Coupled congestion control for RTP media

draft-welzl-rmcat-coupled-cc-03

Michael Welzl, Safiqul Islam, Stein Gjessing

REDUCING INTERNET TRANSPORT LATENCY

RMCAT
90th IETF Meeting
Toronto, CA
24 July 2014

Context, background

- Having multiple congestion controlled flows from the same sender compete on the same bottleneck is detrimental
  - By first combining their congestion controllers in the sender, we can better control fairness (with priorities) and get less delay and loss
  - Ideally, want to be like one flow

- Two elements: 1) shared bottleneck detection (sbd), 2) coupled congestion control
  - In rtcweb, 1) can sometimes be very easy: same 6-tuple. But measurement-based sbd enables broader application of 2) (same sender, different receivers)

Context, background /2

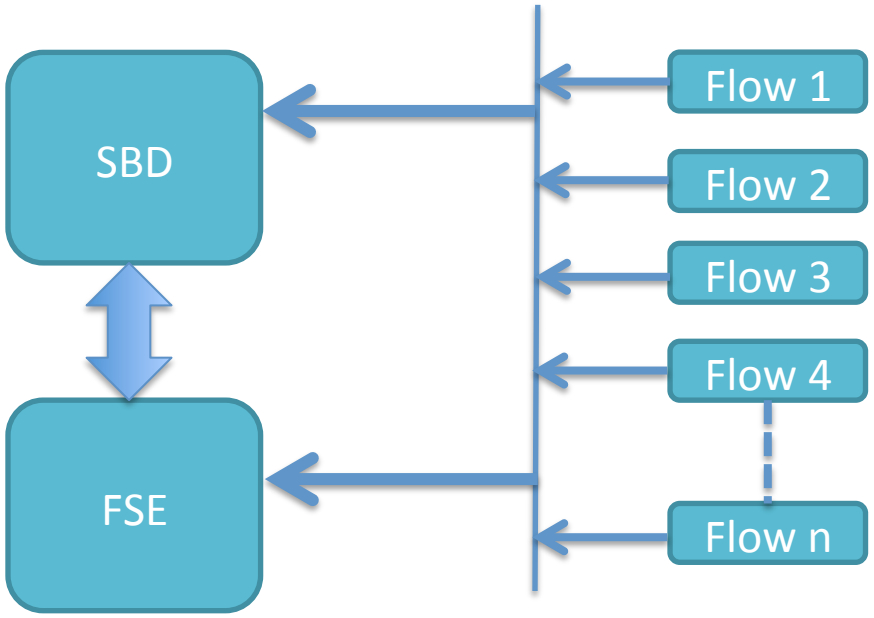
- When possible, 2) is best done by scheduling packet transmission from different sources with a single congestion controller
  - draft-johansson-rmcat-scream-cc-00.txt
  - Congestion Manager (CM)

- Disadvantages:
  - RMCAT is about RTP-based applications in general, not only rtcweb; combining multiple applications is very hard
  - could be difficult to switch on/off

- Hence, goal of draft-welzl-rmcat-coupled-cc: achieve benefits with minimal changes to existing congestion controllers
  - For SBD, only describe 6-tuple based method in the document; have separate document for measurement-based SBD

Prior draft versions

- To enable loose coupling of multiple apps, started with passive FSE
  - didn't work well
- Then, active FSE
  - simple: just give each flow a priority-weighted Nth of the total rate
  - also didn't work well



for all flows i in FG do
FSE_R(i) = (P(i)*S_CR)/S_P
send FSE_R(i) to the flow I
end for

- Here, "didn't work well" means: fairness/prioritization worked just fine, but we did not get reduced delay and loss

What's new in -03?

