Comparison: GCC - NADA
GCC: Emulation Results

RMCAT - Interim meeting

Cesar Magalhaes & Stefan Holmer, Google
Luca De Cicco & Gaetano Carlucci, Politecnico di Bari

Outline

- TestBed Settings: Simulator and Emulator
- GCC - NADA: Simulation Comparison
- GCC: Emulation results
- Open Issues and Future Work

Introduction

- Comparison of GCC-03 and NADA-06, based on eval-test-01.
- Implementations and simulator framework available here.
- Results presented with:
  - Bar charts or tables of throughput, delay and packet loss.
  - Line charts of throughput, delay and packet loss dynamics.

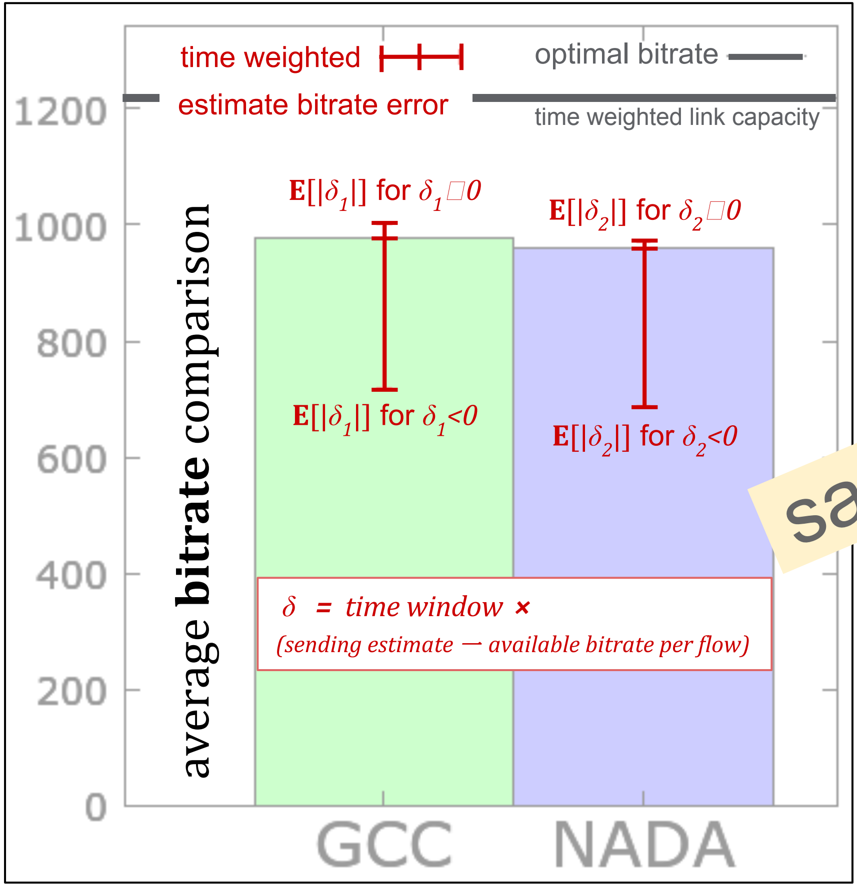
- Compared RMCAT proposals:

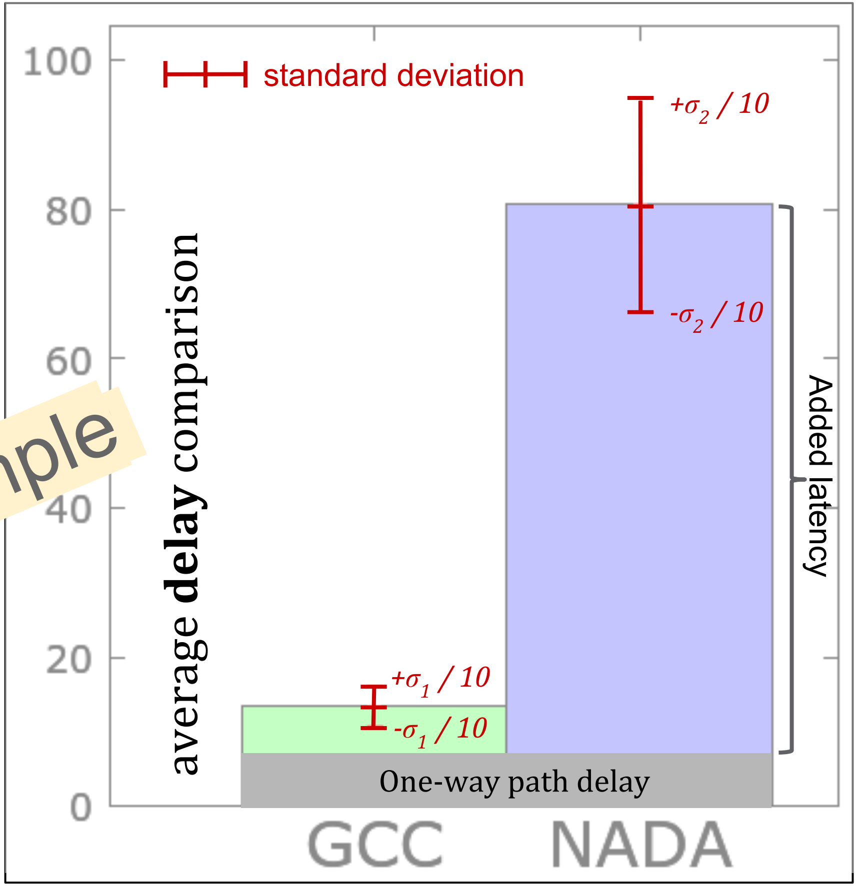
	Implemented according to:	Code available
GCC: Google Congestion Control	Internet draft	here
NADA: Network-Assisted Dynamic Adaptation	Internet draft*	here

Simulation Codec / Sender** parameters:
Min bitrate = 50 kbps
Max bitrate = 2500 kbps

*) NADA's rate shaping buffer was not included. **) NADA's suggested Min/Max are 150,1500 kbps

[Test case scenario description]





The bitrate error bars show average positive and negative deviation from the optimal bitrate.

Simulator: TestBed settings

https://tools.ietf.org/html/draft-ietf-rmcat-eval-test-01

Default evaluation test bed parameters -- Specified otherwise

one-way path delay	*50ms*
bottleneck queueing size	*300ms*
maximum end-to-end jitter	*30ms*
path loss ratio	*0%*

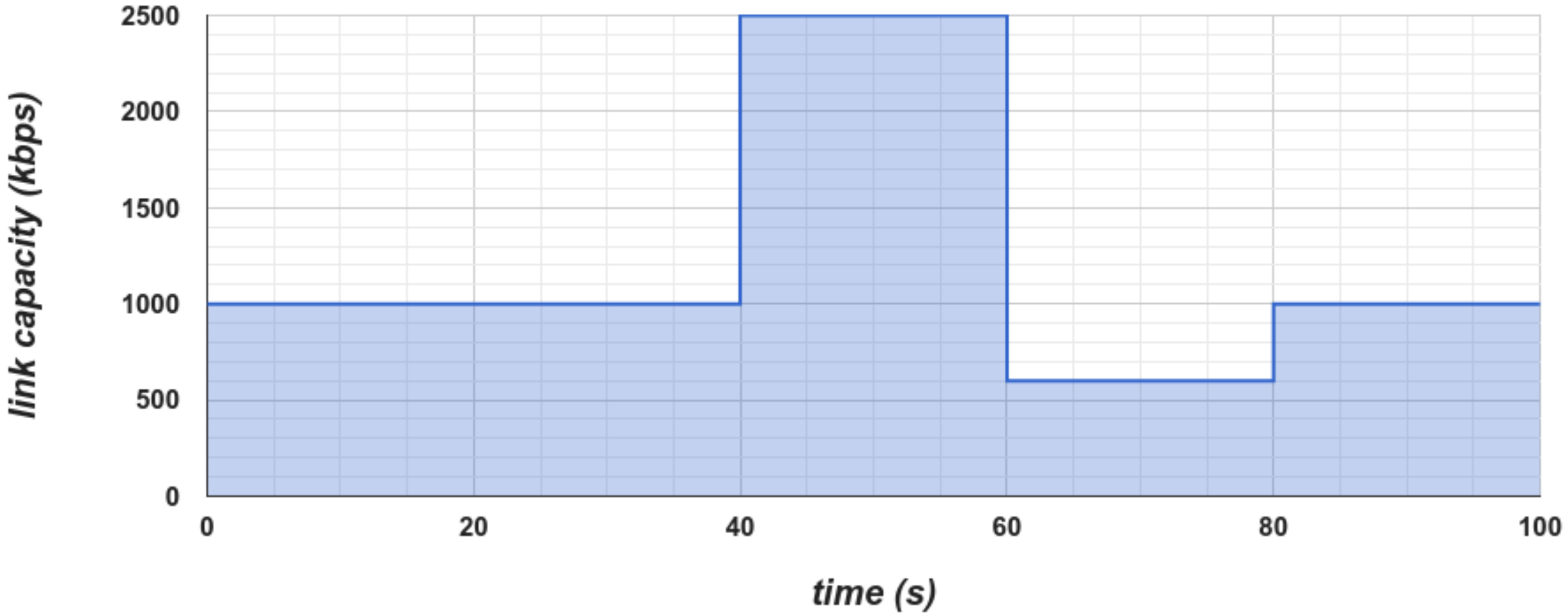
Queue type: Drop-tail
Jitter model: Absolute value of a truncated 2σ gaussian, σ = 15ms (as defined in eval-criteria-03)
Jitter model ✴: $\lvert$ Truncated 3σ gaussian $\rvert$, σ = 5ms

Video source	*CBR @ 30 fps*

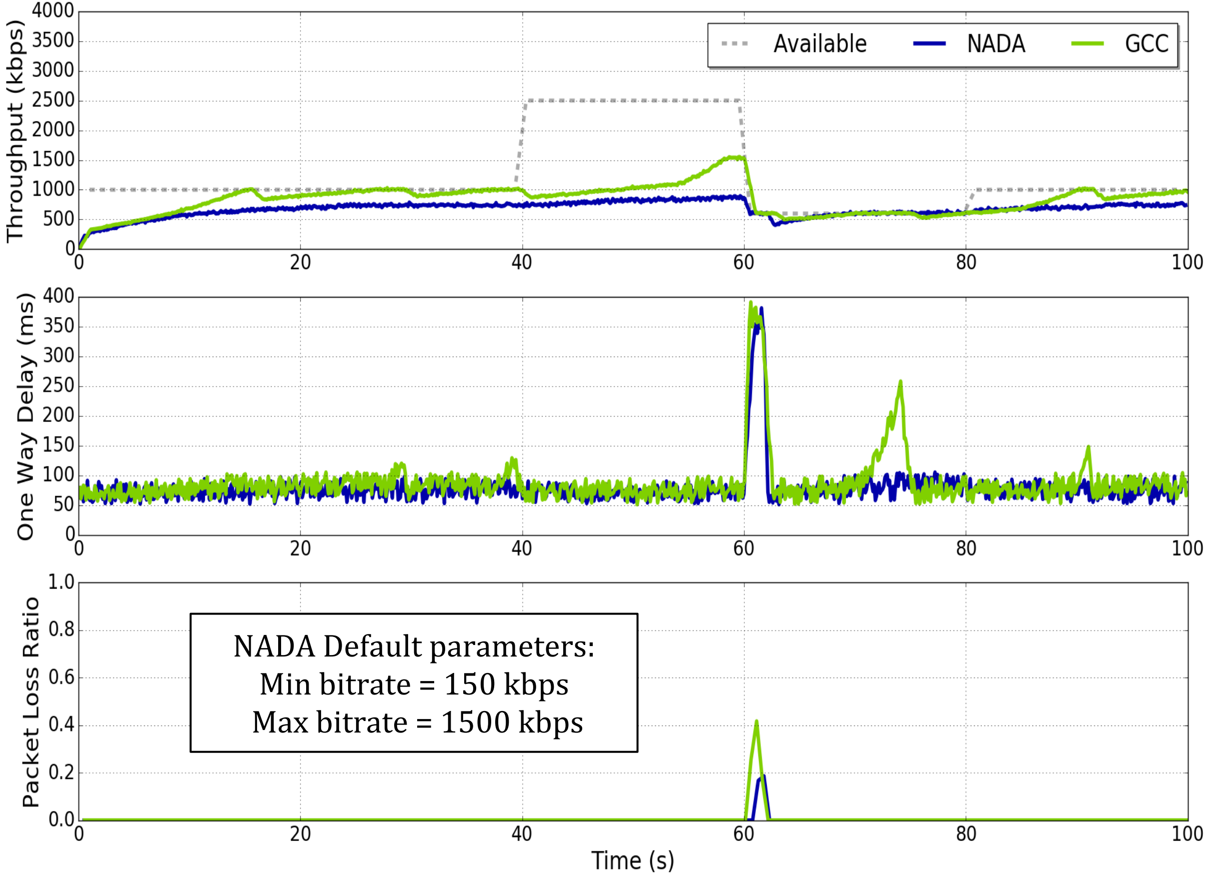
✴) Modified jitter model to get a better comparison where both candidates behave well.

