Promoting the Use of End-to-End Congestion Control in the Internet

IEEE/ACM Transactions on Networking, May 3 1999

Sally Floyd, Kevin Fall

Presenter:

Yixin Hua

About

Winner of the Communications Society William R. Bennett Prize Paper Award, 1999.

Revision of [FF98] Floyd, S., and Fall, K., Promoting the Use of End-to-End Congestion Control in the Internet February 1998, which is a revision of [FF97] Floyd, S., and Fall, K., Router Mechanisms to Support End-to-End Congestion Control. Technical report, February 1997.

Overview

- Introduction
- Existing problems
  - Unfairness
  - Congestion collapse
- Our approach
- Alternative approaches
- Conclusions and future work

Introduction

“To promote the inclusion of end-to-end congestion in the design of future protocols using best-effort traffic,…”

Why? Non-congestion-controlled best-effort traffic’s impacts range from extreme unfairness against competing TCP traffic to potential for congestion collapse. And routers need to **identify** and restrict those best-effort flows in times of congestion.

Bad guys

- **Not “TCP-friendly” flow**: Flow whose long-term arrival rate exceeds that of any conformant TCP in the same circumstances.
- **Unresponsive flow**: Flow that fails to reduce its offered load at a router in response to an increased packet drop rate.
- **Disproportionate-bandwidth**: Flow that uses considerably more bandwidth than other flows in a time of congestion.

Different approaches

- **Per-flow packet scheduling**

  Supply fairness with a cost of increased state, but provide no inherent incentive structure for best-effort flows to use end-to-end congestion control.

- **Incentives for end-to-end congestion control**

  Give a concrete incentive to end-users, developers, and protocol designers to use end-to-end congestion control for best-effort traffic. RLM? (This is the paper about.)

- **Pricing mechanism**

  Result in a risky gamble that network providers will provide additional bandwidth and deploy effective pricing structures fast enough to keep up the growth in unresponsive best-effort traffic in the Internet.

Existing problems

- Unfairness

  Unfairness is from bandwidth starvation that unresponsive flows can inflict on well-behaved responsive traffic.

- Danger of congestion collapse

  It stems from a network busy transmitting packets that will simply be discarded before reaching their final destinations.

Samples – unfairness

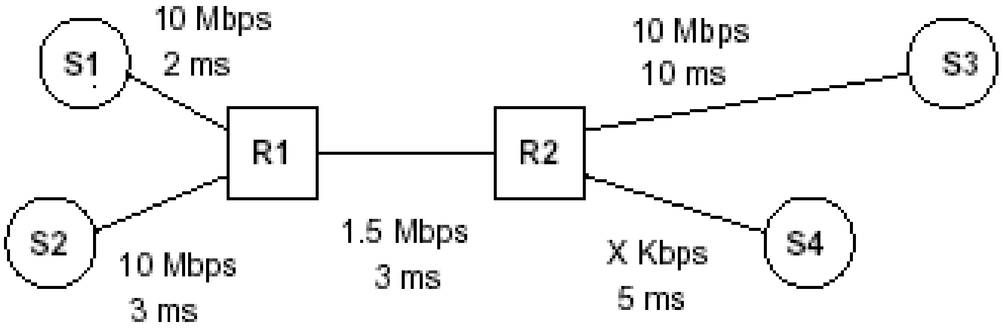
TCP (Good citizen) vs. UDP (Bad guy)

Setup:

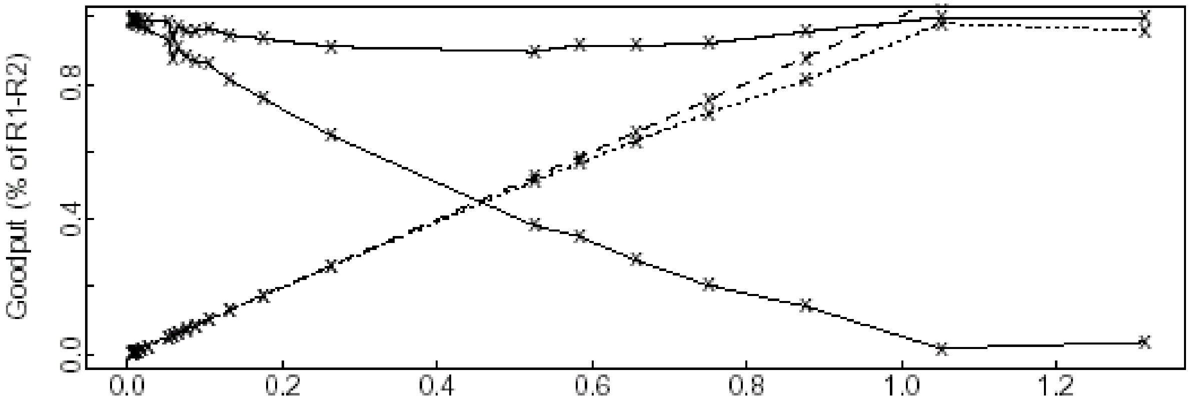
R2-S4 : 10 Mbps

TCP : S1 to S3

UDP : S2 to S4 ranges up to 2Mbps



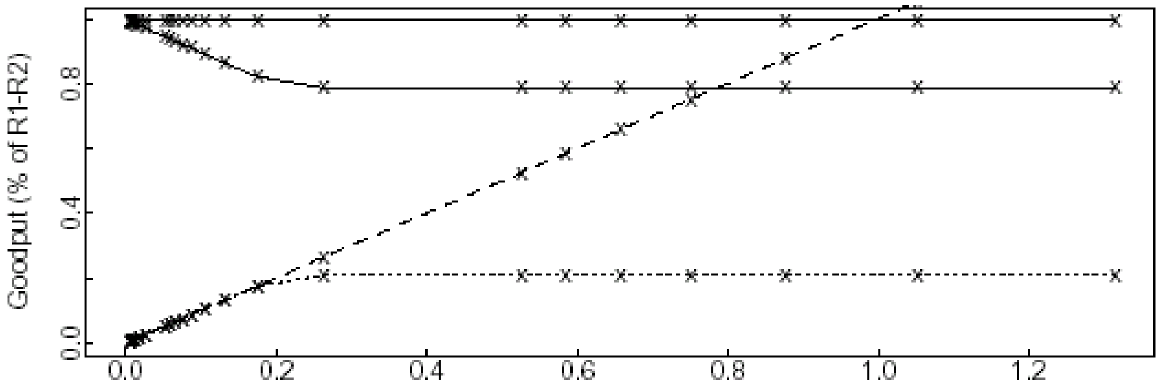
Samples – FCFS



X-axis: UDP Arrival Rate (% of R1-R2). Dashed Line: UDP Arrivals; Dotted Line: UDP Goodput; Solid Line: TCP Goodput; Bold line: Aggregate Goodput

Figure 2: Simulations showing extreme unfairness with three TCP flows and one UDP flow, and FCFS scheduling.

Samples – WRR



X-axis: UDP Arrival Rate (% of R1-R2). Dashed Line: UDP Arrivals; Dotted Line: UDP Goodput;
Solid Line: TCP Goodput; Bold line: Aggregate Goodput

Figure 3: Simulations with three TCP flows and one UDP flow, with WRR scheduling. There is no unfairness.

Congestion collapse

- Classic congestion collapse
- Congestion collapse from undelivered packets
- Fragmentation-based congestion collapse
- Congestion collapse from increased control traffic
- Congestion collapse from stale packets

Congestion collapse 1

- **Classic congestion collapse.**

  A stable condition that can result in throughput that is a small fraction of normal[Nag 84]. And it has been corrected by the timer improvements and congestion control mechanisms in modern implementations of TCP [Jac88].

- **Congestion collapse from undelivered packets.**

  It arises when bandwidth is wasted by delivering packets through the network that are dropped before reaching their ultimate destination. And the danger is due to the deployment of open-look application not using end-to-end congestion control. It is not stable, and returns to normal once the load is reduced.

Congestion collapse 2

- **Fragmentation-based congestion collapse.**

  It consists of the network transmitting fragments or cells of packets that will be discarded at receiver because they cannot be reassembled into a valid packet. Early Packet Discard [RF95] and Path MTU discovery [KMMP88] address to this problem.