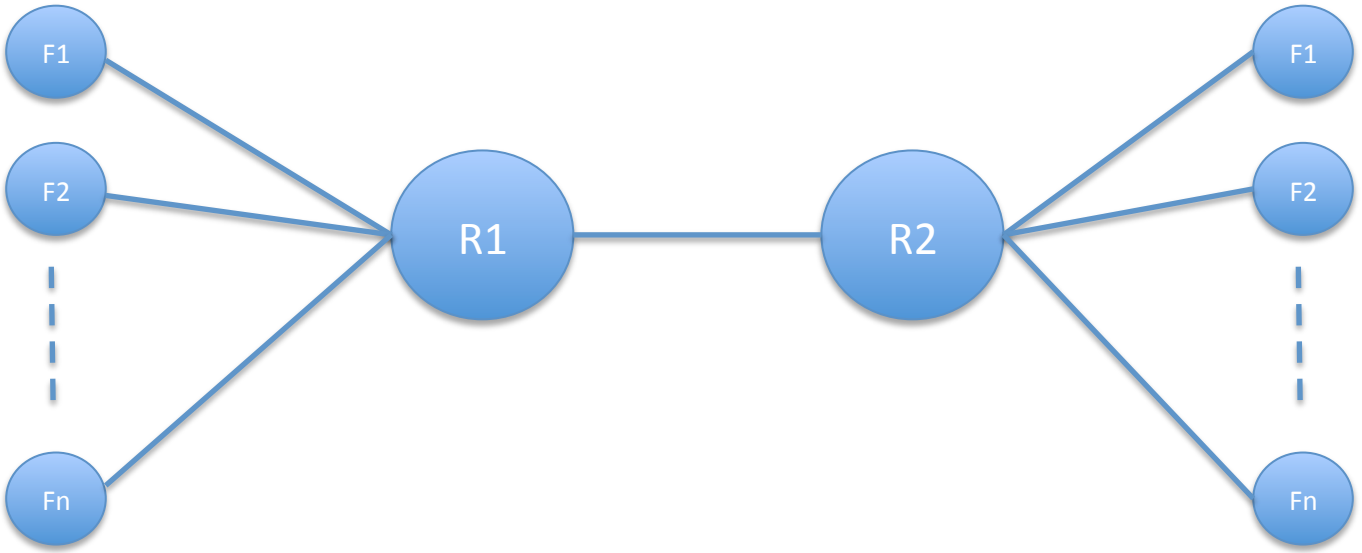
- Success! with a new algorithm
  - Tested in simulations with RAP (rate-based AIMD cc.) and TFRC (well-known rate-based media cc
    - Next steps: simulate with LEDBAT and combinations of different cc's, then real-life test in Chromium
  - Results documented in:
    - Safiqul Islam, Michael Welzl, Stein Gjessing, Naeem Khademi: "Coupled Congestion Control for RTP Media", accepted for publication, ACM SIGCOMM Capacity Sharing Workshop (CSWS 2014), 18 August 2014, Chicago, USA.
    - Safiqul Islam, Michael Welzl, Stein Gjessing, Naeem Khademi: "Coupled Congestion Control for RTP Media", University of Oslo Department of Informatics technical report 440, May 2014.
- Papers, code etc. available via: http://heim.ifi.uio.no/safiquli/coupled-cc/index.html

The Conservative Active FSE algorithm

- No congestion (flow wants to increase): do as before
- Congestion (flow wants to decrease): proportionally reduce total rate to better emulate the behavior of one flow
  - e.g. flow 1 goes from 1 to ½ => total goes from X to X/2

- To prevent flows from either ignoring congestion or overreacting, a timer keeps them from changing their rates immediately after the common rate reduction that follows a congestion event
  - Timer is set to 2 RTTs of the flow that experienced congestion
  - Reasoning: assume that a congestion event can persist for up to one RTT of that flow, with another RTT added to compensate for fluctuations in the measured RTT value

