Coupled congestion control for RTP media

draft-welzl-rmcat-coupled-cc-02

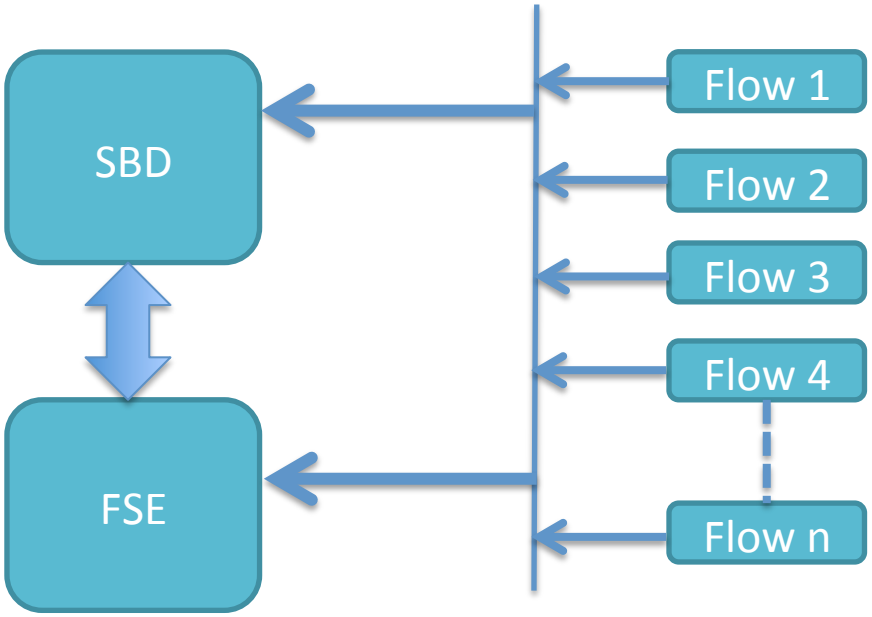
Michael Welzl, Safiqul Islam, Stein Gjessing

RMCAT @ 88th IETF Meeting
Vancouver, BC, Canada
6 November 2013

FP7 RITE
Reducing Internet Transport Latency

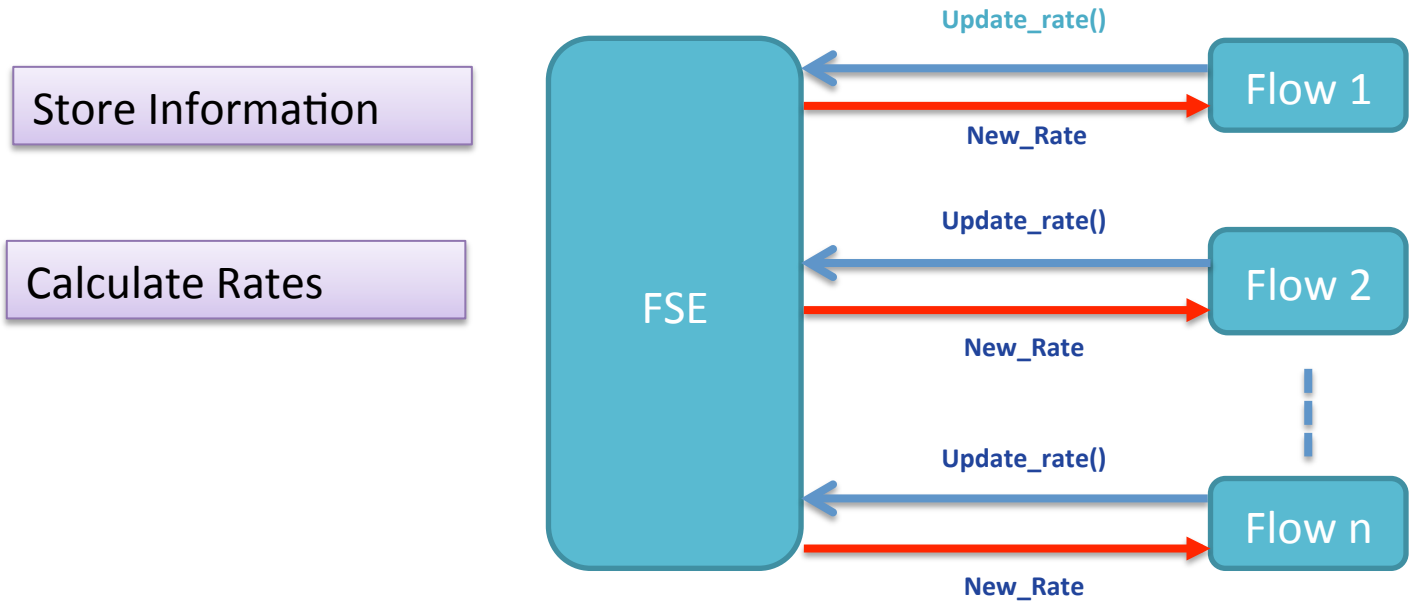
Flow State Exchange (FSE)