Evaluation test 5.1

Single RMCAT flow, variable link capacity:



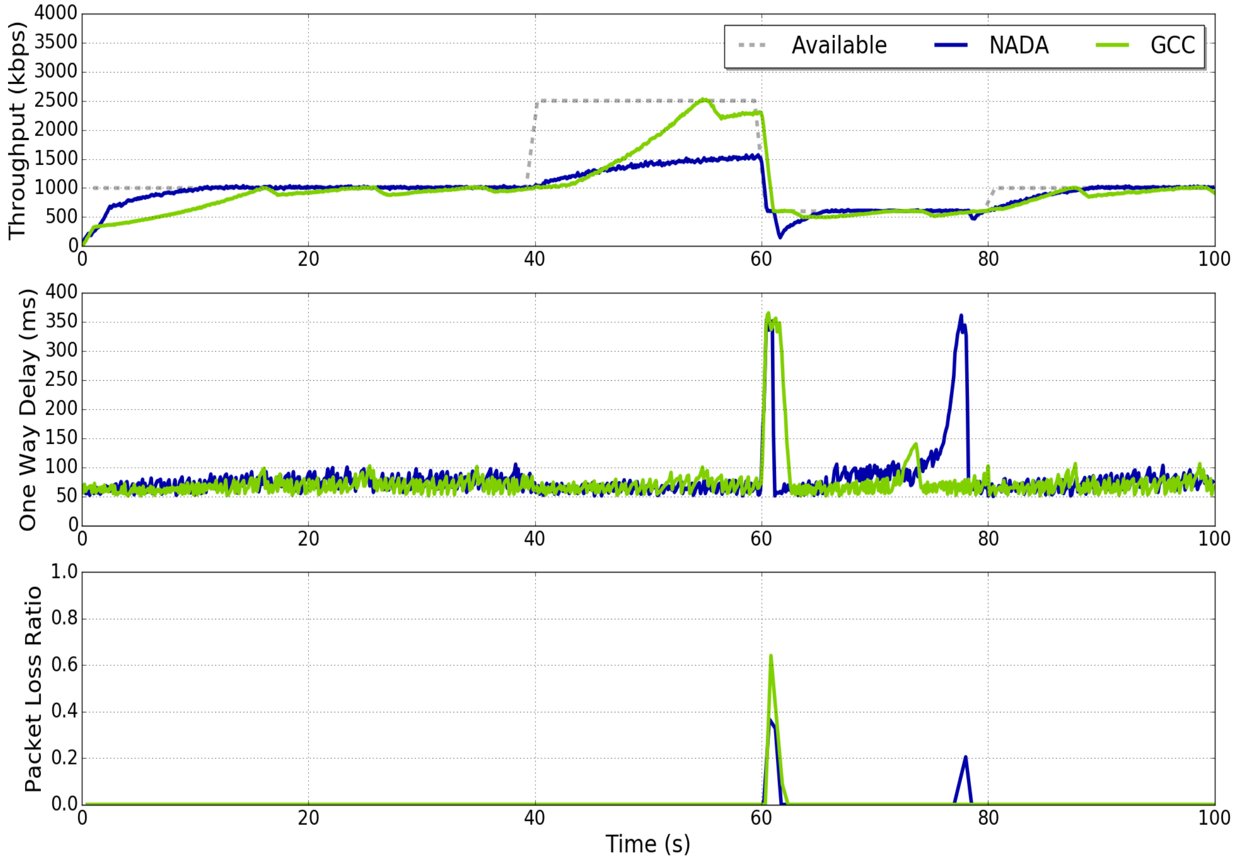
Evaluation test 5.1 default parameters - |original jitter model|



GCC	NADA
Channel utilization	
67.8%	54.3%
Queuing delay (ms) 5% percentile - mean - 95% percentile	
7 - 34.0 - 72	6 - 26.6 - 44
Loss ratio	
0.48%	0.18%

GCC is overshooting at 70 s, due to the high jitter and the capacity drop.
NADA doesn't try to adapt due to the high jitter.

Evaluation test **5.1** *Jitter model* ✶

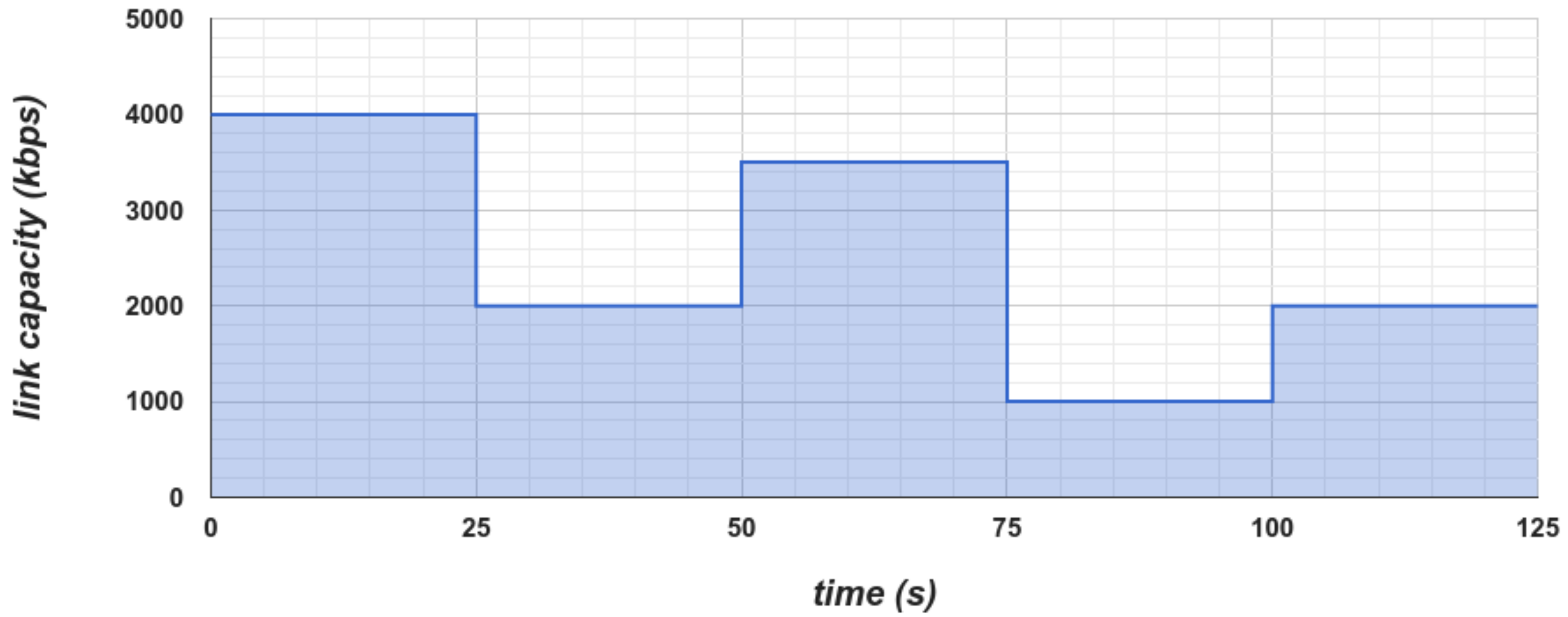


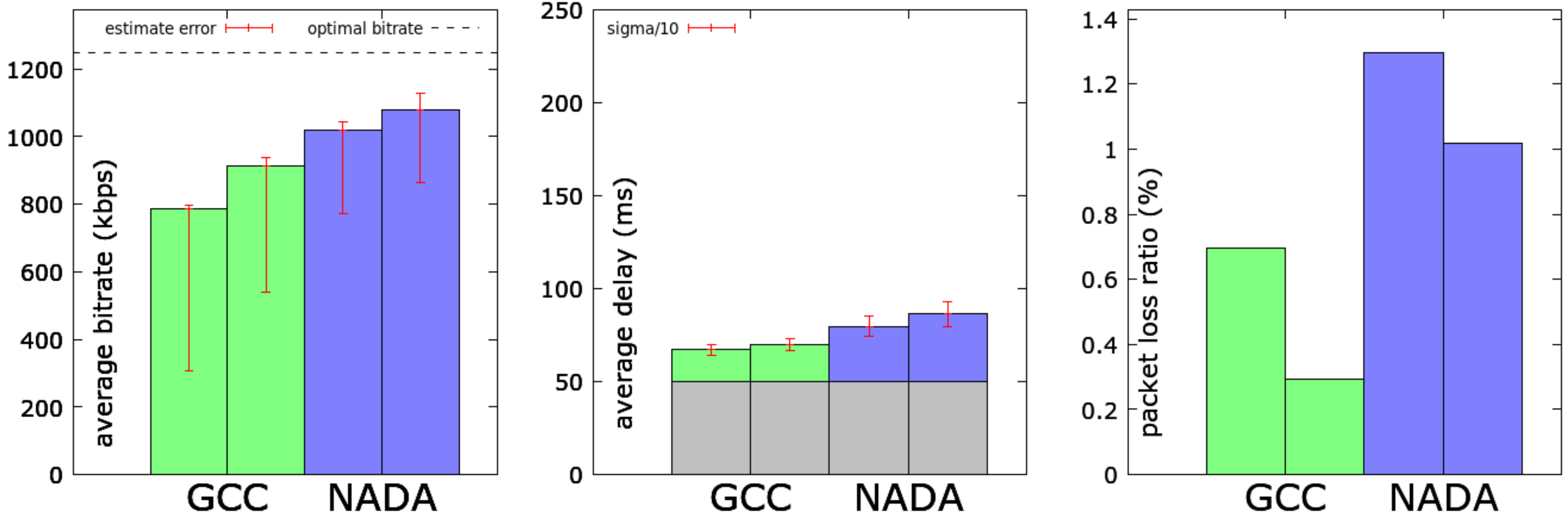
GCC	NADA
Channel utilization	
79.5%	76.9%
Queuing delay (ms) 5% percentile - mean - 95% percentiles	
4 - 22.9 - 42	5 - 26.6 - 49
Loss ratio	
0.79%	0.48%

Less jitter is beneficial to NADA. NADA still isn't able to fully utilize the link at 2500 kbps, possibly due to missing rate shaping buffer. Otherwise comparable.

Evaluation test 5.2

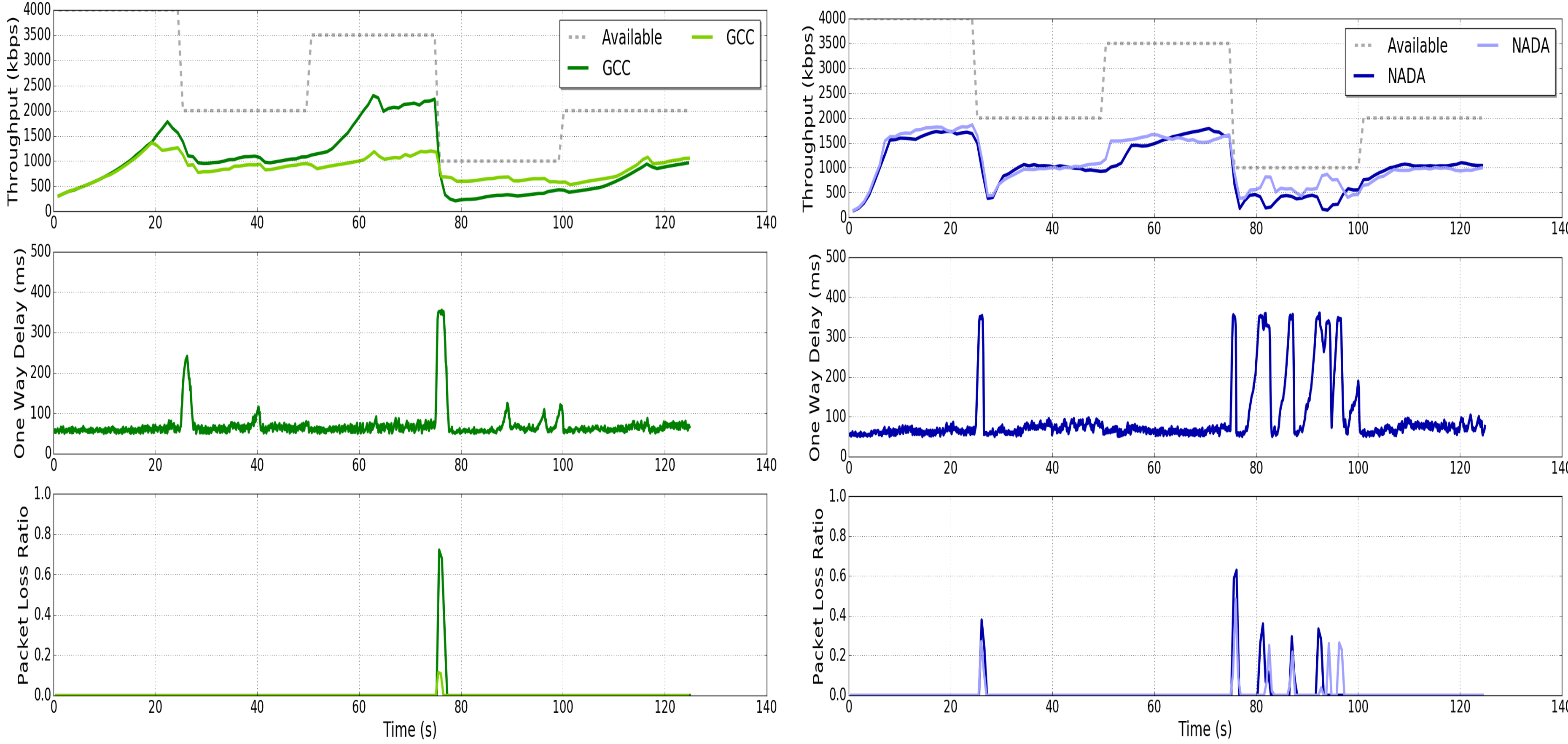
Two RMCAT flows, variable link capacity:





NADA experiences a bit higher packet loss due to the low capacity section.

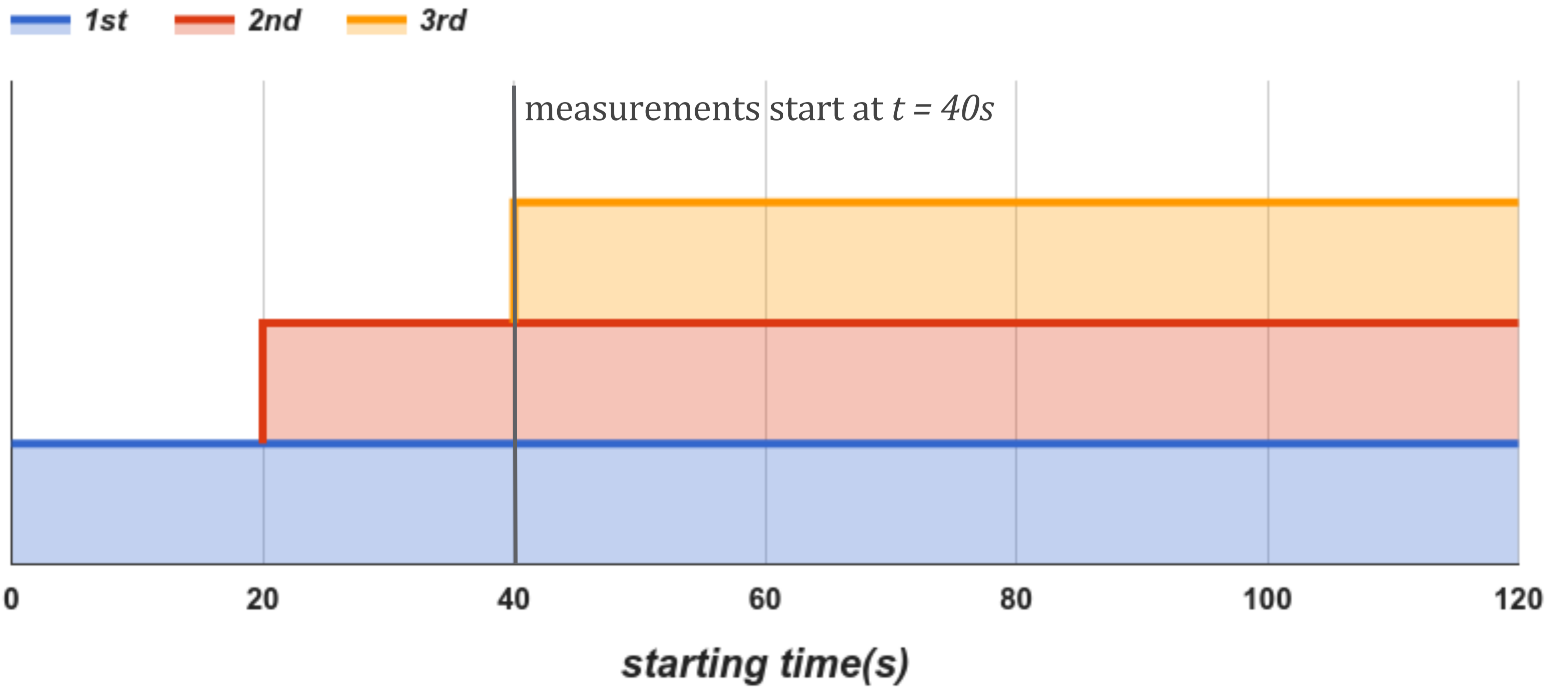
Evaluation test **5.2** *Jitter model* ✶



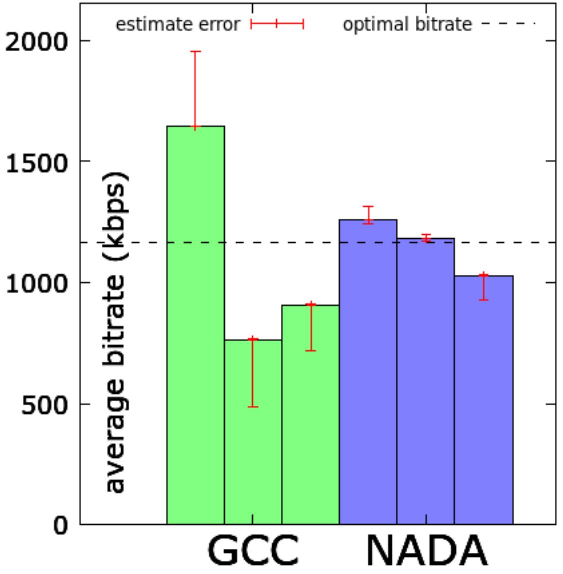
NADA becomes unstable when the bottleneck drops to 1000 kbps. A lower max bitrate helps avoid this. Otherwise the proposals are comparable.

