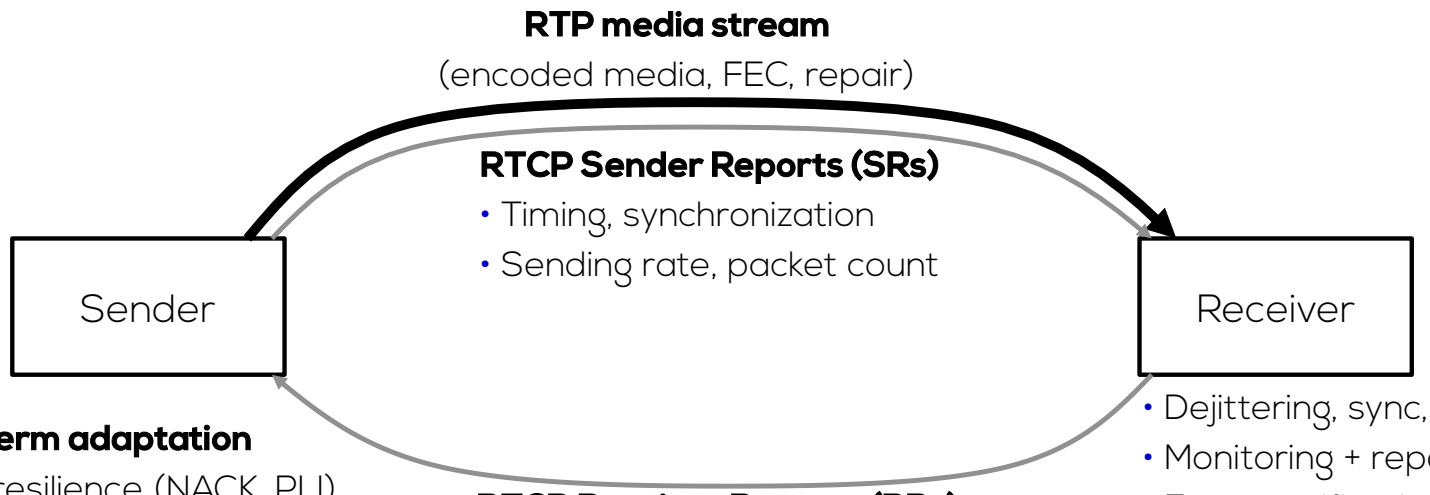
Performance Analysis of Receive-Side Real-Time Congestion Control (RRTCC) for WebRTC

Google's Congestion Control Algorithm

Packet Video 2013
13th December, Friday
Varun Singh, Albert Lozano, Jörg Ott

RTP and RTCP



Short-term adaptation

- Error-resilience (NACK, PLI)
- Congestion control
- Adaptive source coding

Long-term adaptation

- Codec choice
- Packetization size
- FEC, interleaving

RRTCC

- Sender-side based on **TFRC**
  - RTT, fractional loss, bytes_sent
- Receiver-side based on variation in inter-arrival time
  - Estimation based on a Kalman filter.
  - Indicates Receiver Estimate (REMB) in RTCP
  - RTCP sent every 1 sec.
- Sender decides based on TFRC and REMB