Hoping to have a draft by the next IETF



Previous version: only passive

- Goal: Minimal change to existing CC
  - each time it updates its sending rate (New_CR), the flow calls update (New_CR, New_DR), and gets the new rate
  - Complicates the FSE algorithm and resulting dynamics (e.g. need dampening to avoid overshoot by slowly-reacting flows)

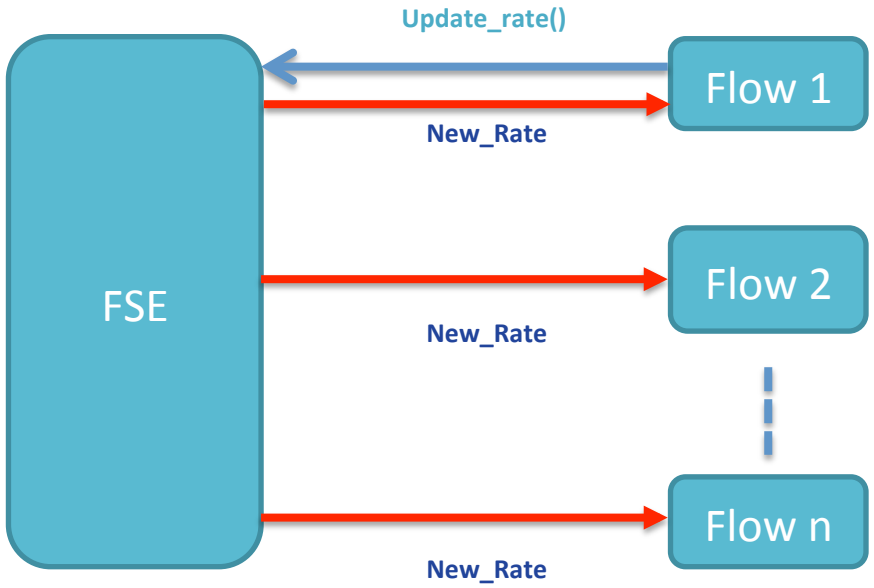


Now added: FSE - Active

- Actively initiates communication will all the flows
  - Perhaps harder to use, but simpler algorithm and “nicer” resulting dynamics

