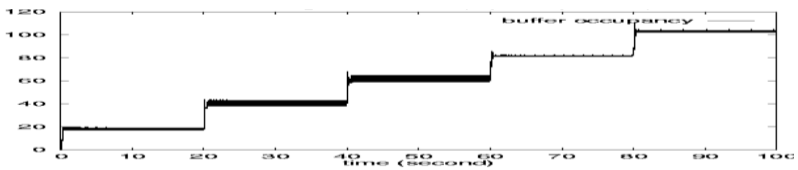
- Excessive backlog
- Unfairness to older sources

Theorem

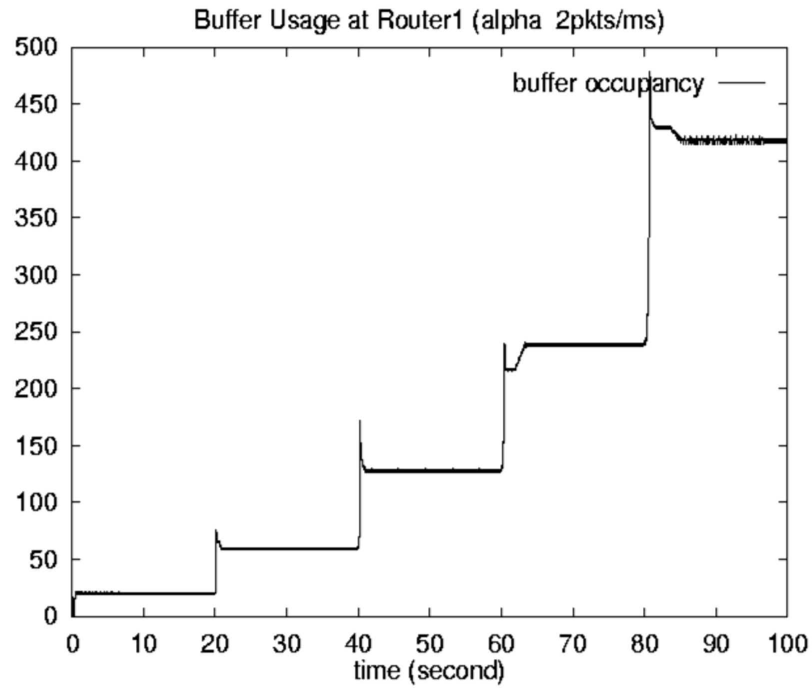
A relative error of ε_s *in propagation delay estimation distorts the utility function to*

$$\hat{U}_s(x_s) = (1+e_s)a_s \log x_s + e_s x_s$$

Evalidation



Without estimation error



With estimation error

- Single link, capacity = 6 pkt/ms, α_s = 2 pkts/ms, d_s = 10 ms
- With finite buffer: Vegas reverts to Reno

Validation

Source rates (pkts/ms)

#	src1	src2	src3	src4	src5
1	5.98 (6)				
2	2.05 (2)	3.92 (4)			
3	0.96 (0.94)	1.46 (1.49)	3.54 (3.57)		
4	0.51 (0.50)	0.72 (0.73)	1.34 (1.35)	3.38 (3.39)	
5	0.29 (0.29)	0.40 (0.40)	0.68 (0.67)	1.30 (1.30)	3.28 (3.34)

#	queue (pkts)	baseRTT (ms)
1	19.8 (20)	10.18 (10.18)
2	59.0 (60)	13.36 (13.51)
3	127.3 (127)	20.17 (20.28)
4	237.5 (238)	31.50 (31.50)
5	416.3 (416)	49.86 (49.80)

Mathematical models

Why mathematical models?

Dynamical systems model of CC

Convex optimization primer

Reverse engr: equilibrium properties

Forward engr: FAST TCP

Reno design

Packet level

```
ACK:   W  ←  W + 1/W
Loss:  W  ←  W – 0.5W
```

Flow level

- Equilibrium $$x_i T_i = \frac{\sqrt{1.5}}{\sqrt{q_i}}$$ (Mathis formula 1996)

- Dynamics $$\dot{w}_i(t) = \frac{1}{T_i}\left(1 - \frac{2}{3} \cdot w_i^2(t) q_i(t)\right)$$

Reno design

Packet level

- Designed and implemented first

Flow level

- Understood afterwards

Flow level dynamics determines

- Equilibrium: performance, fairness
- Stability

Design flow level equilibrium & stability

Implement flow level goals at packet level

Forward engineering

1. Decide congestion measure
   - Loss, delay, both
2. Design flow level equilibrium properties
   - Throughput, loss, delay, fairness
3. Analyze stability and other dynamic properties
   - Control theory, simulate, improve model/algorithm
4. Iterate 1 – 3 until satisfactory
5. Simulate, prototype, experiment
   - Compare with theoretical predictions
   - Improve model, algorithm, code

Iterate 1 – 5 until satisfactory

Forward engineering

Tight integration of theory, design, experiment

Performance analysis done at design time

- Not after

Theory does not replace intuitions and heuristics

- Refines, validates/invalidates them

Theory provides structure and clarity

- Guides design
- Suggests ideas and experiments
- Explores boundaries that are hard to expt

Packet level description

- **Reno**
  AIMD(1, 0.5)

```
ACK:  W  ←  W + 1/W
Loss: W  ←  W – 0.5W
```

- **HSTCP**
  AIMD(a(w), b(w))

```
ACK:  W  ←  W + a(w)/W
Loss: W  ←  W – b(w)W
```

- **STCP**
  MIMD(a, b)

```
ACK:  W  ←  W + 0.01
Loss: W  ←  W – 0.125W
```

- **FAST**

$$\mathrm{RTT}:\ \mathrm{W} \leftarrow \mathrm{W} \cdot \frac{\mathrm{baseRTT}}{\mathrm{RTT}} + \alpha$$

Flow level: Reno, HSTCP, STCP, FAST

Common flow level dynamics!

$$\dot{w}_i(t) = \kappa(t) \cdot \left(1 - \frac{q_i(t)}{U_i'(t)} \right)$$

window adjustment = $\boxed{\text{control gain}}$ $\boxed{\text{flow level goal}}$

Different gain κ and utility U_i

- They determine equilibrium and stability

Different congestion measure q_i

- Loss probability (Reno, HSTCP, STCP)
- Queueing delay (Vegas, FAST)

Flow level: Reno, HSTCP, STCP, FAST

Common flow level dynamics!

$$\dot{w}_i(t) = \kappa(t) \cdot \left(1 - \frac{q_i(t)}{U_i'(t)}\right)$$

window adjustment = $\boxed{\text{control gain}}$ $\boxed{\text{flow level goal}}$

Small adjustment when close, large far away

- Need to estimate how far current state is wrt target
- Scalable

Reno, Vegas: window adjustment independent of q_i

- Depends only on current window
- Difficult to scale

NetLab

prof steven low

Lee Center

rsrg SISL

Caltech FAST Project

2000 2001 2002 2003 2004 2005 2006 2007

Lee Center

FAST TCP theory

IPAM Wkp

SC02 Demo

WAN-in-Lab Testbed

control & optimization of networks

theory ↔ experiment

testbed ↔ deployment

Collaborators: Doyle (Caltech), Newman (Caltech), Paganini (Uruguay), Tang (Cornell), Andrew (Swinburne), Chiang (Princeton); CACR, CERN, Internet2, SLAC, Fermi Lab, StarLight, Cisco

theory

Internet: largest distributed nonlinear feedback control system

Reverse engineering: TCP is real-time distributed algorithm over Internet to maximize utility

$$\max_{x\geq 0} \sum U_i(x_i) \quad \text{s. t.} \quad Rx \leq c$$

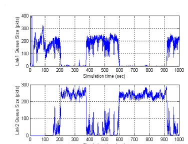
Forward engineering: Invention of FastTCP based on control theory & convex optimization

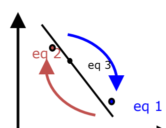
$$\dot{x}_i = \frac{\gamma_i}{T_i}\left(\alpha_i - x_i(t)\sum_l R_{li}\, p_l(t)\right)$$

$$\dot{p}_l = \frac{1}{c_l}\left(\sum_i R_{li}\, x_i(t) - c_l\right)$$

testbed

WAN-in-Lab : one-of-a-kind wind-tunnel in academic networking, with 2,400km of fiber, optical switches, routers, servers, accelerators





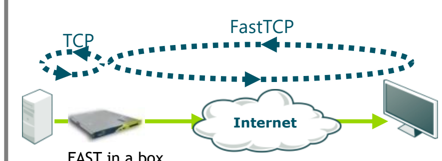
experiment

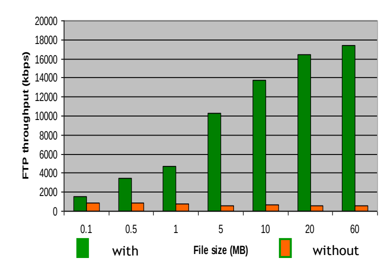
Scientists have used FastTCP to break world records on data transfer between 2002 - 2006