Related Work

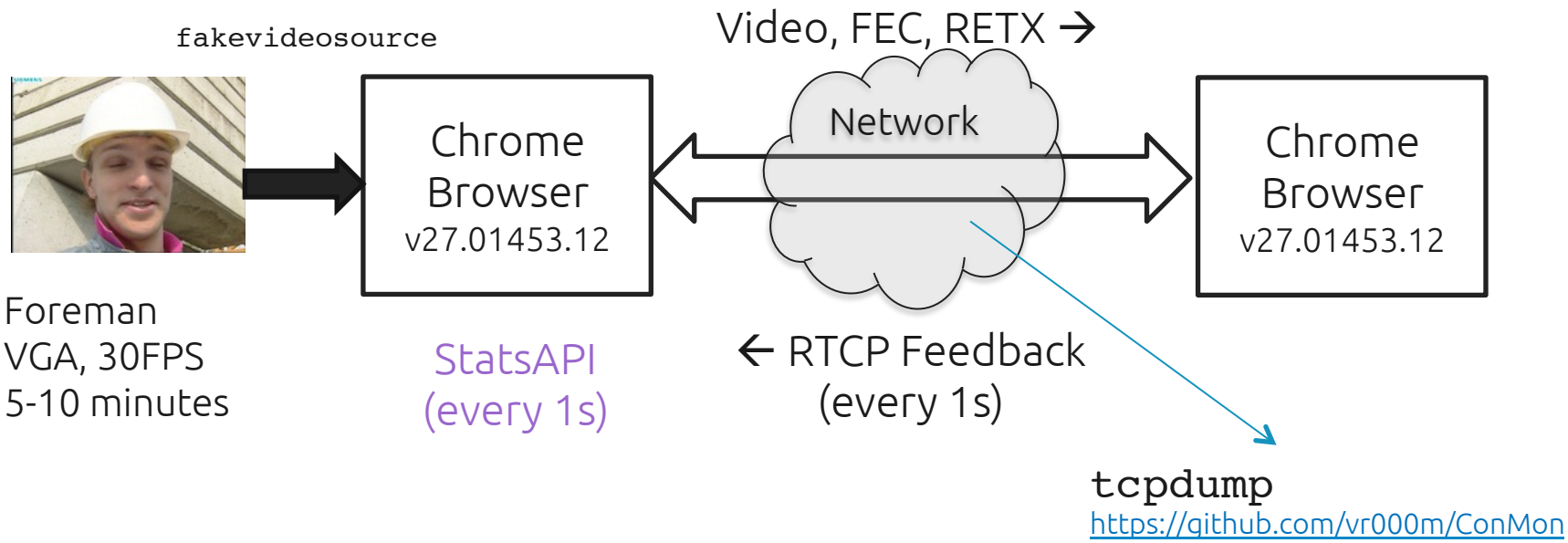
- ***Experimental Investigation of the Google Congestion Control for Real-Time Flows***, Cicco et al.
http://conferences.sigcomm.org/sigcomm/2013/papers/fhmn/p21.pdf

- ***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.

Initial NS-2 Results: http://www.netlab.tkk.fi/~varun/rrtcc-tcp-competition-00.pdf