Some simulation results



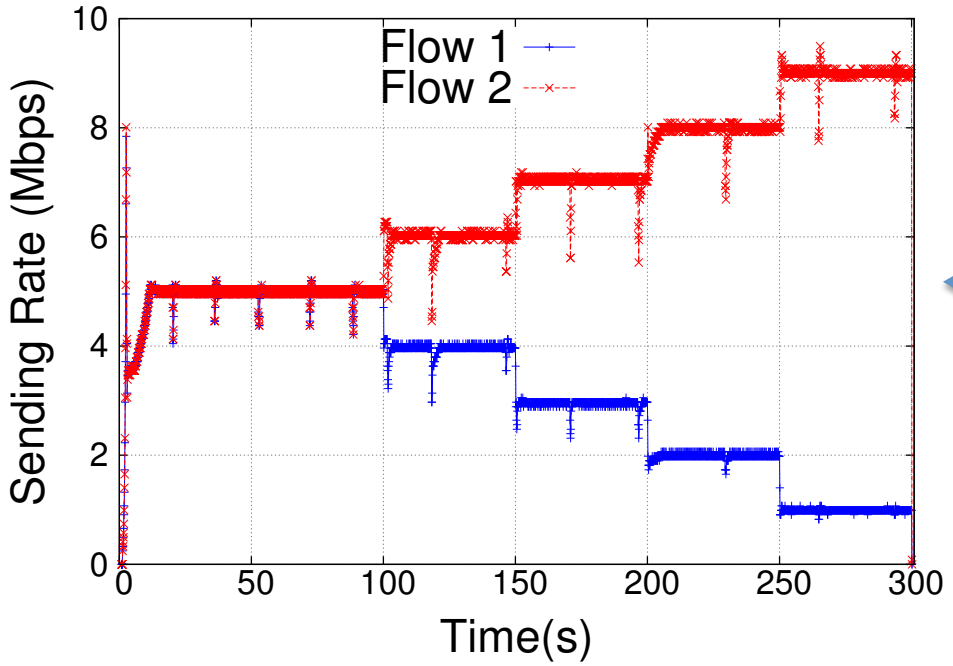
- Bottleneck – 10 Mbps
- Queue-length – 62 Packets (1/2 BDP)
- Packet Size – 1000 Bytes
- RTT – 100 ms
- All tests (except when x-axis = time) ran for 300 seconds, carried out 10 times with random start times picked from first second;
  stddev consistently very small (<= 0.2%)

Fairness / prioritization (TFRC)

Charter:

"Develop a mechanism for identifying shared bottlenecks between groups of flows, and means to flexibly allocate their rates within the aggregate hitting the shared bottleneck."

(requirement F34 in *draft-ietf-rtcweb-use-cases-and-requirements-12*)

