Building Better Network Simulators with Machine Learning

Mohammad Alizadeh

How to evaluate new ideas?

- Randomized Control Trial (RCT): The gold standard – measures reality
  - ➢ Measures reality

    Time consuming and risks disruption

    Not repeatable → poorly suited to iterative design & experimentation

    Not always possible (e.g., new hardware)

- Simulation: A model of reality
  - ➢ Great flexibility, control, repeatability
  - ➢ Widely used in the design, planning and operation of networks

Simulation in emerging AI-driven systems

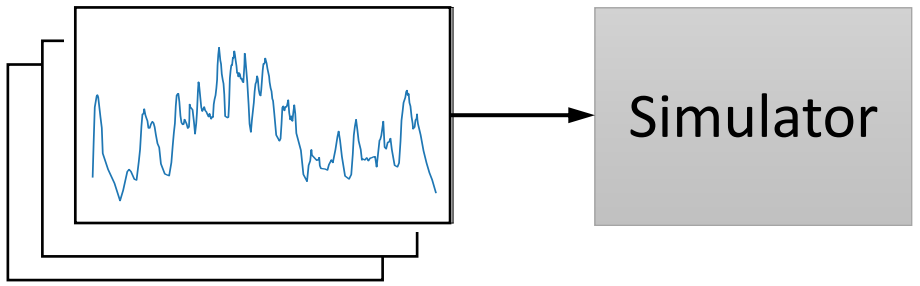
Simulation landscape

Full system

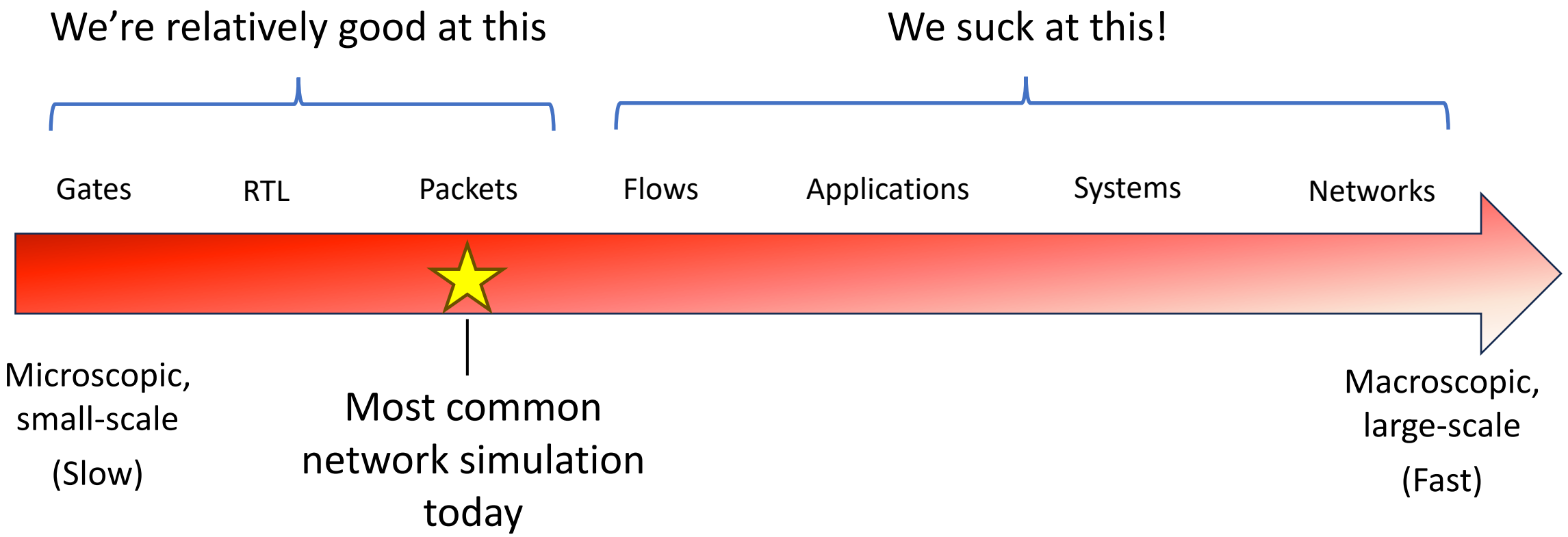
- Model all system components
  Needs detailed knowledge
  Tedious to build

Trace-driven

- Use traces to capture parts of the system
- Lower complexity & effort
  Often less accurate than full-system sim.