Internet2 LSR
SuperComputing BC

deployment

FAST is commercialized by FastSoft; it accelerates world's 2nd largest CDN and Fortune 100 companies





Some benefits

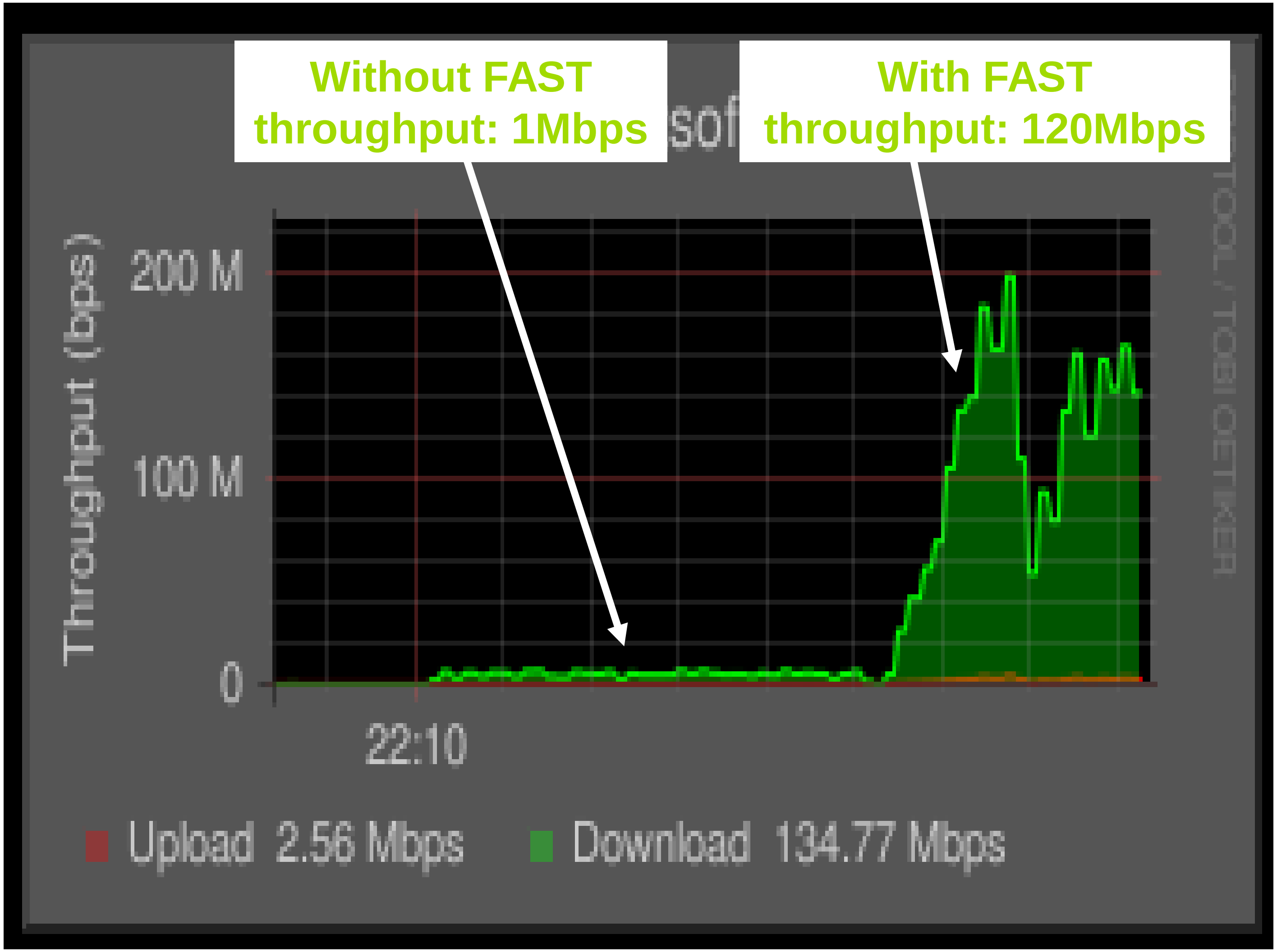
Transparent interaction among components

- TCP, AQM
- Clear understanding of structural properties

Understanding effect of parameters

- Change protocol parameters, topology, routing, link capacity, set of flows
- Re-solve NUM
- Systematic way to tune parameters

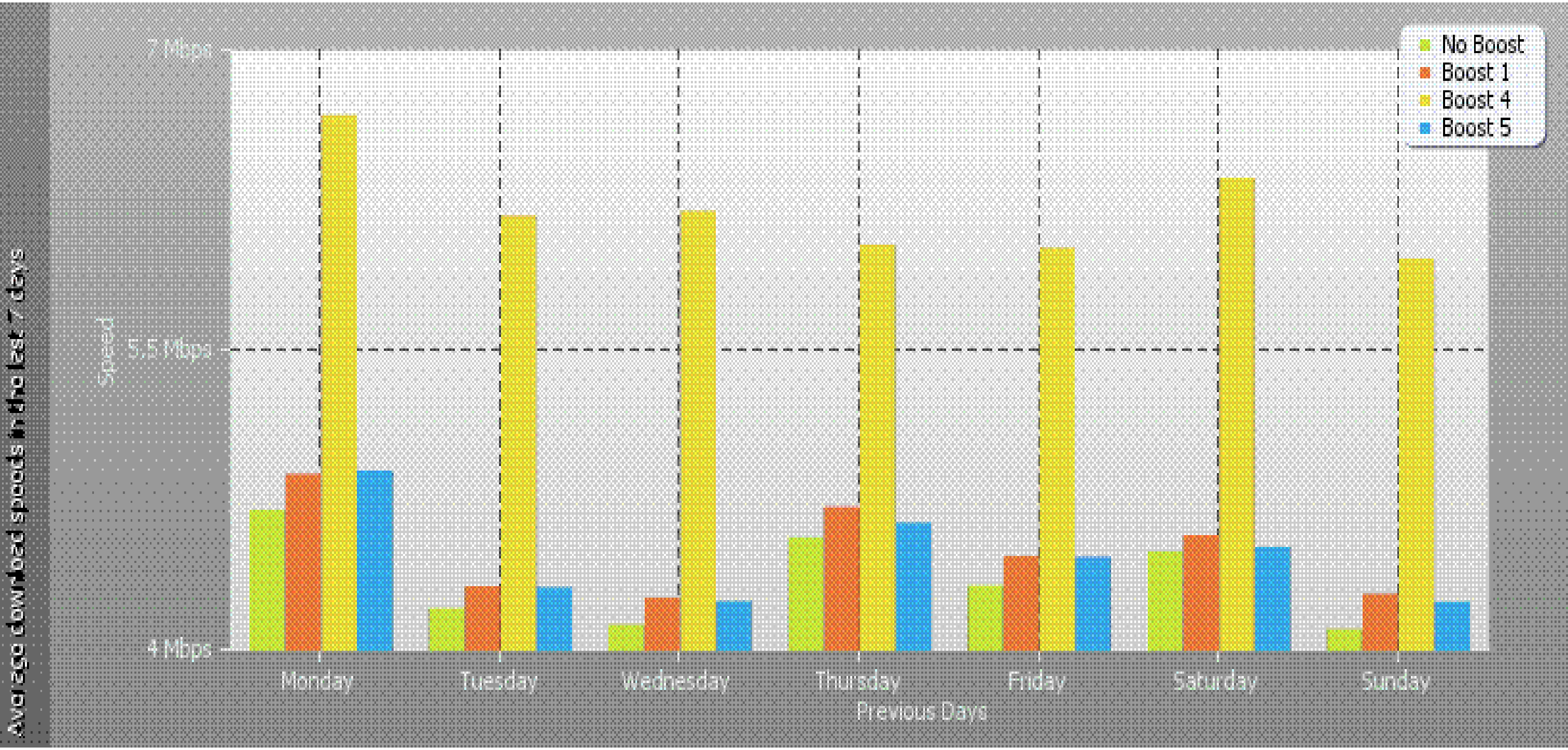
Extreme resilience to loss



Heavy packet loss in Sprint network:
FAST TCP increased throughput by 120x !

SF → New York
June 3, 2007

10G appliance customer data



Average download speed 8/24 – 30, 2009, CDN customer (10G appliance)
FAST vs TCP stacks in BSD, Windows, Linux

Summary: math models

Integration of theory, design, experiment can be very powerful

- Each needs the other
- Combination much more than sum

Theory-guided design approach

- Tremendous progress in the last decade; not as impossible as most feared
- Very difficult; but worth the effort
- Most critical: mindset

How to push theory-guided design approach further ?