Evaluation Setup

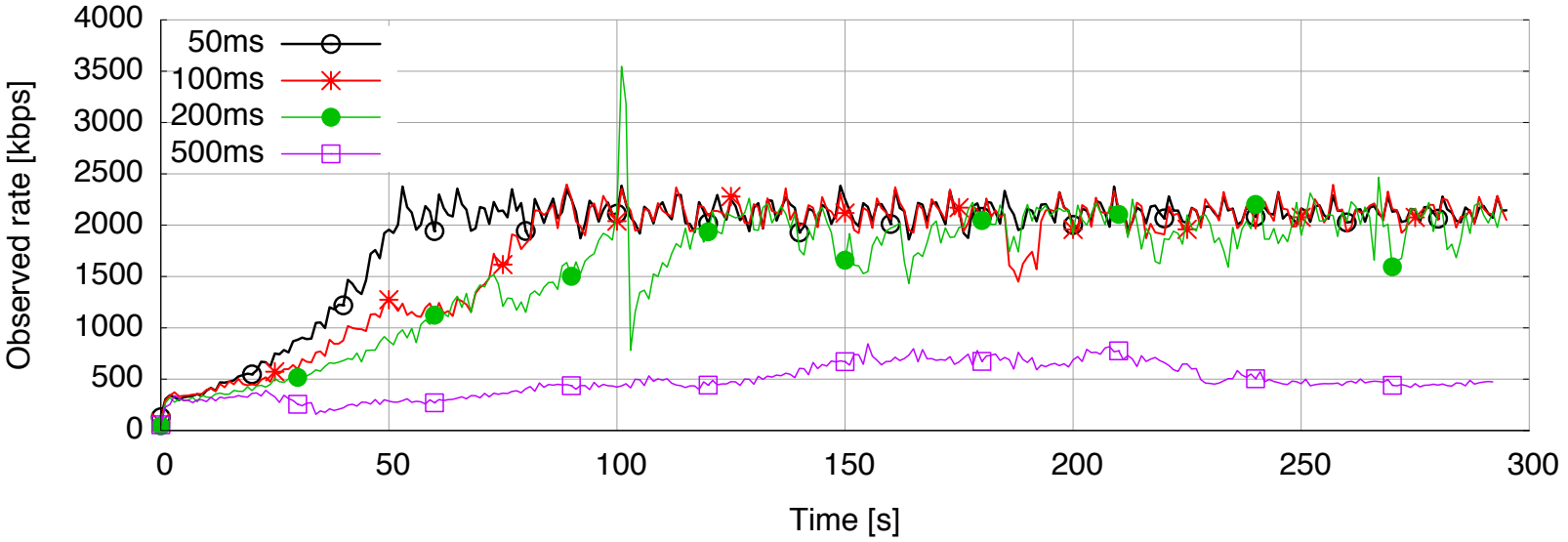


http://media.xiph.org/video/derf/

Single flow

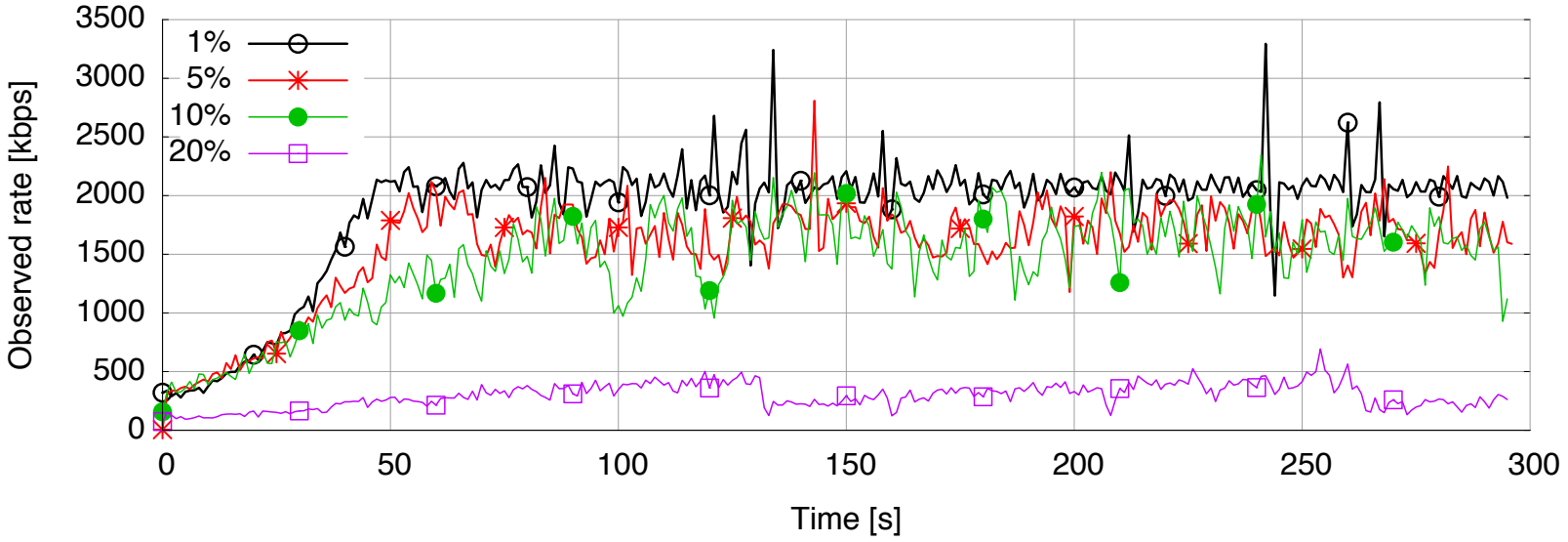
- Fixed capacity with different
  1. path latencies
  2. path losses