From microscopic to macroscopic



Today's simulators are too low-level for many questions

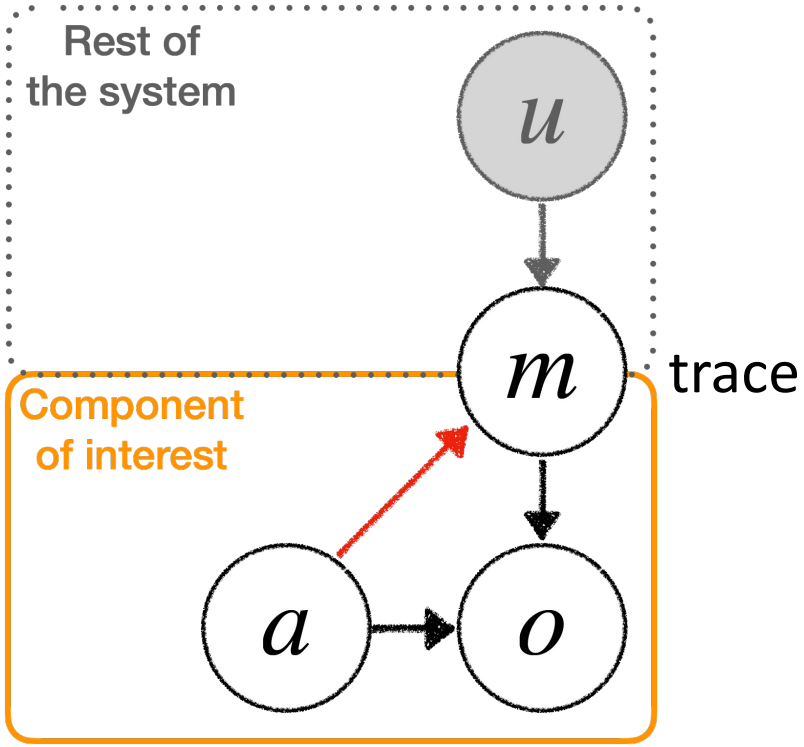
We lack accurate high-level models of real networks,
so we resort to low-level simulation

Vision: Learning high-level network dynamics from data

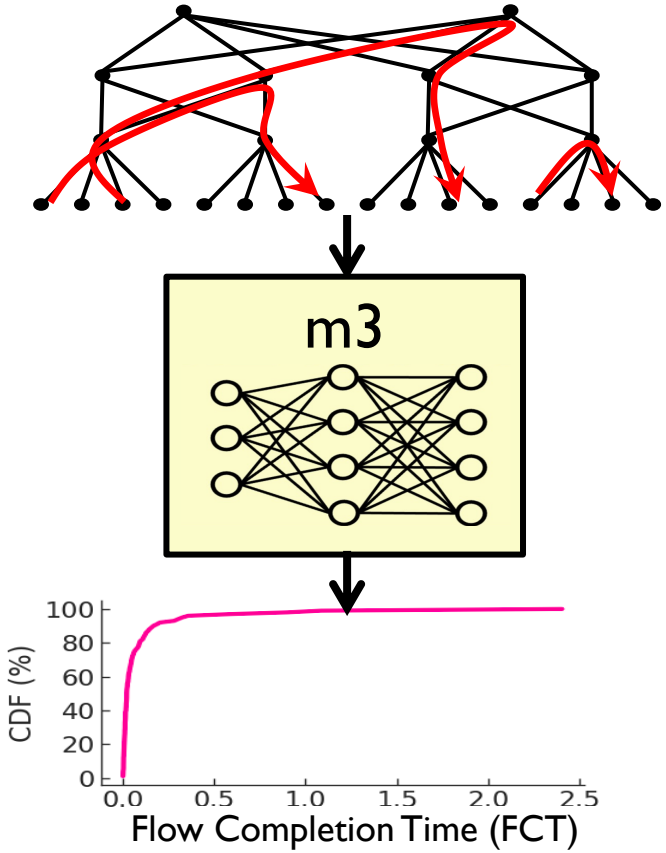
- Networks produce massive amounts of data
- Relatively easy to add instrumentation, run active experiments, …

Can we use large-scale data to learn accurate high-level network models?

