End-to-End is Not Enough: Towards a Coordinated Congestion Control (C³)

Michael König, Martina Zitterbart

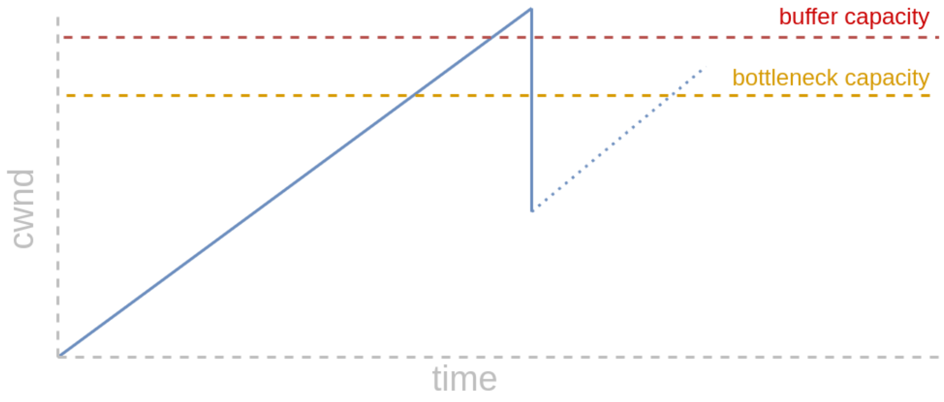
Limits of Classical Congestion Controls

Decades of research
Multitude of congestion control approaches exist (Cubic, Vegas, BBR, Copa, …)

Operate in an end-to-end (E2E) fashion

- Distributed algorithm for each sender/flow
- Knowledge only implicit (estimated)
- No distinction between long- & short-lived flows
  - After flow ends: 80.27 % of flows still in Slow Start [1]

Long-lived flows
Slow convergence and oscillations



Short-lived flows
Longer flow completion times

[1] Nie, Xiaohui, et al. "Dynamic TCP initial windows and congestion control schemes through reinforcement learning." IEEE Journal on Selected Areas in Communications 37.6 (2019): 1231-1247.

Related ML-based Congestion Controls

Approach	Online-Training	View	Flow Type Distinction	Remarks
Remy [1], PCC-Allegro/-Vivace, …	✗	Global	✗	"First" ML-based approach
Aurora/PCC-RL [2]	✓	E2E	✗	Continuation of PCC
MVFST-RL [3]	✓	E2E	✗	Delayed actions: Usage of action history
AUTO [4]	✓	E2E	✗	Preferences for different objectives
Orca [5]	✓	E2E	✗	Two combined control loops
Eagle [6]	✓	E2E	✗	Synthesizes behavior of BBRv1
Iroko [7]	✓	Global	✗	Only static & manually specified topologies
TCP-RL [8]	✓	Group	✓	Determine initial CWND and CCA
IW-DRL [9]	✓	E2E	(✓)	Determine initial CWND

[1] Winstein, Keith, and Hari Balakrishnan. "Tcp ex machina: Computer-generated congestion control." ACM SIGCOMM Computer Communication Review 43.4 (2013): 123-134.
[2] Jay, Nathan, et al. "A deep reinforcement learning perspective on internet congestion control." International Conference on Machine Learning. PMLR, 2019.
[3] Sivakumar, Viswanath, et al. "Mvfst-rl: An asynchronous rl framework for congestion control with delayed actions." arXiv preprint arXiv:1910.04054 (2019).
[4] Li, Xu, et al. "{AUTO}: Adaptive Congestion Control Based on {Multi-Objective} Reinforcement Learning for the {Satellite-Ground} Integrated Network." 2021 USENIX Annual Technical Conference
[5] Abbasloo, Soheil, et al. "Classic meets modern: A pragmatic learning-based congestion control for the internet." Proceedings ACM Special Interest Group on Data Communication 2020.
[6] Emara, Salma, Baochun Li, and Yanjiao Chen. "Eagle: Refining congestion control by learning from the experts." IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, 2020.
[7] Ruffy, Fabian, Michael Przystupa, and Ivan Beschastnikh. "Iroko: A framework to prototype reinforcement learning for data center traffic control." arXiv preprint arXiv:1812.09975 (2018).
[8] Nie, Xiaohui, et al. "Dynamic TCP initial windows and congestion control schemes through reinforcement learning." IEEE Journal on Selected Areas in Communications 37.6 (2019): 1231-1247.
[9] Xie, Ruitaoe, et al. "Adaptive online decision method for initial congestion window in 5G mobile edge computing using deep reinforcement learning." IEEE Journal on Selected Areas in Communications 2019

Concept of C³

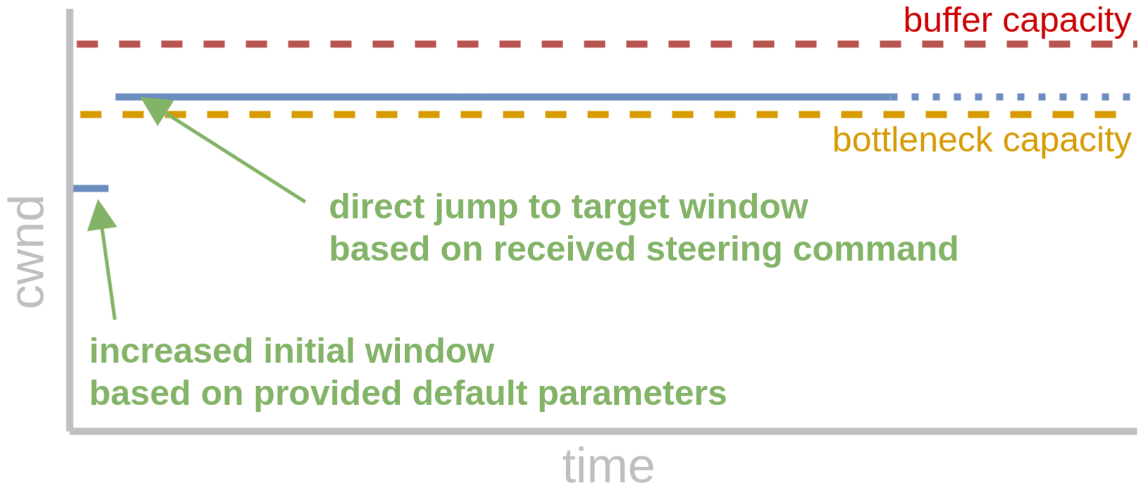
"Congestion Control with SDN-like global view and actions"

- Use global knowledge to compute coordinated control decisions for multiple senders
- Guide the still active end-to-end congestion control
  - Coarse steering intervals – no per-packet control
  - QUIC as transport protocol

- Deep Reinforcement Learning to compute control decisions
  - Many input parameters (large state space)
  - Many possible solutions (large action space)
  - Exploitation of unknown connections
  - Example reward function
    `maximize:`
    `Reward R = α*throughput - β*delay + γ*fairness`

- Reduced flow completion times (FCT)
  - Faster flow start-up via default parameters

- Better resource utilization
  - Virtually instant convergence
  - No oscillations



