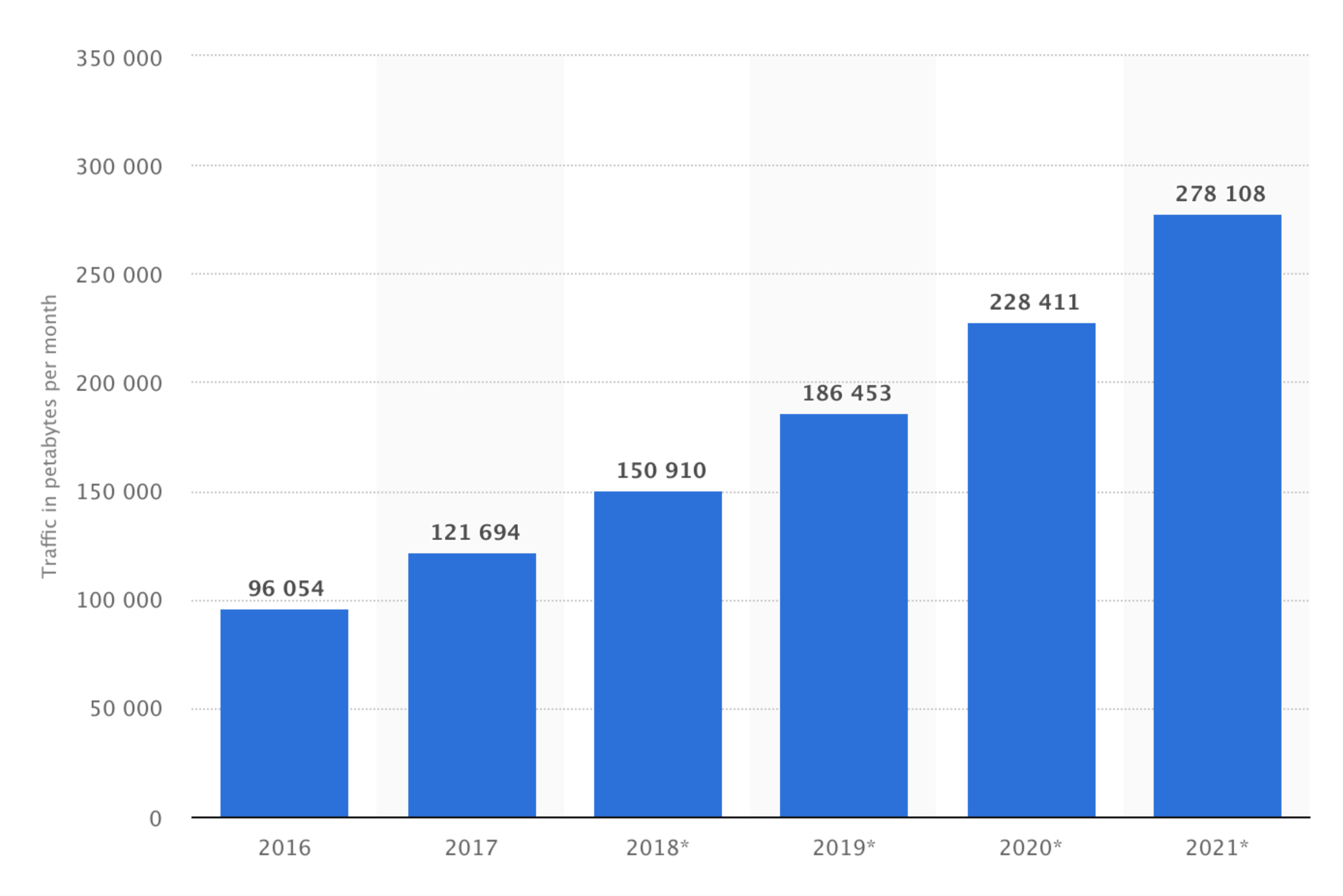
Towards Faster Internet with Reinforcement Learning

Viswanath Sivakumar

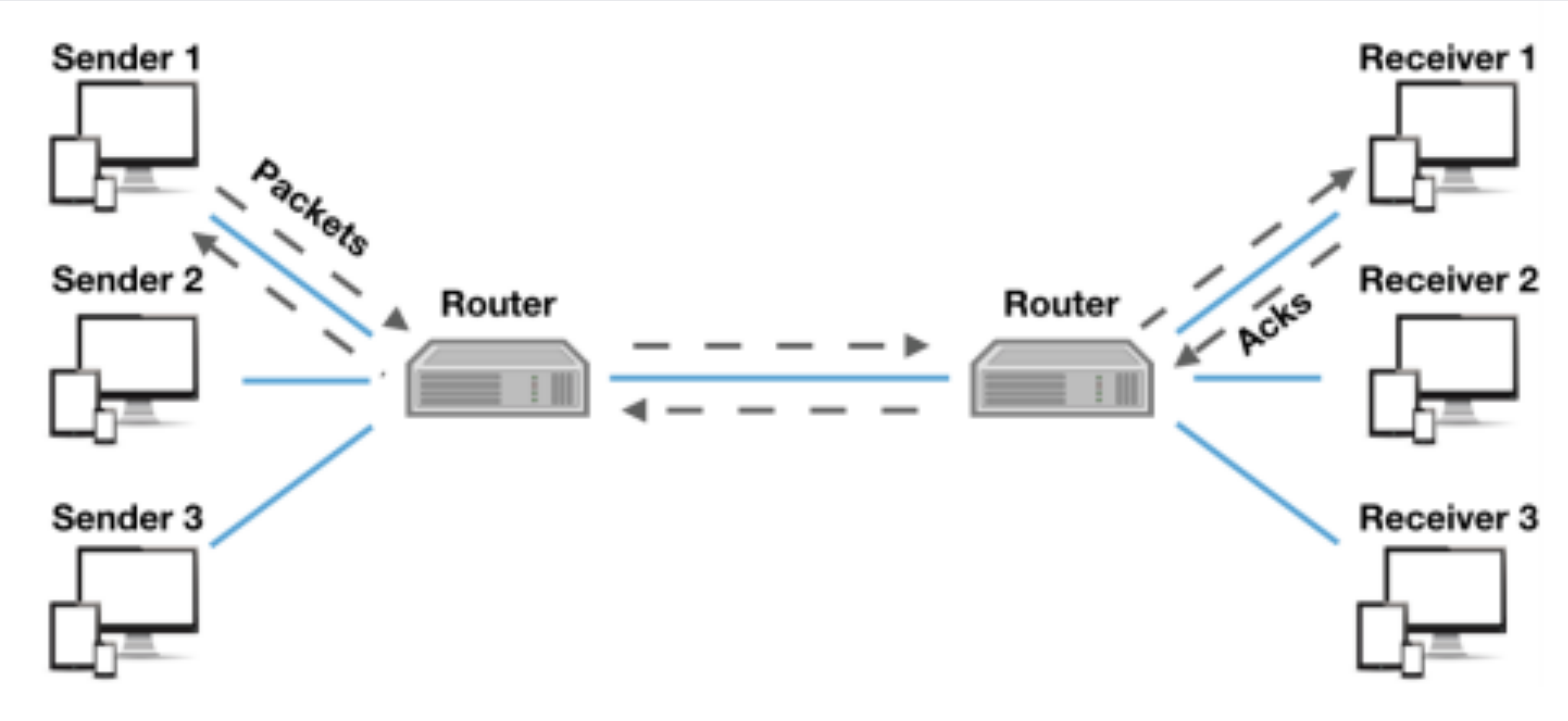
Facebook AI Research (FAIR)

Global Internet Traffic (PB/month)



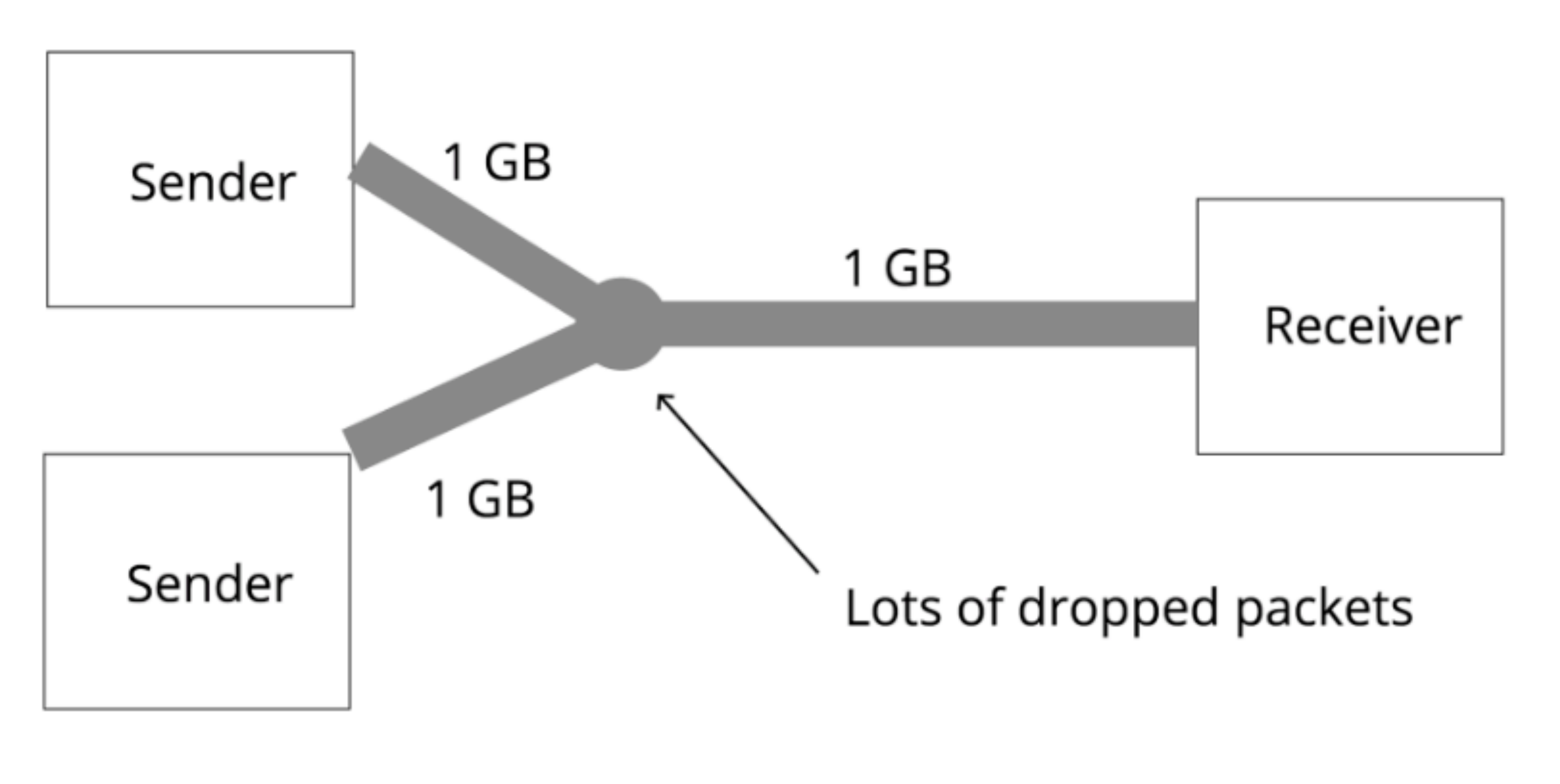
https://www.statista.com/statistics/499431/global-ip-data-traffic-forecast/