Agenda

9:00	Congestion control protocols
10:00	break
10:15	Mathematical models
11:15	break
11:30	Advanced topics
12:30	lunch

ADVANCED TOPICS

Advanced topics

Heterogeneous protocols

Layering as optimization decomposition

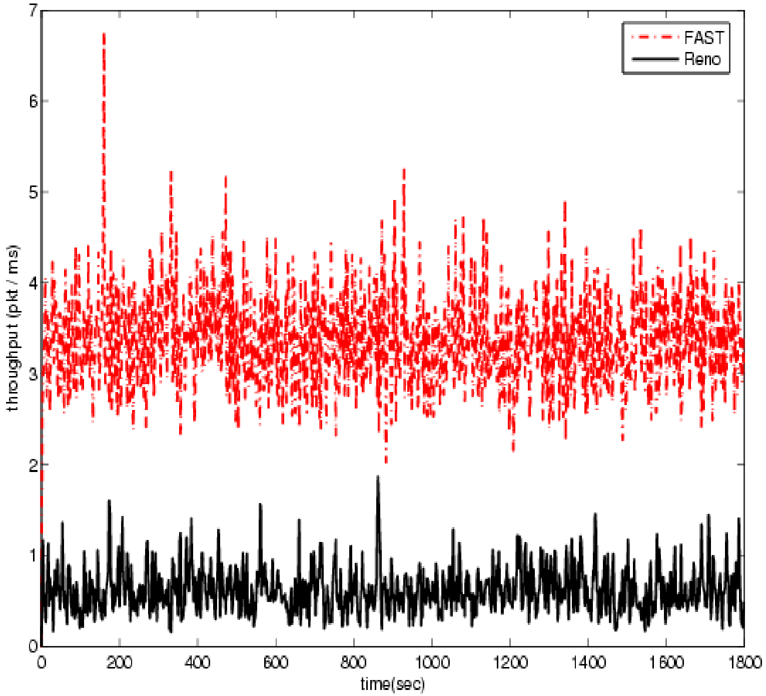
The world is heterogeneous…

- Linux 2.6.13 allows users to choose congestion control algorithms
- Many protocol proposals
  - Loss-based: Reno and a large number of variants
  - Delay-based: CARD (1989), DUAL (1992), Vegas (1995), FAST (2004), …
  - ECN: RED (1993), REM (2001), PI (2002), AVQ (2003), …
  - Explicit feedback: MaxNet (2002), XCP (2002), RCP (2005), …

Some implications

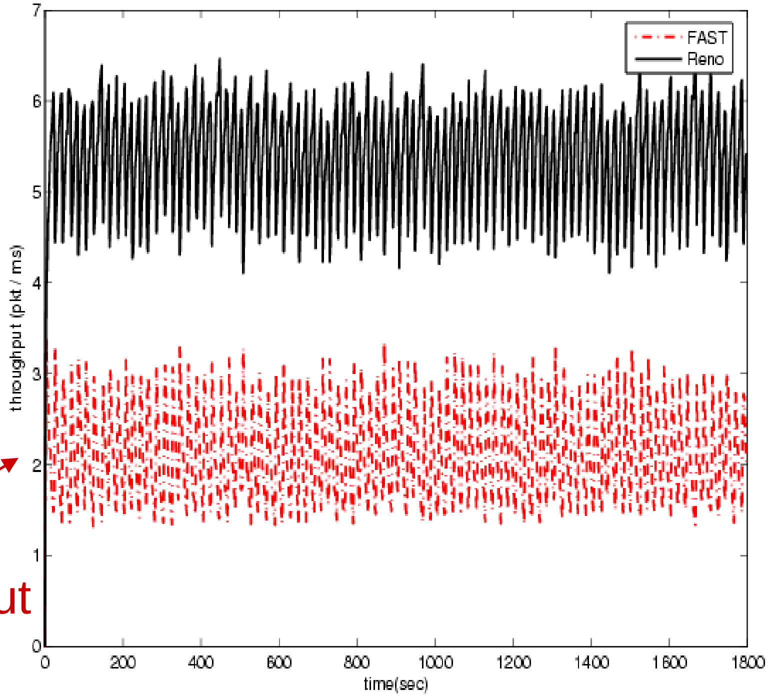
	homogeneous	**heterogeneous**
equilibrium	unique	?
bandwidth allocation on AQM	independent	?
bandwidth allocation on arrival	independent	?

Throughputs depend on AQM



buffer size = 80 pkts

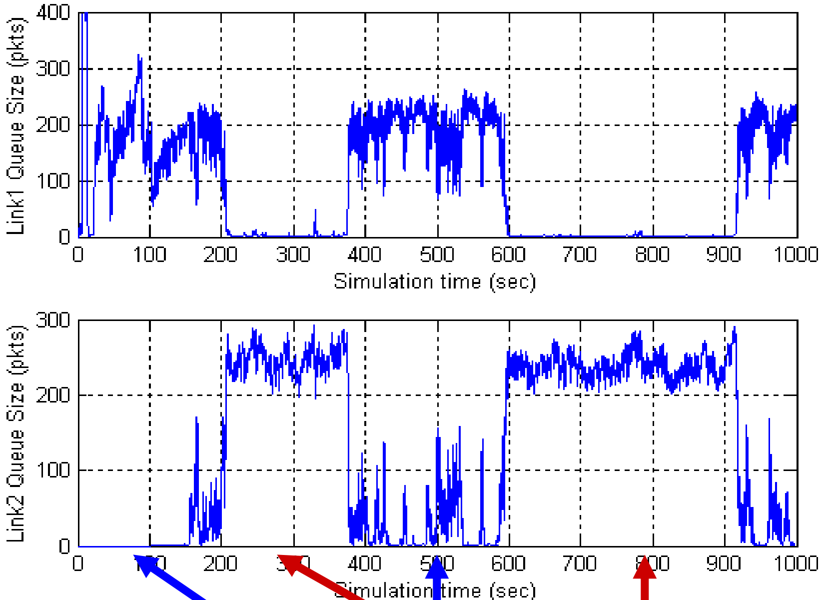
FAST throughput

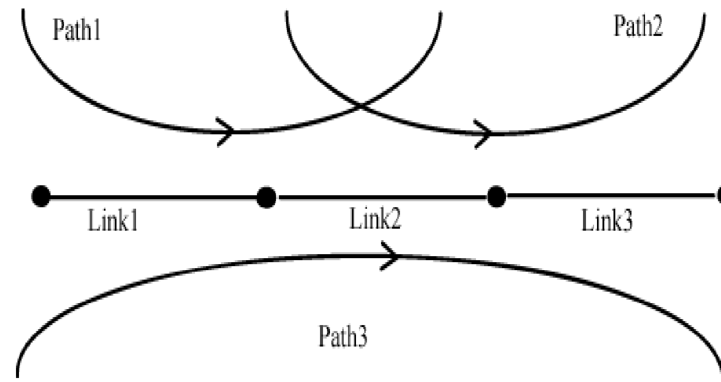


buffer size = 400 pkts

- FAST and Reno share a single bottleneck router
- NS2 simulation
- Router: DropTail with variable buffer size
- With 10% heavy-tailed noise traffic

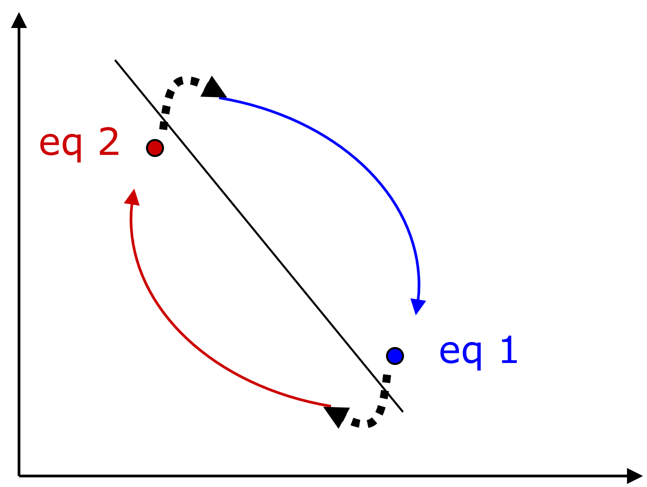
Multiple equilibria: throughput depends on arrival





