Initial Results!

Receiver-side Real-Time Congestion Control (RRTCC)

Google Congestion Control Algorithm

IETF 88, Vancouver

06th November

Varun Singh, Albert Lozano, Jörg Ott

Outline

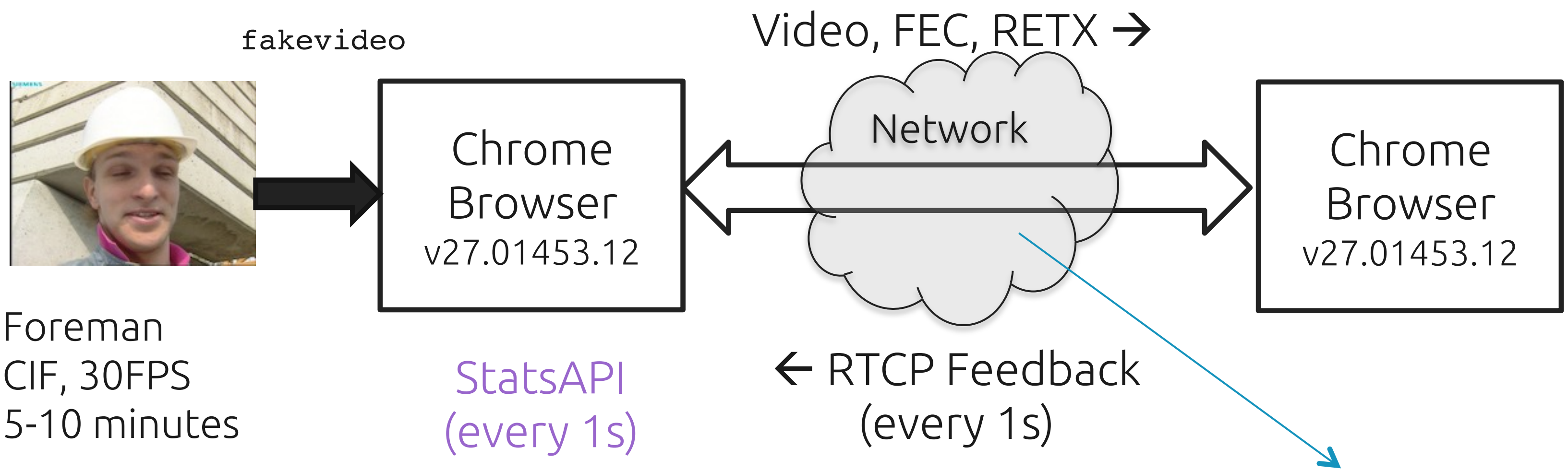
- Related Drafts:
  - draft-alvestrand-rmcat-congestion
  - draft-alvestrand-rmcat-remb-02