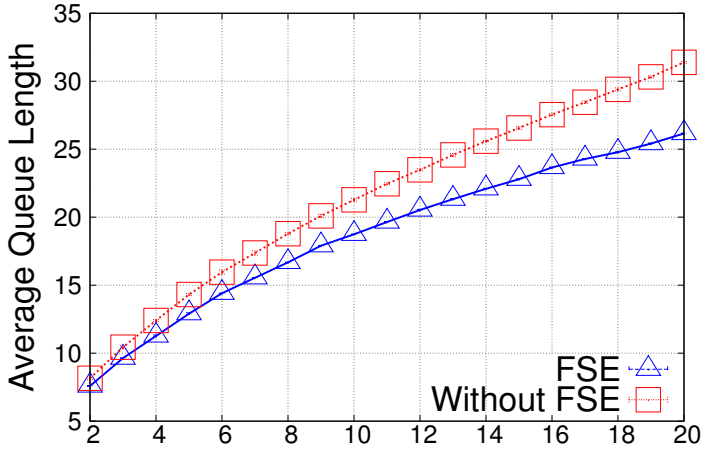


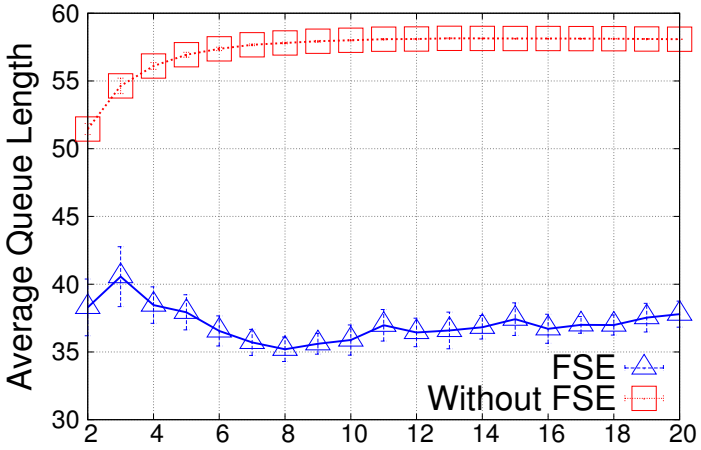
Priority of flow 1 increased over time

RAP

TFRC

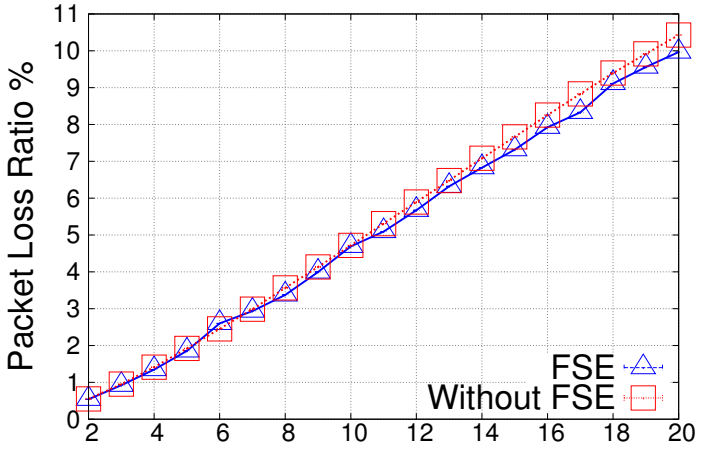
Average Queue

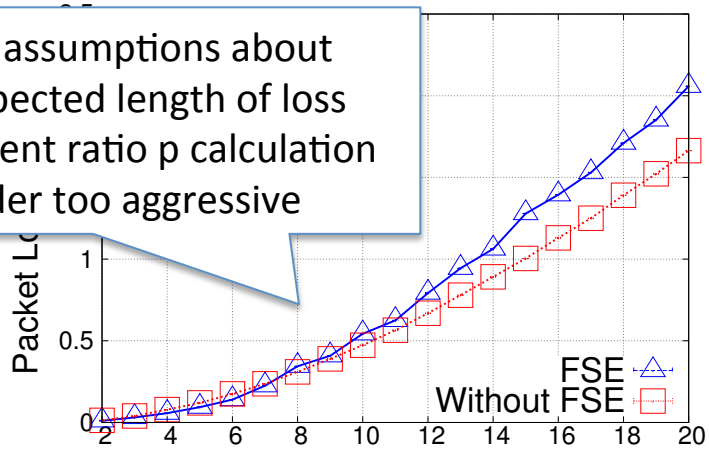




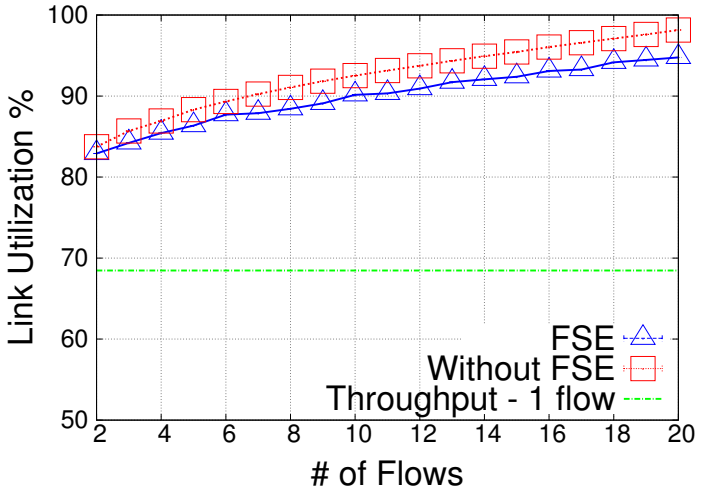
Receiver makes assumptions about sending rate (expected length of loss interval) ➔ loss event ratio p calculation wrong ➔ sender too aggressive