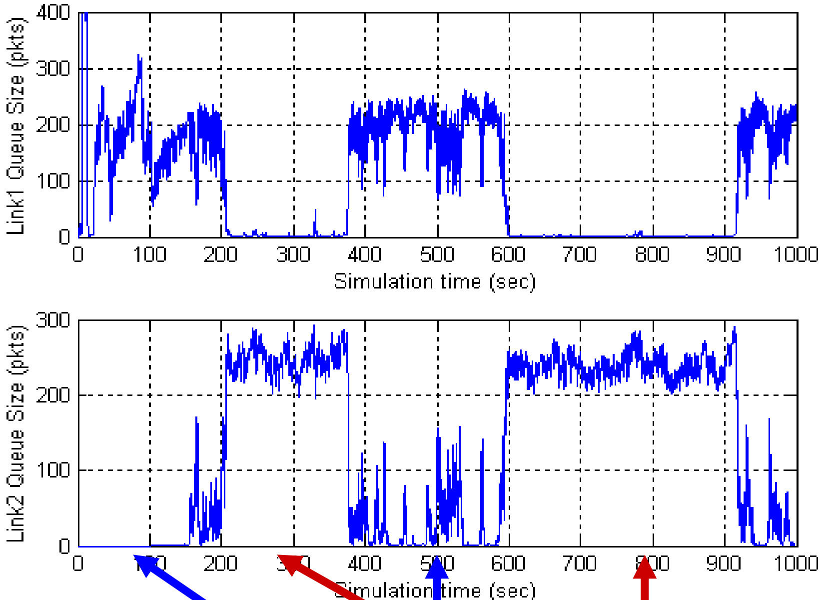
Dummynet experiment

	eq 1	eq 2
Path 1	52M	13M
path 2	61M	13M
path 3	27M	93M

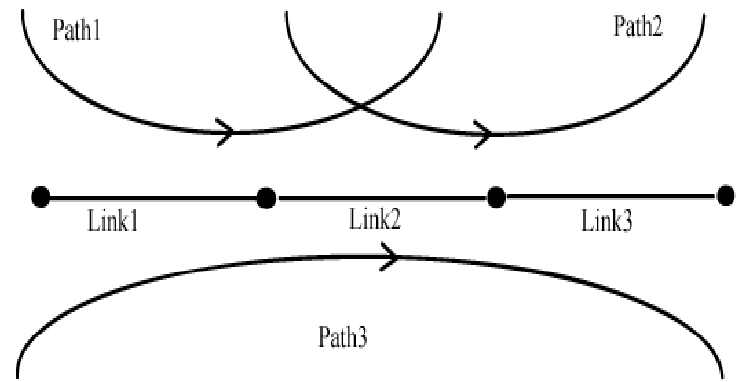


Tang, Wang, Hegde, Low, Telecom Systems, 2005

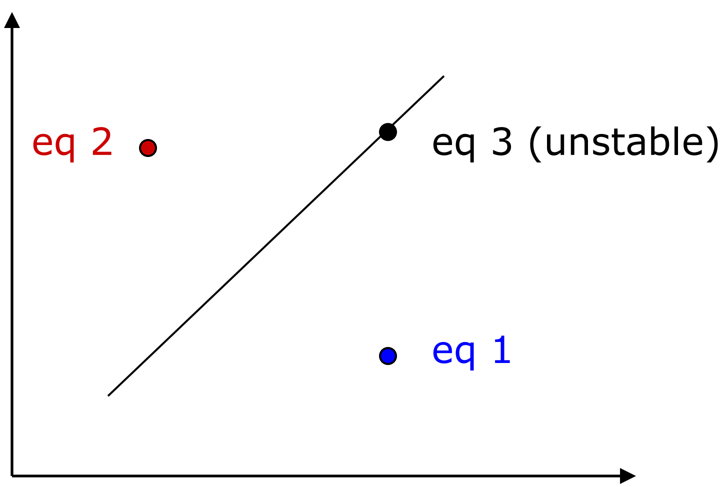
Multiple equilibria: throughput depends on arrival



	eq 1	eq 2
Path 1	52M	13M
path 2	61M	13M
path 3	27M	93M



Dummynet experiment



Tang, Wang, Hegde, Low, Telecom Systems, 2005

- Duality model:

$$\max_{x \geq 0} \sum U_i(x_i) \quad \text{s.t. } Rx \leq c \qquad x_i^* = F_i\left(\sum_l R_{li} p_l^*, x_i^*\right)$$

- Why can't use F_i's of FAST and Reno in duality model?

They use different prices!

$$F_i = x_i + \frac{\gamma_i}{T_i}\left(\alpha_i - x_i \sum_l R_{li} p_l\right)$$ delay for FAST

$$F_i = \frac{1}{T_i^2} - \frac{x_i^2}{2} \sum_l R_{li} p_l$$ loss for Reno

- Duality model:

$$\max_{x\geq 0} \sum U_i(x_i) \quad \text{s.t. } Rx \leq c \qquad\qquad x_i^* = F_i\left(\sum_l R_{li} p_l^*, x_i^*\right)$$

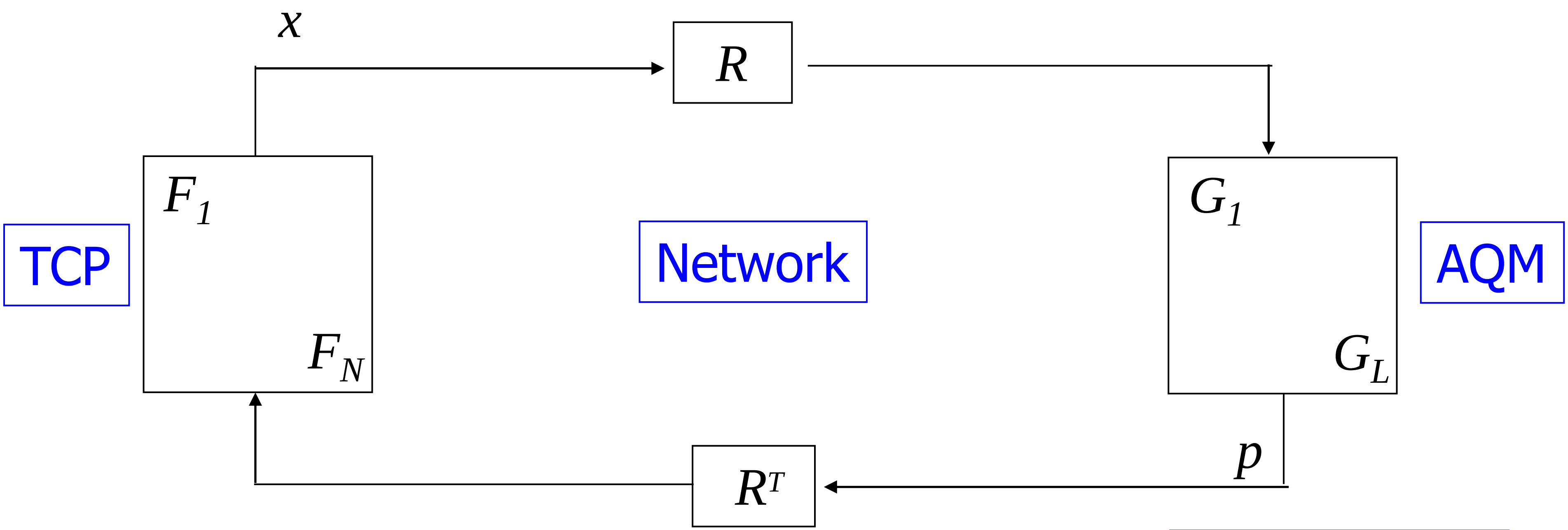
- Why can't use F_i's of FAST and Reno in duality model?

They use different prices!

$$F_i = x_i + \frac{\gamma_i}{T_i}\left(\alpha_i - x_i \sum_l R_{li} p_l\right) \qquad \dot{p}_l = \frac{1}{c_l}\left(\sum_i R_{li} x_i(t) - c_l\right)$$

$$F_i = \frac{1}{T_i^2} - \frac{x_i^2}{2}\sum_l R_{li} p_l \qquad \dot{p}_l = g_l\left(p_l(t), \sum_i R_{li} x_i(t)\right)$$