Rest of this Talk…



CausalSim [NSDI'23]:
Unbiased trace-driven simulation

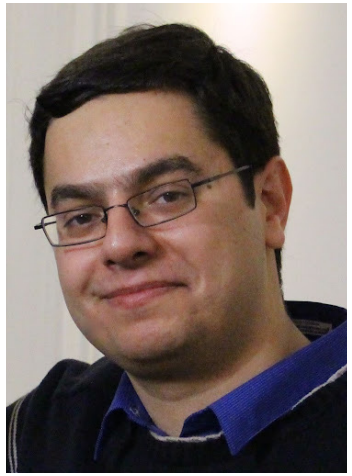


m3 [SIGCOMM'24]:
Fast flow-level performance estimation

CausalSim: A Causal Framework for Unbiased Trace-Driven Simulation








Abdullah Alomar*

Pouya Hamadanian*

Arash Nasr-Esfahany*

Anish Agarwal

Mohammad Alizadeh

Devavrat Shah

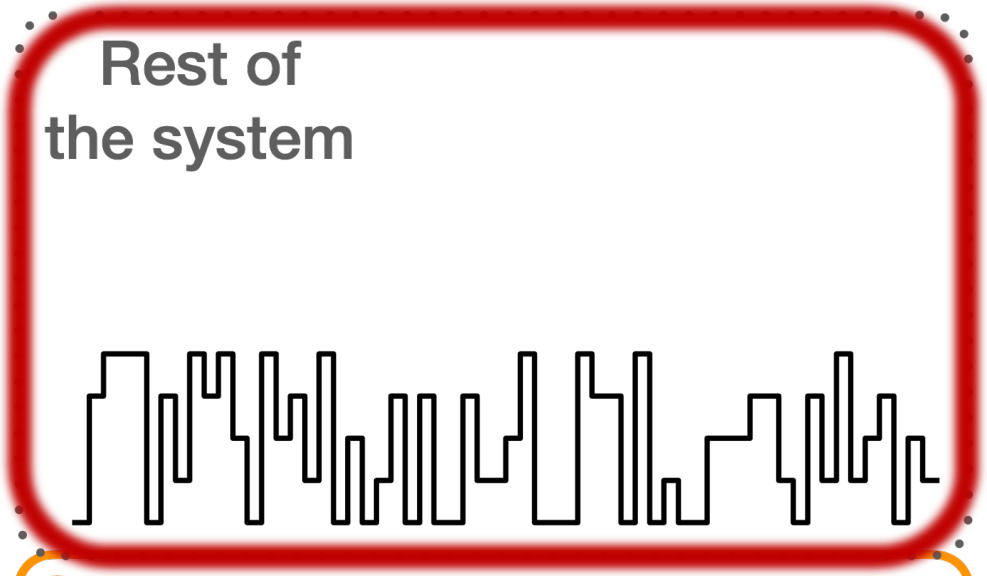
Massachusetts Institute of Technology

MIT CSAIL

*Equal Contribution

Trace-Driven Simulation

Collected Trace



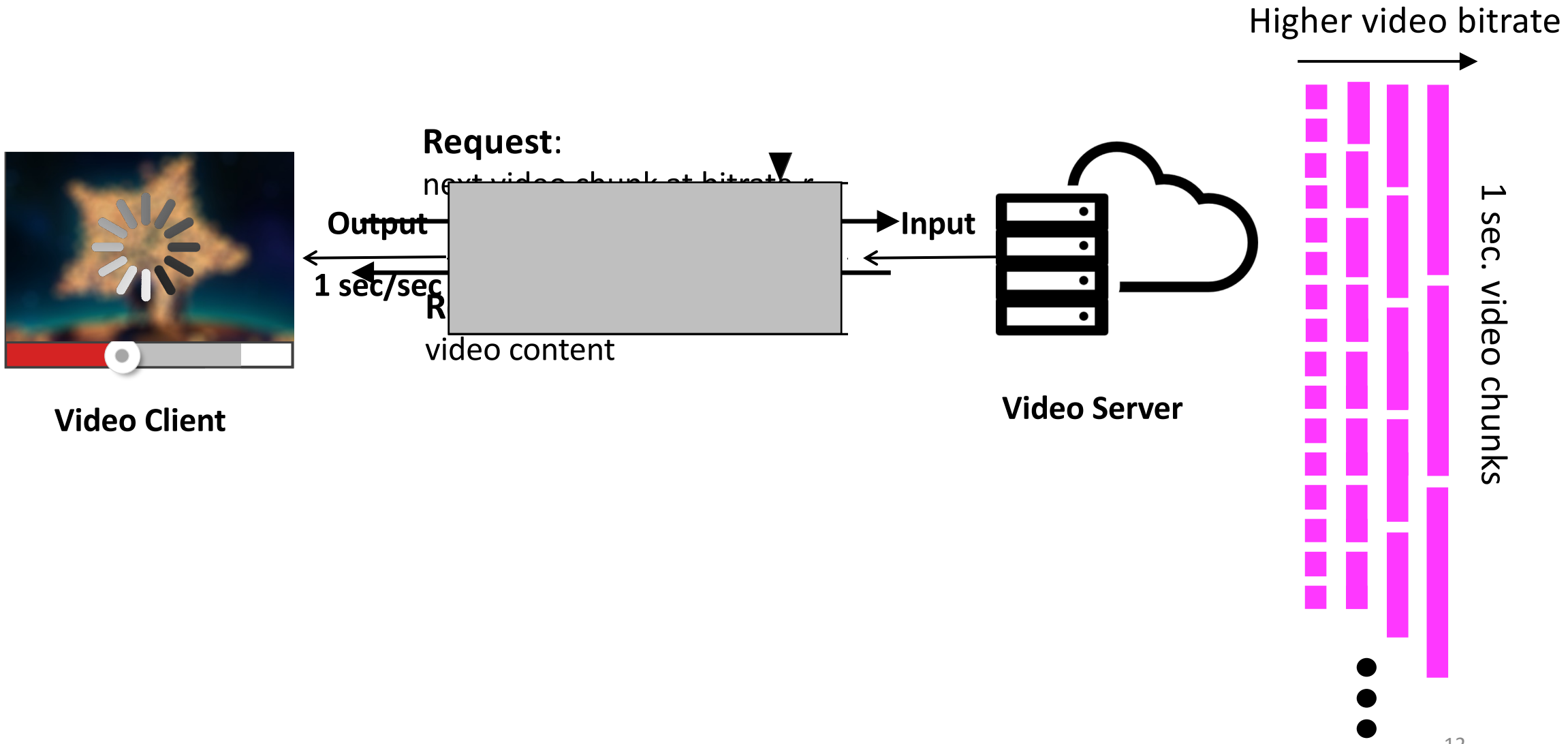
Simulation

- ***Exogenous trace*** assumption:
  Simulated interventions would not affect the replayed trace