Basic Architecture of C³

1. Data Collection

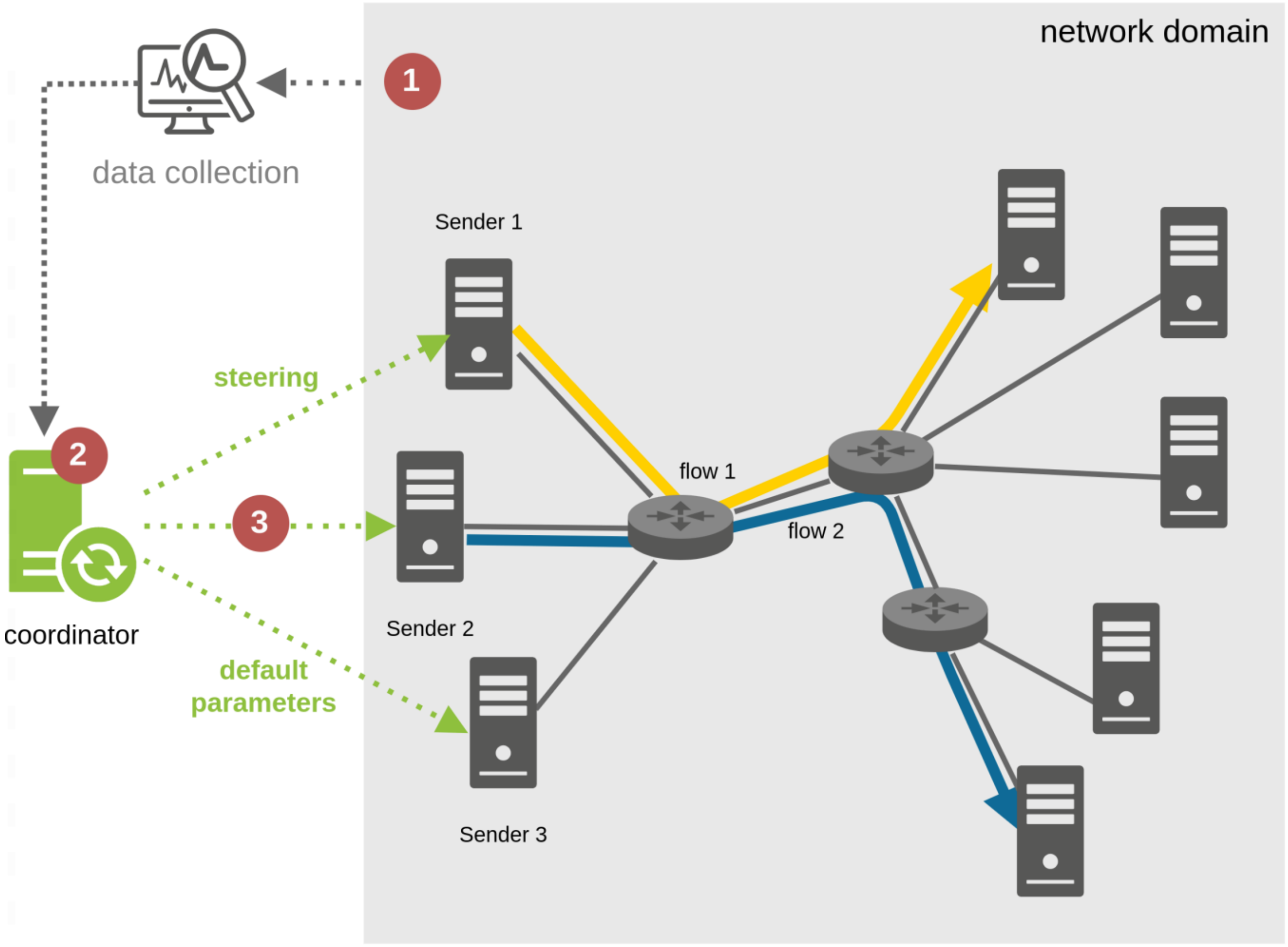
- Global view of the network domain

2. Coordination

- Establish global and explicit network state
- Compute coordinated control decisions (via Reinforcement Learning)

3. Flow Parametrization

- Coordinated steering of multiple senders
  - Coarse steering intervals – no per-packet control
  - Setting Min/Max/Current cWnd
- Providing pro-actively suitable default parameters
  - Setting $cWnd_{initial}$



1. Data Collection

Combine input from multiple entities:

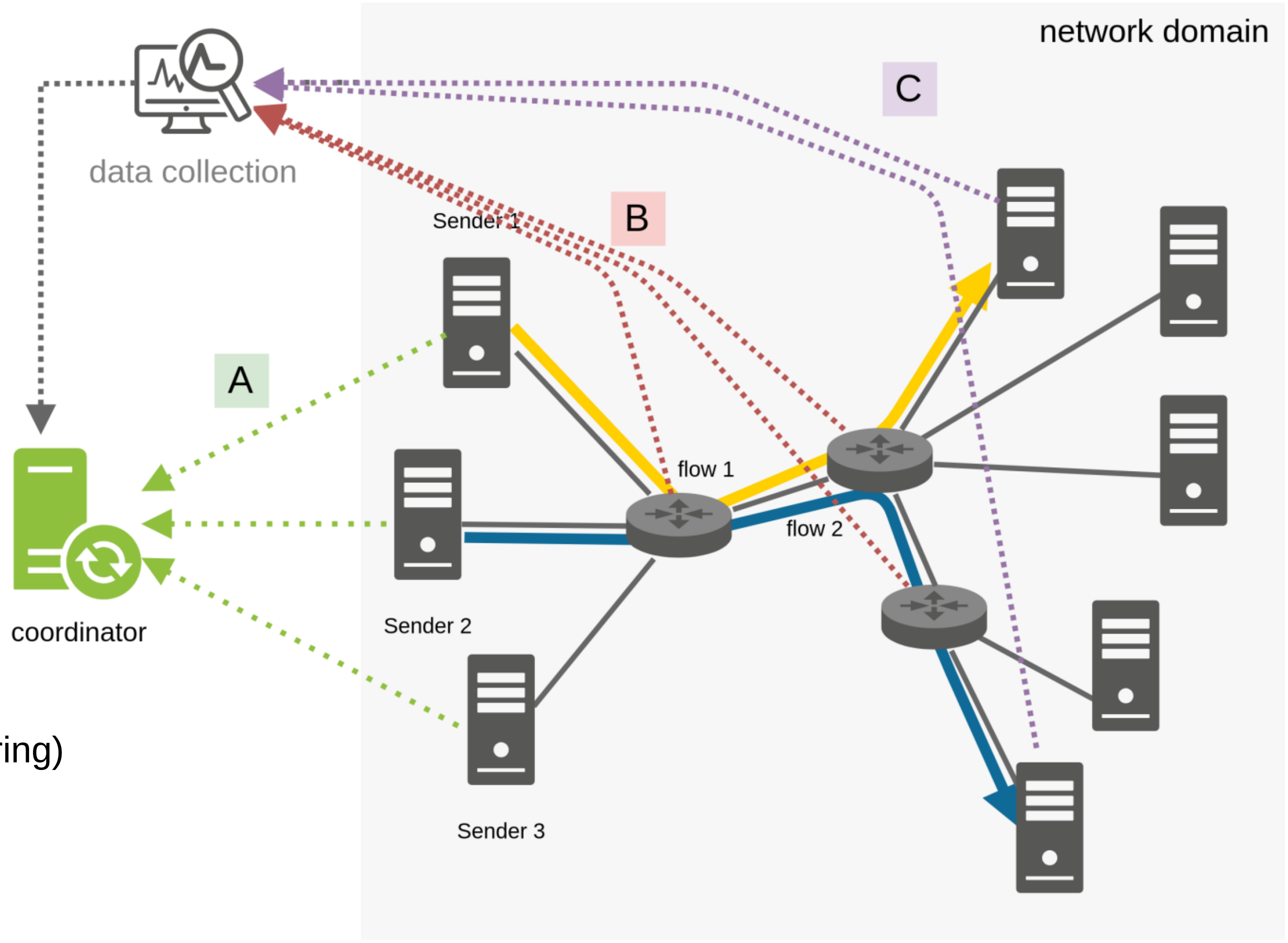
A Senders
- Round trip times
- Loss rates
- ...

Directly from senders

B Intermediate Systems
- Link bandwidths
- Buffer sizes and queue utilizations
- ...

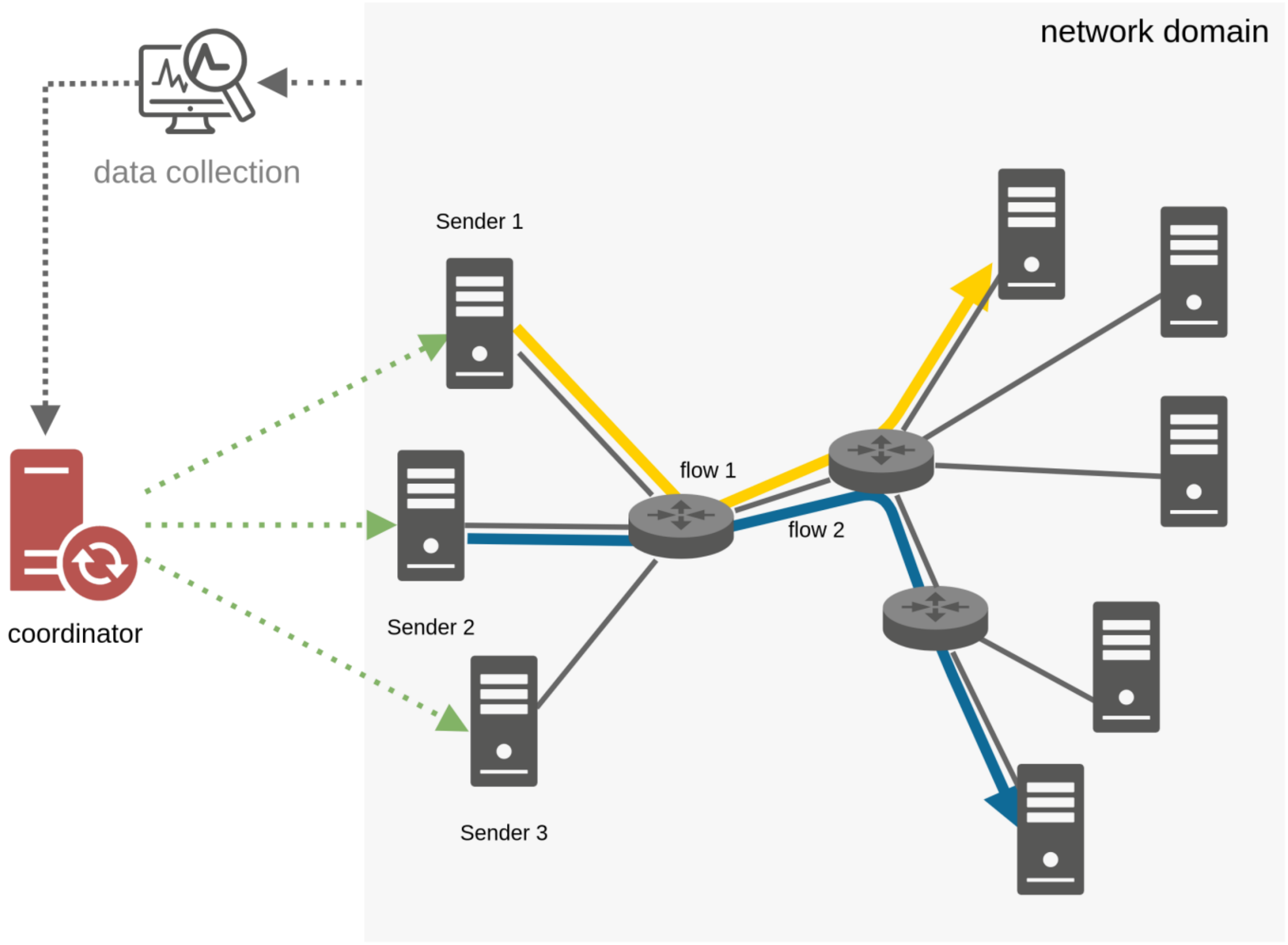
C Receivers
- Delivery rates
- Receive buffer sizes
- ...