Store Information

Calculate Rates

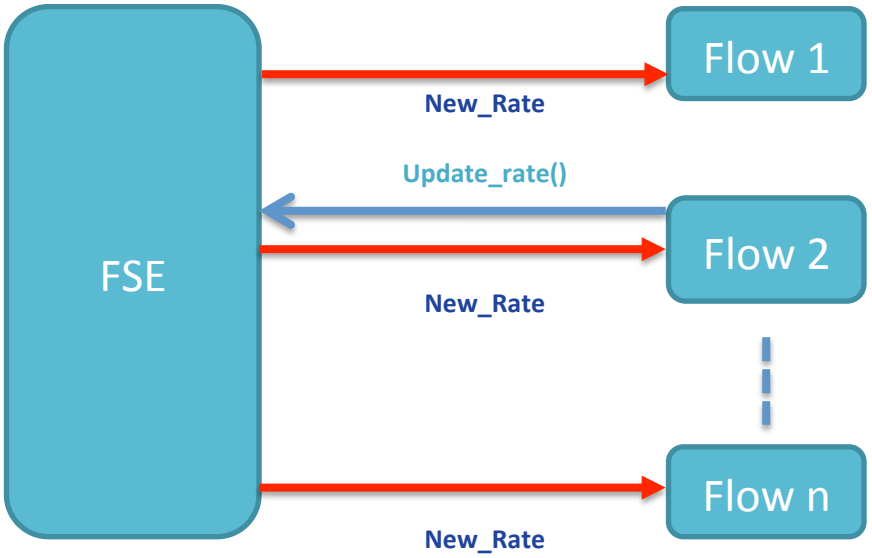


Now added: FSE - Active

- Actively initiates communication will all the flows
  - Perhaps harder to use, but simpler algorithm and “nicer” resulting dynamics

Store Information

Calculate Rates



Active algorithm

- Every time the congestion controller of a flow determines a new sending rate CC_R, the flow calls UPDATE
  - Updates the sum of all rates, calculates the sending rates for all the flows and distributes them to all registered flows

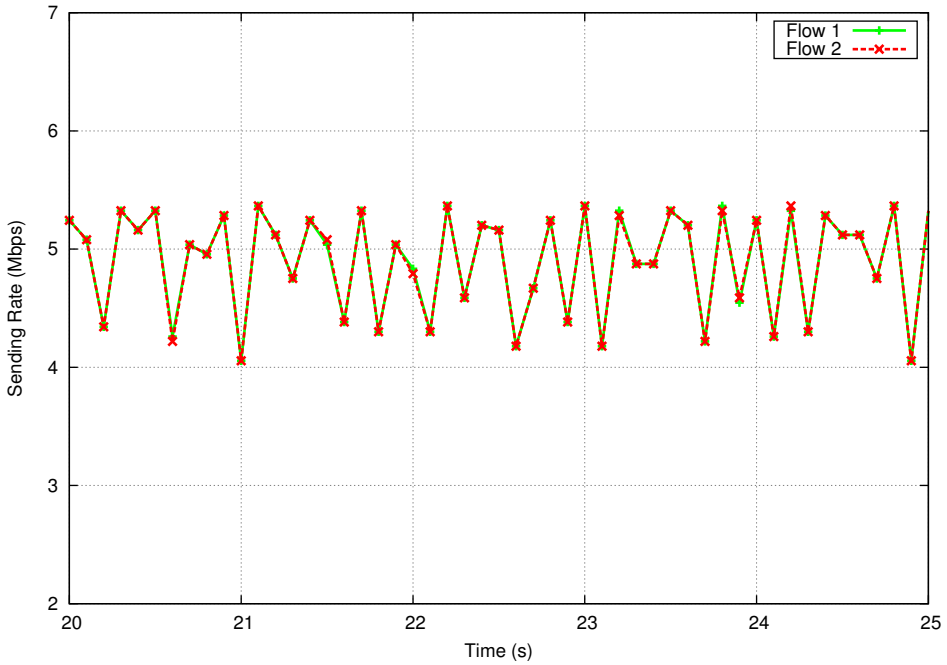
- Essentially all that is left in this version:

```
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
```

