XCP: eXplicit Control Protocol

Dina Katabi
MIT Lab for Computer Science

dk@mit.edu
www.ana.lcs.mit.edu/dina

Sharing the Internet Infrastructure

- Is fundamental
  - ❑ Much research in Congestion Control, QoS, DiffServ, Pricing …
- Is difficult because of Scale!

Two Types of Requirements:

1. Efficiency: Use links to maximum capacity
2. Allocation: What is the share of each user?
   - Fairness; Differential Bandwidth Allocation; Priority …

Traditionally, a single mechanism controls both Efficiency and Allocation

Example: In TCP, it is Additive-Increase Multiplicative-Decrease (AIMD)

XCP Approach: Decouple Efficiency and Allocation Controls

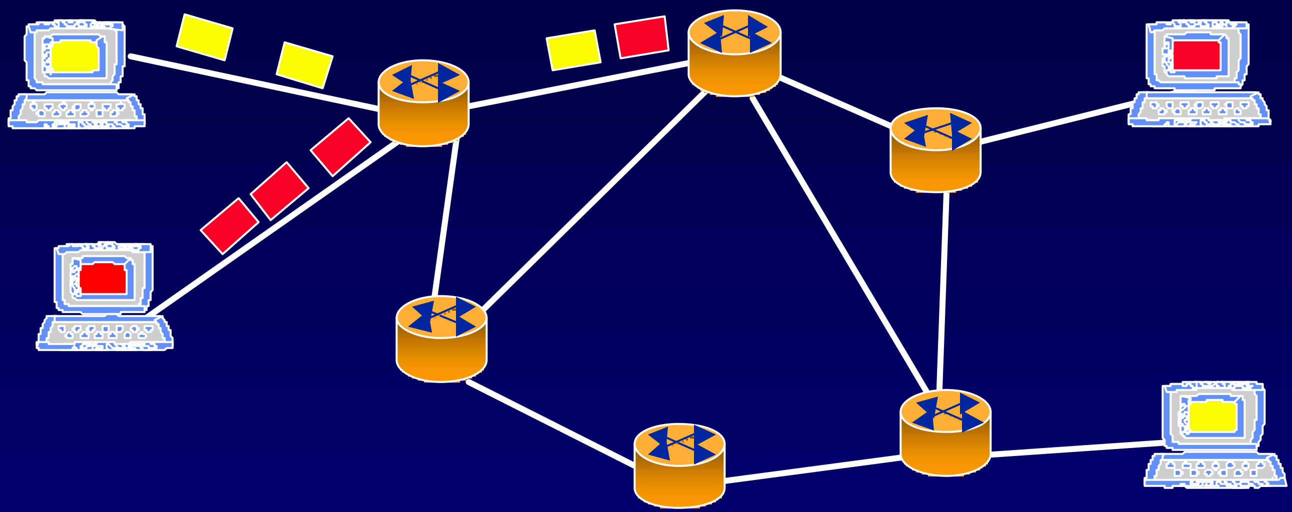
1. Find best mechanism to control aggregate traffic at a link to achieve efficient links utilization
2. Find best mechanism to shuffle the bandwidth in the aggregate traffic to converge to the desired allocation

Decoupling Efficiency Control from Allocation Control

Sharing Internet Resources

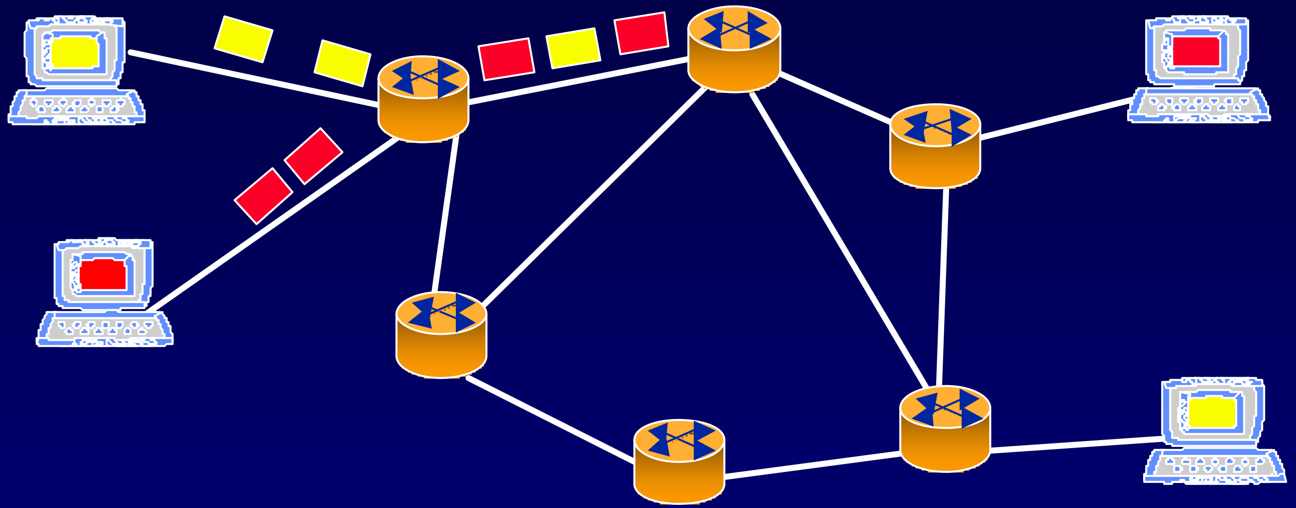
Show it via examples …

Example 1: Congestion Control

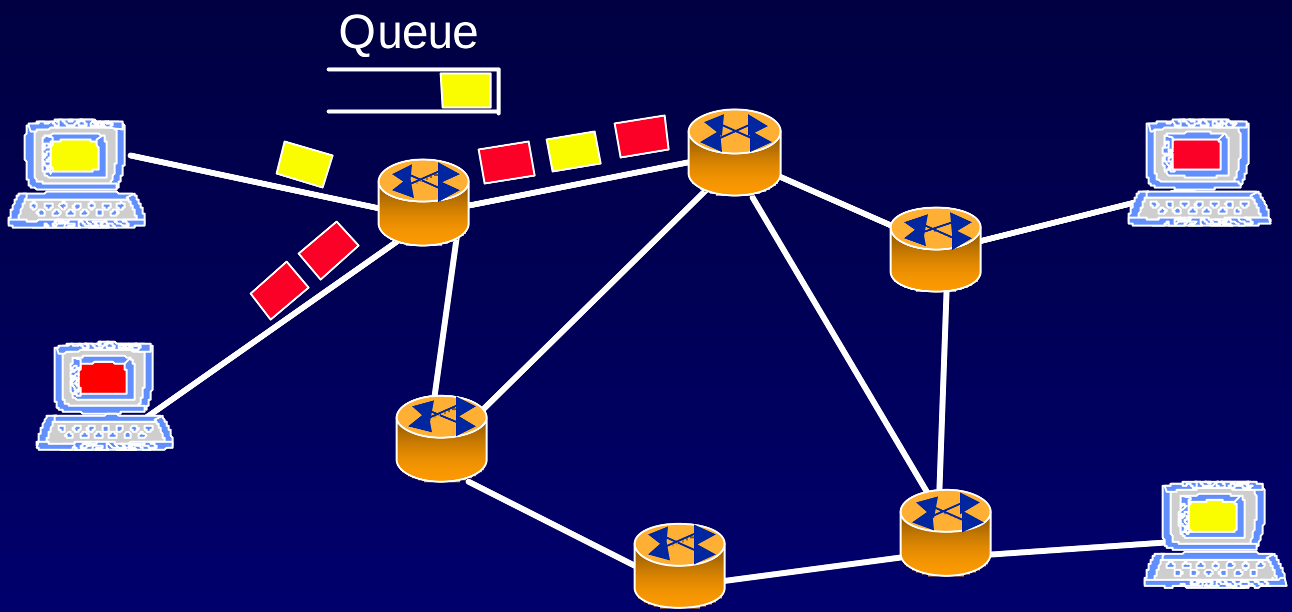


Example 1:

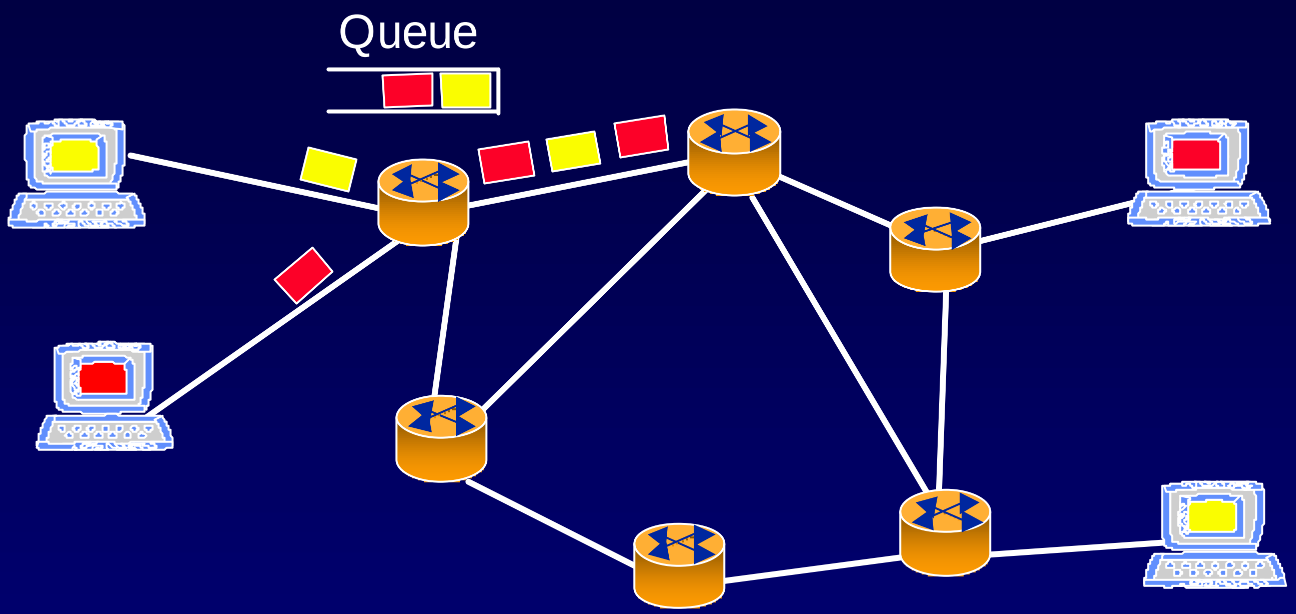
Congestion Control



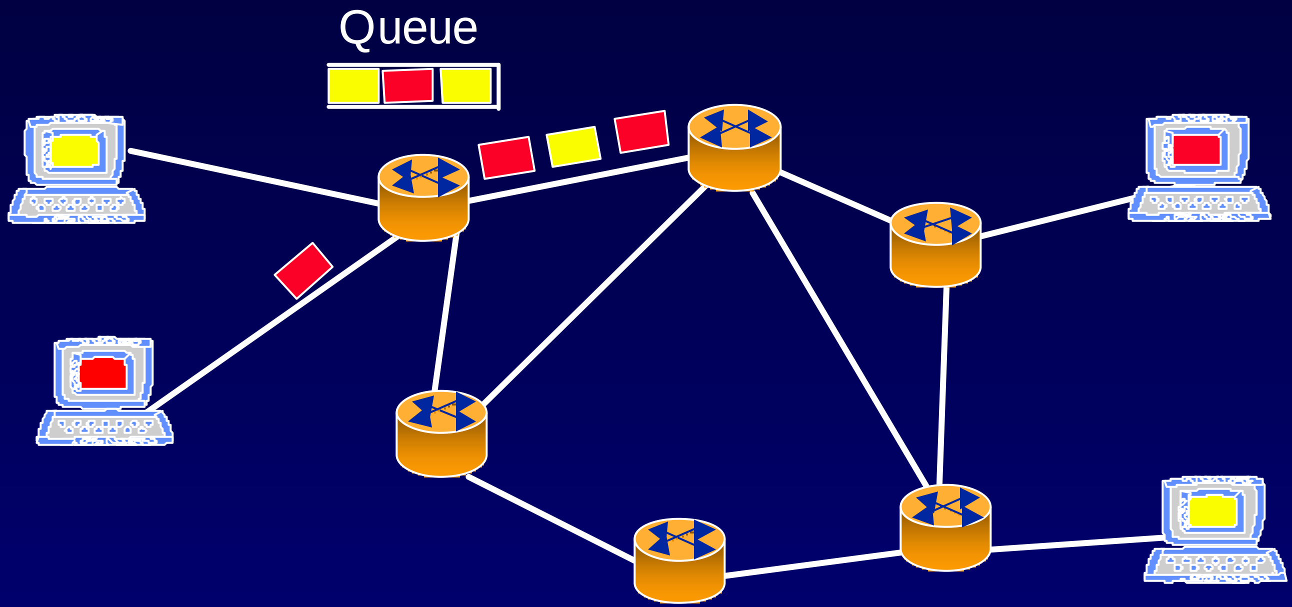
Example 1: 	Congestion Control



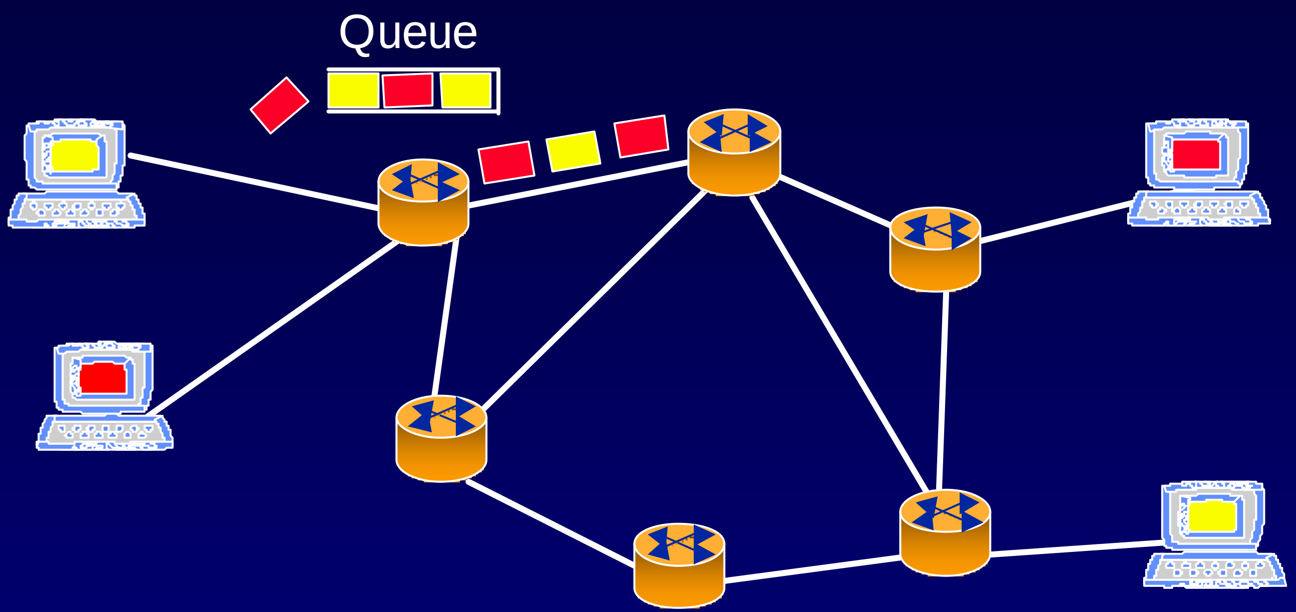
Example 1: Congestion Control



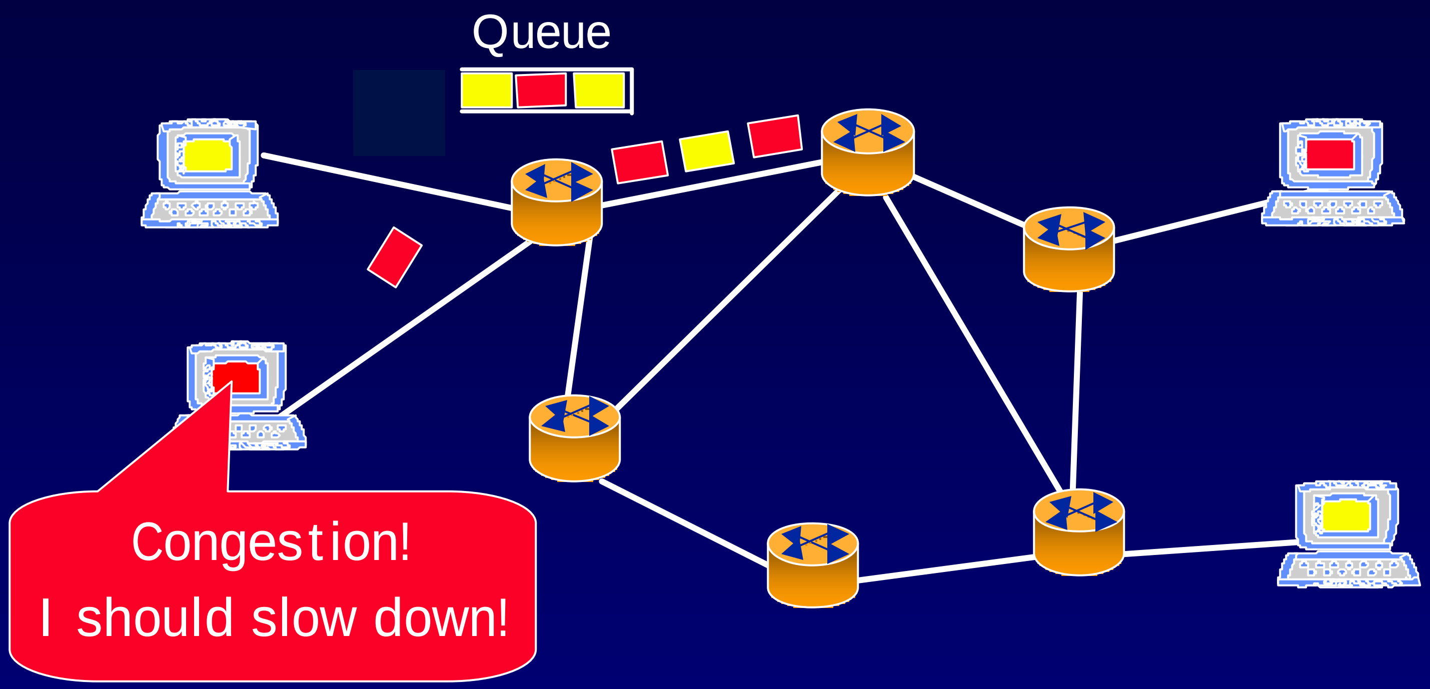
Example 1: Congestion Control

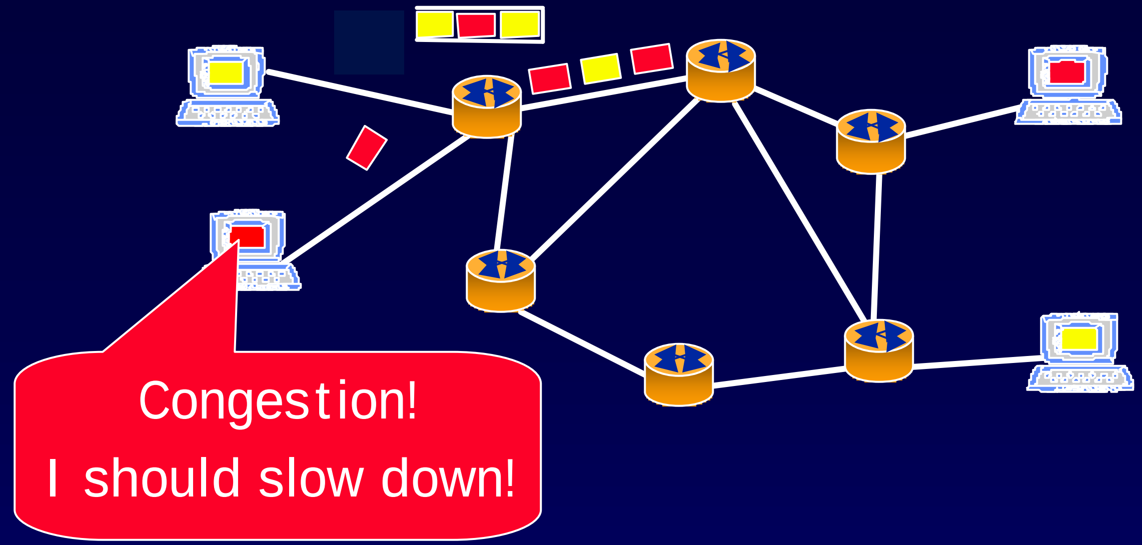


Example 1: Congestion Control



Example 1: Congestion Control



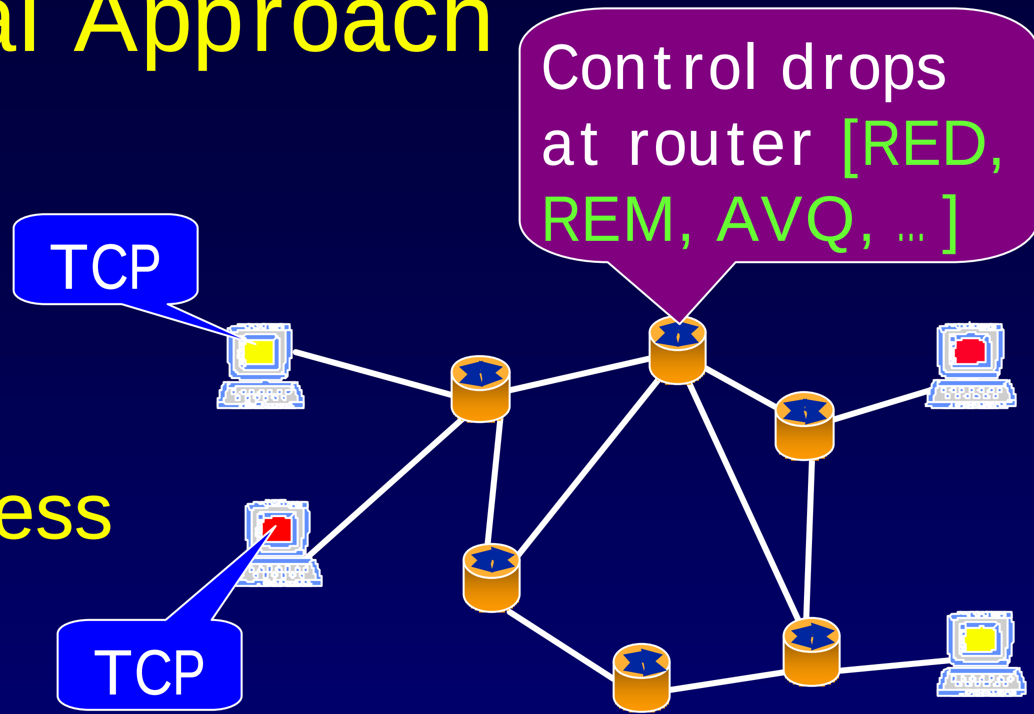


The Congestion Control Problem

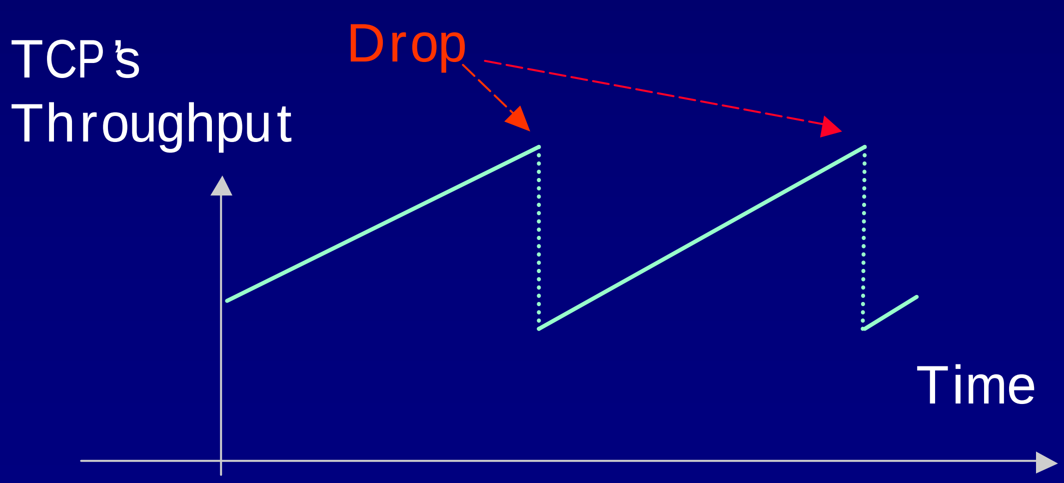
Control the sources' rates to get:

- Efficiency: good link utilization, small queues, few drops
- Fairness: Senders congested at same link get equal throughput

Traditional Approach



TCP couples
Efficiency & Fairness

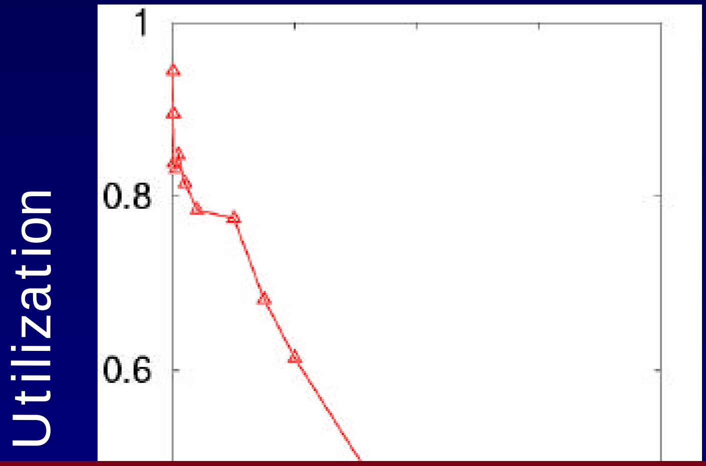


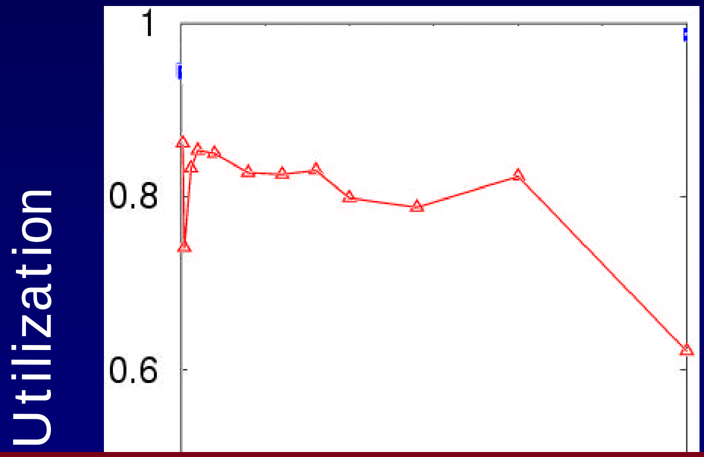
TCP uses AIMD:

- No Drop: Increase by a constant increment (i.e., 1 packet/RTT)
- Drop: Halve throughput

Problems with Current Approaches:

- Good performance requires parameter tuning [RED, ARED, REM, PI-controller, AVQ, …]
- Inefficient as bandwidth or delay increases [Low02]





⇒ Need to change congestion control because:

- Bandwidth is increasing (demands for it are increasing too!) making TCP more inefficient
- Delay is already a problem

Congestion Control is Inefficient Because:

- Congestion feedback is binary (i.e., drop or no-drop) and indifferent to the degree of congestion
  - As a result, TCP oscillates between over-utilizing the link and under-utilizing it

Solution:

Efficient congestion control requires **Explicit** feedback

(I.e., routers tell senders the degree of congestion)

Why Current Approaches Don't Use Expressive Feedback?