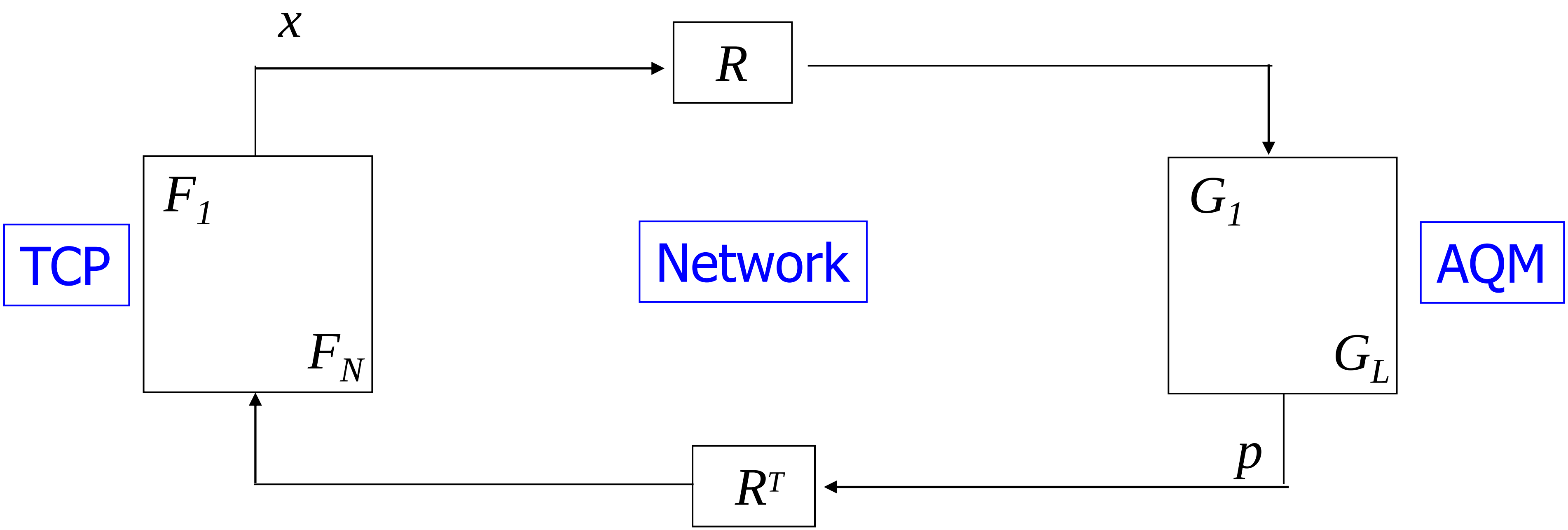
Homogeneous protocol



same price for all sources

$$x_i(t+1) \;=\; F_i\left(\sum_l R_{li}\, p_l(t),\; x_i(t)\right)$$

Heterogeneous protocol



$$x_i(t+1) \;=\; F_i\left(\sum_l R_{li}\, p_l(t),\; x_i(t)\right)$$

$$x_i^j(t+1) \;=\; F_i^j\left(\sum_l R_{li}\, m_l^j\big(p_l(t)\big),\; x_i^j(t)\right)$$

heterogeneous prices for type *j* sources

Heterogeneous protocols

- Equilibrium: p that satisfies

$$x_i^j(p) \;=\; f_i^{\,j}\left(\sum_l R_{li} m_l^j(p_l)\right)$$

$$y_l(p) := \sum_{i,j} R_{li}^j x_i^j(p) \begin{cases} \leq c_l & \\ = c_l & \text{if } \; p_l > 0 \end{cases}$$

Duality model no longer applies !

- p_l can no longer serve as Lagrange multiplier

Heterogeneous protocols

- Equilibrium: p that satisfies

$$x_i^j(p) = f_i^j\left(\sum_l R_{li} m_l^j(p_l)\right)$$

$$y_l(p) := \sum_{i,j} R_{li}^j x_i^j(p) \begin{cases} \leq c_l \\ = c_l & \text{if } \; p_l > 0 \end{cases}$$

Need to re-examine all issues

- Equilibrium: exists? unique? efficient? fair?
- Dynamics: stable? limit cycle? chaotic?
- Practical networks: typical behavior? design guidelines?

Notation

- Simpler notation: p is *equilibrium* if

  $y(p) \;=\; c$ on bottleneck links

- Jacobian: $\mathbf{J}(p) := \dfrac{\partial y}{\partial p}(p)$

- Linearized dual algorithm:

$$\partial \dot{p} = \gamma\, \mathbf{J}(p^*)\;\partial p(t)$$

Tang, Wang, L., Chiang, ToN, 2007
Tang, Wei, L., Chiang, ToN, 2010

Existence

Theorem

Equilibrium p exists, despite lack of underlying utility maximization

- Generally non-unique
  - There are networks with unique bottleneck set but infinitely many equilibria
  - There are networks with multiple bottleneck set each with a unique (but distinct) equilibrium

Regular networks

Definition

A *regular network* is a tuple (R, c, m, U) for which all equilibria p are locally unique, i.e.,

$$\det \mathbf{J}(p) := \det \frac{\partial y}{\partial p}(p) \neq 0$$

Theorem

- Almost all networks are regular
- A regular network has finitely many and odd number of equilibria (e.g. 1)

Global uniqueness

$$\dot{m}_l^j \in [a_l, 2^{1/L} a_l] \quad \text{for any } a_l > 0$$
$$\dot{m}_l^j \in [a^j, 2^{1/L} a^j] \quad \text{for any } a^j > 0$$

Theorem

- If *price heterogeneity* is small, then equilibrium is globally unique

Implication

a network of RED routers with slope inversely proportional to link capacity almost always has globally unique equilibrium

Local stability

$$\dot{m}_l^j \in [a_l, 2^{1/L} a_l] \quad \text{for any } a_l > 0$$
$$\dot{m}_l^j \in [a^j, 2^{1/L} a^j] \quad \text{for any } a^j > 0$$

Theorem

- If *price heterogeneity* is small, then the unique equilibrium p is locally stable
- If all equilibria p are locally stable, then it is globally unique

Linearized dual algorithm: $\delta\dot{p} = \gamma\, \mathbf{J}(p^*)\ \delta p(t)$

Equilibrium p is *locally stable* if

$$\operatorname{Re} \lambda\big(\mathbf{J}(p)\big) < 0$$

Summary

	homogeneous	**heterogeneous**
equilibrium	unique	non-unique
bandwidth allocation on AQM	independent	dependent
bandwidth allocation on arrival	independent	dependent

Interesting characterizations of equilibrium …
But not much understanding on dynamics

Efficiency

Result

- Every equilibrium p^* is Pareto efficient

Proof:

- Every equilibrium p^* yields a (unique) rate $x(p^*)$ that solves

$$\max_{x \geq 0} \sum_j \sum_i \lambda_i^j(p^*) U_i^j(x_i^j) \quad \text{s. t.} \quad Rx \leq c$$

Efficiency

Result

- Every equilibrium p^* is Pareto efficient

- Measure of optimality

$$V^* \;:=\; \max_{x \geq 0} \sum_j \sum_i U_i^j(x_i^j) \quad \text{s. t.} \quad Rx \leq c$$

- Achieved: $$V(p^*) \;:=\; \sum_j \sum_i U_i^j(x_i^j(p^*))$$

Efficiency

Result

- Every equilibrium p^* is Pareto efficient
- Loss of optimality:

$$\frac{V(p^*)}{V^*} \geq \frac{\min \ \dot{m}_l^j}{\max \ \dot{m}_l^j}$$

- Measure of optimality

$$V^* := \max_{x \geq 0} \sum_j \sum_i U_i^j(x_i^j) \quad \text{s.t.} \quad Rx \leq c$$

- Achieved: $\quad V(p^*) := \sum_j \sum_i U_i^j(x_i^j(p^*))$

Efficiency

Result

- Every equilibrium p^* is Pareto efficient
- Loss of optimality:

$$\frac{V(p^*)}{V^*} \geq \frac{\min \ \dot{m}_l^j}{\max \ \dot{m}_l^j}$$

e.g. A network of RED routers with default parameters suffers no loss of optimality

Intra-protocol fairness

Result

- Fairness among flows within each type is unaffected, i.e., still determined by their utility functions and Kelly's problem with reduced link capacities

Proof idea:

- Each equilibrium p chooses a partition of link capacities among types, $c^j := c^j(p)$
- Rates $x^j(p)$ then solve

$$\max_{x^j \geq 0} \sum_i U_i^j(x_i^j) \quad \text{s. t.} \quad R^j x^j \leq c^j$$

Inter-protocol fairness

Theorem

- Any fairness is achievable with a linear scaling of utility functions

$$\bar{x}^j \;:=\; \arg\max_{x^j \geq 0} \sum_i U_i^j(x_i^j) \quad \text{s.t.} \quad R^j x^j \leq c$$

$$\text{all achievable rates} \;\; X \;:=\; \left\{ x = \sum_j a^j \bar{x}^j \right\}$$

Slow timescale control

Slow timescale scaling of utility function

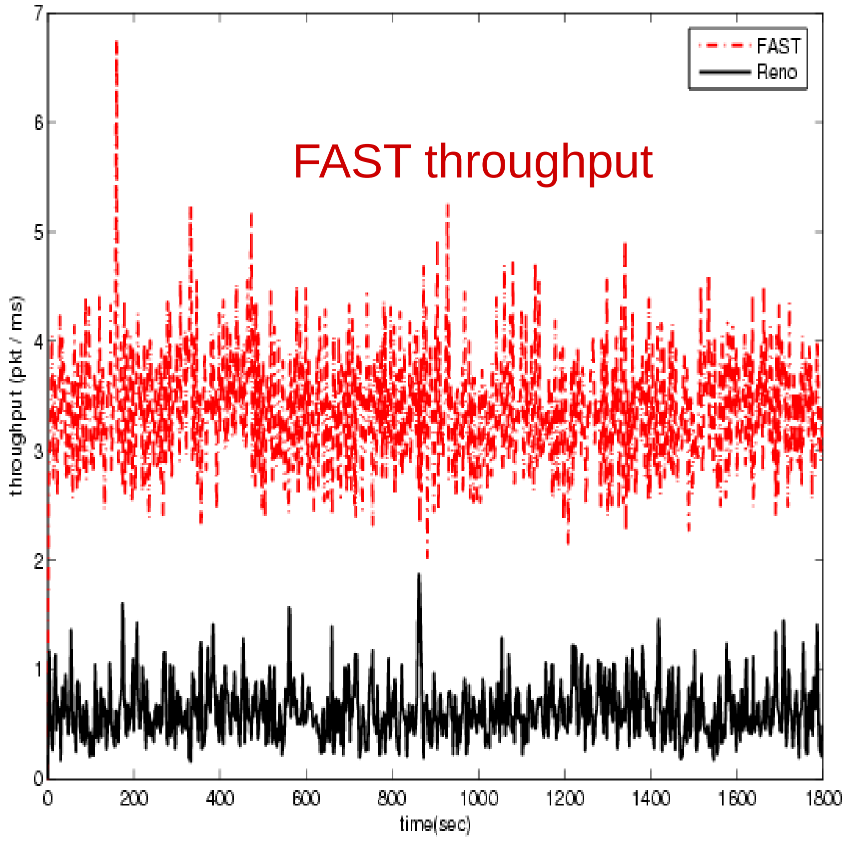
$$x_i^j(t) \;=\; f_i^j\left(\frac{q_i^j(t)}{\mu_i^j(t)}\right)$$

