RL-based Internet Congestion Control

Ashwin Shenai (180156), Astitva Chaudhary (180157)
EE698V: Advanced Topics in ML for Communication Networks
Fall 2022

Project Goals: Proposed

- Implement paper on Multi-Objective Congestion Control.
- Integrate MOCC with Orca and DeepCC (if time permits).
- Study performance of new architecture with different RL algorithms - DDPG, TD3, PPO, SAC, etc.

Pantheon

- Pantheon of Congestion Control: It is a community evaluation platform for academic research on congestion control that reduces the need to reinvent the wheel in the evaluations of new internet congestion-control algorithms.
- Pantheon provides:
  1) A collection of 17 working implementations of congestion-control schemes, all of them continuously verified to compile and run by a continuous integration system.

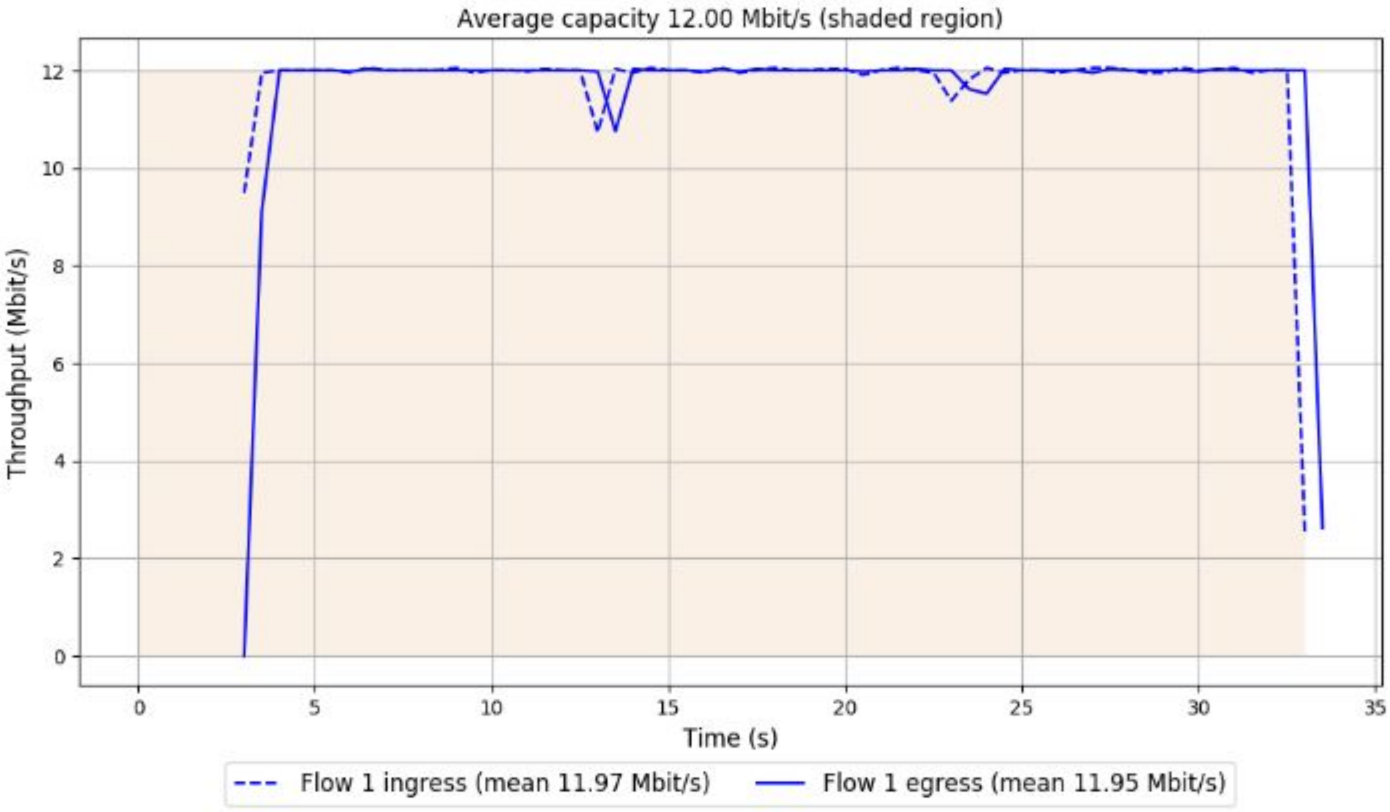
  2) A testbed of measurement nodes on wired and cellular networks.

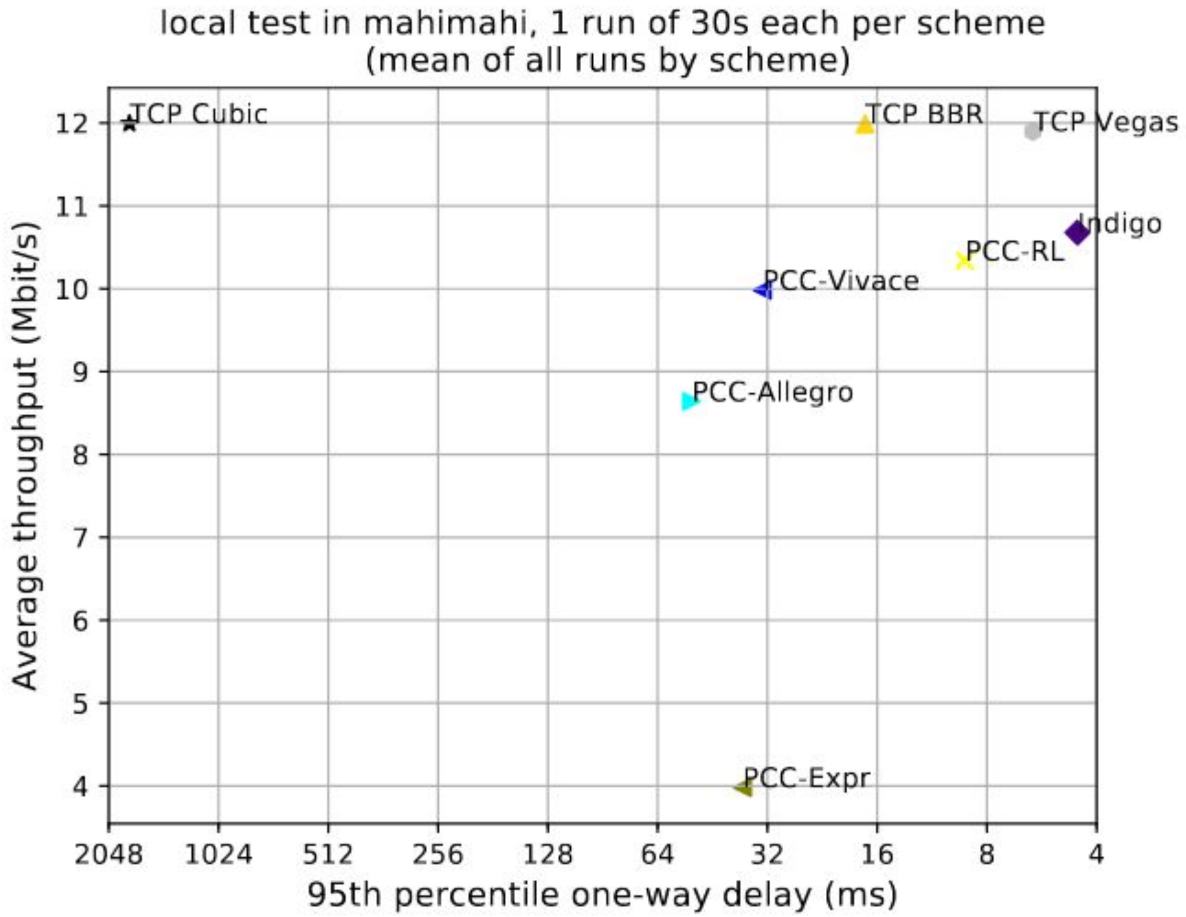
  3) A collection of network emulators that can be used to test congestion-control schemes locally.

- The Pantheon performs several types of measurements on a roughly weekly basis. All measurements run a particular congestion-control scheme between two endpoints, measuring the departure time of each IP datagram (at the sender) and the arrival time of the same IP datagram (at the receiver), if it arrives. These raw logs are available for each measurement. For each scheme, it also calculates and plots aggregate statistics, e.g., the throughput, one-way delay (95th percentile), loss rate, etc.
- To run a new scheme, submit a pull request, the Travis-CI system will automatically verify that the scheme compiles and runs in emulation.

Test 1

12 Mbps trace

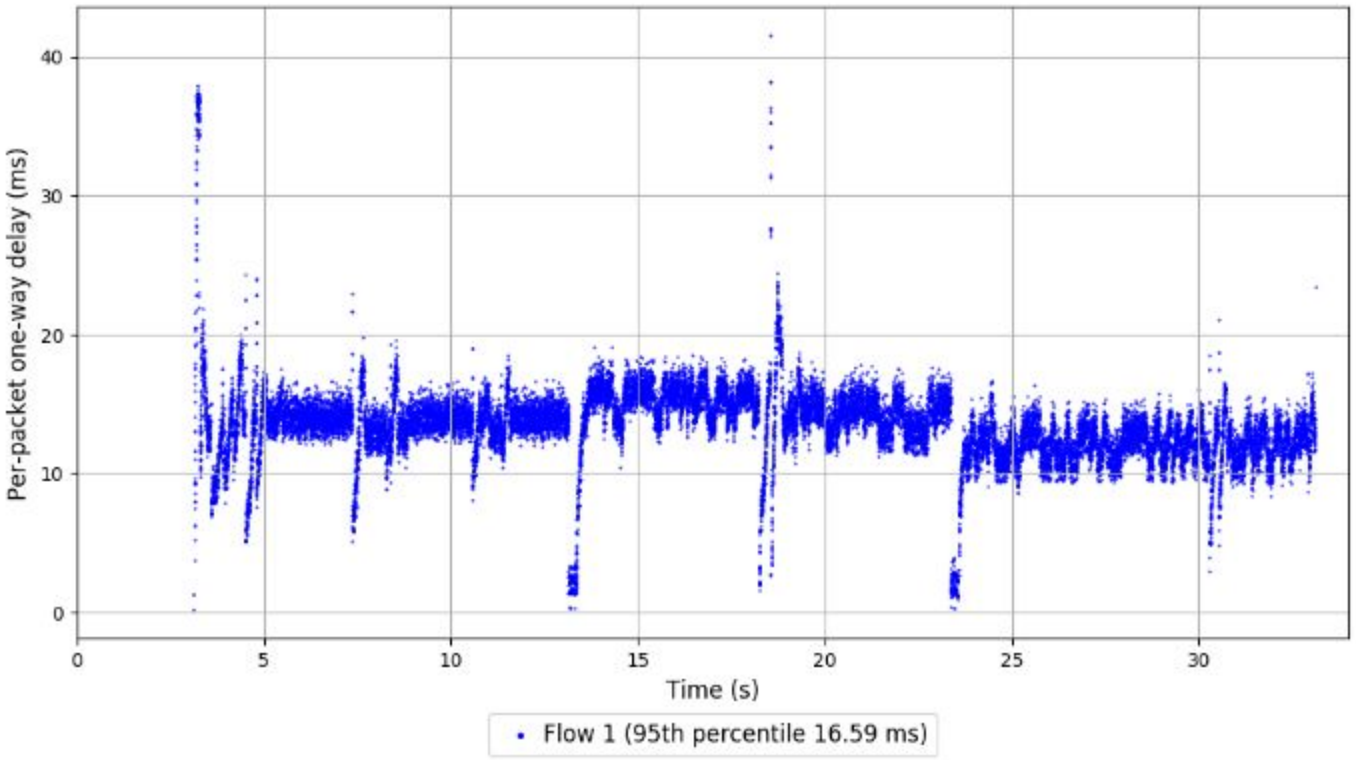


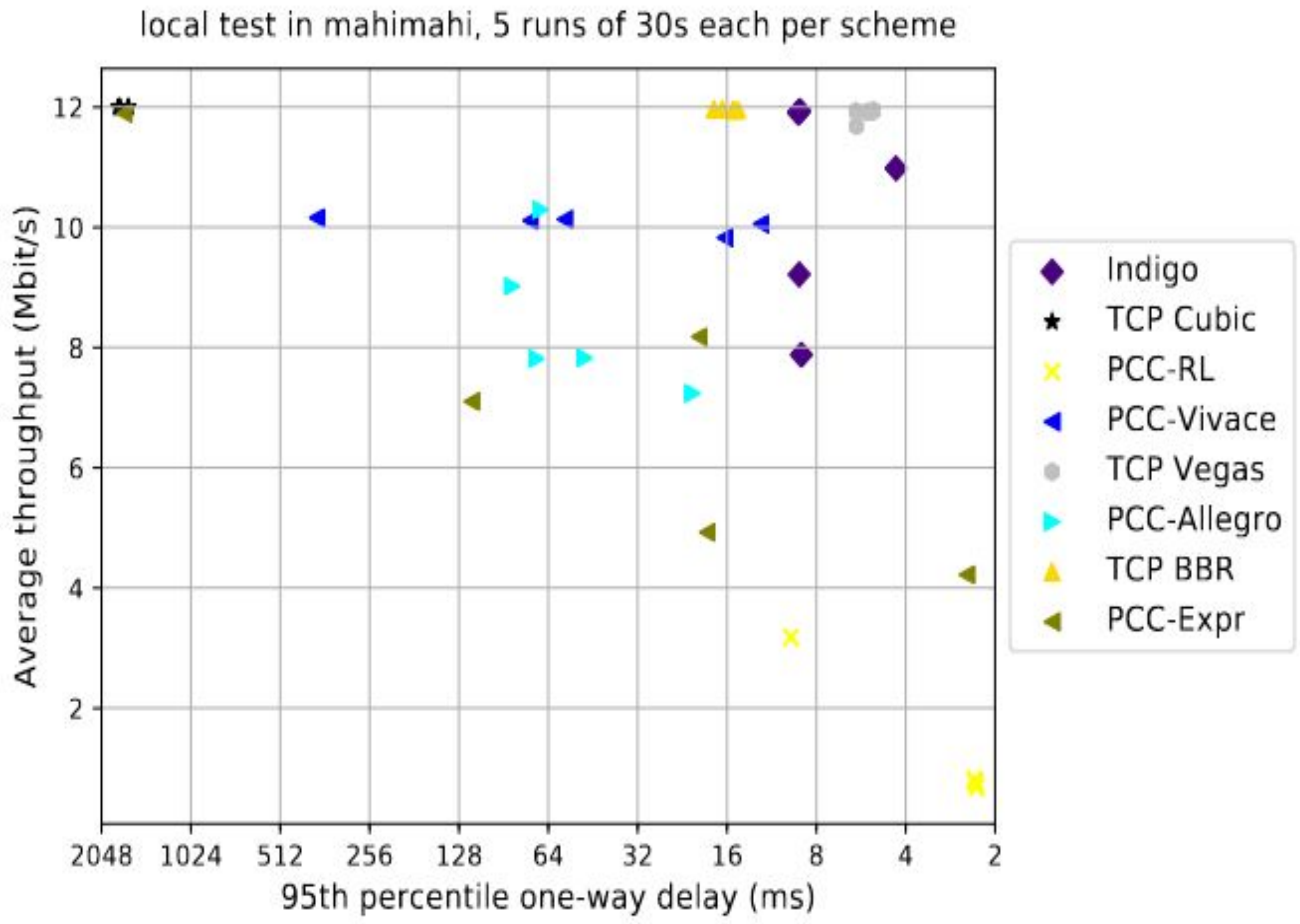


local test in mahimahi, 1 run of 30s each per scheme
(mean of all runs by scheme)
Average throughput (Mbit/s)
95th percentile one-way delay (ms)
12
11
10
9
8
7
6
5
4
2048
1024
512
256
128
64
32
16
8
4
TCP Cubic
TCP BBR
TCP Vegas
Indigo
PCC-RL
PCC-Vivace
PCC-Allegro
PCC-Expr

Test 2

Multiple runs

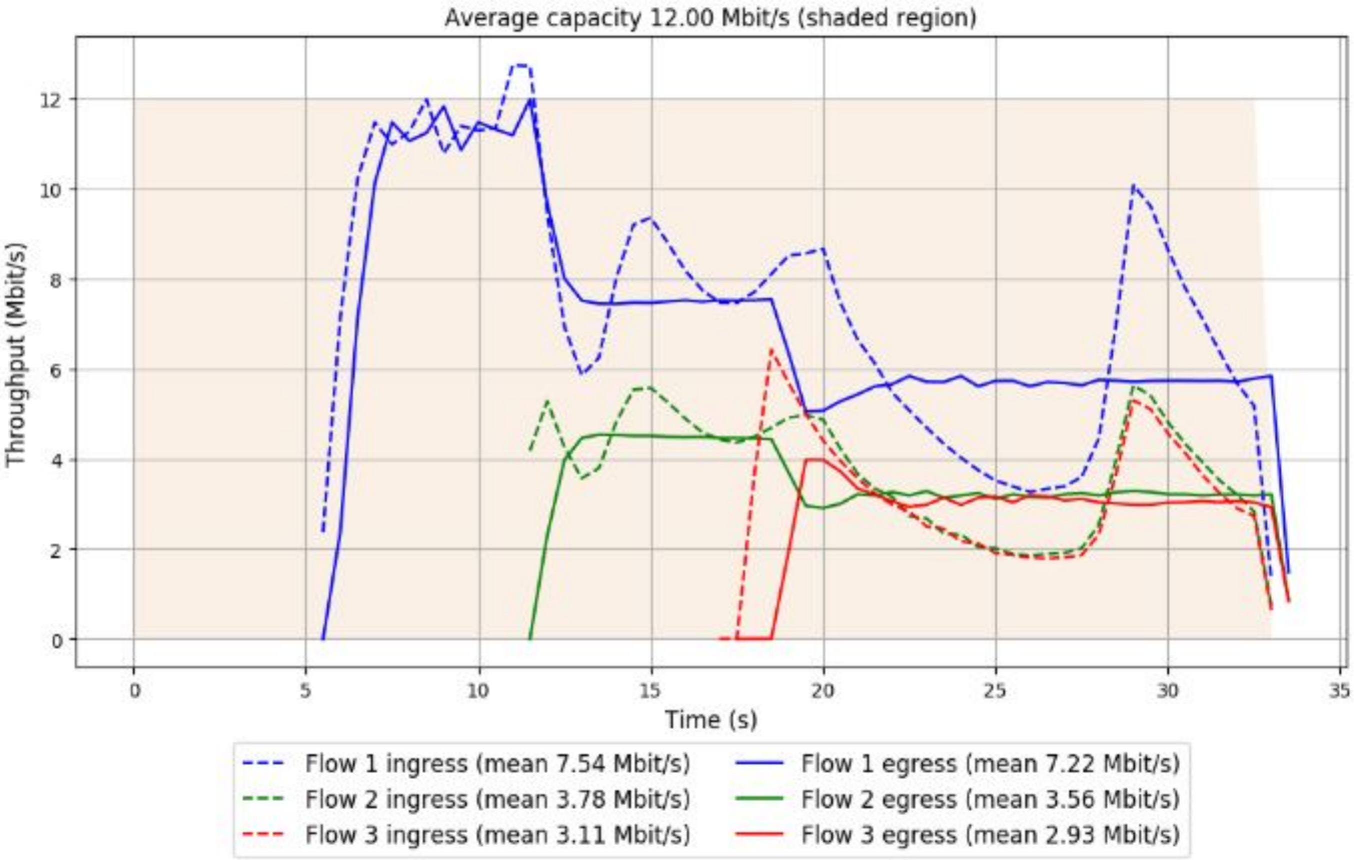


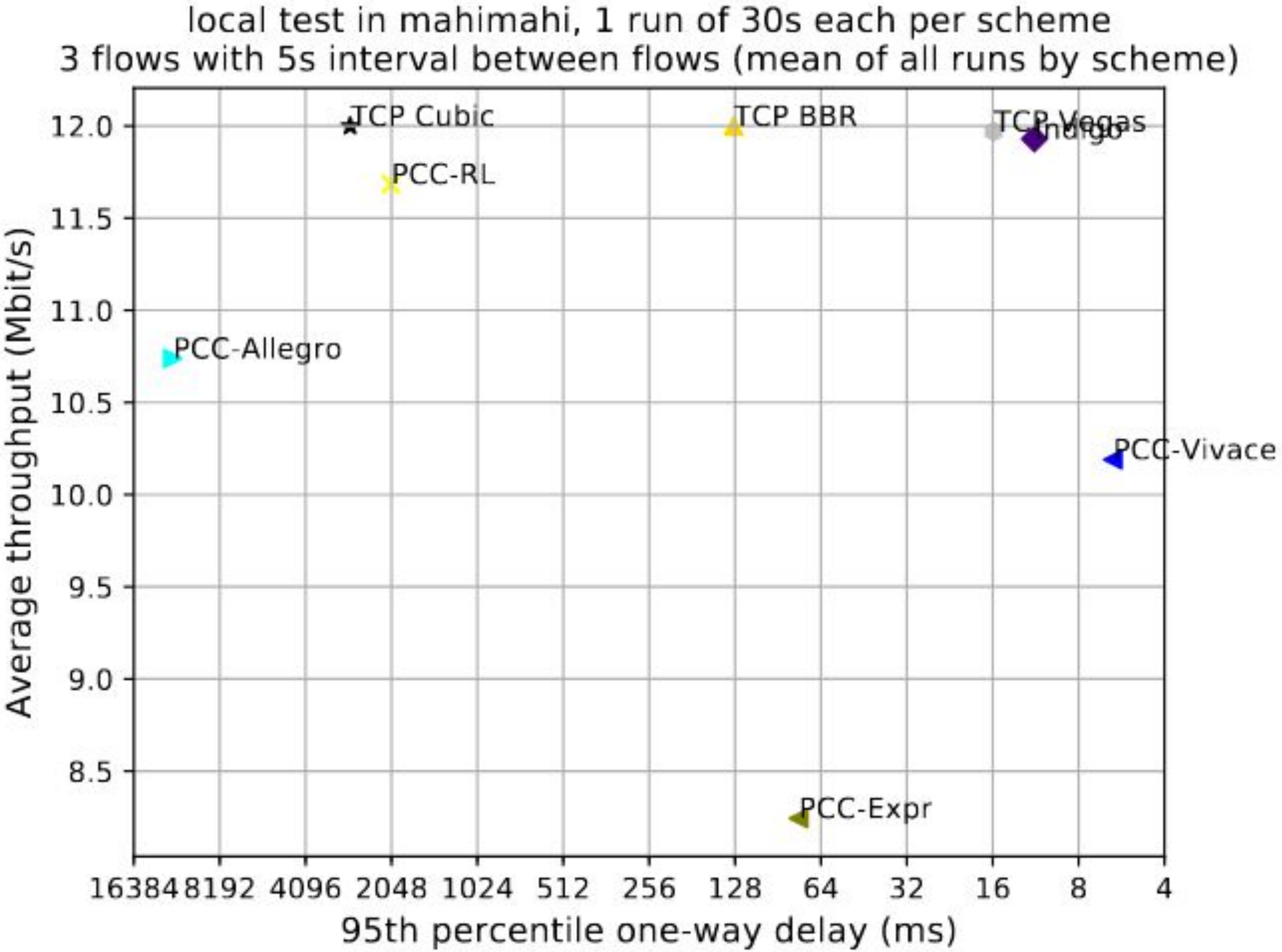


local test in mahimahi, 5 runs of 30s each per scheme
Average throughput (Mbit/s)
12
10
8
6
4
2
2048
1024
512
256
128
64
32
16
8
4
2
95th percentile one-way delay (ms)
Indigo
TCP Cubic
PCC-RL
PCC-Vivace
TCP Vegas
PCC-Allegro
TCP BBR
PCC-Expr