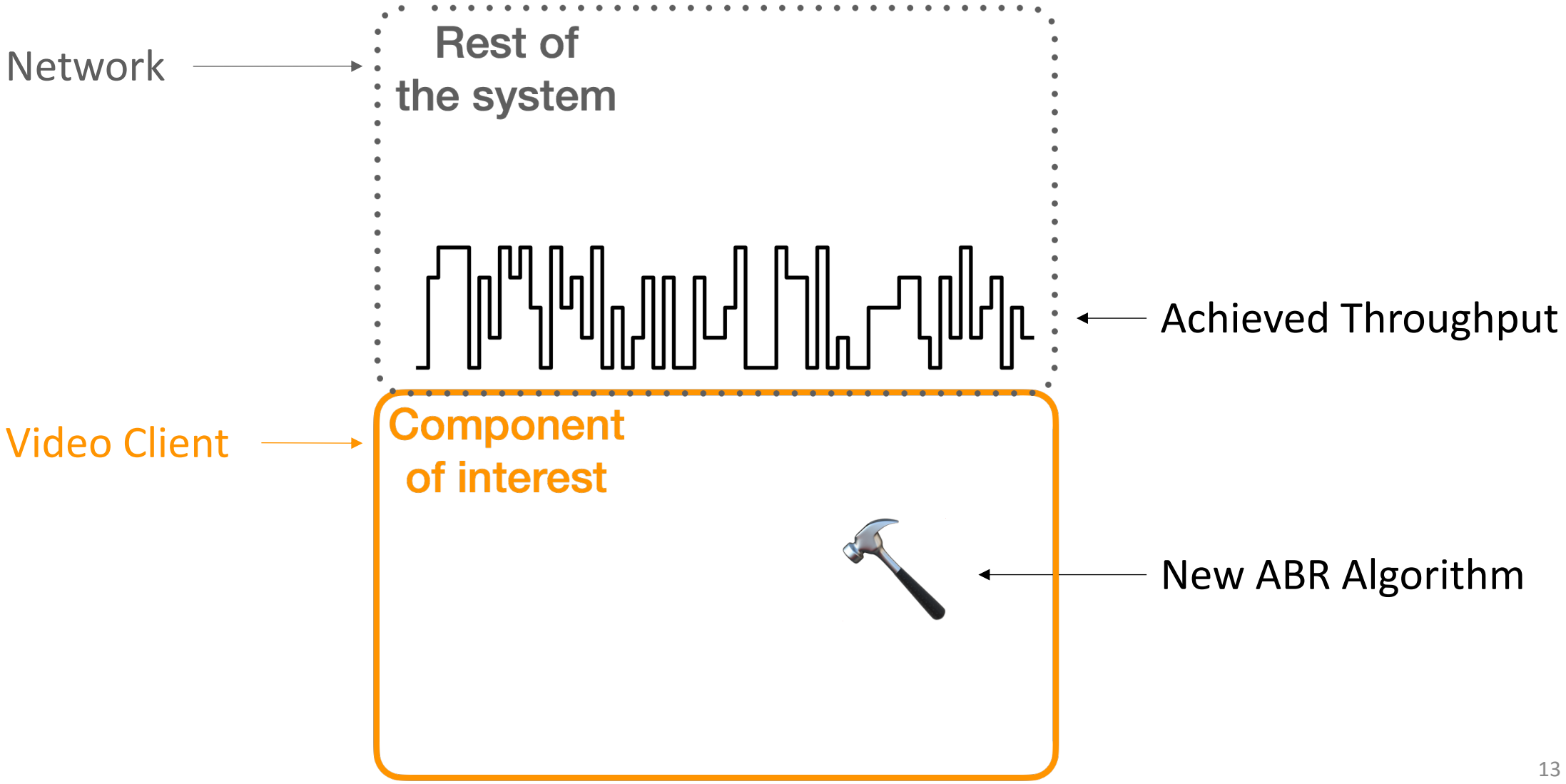
Exogenous trace assumption does not hold for many real-world traces

… hurts accuracy, can lead to completely wrong conclusions

Example: Adaptive Bitrate Video Streaming (ABR)