Unexpressive & Scalable

TCP, TFRC, Binomial, …

Expressive & Unscalable

In ATM: ERICA, Charny's, OSU, …

(almost none in the Internet)

Answer: Per-flow state in routers ⇒ Doesn't Scale!

(Flow: packets from same sender)

Unexpressive & Unscalable

Expressive & Scalable

Efficiency Problem:

- Efficient link utilization needs expressive feedback
- In coupled systems, expressive feedback led to per-flow state (Unscalable!)

Solution: Use Decoupling

- Decoupling looks at efficiency as a problem about aggregate traffic
  - Match aggregate traffic to link capacity and drain the queue
- Benefits: No need for per-flow information

Fairness Control

Router computes a flow's fair rate explicitly

↓

To make a decision, router needs state of all flows

↓

Unscalable

Shuffle bandwidth in aggregate to converge to fair rates

↓

To make a decision, router needs state of this flow

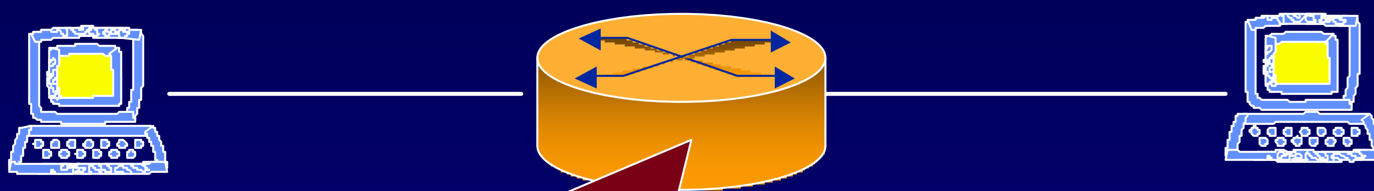
↓

Put a flow's state in its packets [Stoica]

↓

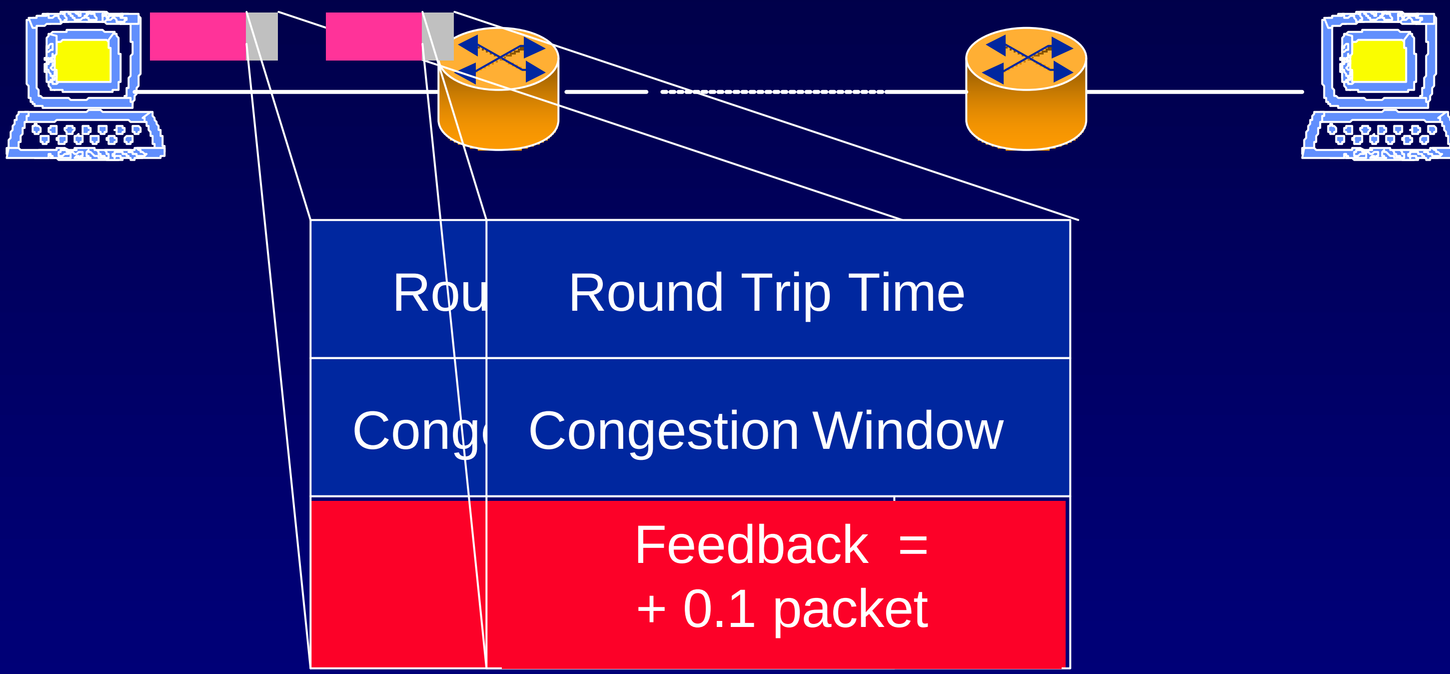
Scalable

XCP: An eXplicit Control Protocol



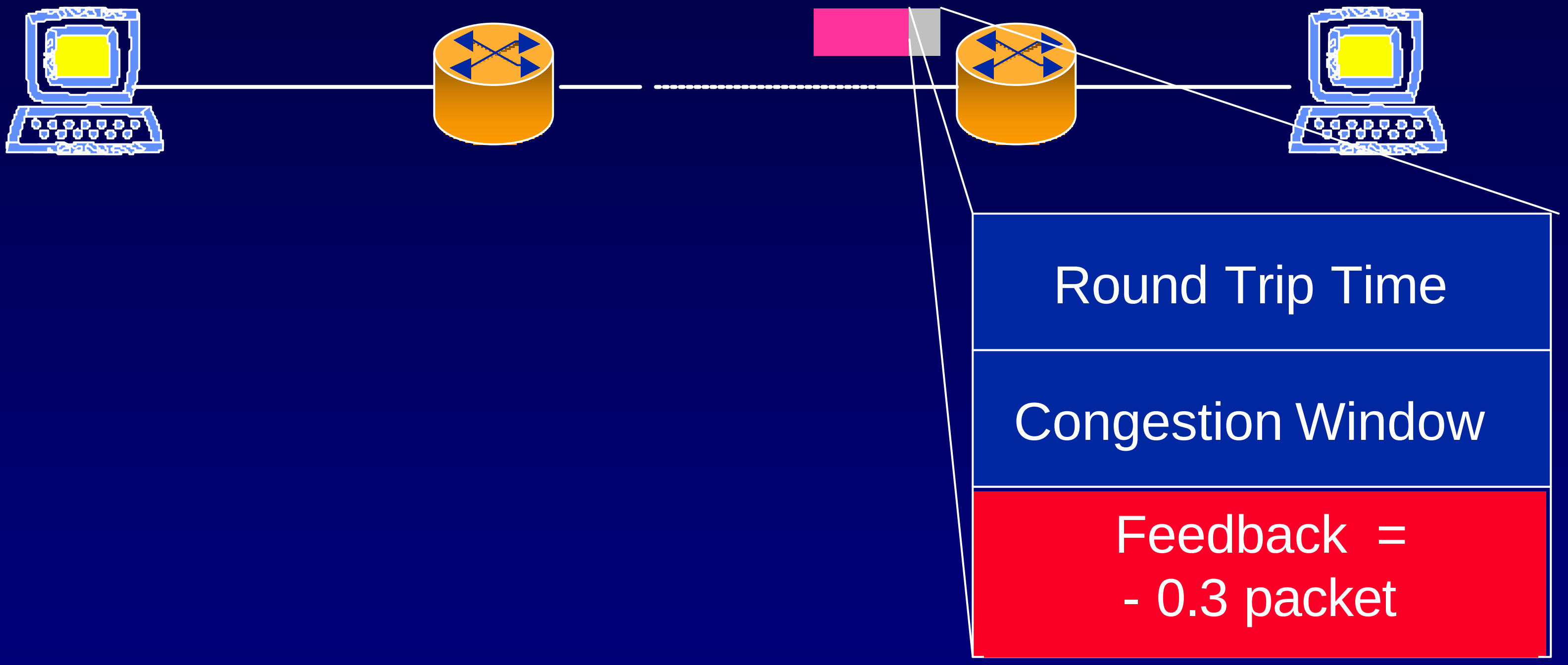
1. Efficiency Controller
2. Fairness Controller

How does XCP Work?

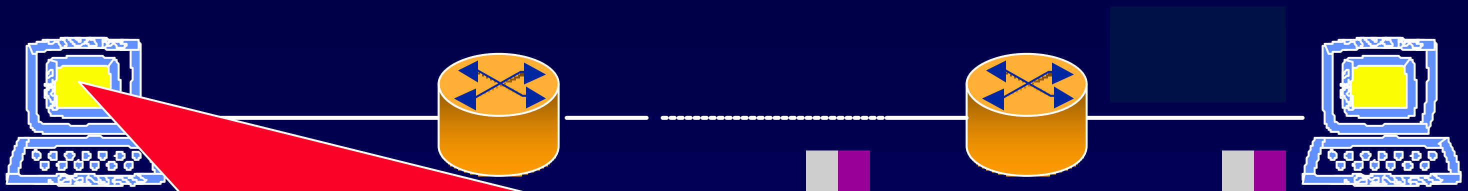


Congestion Header

How does XCP Work?