Data Collection (external monitoring)



2. Coordination

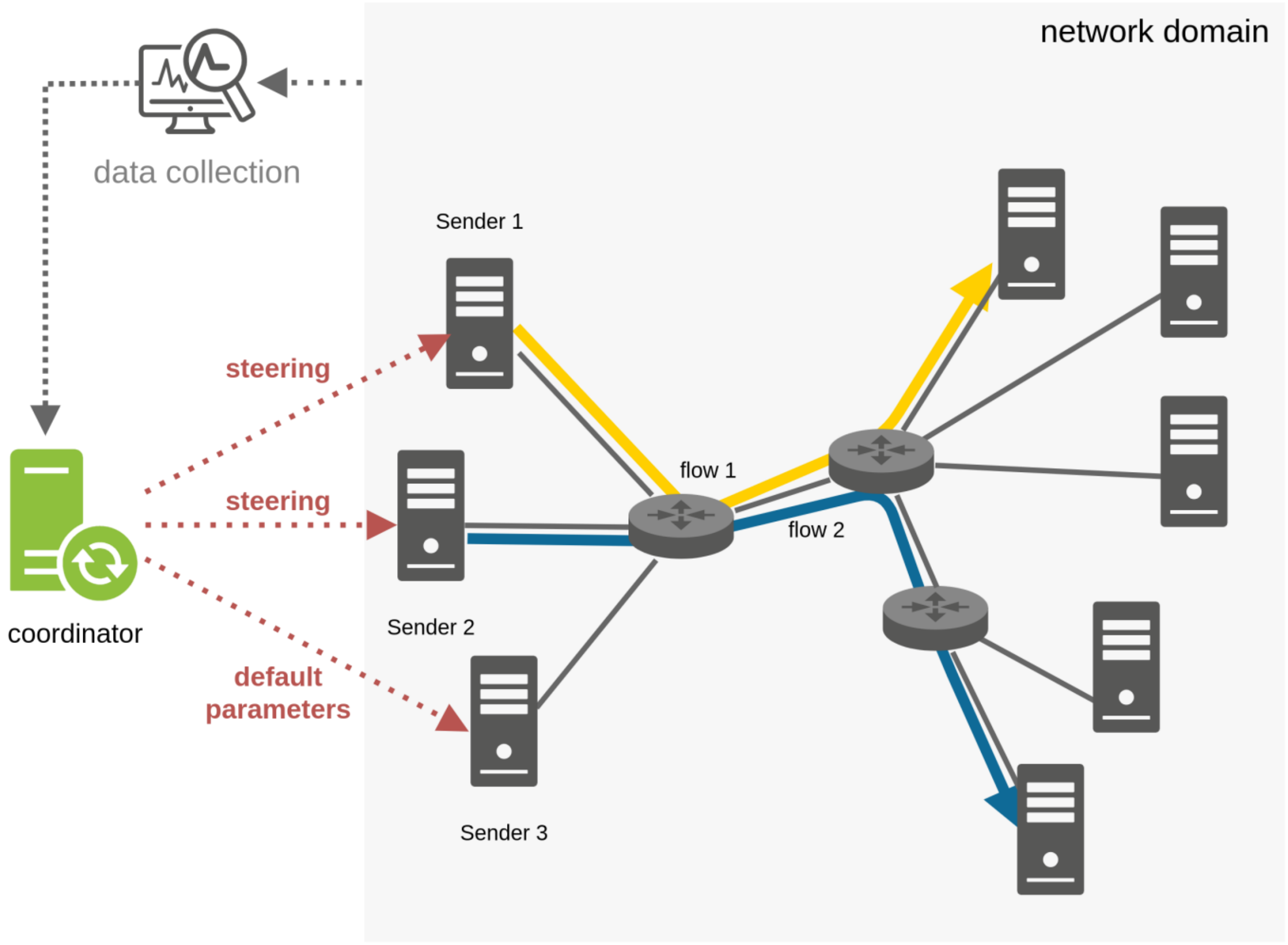
- Merge global and explicit network state
- Compute coordinated control decisions for multiple senders to ensure
  - High Throughput
  - Low Delay
  - Fairness
- Via DRL-Agent
  with suitable reward function



3. Flow Parametrization

Fine-tuned parametrization
of multiple senders and flows

- Coordinated steering (not control) for long-lived flows
  - Flow 1: cWnd = 42 MSS
  - Flow 2: cWnd = 80 MSS
- Providing default parameters (especially relevant for short-lived flows)
  - Sender 3: $cWnd_{initial}$ = 20 MSS

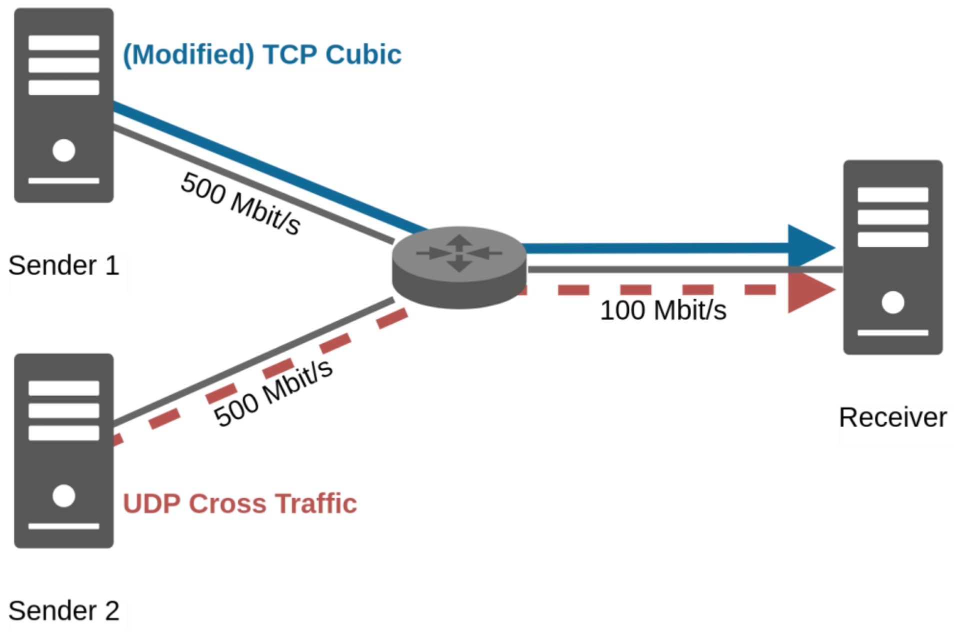


Non-ML Baseline: Conceptional Evaluation C³

- Goal: Demonstrate C³'s advantages

- Two senders:
  - Sender 1: (Modified) TCP Cubic as C³ stand-in
  - Sender 2: Periodic cross traffic via UDP

- Evaluated in cloud environment (bwCloud)



Conceptional Evaluation C³

Karlsruhe Institute of Technology

Better resource utilization
(higher average throughput/goodput)