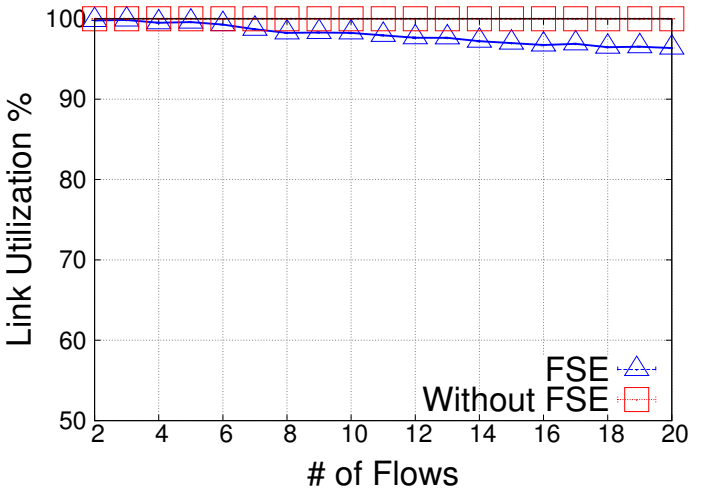
Packet Loss Ratio





Link Utilization



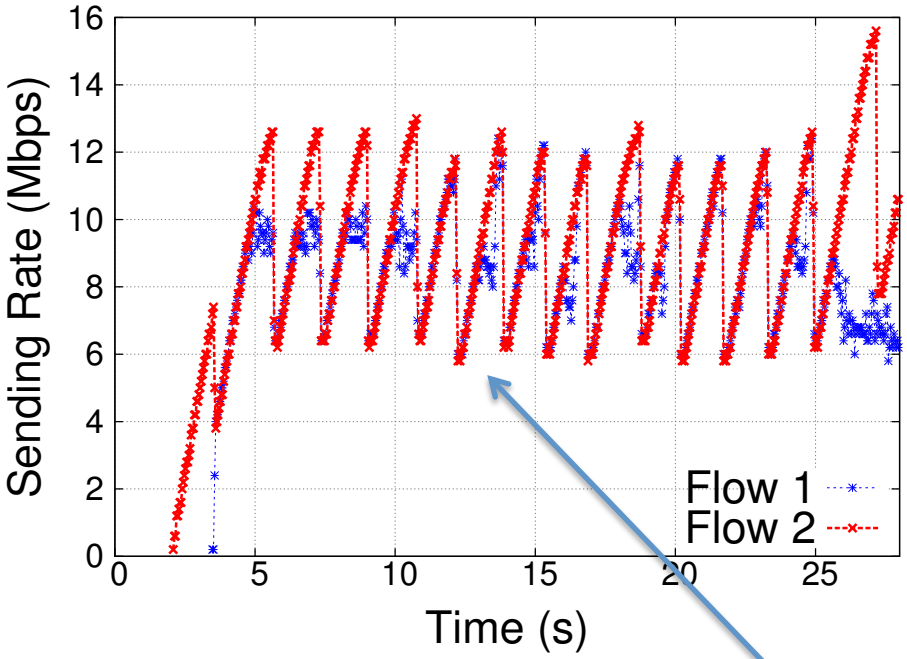


How to evaluate app-limited flows?