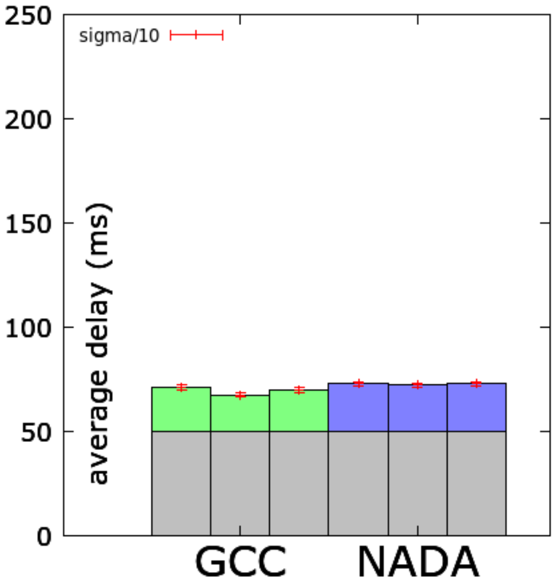
Evaluation test 5.4

Self-fairness: Three RMCAT flows, starting **20s** apart



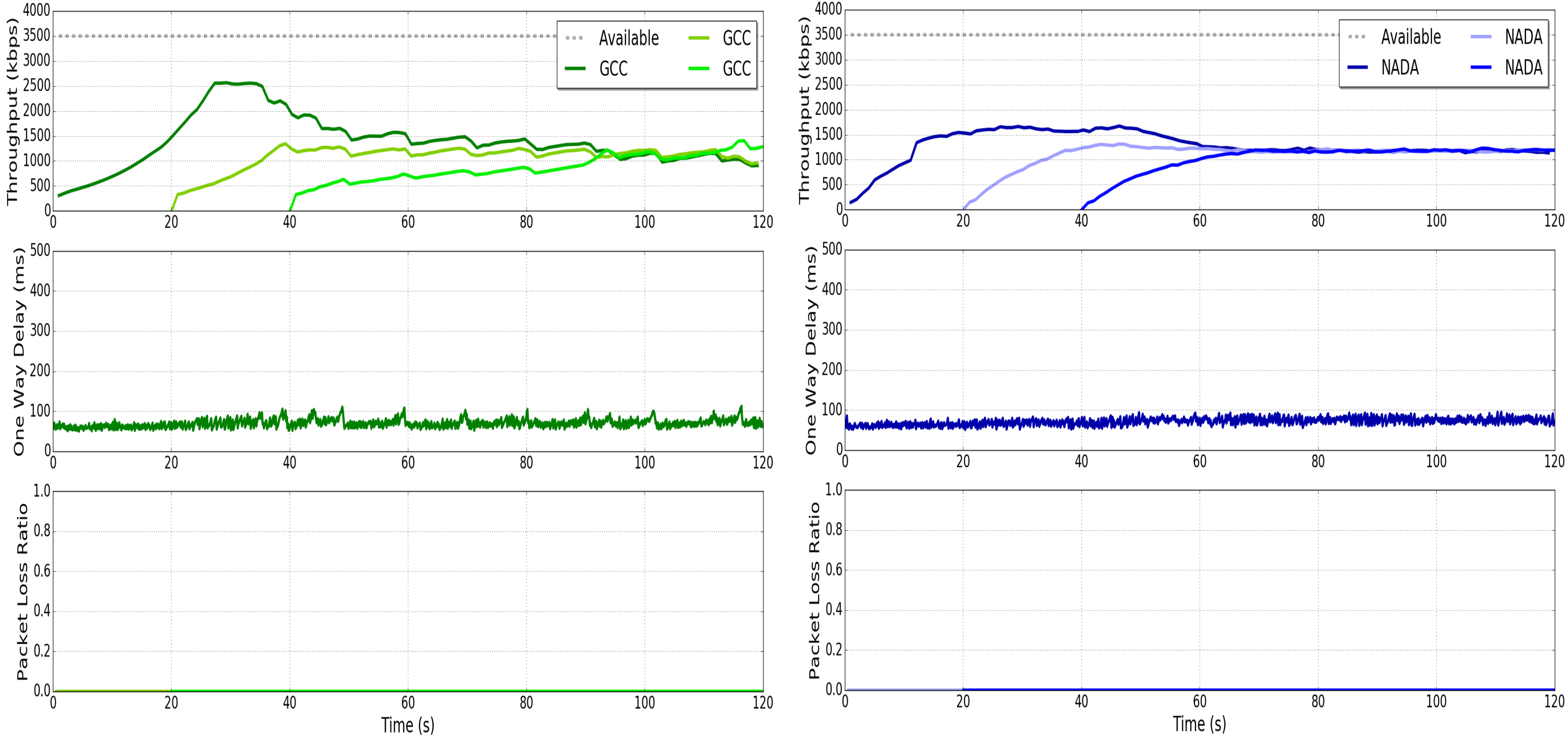
Evaluation test **5.4** *Jitter model* ✴





The optimal bitrate line corresponds to the perfect fair share.
Slower GCC convergence gives advantage for the first flow.

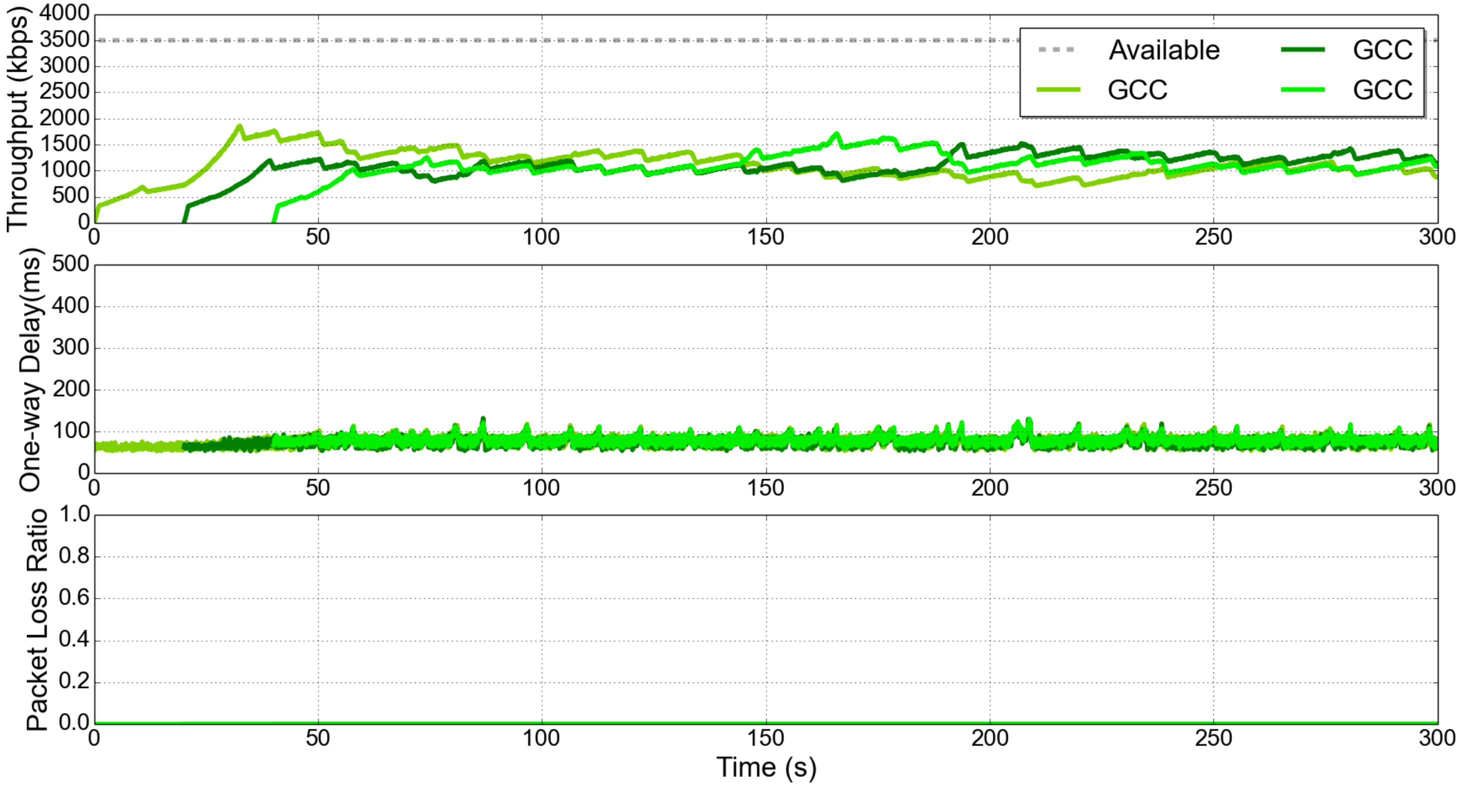
Evaluation test 5.4 *Jitter model* ✶



GCC converges more slowly. Notable that no single NADA flow goes above 2 Mbps.

Google

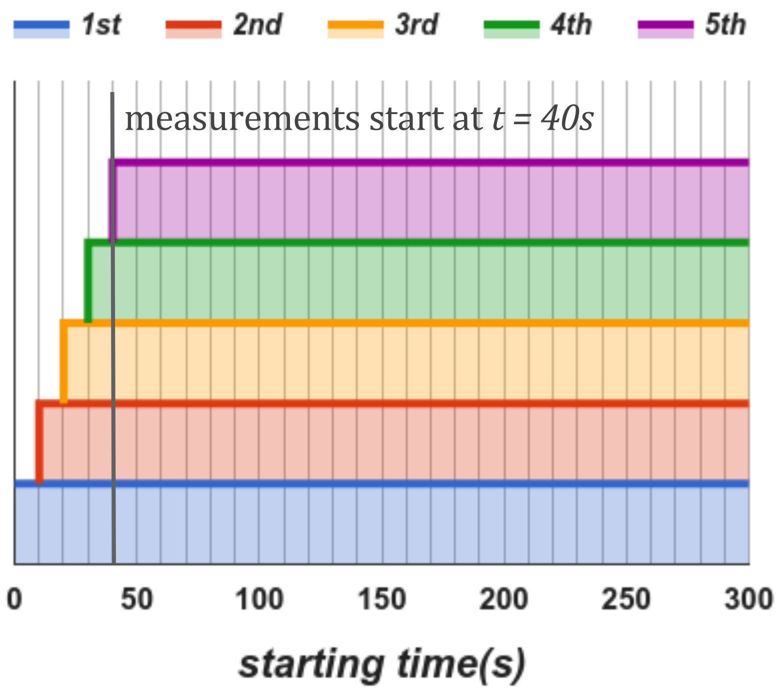
Evaluation test **5.4** *Jitter model* ✶

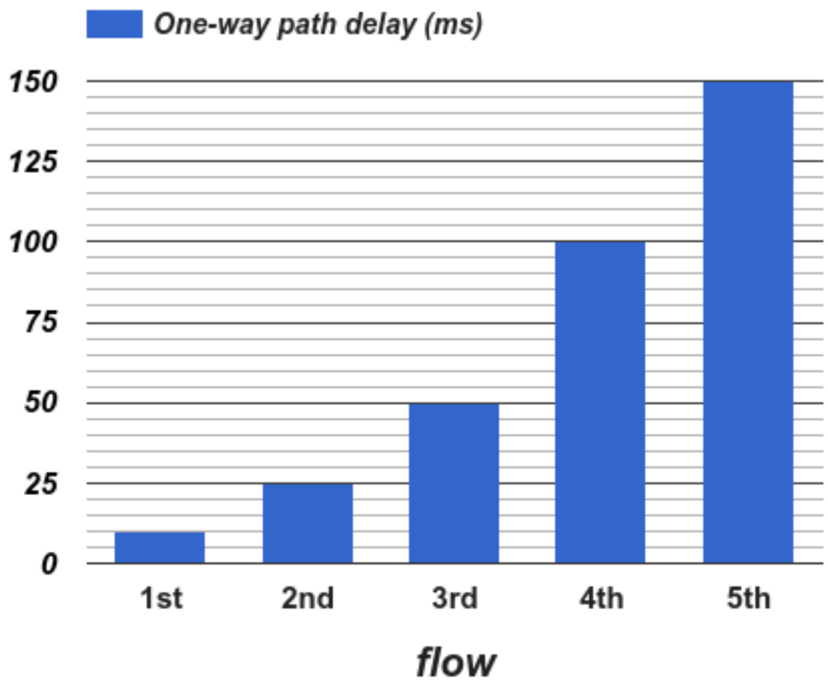


Longer run to show that the flows actually converge to something fair.