Cubic	Modified Cubic	Improvement
59.0 Mbit/s	82.3 Mbit/s	39.5 %

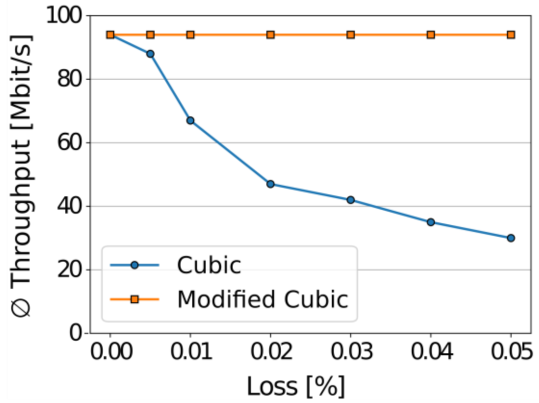
Average Goodput
(Bottleneck link 100 Mbit/s
with 70 Mbit/s of periodic UDP cross traffic)

Better flow completion times (FCT)
for short-lived flows

Cubic	Modified Cubic	Improvement
204 ms	106 ms	51.9 %

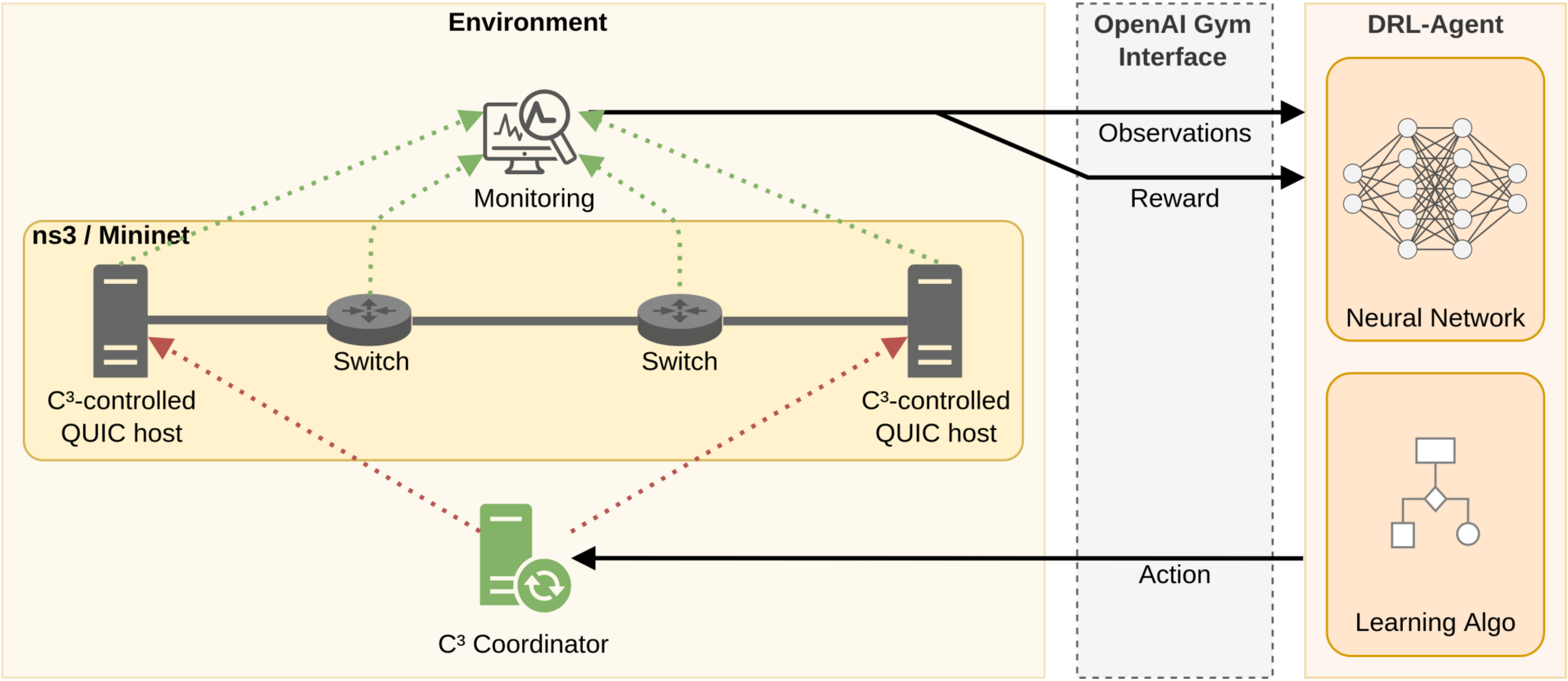
Flow Completion Times
(RTT 50 ms & 50 KiB flows)

Better performance for environments with non-congestion based packet losses



Modified Cubic = C³ concepts mapped to existing TCP Cubic.

Setup for Reinforcement Learning

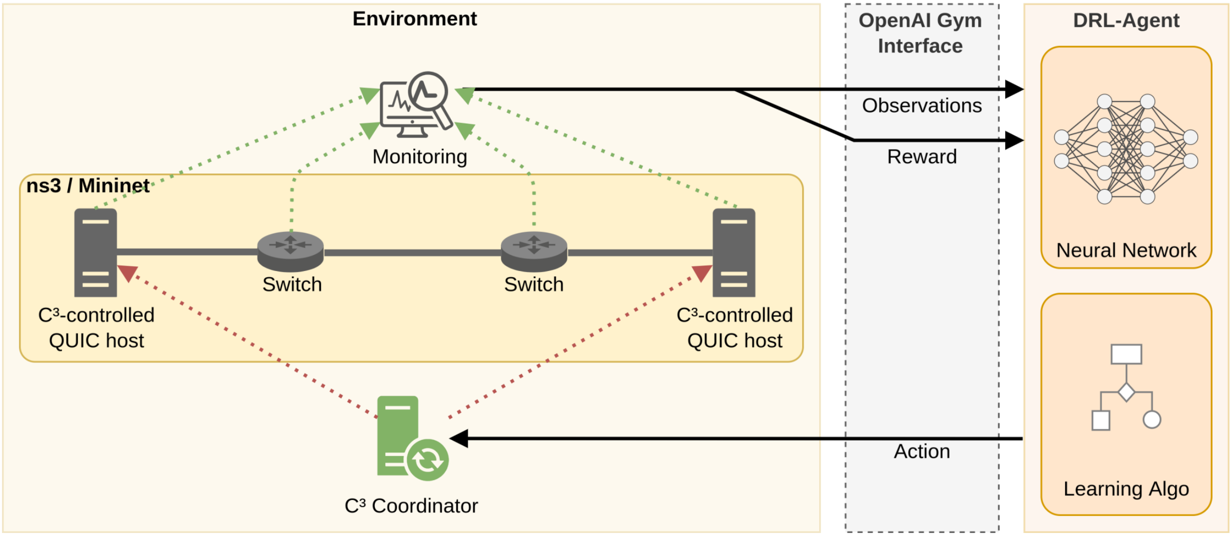


Major Components

- Three components
  - Environment
  - OpenAI Gym Interface
  - Agent

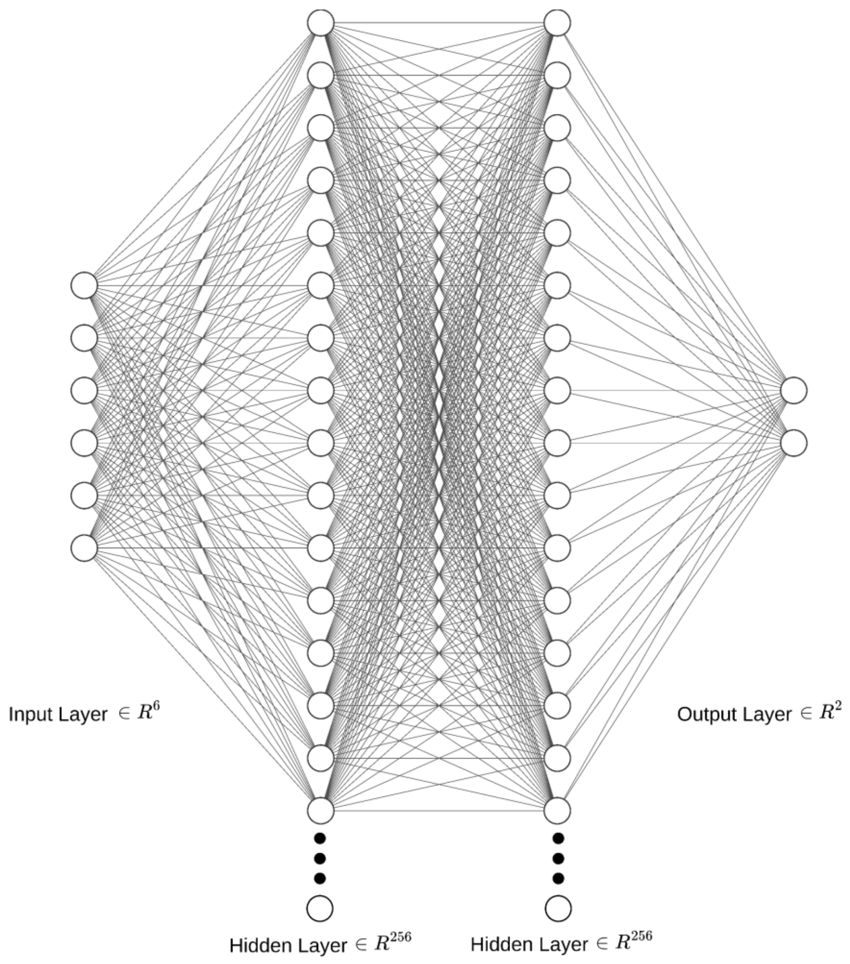
- Agent
  - Based on PyTorch, RLlib, Ray
  - Simple neural network
  - Several learning algorithms considered (PPO, DQN, DDPG, A2C, …)

- Agent and Environment separated by OpenAI Gym interface (ns3-gym [1])



[1] Gawłowicz, Piotr, and Anatolij Zubow. "ns3-gym: Extending openai gym for networking research." arXiv preprint arXiv:1810.03943 (2018).