How does XCP Work?



Congestion Window = Congestion Window + Feedback

Routers compute feedback without keeping any per-flow state

How Does an XCP Router Compute the Feedback?

Efficiency Controller

Goal: Matches input traffic to link capacity & drains the queue

Looks at aggregate traffic & queue

MIMD

Algorithm:

Aggregate traffic changes by Δ

Δ ~ Spare Bandwidth

Δ ~ - Queue Size

So, $D = a\ d_{avg}\ Spare - b\ Queue$

Fairness Controller

Goal: Divides Δ between flows to converge to fairness

Looks at a flow's state in Congestion Header

AIMD

Algorithm:

If Δ > 0 ⇒ Divide Δ equally between flows

If Δ < 0 ⇒ Divide Δ between flows proportionally to their current rates

(Proven to converge to fairness)

Windows change by Δ every d_{avg}

Traffic rate changes by $\frac{\Delta}{d_{avg}}$ every d_{avg}

Rate $r(t)$ changes per time unit by $\dot{r} = \frac{\Delta}{d_{avg}^2}$

$$\dot{r} = \frac{a\ S(t)}{d_{avg}} - \frac{b\ Q(t)}{d_{avg}^2}$$

Efficiency (

$D = a\ d_{avg}$ Spar

Theorem: Syste

(I.e., converges to efficiency) for any link bandwidth, delay, number of sources if:

$$0 < a < \frac{\pi}{4\sqrt{2}} \quad and \quad b = a^2\sqrt{2}$$

No Parameter Tuning

Need to estimate number of flows N

$$N = \sum_{pkts\ in\ d_{avg}} \frac{\cancel{RTT_i}}{\cancel{d_{avg}} \times Cwnd_i}$$

No Per-Flow State

It Is Tricky …

Efficiency Controller

D = a d_{avg} Spare - b Queue

Theorem: System is stable (I.e., converges to efficiency) for any link bandwidth, delay, number of sources if:

$$0 < a < \frac{p}{4\sqrt{2}} \quad and \quad b = a^2\sqrt{2}$$

No Parameter Tuning

Fairness Controller

Algorithm:

If $\Delta > 0 \Rightarrow$ Divide Δ equally between flows

If $\Delta < 0 \Rightarrow$ Divide Δ between flows proportionally to their current rates

Need to estimate number of flows *N*

$$N = \sum_{pkts\ in\ d_{avg}} \frac{\cancel{RTT_i}}{\cancel{d_{avg}} \times Cwnd_i}$$

No Per-Flow State

Implementation

Implementation uses few multiplications & additions per packet → Practical!

Gradual Deployment

XCP can co-exist with TCP and can be deployed gradually

Performance

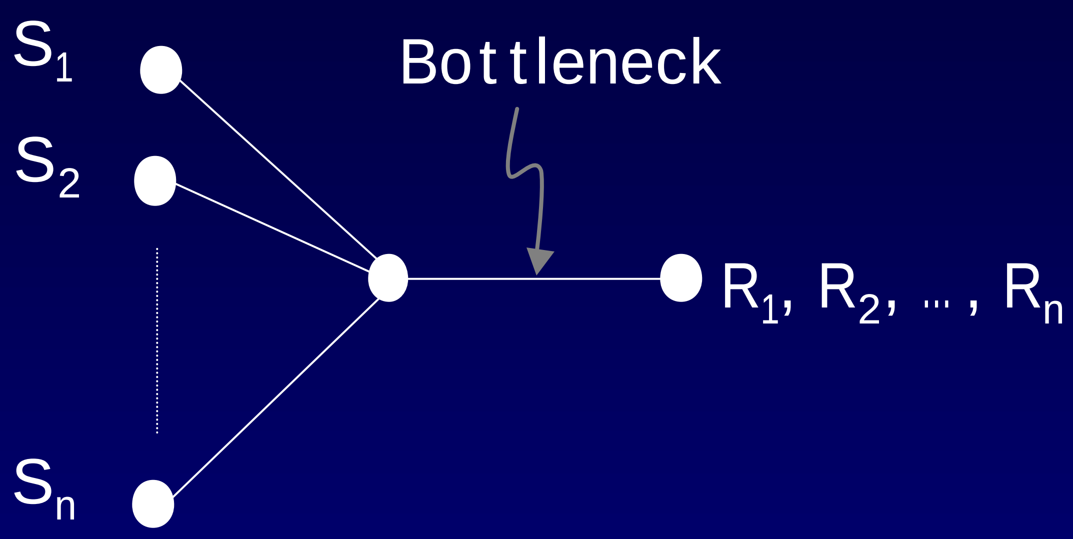
Simulations Show XCP is Better

- Extensive Simulations
- Compared with TCP over DropTail, RED, REM, AVQ, CSFQ

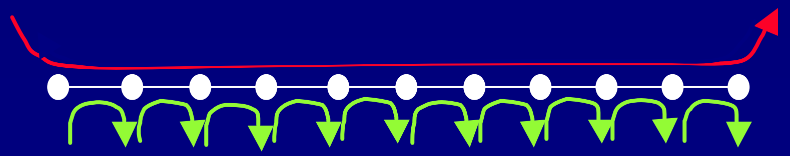
XCP:

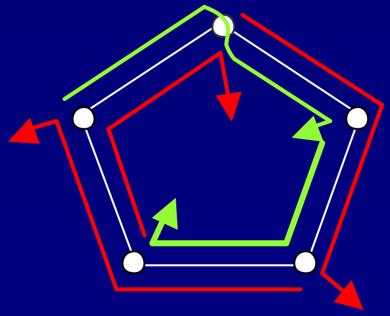
- Better utilization
- Near-zero drops
- Fairer
- Efficient & robust to increase in bandwidth
- Efficient & robust to increase in delay

Subset of Results



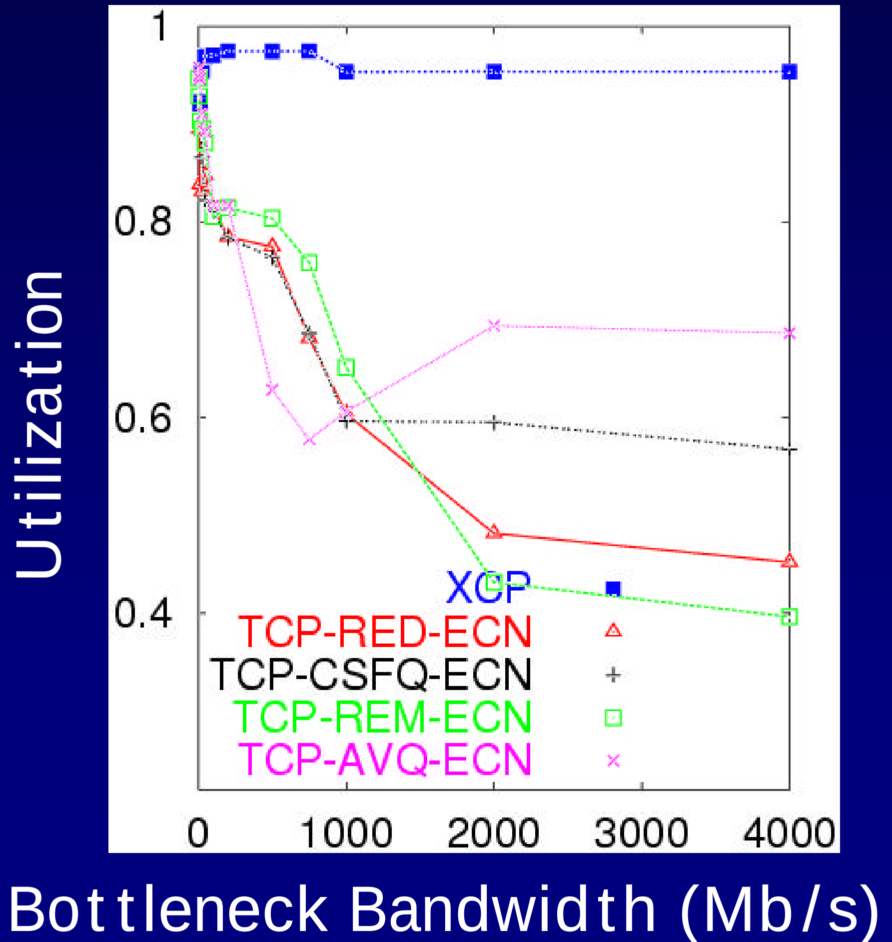
Similar behavior over:



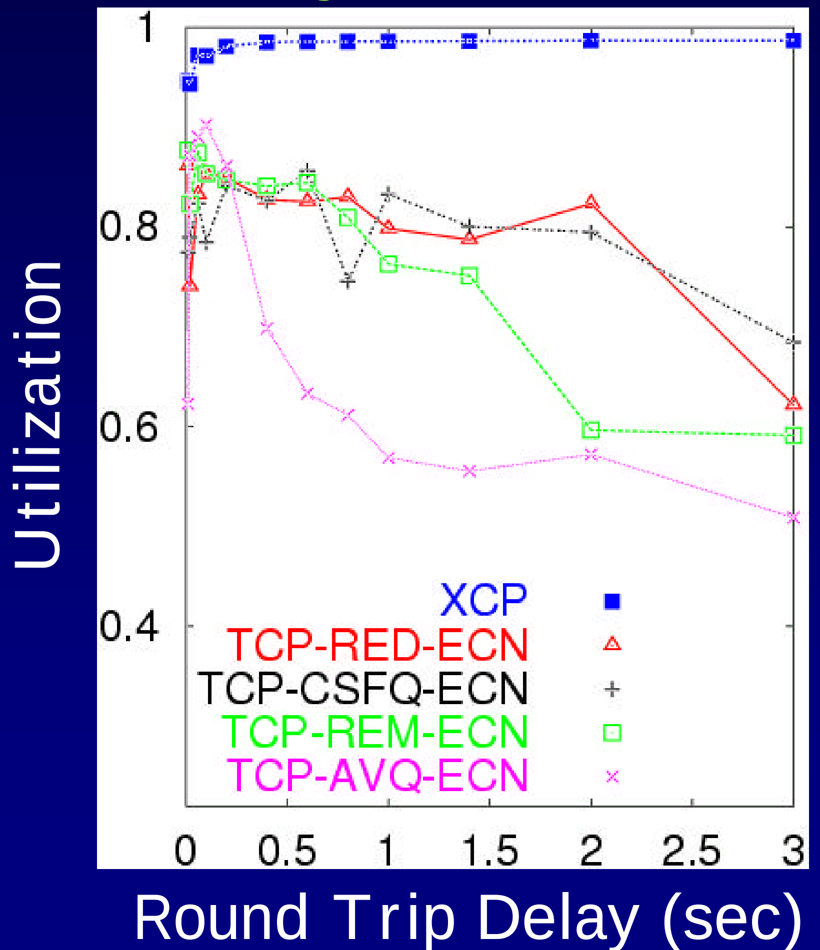


XCP Remains Efficient as Bandwidth or Delay Increases

Utilization as a function of Bandwidth

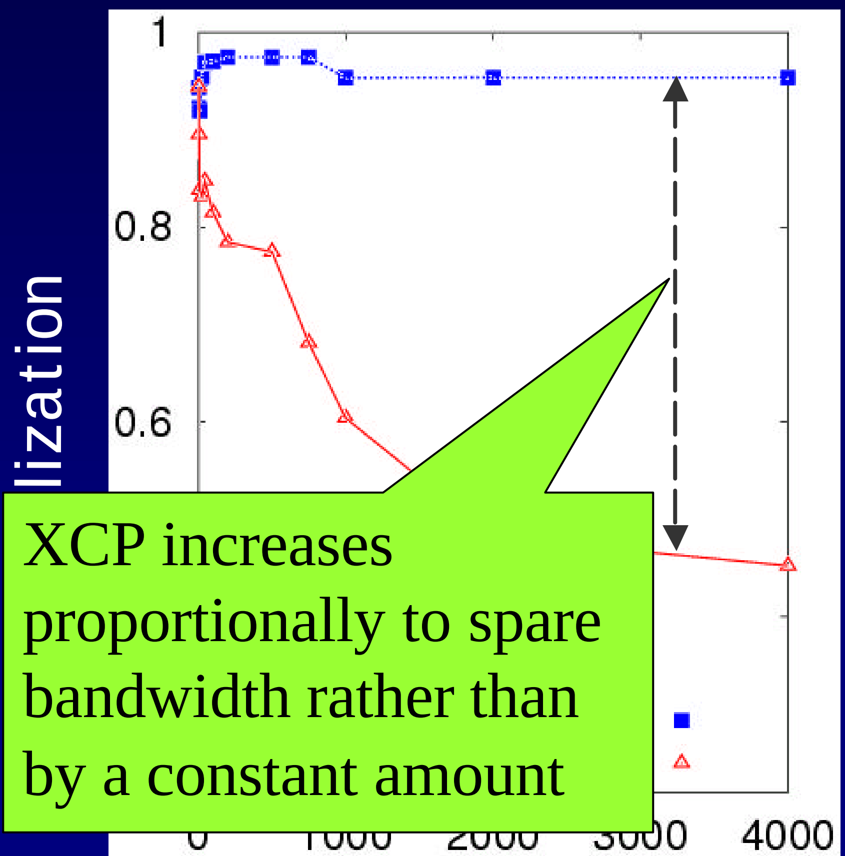


Utilization as a function of Delay



XCP Remains Efficient as Bandwidth or Delay Increases

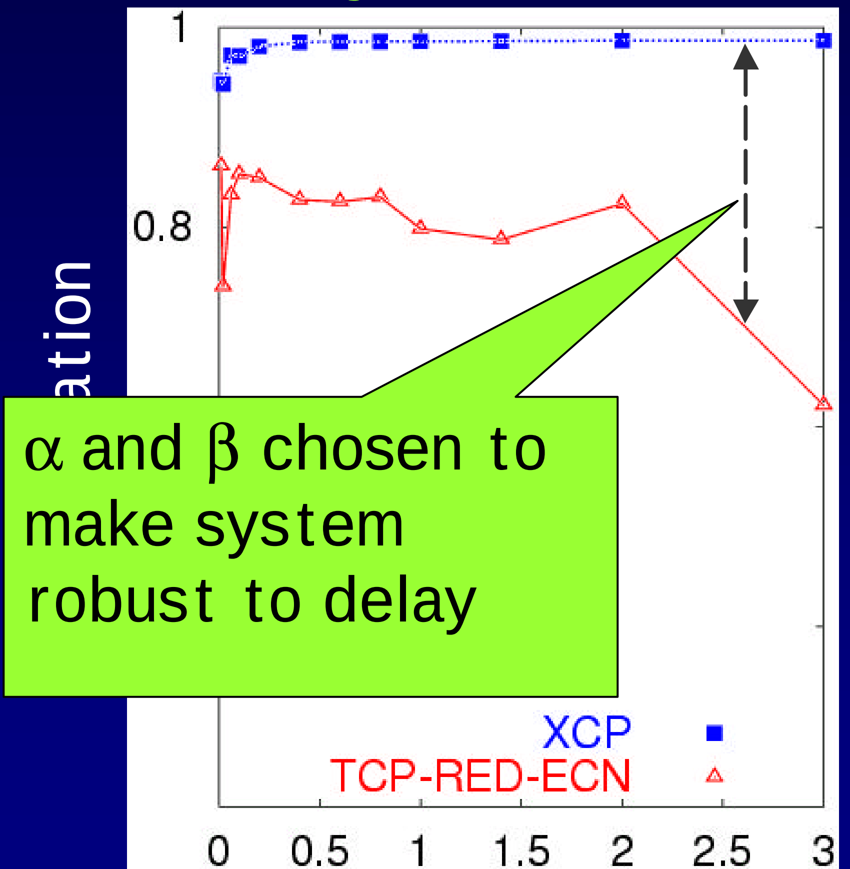
Utilization as a function of Bandwidth



XCP increases proportionally to spare bandwidth rather than by a constant amount

Bottleneck Bandwidth (Mb/s)

Utilization as a function of Delay

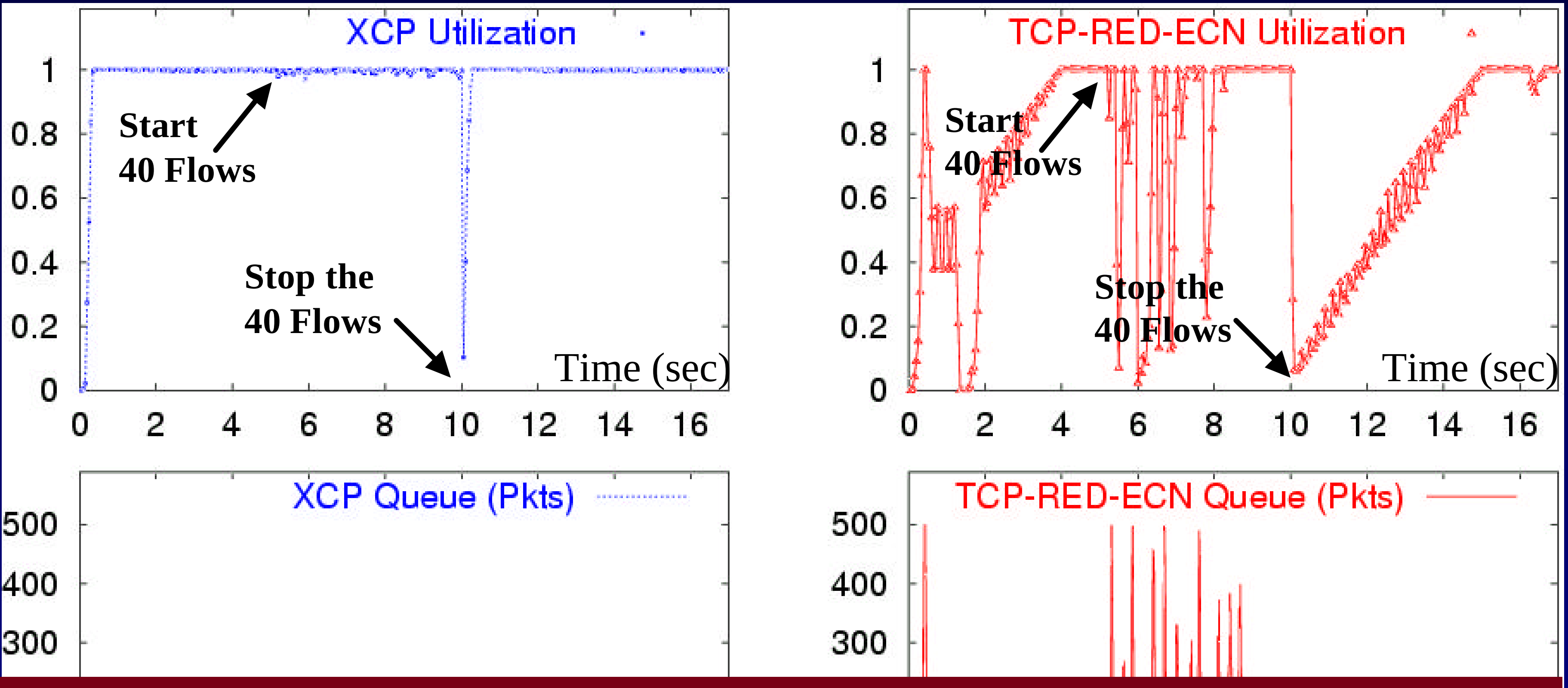


α and β chosen to make system robust to delay

Round Trip Delay (sec)