Evaluation test 5.5

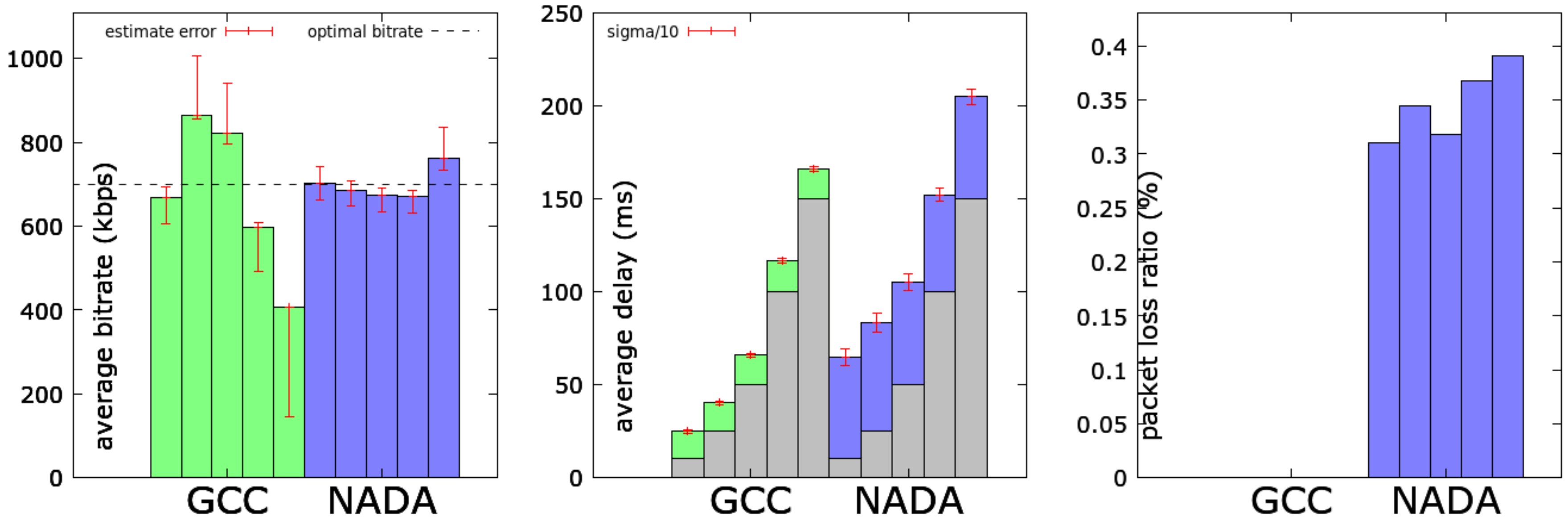
Round-trip fairness: Five RMCAT flows, starting **10s** apart. **Distinct RTTs**





Constant link capacity = ***3500 kbps***

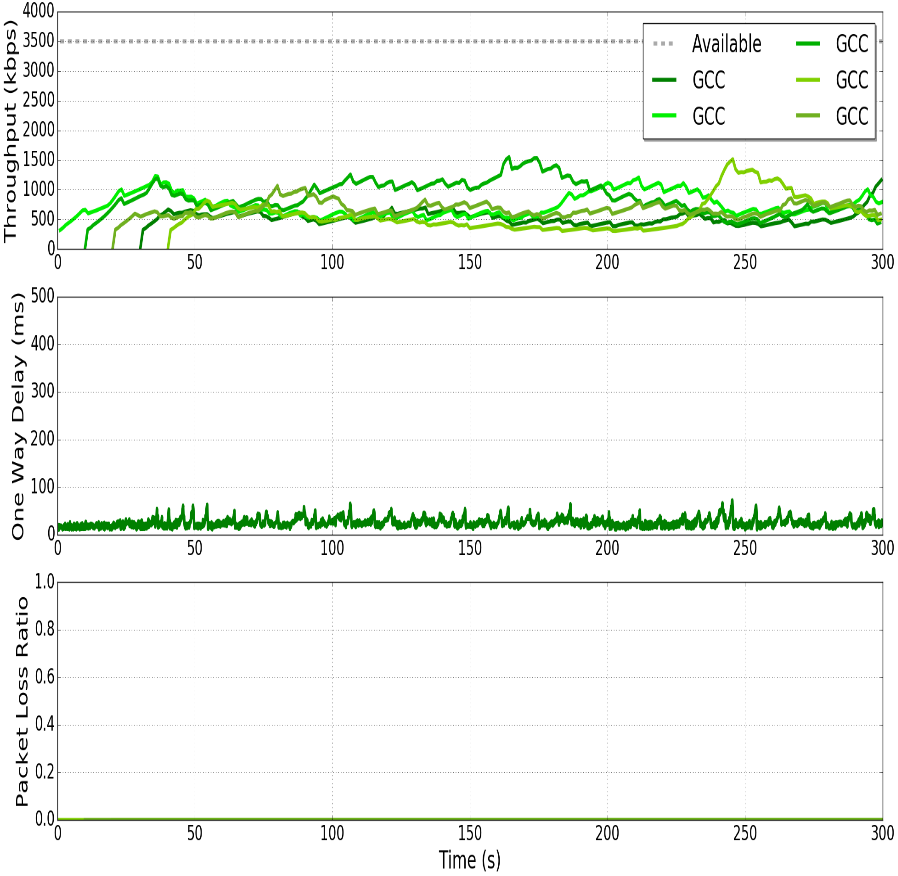
Evaluation test **5.5** *Jitter model* ✶

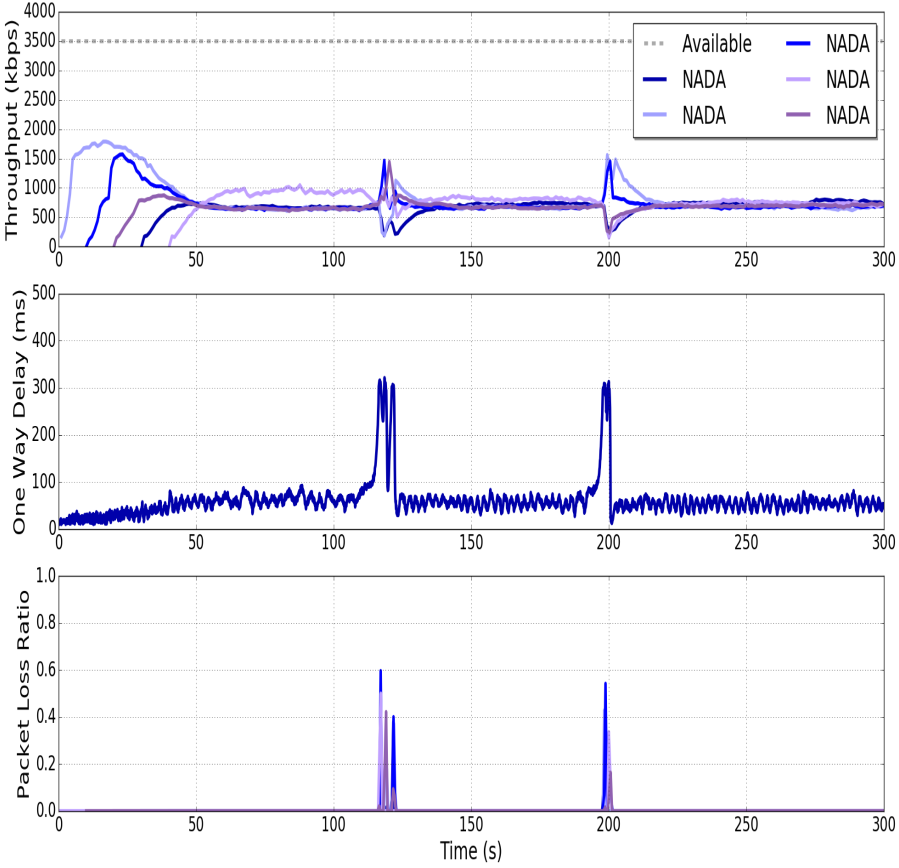


The optimal bitrate line corresponds to the perfect fair state, NADA flows share the link more equally.
A bit lower delay and loss rate for GCC.

Evaluation test **5.5** *Jitter model* ✴

Delay is plotted for the **first** flow, one-way path delay = **10ms**

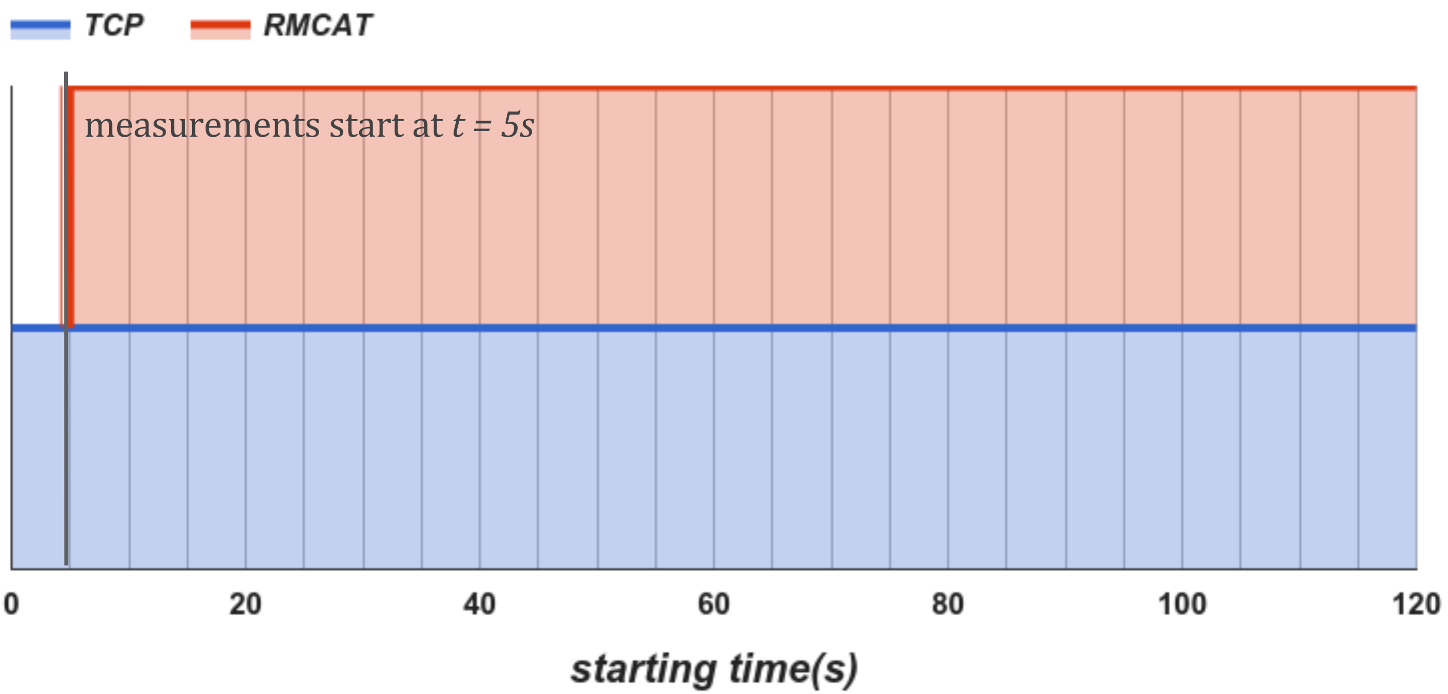




NADA converges better, but shows instability at times.

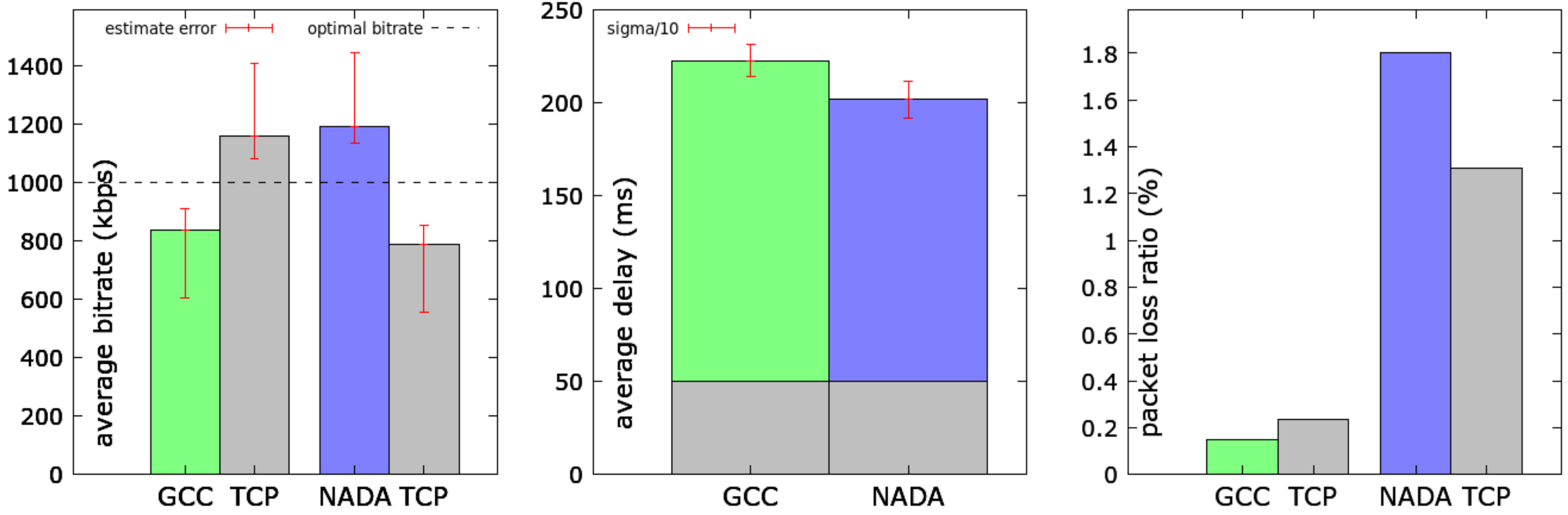
Evaluation test 5.6

Long TCP fairness: One TCP and one RMCAT flow, starting **5s** apart.

