How TCP congestion control is broken
A partial list...

Efficiency

Tends to fill queues
  • creates latency and loss

Slow to converge
  • for short flows or links with high bandwidth•delay product

Loss ≠ congestion

Often does not fully utilize bandwidth
A partial list...

Fairness

Unfair to large-RTT flows (less throughput)

Unfair to short flows if ssthresh starts small

Equal rates isn’t necessarily “fair” or best

Vulnerable to selfish & malicious behavior

• TCP assumes everyone is running TCP!
Fills queues: adds loss, latency

Slow to converge

Loss $\neq$ congestion

May not utilize full bandwidth

Unfair to large-RTT

Unfair to short flows

Is equal rates really “fair”? 

Vulnerable to selfishness
Limitations of TCP CC

- Fills queues: adds loss, latency
- Slow to converge
- Loss ≠ congestion
- May not utilize full bandwidth
- Unfair to large-RTT
- Unfair to short flows
- Is equal rates really “fair”? 
- Vulnerable to selfishness

Hard to use only end-to-end information to find ‘right’ rate

Obvious solution: Get more info from network
Limitations of TCP CC

- Fills queues: adds loss, latency
- Slow to converge
- Loss ≠ congestion
- May not utilize full bandwidth
- Unfair to large-RTT
- Unfair to short flows
- Is equal rates really “fair”? 
- Vulnerable to selfishness

- Hard to use only end-to-end information to find ‘right’ rate
- Obvious solution: Get more info from network
- Incentive issues
Congestion control with help from the network
Random early detection (RED)

- Drops more packets (randomly) as congestion increases
- Mechanism is entirely within routers

Explicit Congestion Notification (ECN)

- Mark bit in header instead of dropping

But what does the source really want?

- Just tell me the right rate, already!
- eXplicit Control Protocol (XCP)
- Rate Control Protocol (RCP)
Flows finish slowly

= fair queueing

[Dukkipati & McKeown '05]

Figure 1. The top plot shows the average flow duration versus flow size. The middle plot shows the number of active flows versus time. In both cases, TCP and XCP result in much longer flow durations compared to ideal PS, as shown in the bottom plot. The graph is representative of hundreds of graphs we obtained for different network conditions and traffic models.

In this section, we will try to explain why both mechanisms cause flows to finish slowly. There are several reasons for the long duration of flows with TCP. First, it takes "slow-start" several round-trip times to finish, which means the fair-share rate is lower and therefore operates below – often well below – their fair-share rate. Because of this, for a given network load, there are several reasons for the long duration of flows with TCP. There are four main reasons: (1) Flows start too slowly and are therefore artificially stretched over multiple round-trip times, (2) Bandwidth is allocated unfairly to some flows at the expense of others; (3) Flows are stretched over multiple round-trip times, and (4) Timeouts and retransmissions due to packet losses (TCP). We will examine each reason in turn, and use a simple deterministic example in Figure 3. In the AIMD phase, it is slow in catching up with any spare capacity. Slow-start plus slow adaption prolong flows unnecessarily. There seem to be four main reasons why both mechanisms result in such long flow durations. In PS, both flows would complete in one RTT, whereas in TCP, the number of active flows would be equal to the number of flows times the number of RTTs, which is six times higher than in PS. In XCP, the number of active flows is equal to the number of flows times the number of RTTs, which is ten times higher than in PS. For a given network load, there are several reasons why both mechanisms result in such long flow durations. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In XCP, the number of active flows equals the equilibrium number of flows in system times the number of RTTs. In TCP, the number of active flows equals the equilibrium num
Many flows waiting

[Dukkipati & McKeown '05]

Fig. 1. The top plot shows the average flow duration versus flow size under various network conditions (explained in the caption) were chosen to be representative of traffic over a backbone link today, and the middle and bottom plots the number of active flows versus time and the bottom plot shows the number of active flows versus time. In both plots, XCP results in a much lower number of flows compared to TCP and PS. The reason for this is that XCP allows for more flows to be active at any one time than TCP or PS, which results in a lower fair-share rate. Because of this, for a given network load, there are fewer flows in progress with XCP than with TCP or PS, which results in a lower link utilization.