Test 3

Multiple flows

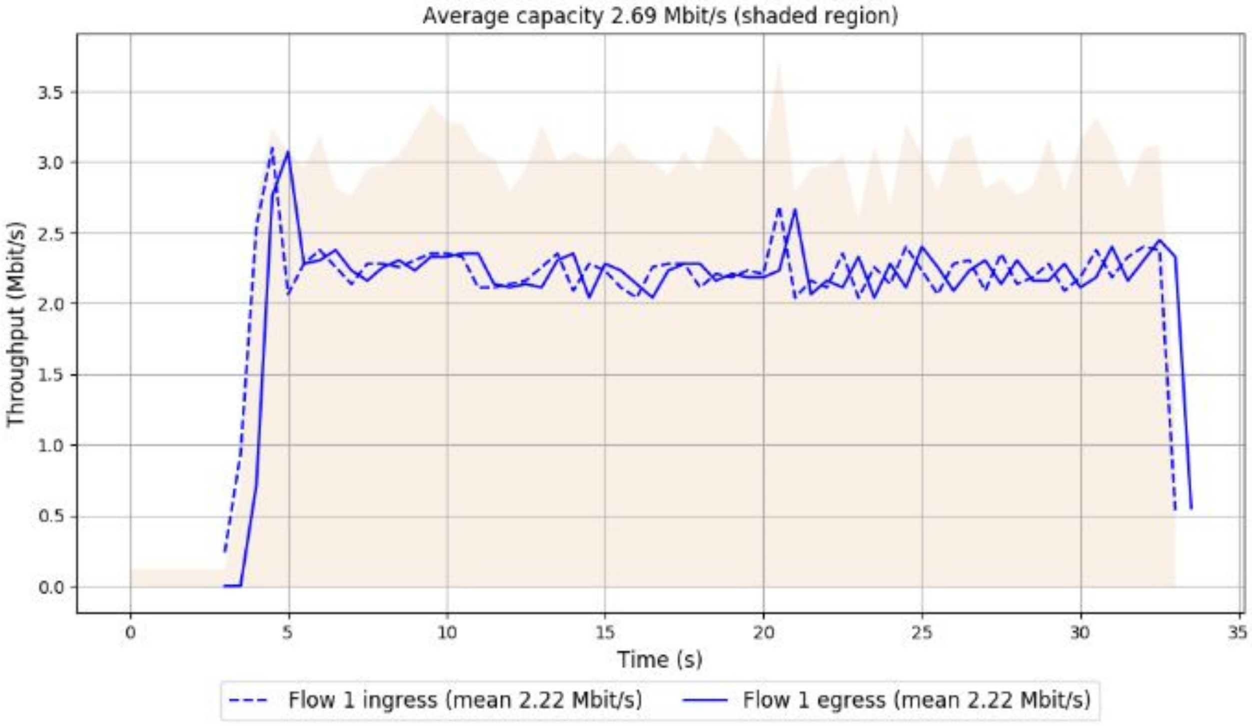


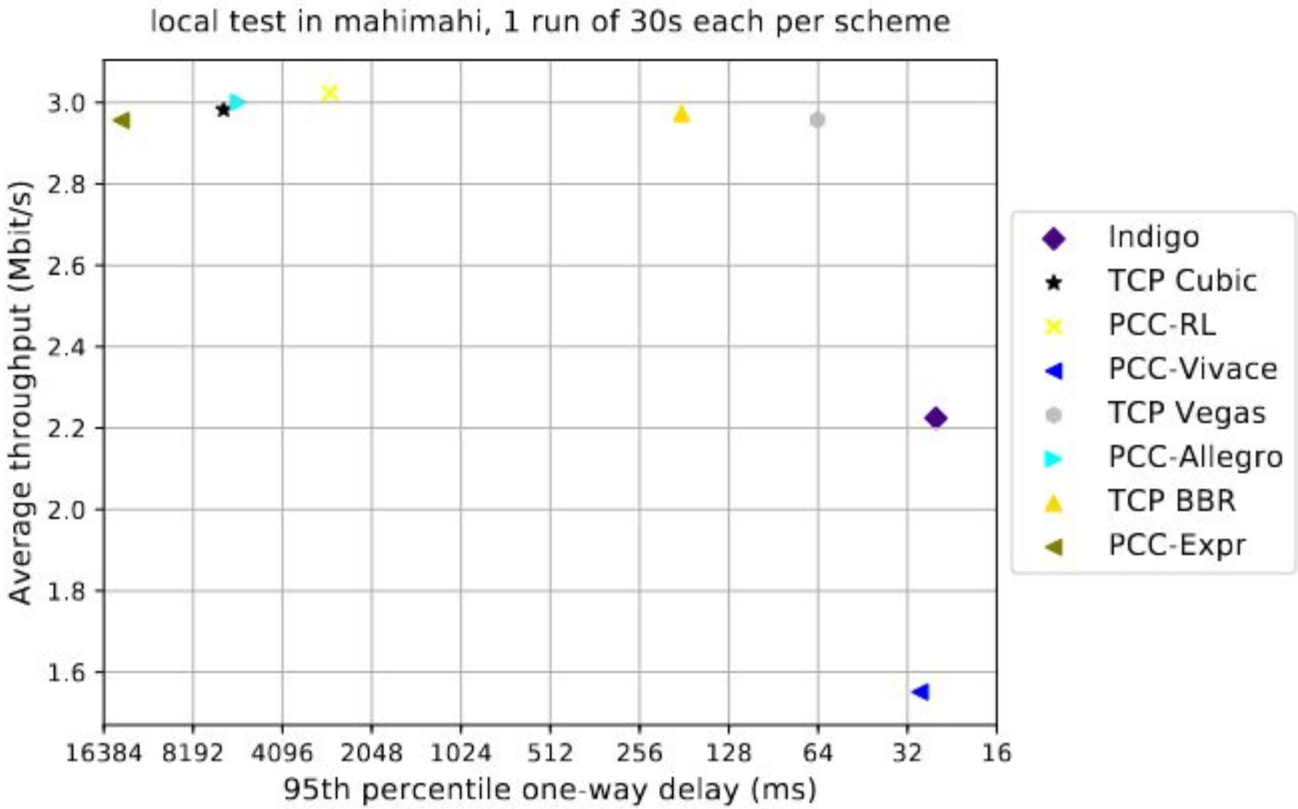


local test in mahimahi, 1 run of 30s each per scheme
3 flows with 5s interval between flows (mean of all runs by scheme)
Average throughput (Mbit/s)
95th percentile one-way delay (ms)
12.0
11.5
11.0
10.5
10.0
9.5
9.0
8.5
16384
8192
4096
2048
1024
512
256
128
64
32
16
8
4
TCP Cubic
TCP BBR
TCP Vegas
Indigo
PCC-RL
PCC-Allegro
PCC-Vivace
PCC-Expr

Test 4

Poisson distributed trace

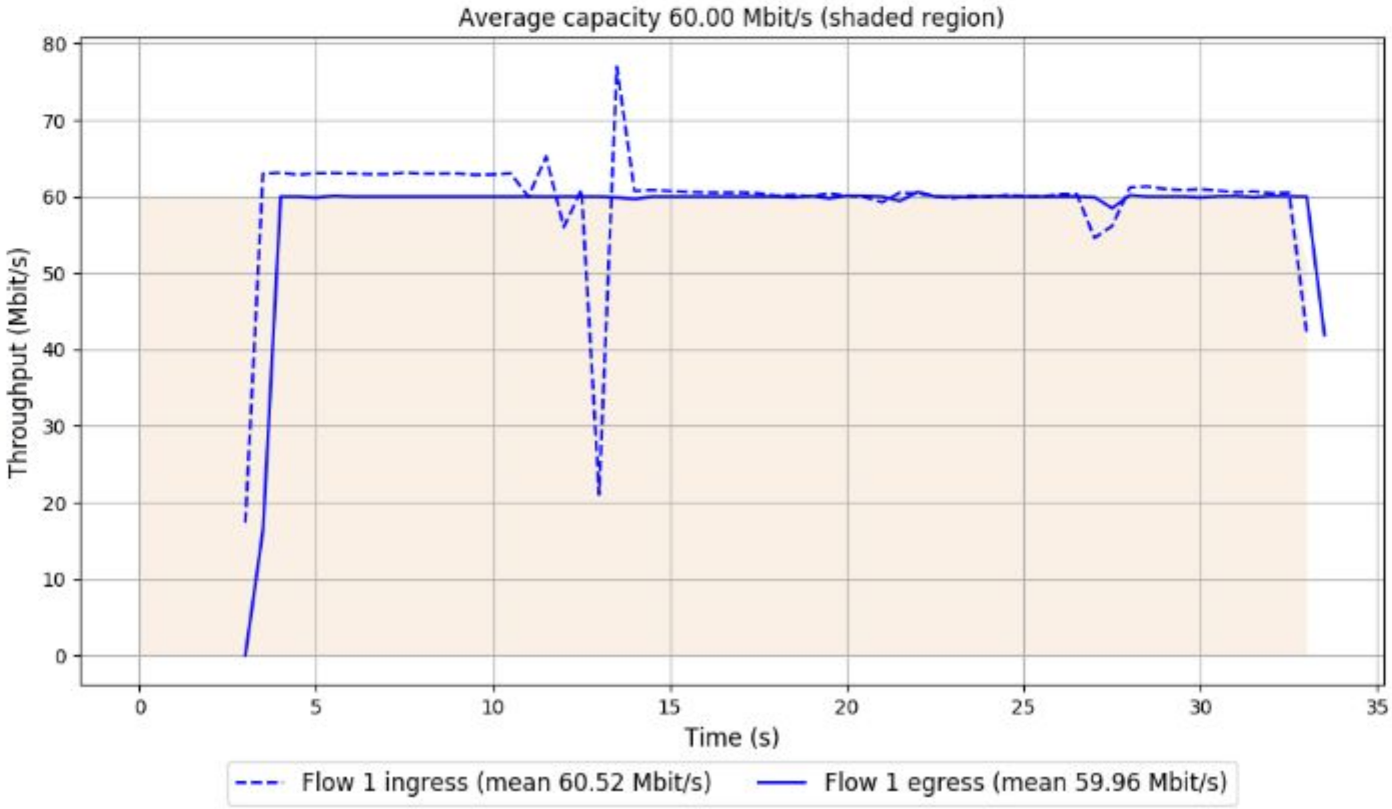


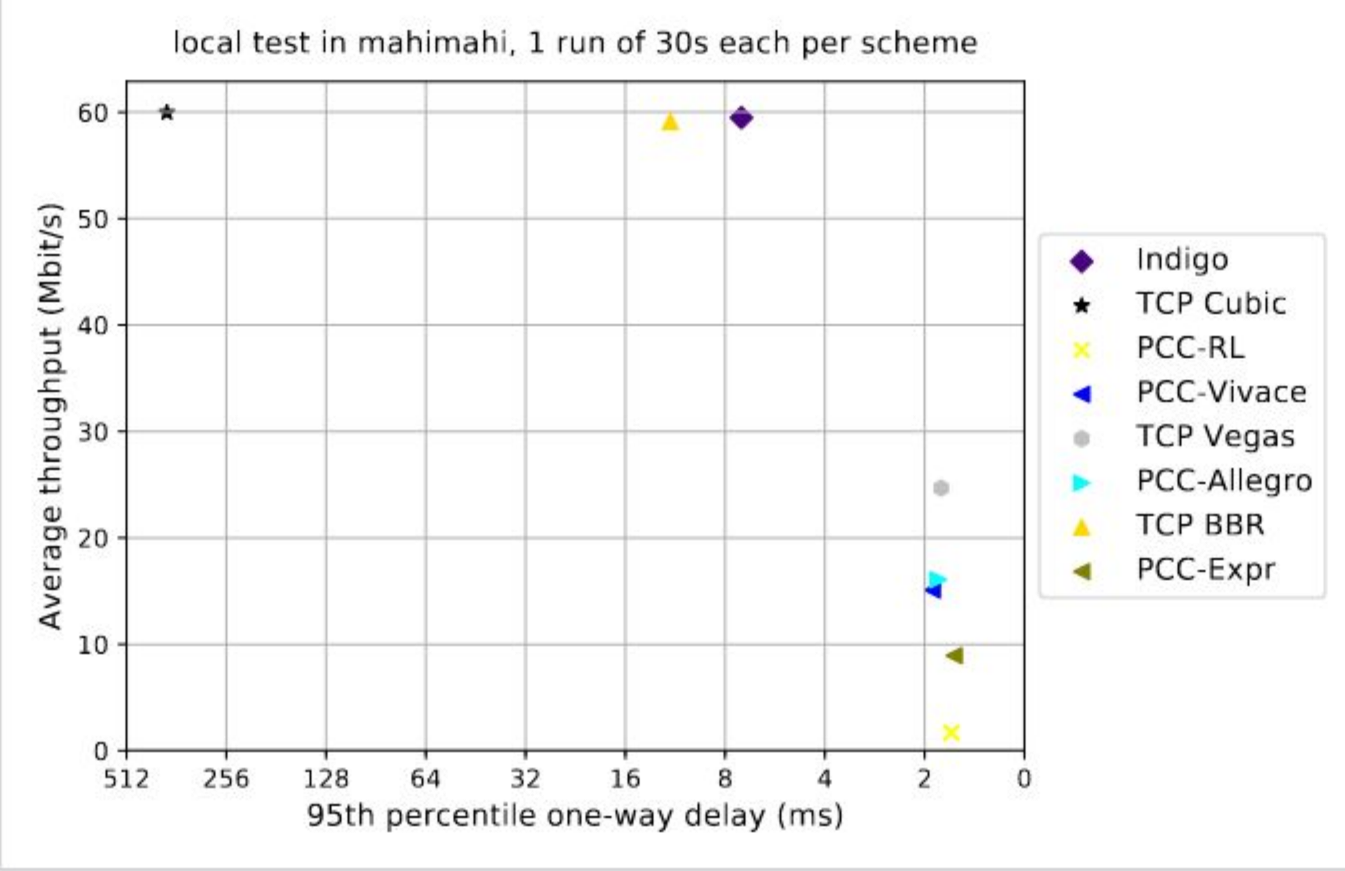


local test in mahimahi, 1 run of 30s each per scheme
Average throughput (Mbit/s)
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
16384
8192
4096
2048
1024
512
256
128
64
32
16
95th percentile one-way delay (ms)
Indigo
TCP Cubic
PCC-RL
PCC-Vivace
TCP Vegas
PCC-Allegro
TCP BBR
PCC-Expr

Test 5

60 Mbps trace

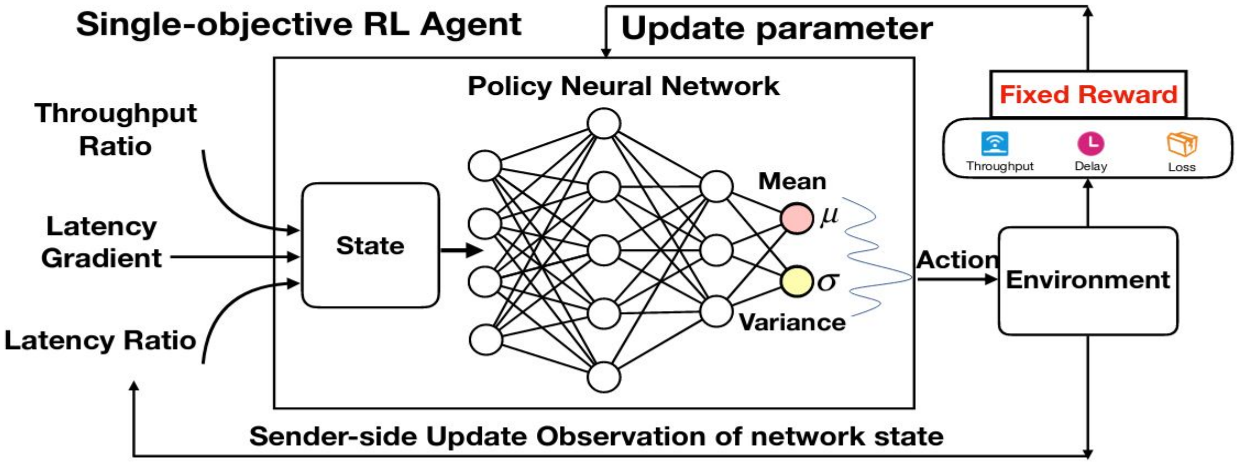




local test in mahimahi, 1 run of 30s each per scheme
Average throughput (Mbit/s)
0
10
20
30
40
50
60
512
256
128
64
32
16
8
4
2
0
95th percentile one-way delay (ms)
Indigo
TCP Cubic
PCC-RL
PCC-Vivace
TCP Vegas
PCC-Allegro
TCP BBR
PCC-Expr

Aurora

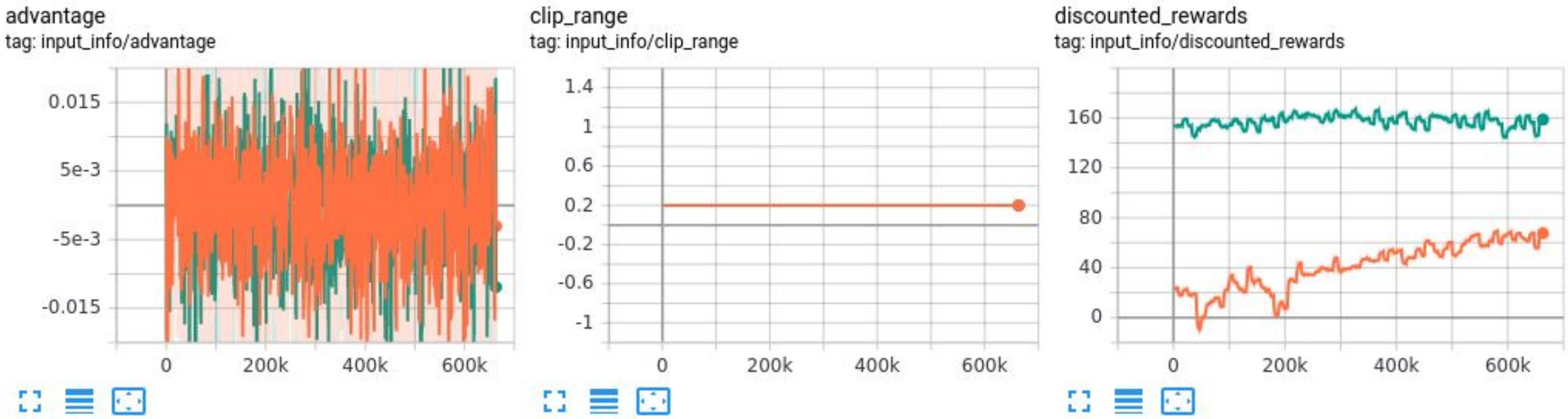
- RL Algorithm: Proximal Policy Optimization (PPO1)
- Architecture (of NN): [32, 16]
- History Length: 10
- Features: [‘sent latency inflation’, ‘latency ratio’, ‘send ratio’]
- Gamma: 0.99



Aurora: Implementation Details

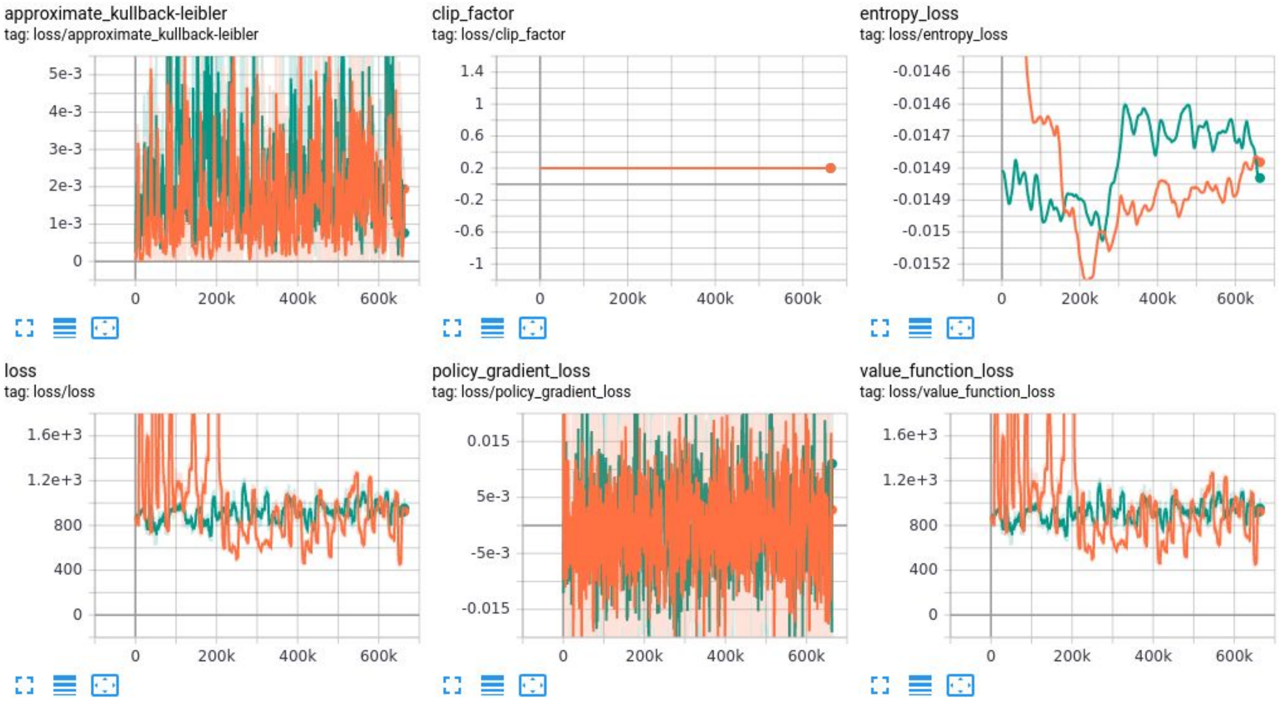
- States: Fixed-length history of statistics vector - latency gradient, latency ratio, sending ratio
- Actions: Adjust sending rate every monitoring interval
- Reward: 10**throughput* - 1000**latency* - 2000**loss*
- RL Algorithm: Proximal Policy Optimization. The PPO algorithm combines ideas from A2C (having multiple workers) and TRPO (it uses a trust region to improve the actor). The main idea is that after an update, the main policy should not be too far from the old policy. For that, PPO uses clipping to avoid too large updates. It empirically performs at least as close to TRPO.