Different Latencies



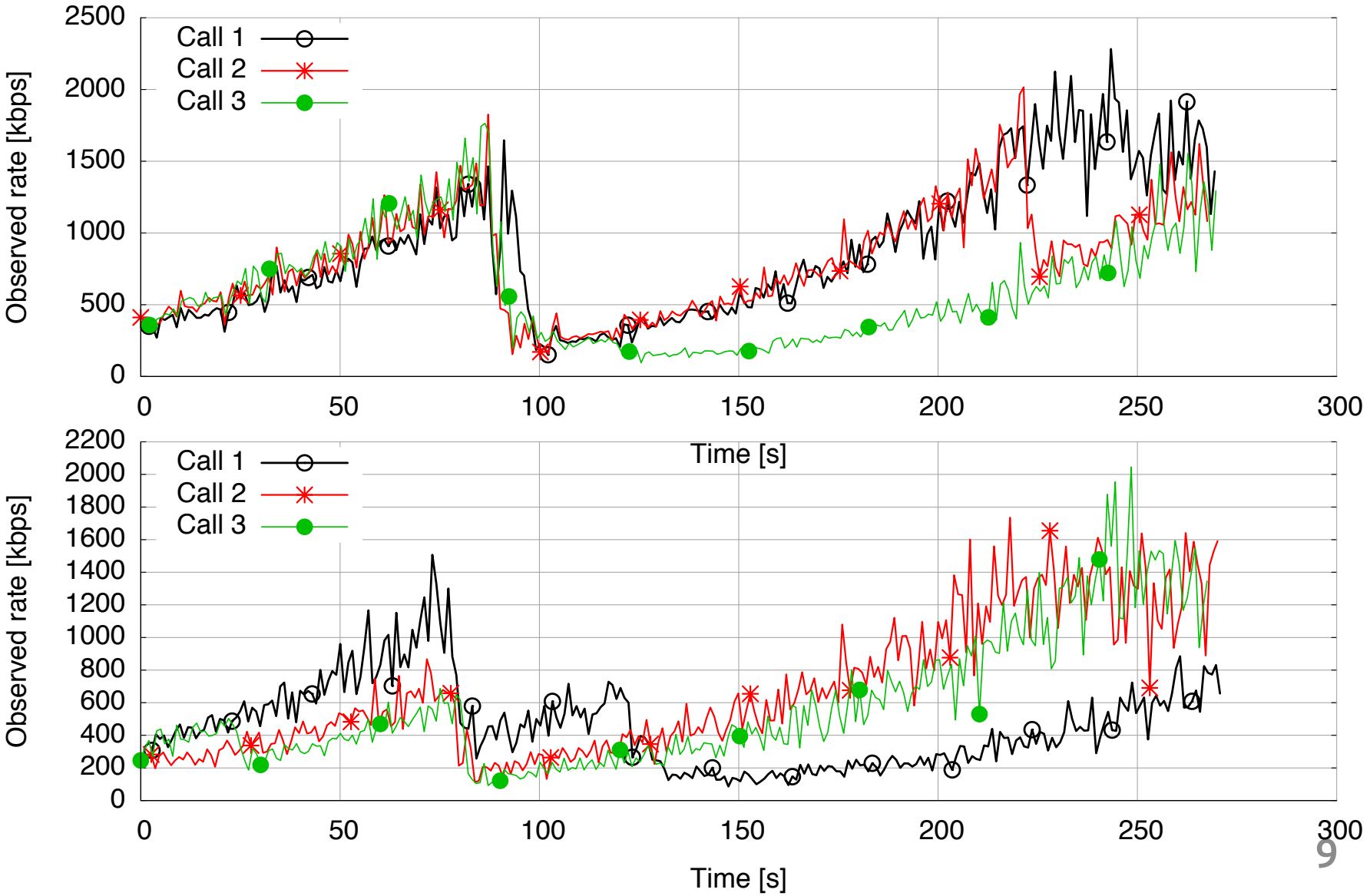
	Rate (Kbps)	RTT (ms)	Residual Loss (%)	Packet Loss (%)
0 ms	1949.7±233.62	9.57±2.41	0.011	0.011
50 ms	1913.56±254.86	102.51±1.44	0.05	0.05
100 ms	1485±268.11	202.57±3	0.06	0.06
200 ms	560.82±129.57	401.91±3.33	0.33	0.4
500 ms	255.67±45.85	1001.36±3.99	0.35	0.37