In this section, we will try to explain why both mechanisms stretch flows to last many Round Trip Times (RTT) even if they are capable of finishing within one/few RTTs. There seem to be four main reasons for the long duration of flows with TCP.

1. Buffers are filled (TCP) and therefore delay all packets, and (4) Timeouts and retransmissions due to packet losses (TCP). We will examine each reason in turn, and use simple examples to clarify each factor.

A. Stretching flows to last many Round Trip Times (RTT) even if they are capable of finishing within one/few RTTs

Once a flow is in the AIMD phase, it is slow in catching up with any spare capacity. Slow-start plus slow adaptation due to AIMD results in long flow durations. This is illustrated by AIMD results in long flow durations. This is illustrated by AIMD.

B. Slow-start, the FCT for a flow of size 50 pkts is 50 times the RTT, whereas if TCP never entered the congestion-avoidance phase, the expected FCT would be 50 packets × RTT. This is 50 times the RTT. In TCP, the FCT is larger. The number of active flows at any one time is lower and therefore operate below – often well below – their fair-share rate. Because of this, for a given network load, there are fewer flows in progress with XCP than with TCP or PS, which results in a lower link utilization.

C. In PS, both flows would complete in one RTT, with size 50 pkts, arrive at the start of every RTT beginning at the same instant. In XCP and TCP, the equilibrium number of flows in system is 1.4, link-capacity = 2.4 Gbps, Round Trip Time = 100 ms, offered load = 0.5 C/L.

D. So why do TCP and XCP result in such long flow durations? One reason is that TCP and XCP adapt slowly to the network conditions and traffic models. The value of the congestion window is slowly increased, and therefore the flow duration is prolonged. This is illustrated in Figure 2, which shows an example with TCP and XCP. In the top plot, we can see that most flows never finish faster, which means the fair-share rate is lower and therefore the link utilization is lower. Slow-start adds to the duration of each flow even further. XCP is designed to work well in networks with large per-flow capacities, and TCP attempts to work well in networks with small per-flow capacities.
Rate Control Protocol [Dukkipati, Kobayashi, Zhang-Shen, McKeown, IWQoS 2005]

Router’s algorithm:

- Compute fair per-flow rate $R(t)$ at time $t$ as whatever will fill up the link capacity (roughly)
- Tell end-hosts about this by putting the value in packets, and recompute every RTT
RCP rate computation

\[ R(t) = R(t - d_0) + \frac{\alpha(C - y(t)) - \beta \frac{q(t)}{d_0}}{\hat{N}(t)} \]

Simpler than XCP:

- rates instead of windows
- thus, feedback doesn’t depend on a flow’s RTT
- thus, same feedback to everyone

(How can you estimate # flows?)
Estimating the number of flows

If guess is wrong, what happens?

- Queue builds up; will reduce rate in next round
- Possibly this estimator could be improved

\[ \hat{N}(t) = \frac{C}{R(t - d_0)} \]
RCP finishes flows quickly.

![Graph showing the comparison of flow completion times for different protocols. The x-axis represents flow size in packets, and the y-axis represents average flow completion time in seconds. The graph includes lines for XCP (avg.), TCP (avg.), RCP (avg.), Slow-Start, PS, and PS with XCP. The legend indicates that RCP finishes flows more quickly than TCP and XCP, and the maximum delay for RCP is smaller than for TCP and XCP. The graph also shows that the maximum delay for XCP and RCP is more than 30 times higher for flows around 2000 packets.]
Enforcing fairness and isolation

Based on slides by Ion Stoica
Problem: no isolation across flows

Assume router uses First In First Out (FIFO) queue

No protection: if a flow misbehaves it will hurt the other flows

Example: 1 UDP (10 Mbps) and 31 TCP’s sharing RED

![Throughput graph showing UDP flow misbehaving](image)
A first solution

**Round robin** among different flows [Nagle ’87]

- One queue per flow
- while (1) { send one packet from each queue }
Round robin discussion