PPO [64, 32]: Training Statistics

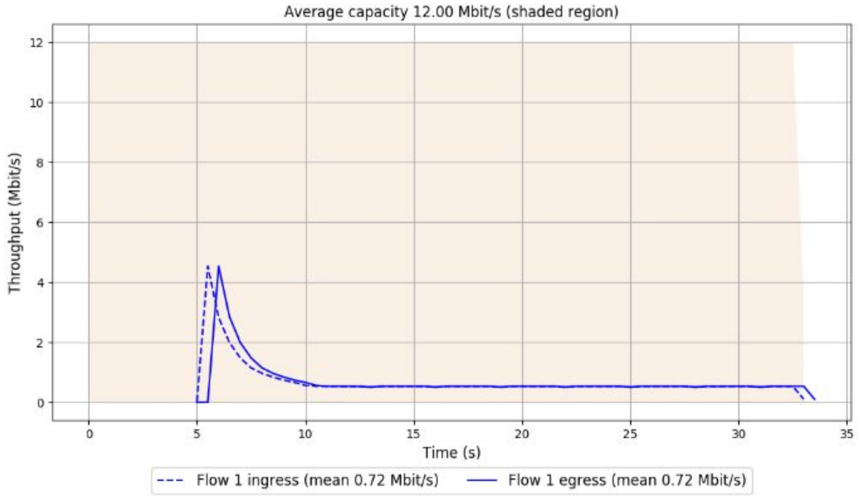


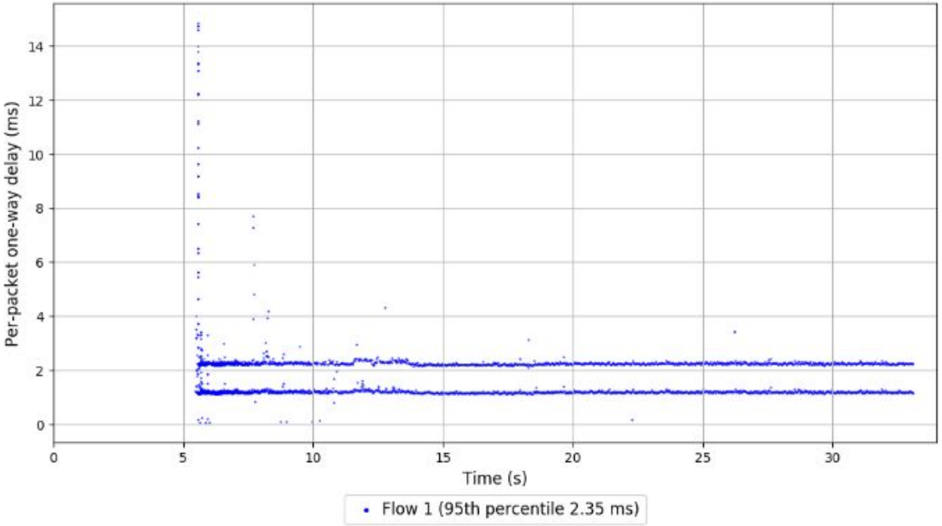
Orange graph - First checkpoint, Green graph - Last checkpoint

PPO [64, 32]: Training Statistics



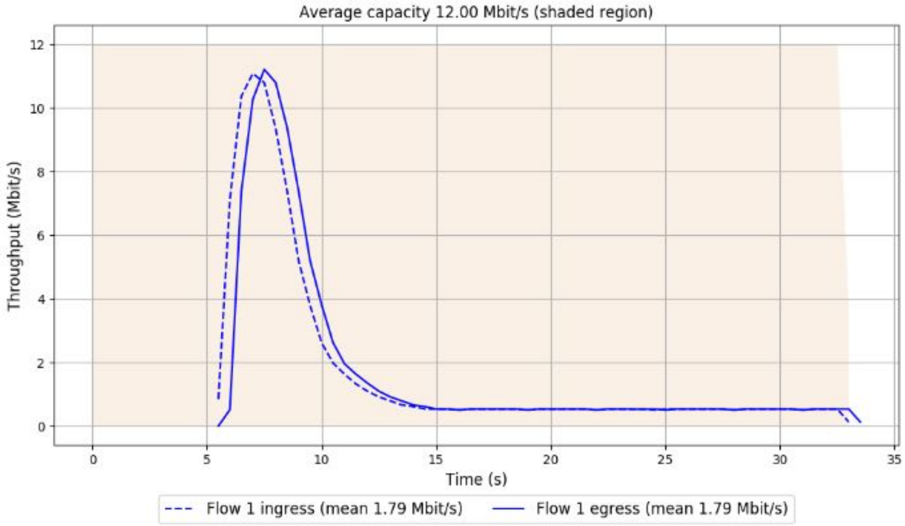
Test 1: PPO [64, 32]

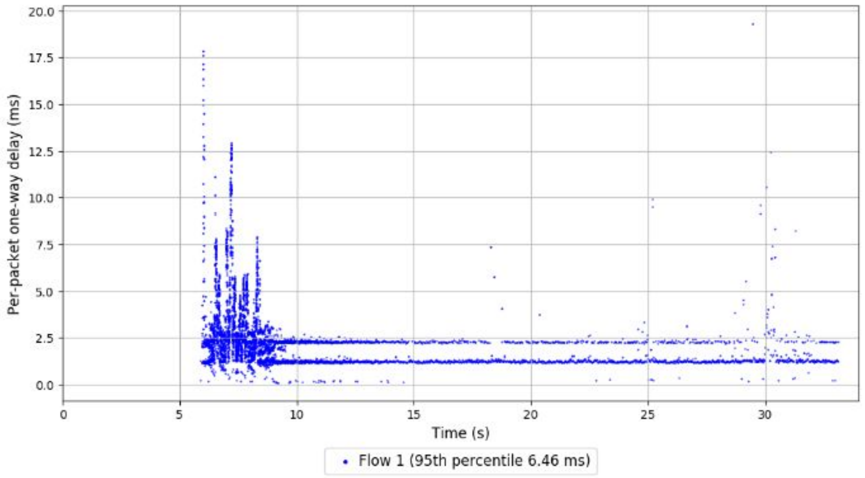




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	0.72	2.35	0.00

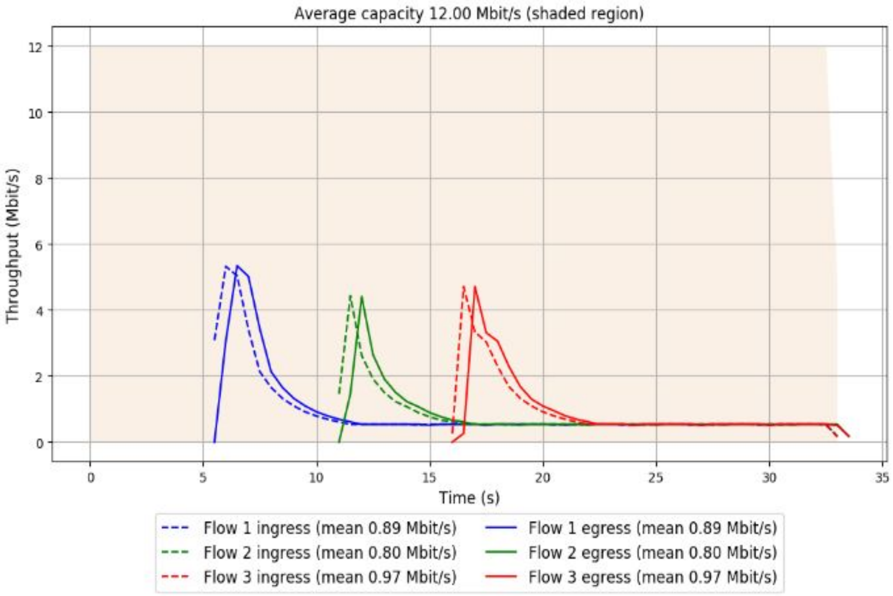
Test 2: PPO [64, 32]

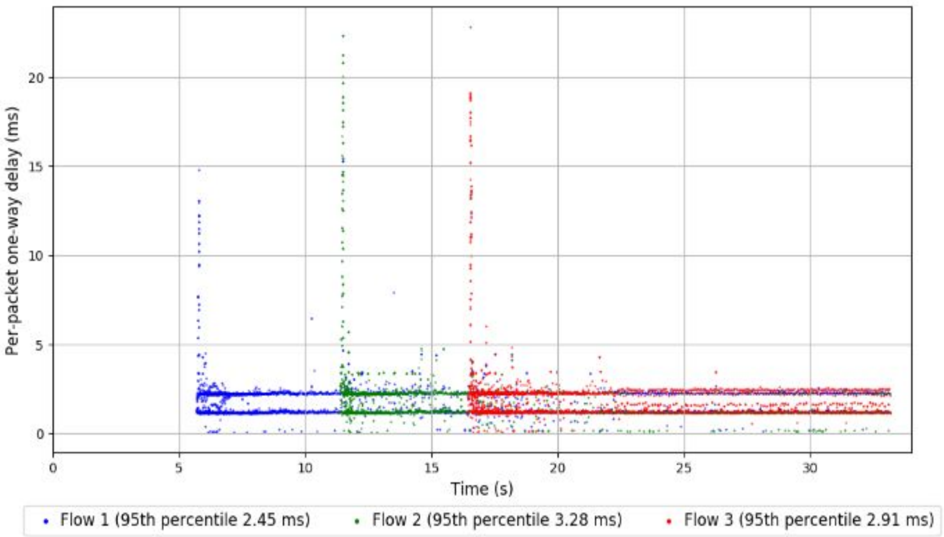




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	5	0.98	3.25	0.00

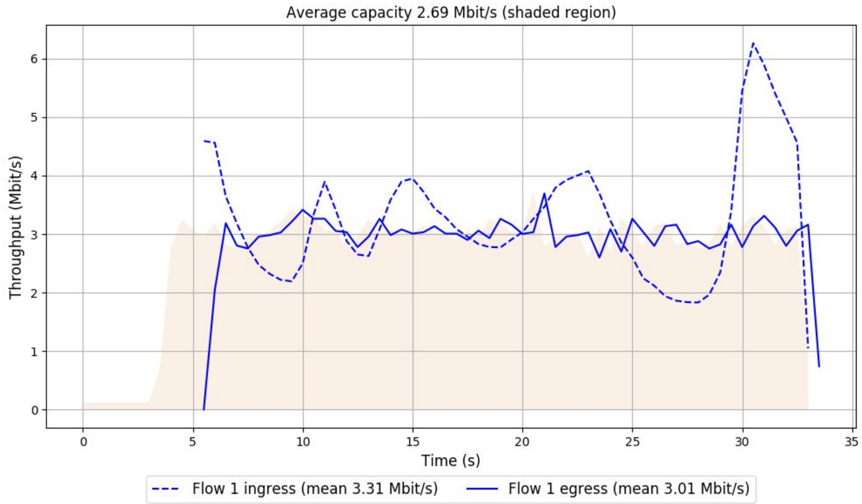
Test 3: PPO [64, 32]

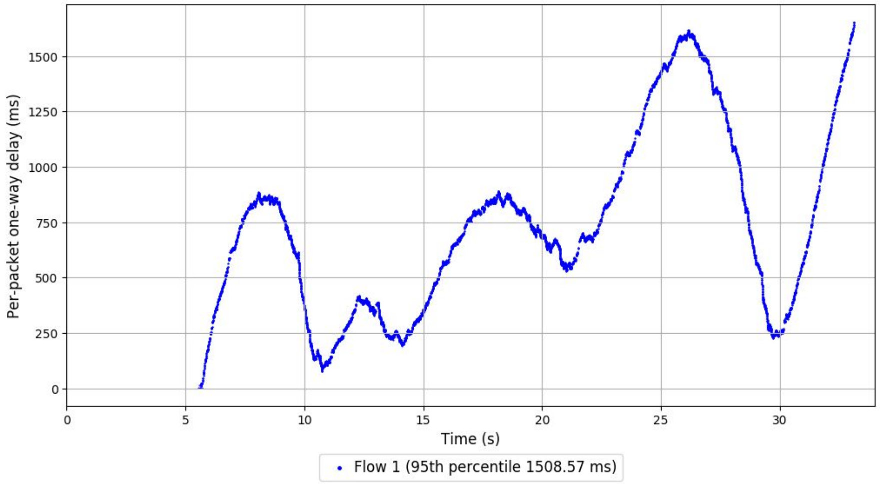




scheme	# runs	mean avg tput (Mbit/s) flow 1	flow 2	flow 3	mean 95th-%ile delay (ms) flow 1	flow 2	flow 3	mean loss rate (%) flow 1	flow 2	flow 3
PCC-RL	1	0.89	0.80	0.97	2.45	3.28	2.91	0.00	0.00	0.00

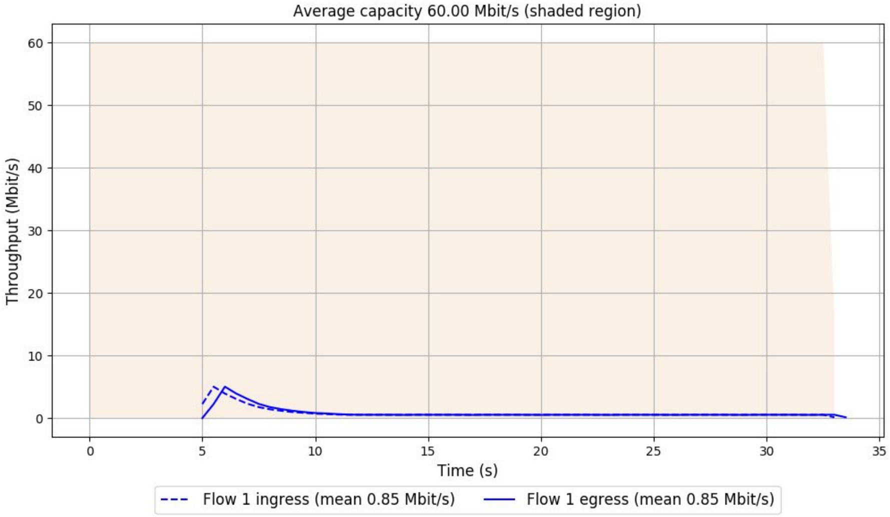
Test 4: PPO [64, 32]

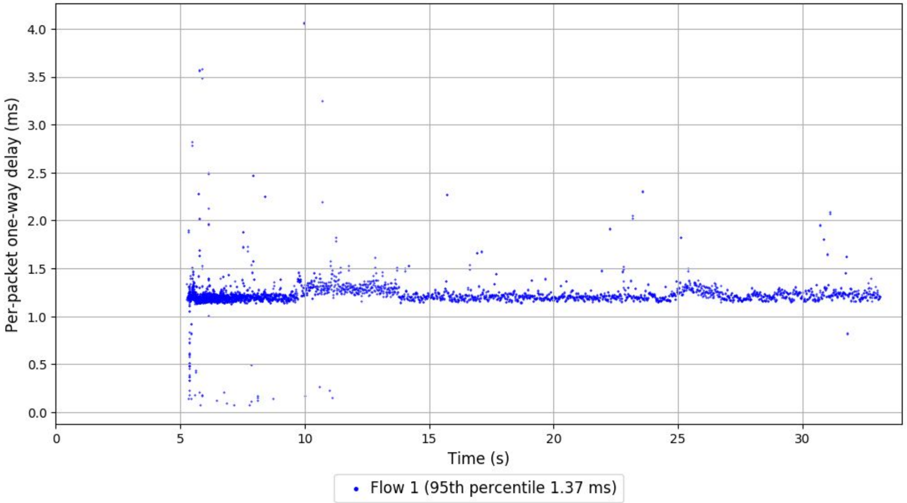




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	3.01	1508.57	8.92

Test 5: PPO [64,32]





scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	0.85	1.37	0.00

Aurora: Network Architecture Tests

RL Algorithm: Proximal Policy Optimization (PPO1)

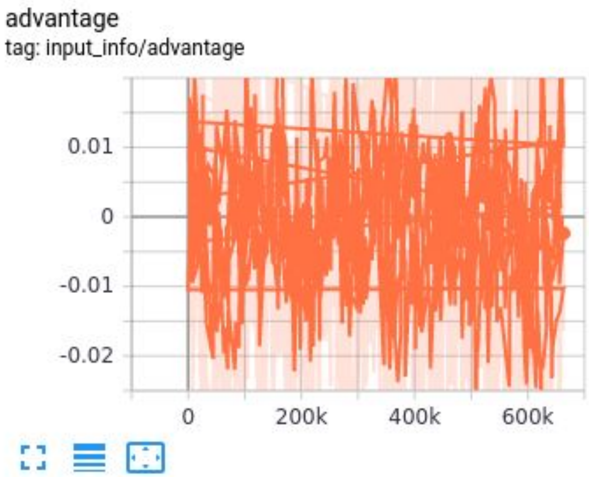
Architecture (of NN): [32, 16] ➜ **[128, 8]**

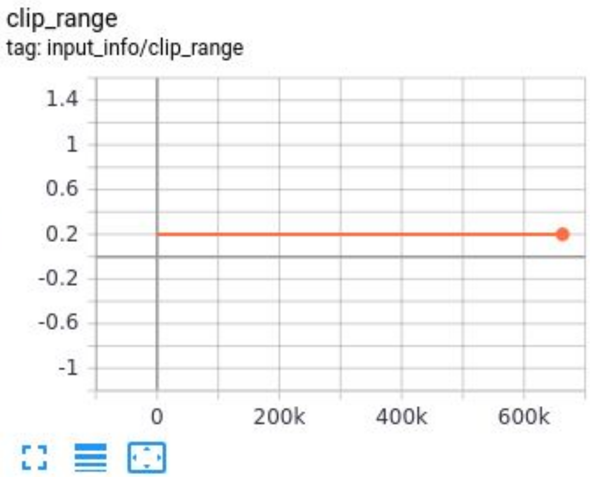
History Length: 10

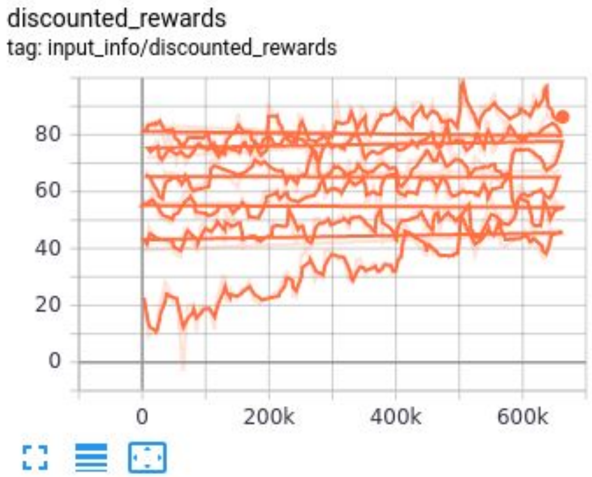
Features: [‘sent latency inflation’, ‘latency ratio’, ‘send ratio’]

Gamma: 0.99

PPO [128, 8]: Training Statistics

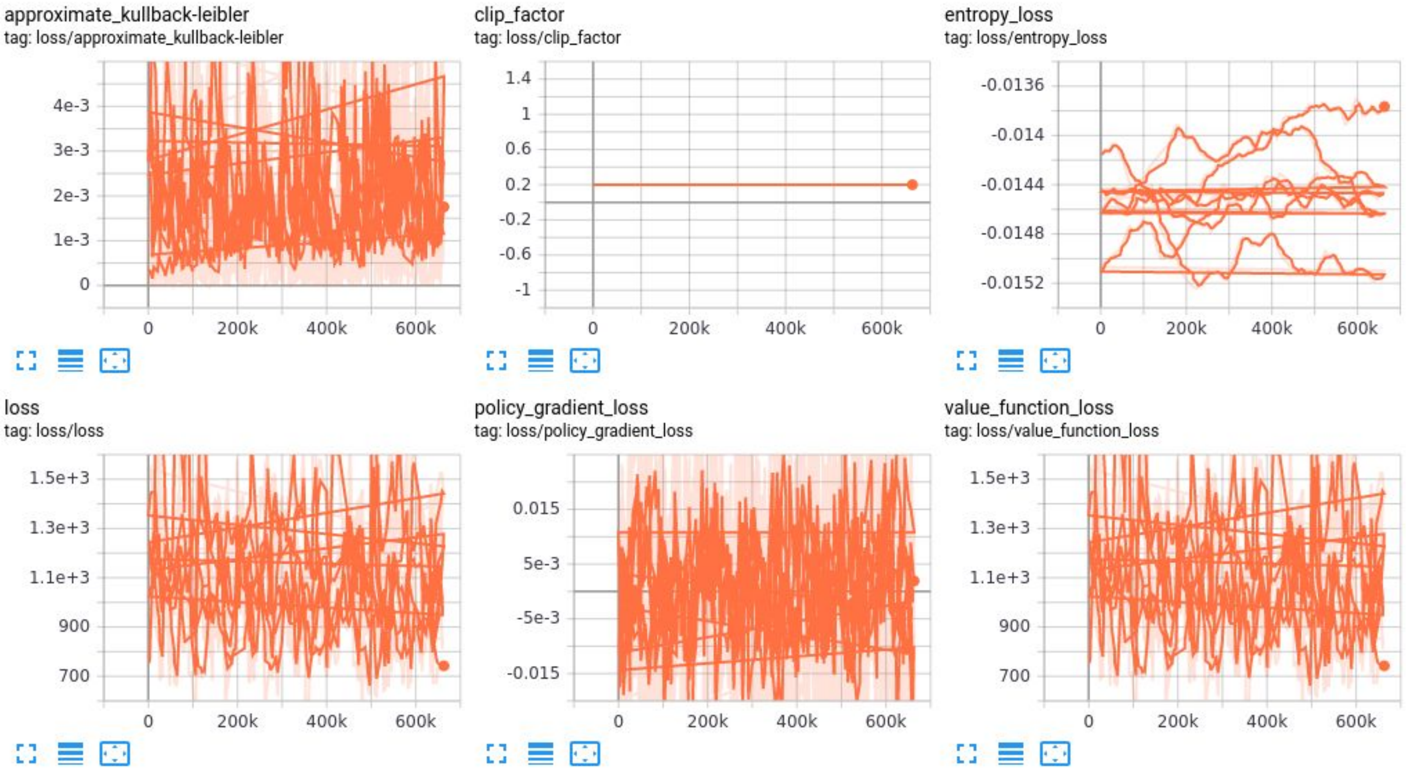




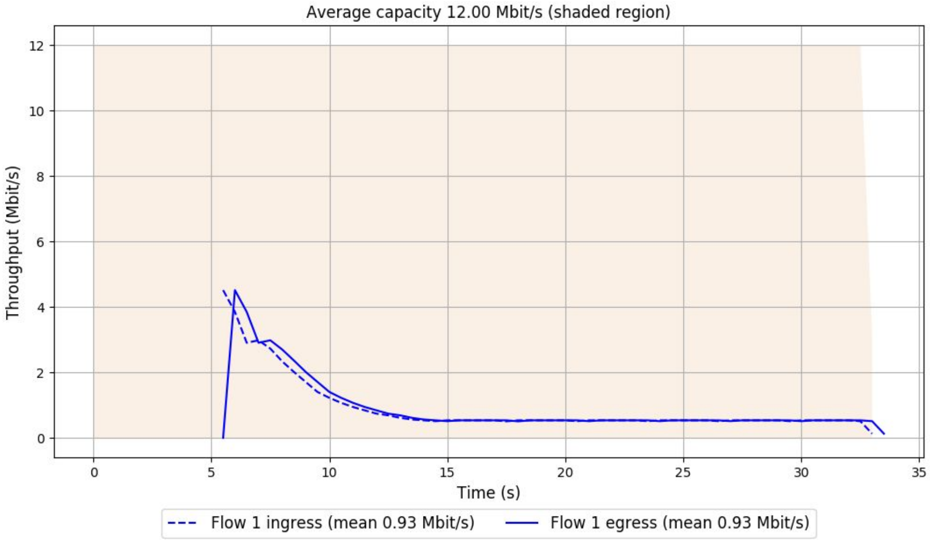


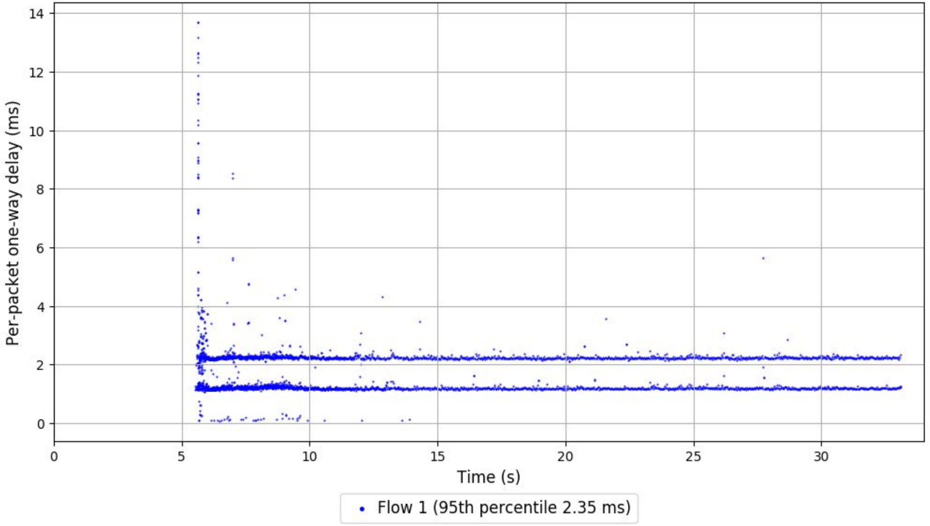
All six training checkpoints - discounted rewards increase with each checkpoint