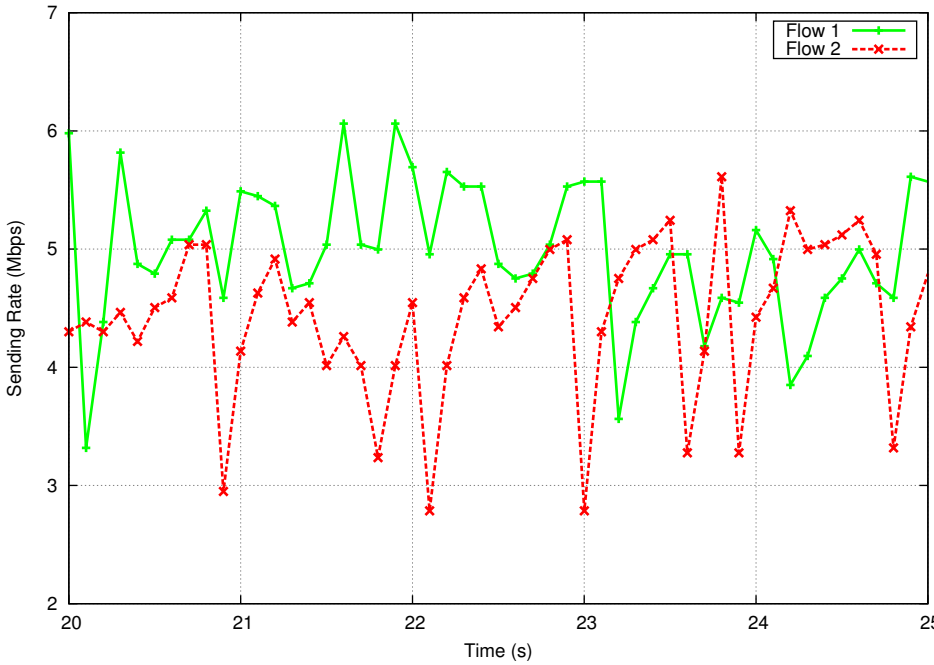
- Designed to be as simple as possible
  - Lacks 1 feature (to be included in the next version): immediately using the capacity freed by application-limited flows

Dynamic behavior: Rate Adaptation Protocol RAP (= rate-based AIMD)

All simulations in this presentation:
Dumbbell topology with bottleneck 10Mb, 1 BDP (13 packets)
drop tail queue, RTT = 10 ms, duration 300 seconds