Neural Network Model



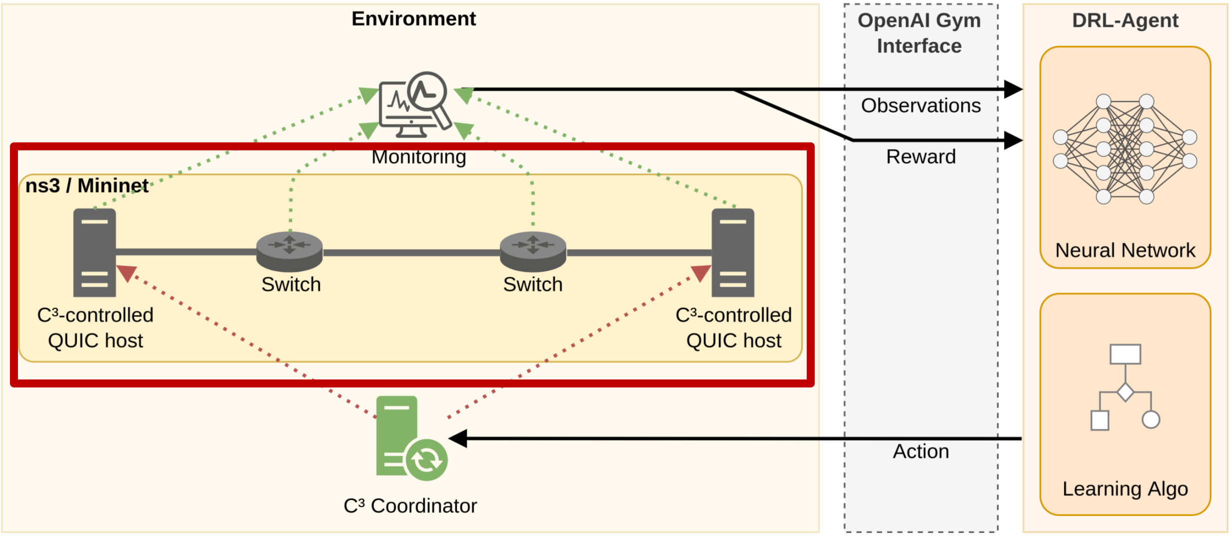
- Simple initial NN-model
  - Fully-connected neural network
- Input layer depends on state space
  - Currently 6 features
- 2 hidden layers
  - 256 neurons each
- Output layer depends on sender systems
  - Relative bottleneck send rate
    (total “budget” per host)

Network Realization

- NS-3 (simulation)
- ~~Mininet (emulation)~~

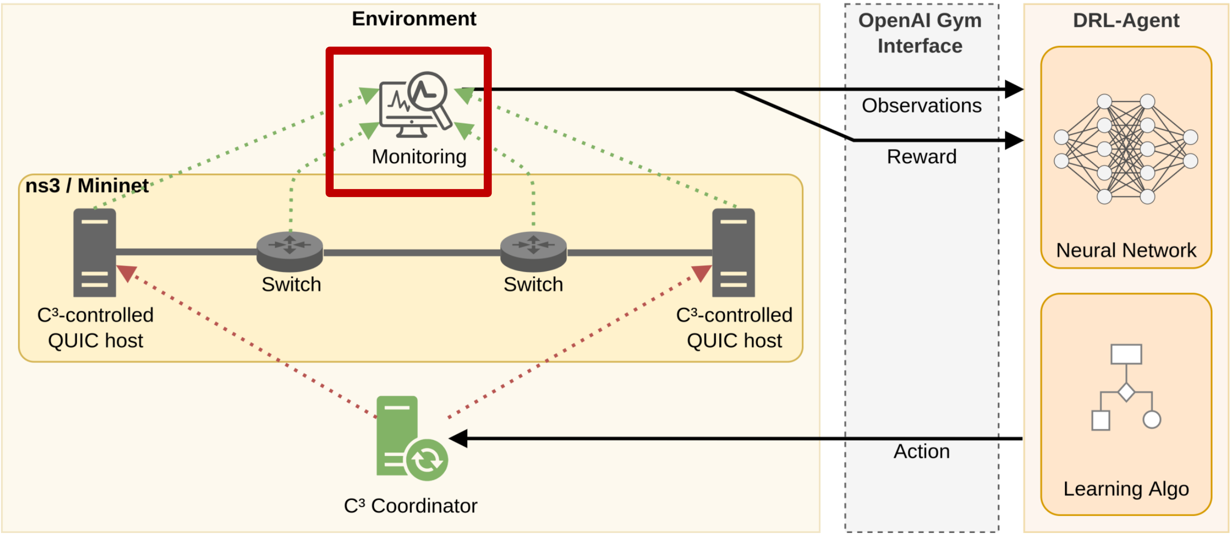
Initially simple scenarios

- Topologies
  - Line
  - Dumbbell
  - …
- Traffic
  - 1 single flow
  - 2 competing flows
  - …



Monitoring

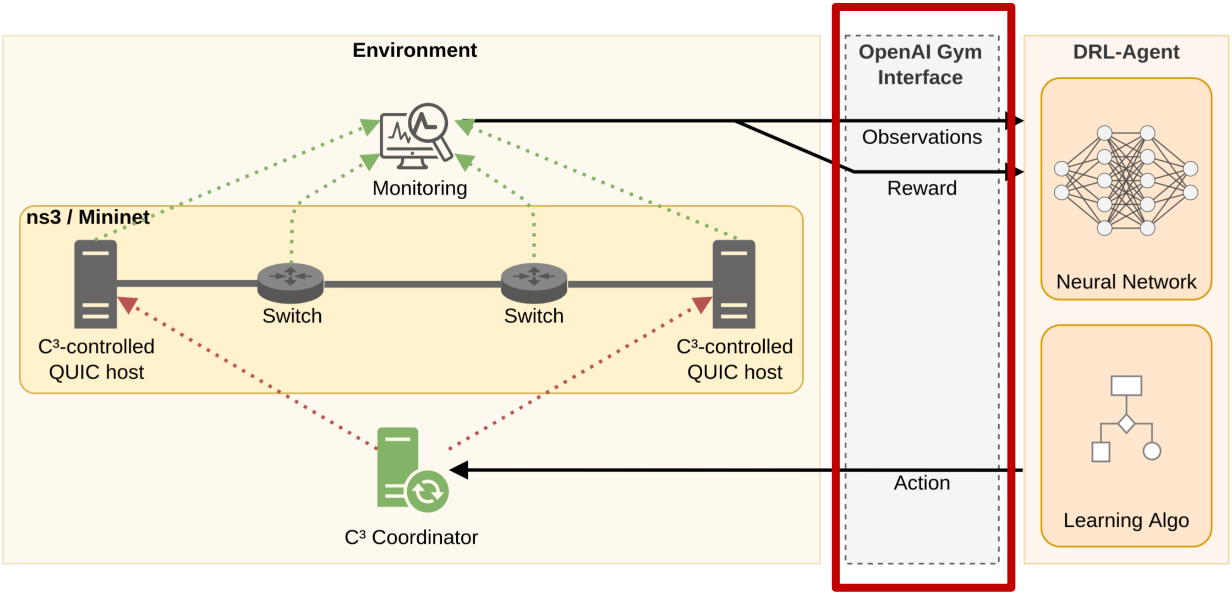
- Periodical collection of information about the network's entities (observed metrics)
  - Throughput
  - Bandwidths
  - Queue lengths
  - Packet drops
  - …
- Communication: out-of-band