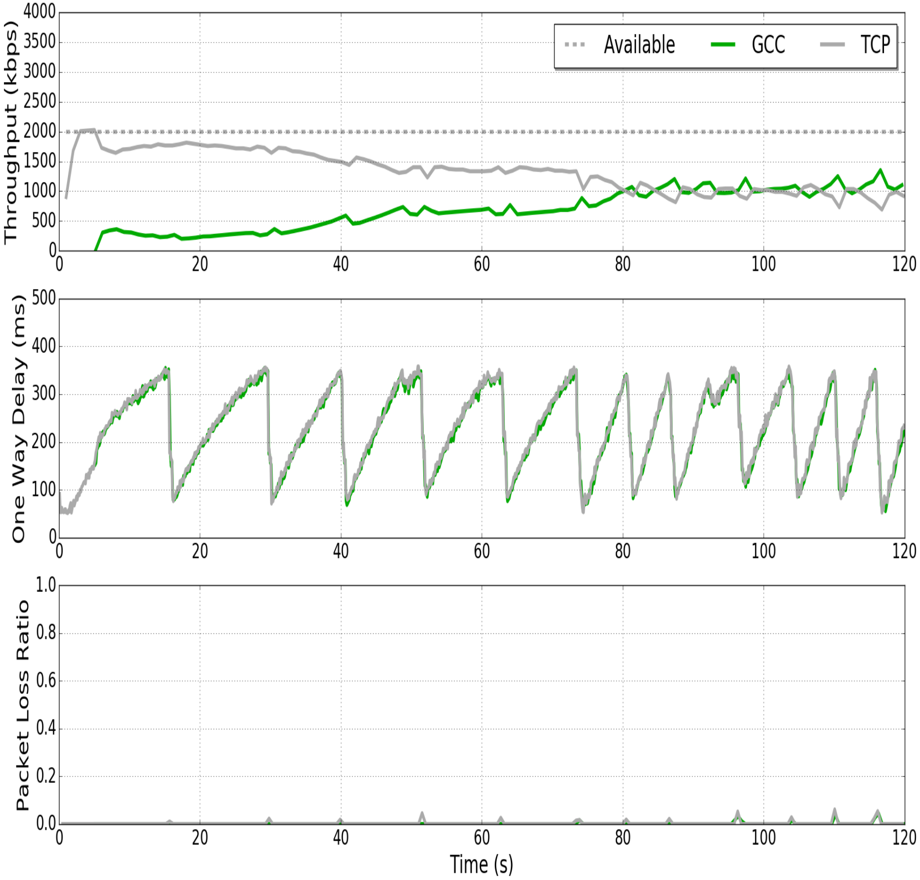


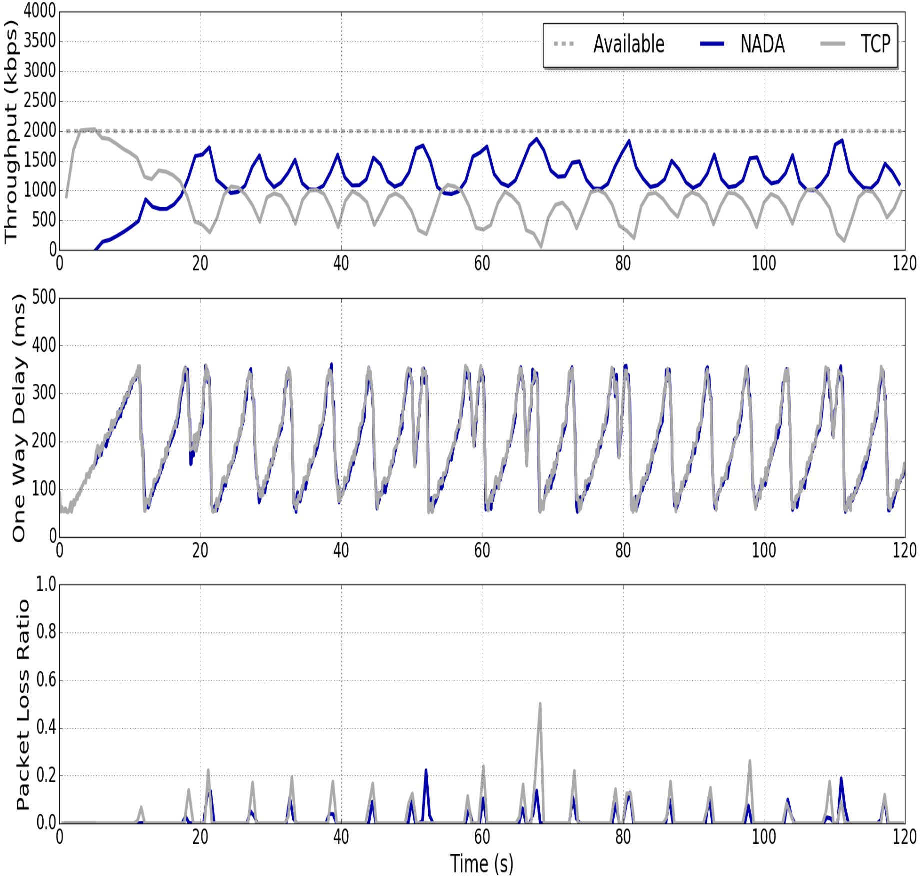
Google

Evaluation test **5.6** *Jitter model* ✶



Evaluation test **5.6** *Jitter model* ✶

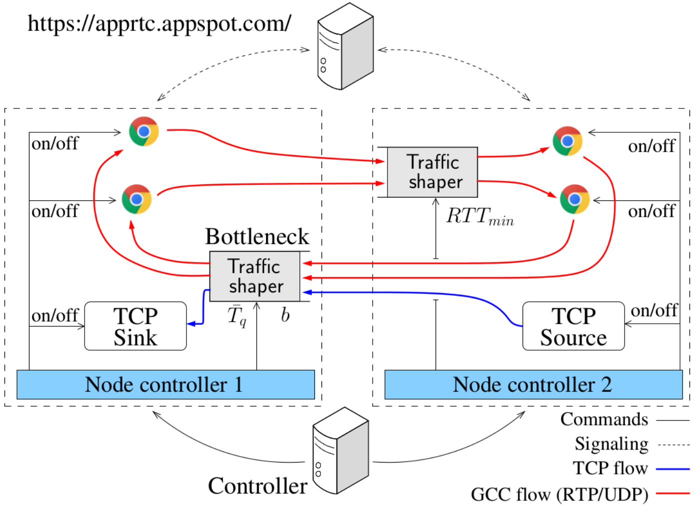




GCC converges more slowly but oscillates less.

Emulator: TestBed settings

- Chromium version: **M45**
- Video Encoder: VP8
- Video sequence: fourpeople_1280x720_30.yuv
- Max-Min Video Encoder bitrate range: 50-2000 kbps
- Signalling: https://apprtc.appspot.com/



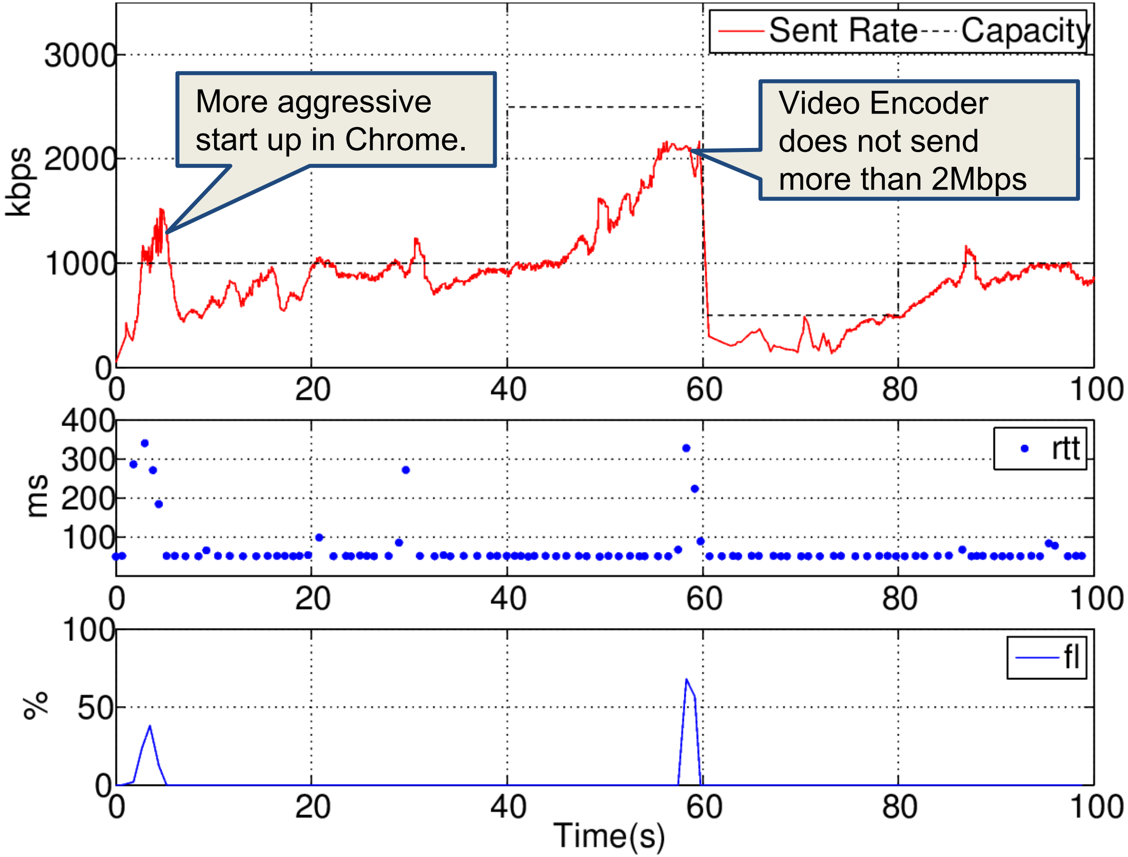
WAN Emulation:

- iproute2: tc+tbf module to set link capacity *b* constraint and buffer *Tq* size on Node 1
- NetEm to set propagation delay on Node 2

Bottleneck parameters:

- Drop tail
- Queueing size: 300ms
- Min one-way path latency: 25ms

GCC Emulation - Evaluation test **5.1**



Channel Utilization

84%

Queuing (ms)

5% percentile - mean - 95% percentile

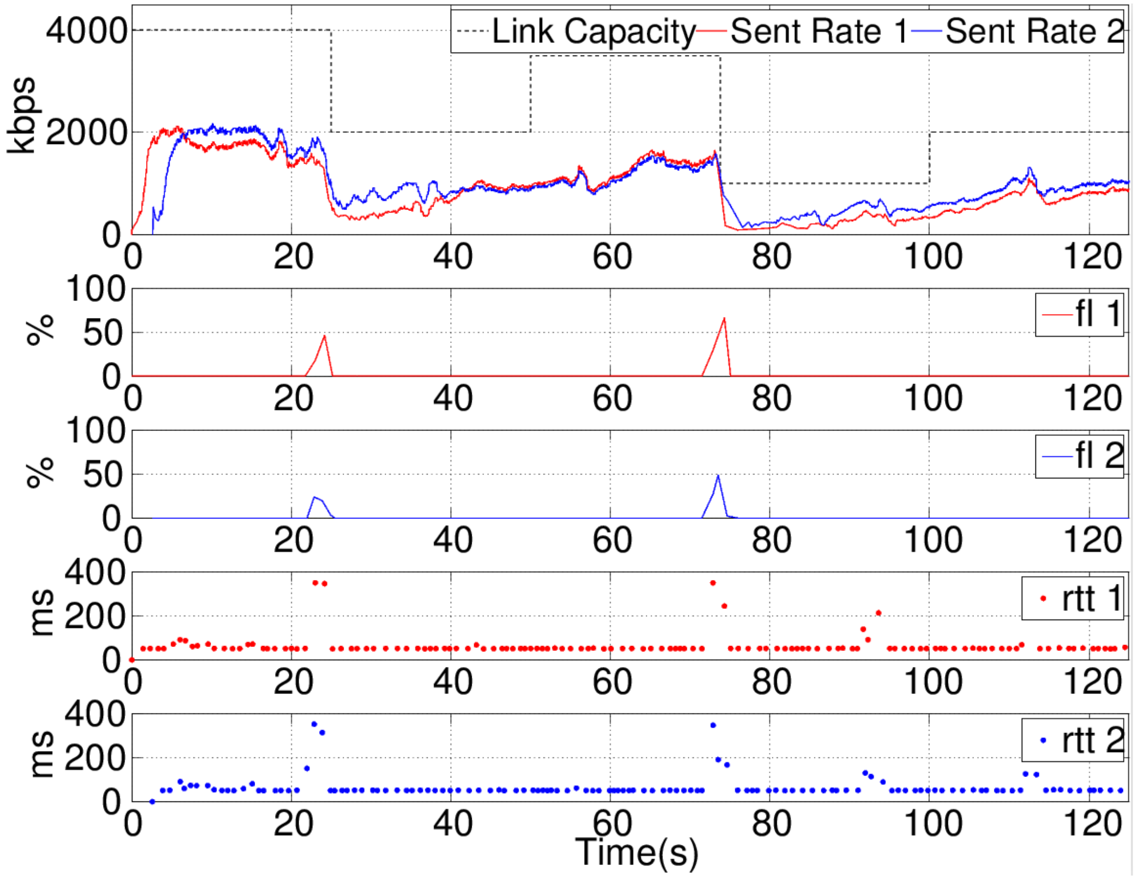
1	20	195

Loss percentage

0.02%

Google

GCC Emulation - Evaluation test 5.2



Channel Utilization

flow 1	43%
flow 2	45%

Queuing (ms)

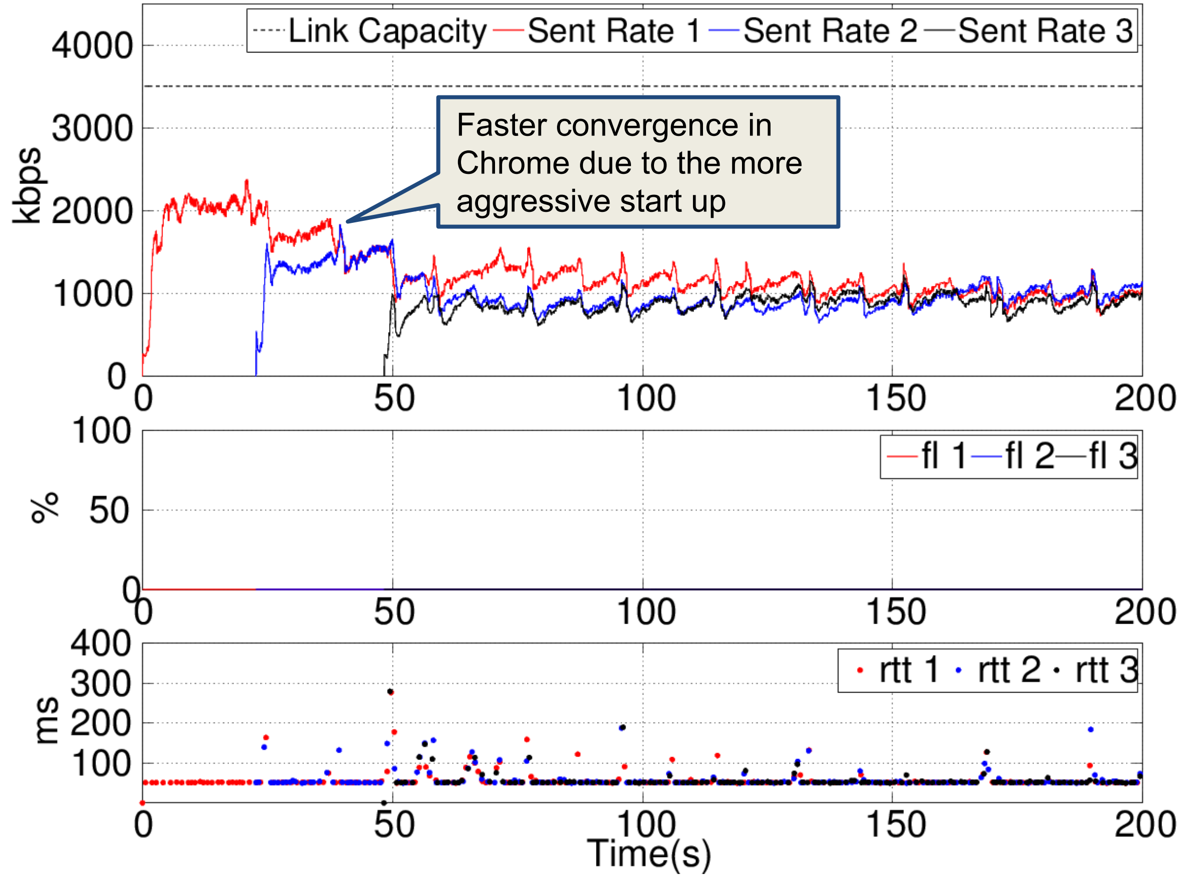
5% percentile - mean - 95% percentile

flow 1	0	15	96
flow 2	0	15	103

Loss percentage

flow 1	1%
flow 2	0.95%

GCC Emulation - Evaluation test 5.4



Channel Utilization

flow 1	30%
flow 2	29%
flow 3	28%

Queuing (ms)