PPO [128, 8]: Training Statistics



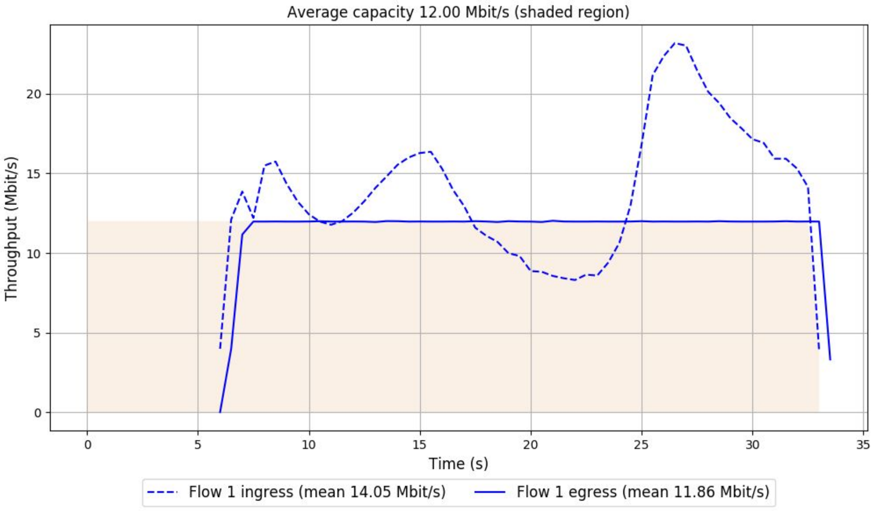
Test 1: PPO [128, 8]

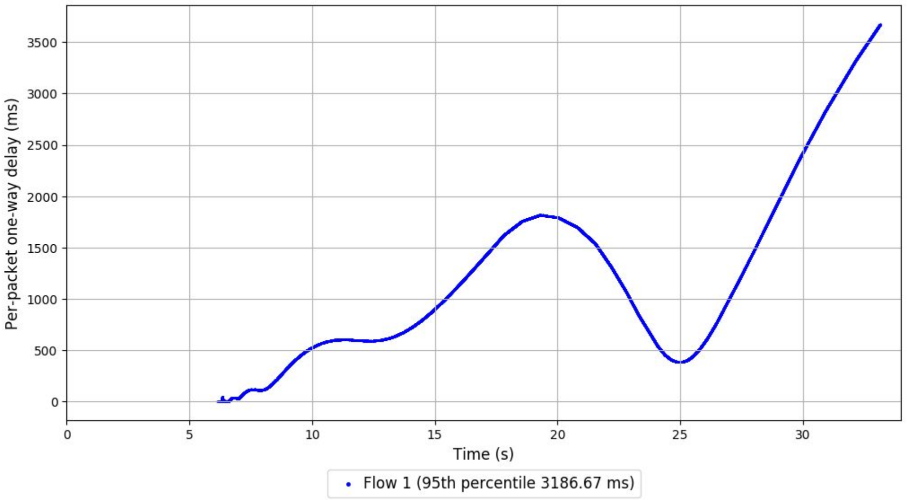




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	0.93	2.35	0.00

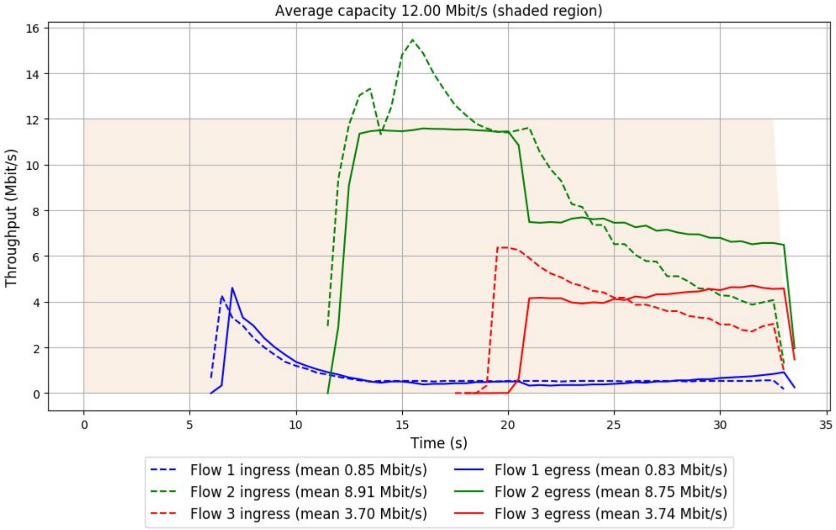
Test 2: PPO [128, 8]

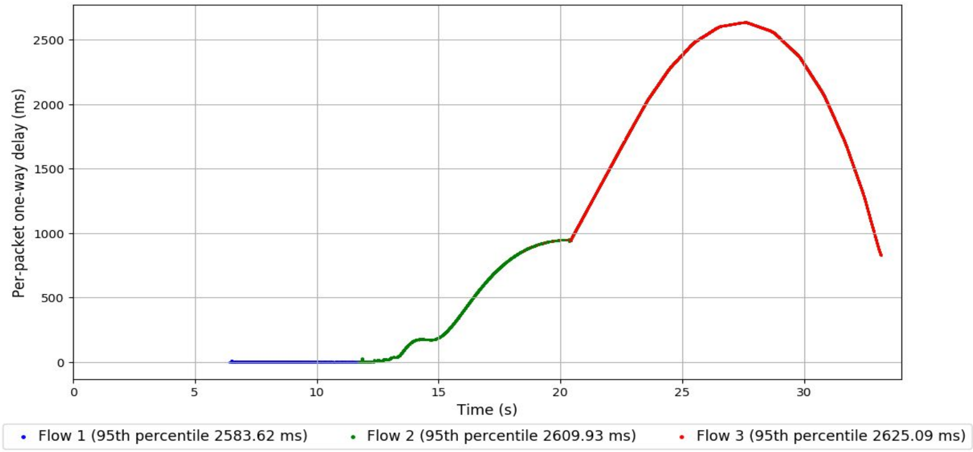




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	5	9.63	2880.08	11.33

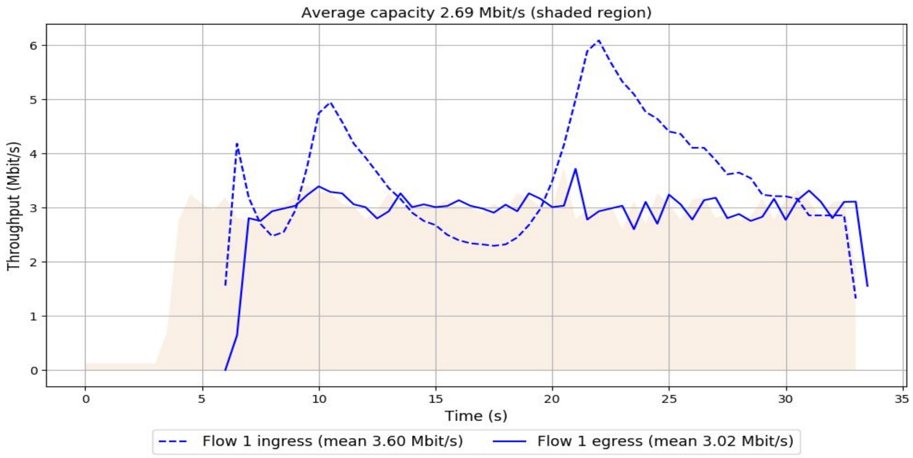
Test 3: PPO [128, 8]

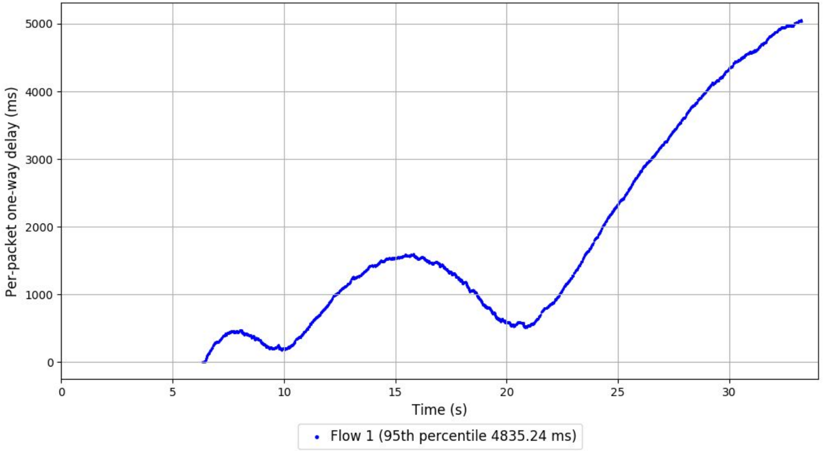




scheme	# runs	mean avg tput (Mbit/s)			mean 95th-%ile delay (ms)			mean loss rate (%)		
		flow 1	flow 2	flow 3	flow 1	flow 2	flow 3	flow 1	flow 2	flow 3
PCC-RL	1	0.83	8.75	3.74	2583.62	2609.93	2625.09	2.07	1.77	4.35

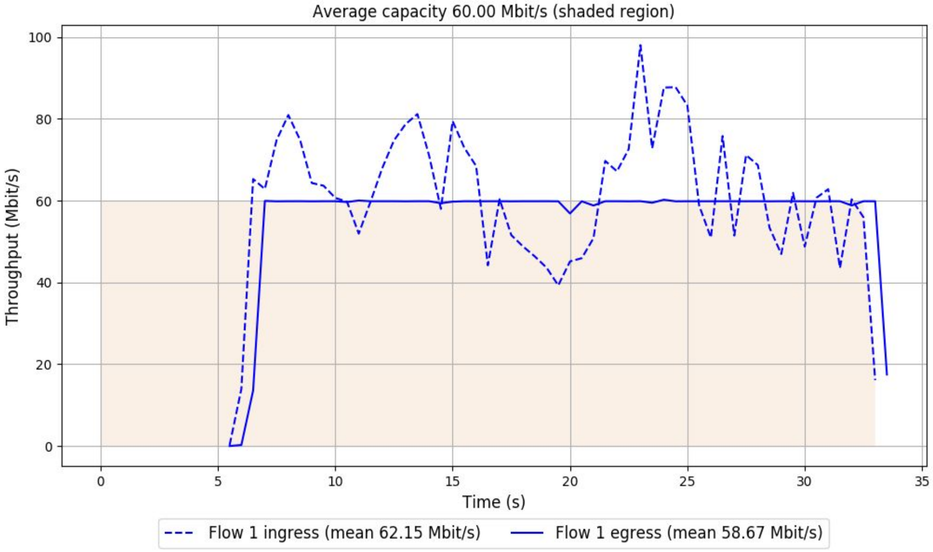
Test 4: PPO [128, 8]

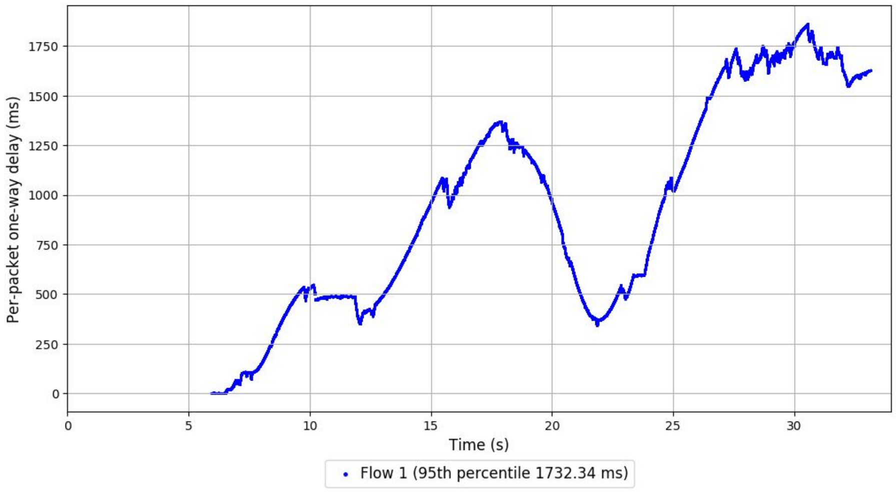




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	3.02	4835.24	16.13

Test 5: PPO [128,8]





scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	58.67	1732.34	5.60

Aurora: Network Architecture Tests

RL Algorithm: Proximal Policy Optimization (PPO1)

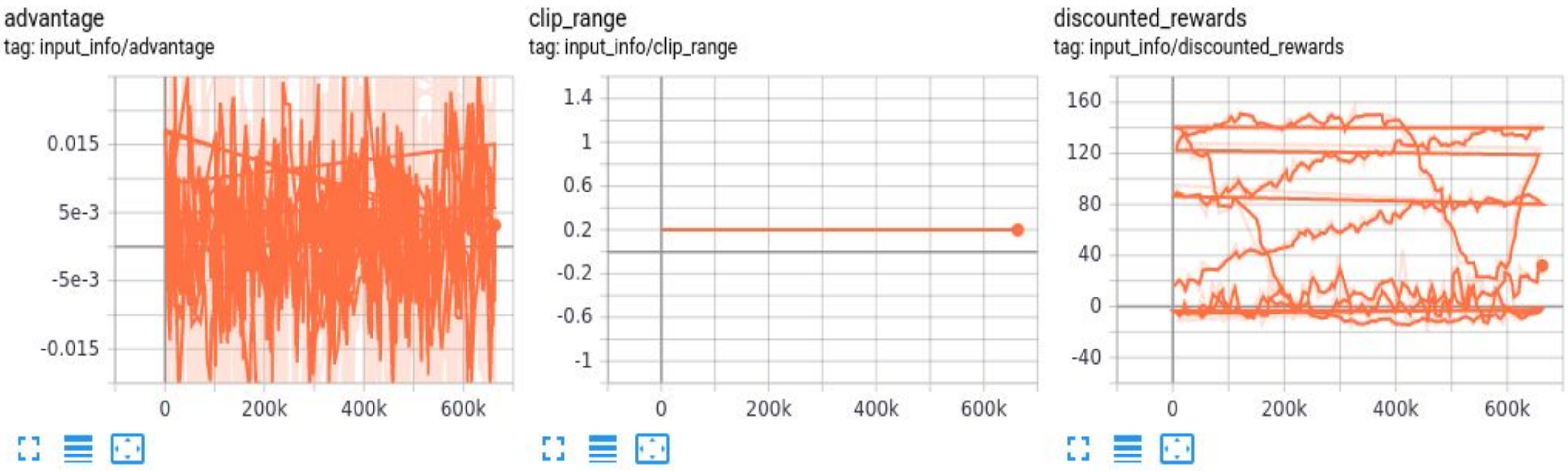
Architecture (of NN): [32, 16] ➜ **[64, 32, 64]**

History Length: 10

Features: [‘sent latency inflation’, ‘latency ratio’, ‘send ratio’]

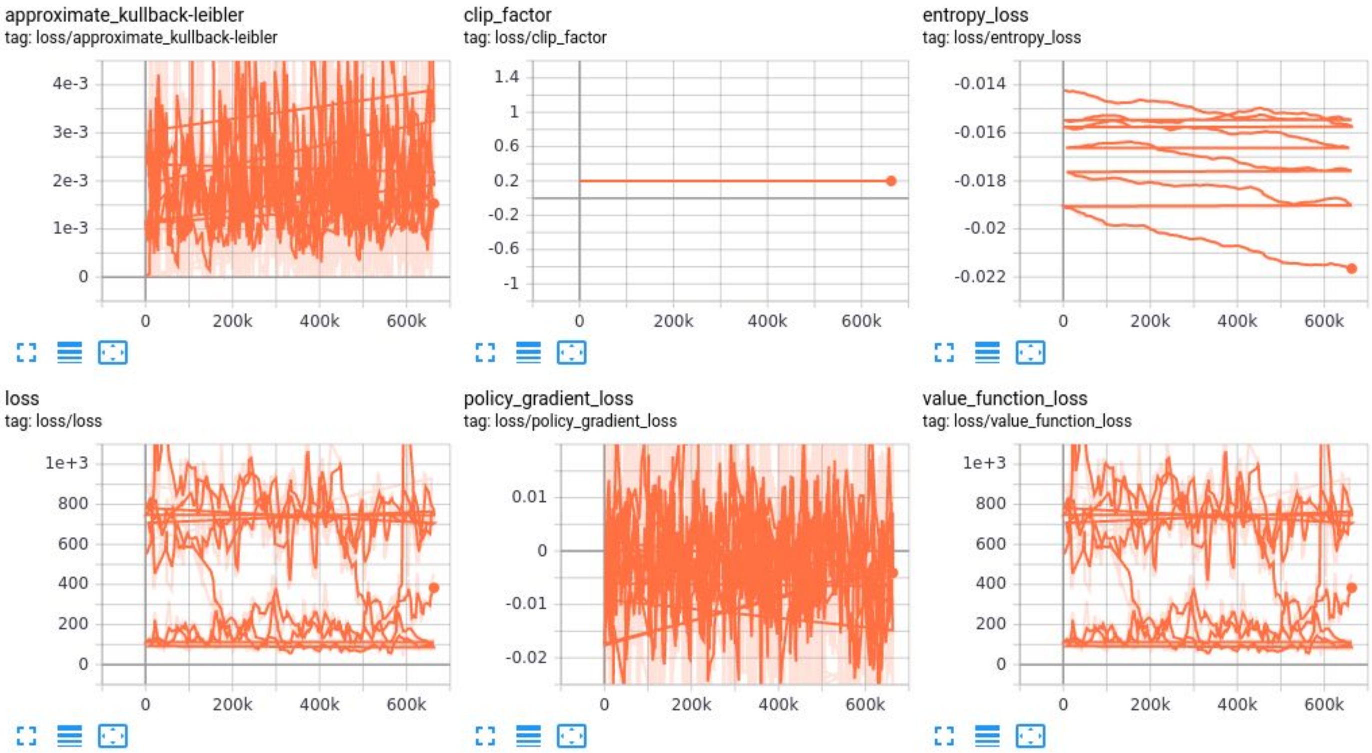
Gamma: 0.99

PPO [64, 32, 64]: Training Statistics

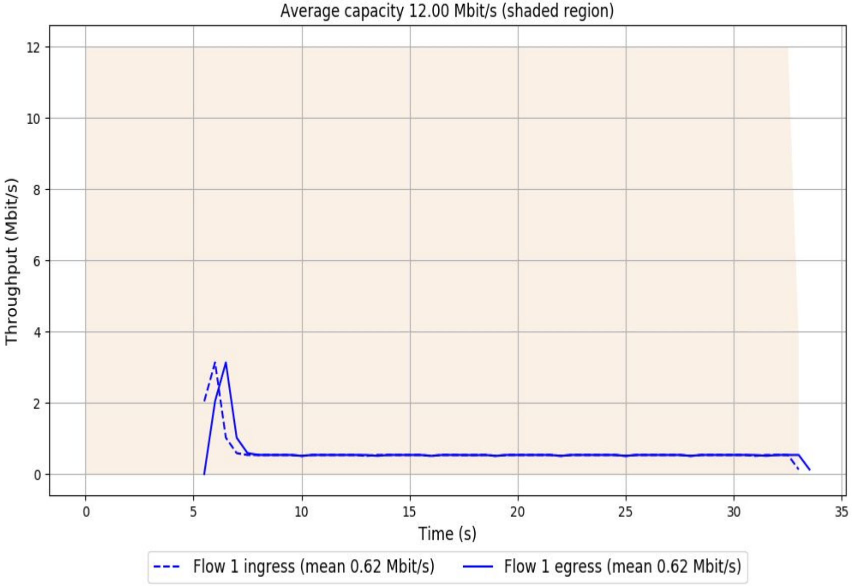


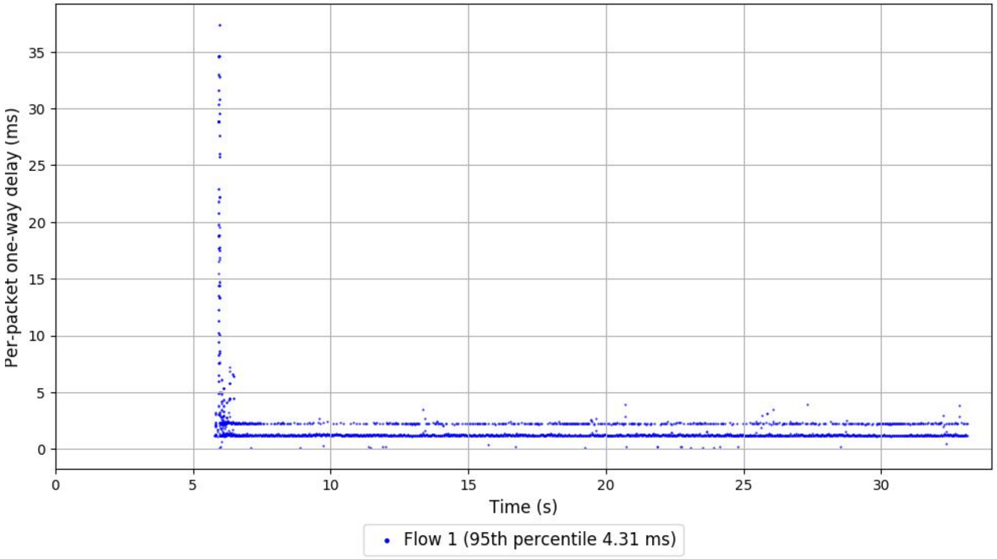
All six training checkpoints - discounted rewards increase with each checkpoint

PPO [64, 32, 64]: Training Statistics



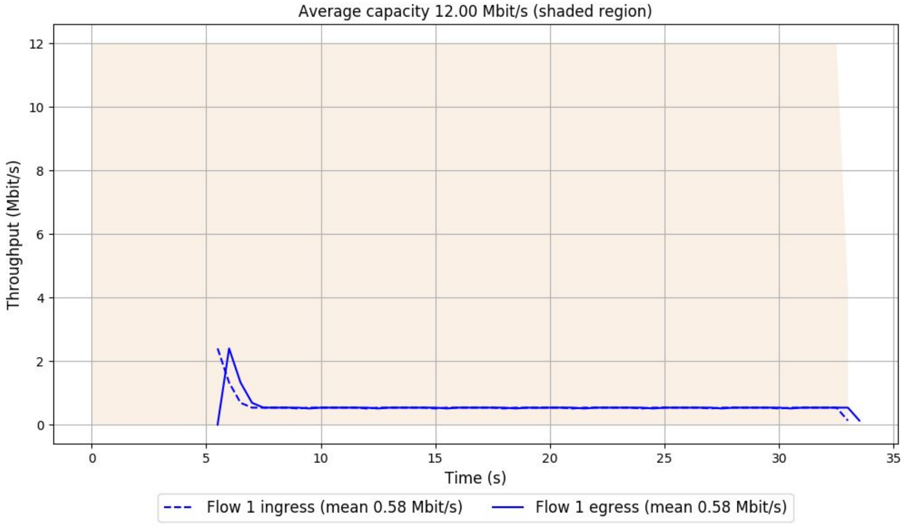
Test 1: PPO [64, 32, 64]