- Single Flow
  - Different losses, latency, queue length

- 3 RMCAT flows
  - Start together
  - Start at 30s apart

- RMCAT vs long TCP flows

Evaluation Setup

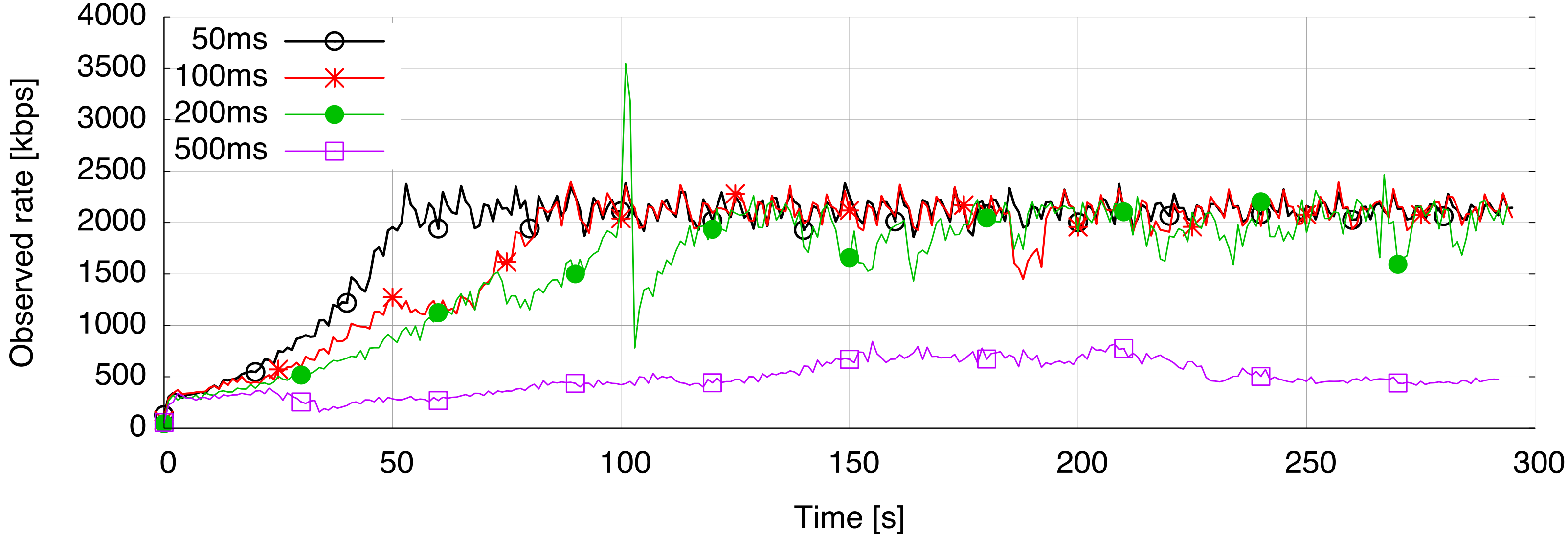


http://media.xiph.org/video/derf/

Single flow

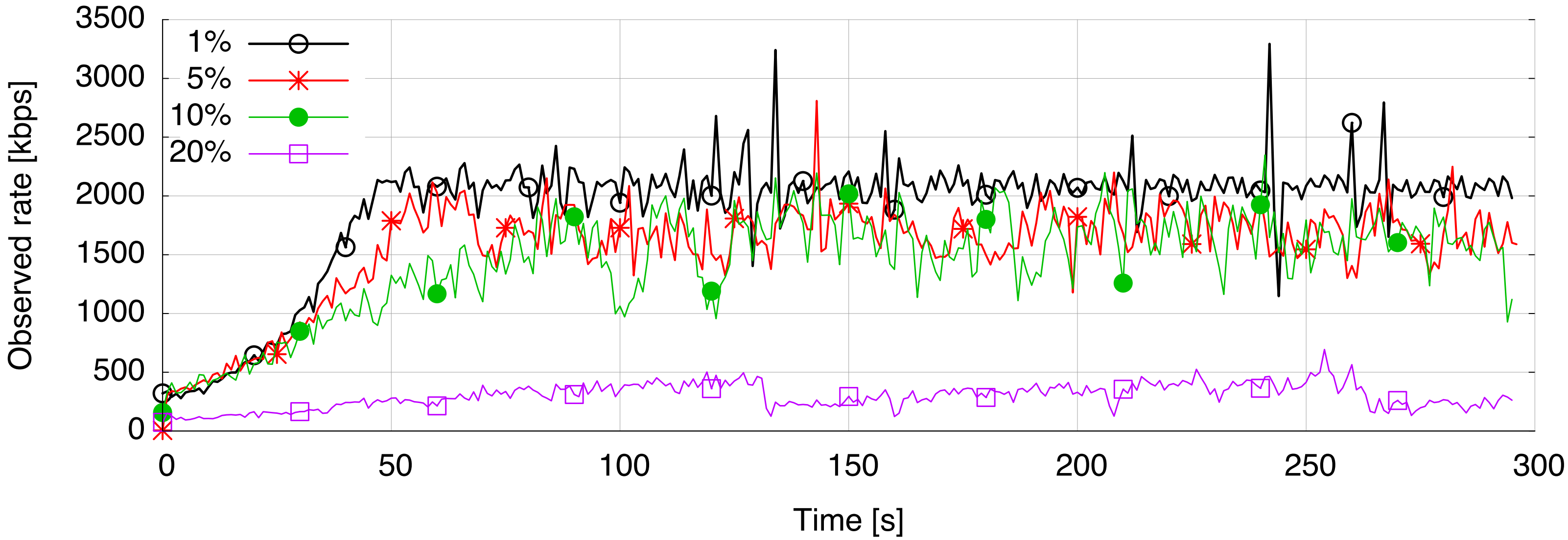
- Fixed capacity with different
  1. path latencies
  2. path losses
  3. router queue size