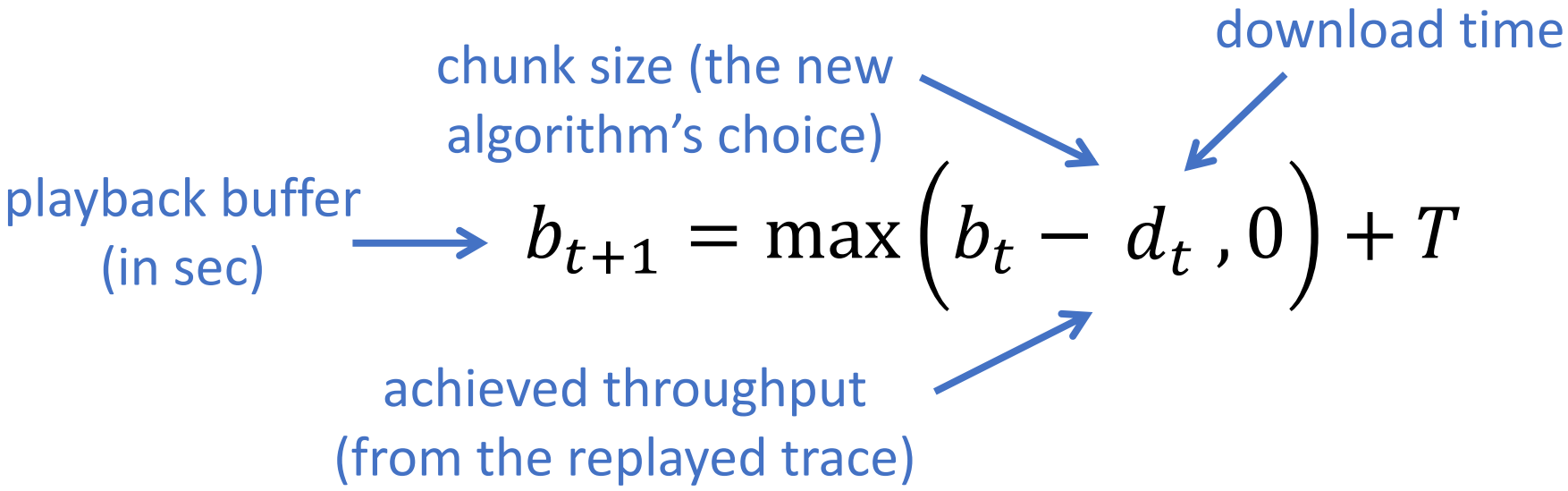


Trace-driven simulation for ABR



Trace-driven simulation for ABR

- "ExpertSim" [e.g., Yin et al. SIGCOMM'15, Mao et al. SIGCOMM'17]



The Puffer dataset

- The Puffer Randomized Control Trial [Yan et al. NSDI 2020]
  - July 2020 – June 2021
  - 5 ABR algorithms
  - 56M downloaded chunks over 230K streaming sessions
  - 3.5 years worth of streamed video

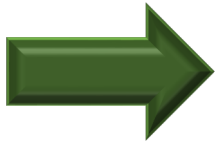
Task: Given the traces for all except one ABR algorithm, simulate the held out algorithm on the same paths

A typical trace

Chunk #	1	2	3	
Bitrate	360p	480p	480p	
Achieved Throughput	1Mbps	0.8Mbps	1.2Mbps	• • •
Playback Buffer	5s	3s	7s	

Algorithm
BBA
("source")

A simulated trajectory



Chunk #	1	2	3	
Bitrate	**720p**	**?**	**?**	
Achieved Throughput	**?**	**?**	**?**	• • •
Playback Buffer	**?**	**?**	**?**	

