Tightly Coupled Congestion Control in WebRTC

Michael Welzl
Context

- **draft-ietf-rtcweb-use-cases-and-requirements-09.txt**:
  - “The browser MUST support prioritization of streams and data.”
    e.g.: online game (control traffic = important) with audio (less important)

- **IETF RTP Media Congestion Avoidance Techniques (RMCAT) WG** is now chartered
  - Focus broader, but really made for WebRTC
  - Charter contains:
    “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.”
Why is this a big deal?

• Because doing it right gives us extra benefits

• Because it’s a unique chance to do it right
Up to now: doing it wrong

- Priorities in practice, in today’s Internet
  - More flows get more than one
  - Mostly TCP, with congestion control trying to reach a certain notion of fairness
    - This “fairness” has been criticized a lot (e.g.: depend on RTT)
    - Doesn’t get better with N vs. M TCP’s

More queue growth → more delay and packet drops
How to fix this

• The problem can be solved with a single Congestion Control instance (as with the Congestion Manager, RFC3124)
  – But solving it in general is hard – RFC3124 leaves some key issues unresolved + benefits weren’t shown
    • shared bottleneck or not?
    • overall less aggressive CC – bad e.g. for short flows?
    ... all at the cost of a complex implementation!

• But we could do this right for rtcweb
  – Common bottleneck is assumed (all-over-one-5-tuple)
  – long connections are somewhat likely
Doing it right

• Act like one flow, with aggression tuned correctly (or: as desired by the user)
  – Less delay
  – Less packet loss
  – Less signalling
    (N flows don’t need N*feedback about the same path)
  – More controllable behavior
    (sender-side scheduling vs. “fighting it out” on the bottleneck)
  – Better performance for short or application-limited flows
    (TCP does use it or lose it; with shared congestion control, if flow 1 doesn’t use it, maybe flow 2 does.
    Skip slow start: again less queuing delay from slow start overshoot)
Prototypical example

“Flow State Exchange” (FSE)

- The result of searching for minimum-necessary-standardization: only define what goes in / out, how data are maintained
  - Could reside in a single app (e.g. browser) and/or in the OS
  - Probably needed on sender and receiver side

Traditional CM

- Traditional CM
- FSE-based CM
- Another possible implementation of flow coordination
Conclusion

- RMCAT has been formed, and quite some interesting things could come out of it
- Coupled congestion control is one of them
- Ideally, yields perfect fairness control, reduced delay, less drops, better performance in general
- Realization: FSE is one possible way; discussions have only just begun
Thank you!

Questions?