- **Congestion collapse from increased control traffic.**

  Due to increasing load and increasing congestion, an increasing-large fraction of the bytes transmitted on congested links belong to control traffic, and an increasingly-small fraction of bytes transmitted correspond to data actually delivered to network application.

- **Congestion collapse from stale and unwanted packets.**

  It happens when congested link carrying packets user no longer wanted.

Samples – Congestion Collapse

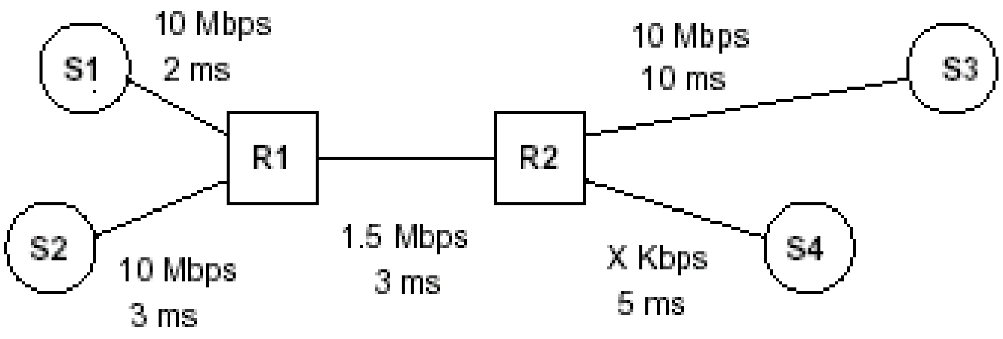
TCP (Good citizen) vs. UDP (Bad guy)

ns-2 setup:

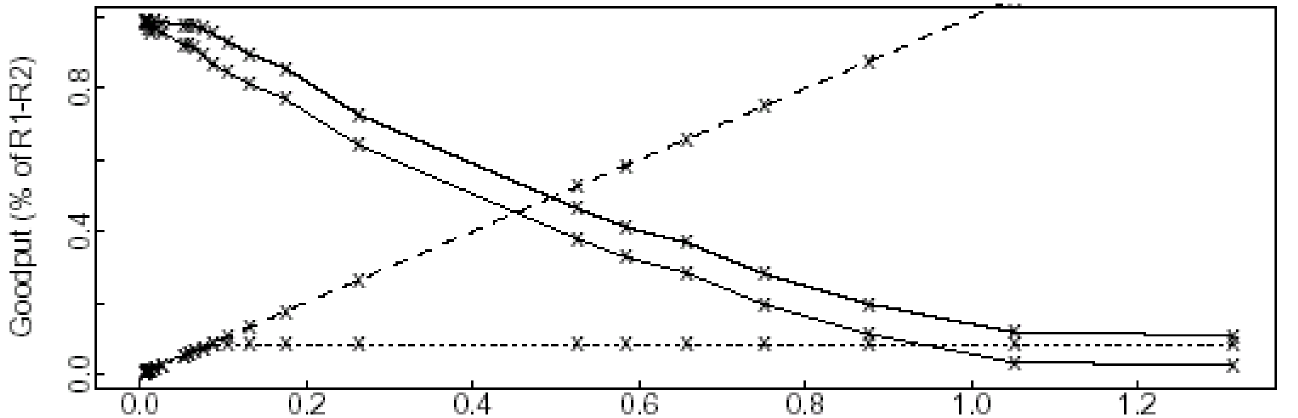
R2-S4 : 128Kbps

TCP : S1 to S3

UDP : S2 to S4 ranges up to 2Mbps



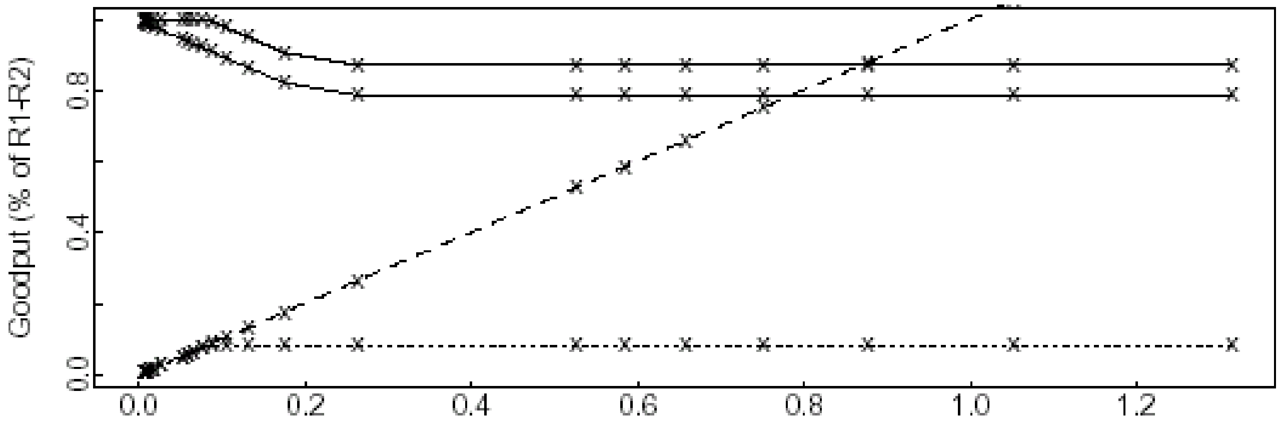
Samples – FCFS 1



X-axis: UDP Arrival Rate (% of R1-R2). Dashed Line: UDP Arrivals; Dotted Line: UDP Goodput; Solid Line: TCP Goodput; Bold line: Aggregate Goodput

Figure 4: Simulations showing congestion collapse with three TCP flows and one UDP flow, with FCFS scheduling.

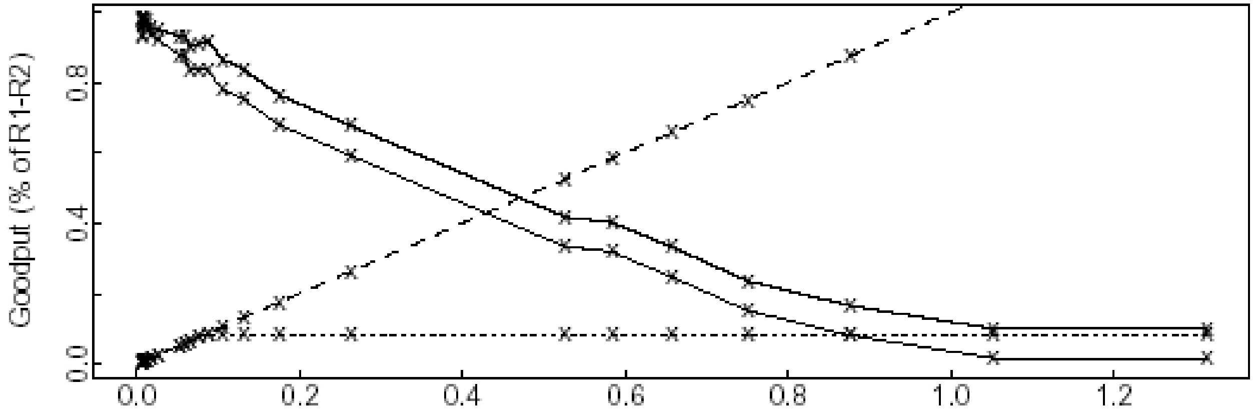
Samples – WRR 2



X-axis: UDP Arrival Rate (% of R1-R2). Dashed Line: UDP Arrivals; Dotted Line: UDP Goodput; Solid Line: TCP Goodput; Bold line: Aggregate Goodput

Figure 5: Simulations with three TCP flows and one UDP flow, with WRR scheduling. There is no congestion collapse.

Samples – FCFS 2



X-axis: UDP Arrival Rate (% of R1-R2). Dashed Line: UDP Arrivals; Dotted Line: UDP Goodput; Solid Line: TCP Goodput; Bold line: Aggregate Goodput

Figure 6: Simulations with one TCP flow and three UDP flows, showing congestion collapse with FIFO scheduling.