Learning Process

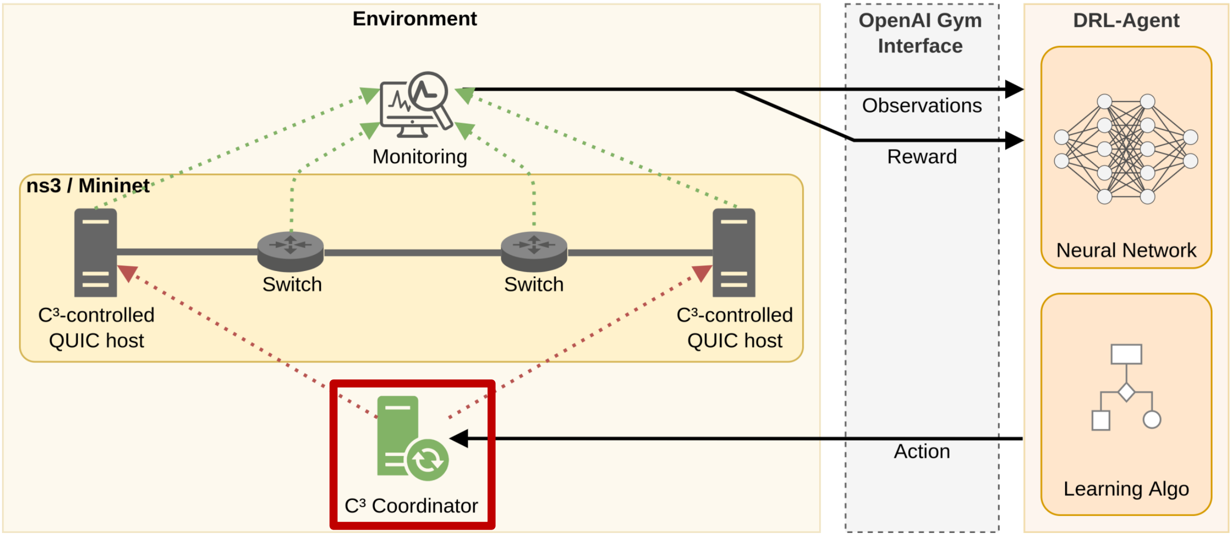
In each time step

- Transfer of observed metrics
- Derivation of current reward
- Computation of next action
  (relative send rate $rate_{rel}$ for each host)



Coordination

- Receives actions from agent
  - Relative send rate $rate_{rel}$
    (relative to bottleneck bandwidth)
- Computes absolute send rate $rate_{abs}$
- Sets $rate_{abs}$ for each sender
  - “Host-budget” distributed between all sender’s flows
  - Communication: out-of-band

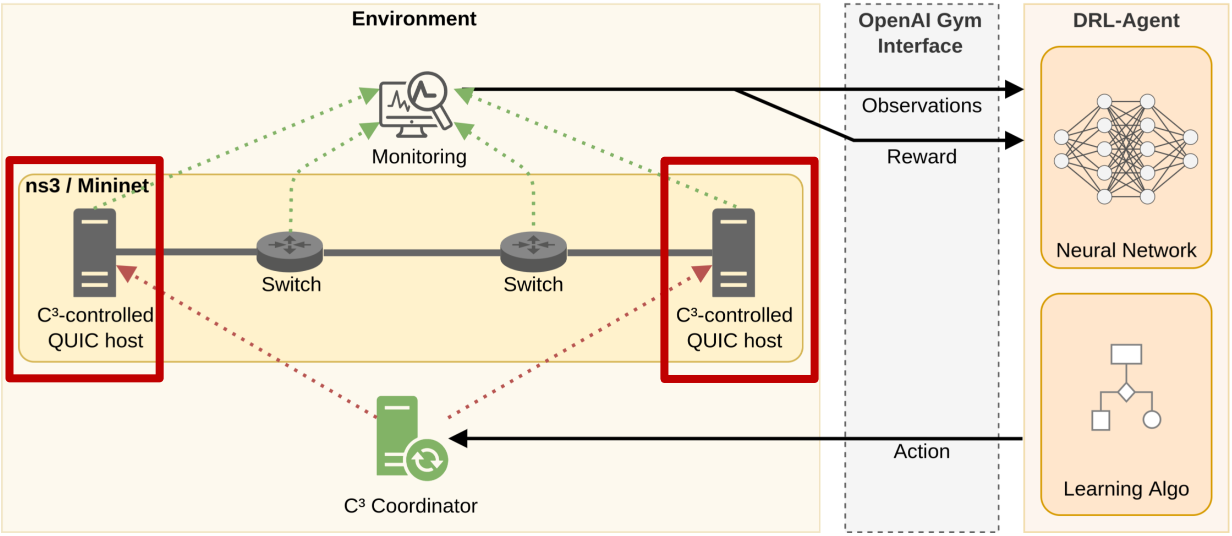


Institute of Telematics
Research Group Prof. Zitterbart

C³-enabled Sender

- Sender receive $rate_{abs}$
- Calculates $rate_{flow} = rate_{abs} / \#flows$
- Calculates $cWnd_{flow} = rate_{flow} * RTT_{min}$

- Adapted QUIC [1] as end-to-end transport protocol
  - cWnd-modifying functions adapted
    - onAck()
    - onDupAck()
    - onLoss()
  - New externally callable functions
    - Set fixed cWnd
    - Set initial cWnd
    - Set bounded cWnd (min/max)



[1] De Biasio, Alvise, et al. "A QUIC Implementation for ns-3." Proceedings of the 2019 Workshop on ns-3. 2019.

Reward Function

Queue length of each interface
(normalized to maximum interface queue length)

→ Penalizes long queues

→ Penalizes host budget for hosts without active flows

$$R \leftarrow \sum_{i\, \in hosts} \boxed{\frac{bw_i}{bw_{i_{max}}}} \boxed{- \frac{q_i}{q_{i_{max}}}} \boxed{- penalty_{drop}(i)} \boxed{- penalty_{sending}} \boxed{- penalty_{unfairness}}$$

Throughput of each sender interface
(normalized to maximum interface bandwidth)

→ Rewards high bandwidth utilization

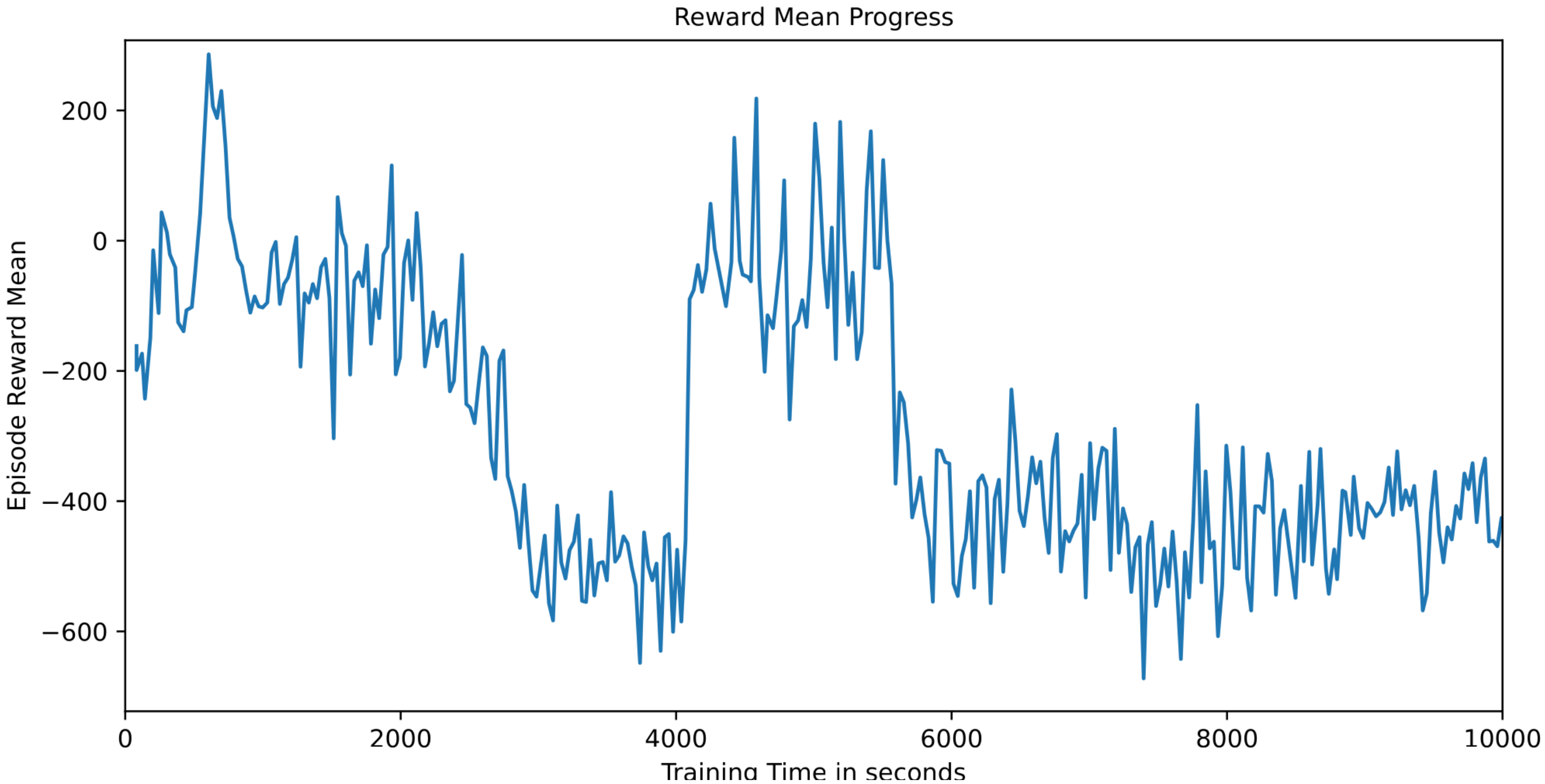
Signal indicating packets dropped at interface