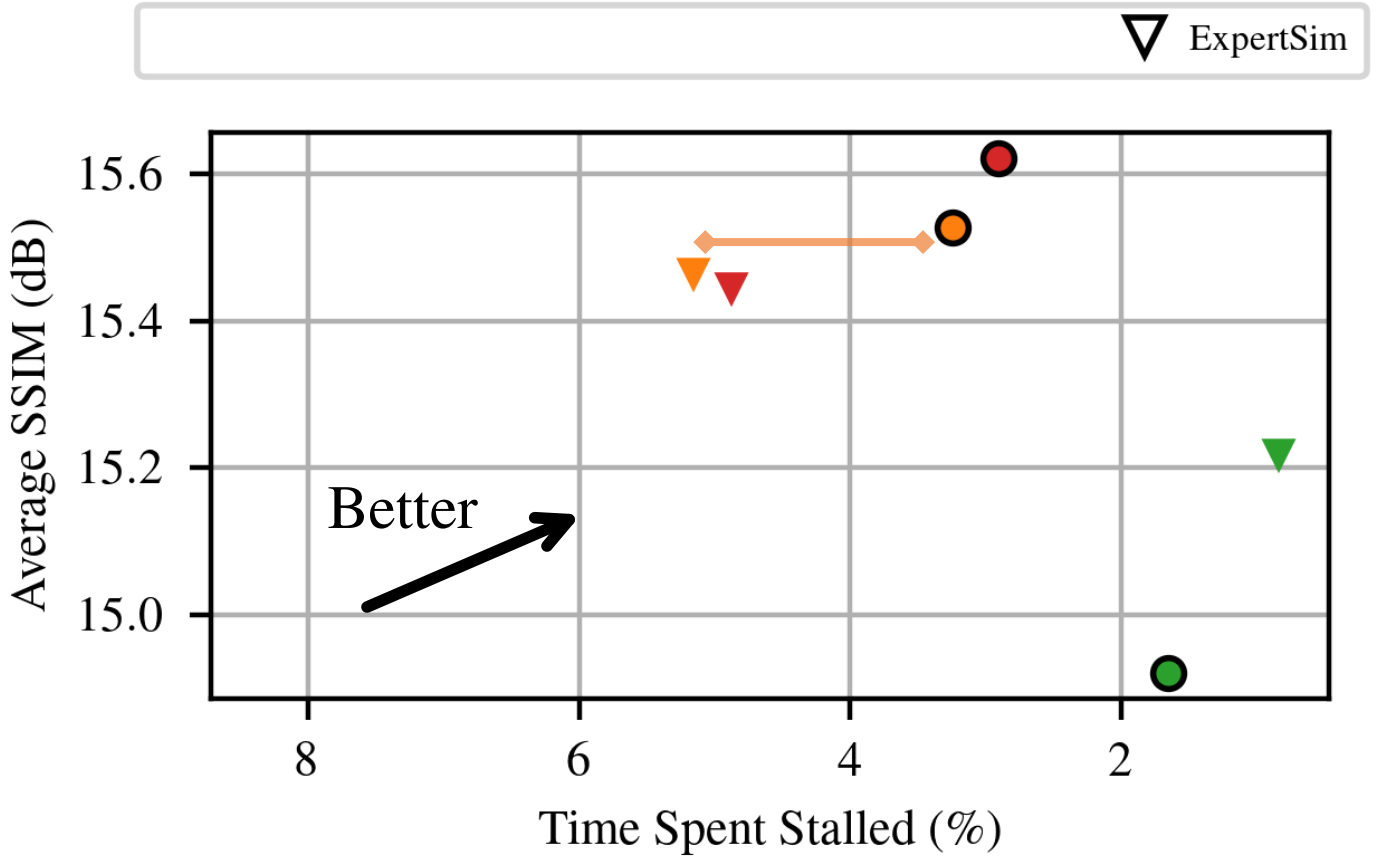
Algorithm
BOLA1
("target")

How accurate is trace-driven simulation?