Advantages: protection among flows

- Misbehaving flows will not affect the performance of well-behaving flows
- FIFO does not have such a property

Disadvantages:

- More complex than FIFO: per flow queue/state
- Biased toward large packets: a flow receives service proportional to the number of packets
Fair Queueing (FQ) [DKS’89]

Define a fluid flow system: a system in which flows are served continuously

- essentially, bit-by-bit round robin

Advantages

- Each flow will receive exactly its max-min fair rate
- ...and exactly its fair per-packet delay
- ...regardless of packet sizes
If link congested, compute $f$ such that

$$\sum_i \min(r_i, f) = C$$

$f = 4$:
- $\min(8, 4) = 4$
- $\min(6, 4) = 4$
- $\min(2, 4) = 2$
Implementing Fair Queueing

What we just saw was bit-by-bit round robin

But can’t interrupt transfer of a packet (why not?)

Idea: serve packets in the order in which they would have finished transmission in the fluid flow system

Strong guarantees: same as having a virtual link of the max-min fair capacity. Each flow gets:

- Exactly its max-min fair rate (+/- one packet size)
- Exactly its max-min fair per-packet delay (+/- one packet size) or better
Flow 1 (arrival traffic)

Flow 2 (arrival traffic)

Service in fluid flow system

Packet system
Recall: “serve packets in the order in which they would have finished transmission in the fluid flow system”

So, need to compute finish time of each packet in the fluid flow system

... but new packet arrival can change finish times of existing packets (perhaps all)!

Updating those times would be expensive

Solution: virtual time
Solution: Virtual Time

Key Observation: finish times may change when a new packet arrives, but the finish order doesn’t

• Only the order is important for scheduling

Solution: maintain the number of rounds needed to send the remaining bits of the packet

• New packet arrival doesn’t change # remaining rounds
• Does change rounds executed per unit time, but that’s ok

System virtual time = index of the final round in the bit-by-bit round robin scheme
System Virtual Time: $V(t)$

Measure service, instead of time

Slope of $V(t) = \text{rate at which every active flow receives service}$

- $C = \text{link capacity}$
- $N(t) = \text{number of active flows in fluid flow system at time } t$

\[
\frac{\partial V(t)}{\partial t} = \frac{C}{N(t)}
\]
Define

- \( F_i^k \) = virtual finishing time of packet \( k \) of flow \( i \)
- \( a_i^k \) = arrival time of packet \( k \) of flow \( i \)
- \( L_i^k \) = length of packet \( k \) of flow \( i \)

Virtual finishing time of packet \( k+1 \) of flow \( i \) is

\[
F_i^{k+1} = \max(V(a_i^k), F_i^k) + L_i^{k+1}
\]

Order packets by increasing virtual finishing time, and send them in that order.
Weighted Fair Queueing (WFQ)

What if we don't want exact fairness?

- Maybe web traffic is more important than file sharing

Assign weight $w_i$ to each flow $i$

And change virtual finishing time to

$$F_{i}^{k+1} = \max(V(a_{i}^{k}), F_{i}^{k}) + \frac{L_{i}^{k+1}}{w_{i}}$$
FQ does not eliminate congestion; it just manages the congestion

Provides isolation between flows

- complete isolation?

Still need both end-host and router-based congestion control

- End-host congestion control to adapt rate
- Router congestion control to protect/isolate
Rethinking “fairness”: Congestion pricing
The Internet routes money; packets are just a side effect.

– Unknown, via Dave Clark
What is “fair”?

Flow rate equality!

Easily circumvented

Doesn’t even optimize for any metric of interest

Fig. 1: Poppycock.
Plentiful: use as much as you want

- air
- advisor’s grant money

Scarce: pay for what you want

- price set by market
- result (under assumptions): socially optimal allocation

Fig. 2: Invisible hand of the market.
Briscoe’s main points

Flow rate fairness (FRF) is not useful

Cost fairness is useful

Flow rate fairness is hard to enforce

Cost fairness is feasible to enforce

Fig 3: Briscoe.
FRF not useful

Doesn’t equalize benefits

• e.g., SMS message vs. a packet of a video stream

Doesn’t equalize costs

• e.g., “parking lot” network: long flow causes significant congestion but is given equal rate by fair queueing

Therefore, doesn’t equalize cost or benefit
FRF not useful

Myopic: no notion of fairness across time

In summary, FRF does not optimize utility

- except for strange definitions of utility...