→ Penalizes packet drops

Standard deviation of the send rates

→ Penalizes unfairness

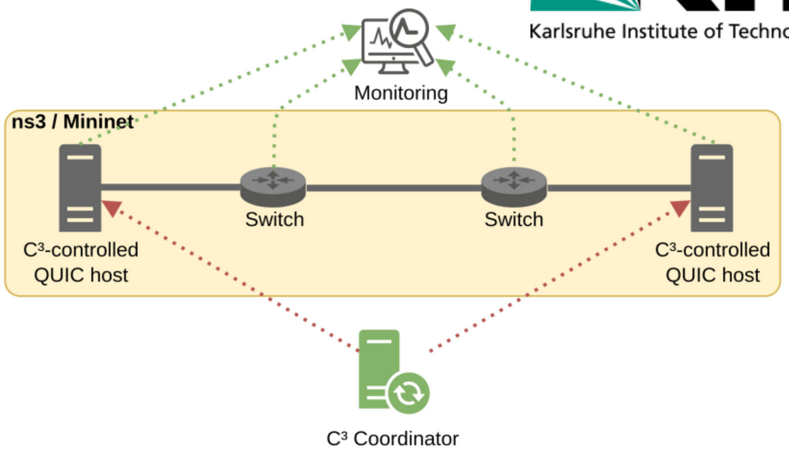
First Attempt: Mininet



➡ No convergence / no upwards trend

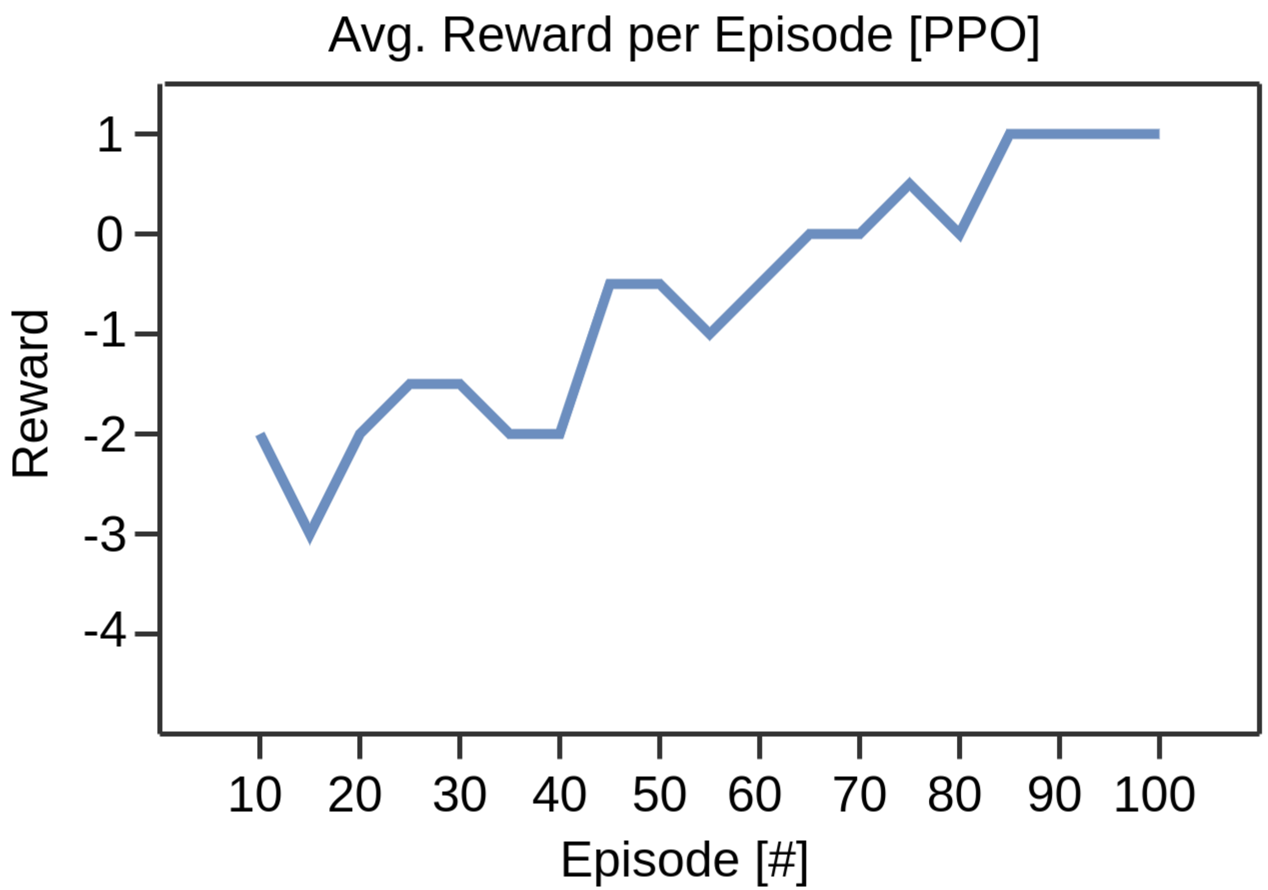
(most likely due to micro-bursts caused by scheduling artefacts in Mininet)

Preliminary Evaluation Results



- One agent gets trained in three scenarios (ns3 simulation environment)
- Scenario 1: Single flow
  - Agent learns to set send rate (cwnd) = bottleneck bandwidth
  - Keep CWND steady
- Scenario 2: Two competing flows (parallel start)
  - Agent learns to set send rate (cwnd) = bottleneck bandwidth / 2
  - Keep CWND steady
- Scenario 3: Two competing flows (staggered start)
  - Agent learns to set send rate (cwnd) = bottleneck bandwidth
  - When second sender starts: set send rate (cwnd) = bottleneck bandwidth / 2
  - Keep CWND steady

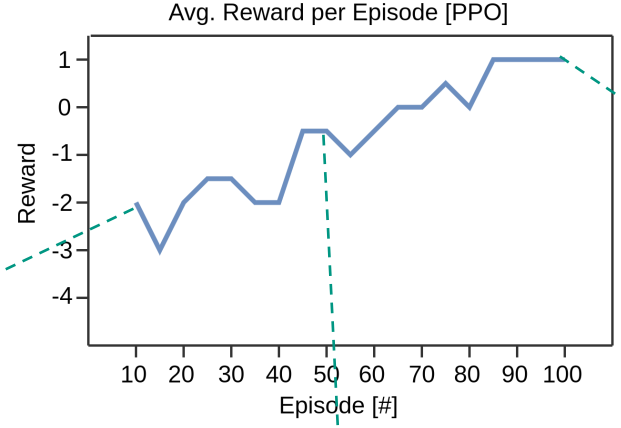
Preliminary Results of Scenario 1 – NS-3