Network Congestion



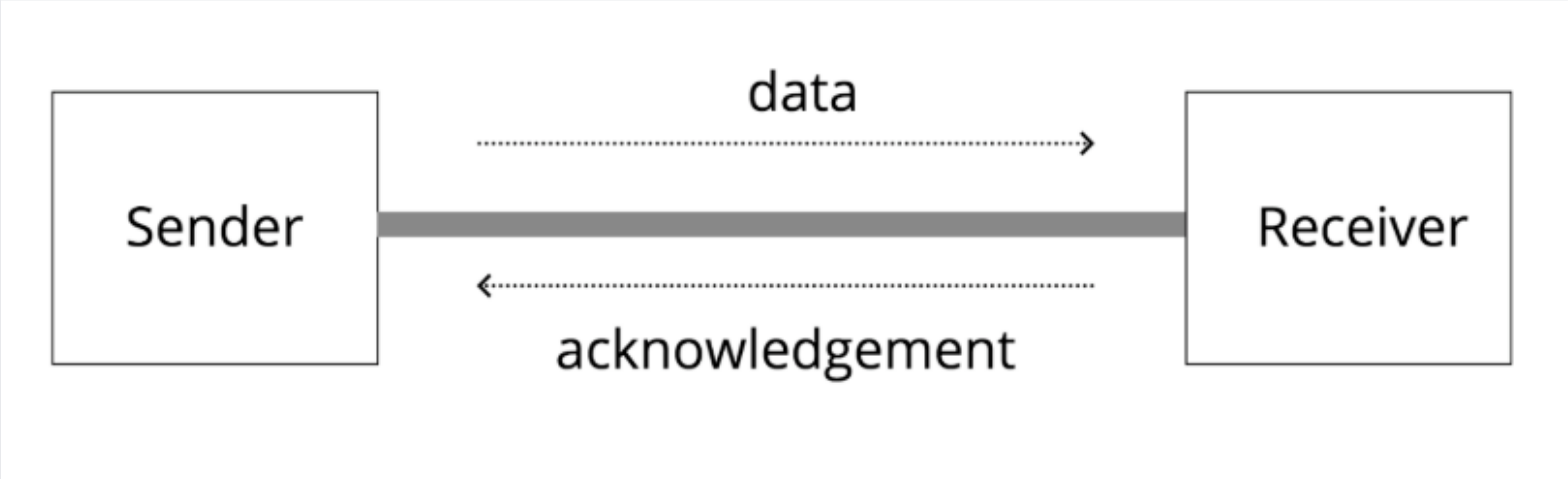
http://www.squidarth.com/rc/programming/networking/2018/07/18/intro-congestion.html

Computer Networks 101



http://www.squidarth.com/rc/programming/networking/2018/07/18/intro-congestion.html

Computer Networks 101



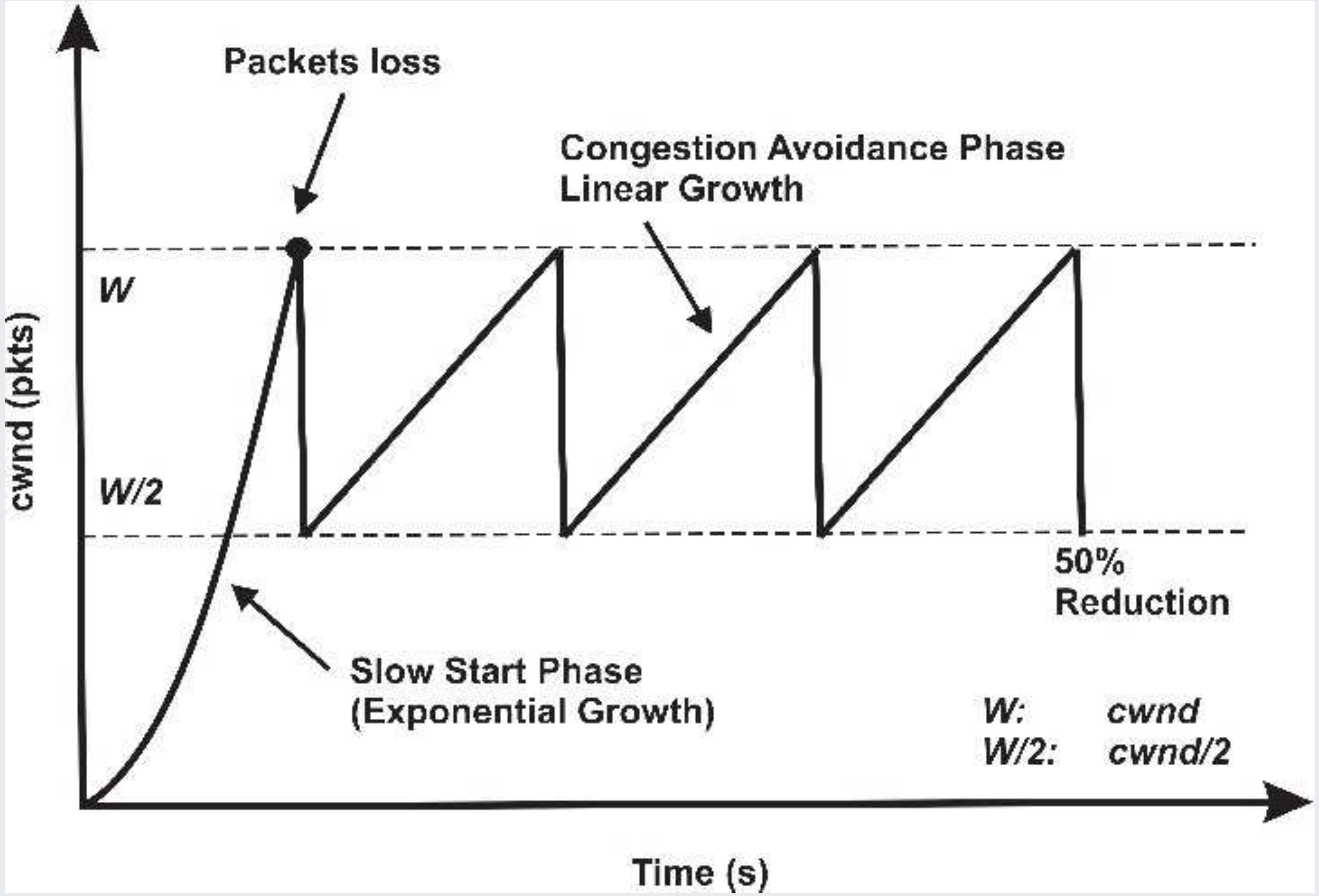
ACK tells you:

- The largest packet number received
- Measured round-trip time (RTT)
- …

http://www.squidarth.com/rc/programming/networking/2018/07/18/intro-congestion.html

Traditional Congestion Control

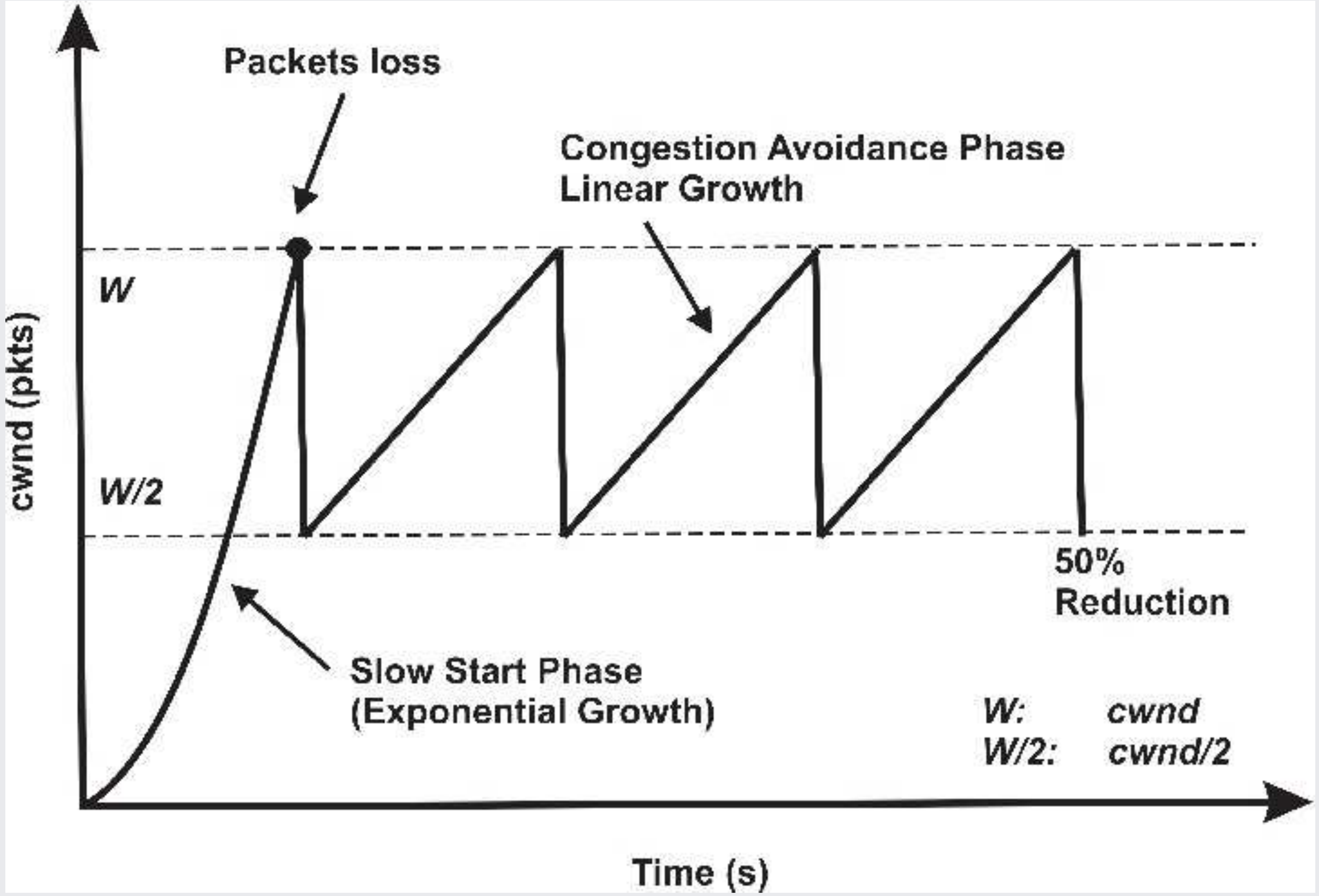
- Hand-engineered
- Decentralized
- Reactive, not predictive
- Often too conservative



Experimental evaluation of TCP congestion control mechanisms in short and long distance networks, Mohamad et al.