So, even cooperating entities should not use it!
Cost fairness is useful

Economic entities pay for the costs they incur

- This is “fair” (in a real-world sense), not “equal”—and that’s fine

In other words, networks charge packets for the congestion they cause

- Can networks lie about congestion?
- Yes. So it’s really a market price, not exactly congestion

Result: senders want to maximize utility

- Will balance benefit with cost (utility = benefit – cost)
Example: light & heavy traffic

[Briscoe 2009]

Key point: Benefit per bit is high for light flow and low for heavy flow.
Frank Kelly 1997: Cost fairness maximizes aggregate utility

i.e.: any different outcome results in suboptimal utility

Why won't anyone listen to Kelly? Hello?! ... where did everybody go?
Kelly’s model (one congested link)

Each user $i$ has utility $U_i(r_i)$ for rate $r_i$

Each user $i$ pays $p_i$ for access to link (its own choice)

Link sets price per unit bandwidth: $p = (\text{Sum } p_j) / C$

• thus, $r_i = p_i / p = C p_i / (\text{Sum } p_j)$

**Theorem:** assuming $U_i$ concave, strictly increasing, and continuously differentiable, then

• A competitive equilibrium exists: setting of $p_i$s in which no user can improve their utility given current price
• This equilibrium maximizes $\text{Sum } U_i(r_i)$
Run your flow longer

Create more flows (similar to sybil attack)

- Multiple TCP connections between same source/destination (web browsers)
- Spoof source IP / MAC address
- Multiple flows to other destinations (BitTorrent)
Cost fairness is enforceable

You send me a packet; I handle delivery and charge you for it

How much do I charge?

• Depends on cost on entire remainder of path!

Not the only way of arranging payments, but it is convenient

• payments are between neighbors that already have an economic relationship
Mechanism: Re-Feedback

Key property: every hop knows total congestion along downstream path

First packet

Second packet
Previous explanation was in terms of money, but doesn’t have to directly involve money

- Re-feedback is a mechanism
- Doesn’t imply a particular way of implementing congestion pricing

Possible variants of congestion pricing

- pay per packet?
- monthly allowance?
- only at edges?
- between all ISPs?
Discussion: What if...

Host running a persistent “light” job is interrupted by heavy flows congesting the net?

Host is compromised? (botnet) Who pays?

If we want cost fairness, is Weighted Fair Queueing useless?

- No: provides mechanism to isolate flows, virtualize links
- e.g., could use congestion pricing to set WFQ’s weights
“It just isn’t realistic to create a system the size of the Internet and define fairness within the system without reference to fairness outside the system.”

Cost fairness optimizes aggregate utility and is feasible to enforce

Flow rate fairness does not optimize utility and is not feasible to enforce

• Cease publication on the topic and stop teaching it in undergraduate courses
Announcements
Announcements

Wed: No class

Fri: Assignment 1 due
  • Accepting late submissions (-15%) till 11am Mon Feb 13
  • No credit thereafter

Mon: “Modern congestion control”
  • Mo Dong lectures

Wed: Project proposals due

Web page updates to come…
Project proposals due 11am Wednesday Feb 15

• Submit via email to Brighten
• 1/2 page, plaintext

Describe:

• the problem you plan to address
• what will be your first steps
• what is the most closely related work, and why it has not addressed your problem
  - at least 3 full academic paper citations (title, authors, publication venue, year) plus paper URLs
• if there are multiple people on your project team, who they are and how you plan to partition the work
Project proposals

Talk to us if...

- You need a project idea
- You’d like advice on a project idea
- You need partners
- You’re just a nice person and want to say hi

After submission

- Course staff will give feedback and approve or request changes
- Proposal is 5% of course grade

See also course syllabus