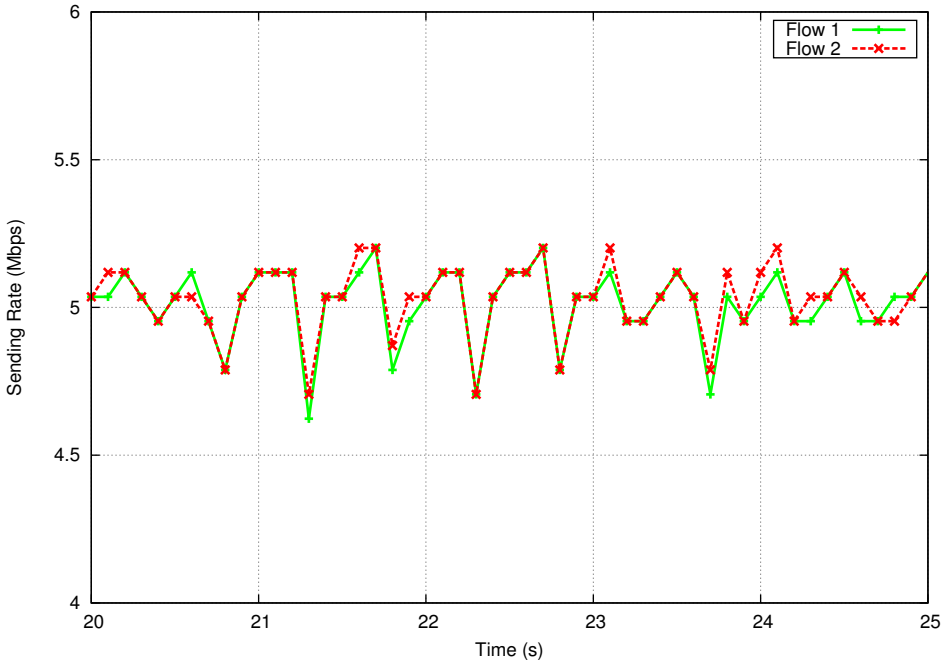


With FSE

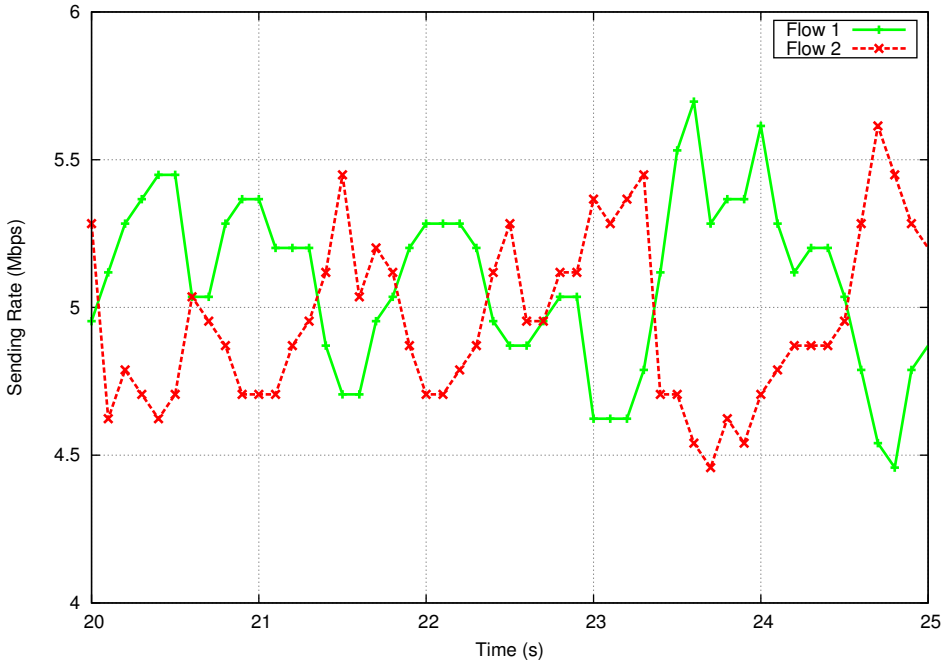


Without FSE

Dynamic behavior: TFRC



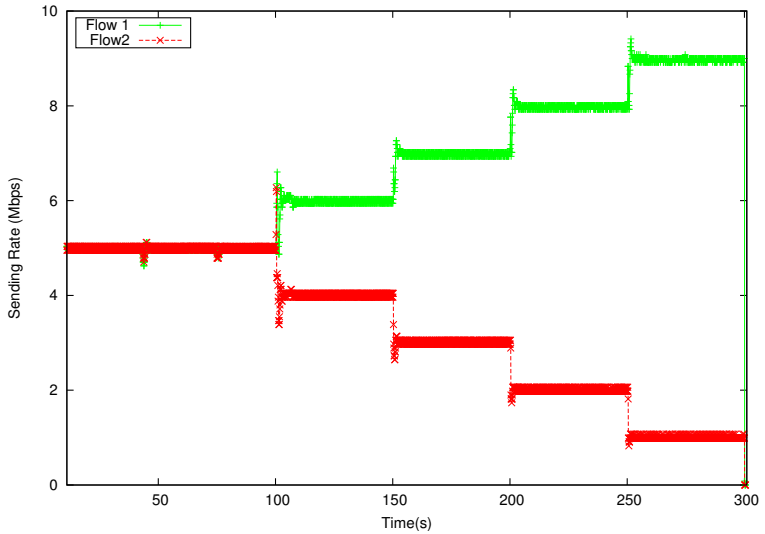
With FSE



Without FSE

FSE goals

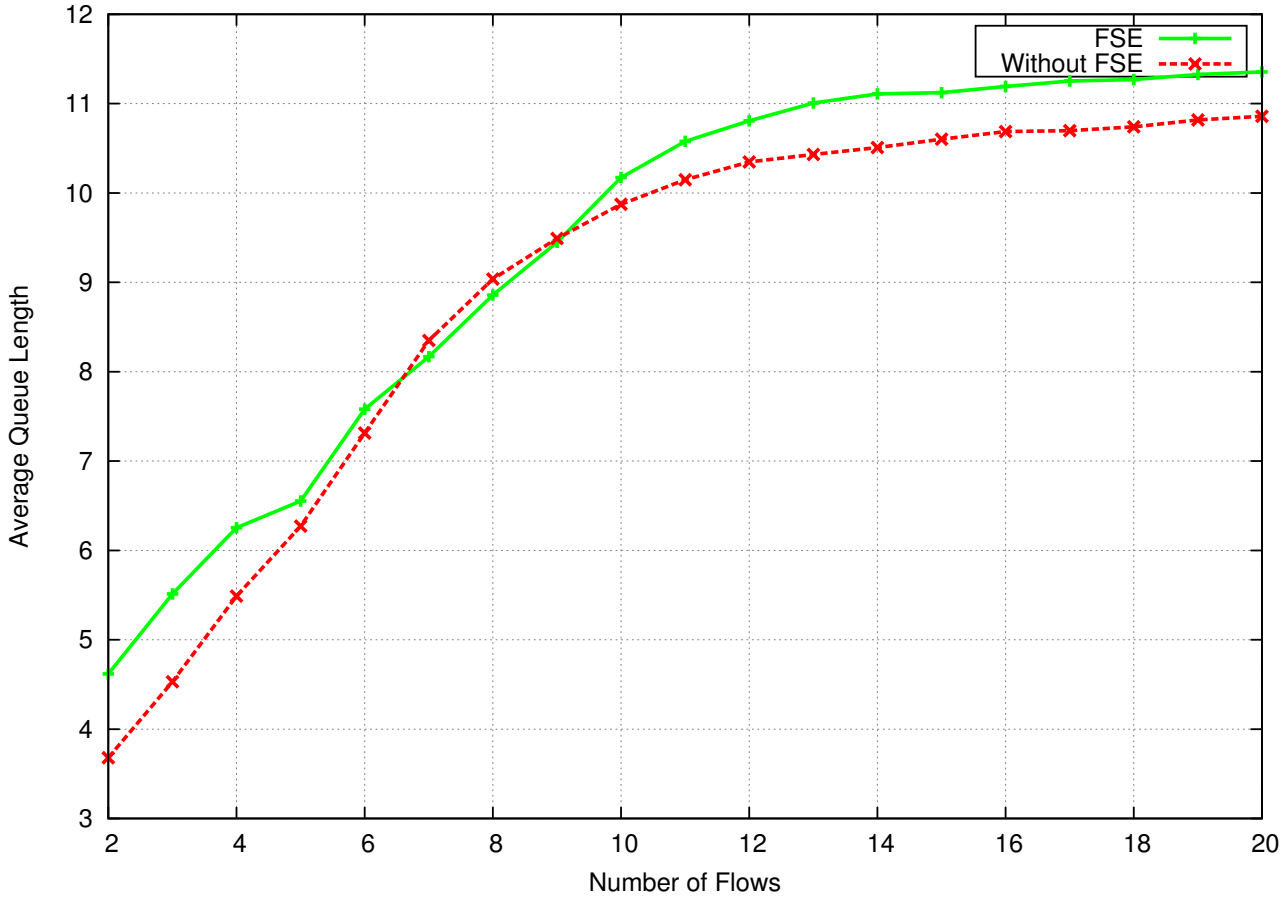
- Charter:
  "Develop a mechanism for identifying shared bottlenecks between groups of flows, and means to flexibly allocate their rates within the aggregate hitting the shared bottleneck."
  (requirement F34 in *draft-ietf-rtcweb-use-cases-and-requirements-12*)
  - This works perfectly
  - Also did in the previous version