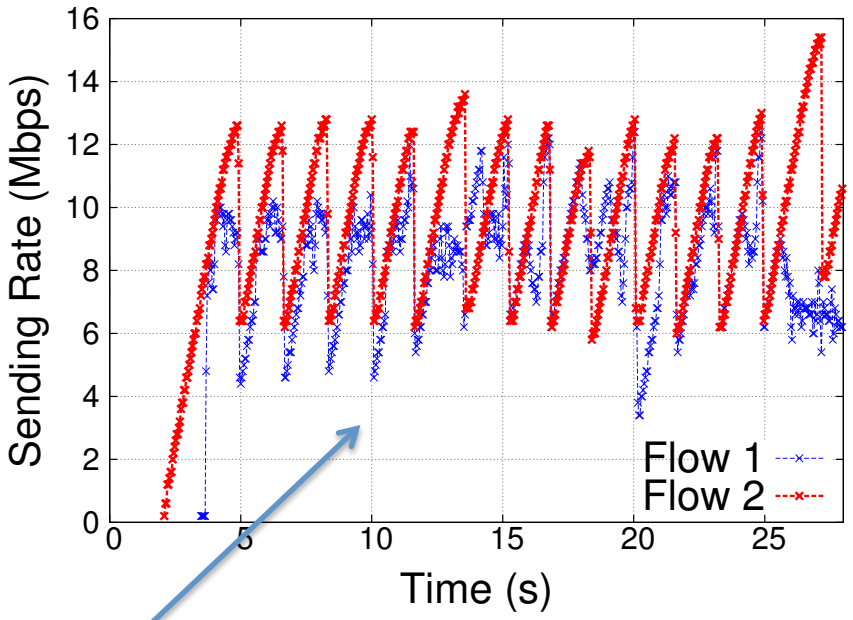
- Not easy: who is in control?
- RMCAT codec model not available yet
- From a transport point of view, the send buffer can either run empty or not, with variations in how quickly changes between these two states occur
  - We used a non-reacting video trace of a person talking in a video conference with a well-known H264 encoder (X264) to steer the app sending rate
    - I-frame in the beginning, rest was mostly P-frames

1 app-limited flow, 1 greedy flow (RAP)

FSE

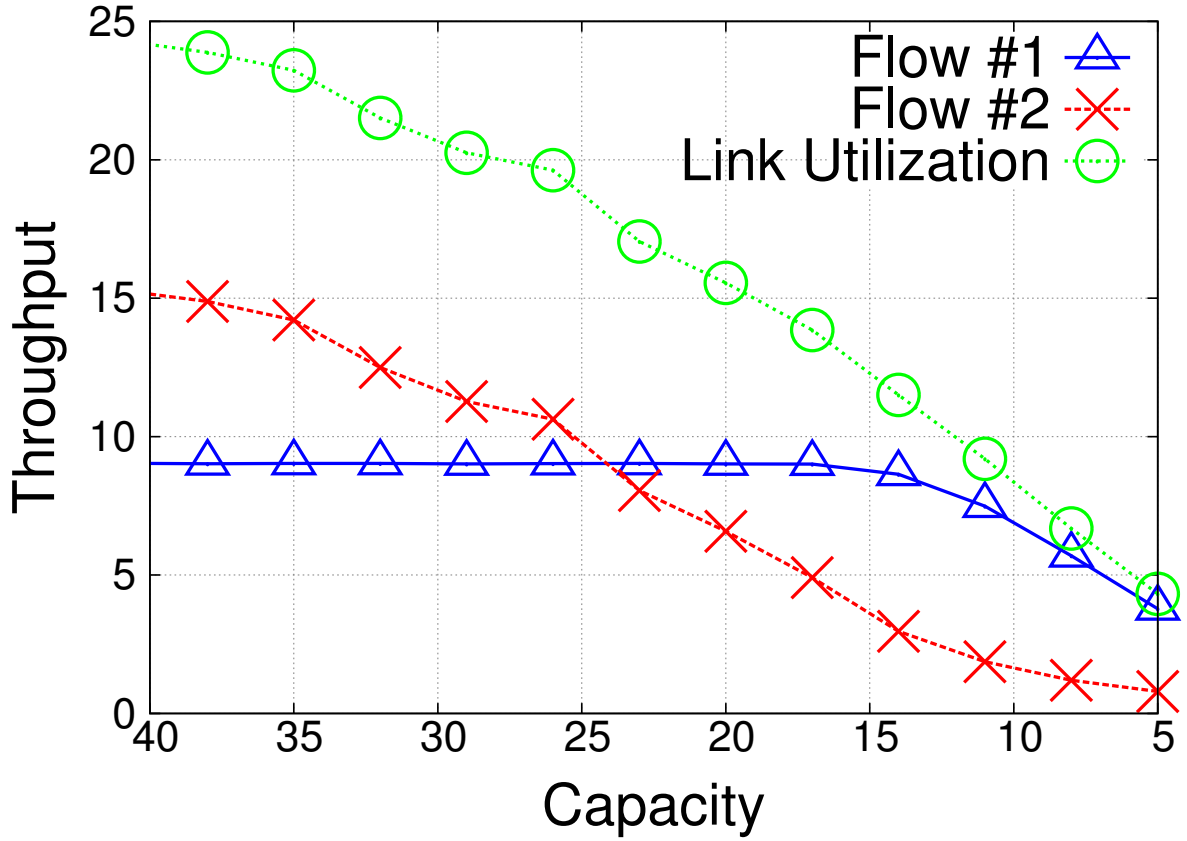


Without FSE



FSE-controlled flows proportionally reduce the rate in case of congestion; without FSE, synchronization causes app-limited flow to over-react