scaling of end--to-end price

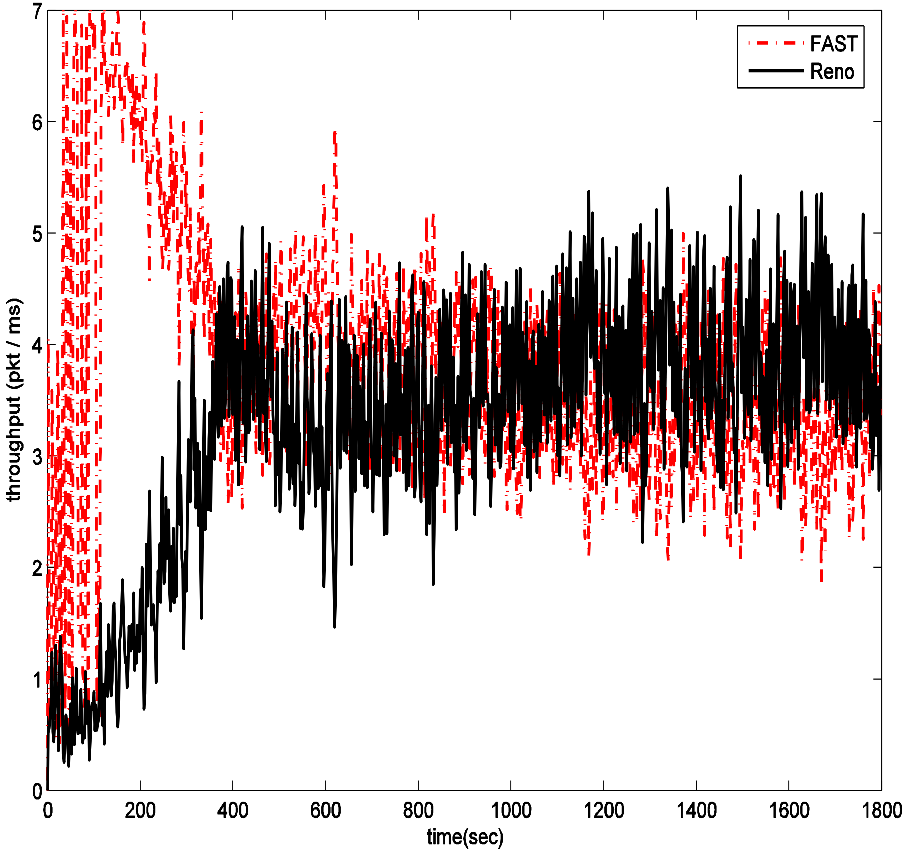
$$\mu_i^j(\mathrm{t}+1) \;=\; \kappa_i^j \mu_i^j(t) + (1-\kappa_i^j)\frac{\sum_l m_l^j(p_l(t))}{\sum_l p_l(t)}$$