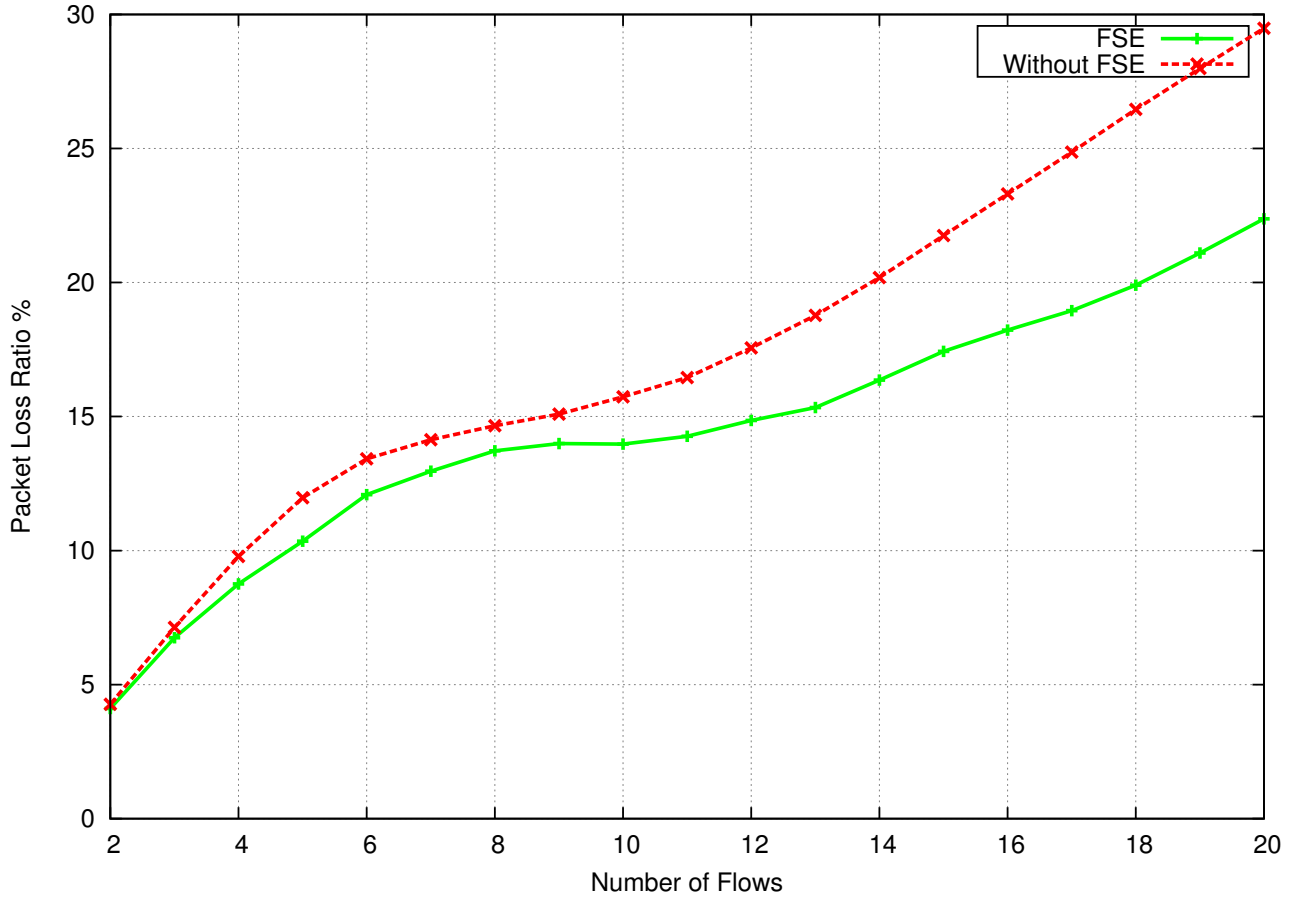
Priority of flow 1 increased over time

- But: because this avoids competition between flows, we expect reduced queuing delay and loss as a side effect