Samples – WRR 2

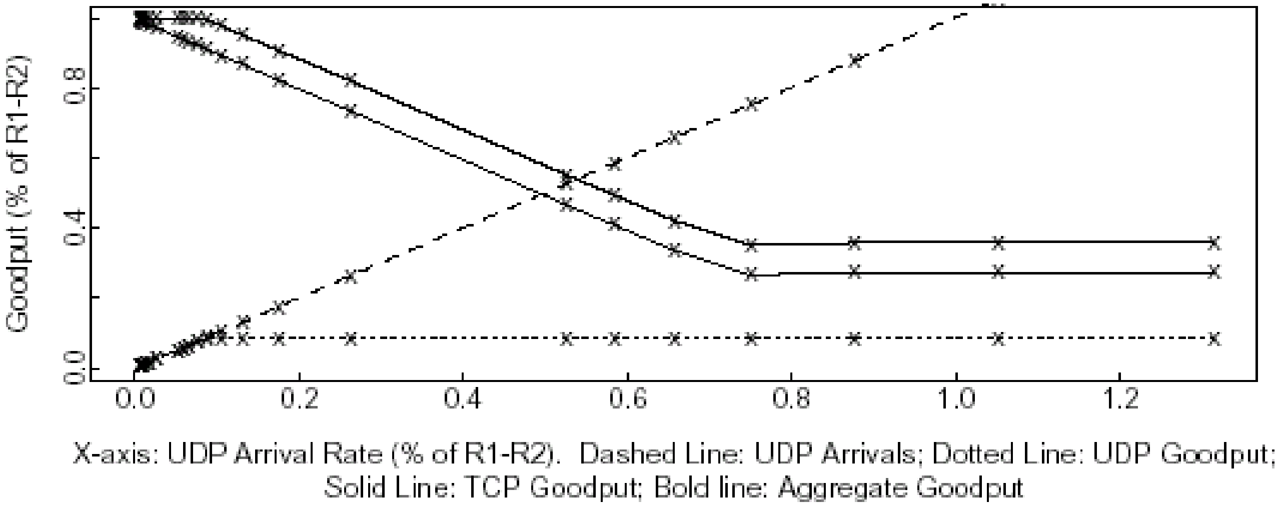


Figure 7: Simulations with one TCP flow and three UDP flows, showing congestion collapse with WRR scheduling.