Different Latencies



	Rate (Kbps)	RTT (ms)	Residual Loss (%)	Packet Loss (%)
0 ms	1949.7±233.62	9.57±2.41	0.011	0.011
50 ms	1913.56±254.86	102.51±1.44	0.05	0.05
100 ms	1485±268.11	202.57±3	0.06	0.06
200 ms	560.82±129.57	401.91±3.33	0.33	0.4
500 ms	255.67±45.85	1001.36±3.99	0.35	0.37

Different Loss

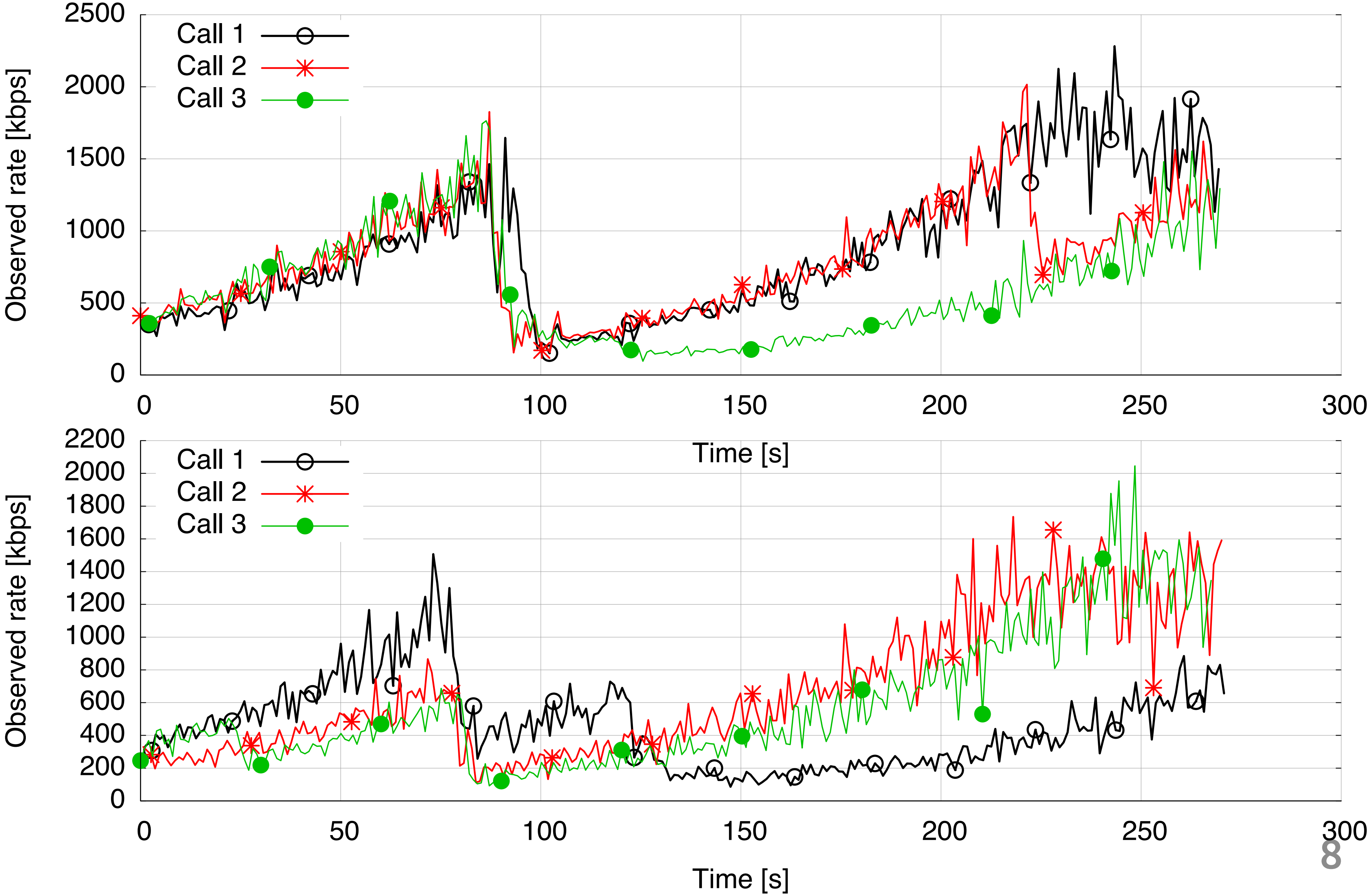


	Rate (Kbps)	RTT (ms)	Residual Loss (%)	Packet Loss (%)
0%	1949.7±233.62	9.57±2.41	0.011	0.011
1%	1986.91±256.78	8.12±1.86	0.09	2
5%	1568.74±178.52	6.98±1.79	0.23	9.77
10%	1140.82±161.92	6.28±3.24	0.49	19.02
20%	314.4±61.98	5.42±4.03	2.43	36.01