Traditional Congestion Control

- Hand-engineered
- Decentralized
- Reactive, not predictive
- Often too conservative



Experimental evaluation of TCP congestion control mechanisms in short and long distance networks, Mohamad et al.

30+ years of active research!

mvfst-rl

An asynchronous RL platform for congestion control in QUIC transport protocol

mvfst-rl 1/3

QUIC Networking Stack:

- Handles Facebook production traffic
- UDP-based
- User-space (as opposed to Kernel) congestion control

github.com/facebookincubator/mvfst

mvfst-rl 1/3

QUIC Networking Stack:

- Handles Facebook production traffic
- UDP-based
- User-space (as opposed to Kernel) congestion control

github.com/facebookincubator/mvfst

Rapid Experimentation

mvfst-rl 2/3

Pantheon Network Emulator:

- Emulation of 18 real-world network scenarios
- Bayesian Optimization for calibration

1. Calibrated emulator (Nepal to AWS India)
2. Calibrated emulator (Mexico cellular to AWS California)
3. Calibrated emulator (AWS Brazil to Colombia cellular)
4. Calibrated emulator (India to AWS India)
5. Calibrated emulator (AWS Korea to China)
6. Calibrated emulator (AWS California to Mexico)
7. Token-bucket based policer (bandwidth 12mbps, RTT 20ms)
8. Token-bucket based policer (bandwidth 60mbps, RTT 20ms)
9. Token-bucket based policer (bandwidth 108mbps, RTT 20ms)
10. Token-bucket based policer (bandwidth 12mbps, RTT 100ms)
11. Token-bucket based policer (bandwidth 60mbps, RTT 100ms)
12. Token-bucket based policer (bandwidth 108mbps, RTT 100ms)
13. Severe ACK aggregation (1 ACK every 100ms)
14. Severe ACK aggregation (10 ACKs every 200ms)
15. Bottleneck buffer = BDP/10
16. Bottleneck buffer = BDP/3
17. Bottleneck buffer = BDP/2
18. Bottleneck buffer = BDP

pantheon.stanford.edu

mvfst-rl 2/3

Pantheon Network Emulator:

- Emulation of 18 real-world network scenarios
- Bayesian Optimization for calibration

1. Calibrated emulator (Nepal to AWS India)
2. Calibrated emulator (Mexico cellular to AWS California)
3. Calibrated emulator (AWS Brazil to Colombia cellular)
4. Calibrated emulator (India to AWS India)
5. Calibrated emulator (AWS Korea to China)
6. Calibrated emulator (AWS California to Mexico)
7. Token-bucket based policer (bandwidth 12mbps, RTT 20ms)
8. Token-bucket based policer (bandwidth 60mbps, RTT 20ms)
9. Token-bucket based policer (bandwidth 108mbps, RTT 20ms)
10. Token-bucket based policer (bandwidth 12mbps, RTT 100ms)
11. Token-bucket based policer (bandwidth 60mbps, RTT 100ms)
12. Token-bucket based policer (bandwidth 108mbps, RTT 100ms)
13. Severe ACK aggregation (1 ACK every 100ms)
14. Severe ACK aggregation (10 ACKs every 200ms)
15. Bottleneck buffer = BDP/10
16. Bottleneck buffer = BDP/3
17. Bottleneck buffer = BDP/2
18. Bottleneck buffer = BDP

pantheon.stanford.edu

Real World Training Arena

mvfst-rl 3/3

Torchbeast PyTorch RL:

- Asynchronous RL training
- Distributed actors
- PyTorch frontend
- Highly efficient C++ implementation

```
|-----------------|     |-----------------|                  |-----------------|
|     ACTOR 1     |     |     ACTOR 2     |                  |     ACTOR n     |
|-------|         |     |-------|         |                  |-------|         |
|       |  .......|     |       |  .......|     .   .   .    |       |  .......|
|  Env  |<-.Model.|     |  Env  |<-.Model.|                  |  Env  |<-.Model.|
|       |->.......|     |       |->.......|                  |       |->.......|
|-----------------|     |-----------------|                  |-----------------|
   ^     I                 ^     I                              ^     I
   |     I                 |     I                              |     I Actors
   |     I rollout         |     I rollout               weights|     I send
   |     I                 |     I                     /--------/     I rollouts
   |     I          weights|     I                     |              I (frames,
   |     I                 |     I                     |              I  actions
   |     I                 |     v                     |              I  etc)
   |     L=======>|--------------------------------------|<===========J
   |              |.........      LEARNER                |
   \--------------|..Model.. Consumes rollouts, updates  |
     Learner      |.........       model weights         |
      sends       |--------------------------------------|
     weights
```

github.com/facebookresearch/torchbeast

mvfst-rl 3/3

Torchbeast PyTorch RL:

- Asynchronous RL training
- Distributed actors
- PyTorch frontend
- Highly efficient C++ implementation

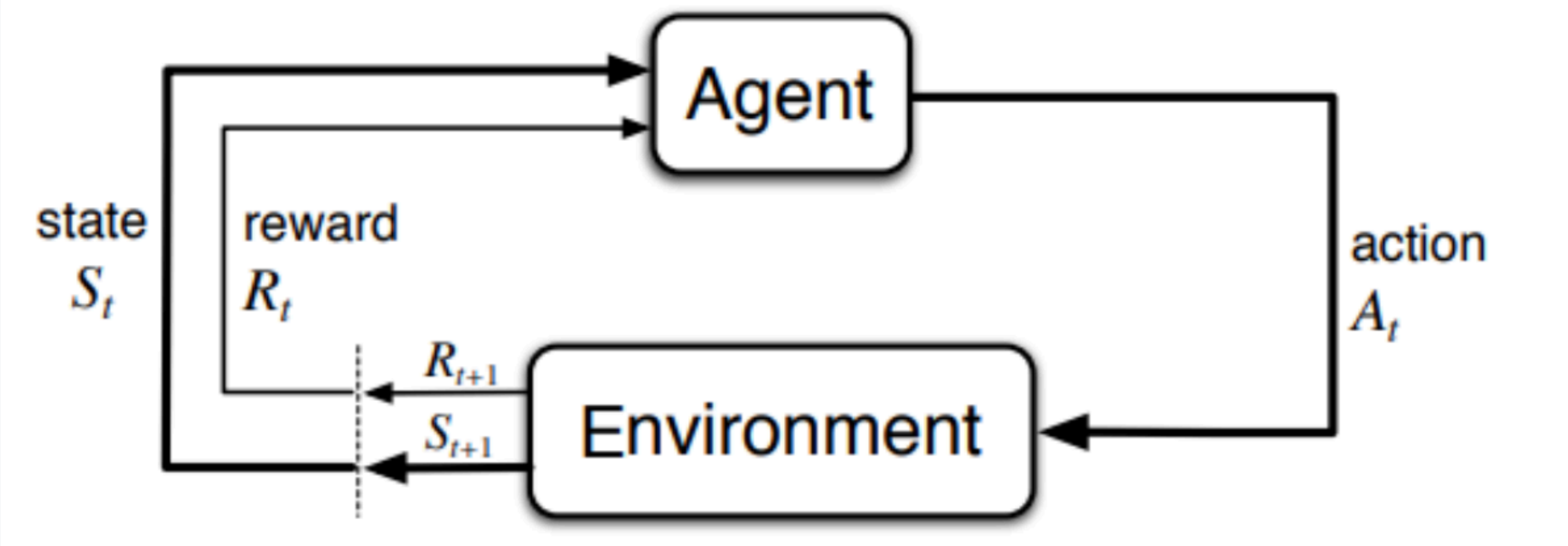
```
|-----------------|     |-----------------|                  |-----------------|
|     ACTOR 1     |     |     ACTOR 2     |                  |     ACTOR n     |
|-------|         |     |-------|         |                  |-------|         |
|       |  .......|     |       |  .......|     .   .   .    |       |  .......|
|  Env  |<-.Model.|     |  Env  |<-.Model.|                  |  Env  |<-.Model.|
|       |->.......|     |       |->.......|                  |       |->.......|
|-----------------|     |-----------------|                  |-----------------|
   ^     I                 ^     I                              ^     I
   |     I                 |     I                              |     I Actors
   |     I rollout         |     I rollout               weights|     I send
   |     I                 |     I                     /--------/     I rollouts
   |     I          weights|     I                     |              I (frames,
   |     I                 |     I                     |              I  actions
   |     I                 |     v                     |              I  etc)
   |     L=======>|--------------------------------------|<==========J
   |              |.........      LEARNER                 |
   \--------------|..Model.. Consumes rollouts, updates   |
     Learner      |.........       model weights          |
      sends       |--------------------------------------|
     weights
```