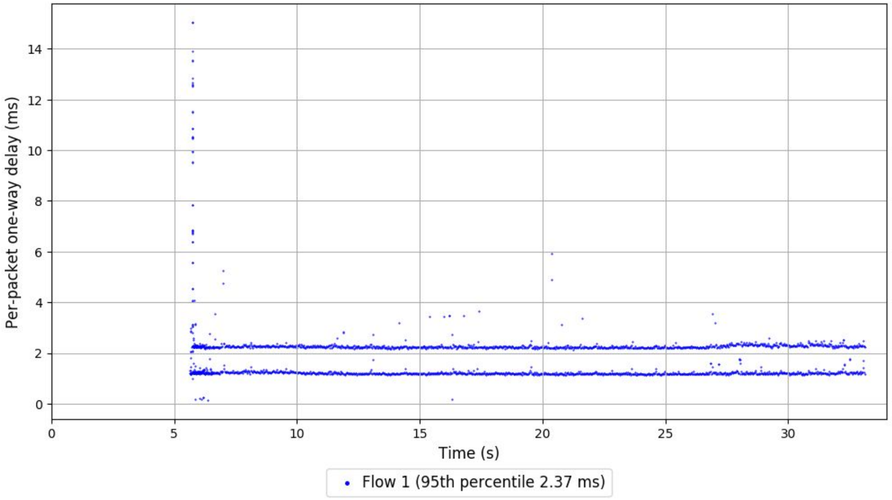




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	0.62	4.31	0.00

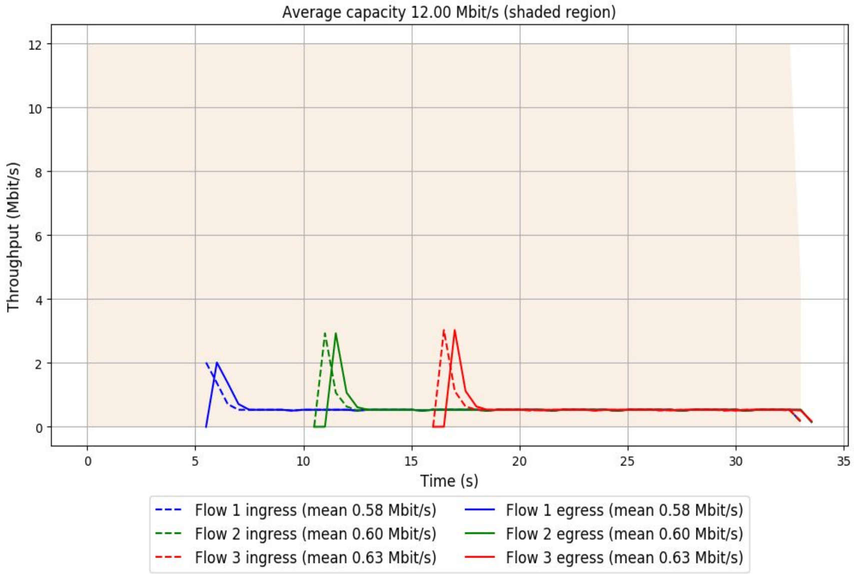
Test 2: PPO [64, 32, 64]

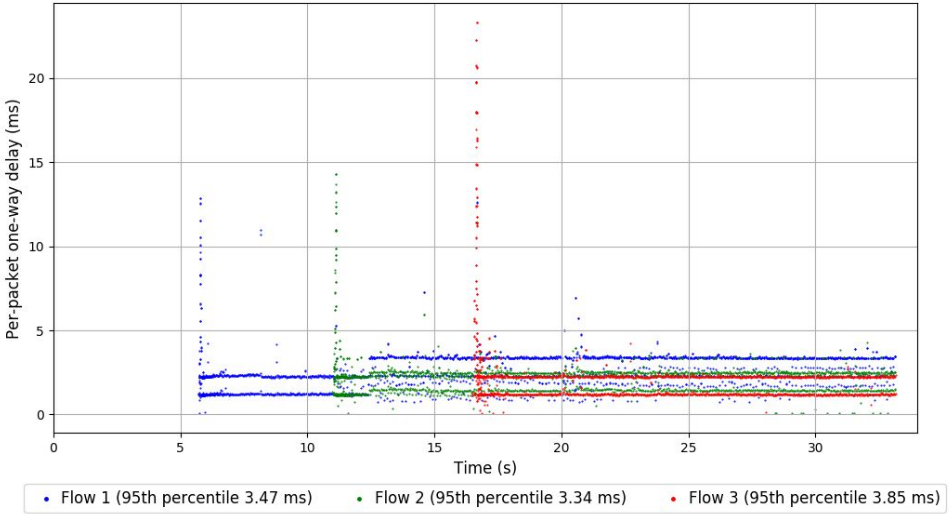




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	5	0.59	2.46	0.00

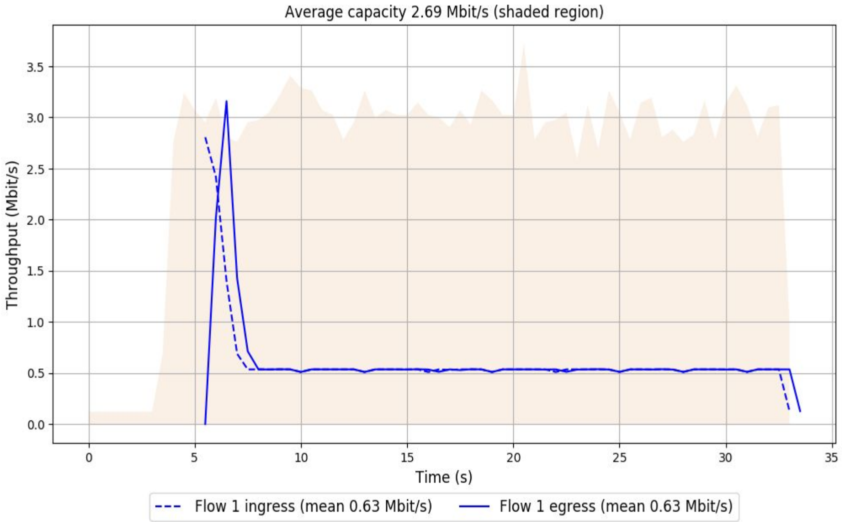
Test 3: PPO [64, 32, 64]

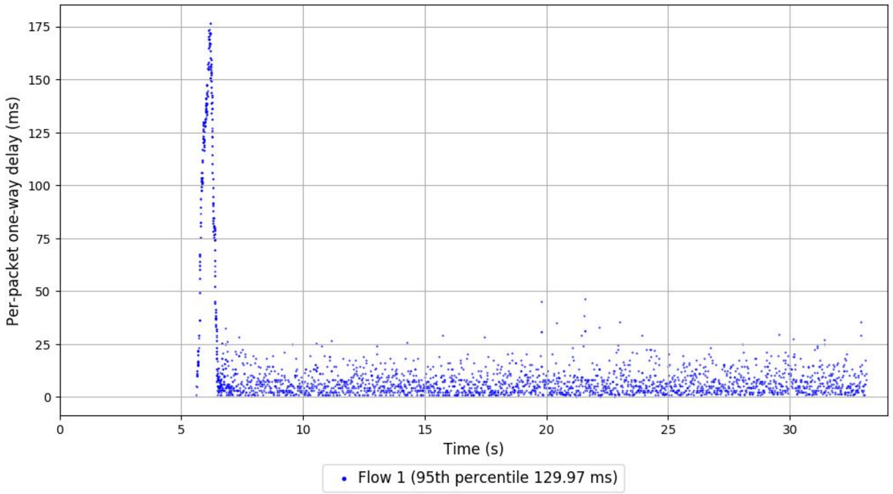




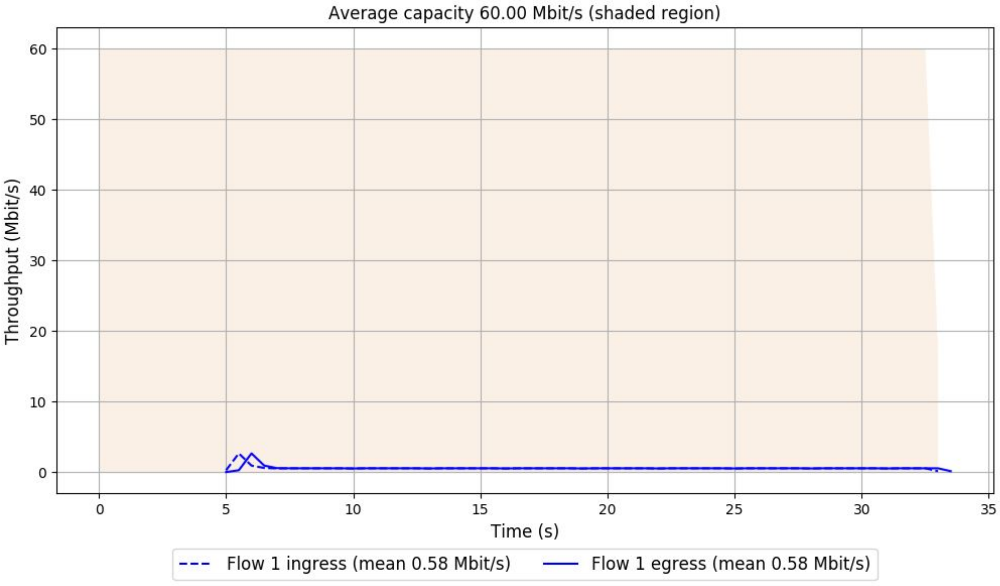
scheme	# runs	mean avg tput (Mbit/s) flow 1	flow 2	flow 3	mean 95th-%ile delay (ms) flow 1	flow 2	flow 3	mean loss rate (%) flow 1	flow 2	flow 3
PCC-RL	1	0.58	0.60	0.63	3.47	3.34	3.85	0.00	0.00	0.00

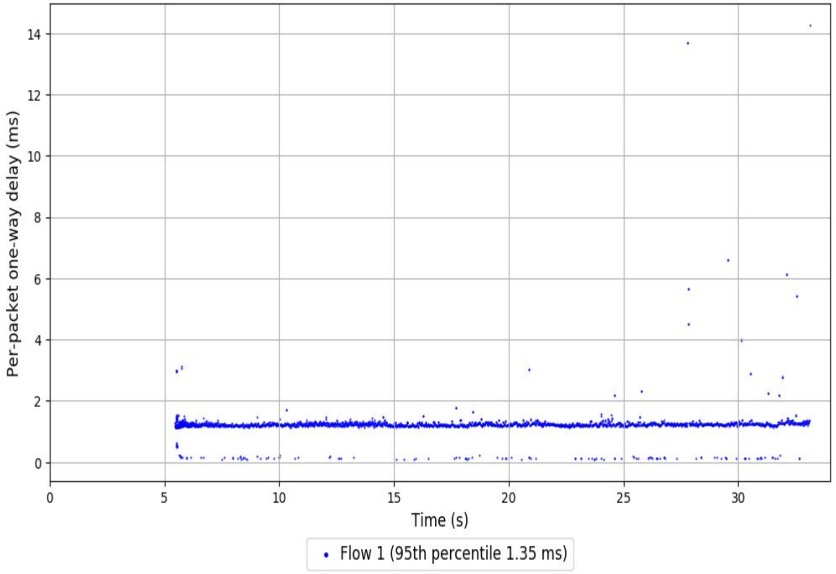
Test 4: PPO [64, 32, 64]





scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	0.63	129.97	0.00





scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	0.58	1.35	0.00

Aurora: TD3

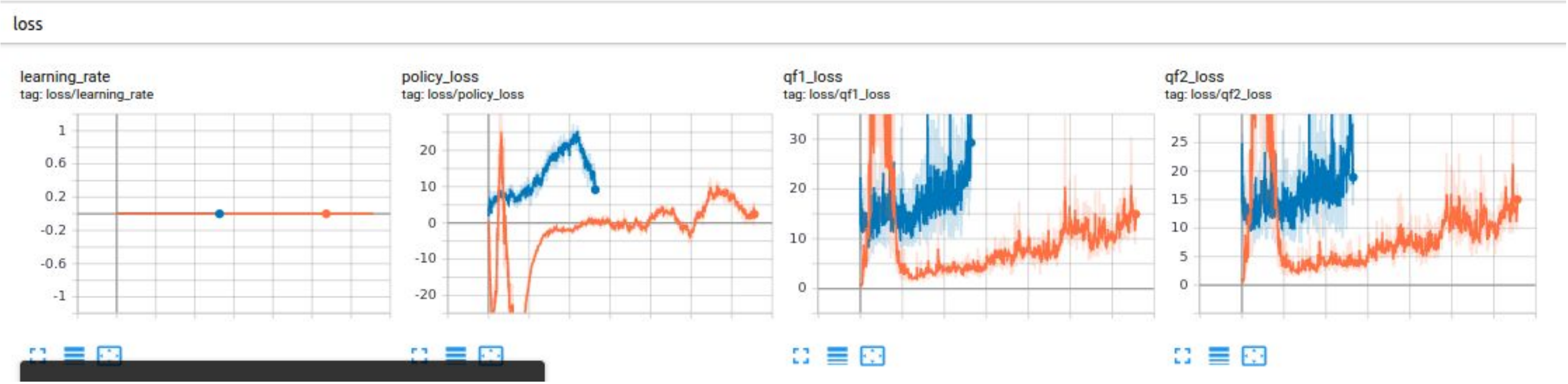
- RL Algorithm: Twin Delayed Deep Deterministic Policy Gradient
- Architecture (of NN): [64, 32]
- History Length: 10
- Features:
- Gamma: 0.99
- Twin Delayed Deep Deterministic Policy Gradient: TD3 addresses function approximation error in Actor-Critic methods. TD3 is a direct successor of DDPG and improves it using three major tricks: clipped Q-Learning, delayed policy update, and target policy smoothing. These tricks result in substantially improved performance over baseline DDPG.

TD3 [64, 32]: Training Statistics

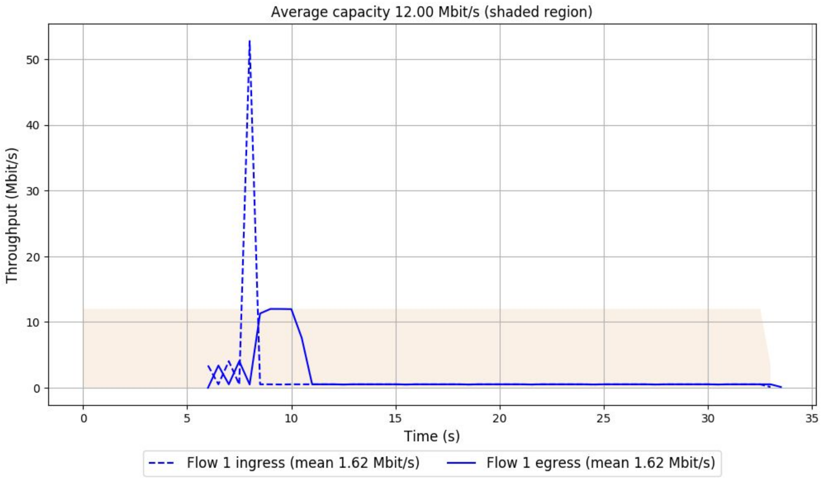
Orange Graph - First Checkpoint, Green Graph - Last Checkpoint

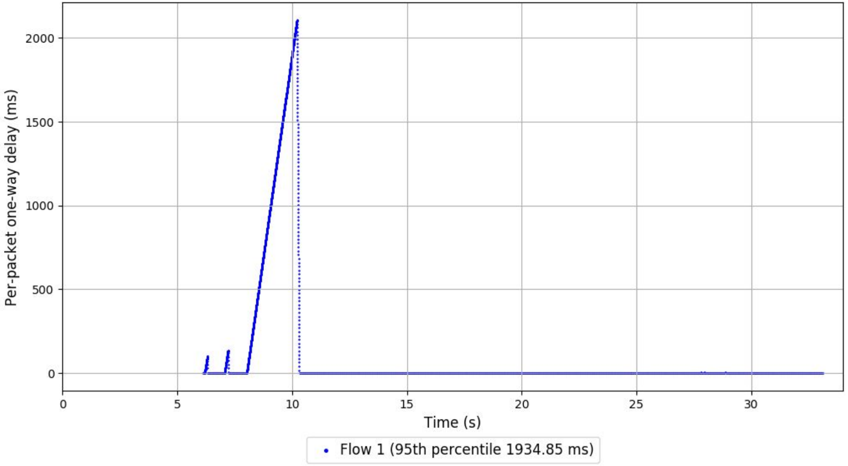
TD3: Divergence

Orange Graph - First Checkpoint, Blue Graph - Last Checkpoint



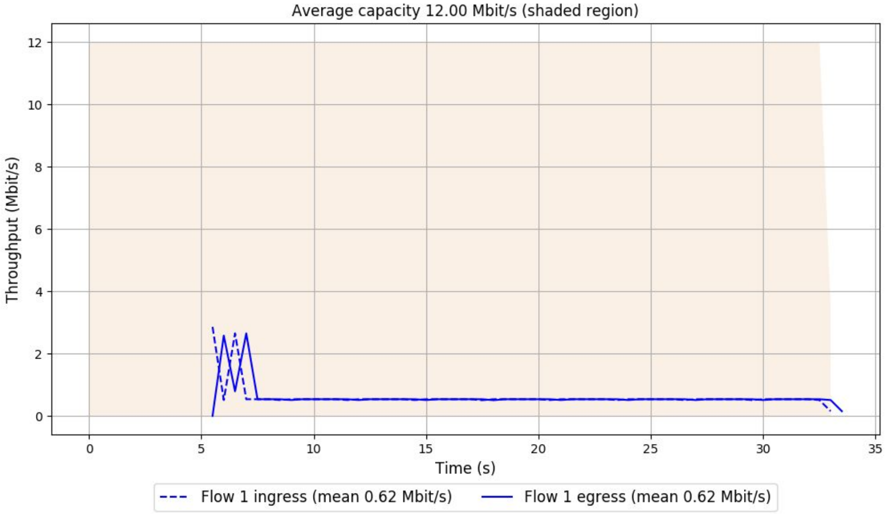
Test 1: TD3 [64, 32]

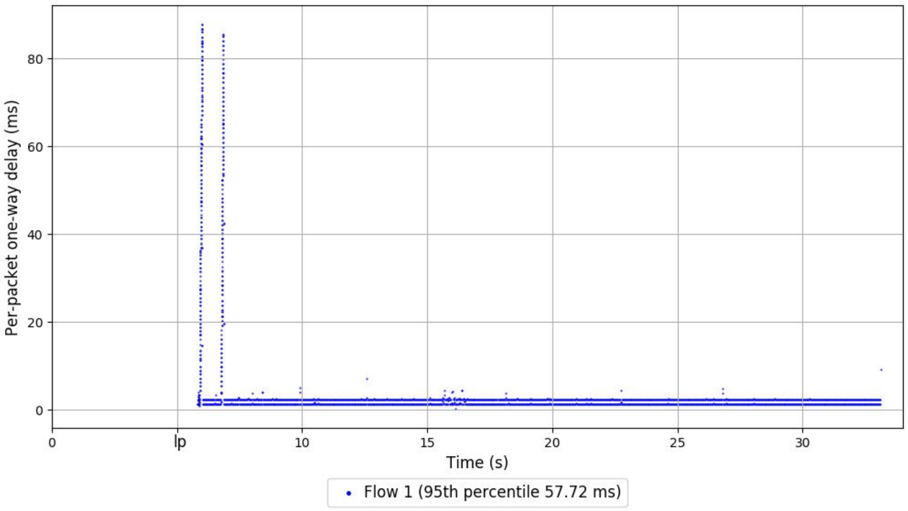




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	1.62	1934.85	0.00

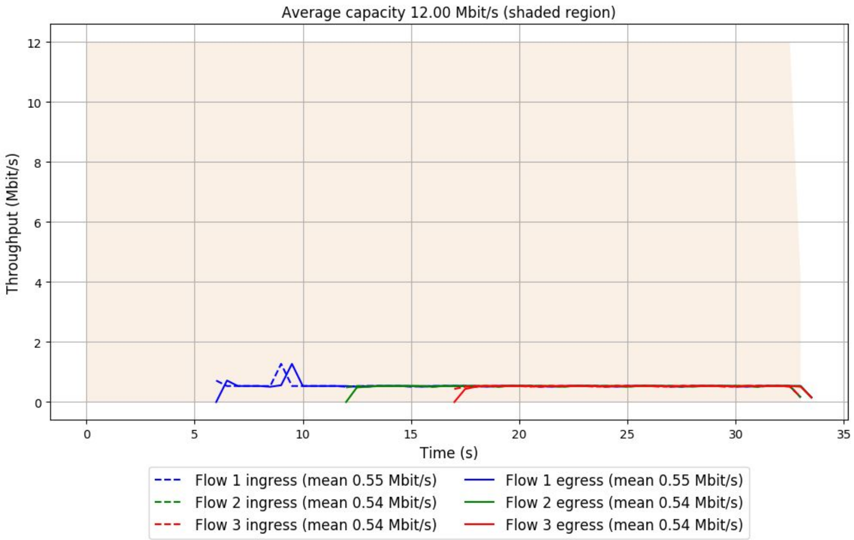
Test 2: TD3 [64, 32]

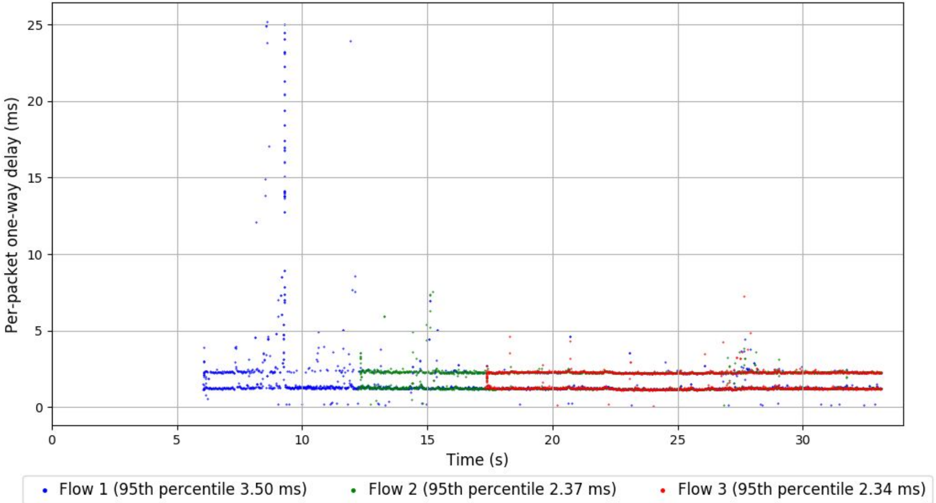




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	5	1.81	349.16	0.27

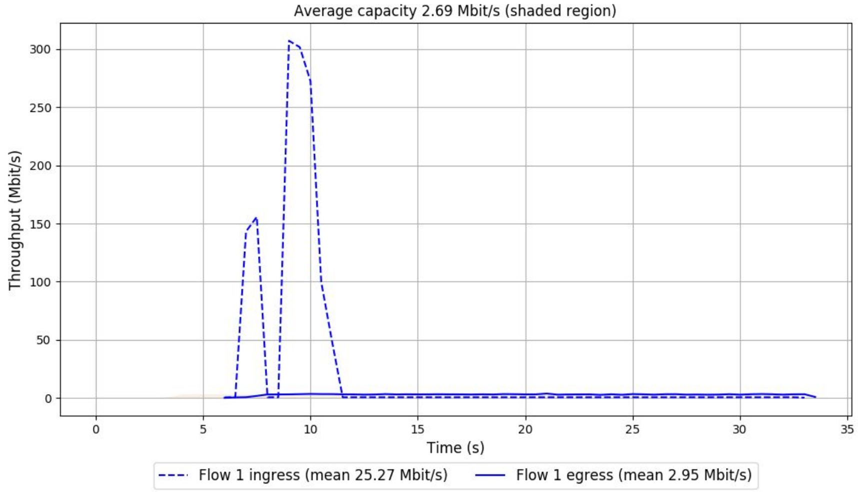
Test 3: TD3 [64, 32]

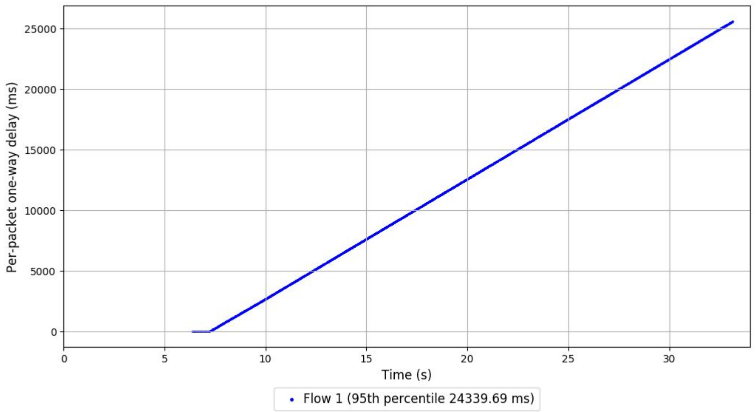




scheme	# runs	mean avg tput (Mbit/s) flow 1	flow 2	flow 3	mean 95th-%ile delay (ms) flow 1	flow 2	flow 3	mean loss rate (%) flow 1	flow 2	flow 3
PCC-RL	1	0.55	0.54	0.54	3.50	2.37	2.34	0.00	0.00	0.15