Different Loss

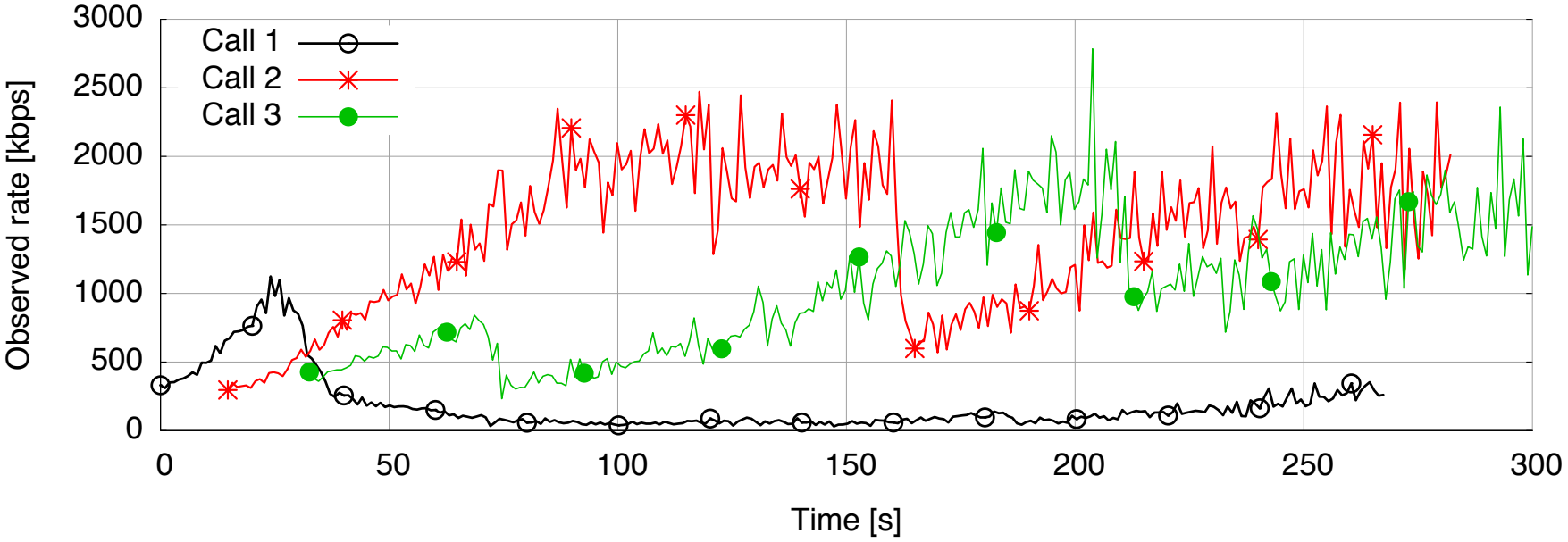


	Rate (Kbps)	RTT (ms)	Residual Loss (%)	Packet Loss (%)
0%	1949.7±233.62	9.57±2.41	0.011	0.011
1%	1986.91±256.78	8.12±1.86	0.09	2
5%	1568.74±178.52	6.98±1.79	0.23	9.77
10%	1140.82±161.92	6.28±3.24	0.49	19.02
20%	314.4±61.98	5.42±4.03	2.43	36.01