Samples – WRR 3

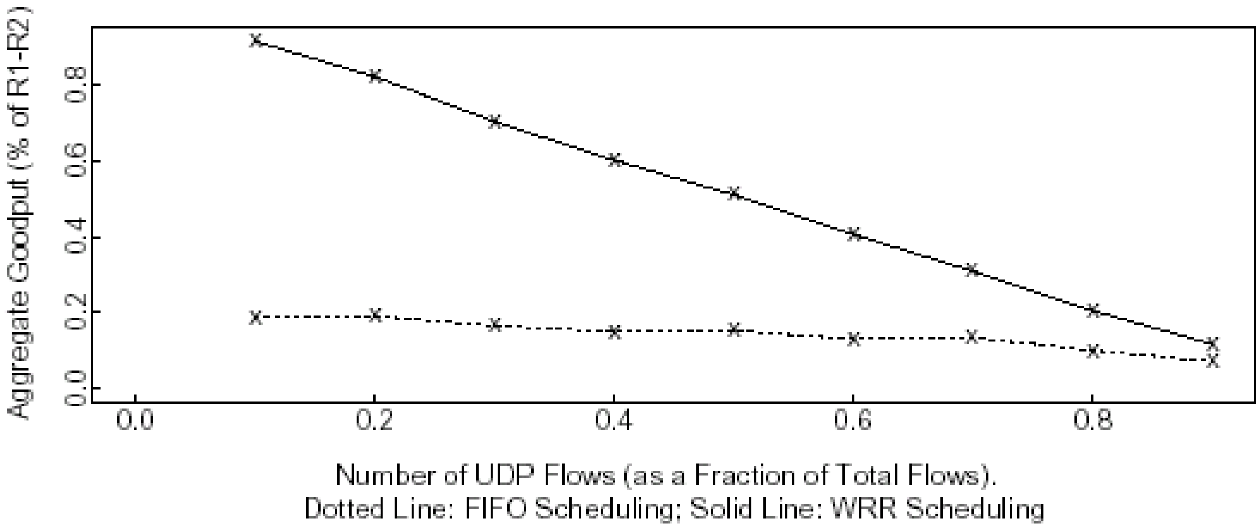
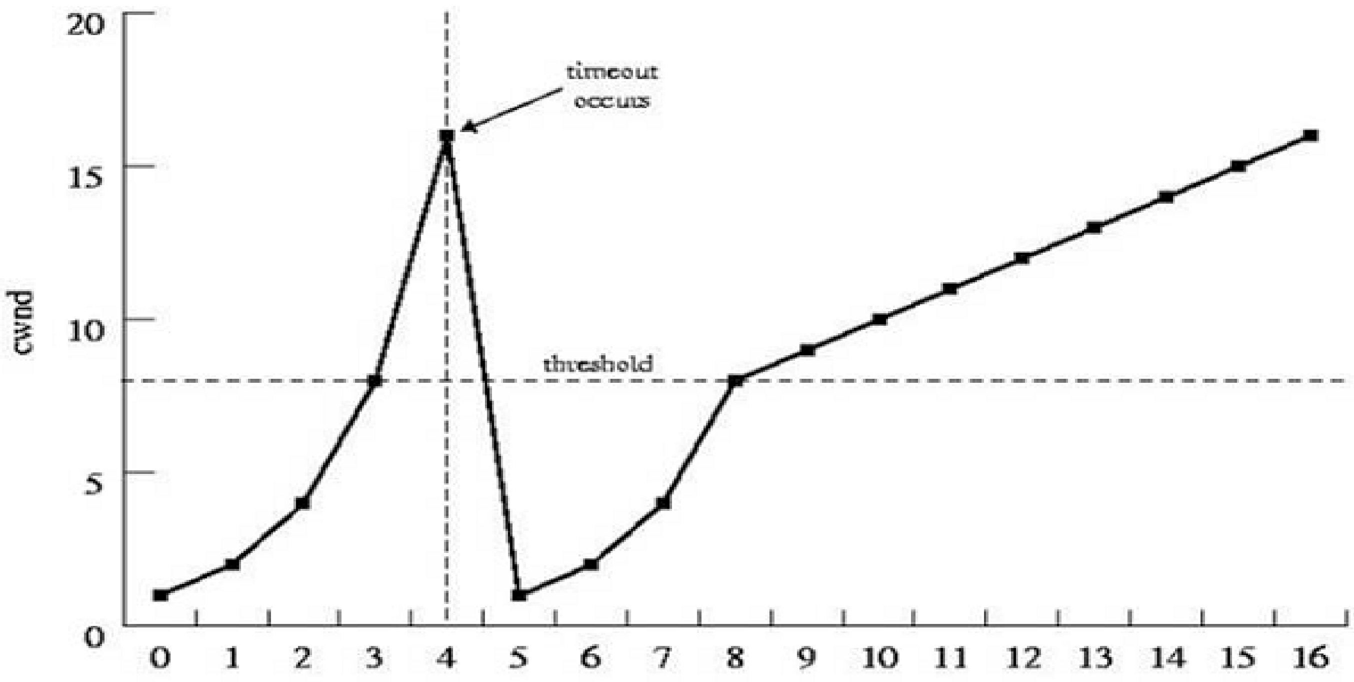


Figure 8: Congestion collapse as the number of UDP flows increases.

Our approach?

- Identify not TCP-friendly flows
- Identify unresponsive flows
- Identify flows with disproportional bandwidth usage
- And Restrict them…

TCP


cwnd
20
15
10
5
0
timeout
occurs
threshold
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Identify not TCP-friendly flows

- Rationale: TCP-friendly flow’s response to congestions

  Maximum sending rate T Bps

$$T \leq \frac{1.5\sqrt{2/3} * B}{R * \sqrt{p}},$$

  Packet drop rate of p

  Packets size of B bytes

  Fair constant roundtrip time R second, including queueing delays

Identify not TCP-friendly flows 1

- Limitation:
  - Per flow basis.
  - Also detects flow with larger packet size or smaller round trip time. Router need to detect packet size, R can be set as twice the propagation delay from attached link.
  - It can be only applied for non-bursty packet drop behavior.
- Response:
  - Restrict not TCP-friendly flow in congestion.
  - Remove restriction when there is no longer any significant link congestion, or flow has reduced its arrival rate appropriately in response to congestion.