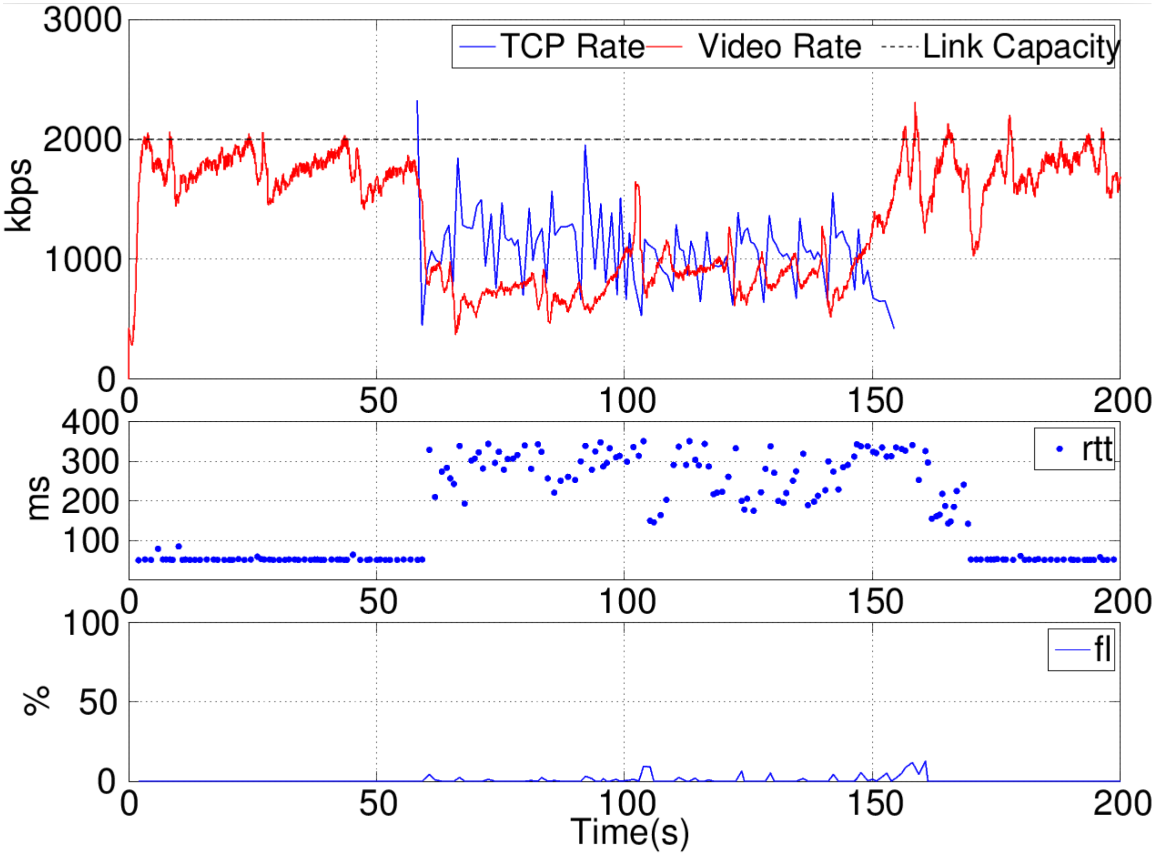
5% percentile - mean - 95% percentile

flow 1	1	10	66
flow 2	1	9	61
flow 3	1	9	65

Loss percentage

flow 1	0.0%
flow 2	0.0%
flow 3	0.0%

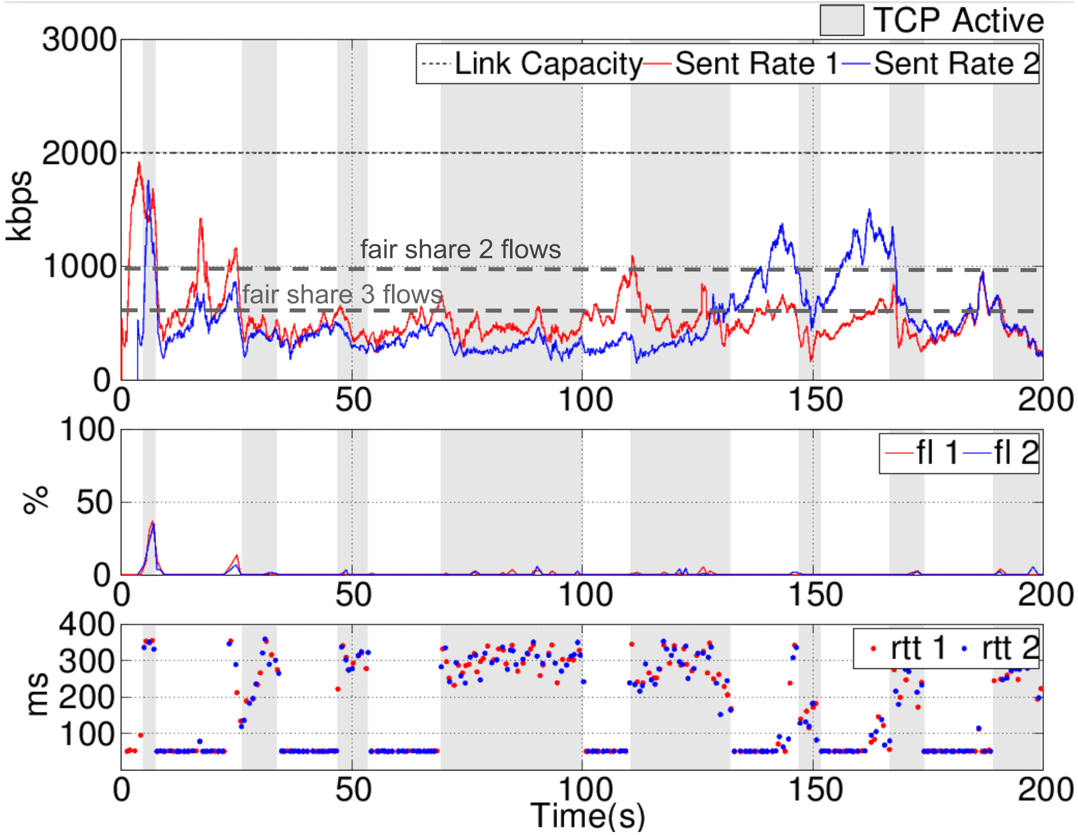
GCC Emulation - Evaluation test **5.6**



Channel Utilization	
Video	45%
TCP	53%
Queuing (ms) 5% percentile - mean - 95% percentile	
Video	125 225 293
Loss percentage	
Video	0.8%

In Emulation TCP-CUBIC is used. Metrics measured only when the two flows coexist.

GCC Emulation - Evaluation test **5.7**



Channel Utilization

flow 1	28%
flow 2	30%

Queuing (ms)

5% percentile - mean - 95% percentile

flow 1	1	109	290
flow 2	1	109	286

Loss percentage

flow 1	0.9%
flow 2	0.7%

The Emulation considers just one TCP flow that is turned on and off in the grey region.

Conclusions - GCC

- Appears to be comparable to NADA in performance.
- Resilient to jitter and does not require tuning of (max/min) bitrates.
  - Important for SFU/MCU implementations.
- In general slower convergence than NADA in Simulation.
- Overall Simulation results are consistent with the Emulation results

Conclusions - NADA[1]

- Fast convergence.
- Low variance in bitrate after convergence.
- Sensitive to jitter, causing NADA to have trouble saturating bottlenecks.
- Stability issues at (relatively) low bottleneck capacities.
- Sensitive to the choice of min and max bitrates.
- With a rate shaping buffer NADA is expected to better utilize the link thanks to lower self-induced jitter. May cause longer delays on high bandwidth links.

[1)] Disclaimer: NADA was implemented from the -06 draft, so there can be bugs.

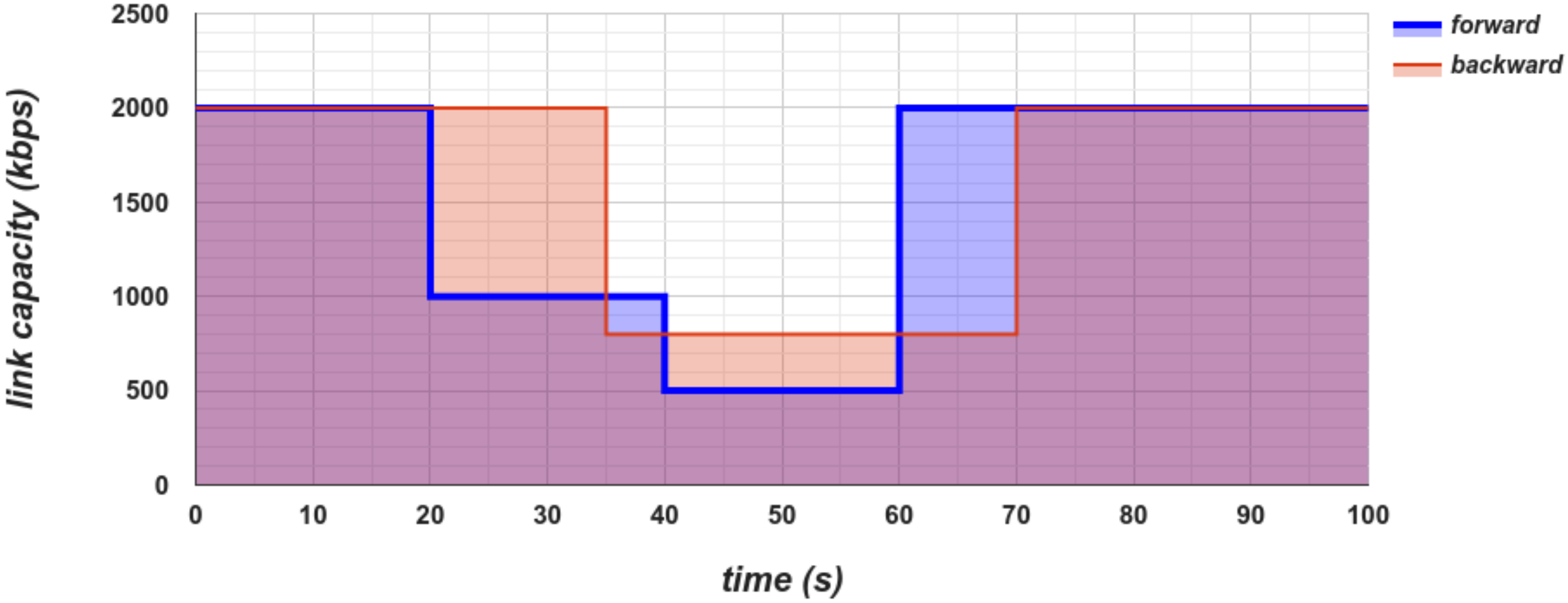
Open Issues and Future Work

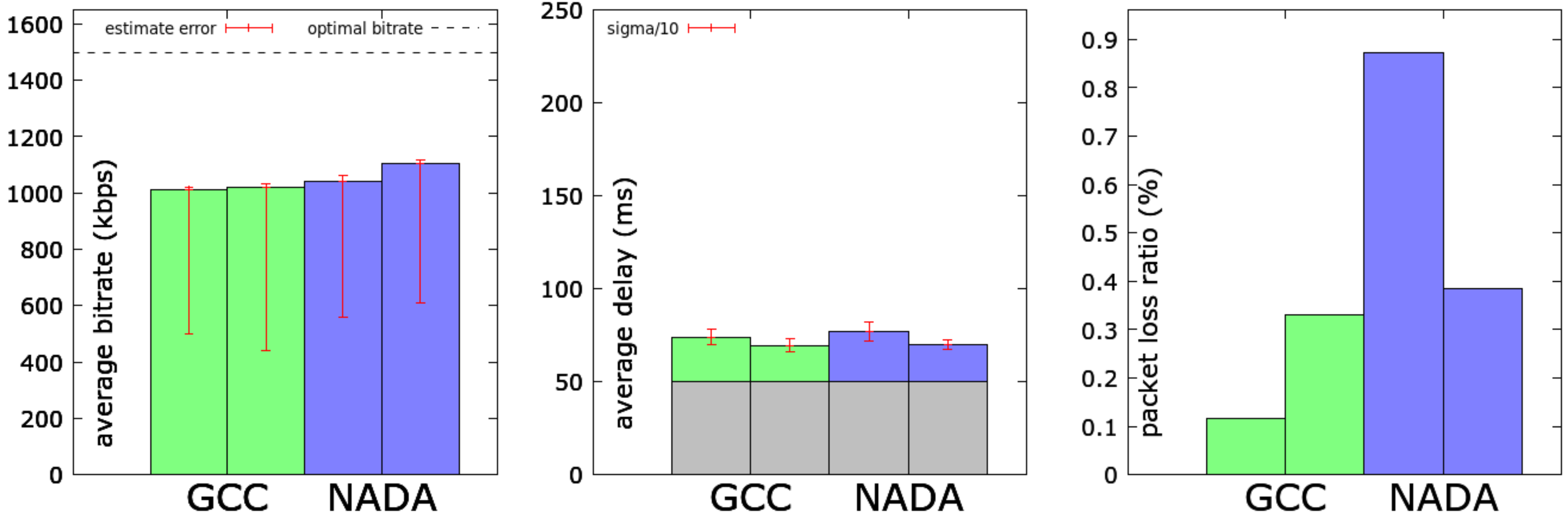
- Improve algorithm convergence.
- Move all the algorithm logic to the sender.
- Improve the loss-based controller.
- Start data collection in the wild.
- Evaluation over wireless networks.