github.com/facebookresearch/torchbeast

Fast Training

RL Basics

<S, A, R, S'>

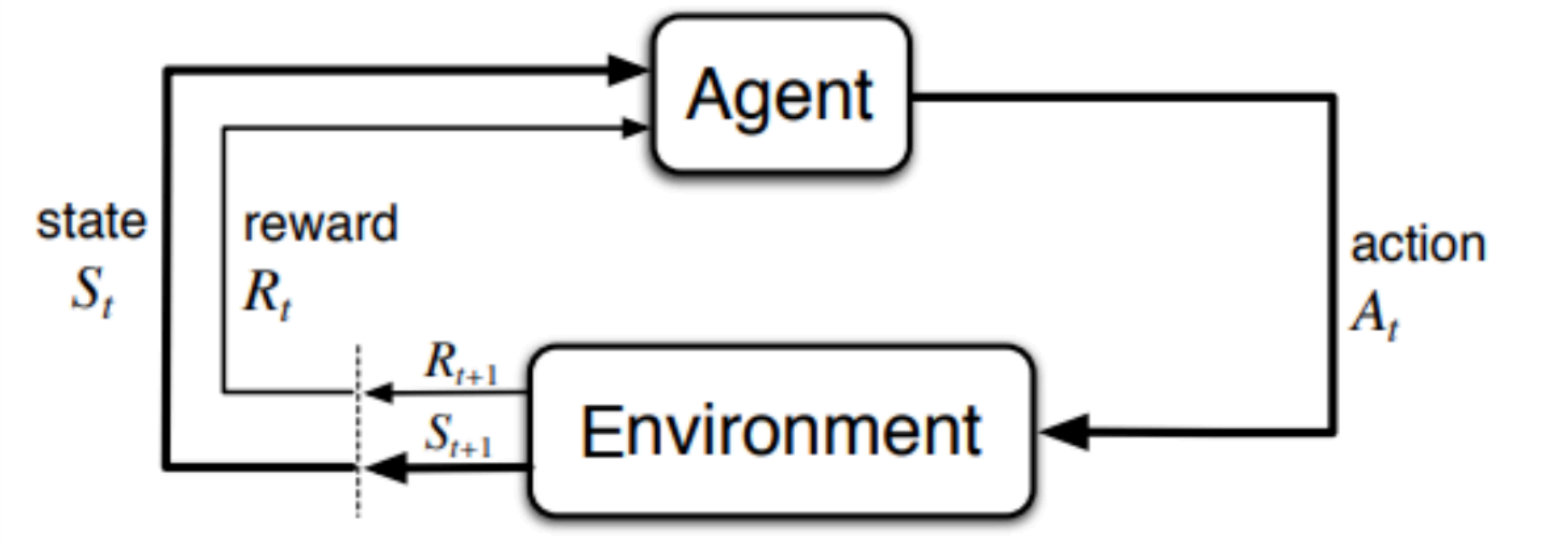


Markov Decision Process (MDP)

- Next state S' depends only on current state S and action A

RL Basics

<S, A, R, S'>



Markov Decision Process (MDP)

- Next state S' depends only on current state S and action A

Policy Gradient Methods:

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

Congestion Control as RL

States:

Network conditions:

- Round-trip time
- Bytes sent
- Packets dropped
- Network delay
- ...

Heterogeneous state space

```
struct TransportInfo {
  std::chrono::microseconds srtt;
  std::chrono::microseconds rttvar;
  std::chrono::microseconds lrtt;
  std::chrono::microseconds mrtt;
  uint64_t writableBytes;
  uint64_t congestionWindow;
  uint64_t pacingBurstSize;
  std::chrono::microseconds pacingInterval;
  uint32_t packetsRetransmitted;
  uint32_t timeoutBasedLoss;
  std::chrono::microseconds pto;
  uint64_t bytesSent;
  uint64_t bytesAcked;
  uint64_t bytesRecvd;
  uint64_t totalBytesRetransmitted;
  uint32_t ptoCount;
  uint32_t totalPTOCount;
  PacketNum largestPacketAckedByPeer;
  PacketNum largestPacketSent;
};
```

Congestion Control as RL

Action:

- Modify congestion window (cwnd)
- Max unacknowledged outstanding bytes

Congestion Control as RL

Action:

- Modify congestion window (cwnd)
- Max unacknowledged outstanding bytes

$$A = \{cwnd, cwnd/2, cwnd - 10, cwnd + 10, cwnd \times 2\}$$

Congestion Control as RL

Action:

- Modify congestion window (cwnd)
- Max unacknowledged outstanding bytes

$$A = \{cwnd, cwnd/2, cwnd - 10, cwnd + 10, cwnd \times 2\}$$

$$cwnd_{t+1} = \mathrm{clip}(\mathrm{update}(cwnd_t, a_t, A), 2, 2000)$$

Congestion Control as RL

Reward:

- Maximize throughput
- Minimize delay

Congestion Control as RL

Reward:

- Maximize throughput
- Minimize delay

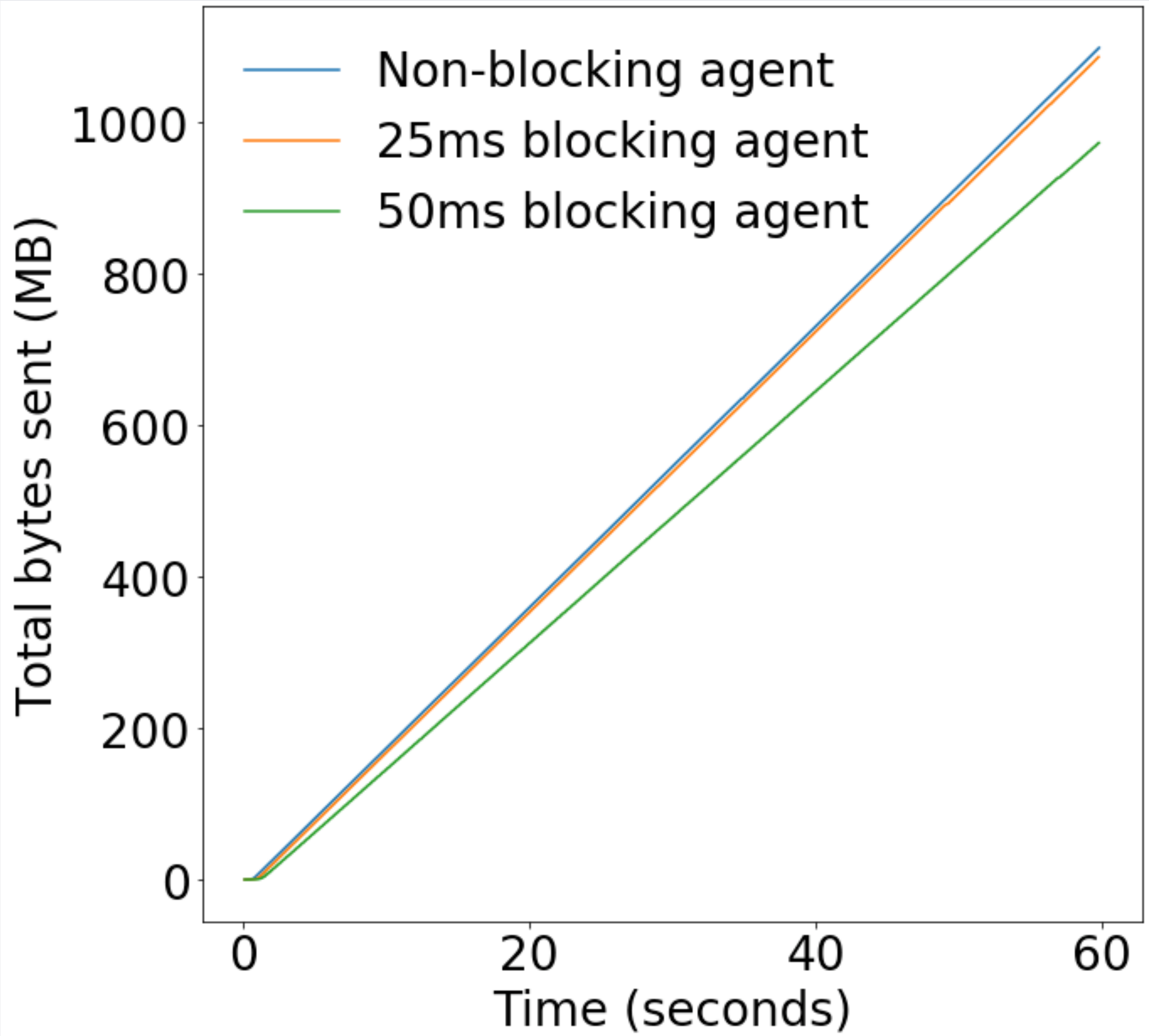
```
reward = throughput - alpha * delay
```

Traditional RL Environments

```
env = ...
state = env.reset()
for _ in range(1000):
  action = model(state)
  state, reward, done = env.step(action)
  if done:
    state = env.reset()
```

Traditional RL Environments

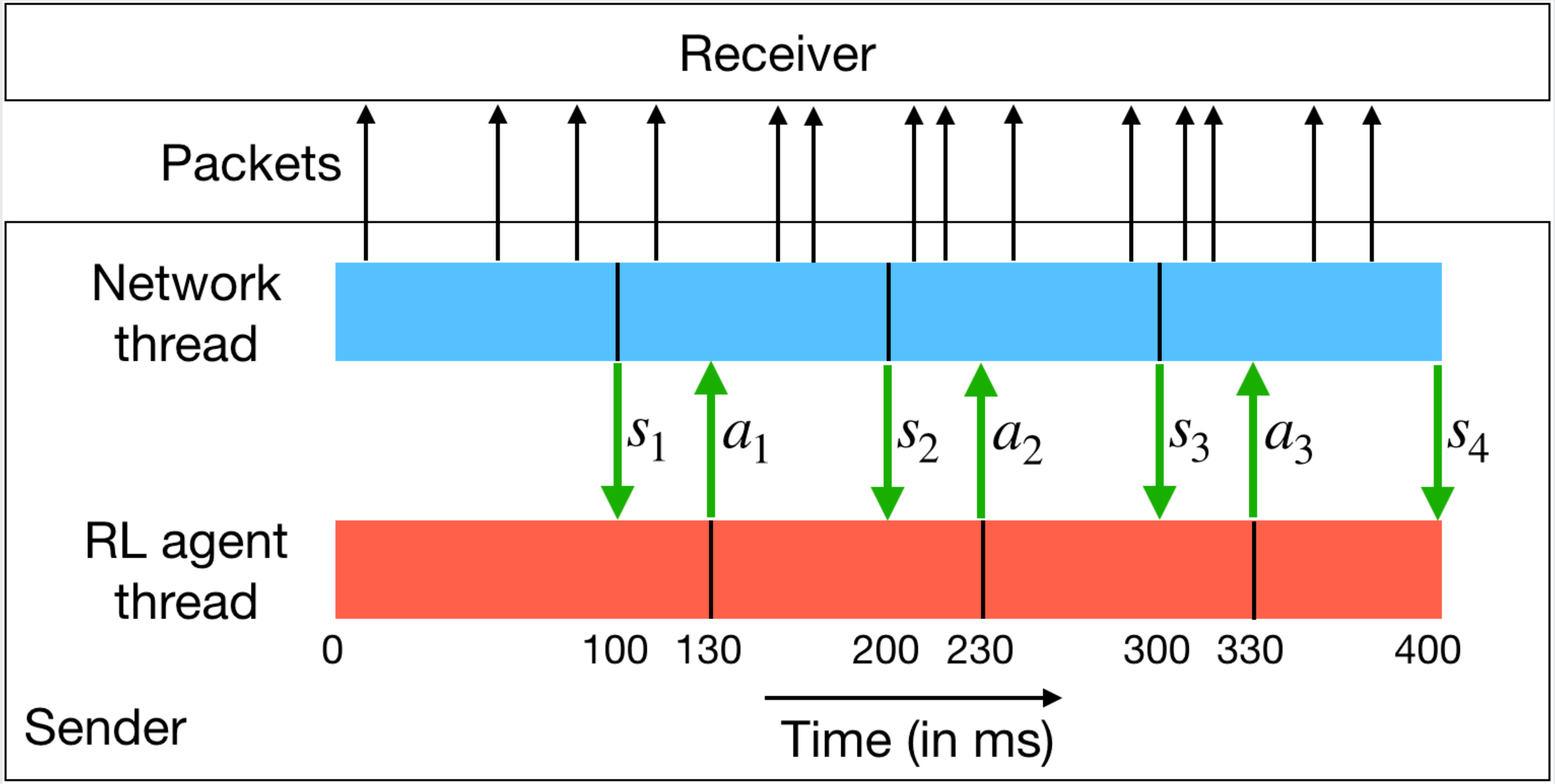
```python
env = ...
state = env.reset()
for _ in range(1000):
  action = model(state)  # Blocks the environment
  state, reward, done = env.step(action)
  if done:
    state = env.reset()
```