slow timescale update of scaling factor

ns2 simulation: buffer=80pks

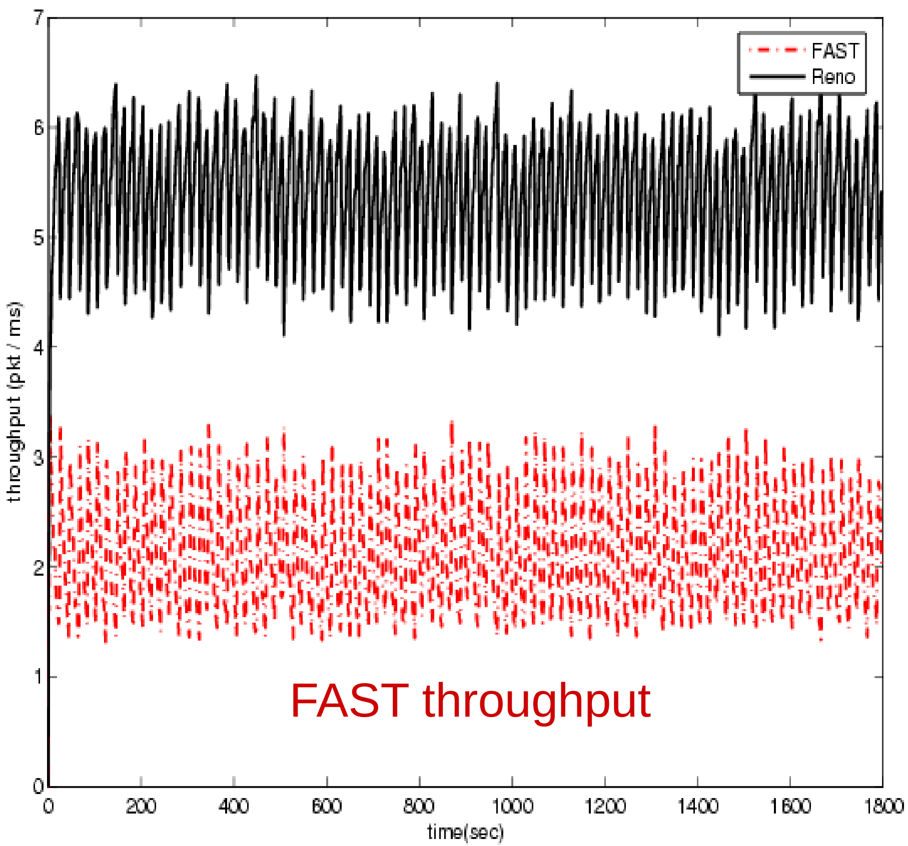


without slow timescale control

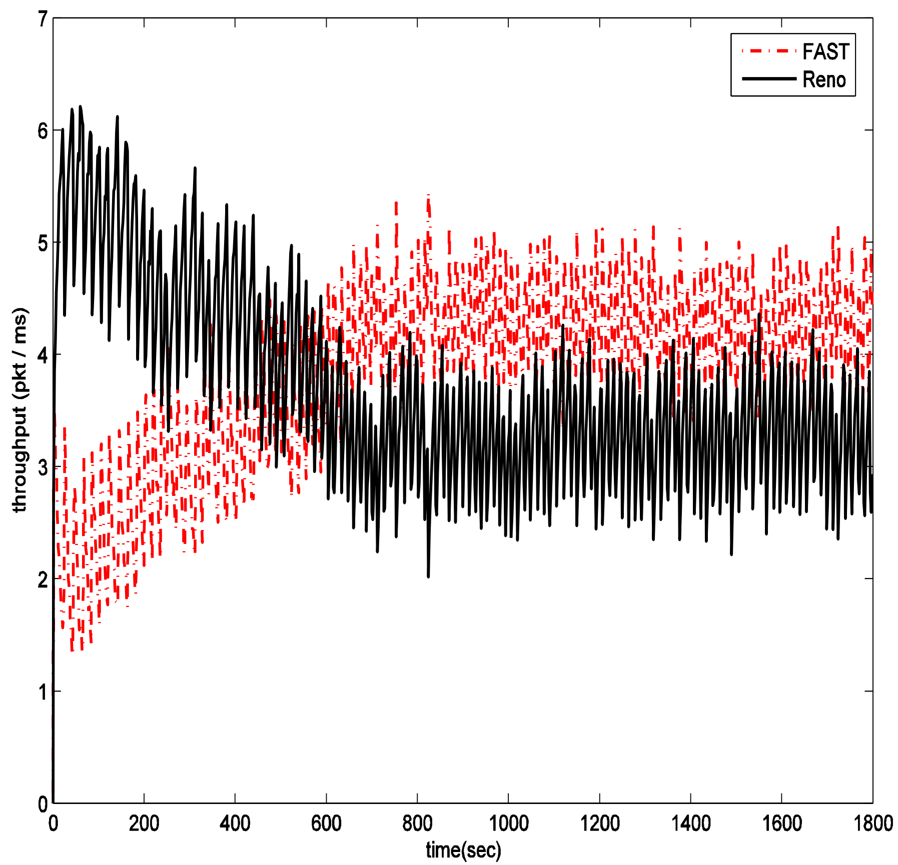


with slow timescale control

ns2 simulation: buffer=400pks



without slow timescale control



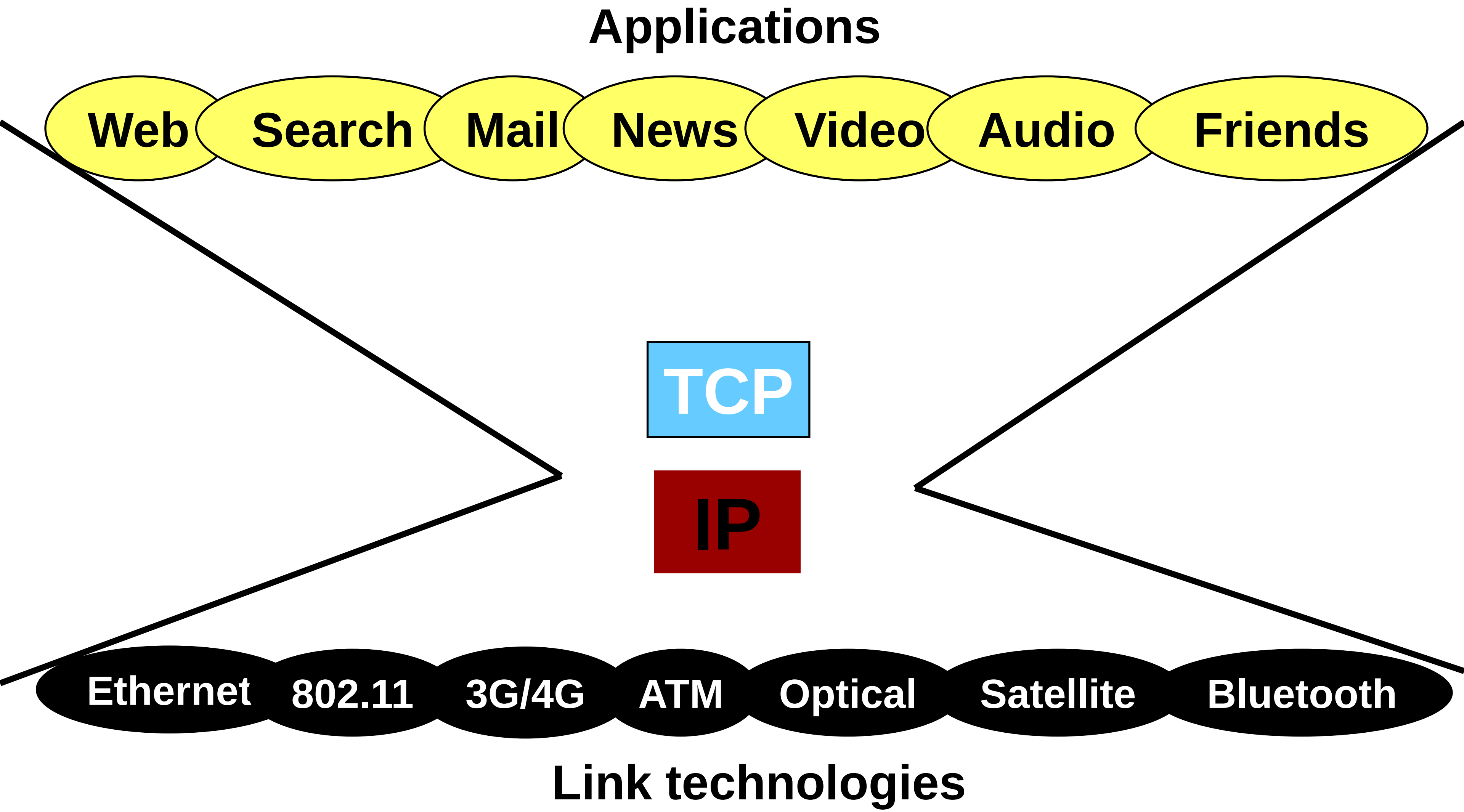
with slow timescale control

Advanced topics

Heterogeneous protocols

Layering as optimization decomposition

The Internet hourglass



But what is architecture

"Architecture involves or facilitates

- System-level function (beyond components)
- Organization and structure
- Protocols and modules
- Risk mitigation, performance, evolution

but is more than the sum of these"

-- John Doyle, Caltech

"... the architecture of a system defines how the system is broken into parts and how those parts interact."

-- Clark, Sollins, Wroclawski, …, MIT

But what is architecture

"Things that persist over time"

"Things that are common across networks"

"Forms that enable functions"

"Frozen but evolves"

"It is intrinsic but artificial"

Key features (John Doyle, Caltech)

- Layering as optimization decomposition
- Constraints that deconstrain
- Robust yet fragile

Layering as opt decomposition

- Each layer designed separately and evolves asynchronously
- Each layer optimizes certain objectives

application	← Minimize response time (web layout)…
transport	← Maximize utility (TCP/AQM)
network	← Minimize path costs (IP)
link	← Reliability, channel access, …
physical	← Minimize SIR, max capacities, …

Layering as opt decomposition

- Each layer is abstracted as an optimization problem
- Operation of a layer is a distributed solution
- Results of one problem (layer) are parameters of others
- Operate at different timescales

application
transport
network
link
physical

Application: utility

$$\max_{x \geq 0} \quad \sum_i U_i(x_i)$$

$$\text{subj to} \quad Rx \leq c(p)$$

$$x \in X$$

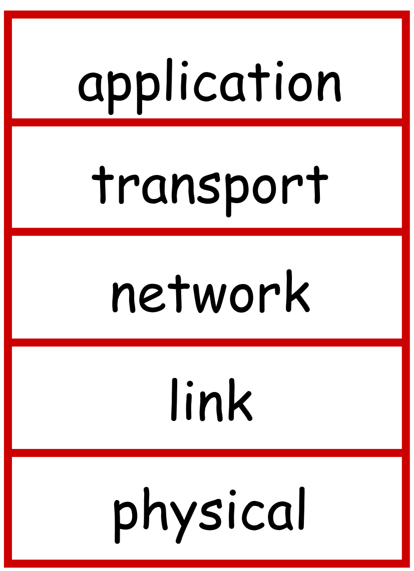
Phy: power

IP: routing

Link: scheduling

Layering as opt decomposition

- Each layer is abstracted as an optimization problem
- Operation of a layer is a distributed solution
- Results of one problem (layer) are parameters of others
- Operate at different timescales



1) Understand each layer in isolation, assuming other layers are designed nearly optimally

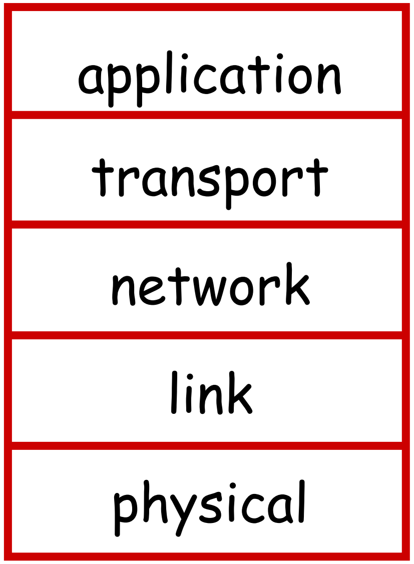
2) Understand interactions across layers

3) Incorporate additional layers

4) Ultimate goal: entire protocol stack as solving one giant optimization problem, where individual layers are solving parts of it

Layering as opt decomposition

Network	generalized NUM
Layers	subproblems
Layering	decomposition methods
Interface	functions of primal or dual vars



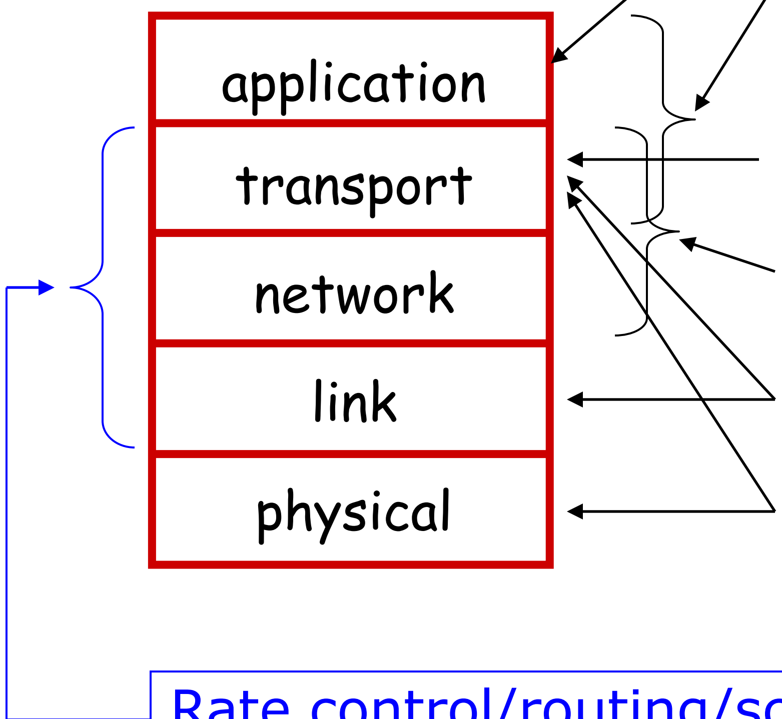
1) Understand each layer in isolation, assuming other layers are designed nearly optimally