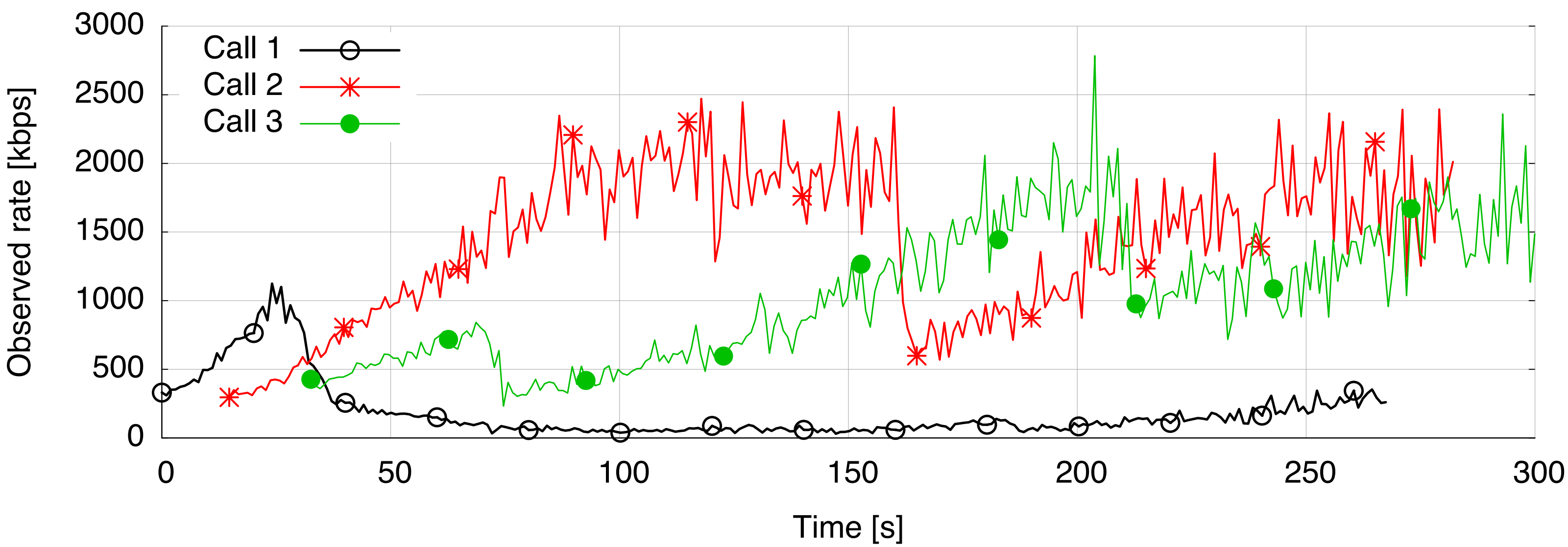
Different Queue Size

Link Capacity = 1 Mbps				
	Rate (Kbps)	**RTT (ms)**	**Residual Loss (%)**	**Packet Loss (%)**
100 ms	362.87±68.18	31.42±11.9	0.79	1.42
1 s	374.64±58.64	27.48±8.81	0.05	0.05
10 s	438.32±31.58	27.16±7.32	0.04	0.04
Link Capacity = 5 Mbps				
	Rate (Kbps)	**RTT (ms)**	**Residual Loss (%)**	**Packet Loss (%)**
100 ms	1965.95±224.02	20.13±4.16	0.02	0.02
1 s	1920.67±234.38	18.45±5.57	0.02	0.02
10 s	1722.08±261.42	17.05±4.52	0.03	0.03