→ Reward converges

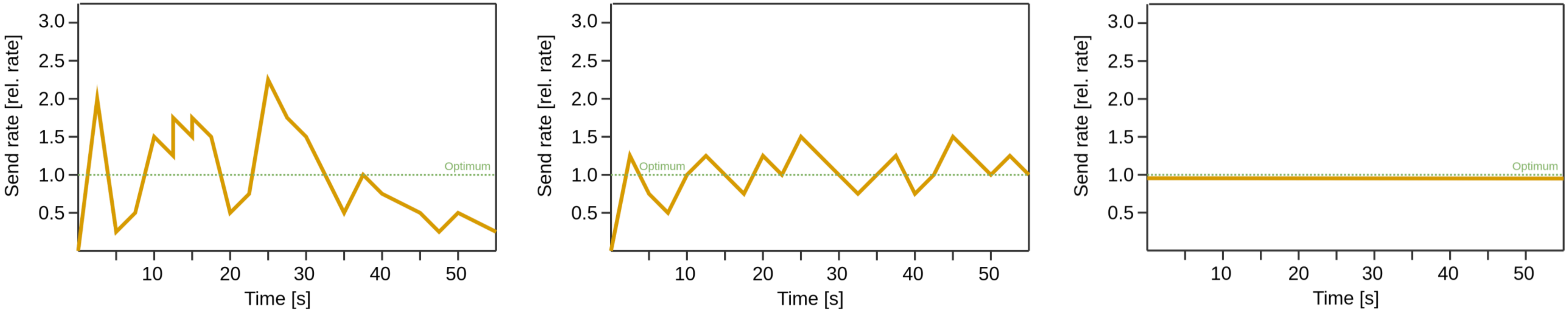
PPO: Proximal Policy Optimzation

Learning Progress of Send Rate Behavior



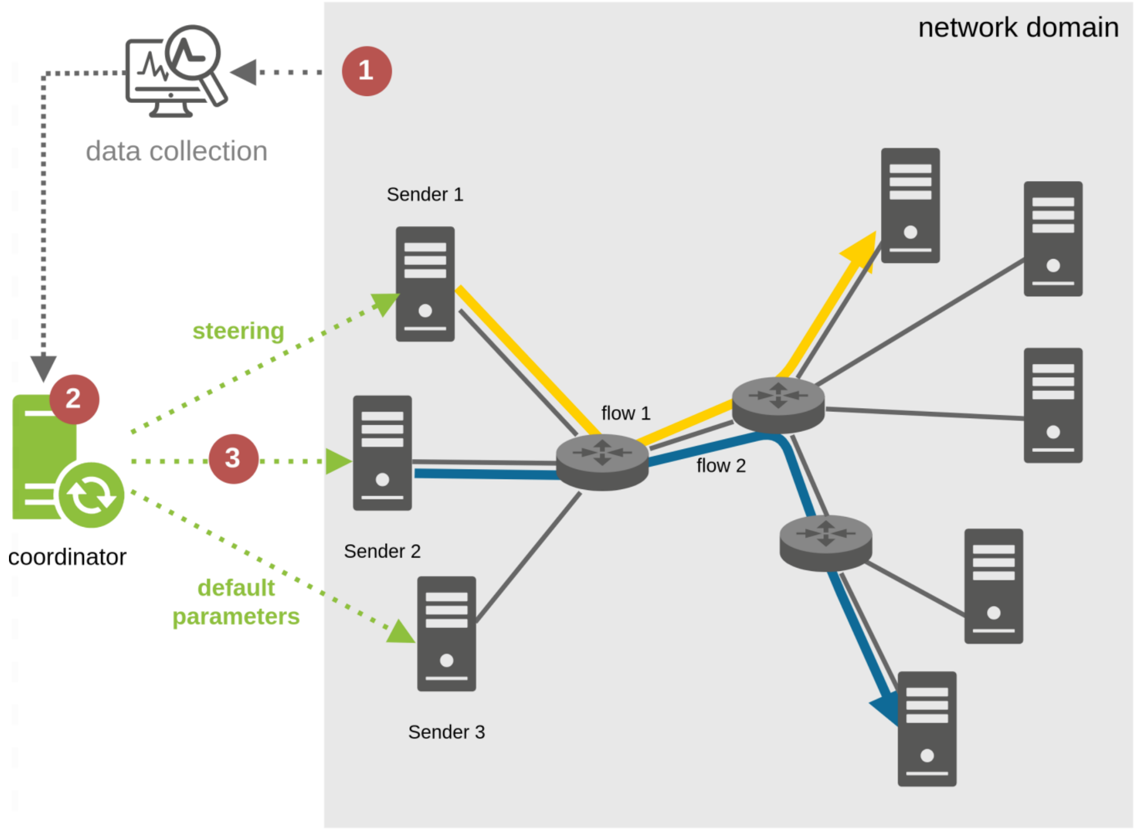
→ Agent successfully learns

in Episode **10** | in Episode **50** | in Episode **100**



Outlook

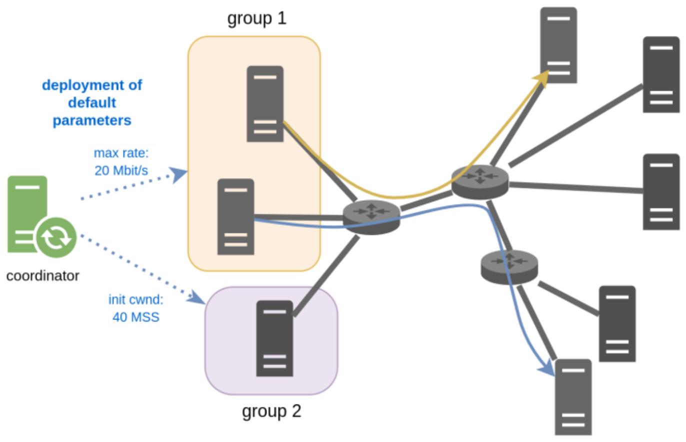
- More sophisticated Neural Network Models
  - Best practices: Dropout and/or normalization layers
  - Handling of delayed actions:
    - RNN , LSTM, GRN, …
    - History of actions
  - Explicitly preserve knowledge about topology: Representation as a (T)GNN
- Tweaking
  - Hyperparameters
  - Composition of reward functions components
- Scalability
  - Minimize amount of communication
  - Split domain in sub-domains -> Multi-Agent RL?
- Influences of in-band communication
  - Delayed monitoring data
  - Competition between data and control traffic



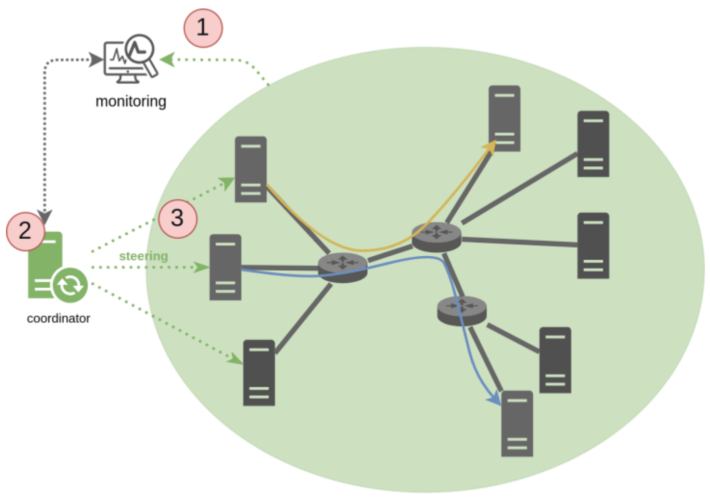
RNN: Recurrent Neural Network
LSTM: Long Short-Term Memory
GRN: Gated Recurrent Network
(T)GNN: (Temporal) Graph Neural Network

Spectrum of Use Cases

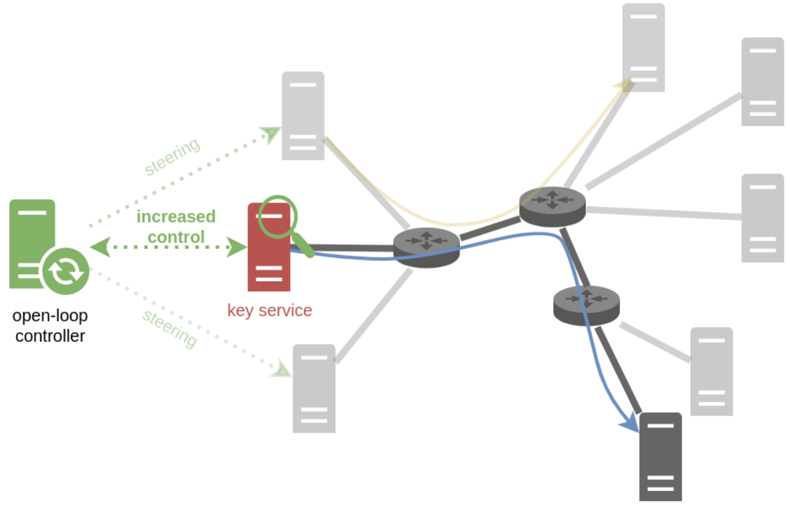
Default Parameters
Broad defaults for many (similar) services



Coordinated Congestion Control
Steering of multiple senders based on combined knowledge



Real-time Online Control
Increased control (beyond steering) of selected key services



Update Frequency & Granularity

Related Work: Non-ML-based Centralized Control

- Congestion Manager [1]
  - "Coordination" of multiple flows within one system – not multiple senders
- OTCP [2]
  - Distribution of default parameters – no steering
- Bandwidth Enforcer [3]
  - Focus on limiting max. throughput

[1] Balakrishnan, Hari, and Srinivasan Seshan. The congestion manager. RFC3124. 2001.
[2] Jouet, Simon, Colin Perkins, and Dimitrios Pezaros. "OTCP: SDN-managed congestion control for data center networks." NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2016.
[3] Kumar, Alok, et al. "BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing." Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. 2015.