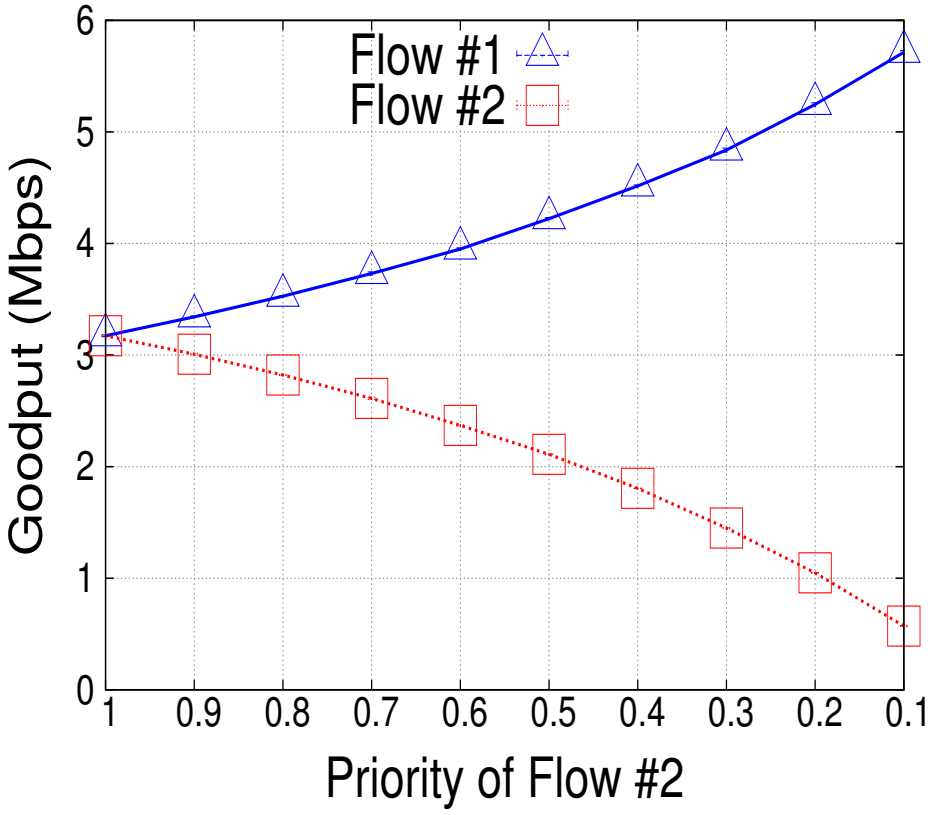
Using priorities to “protect” the app-limited from the greedy flow (RAP)



High-priority (1) application limited flow #1 is hardly affected by a low-priority (0.2) flow #2 as long as there is enough capacity for flow 1

2 FSE controlled flows competing with synthetic traffic (TFRC)

- TMIX synthetic traffic, taken from 60 minute trace of campus traffic at the University of Carolina [TCP Evaluation suite]
  - We used the pre-processed version of this traffic which is adapted to provide an approximate load of 50%