3 RMCAT streams

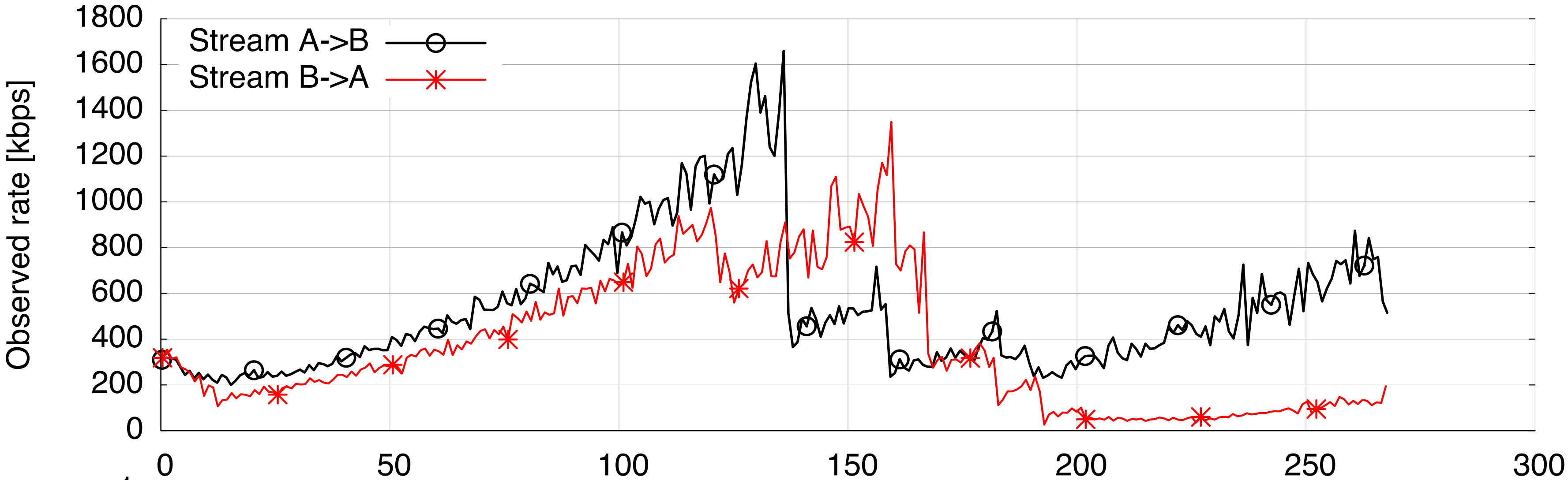


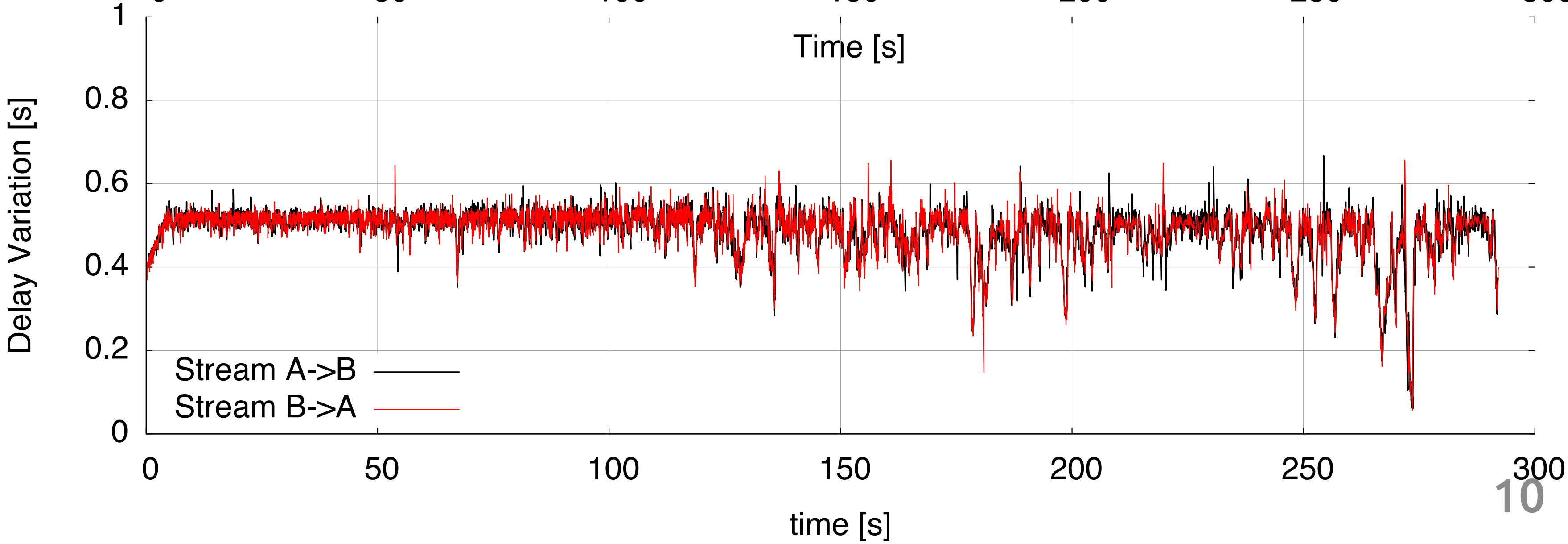
3 RMCAT flows (time-shifted arrival)



In all the cases, the first call reduced its rate
In 20% of the cases it recovered after ~50s.

TCP and RMCAT





Observations

- Up to 20% the average send rate can be FEC
- Retransmissions (retx)
  - Used extensively in low latency scenarios.

- Observed starvation when competing with TCP traffic
- In self-fairness, first flow starves sometimes.

Additional Reading

- ***Performance Analysis of Receive-Side Real-Time Congestion Control for WebRTC***, Singh et al. http://www.netlab.tkk.fi/~varun/singh2013rrtcc.pdf
- ***Experimental Investigation of the Google Congestion Control for Real-Time Flows***, Cicco et al. http://conferences.sigcomm.org/sigcomm/2013/papers/fhmn/p21.pdf
- ***Performance analysis of topologies for Web-based Real-Time Communication (WebRTC)***, A. Lozano https://aaltodoc.aalto.fi/bitstream/handle/123456789/11093/master_Abell%C3%B3_Lozano_Albert_2013.pdf?sequence=1
- ***Understanding the Dynamic Behaviour of the Google Congestion Control***, Cicco et al.