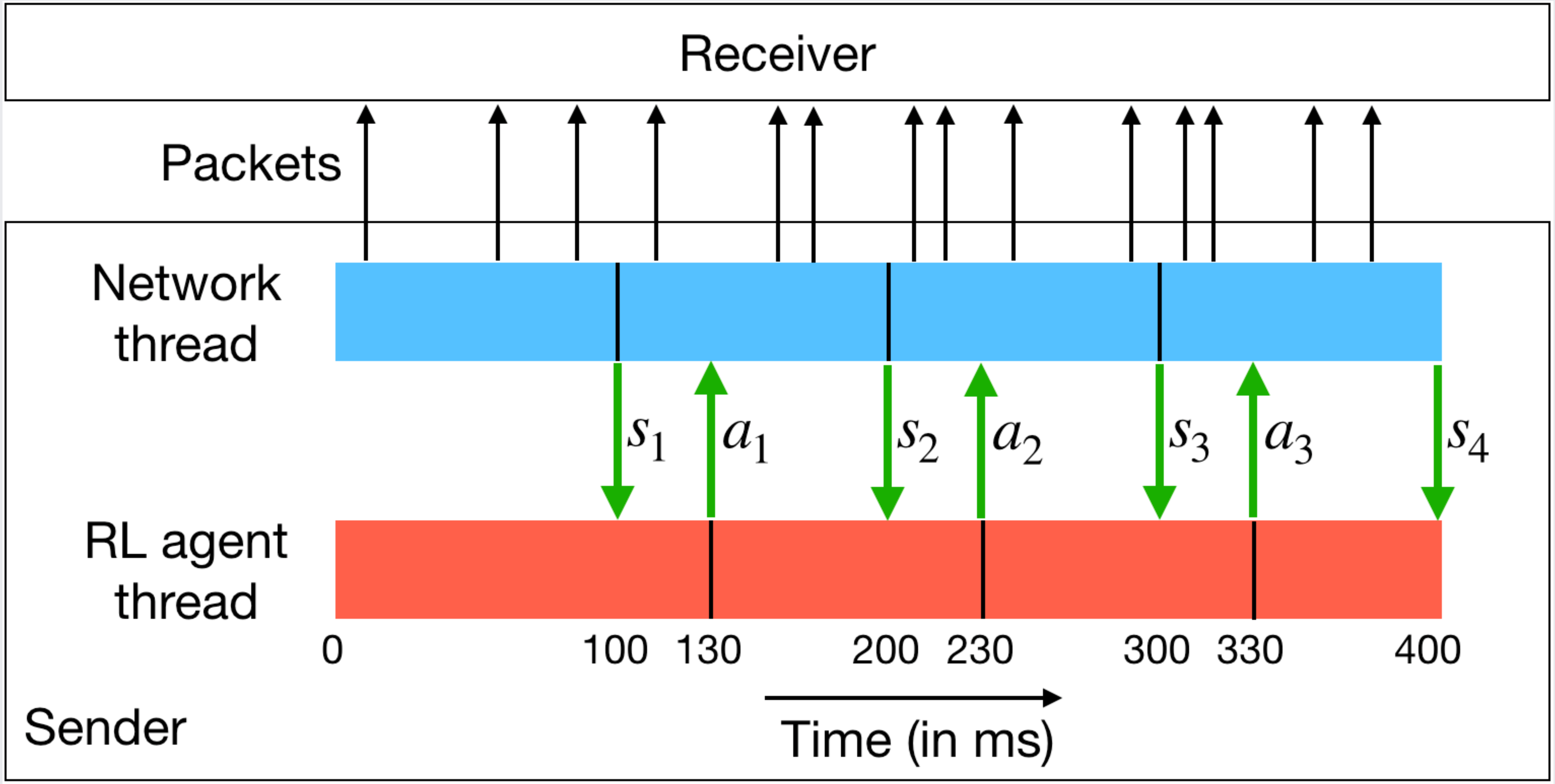
25ms Policy Lookup => 1.1% fewer bytes sent

50ms Policy Lookup => 11.4% fewer bytes sent

Asynchronous Environment



Asynchronous Environment



Not Markovian!

MDP with Delayed Actions

Augmented State Space: $\hat{S} = S \times A^k$

k = Action history length

MDP with Delayed Actions

Augmented State Space: $\hat{S} = S \times A^k$

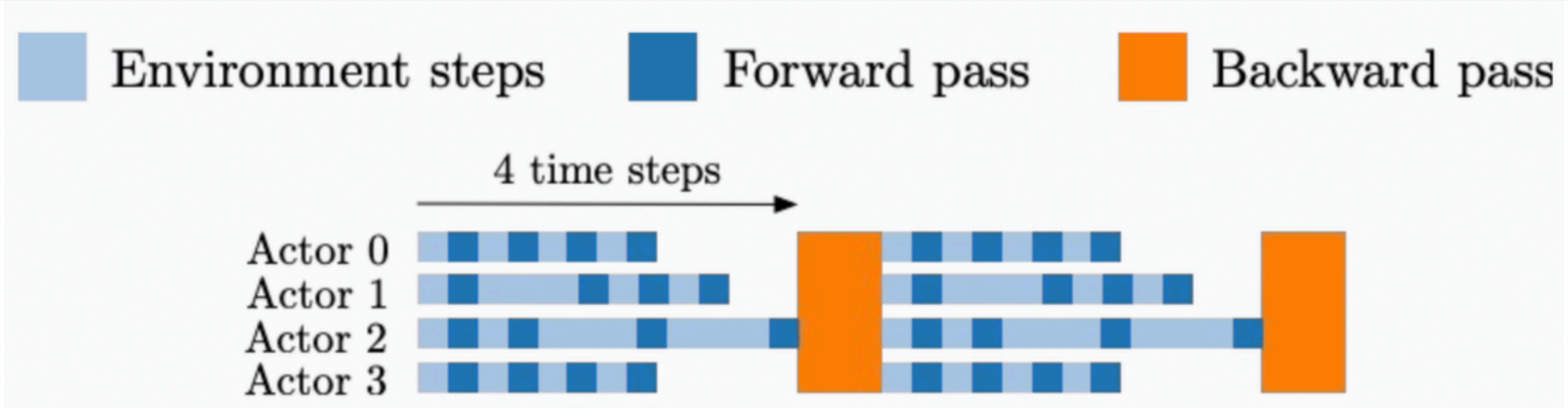
k = Action history length

Environment State:

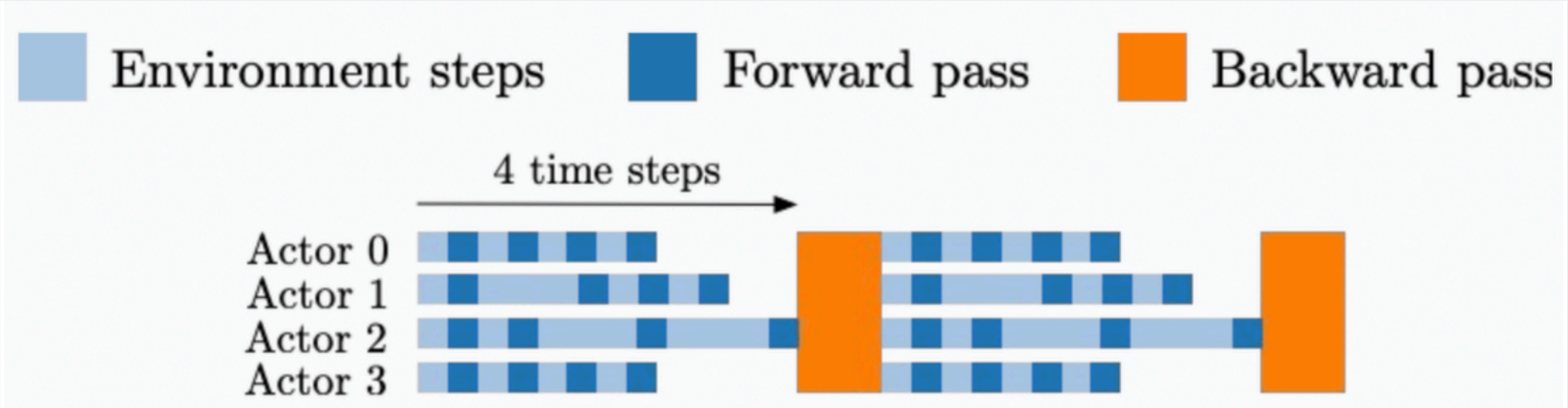
- 20 raw features
- Summary statistics:
  - sum, mean, max, mean, std
  - 20 features x 5 metrics =100 features
- Action history: one-hot actions + cwnd