BACKUP SLIDES

Evaluation test 5.3

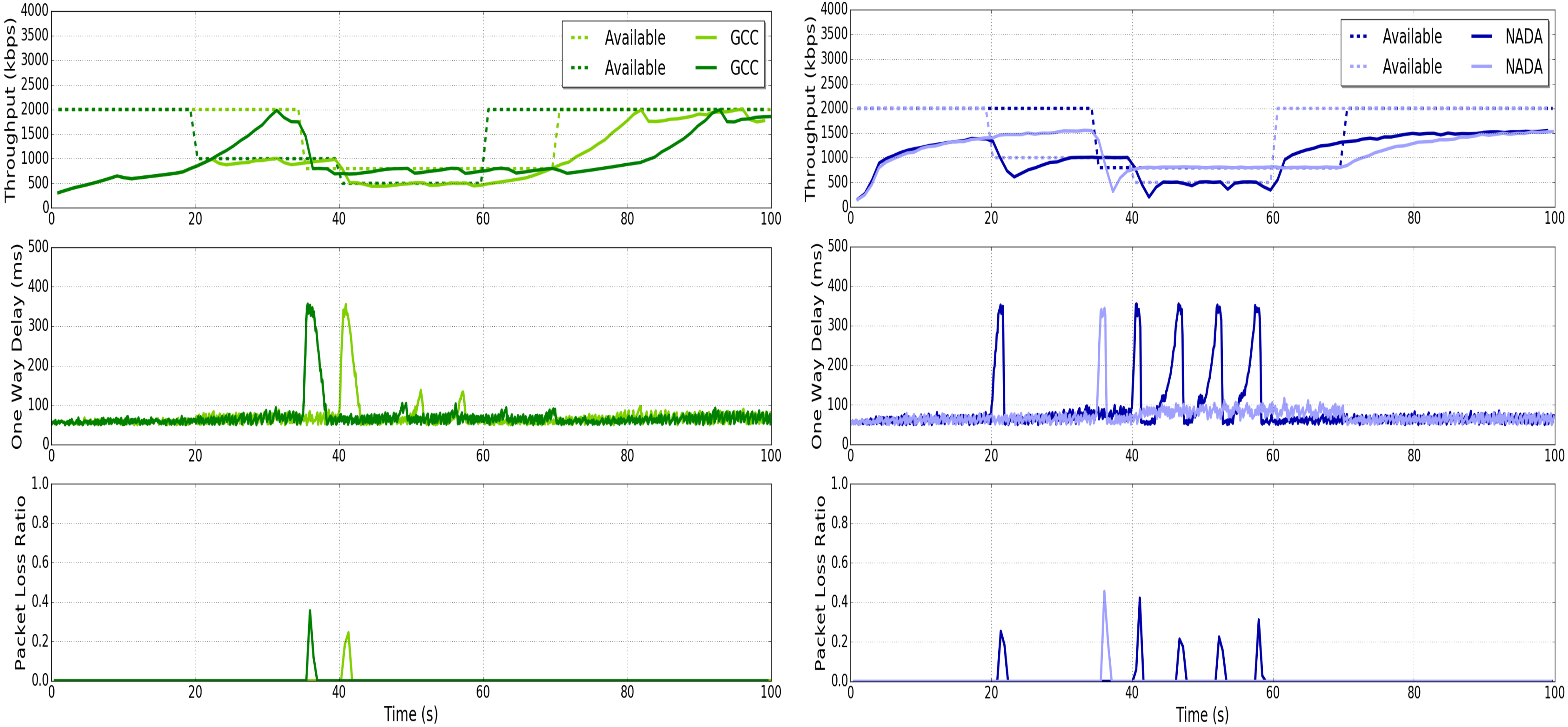
Bidirectional RMCAT flow, variable link capacity:





NADA experiences a bit higher packet loss due to the low capacity section.

Evaluation test **5.3** *Jitter model* ✶



NADA becomes unstable when the bottleneck drops to 1000 kbps. A lower max bitrate helps avoid this. Otherwise the proposals are comparable.

Evaluation test 5.7

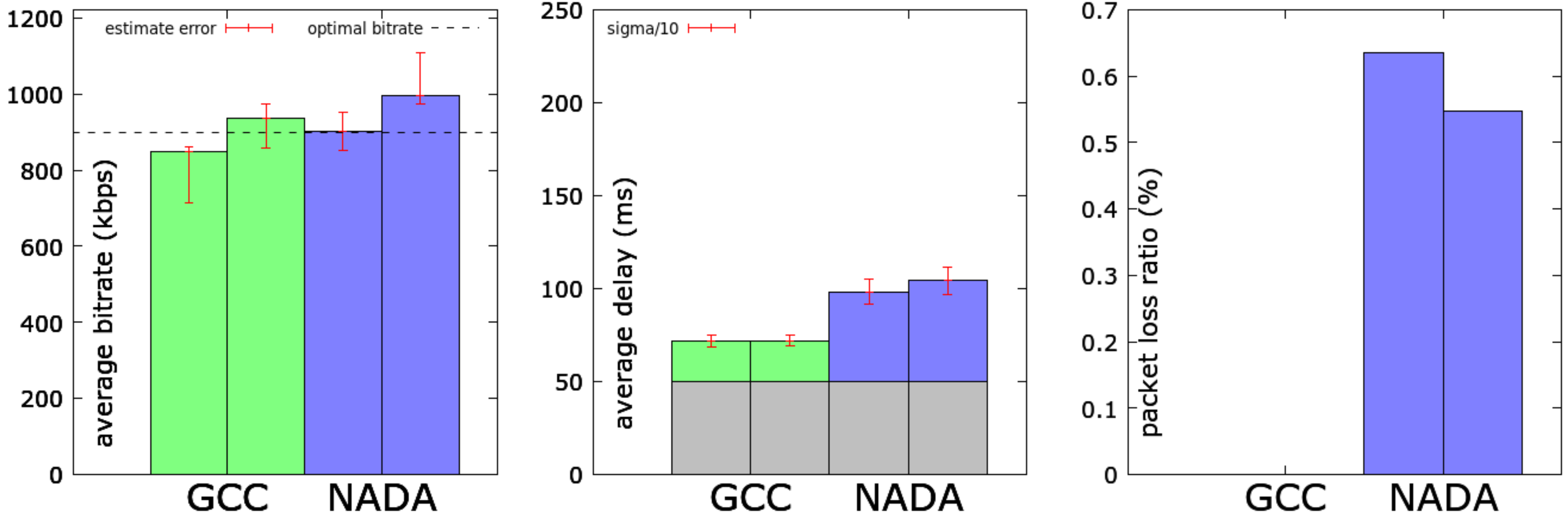
Short TCP fairness: Two RMCAT and 10 short-lived TCP flows.

TCP **file size** (each)	Uniform distribution between [100,1000] kbytes

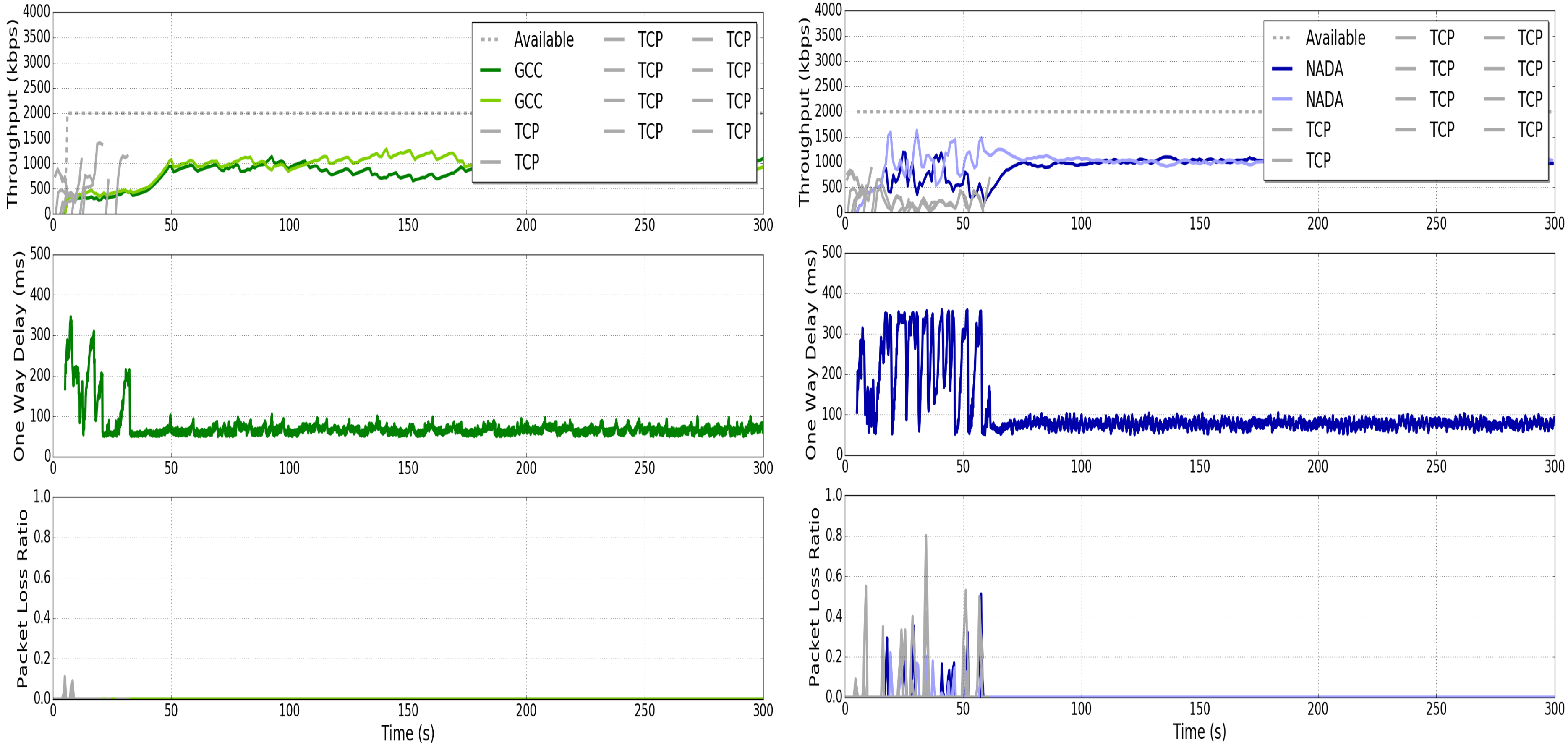
Number of TCP flows	**Starting time**
2	$t = 0$
8	t chosen from exponential distribution with mean $\mu \equiv \lambda^{-1} \coloneqq 10s$

Both random ***size*** and ***starting time*** samples are the same for GCC and NADA

Evaluation test **5.7** *Jitter model* ✶



Evaluation test **5.7** *Jitter model ✴*

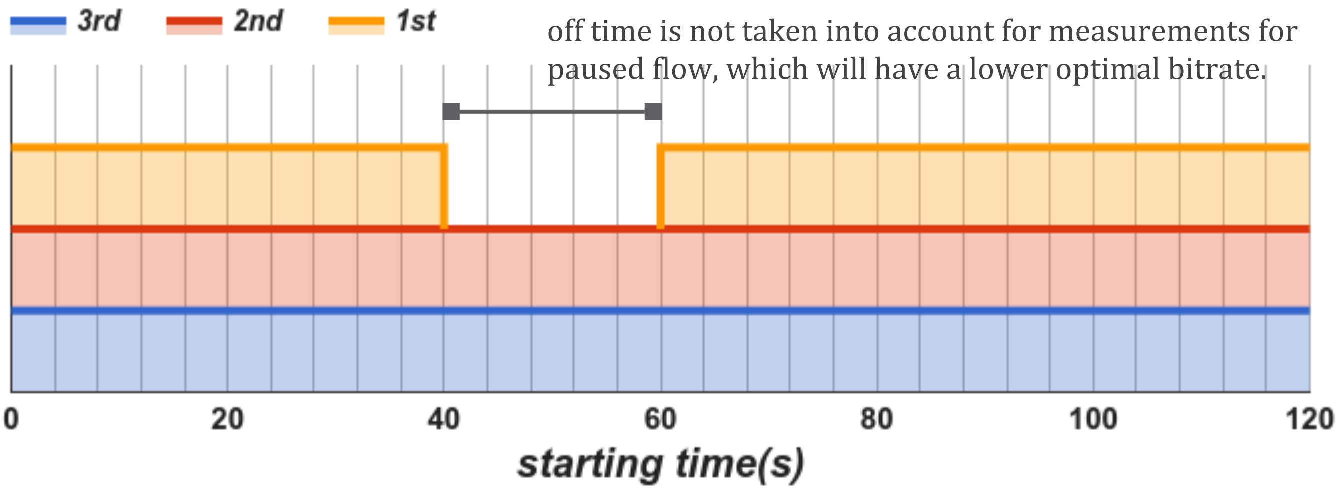


The randomization of the TCP flows is the same, but they finish faster when competing with GCC.