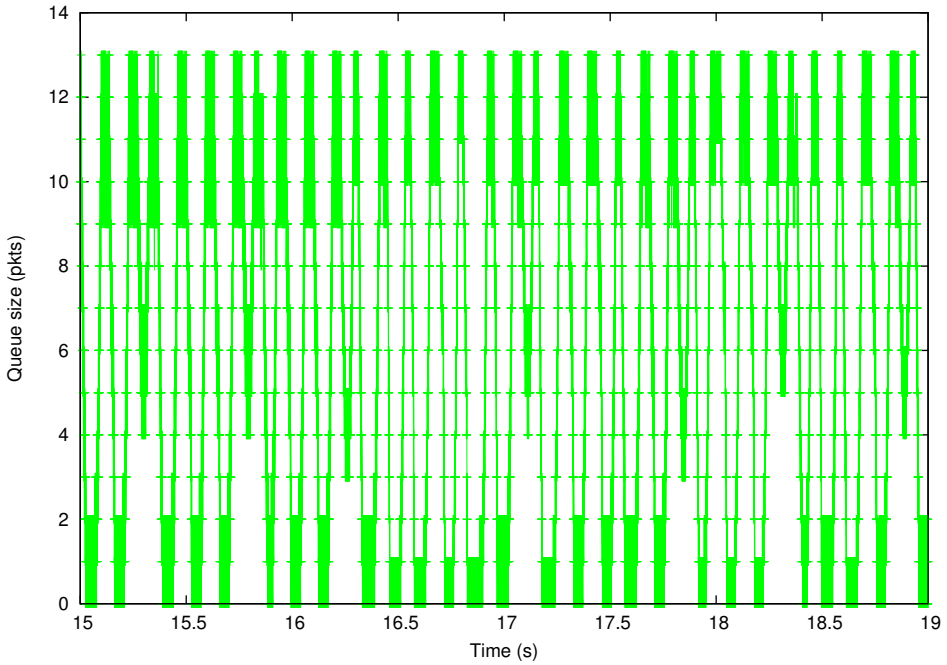
Average queue length (RAP)



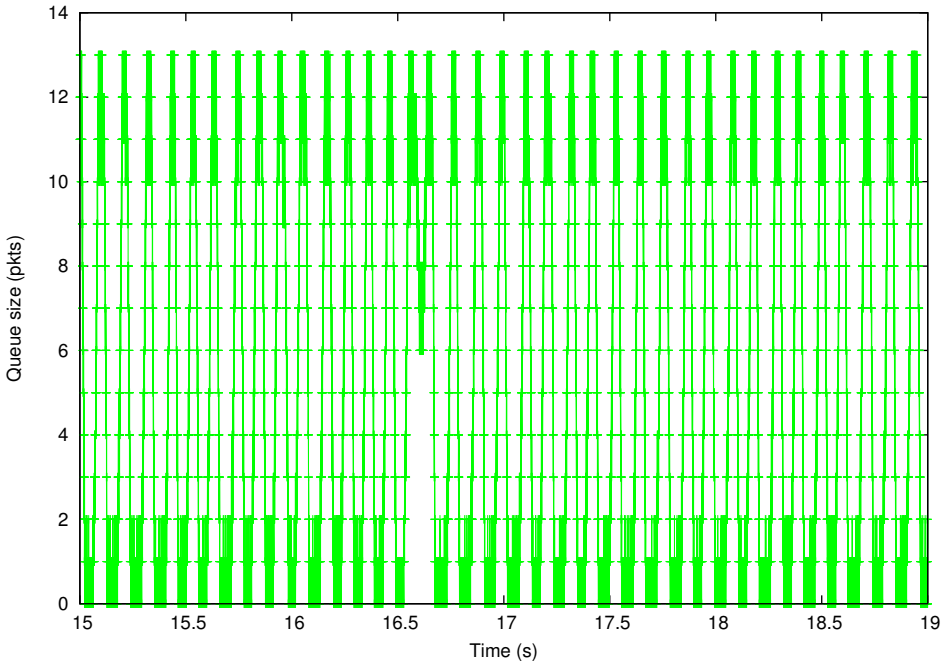
Packet loss ratio (RAP)



What's going on?



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

Current work

- Trying to fix this (proportionally reduce aggregate rate on congestion, but increase by delta/N)
  - Some issues, e.g. slow start

- Why do we have these problems?
  - Because all papers on RFC2140 etc. did not focus on reducing queuing delay
  - RFC2140 cwnd sharing probably has the same problem

Q&A