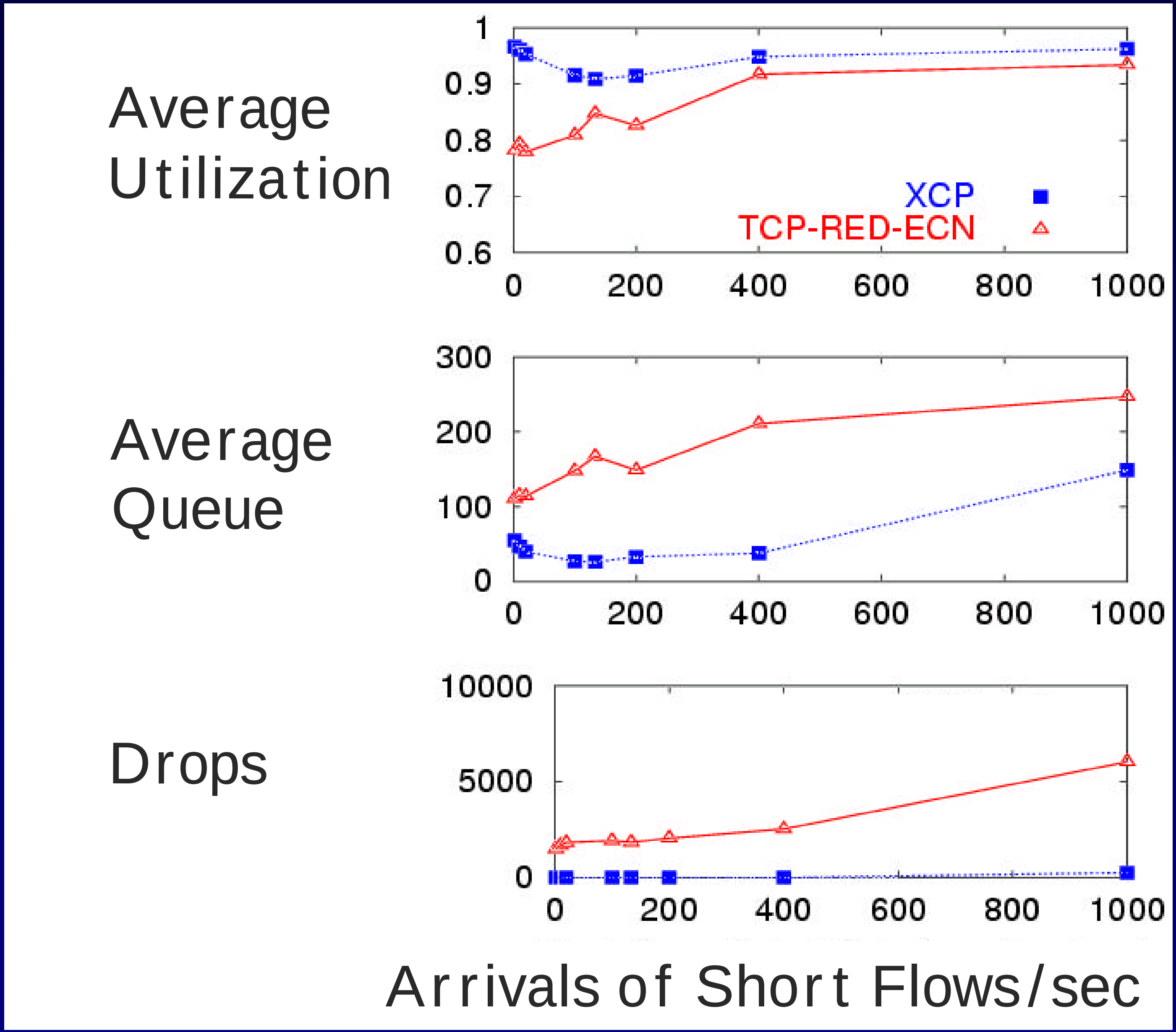
XCP is More Efficient than TCP

RTT = 40ms, C = 100 Mbps



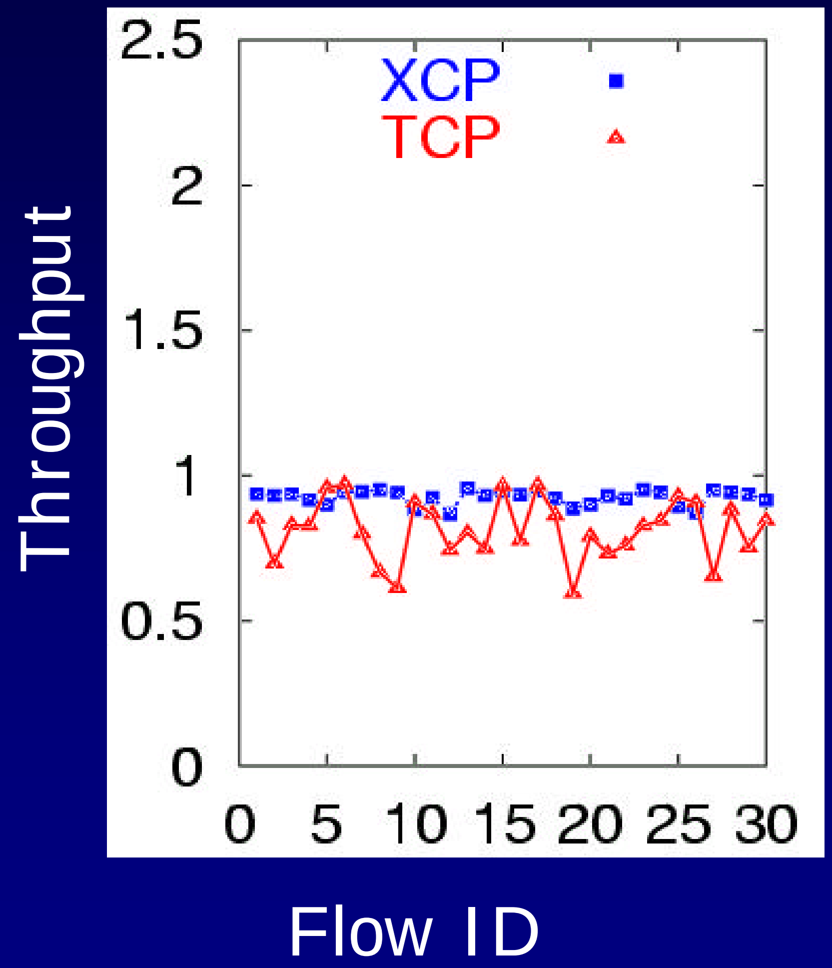
XCP shows fast adaptation!

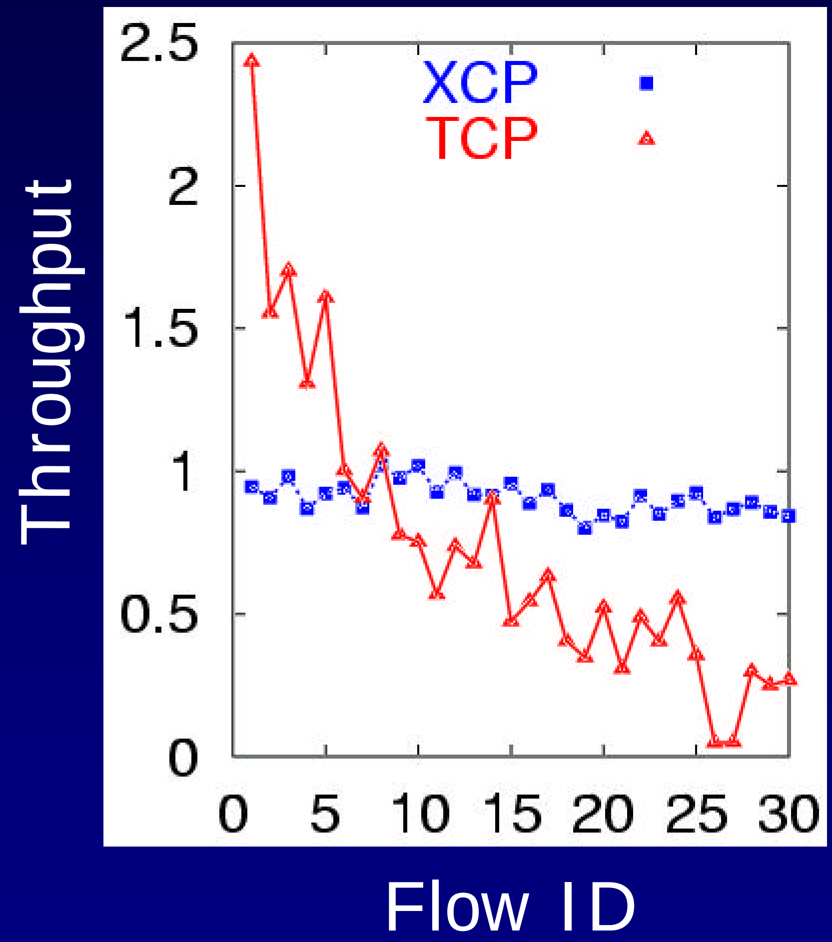
XCP Deals Well with Short Web-Like Flows



