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

Sally Floyd and Kevin Fall

IEEE/ACM Transactions on Networking

May 1999

Outline

- The problem of Unresponsive Flows
  - Fairness problems
  - The danger of congestion collapse
  - Forms of congestion collapse
- The solution: regulating unresponsive flows at the router
  - TCP-friendly flows
  - classifying flows
- Alternative approaches

Approaches for controlling best-effort Internet traffic

- Deploying per-flow scheduling mechanisms to approximate max-min fairness.
- Use end-to-end congestion control with *incentives*
- Rely on pricing mechanisms to control sharing

The problem of Unresponsive Flows

- *Unresponsive flows* do not use end-to-end congestion control and do not reduce their load on the network in response to packet drops.
- Unresponsive behavior causes:
  - unfairness
  - congestion collapse

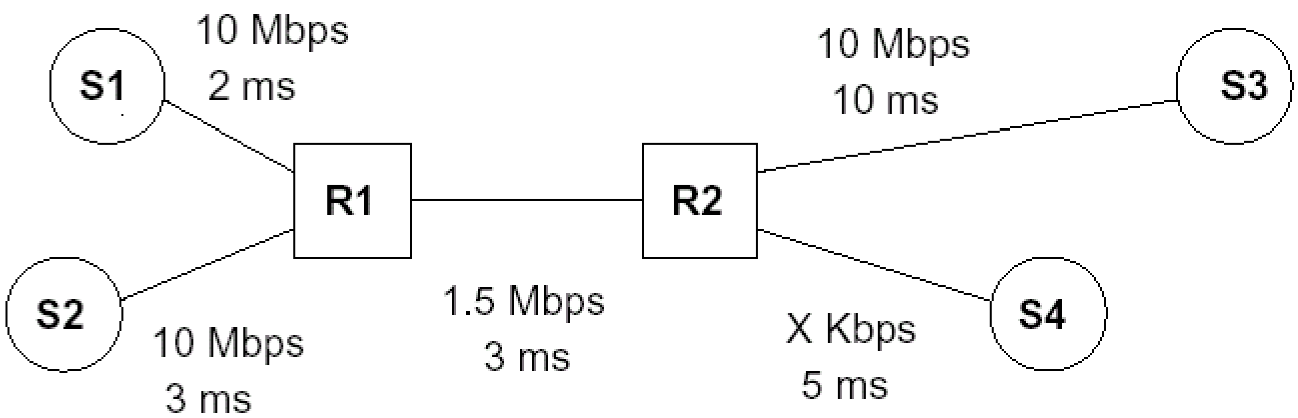

Figure 1: Simulation network.

Fig 1,2,3

Unfairness

- Unresponsive flows can cause bandwidth starvation of *well-behaved* responsive traffic.
- TCP flows competing with unresponsive UDP flows for scarce bandwidth
  - When TCP flows reduce their sending rates in response to congestion indicators, uncooperative UDP flows will capture more of the available bandwidth.

Definitions

- goodput = the capacity delivered to the receiver, excluding duplicate packets
- robust senders
  – send large packets
  – small roundtrip time
  – OR large sender window {helps with fast retransmit}

Definitions

- Fragile senders
  - Large RTT
  - Small congestion window

simple results to remember

- With TCP congestion control, throughput varies *inversely* with connection's roundtrip time.
- With multiple <u>congested</u> gateways, throughput varies as the *inverse* of the square root of the number of congested gateways.
- per-flow scheduling can control the allocation among a set of competing flows.

Congestion Collapse

- occurs when an increase in network load results in a decrease in the useful work done by the network.
- *classical congestion collapse*
- *congestion collapse from undelivered packets*
- *fragmentation-based congestion collapse*
- *congestion collapse from increased control traffic*
- *congestion collapse from stale packets*

Classical Congestion Collapse

- *classical congestion collapse* - is due to unnecessary retransmission of packets
  - – this is a stable condition that can result in throughput that is a small fraction of normal
  - – corrected by Jacobson’s mechanisms

Congestion Collapse from Undelivered Packets

- wasted bandwidth due to pushing packets through the network that are dropped before reaching their destination.
  - author's claim: biggest problem today because of *open-loop applications* not using end-to-end congestion control.
  - not stable: returns to normal when load is reduced

Fig 4-7

Congestion Collapse from Undelivered Packets

- per-flow mechanisms at the router (in Figure 7) cannot guarantee elimination of this form of congestion control.
- Figure 8 shows the limiting case where a very large number of very small bandwidth flows without congestion control threaten congestion collapse in a highly-congested network regardless of scheduling discipline at the router.
- **key claim: essential factor is the absence of end-to-end congestion control for UDP traffic.**

Fragmentation-based Congestion Collapse

- caused by transmitting cells or fragments that will be discarded because they cannot be reassembled
- some fragments are discarded while other fragments are delivered thus wasting capacity
- fixes involve network layer knowledge being given to data link layer, e.g.
  - Early Packet Discard in ATM switches
  - path MTU discovery to minimize packet fragmentation

Congestion Collapse from Increased Control Traffic

- an increasingly large fraction of bytes transmitted belonging to control traffic
  - packet headers
  - routing updates
  - multicast join and prune messages {e.g. RLM}
  - DNS messages

Congestion Collapse from Stale Packets or Unwanted Packets

- occurs when congested links carry packets no longer wanted by the user.
  - when data transfers take too long due to queues are too long {e.g. audio or video jitter}
  - when Web sites unnecessarily *push* Web data that was never requested.