State Vector: $100 + k \times (|A| + 1)$

Asynchronous Off-Policy Training



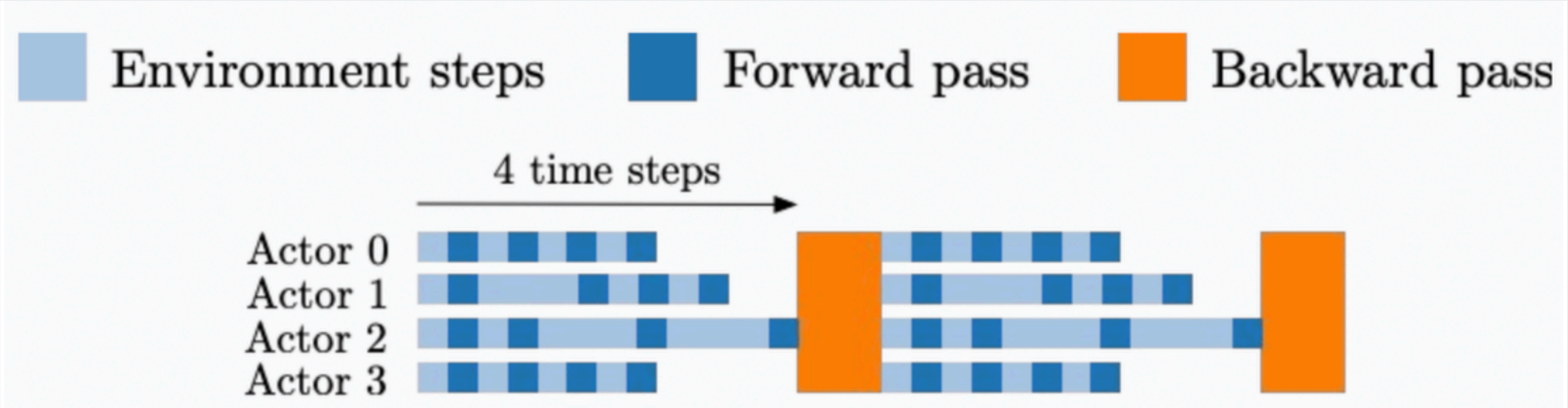
Asynchronous Off-Policy Training



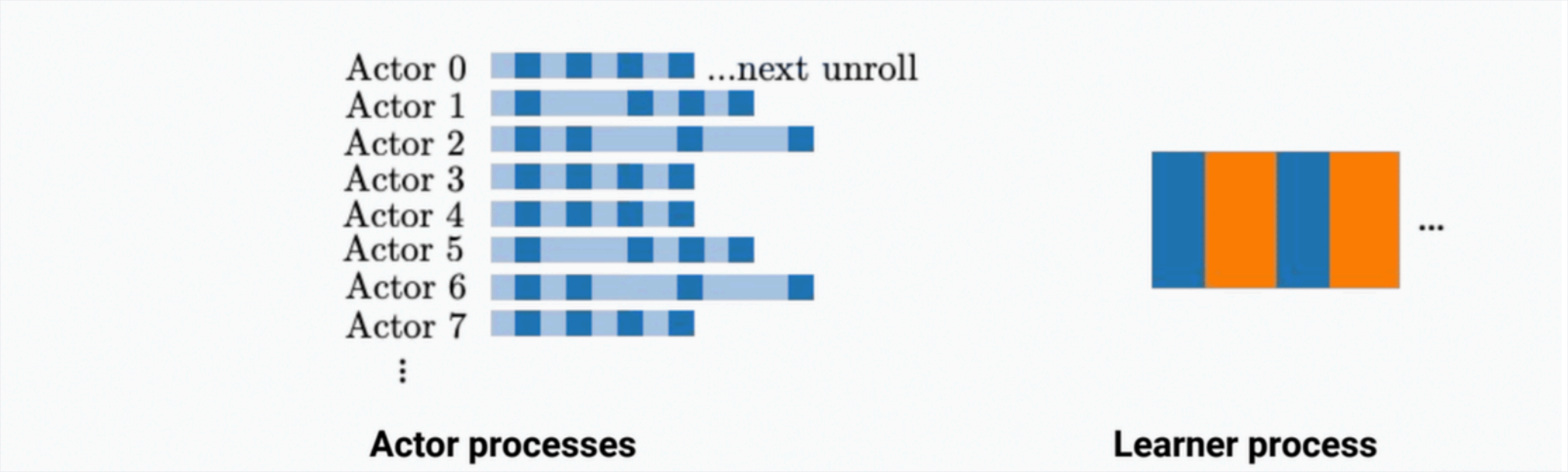
Synchronous Actor-Critic

IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, Espeholt et al.

Asynchronous Off-Policy Training



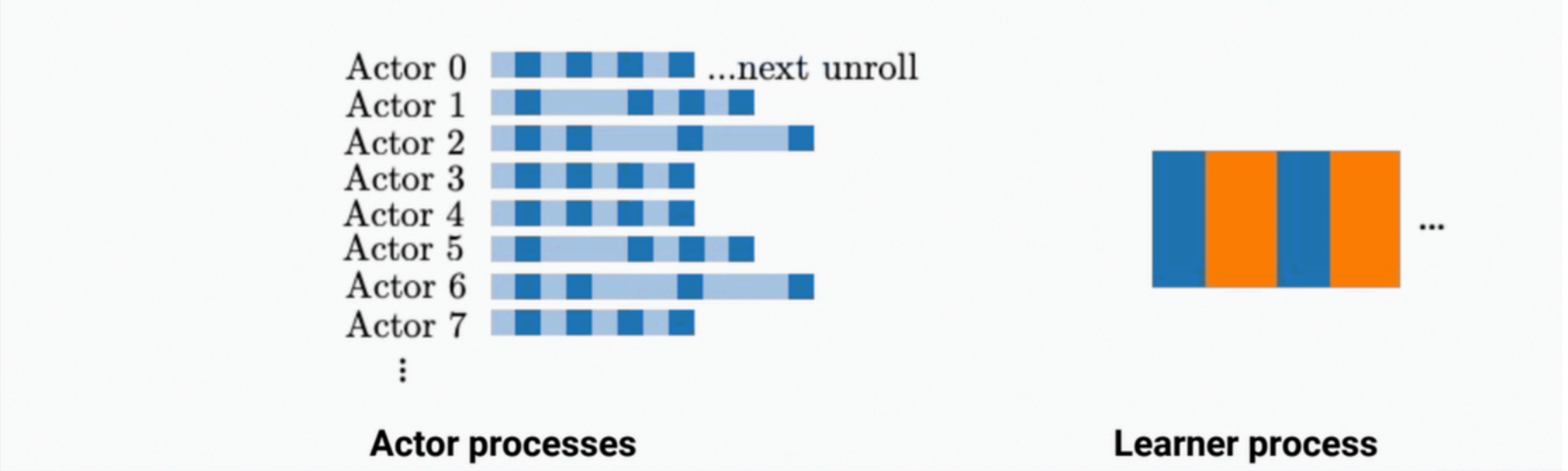
Synchronous Actor-Critic



Asynchronous Actor-Critic

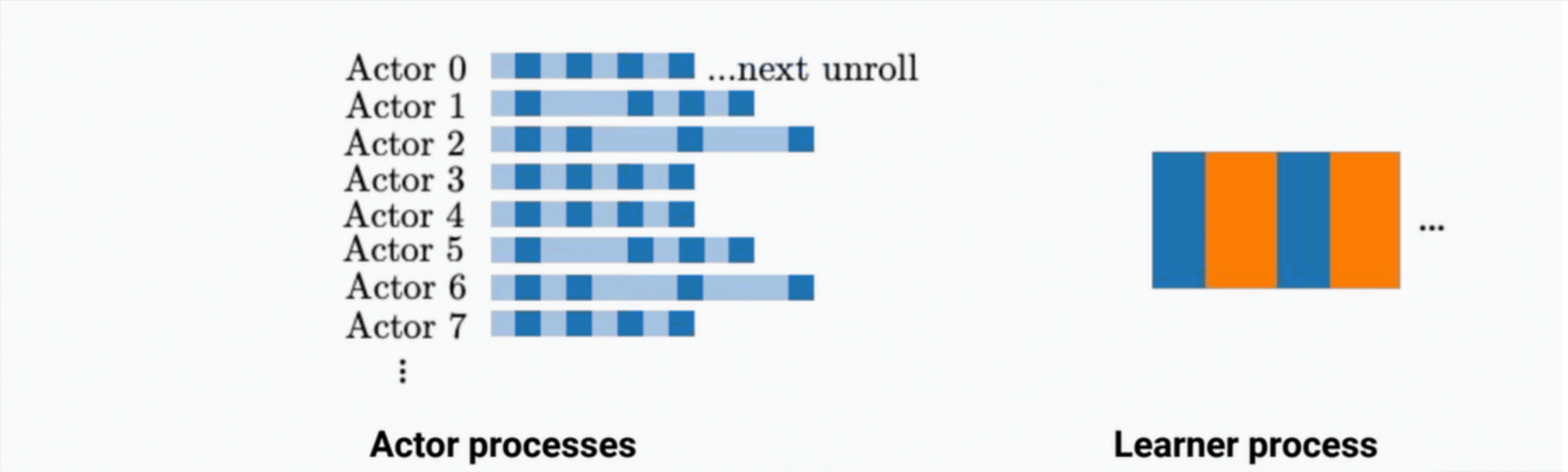
IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, Espeholt et al.