2) Understand interactions across layers

3) Incorporate additional layers

4) Ultimate goal: entire protocol stack as solving one giant optimization problem, where individual layers are solving parts of it

Examples



Optimal web layer: Zhu, Yu, Doyle ’01

HTTP/TCP: Chang, Liu ’04

TCP: Kelly, Maulloo, Tan ’98, ……

TCP/IP: Wang et al ’05, ……

TCP/MAC: Chen et al ’05, ……

TCP/power control: Xiao et al ’01, Chiang ’04, ……

Rate control/routing/scheduling: Eryilmax et al ’05, Lin et al ’05, Neely, et al ’05, Stolyar ’05, Chen, et al ’05

detailed survey in Proc. of IEEE, 2006

Example: dual decomposition

Design via dual decomposition

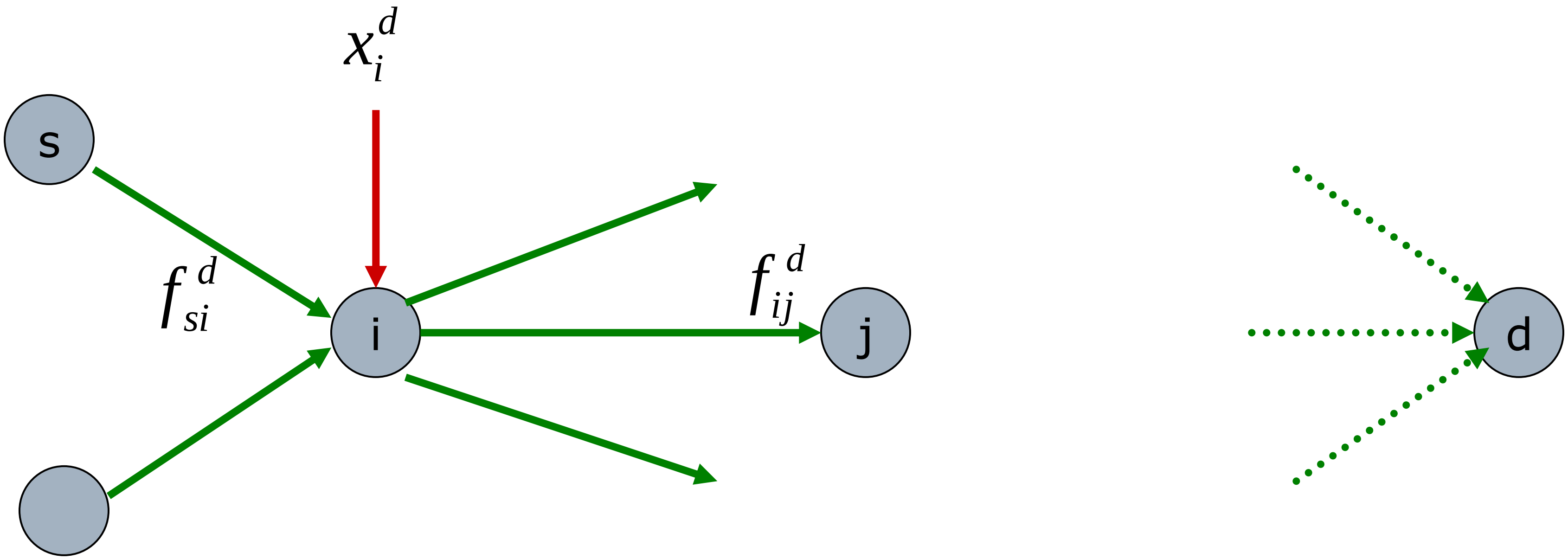
- Congestion control, routing, scheduling/MAC
- As distributed gradient algorithm to jointly solve NUM

Provides

- basic structure of key algorithms
- framework to aid protocol design

Ref:
Cross-layer design in multihop wireless networks Lijun Chen, Steven H. Low and Joh

Wireless mesh network



$$x_i^d \le \sum_j \left(f_{ij}^d - f_{ji}^d \right) \qquad \text{for all } i \in N, d \in D$$

$$x_i^d = 0 \qquad \text{if } i \notin S$$

Wireless mesh network

Underlying optimization problem:

Utility to flows (s,d)

Cost of using links (i,j)

$$\max_{x,f\geq 0} \sum_{(s,d)} U_s^d(x_s^d) - \sum_{(i,j)} \sum_d \lambda_{ij}^d f_{ij}^d$$

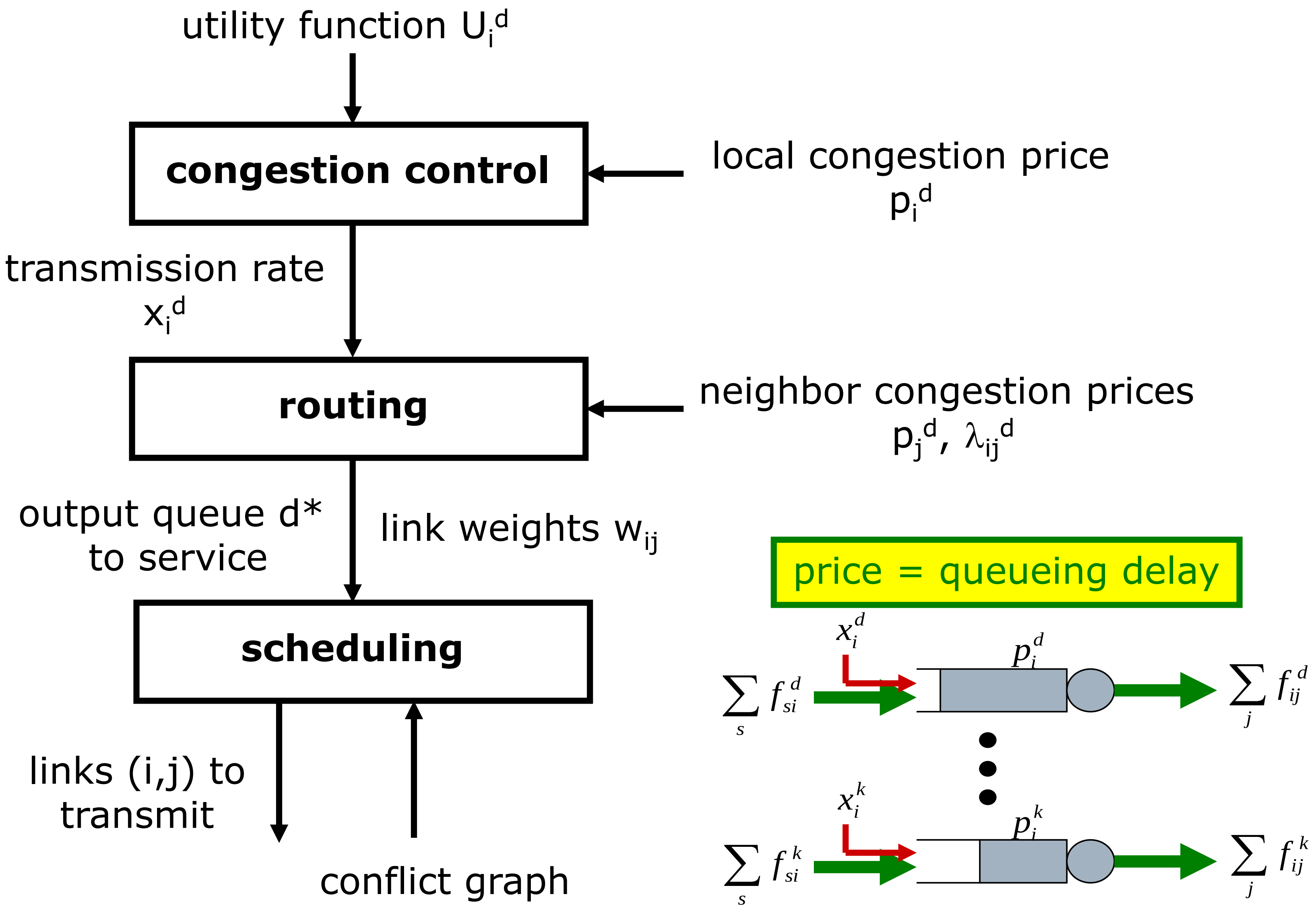
$$\text{s.t.} \quad x_i^d \leq \sum_j \left(f_{ij}^d - f_{ji}^d \right)$$

Local flow constraint

$$f \in \Pi$$

Schedulability constraint

Dual decomposition



Algorithm architecture