Philosophy of Cooperation

- authors believe cooperating flows can coexist if the right incentives are put in place for the competing flows
- paper explores mechanisms that could be deployed to provide incentives for flows to participate in cooperative methods for congestion control.

Classification of Flows

- a flow is defined on the granularity of source and destination IP addresses and port number {each TCP connection is a flow}
- router should regulate flows classified as:
  - unresponsive flows
  - not TCP-friendly flows
  - disproportionate-bandwidth flows

TCP-friendly flows

- A flow is ***TCP-friendly*** if the flow's arrival rate does not exceed the bandwidth of a conformant TCP connection in the same circumstances.
- **major assumption:** TCP is characterized by reducing its congestion window at least by half upon receiving congestion indications and of increasing its congestion window by a constant rate of at most one packet per roundtrip time otherwise AIMD assumption.

TCP-friendly test

- Given a non-bursty packet drop rate of **p** , the maximum sending rate for a TCP connection is **T** bytes/sec., where

$$T \le \frac{1.5 \text{ sqroot } (2/3) * B}{R * \text{ sqroot } (p)}$$

for a TCP connection sending packets of size **B** bytes with a fairly constant roundtrip time (including queuing delays) of **R** seconds.

TCP friendly test

- The test is only applied at level of granularity of a TCP connection.
- An actual TCP flow will generally use less than maximum bandwidth, T.
- Philosophy says it is reasonable for a router to restrict bandwidth of any flow with arrival rate higher than that of any conformant TCP implementation. Is it reasonable??

TCP friendly test

- The measurements should be taken over a sufficiently large time interval (several RTTs).
- The test only applies for non-bursty packet drop behavior. *Blatant commercial for RED?*
- Robust flows may avoid detection, specifically flows with small roundtrip times.

Identifying Unresponsive Flows

- TCP-friendly test is of limited usefulness for routers unable to assume strong bounds on TCP packet sizes and roundtrip times.

A more general test :: verify that a high-bandwidth flow was *responsive*, i.e, its arrival rate decreases appropriately in response to increased packet drop rate.

Identifying Unresponsive Flows

- **Possible unresponsive flow test::** If the steady state drop rate increases by a factor *x* and the presented load for a high-bandwidth flow does not decrease by a factor close to *sqroot (x)* or more, the flow can be deemed *unresponsive*.
- This test needs an estimate of flow's arrival rate (e.g. CSFQ) and packet drop rate over several long intervals.

Unresponsive flows are stealing bandwidth from responsive TCP-friendly flows!

Identifying Disproportionate Bandwidth Flows

- a *disproportionate share* of bandwidth is a significantly larger share than other flows in the presence of suppressed demand from some of the other flows.
- This test could restrict conformant TCP flows (i.e., robust TCP flows).
- A flow is using a disproportionate share of best-effort bandwidth if its fraction of the aggregate arrival rate is more than *log (3n)/n* {natural log} where *n* is the number of flows with packet drops in the recent reporting interval.

Identifying Disproportionate Bandwidth Flows

- They define a flow as having a high arrival rate *relative to the level of congestion* if its arrival rate is greater than ***c/ sqroot(p)*** for some constant **c.**
- Example settings using results from appendix:

with B = 512 bytes and R = 0.05 seconds, c is set to 12,000.

Disproportionate Bandwidth Test [Example]

- A best-effort flow has disproportionate bandwidth if:

estimated arrival rate > 12000/ sqrt (p)

and

estimated arrival rate > log(3n)/n of the best-effort bandwidth.

Alternative Approaches

- per-flow scheduling mechanisms (RR, FQ) to isolate flows
  - Authors claim - incentives are backwards here.
- discusses FIFO and suggests middle ground of Class-Based Queueing (CBQ) or Stochastic Fair Queueing (SFQ)
- Authors question *min-max fairness* and suggests considering the number of congested links on flow path.

Conclusions

- Mechanisms for detecting and restricting unresponsive flows are needed.
- TCP-friendly is the right philosophy, i.e., peaceful coexistence of distinct flow classes.
- These mechanisms would provide an incentive in support of end-to-end congestion control.