Evaluation test 5.8

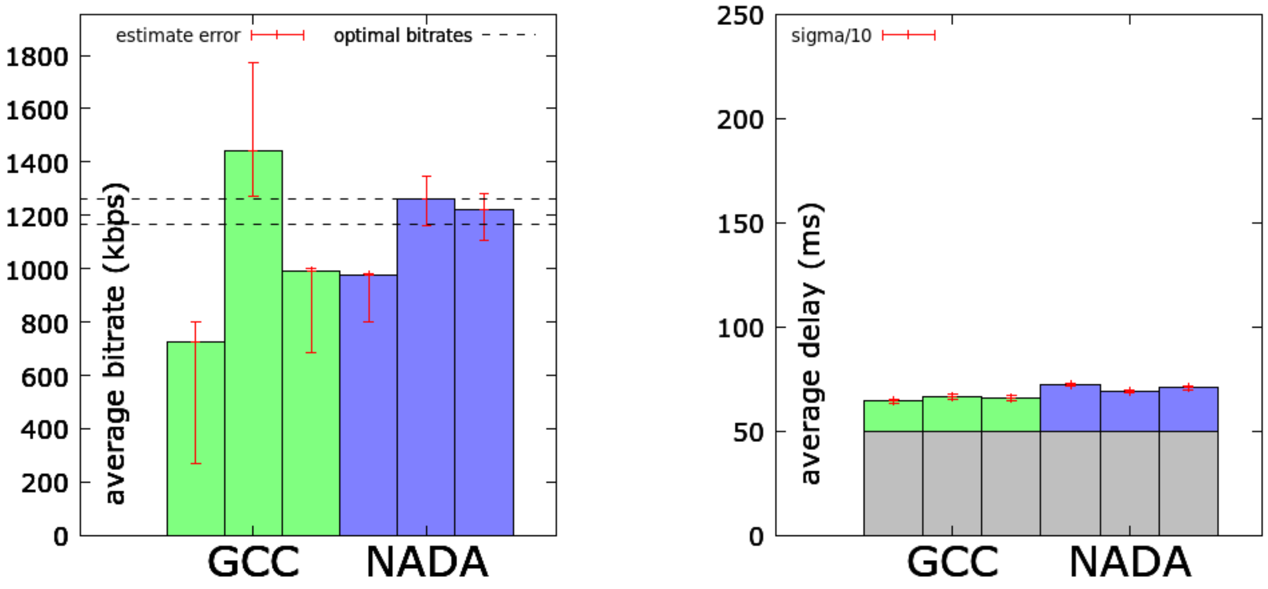
Pause and resume media: Two continuous and one intermittent RMCAT flows

off time is not taken into account for measurements for paused flow, which will have a lower optimal bitrate.



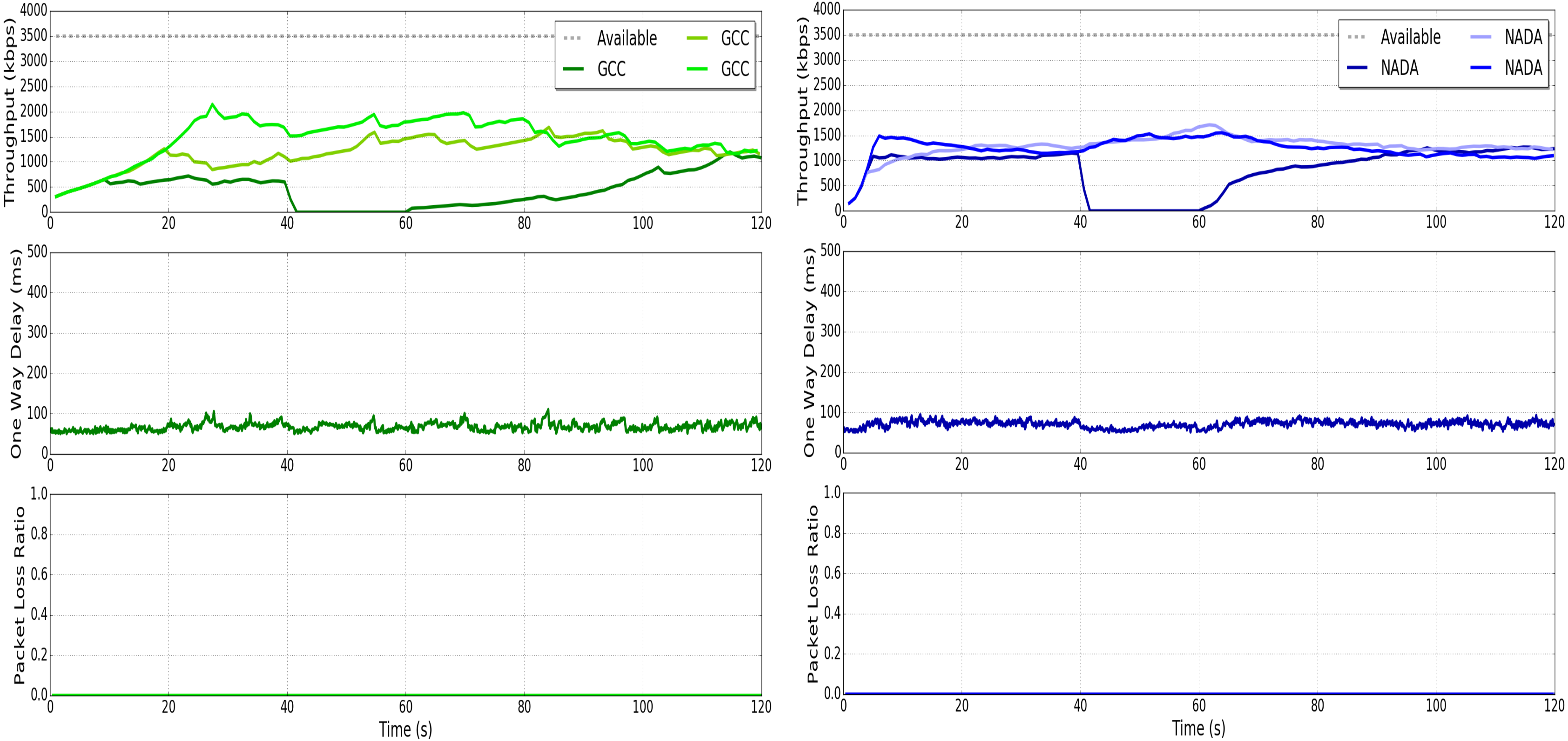
Constant link capacity = ***3500 kbps***

Evaluation test **5.8** *Jitter model* ✴



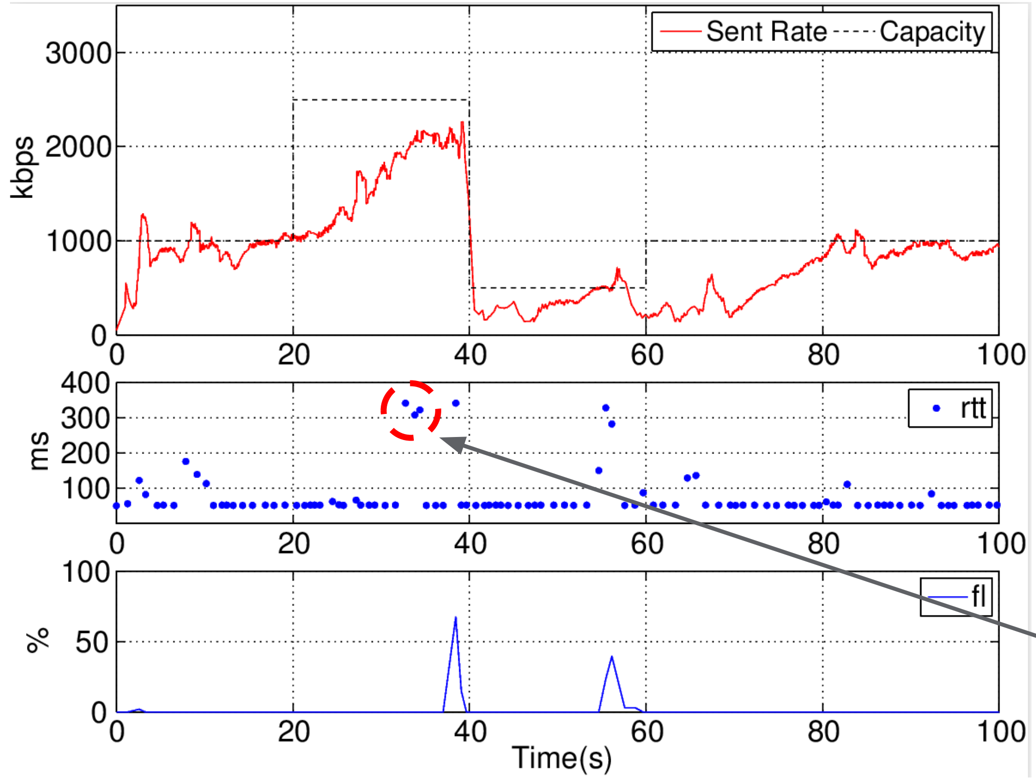
The lower optimal bitrate corresponds to the first (paused) flow.

Evaluation test **5.8** *Jitter model* ✶



Slower convergence for GCC.

Forward Path



rtt inflation due to backward path congestion

Channel Utilization

88%

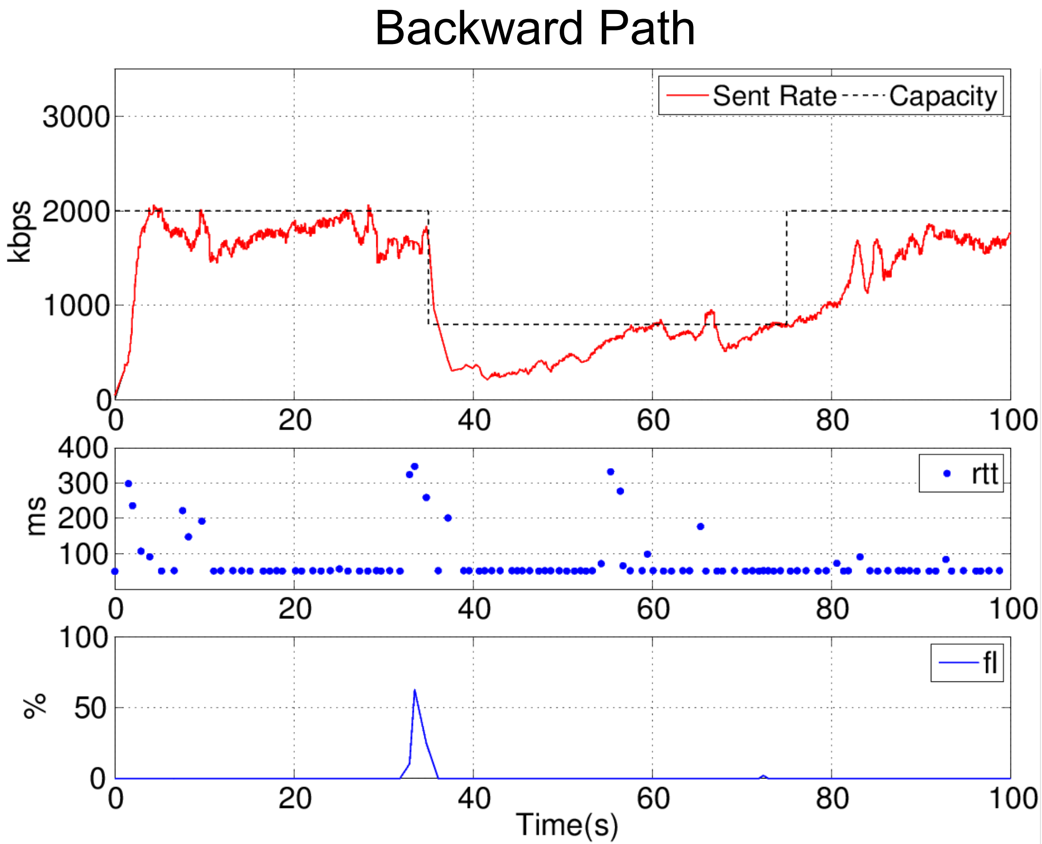
Queuing (ms)
5% percentile - mean - 95% percentile

0 10 180

Loss percentage

0.01%

GCC Emulation - Evaluation test 5.3



Channel Utilization

89%

Queuing (ms)

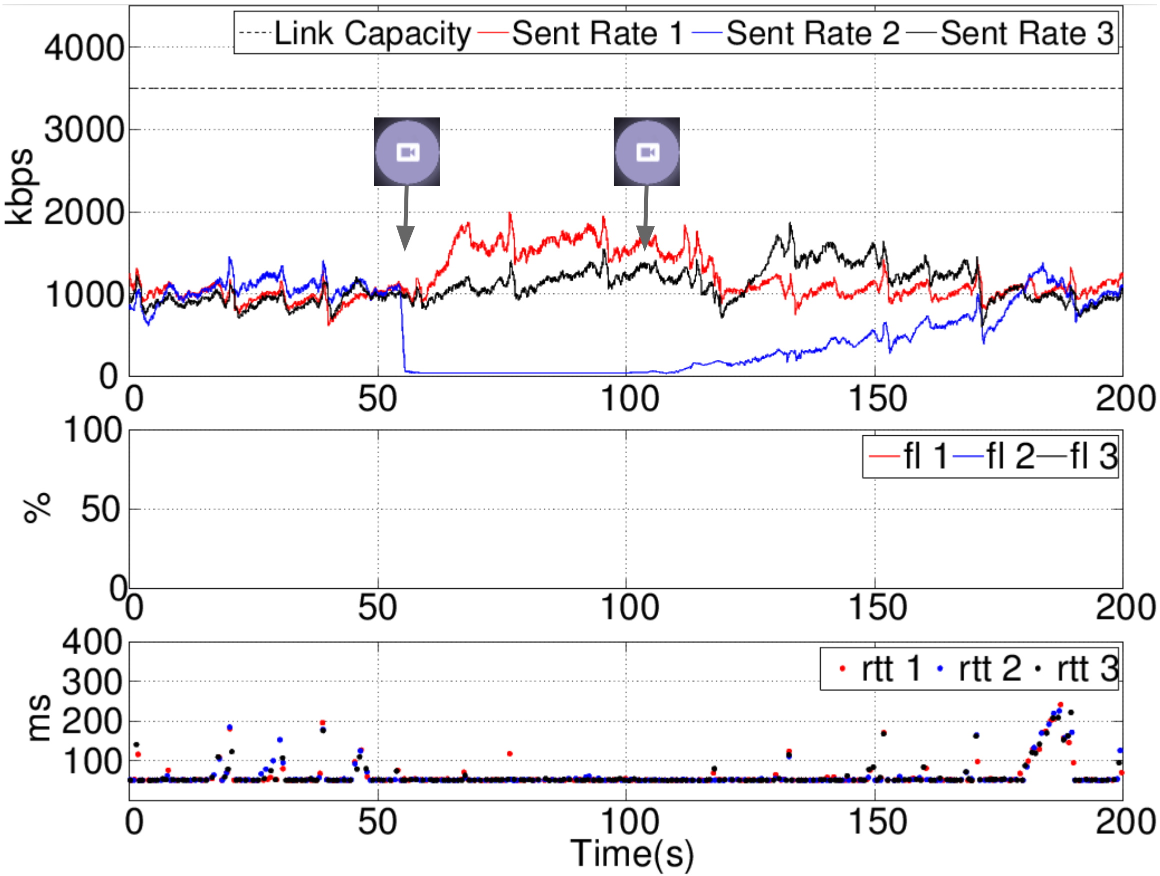
5% percentile - mean - 95% percentile

0	18	159

Loss percentage

0.01%

GCC Emulation - Evaluation test 5.8



Channel Utilization

flow 1	35%
flow 2	22%
flow 3	34%

Queuing (ms)

5% percentile - mean - 95% percentile

flow 1	0	10	74
flow 2	1	10	70
flow 3	1	11	72

Loss percentage

flow 1	0.0%
flow 2	0.0%
flow 3	0.0%

In Emulation the video flow is stopped using WebRTC JavaScript API