Identify not TCP-friendly flows 2

- Example test:

  Test flow with $1.45B/(R\sqrt{p})$, B is the maximum packet size, R twice the T_{prop}, and p is the aggregate packet drop rate for that queue.

  A flow restriction is removed if the arrival rate return less than to $1.22B/(R\sqrt{p})$, for the new packet drop rate p.

Identify unresponsive flows

- Rationale: Arrival rate decreases appropriately in response to increased packet drop.

  If the steady state drop rate increases by a factor of x, and the presented load for a high-bandwidth flow does not decrease by a factor reasonably close to sqrt(x) or more, then the flow can be deemed no to be using congestion control.

Identify unresponsive flows 1

- Limitation:
  - Per flow basis
  - Only apply to high-bandwidth flows.
  - When the packet drop rate remains relatively constant, no flow will be identified as unresponsive.
  - Less straightforward for a flow with a variable demand.
- Response:
  - Restrict unresponsive in congestion.
  - Remove restriction when there is no longer any significant link congestion, or flow has reduced its arrival rate appropriately in response to congestion.

Identify unresponsive flows 2

- Example test:

  If drop rate increases by more than a factor of 4, but flow’s arrival rate has not decreased to below 90% of its previous value.

  Restriction will be removed from an unresponsive flow only if, after an increased packet drop rate, its arrival rate returns to at most half of its arrival rate when it was restricted.

Identify flows using disproportionate bandwidth

- Rationale: How many bandwidth share is used in congestion

  First, check if the arrival rate is greater than ln(3n)/n. (It is close to 1 for n = 2, and grows slowly as a multiple of 1/n.)

  Second, check if the arrival rate is greater than c/sqrt(p) Bps for some constant c. c is close to $1.5\sqrt{2/3}B/R$.

Identify flows using disproportionate bandwidth 1

- Limitation:
  - Gauging the level of unsatisfied demand is problematic.
  - For long RTT TCP flow with persistent demand, a single packet drop represents a significant suppressed demand. But to a short bursty web traffic, it doesn't mean much in term of unsatisfied demand.
- Response:
  - A conservative approach would be to limit the restriction of a high-bandwidth responsive flow so that over the long run, each such flow receives as much bandwidth as the highest-bandwidth unrestricted flow.
  - Restriction is removed when any one of test conditions is no longer true.

Identify flows using disproportionate bandwidth 2

- Example test:

  Check if estimated arrival rate is greater than $12,000/\sqrt{p}$,

  and the arrival rate is also greater than a fraction of ln(3n)/n of the best-effort bandwidth.

Alternative approaches

- Per-flow scheduling

  Provide fairness at router point.

  Encourage flows to make sure their queue in the congested router never goes empty.

- **FCFS scheduling**

  More efficient to implement.

  Reduces the tail of delay distribution.

  Allows bursty transmitted in a bursty way instead of having packets "spread out" and be delayed by scheduler.

Alternative approaches 1

- **Other scheduling algorithms**

  Such as differential treatment or preferential dropping of unresponsive flows, relaxed variants of per-flow scheduling, differential dropping for flows using disproportionate bandwidth, Class Based Queueing, stochastic Fair Queueing, min-max fairness restriction.

- Pricing mechanism

  State required for this would be too complex.

Conclusion and future work

- Need specific proposal for identifying and controlling unresponsive flows.
- Deployment of these mechanism is more meaningful than the accuracy.

Appendix

- **One TCP connection or many?**

  Break one TCP connection into multiple connections to increase throughput. Every single packet drop cause one of N connection window cut in half, that's 1/(2N) of aggregate arrival rate.

  By identifying "flow" as source and destination IP addresses ONLY, multiple connections combine into one flow, and defeat the abuse.
- **Characterizing TCP-friendly flows.**
- **Simulations verifying the "TCP-friendly" characterization.**