mvfst-rl Training Architecture

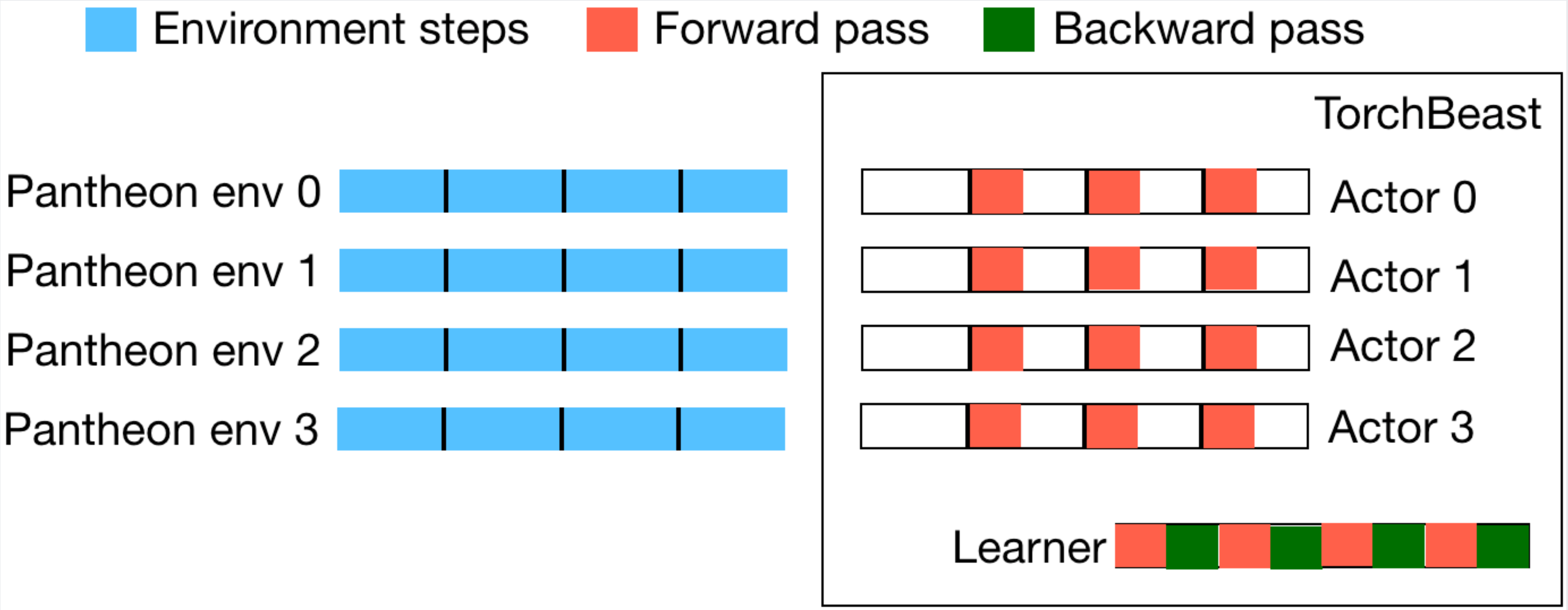


IMPALA:
- Async Actor-Learner
- Sync Environment-Actor

mvfst-rl Training Architecture

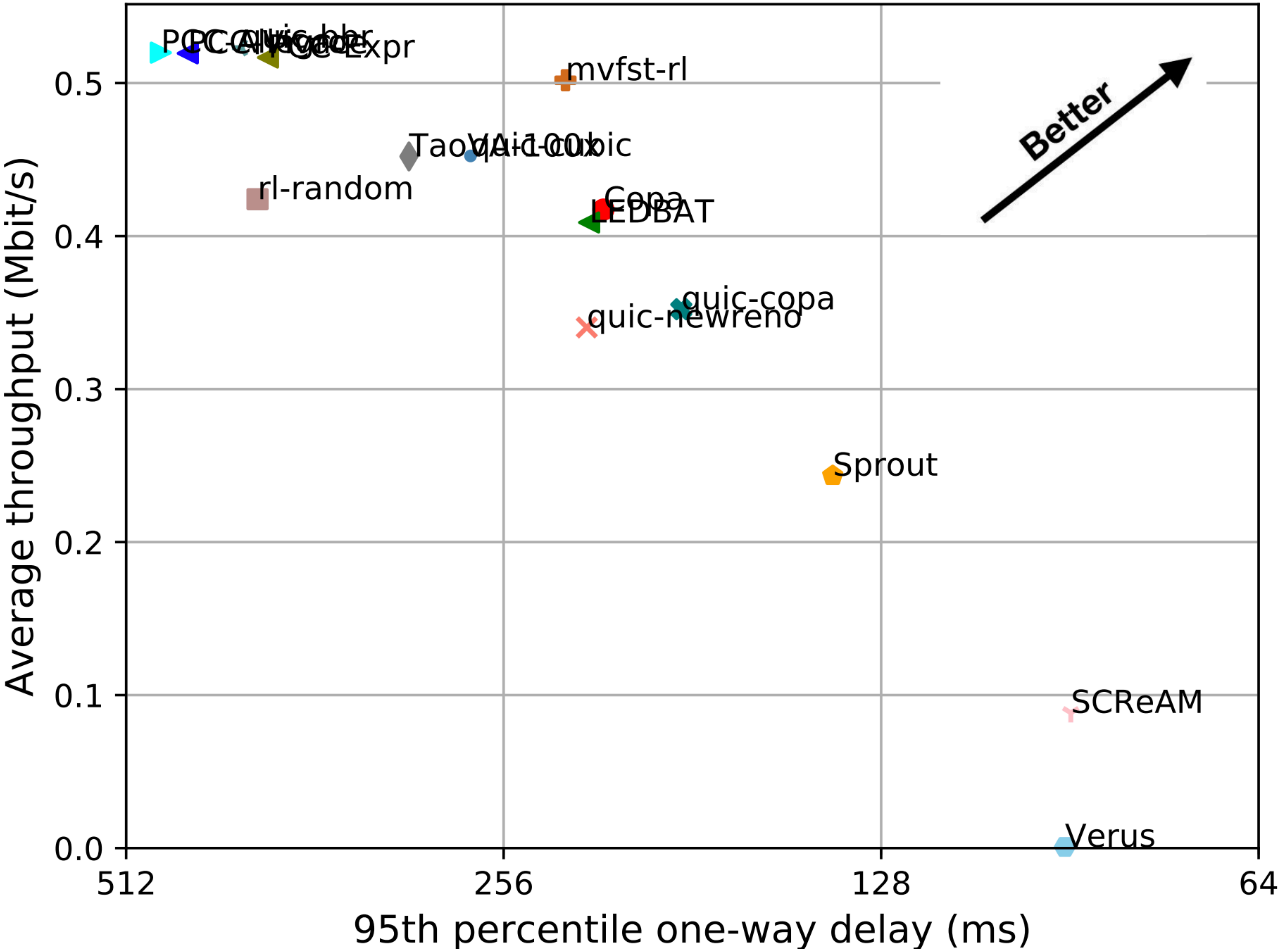


IMPALA:
- Async Actor-Learner
- Sync Environment-Actor

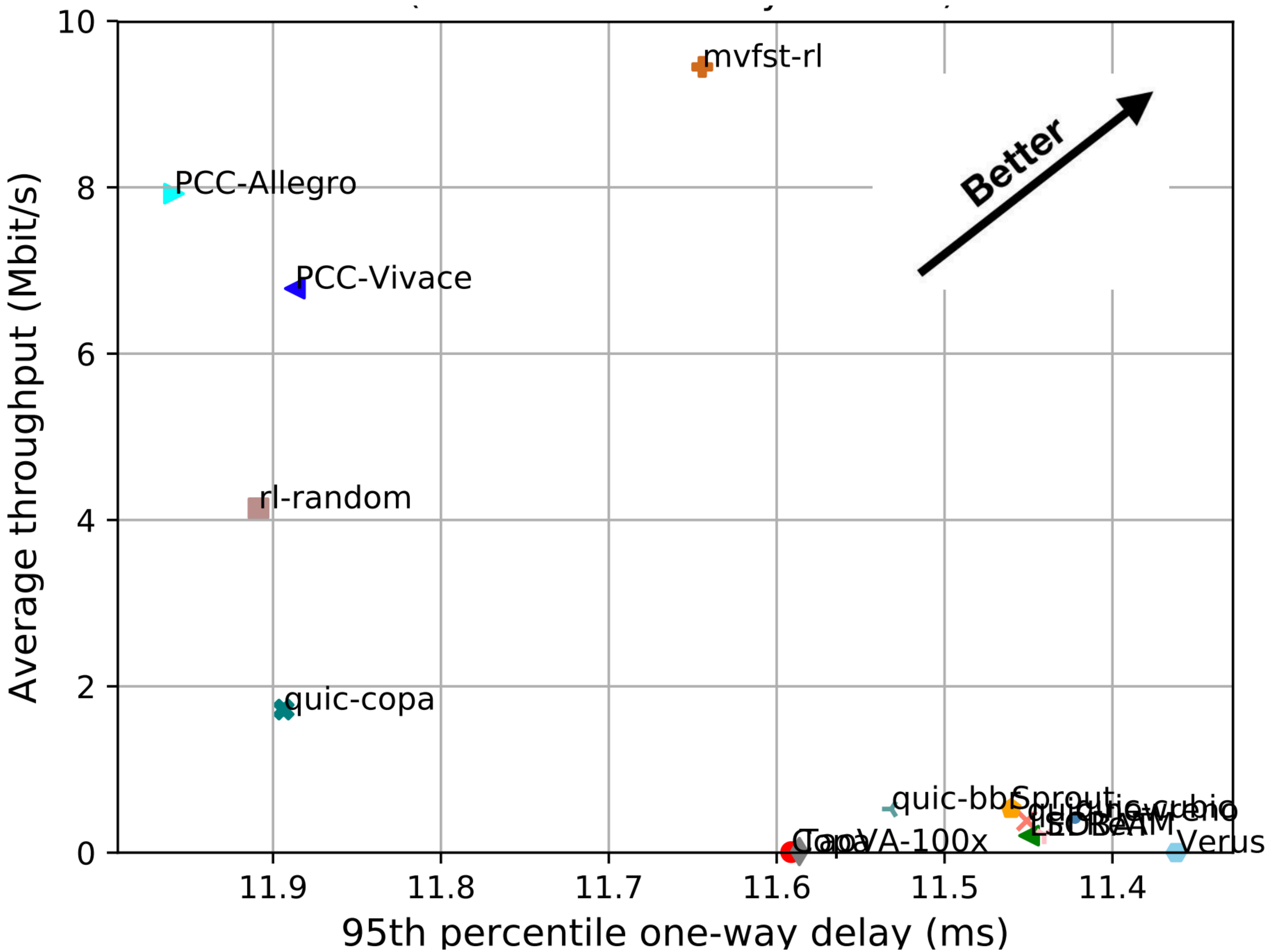


mvfst-rl:
- Async Actor-Learner
- Async Environment-Actor

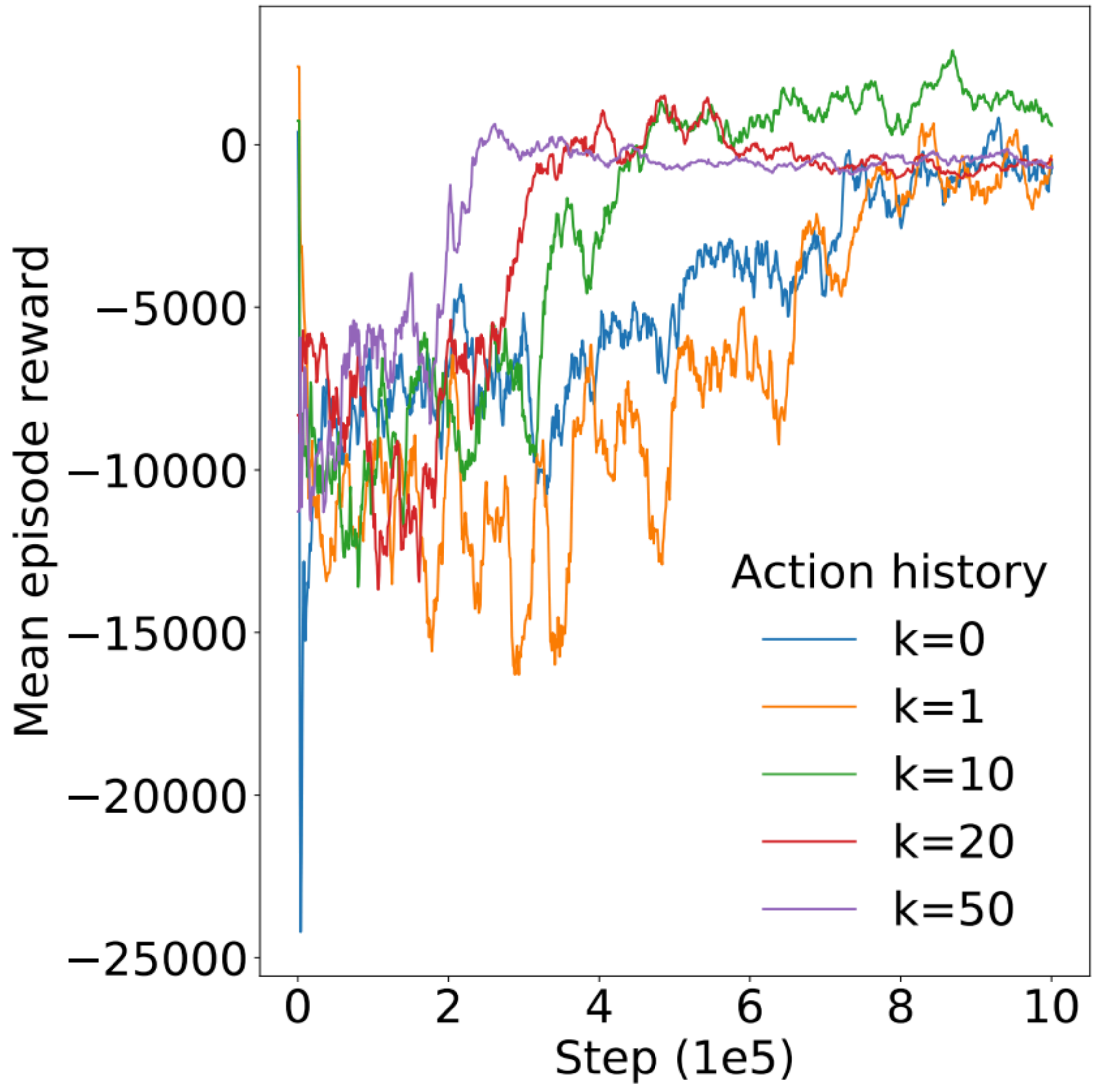
Results



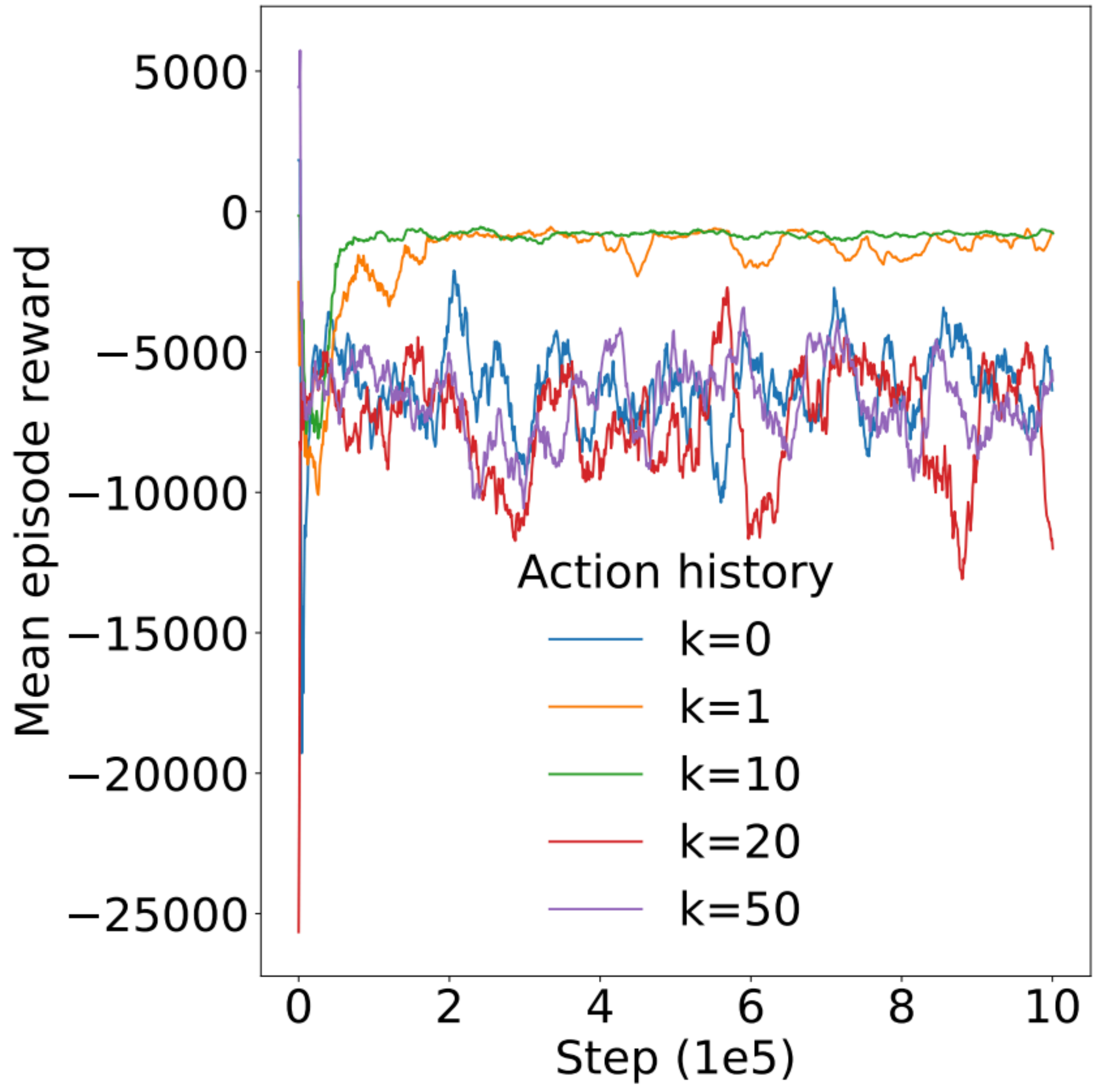
(a) Nepal to AWS India (calibrated)



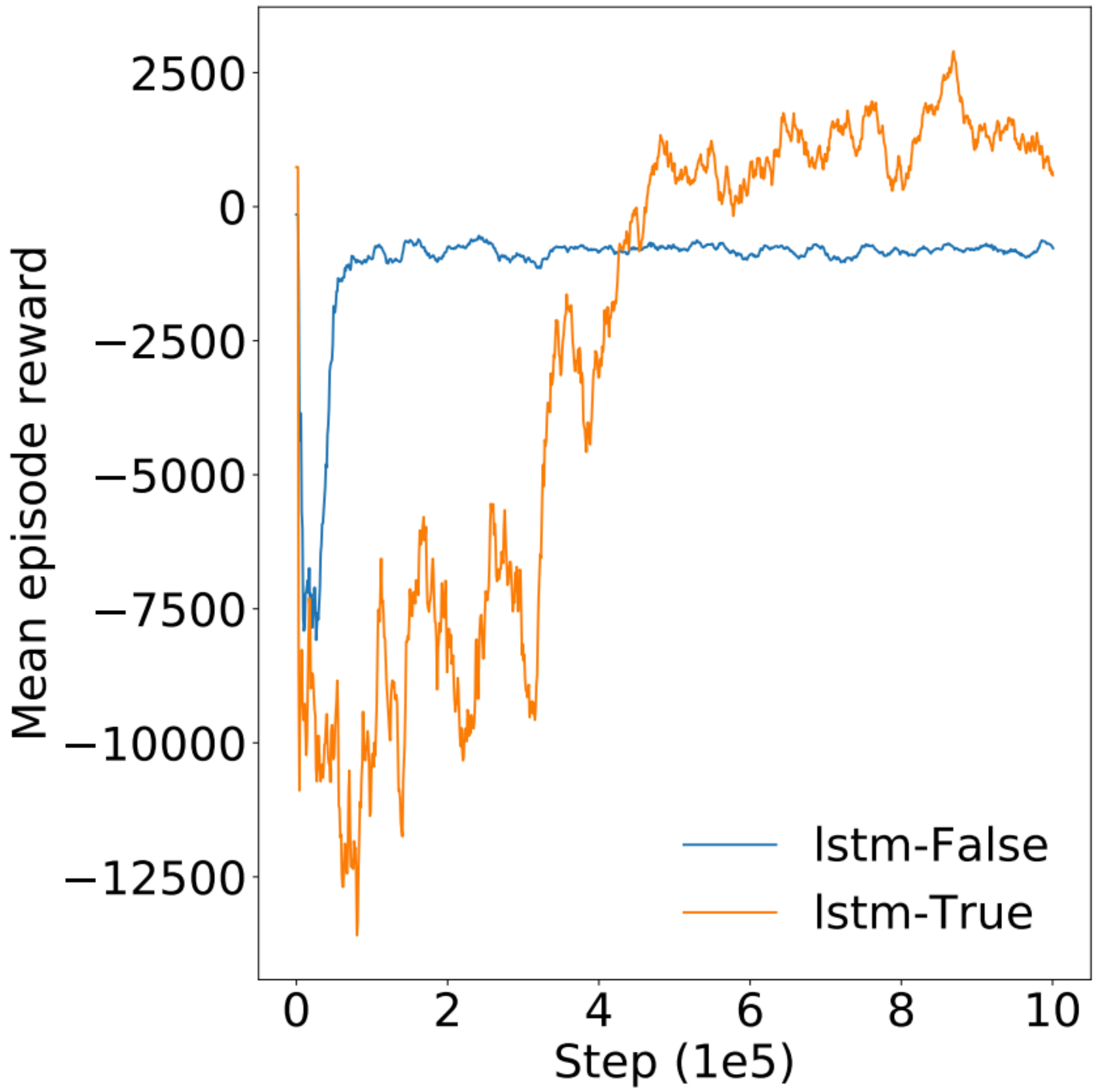
(b) Token-bucket policer (unseen environment)



(c) With LSTM



(d) Without LSTM



(e) Action history = 10

Open Research Questions

Generalization across network policies

Faster policy lookup time

Online learning

RL as an alternative to hand-engineered systems

Internet as a platform for real-world RL system

Slow and steady research

github.com/facebookresearch/mvfst-rl

RL as an alternative to hand-engineered systems

Internet as a platform for real-world RL system

Slow and steady research

github.com/facebookresearch/mvfst-rl

RL as an alternative to hand-engineered systems

Internet as a platform for real-world RL system

Slow and steady research

Questions?