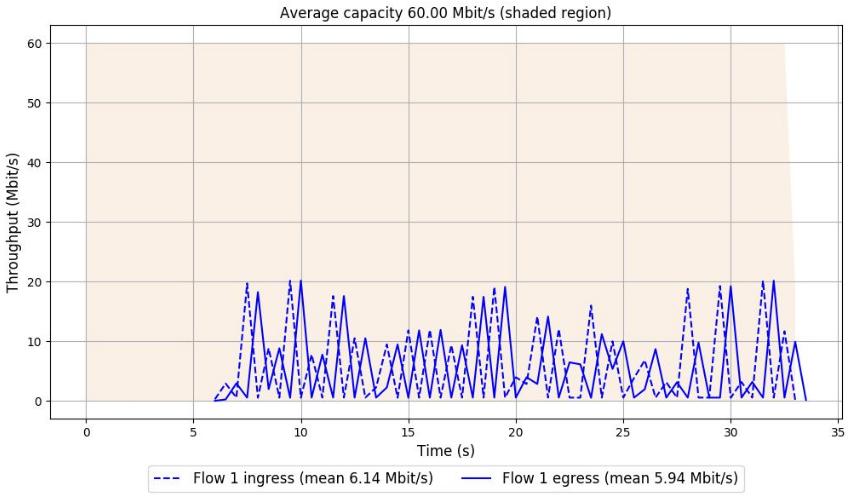
Test 4: TD3 [64, 32]

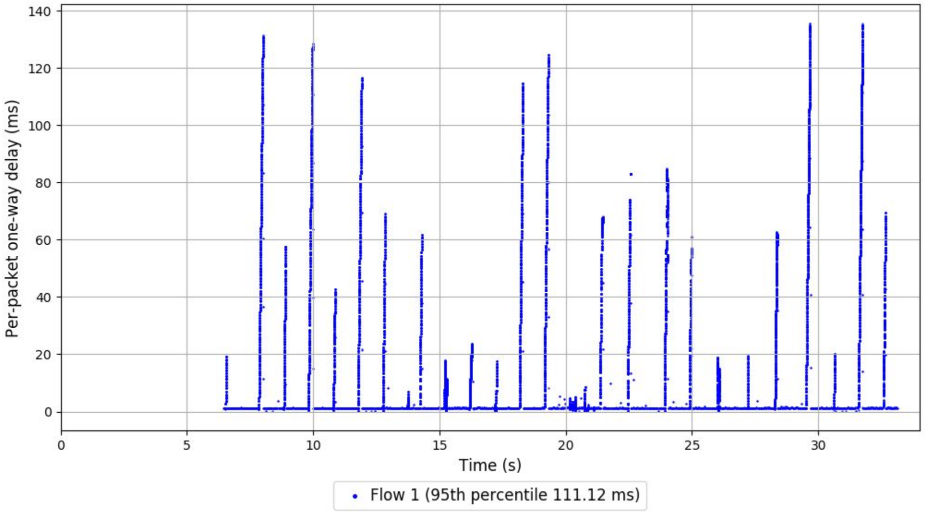




		mean avg tput (Mbit/s)			mean 95th-%ile delay (ms)			mean loss rate (%)		
scheme	# runs	flow 1	flow 2	flow 3	flow 1	flow 2	flow 3	flow 1	flow 2	flow 3
PCC-RL	1	2.53	1.02	9.76	2820.67	7738.59	8570.39	6.41	95.32	32.84

Test 5: TD3 [64,32]



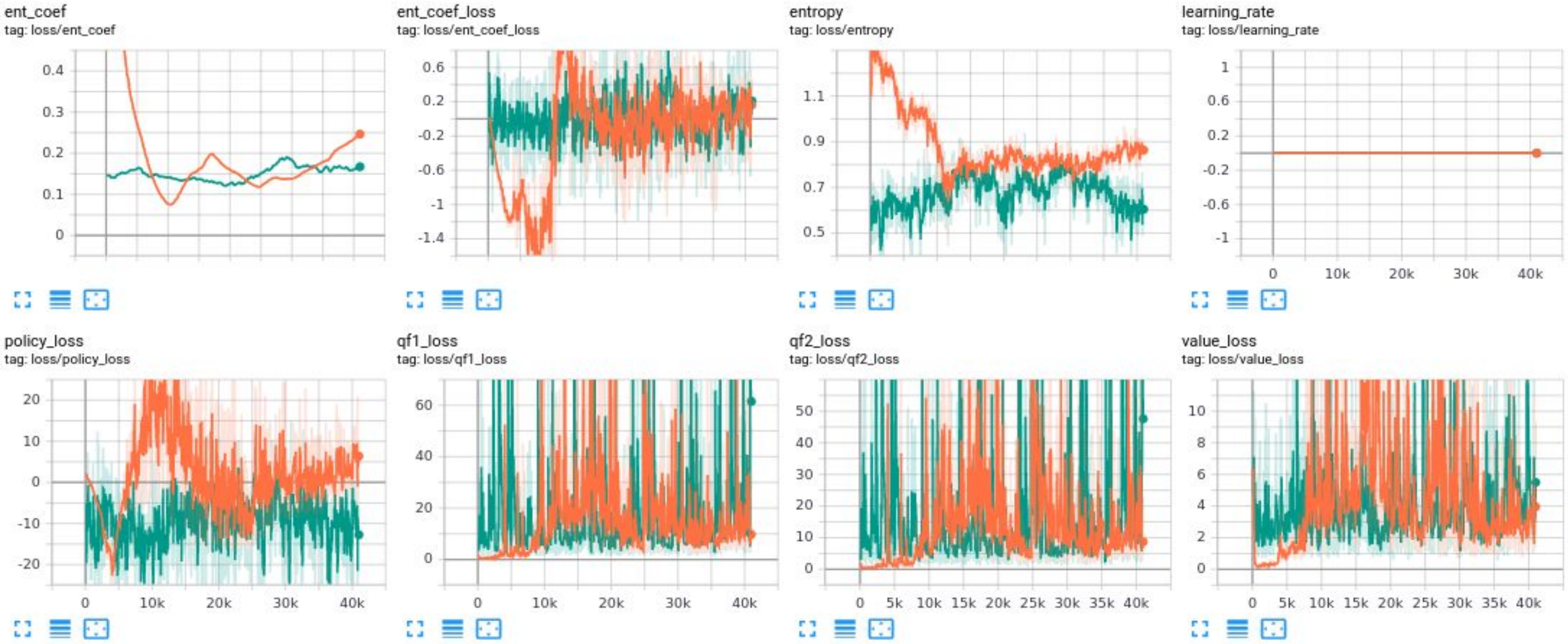


scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	5.94	111.12	3.28

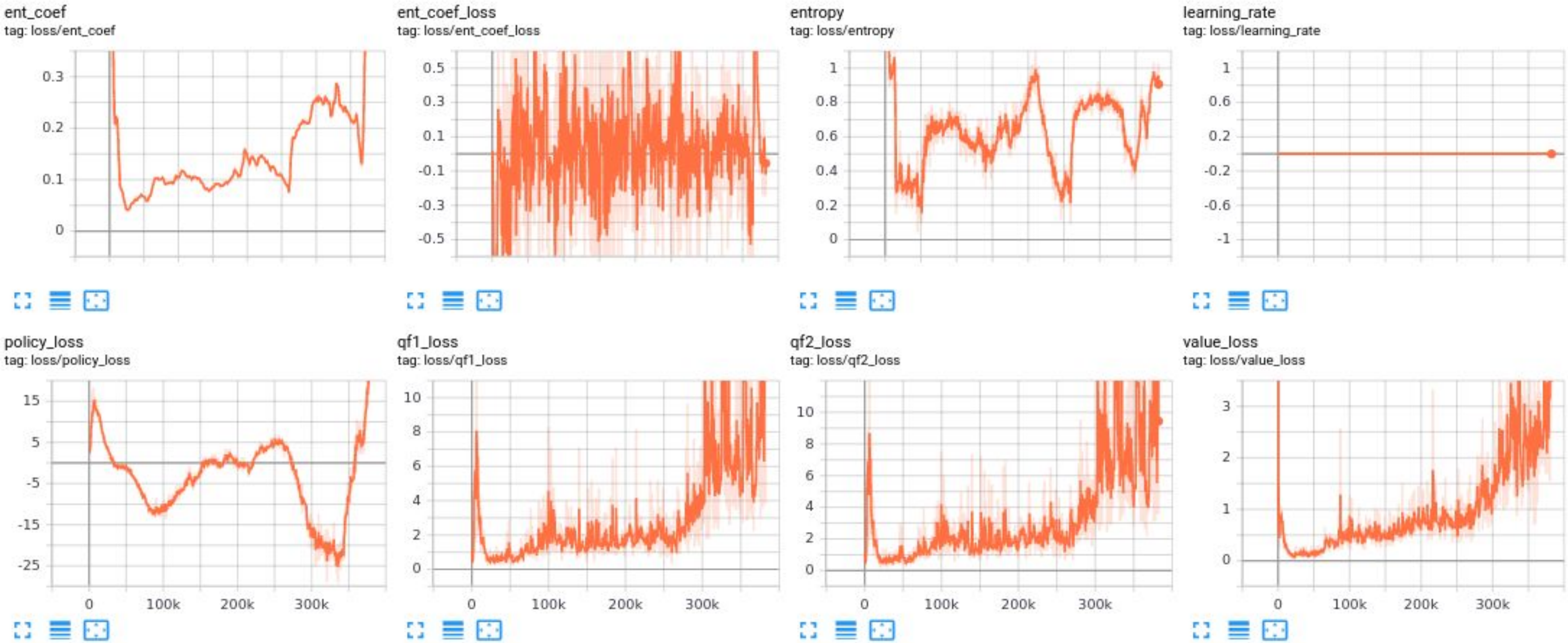
Aurora: SAC

- RL Algorithm: Soft Actor Critic
- Architecture (of NN): [64, 32]
- History Length: 10
- Features:
- Gamma: 0.99
- Soft Actor Critic: SAC is an off-policy maximum entropy deep reinforcement learning with a stochastic actor. SAC incorporates the double Q-learning trick from TD3. A key feature of SAC, and a major difference with common RL algorithms, is that it is trained to maximise a trade-off between expected return and entropy, a measure of randomness in the policy.

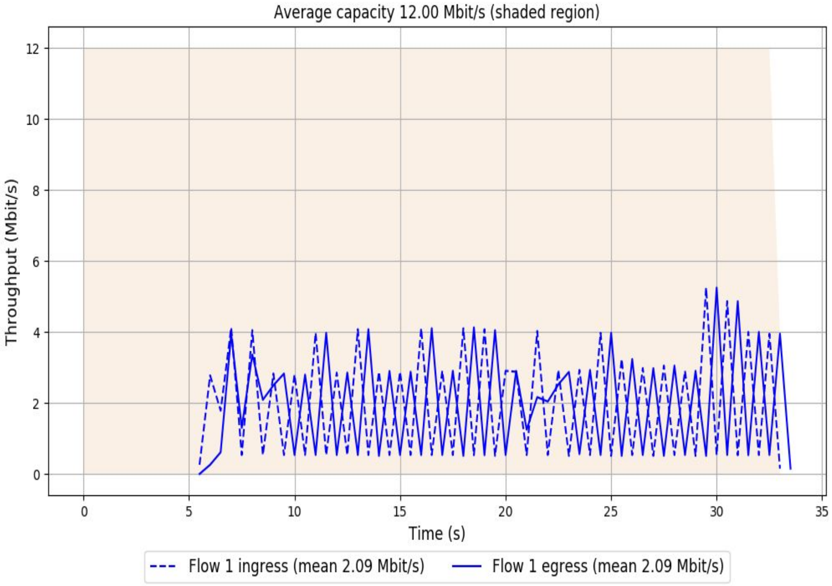
SAC [64, 32]: Training Statistics

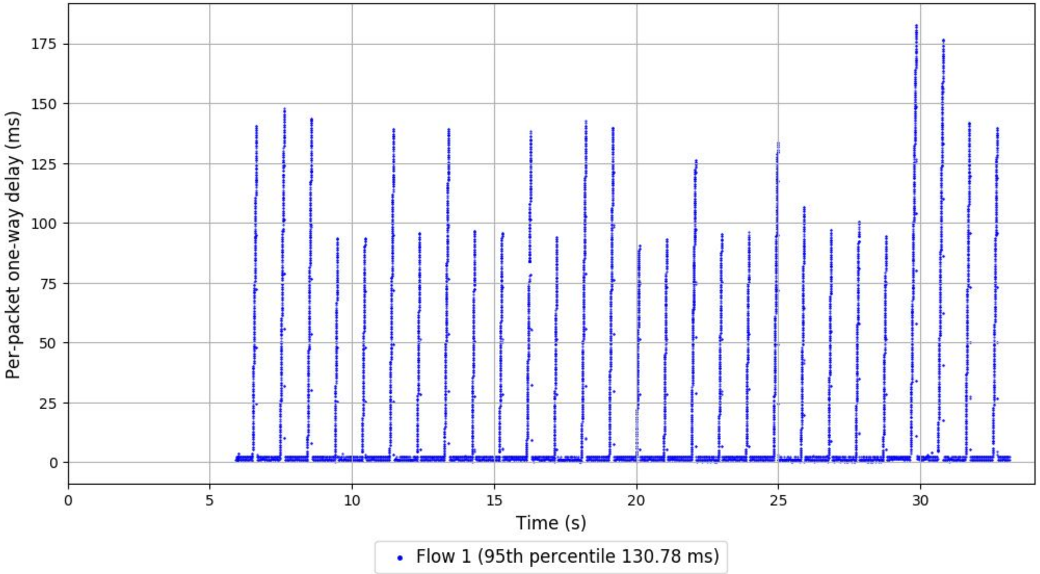


SAC: Divergence



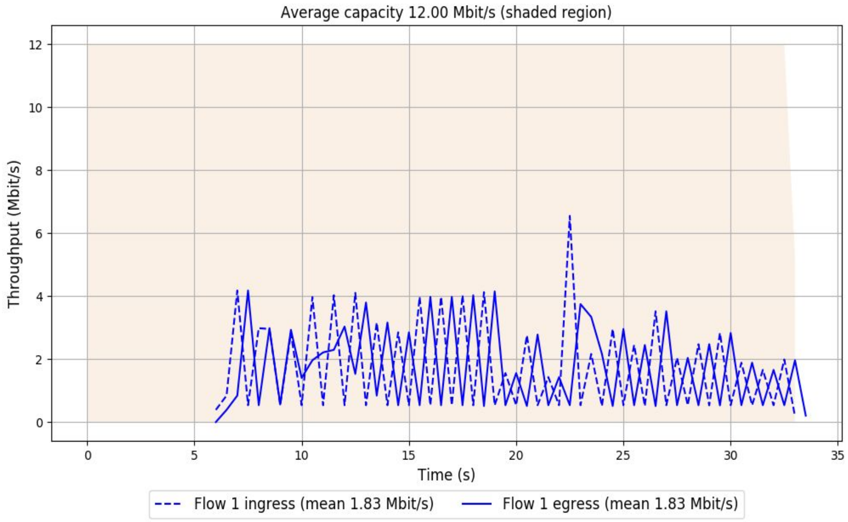
Test 1: SAC [64, 32]

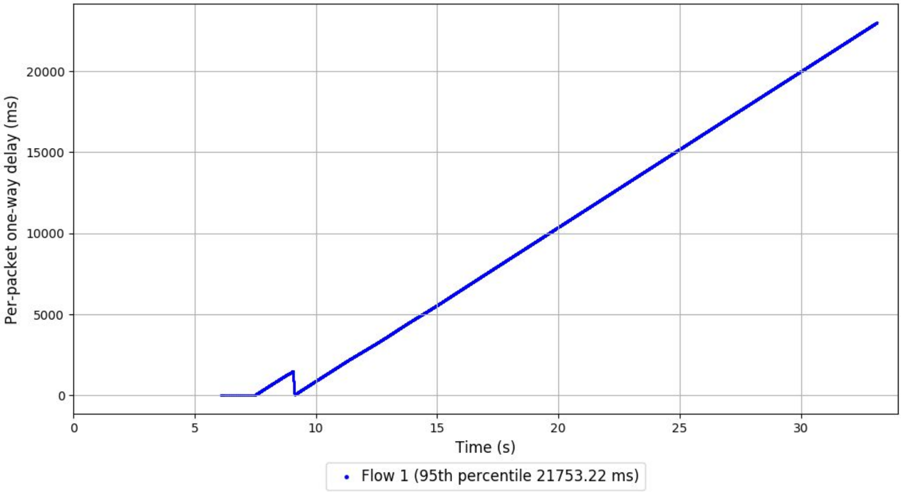




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	2.09	130.78	0.00

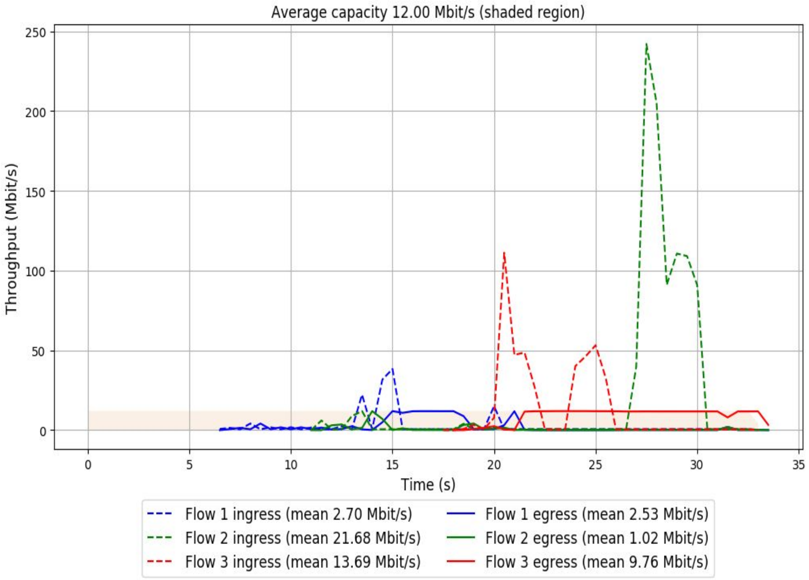
Test 2: SAC [64, 32]

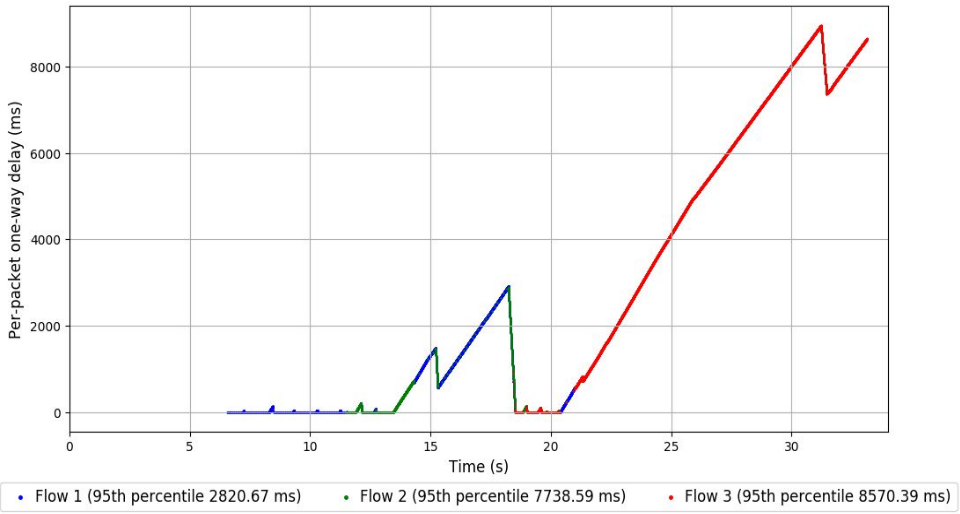




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	5	3.96	4500.13	16.24

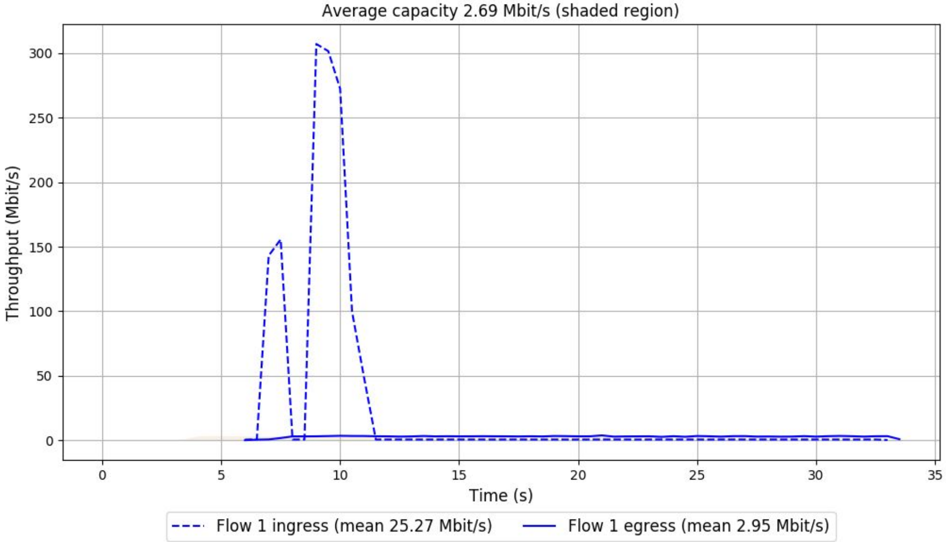
Test 3: SAC [64, 32]

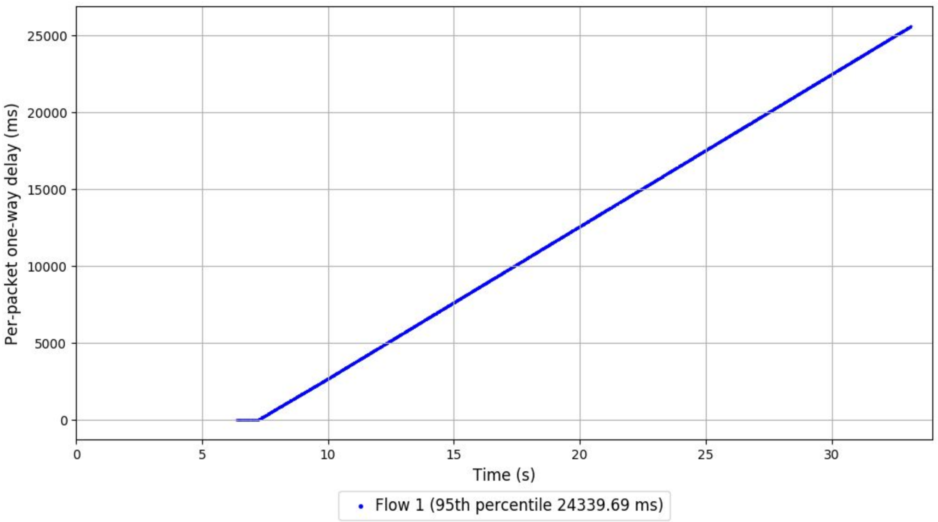




scheme	# runs	mean avg tput (Mbit/s) flow 1	flow 2	flow 3	mean 95th-%ile delay (ms) flow 1	flow 2	flow 3	mean loss rate (%) flow 1	flow 2	flow 3
PCC-RL	1	2.53	1.02	9.76	2820.67	7738.59	8570.39	6.41	95.32	32.84

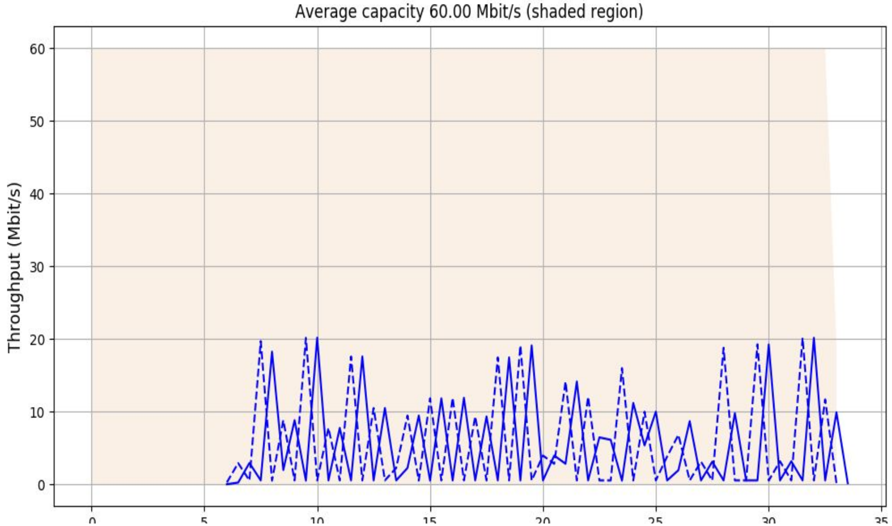
Test 4: SAC [64, 32]