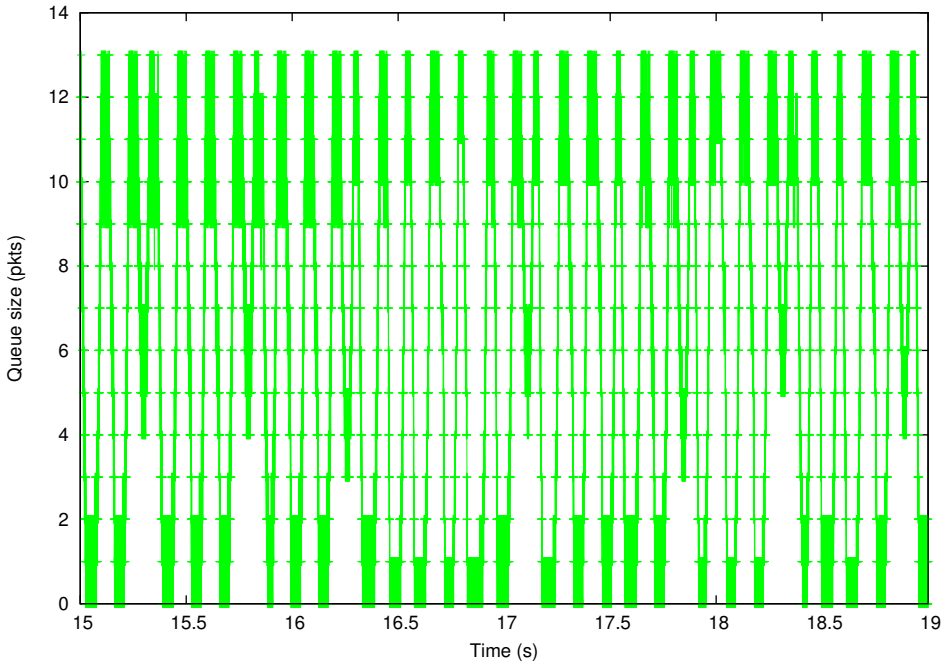


Throughput ratios very close to theoretical values ➔ FSE operation largely unaffected

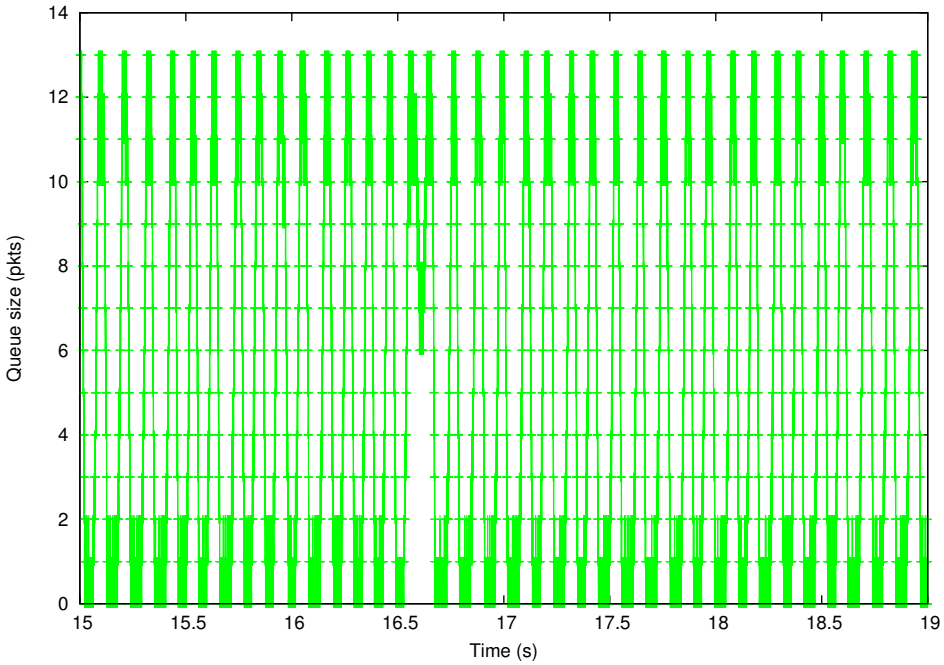
Q&A

Backup slides

What's going on? (previous algorithm)



With FSE



Without FSE

- Queue drains more often without FSE
  - Thought behind expected benefits: coupling emulates *one* flow
    - But, e.g.: 2 flows with rate X each; one flow halves its rate: 2X ➔ 1 ½X
  - When flows synchronize, both halve their rate on congestion, which really halves the aggregate rate: 2X ➔ 1X