XCP is Fairer than TCP

Same Round Trip Delay | **Different Round Trip Delay**





(RTT is from 40 ms to 330 ms)

XCP Summary

- XCP
  - Outperforms TCP
  - Efficient for any bandwidth
  - Efficient for any delay
  - Scalable

- Benefits of Decoupling
  - Efficient utilization becomes about aggregate traffic $\Rightarrow$ No need for per-flow state
  - Stability analysis looks only at Efficiency Controller (independent of number of flows)

Decoupling Efficiency Control from Allocation Control

Sharing Internet Resources

Example 2:

Differential Service

Problem Control sources' rates to get:

- Efficiency :
  - Good utilization, small queues, and few drops
- Differential Bandwidth Allocation [Kelly]):
  - Each user pays a price per unit time
  - Users congested at the same link obtain throughputs proportional to their respective prices

Efficiency Controller

Decoupling allows us to use XCP's Efficiency Controller

Modularization

& Reuse

Allocation Controller

- <u>Goal:</u>
  - Converge to differential bandwidth allocation
  - Decoupling ⇒ Don't have to worry about efficiency

- <u>Algorithm:</u>
  - If $\Delta > 0 \Rightarrow$ Divide Δ equally between flows
    If $\Delta < 0 \Rightarrow$ Divide Δ between flows proportionally to their current rate/price

- <u>Implementation:</u>
  - Substitute the congestion window field by congestion window/price

Round Trip Time
$\frac{\text{Congestion Window}}{\text{Price}}$
Feedback

Benefits of Decoupling

- Allocation Controller can use a new class of algorithms that converge to desired allocation but not to efficiency
  - Doesn't work without decoupling! E.g., modifying TCP to "Increase by one packet & Decrease proportionally to rate/price." drops too many packets

Performance

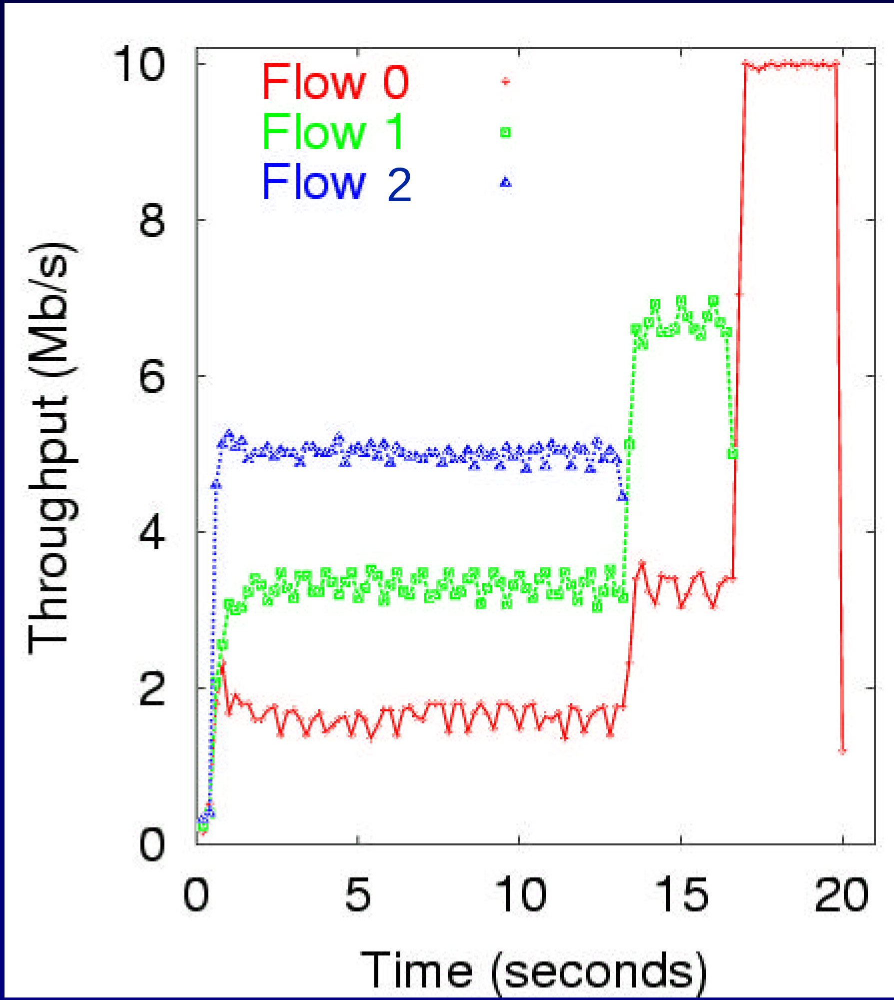
Experiment:

3 sources transferring a 10 MB file each

- Price 0 = 5
- Price 1 = 10
- Price 2 = 15

Result:

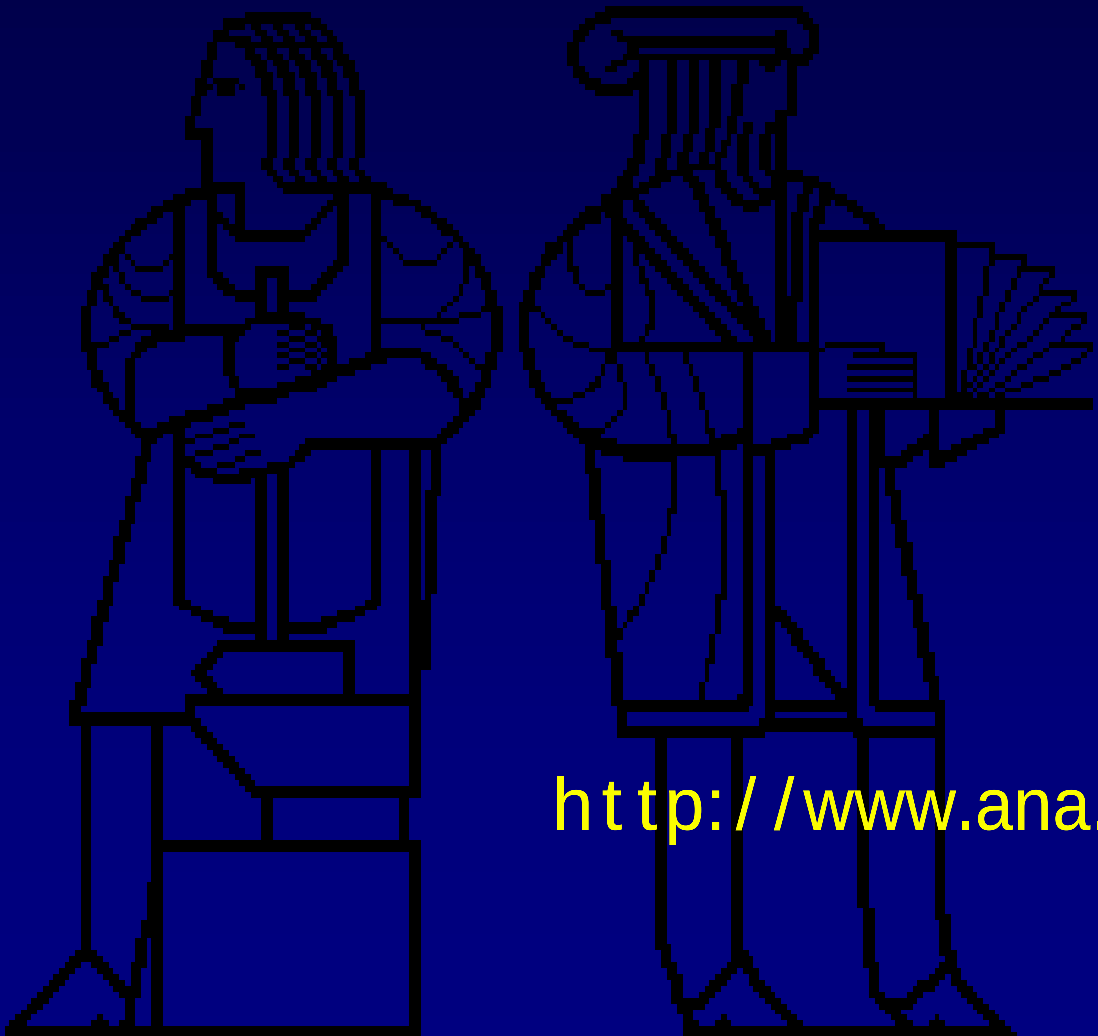
Users share the link proportionally to their prices



Conclusion

- Decoupling Efficiency control from Allocation control is useful for resource management
  - Efficiency control is independent of varying parameters such as number of flows
  - Modularization & reuse of controllers
  - Allocation control does not care about utilization issues $\Rightarrow$ Can use a new class of aggressive allocation algorithms
- Currently applying decoupling to guaranteed service, priority service, reaction over different time scale, …

Questions?



http://www.ana.lcs.mit.edu/dina/XCP