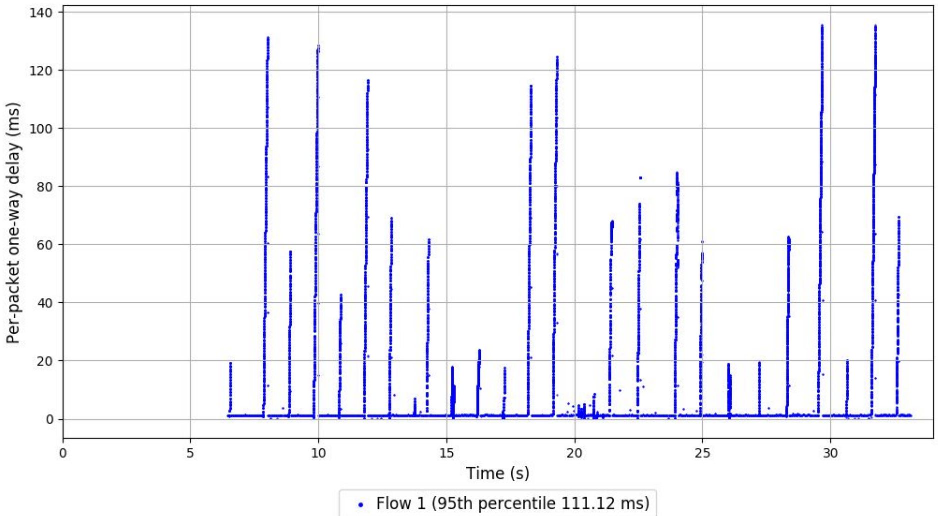




scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	2.95	24339.69	88.31

Test 5: SAC [64,32]





scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
PCC-RL	1	5.94	111.12	3.28

Multi-Objective: Reward Engineering

Reward is a linear combination of throughput, latency, loss

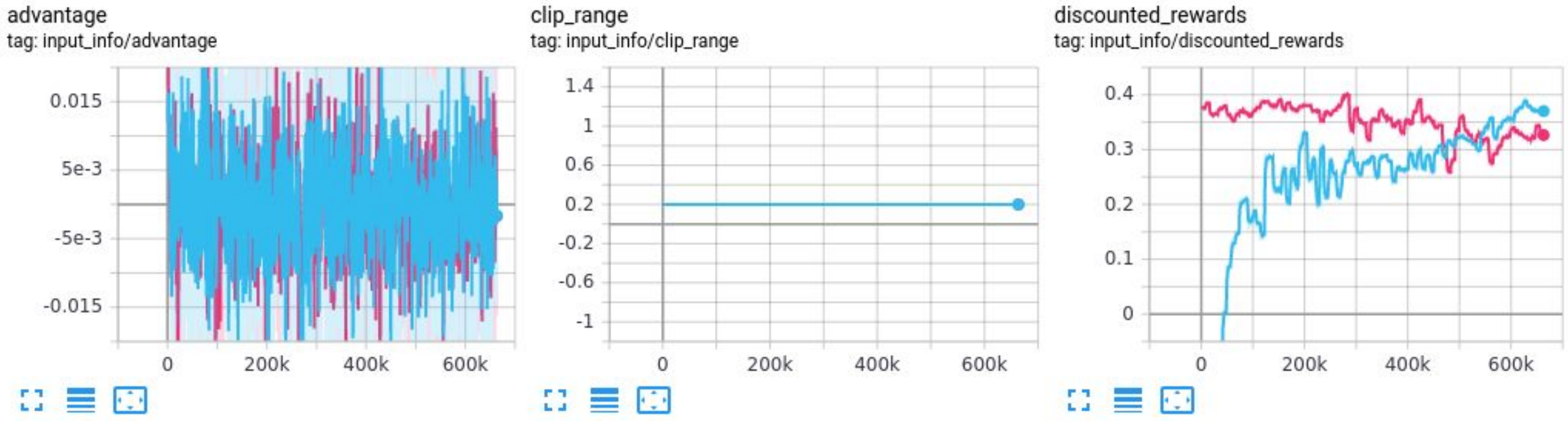
How does the performance change with the individual weights

Reward in earlier experiments: 10*throughput - 1000*latency - 2000*loss

High throughput (expected): 20*throughput - 1000*latency - 2000*loss

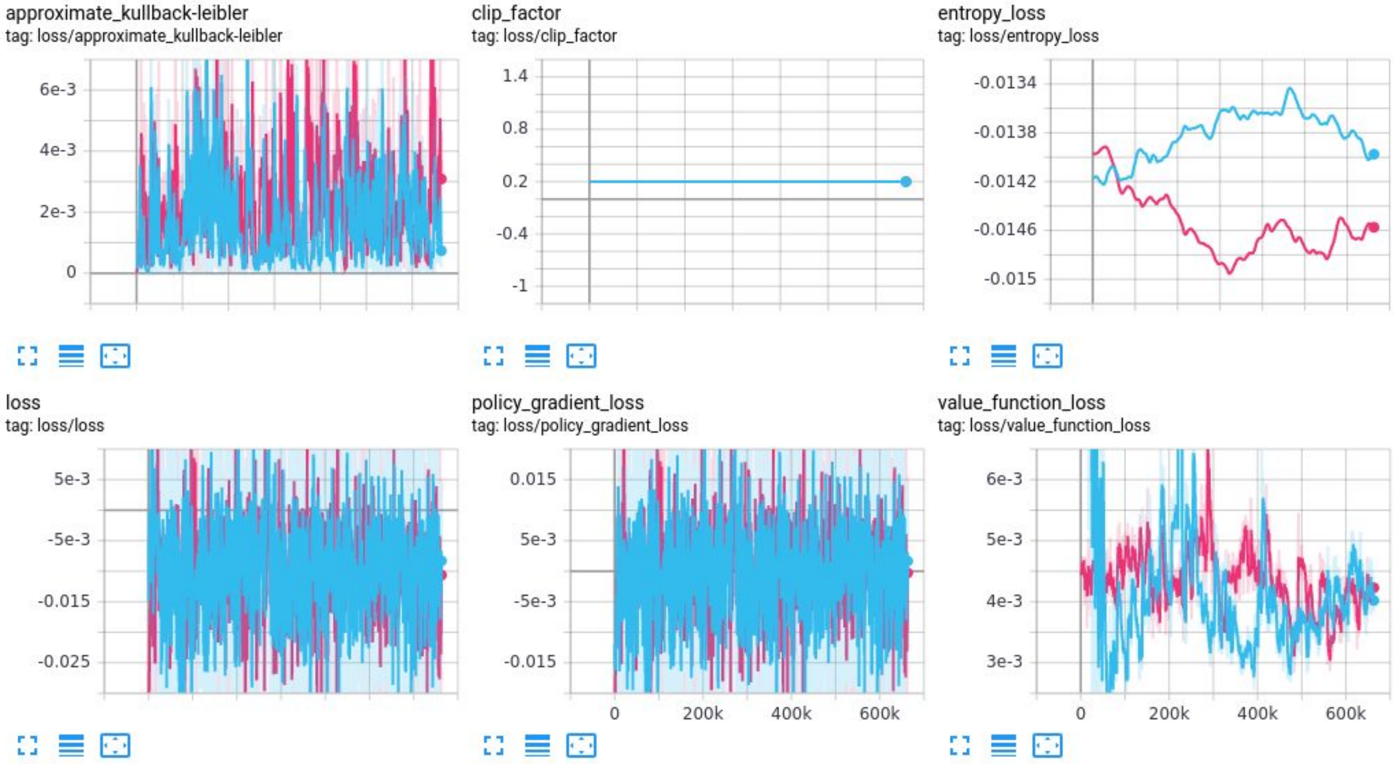
Low latency (expected): 5*throughput - 1000*latency - 2000*loss

PPO [64, 32] HighT: Training Statistics

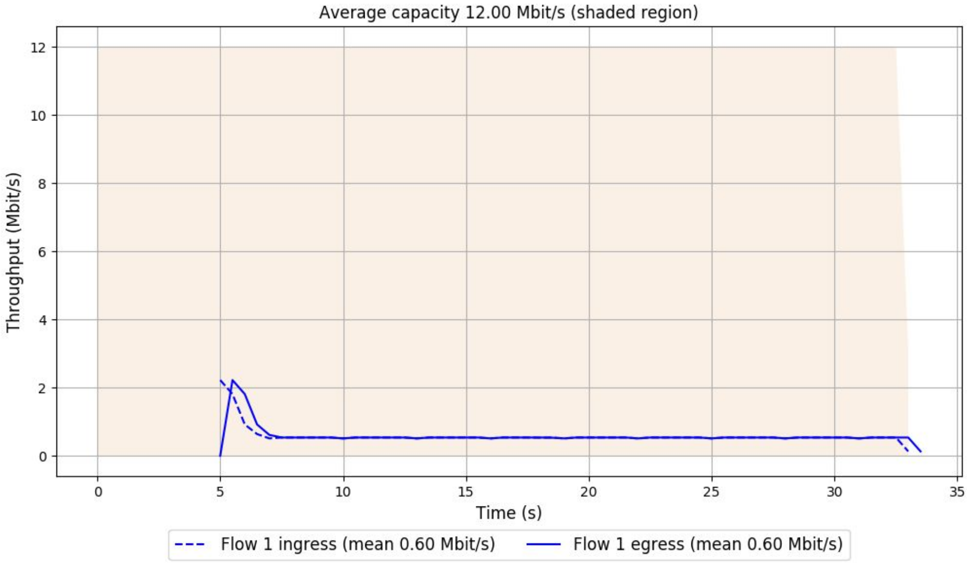


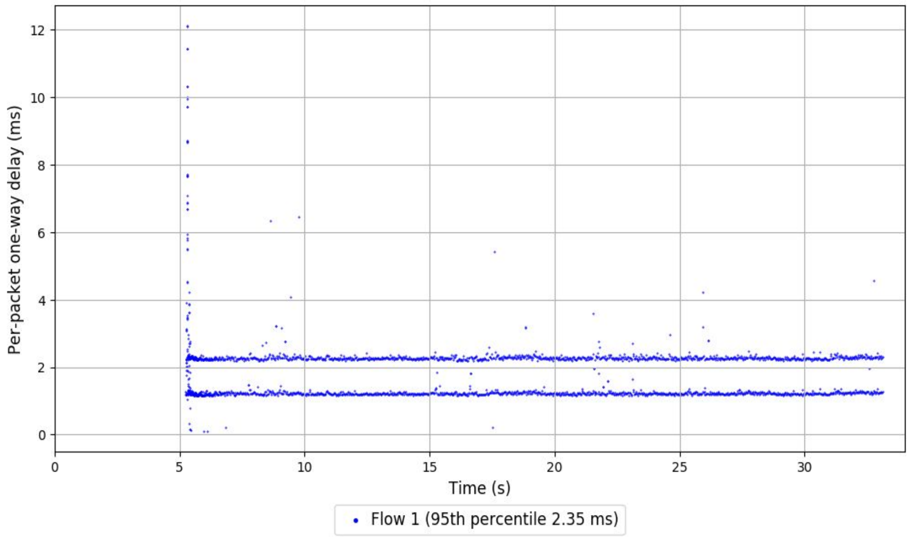
Blue graph - First Checkpoint, Pink graph - Last Checkpoint