3 RMCAT streams

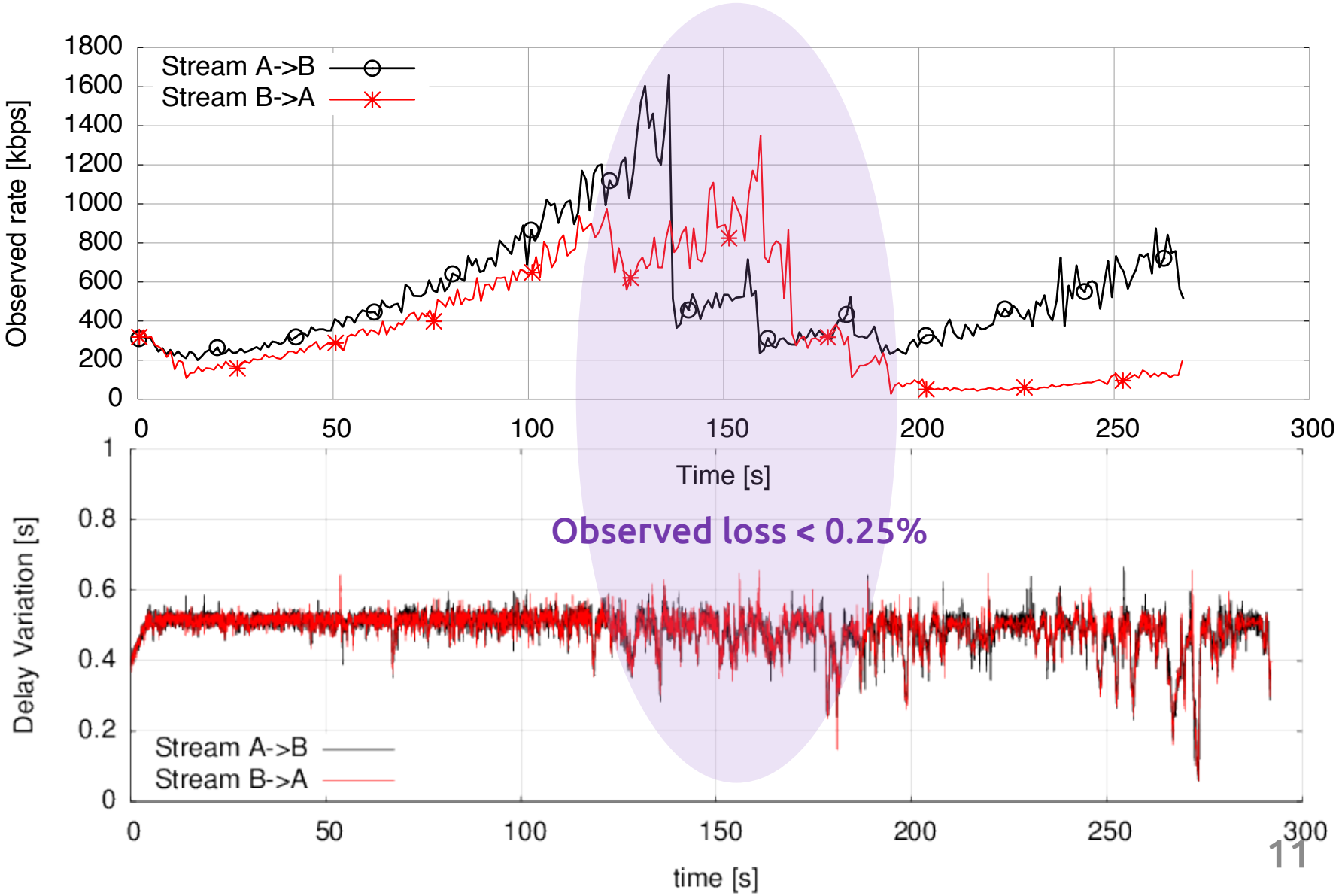


3 RMCAT flows (time-shifted arrival)



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

TCP and RMCAT



Observations

- FEC is ~20% of the average send rate
- Retransmissions (retx)
  - Used extensively in low latency scenarios.

- RRTCC is self-fair
  - flows increase and decreases synchronously
  - first flow collapses when new flows are added later.

- Under-utilizes when competing with TCP traffic

Conclusions

- Works in low delay networks
- Tolerate transient changes
  - varying loss, latency
- Compete within limits against varying amount of cross-traffic.