PPO [64, 32] HighT: Training Statistics

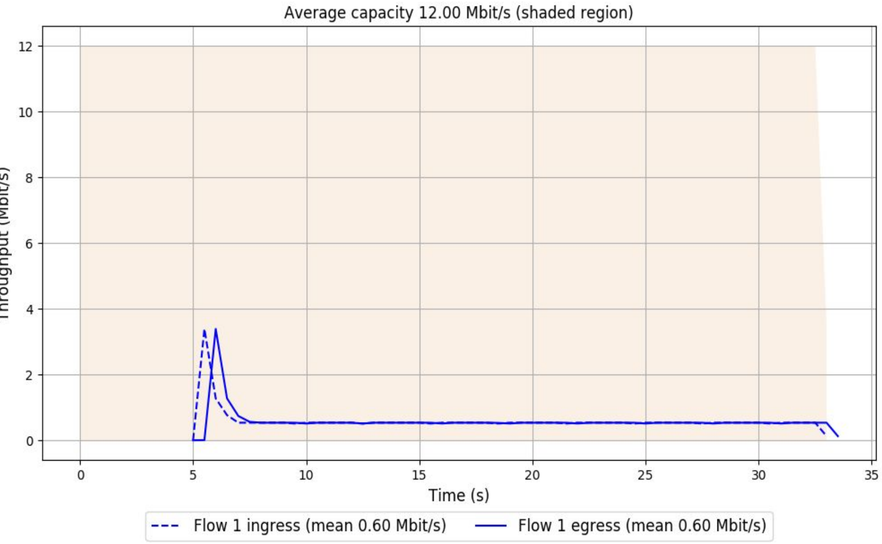


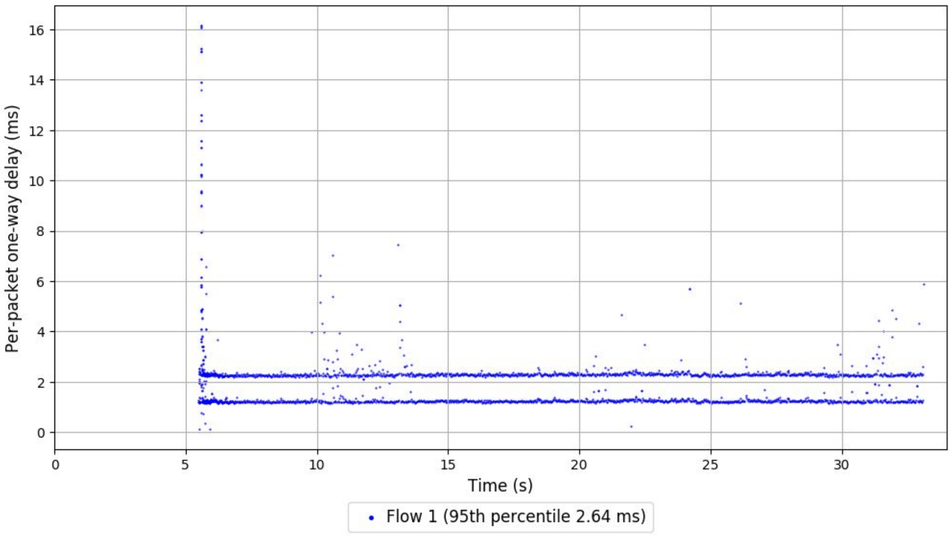
Test 1: PPO [64, 32] HighT



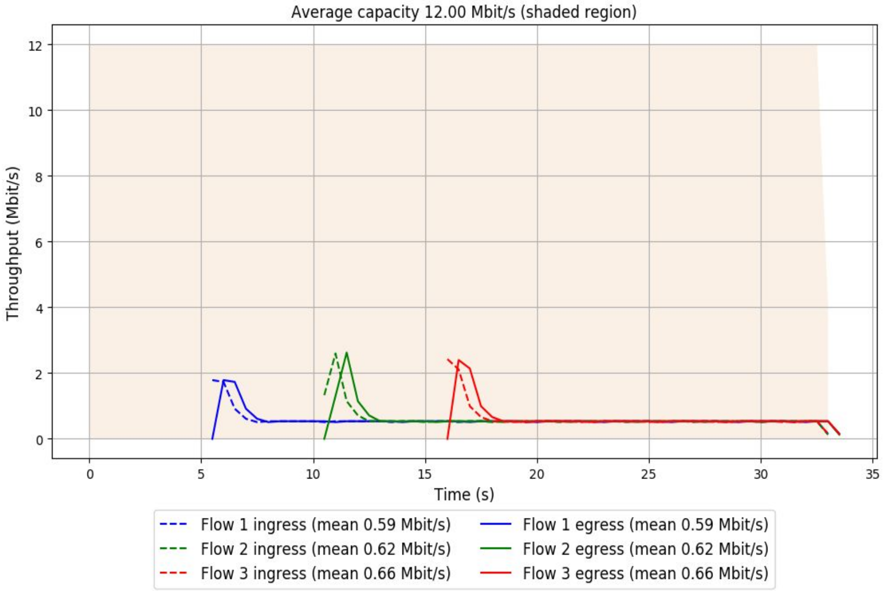


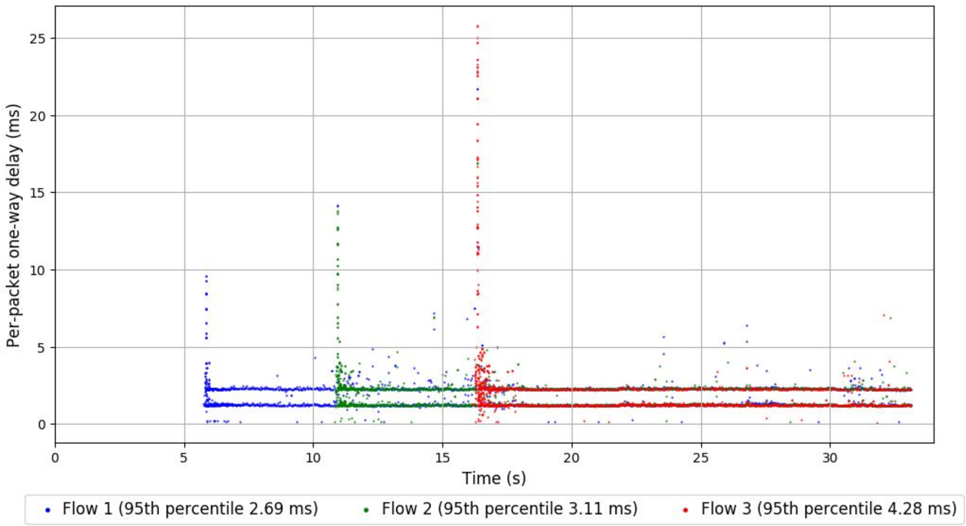
Test 2: PPO [64, 32] HighT



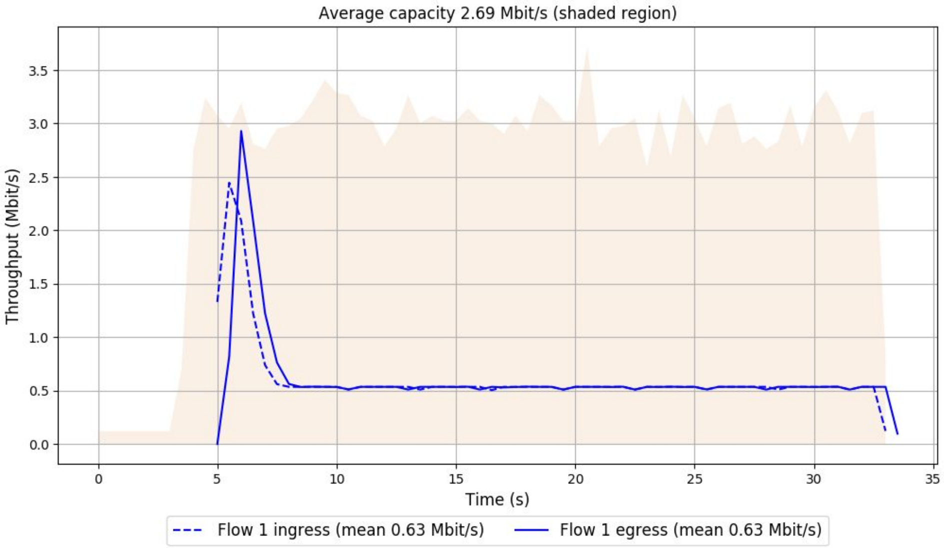


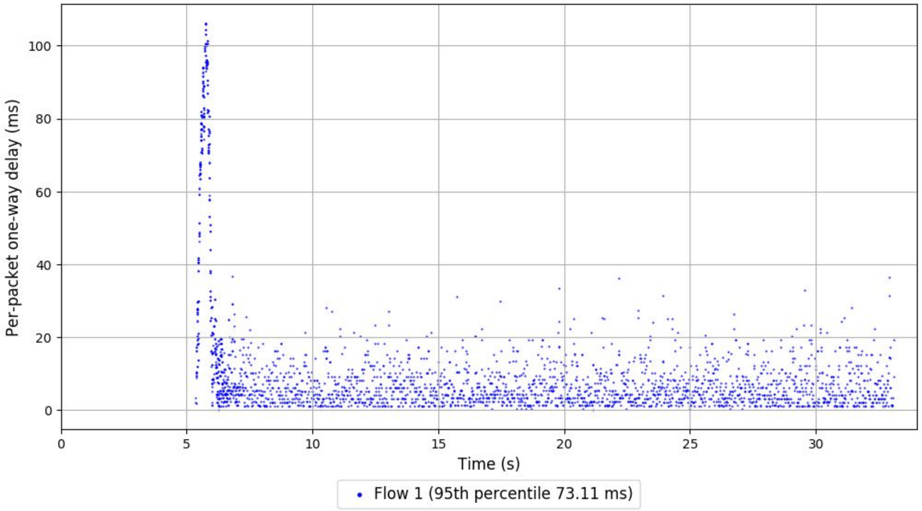
Test 3: PPO [64, 32] HighT



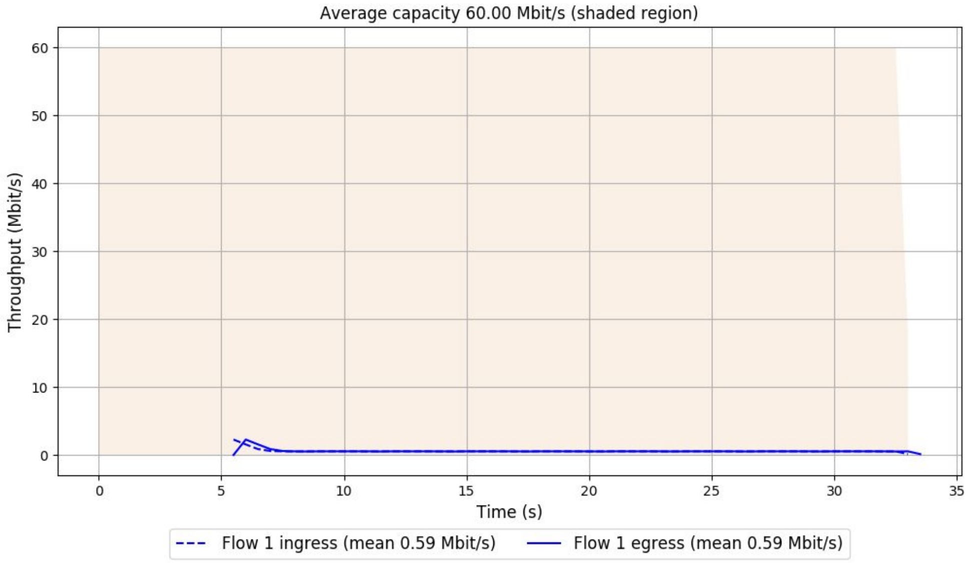


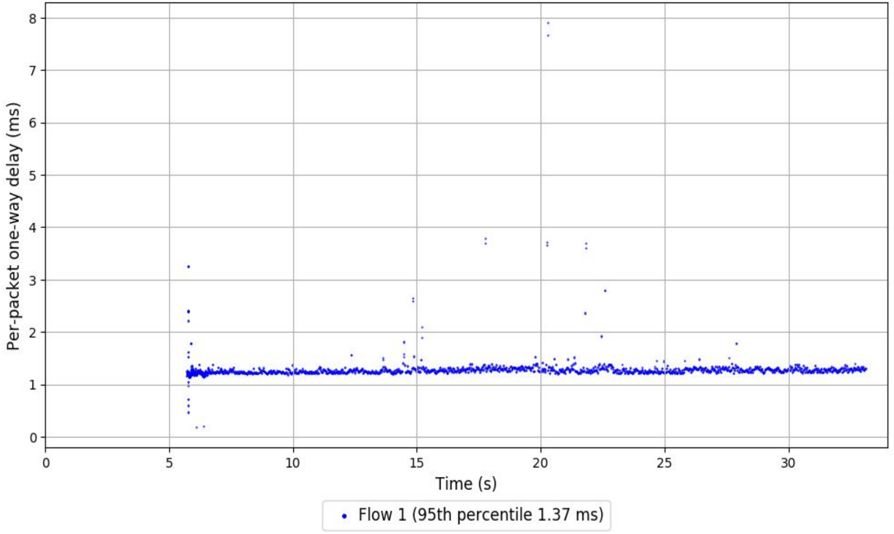
Test 4: PPO [64, 32] HighT



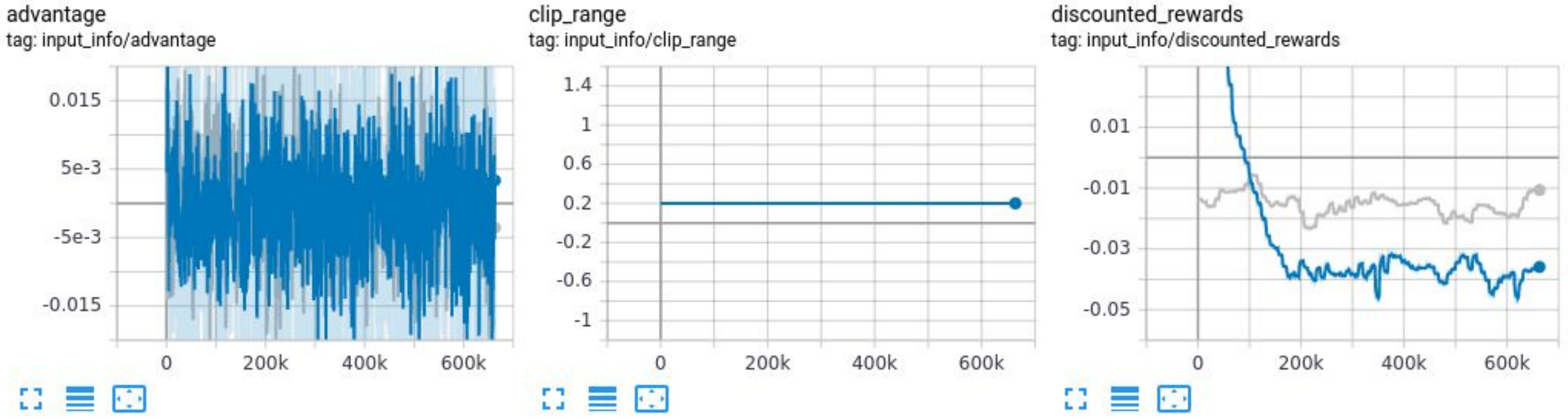


Test 5: PPO [64,32] HighT



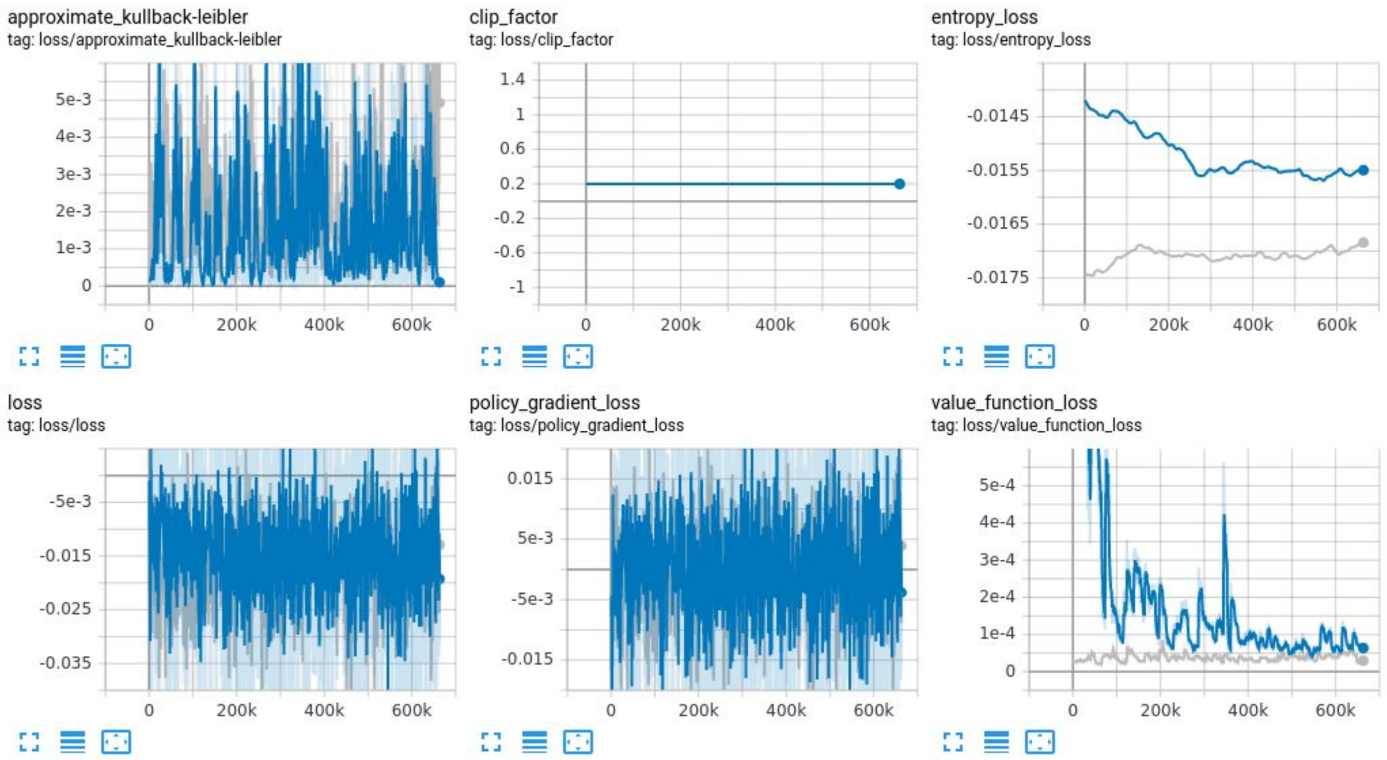


PPO [64, 32] LowLat: Training Statistics

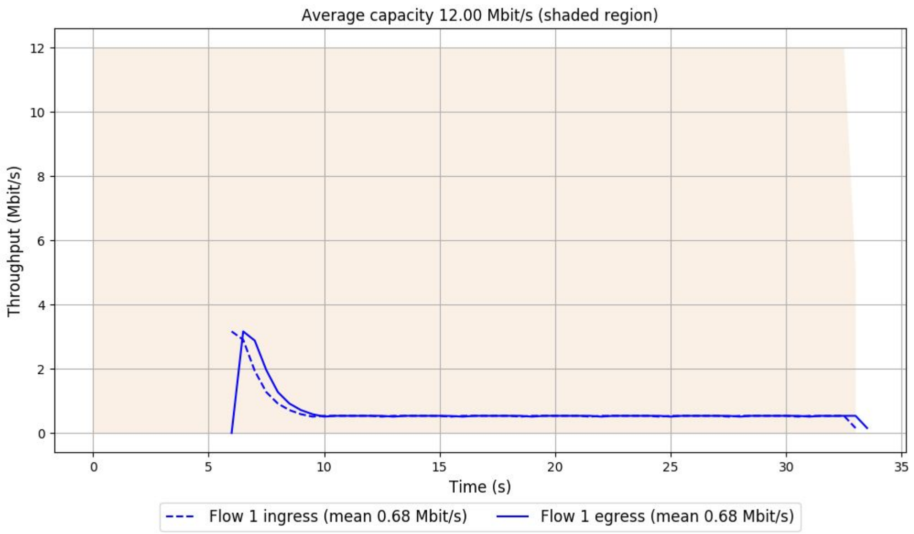


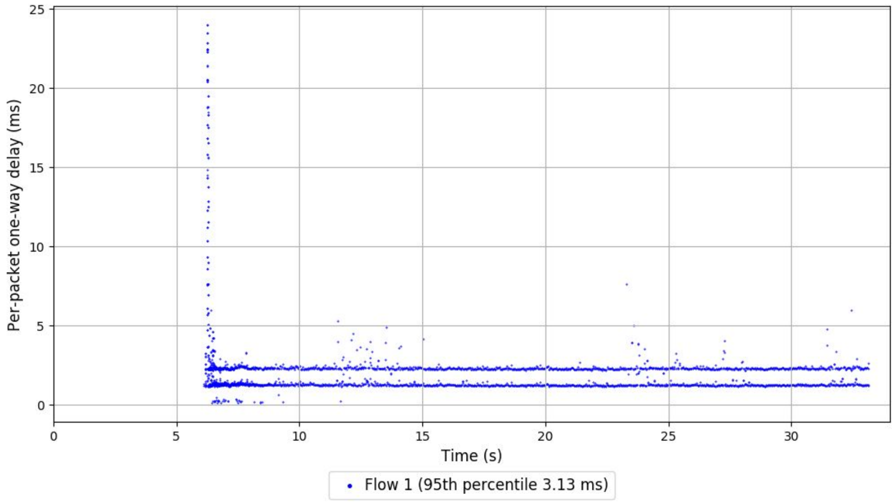
Blue Graph - First Checkpoint, Grey Graph - Last Checkpoint

PPO [64, 32] LowLat: Training Statistics

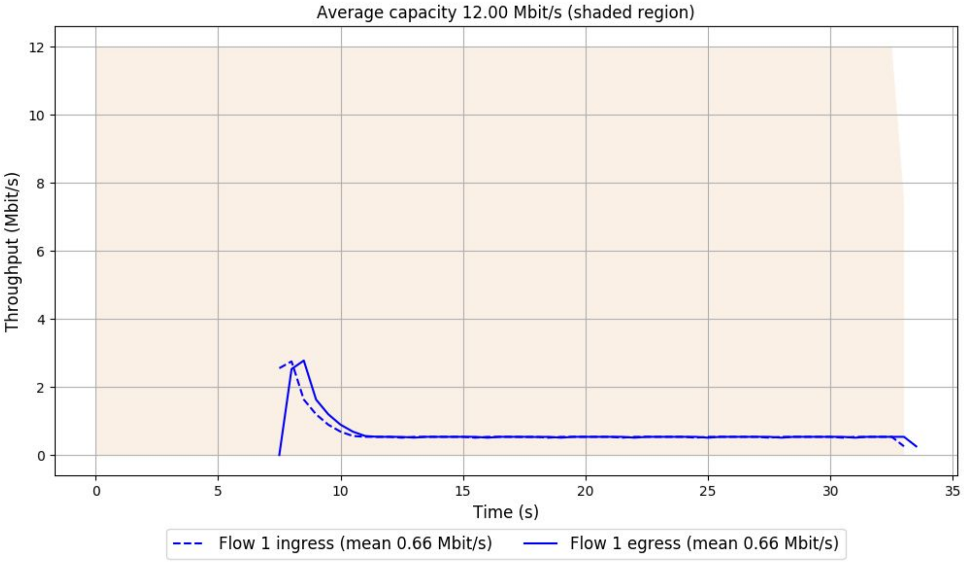


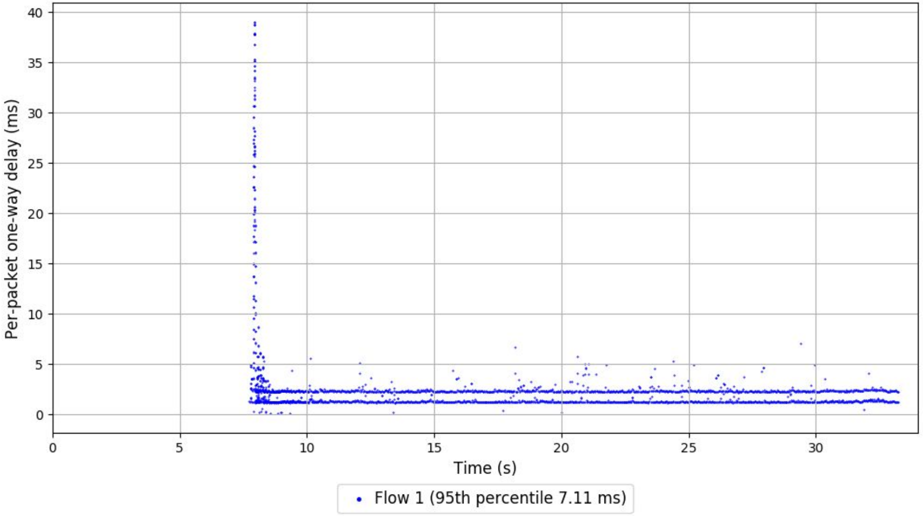
Test 1: PPO [64, 32] LowLat



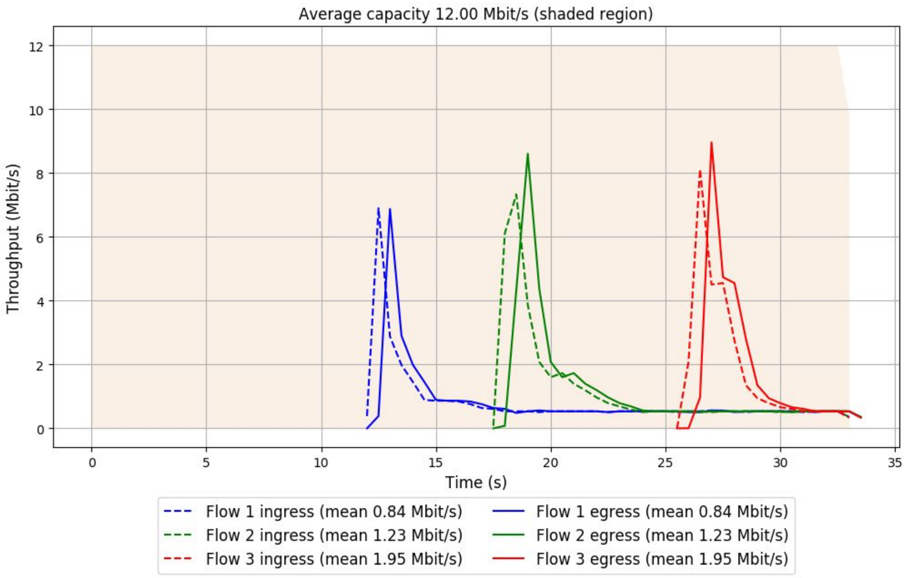


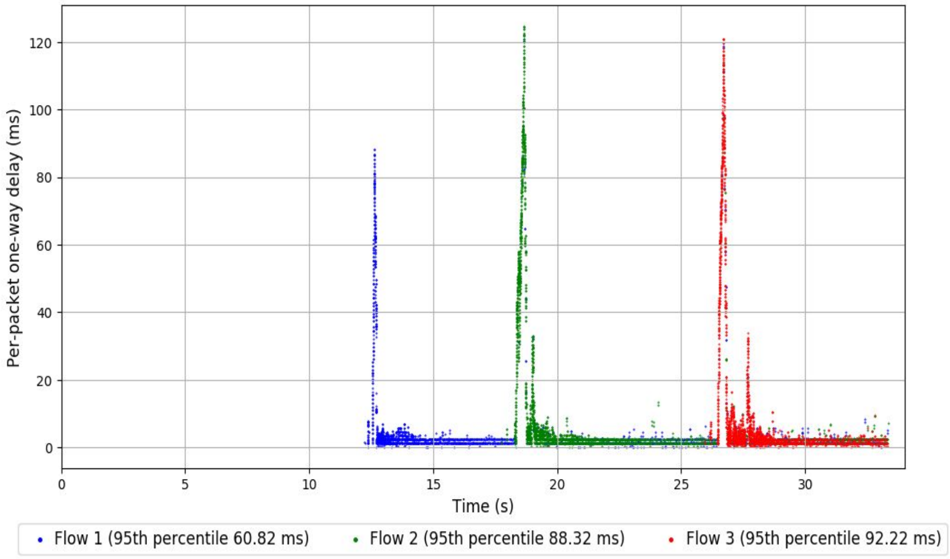
Test 2: PPO [64, 32] LowLat



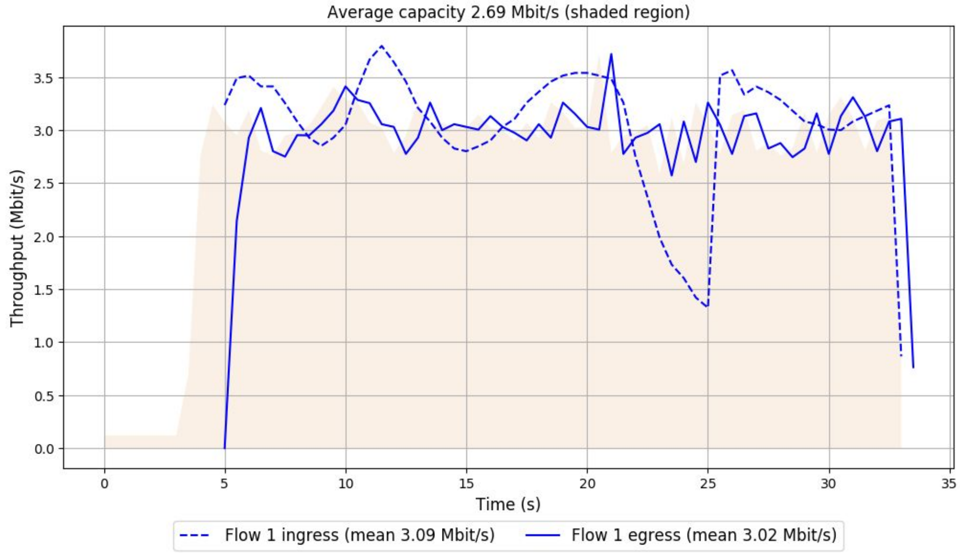


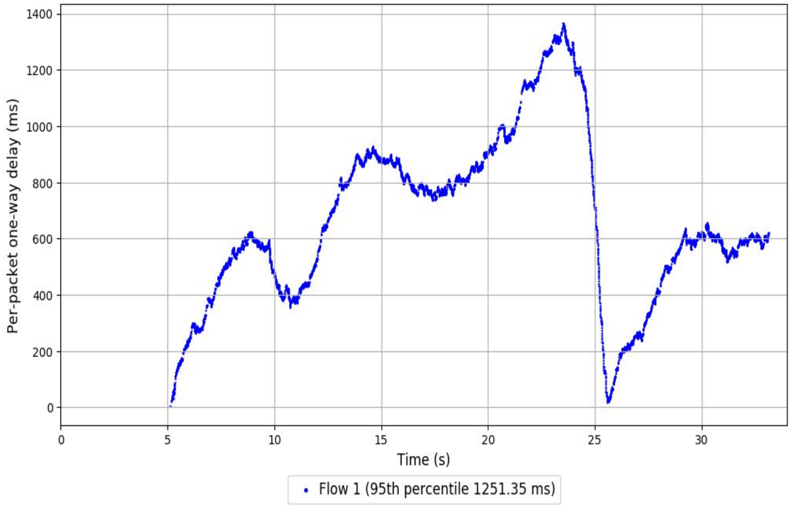
Test 3: PPO [64, 32] LowLat



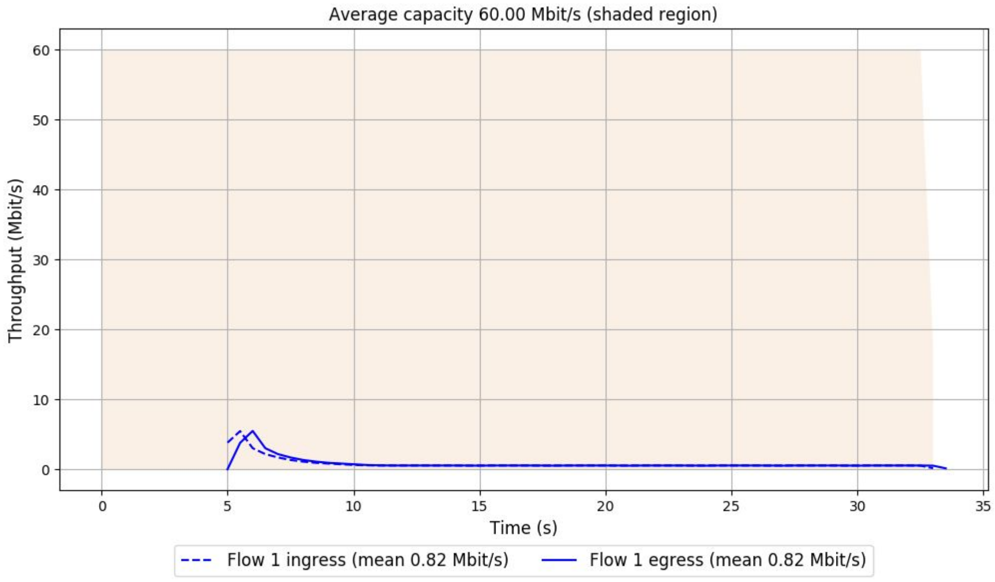


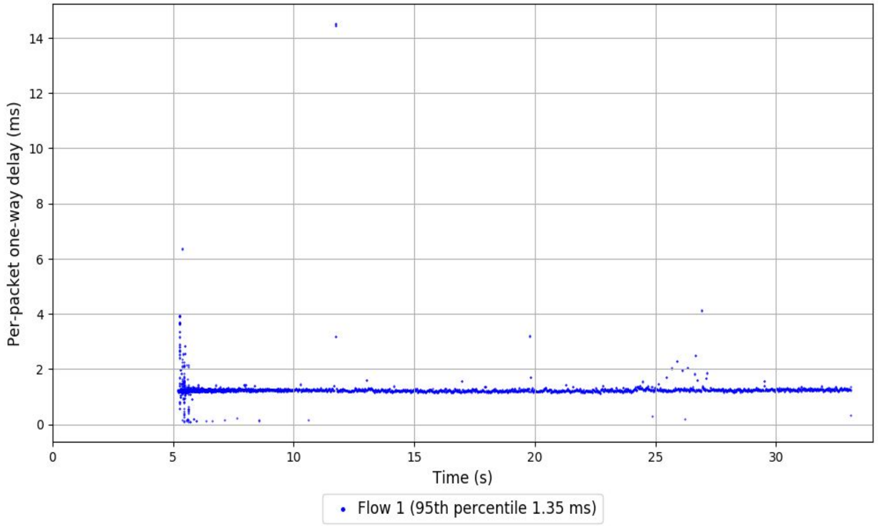
Test 4: PPO [64, 32] LowLat





Test 5: PPO [64,32] LowLat





Roadblocks

- Code/libraries used are outdated
  - Python2 -> Python3
  - Tensorflow 1.14 -> Tensorflow 2+
- Outdated kernel issues with Orca and DeepCC
- All models trained on CPU

Further Avenues

- Model-based approaches
- Meta-Learning
- Reward Engineering
- Competitive Learning

References

[1] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, Aviv Tamar. A Deep Reinforcement Learning Perspective on Internet Congestion Control. *Proceedings of the 36th International Conference on Machine Learning,* PMLR 97:3050-3059, 2019.

[2] Yiqing Ma, Han Tian, Xudong Liao, Junxue Zhang, Weiyan Wang, Kai Chen, and Xin Jin. 2022. Multi-objective congestion control. In Proceedings of the Seventeenth European Conference on Computer Systems (EuroSys '22). Association for Computing Machinery, New York, NY, USA, 218–235. https://doi.org/10.1145/3492321.3519593

[3] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: the training ground for internet congestion-control research. In Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC '18). USENIX Association, USA, 731–743.

[4] https://stable-baselines.readthedocs.io/en/master/index.html

[5] https://spinningup.openai.com/en/latest/index.html