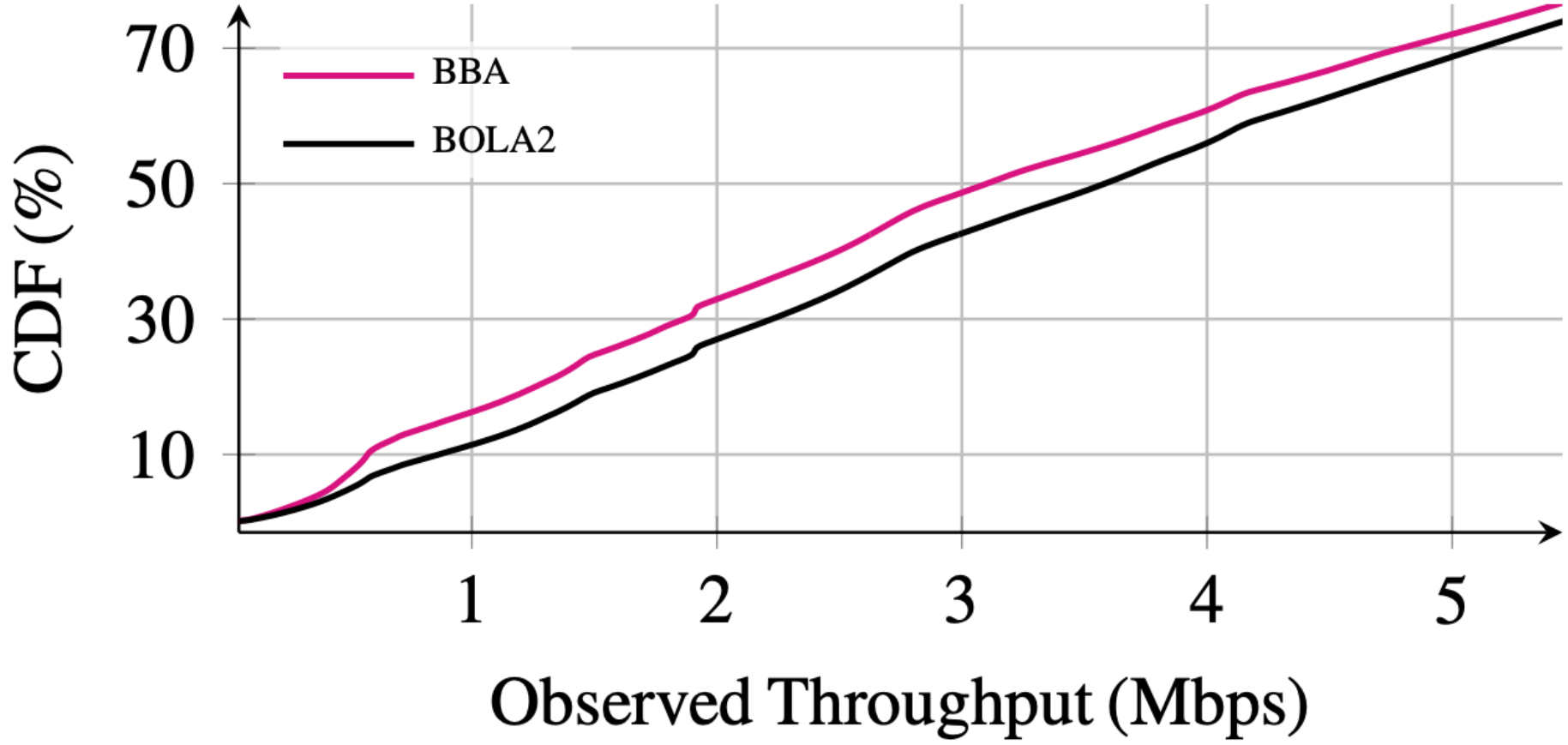
- Source data: The Puffer Dataset with 5 algorithms
- Target algorithm (unseen): BBA, BOLA1, BOLA2

Exogenous trace assumption

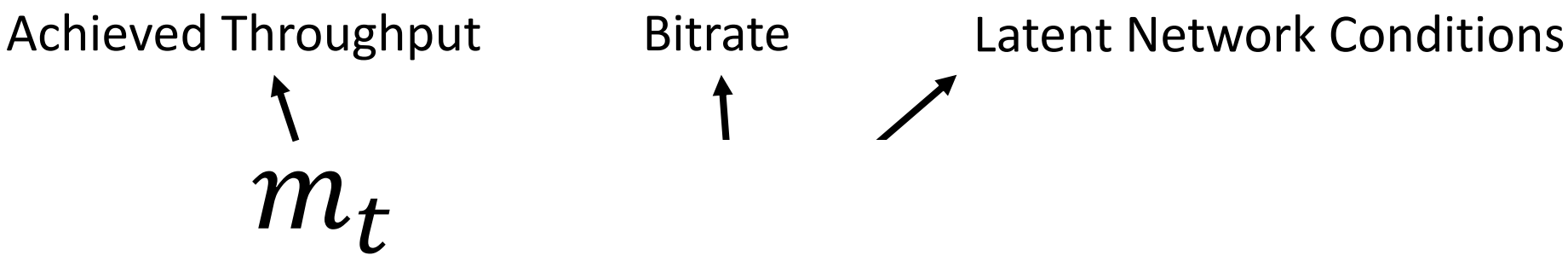
Bitrates chosen by the ABR algorithm affect the achieved throughput



ABR algorithms affect throughput



Can we relax the exogenous trace assumption?



Both u_t and $f(\cdot)$ are unknown

Towards a solution

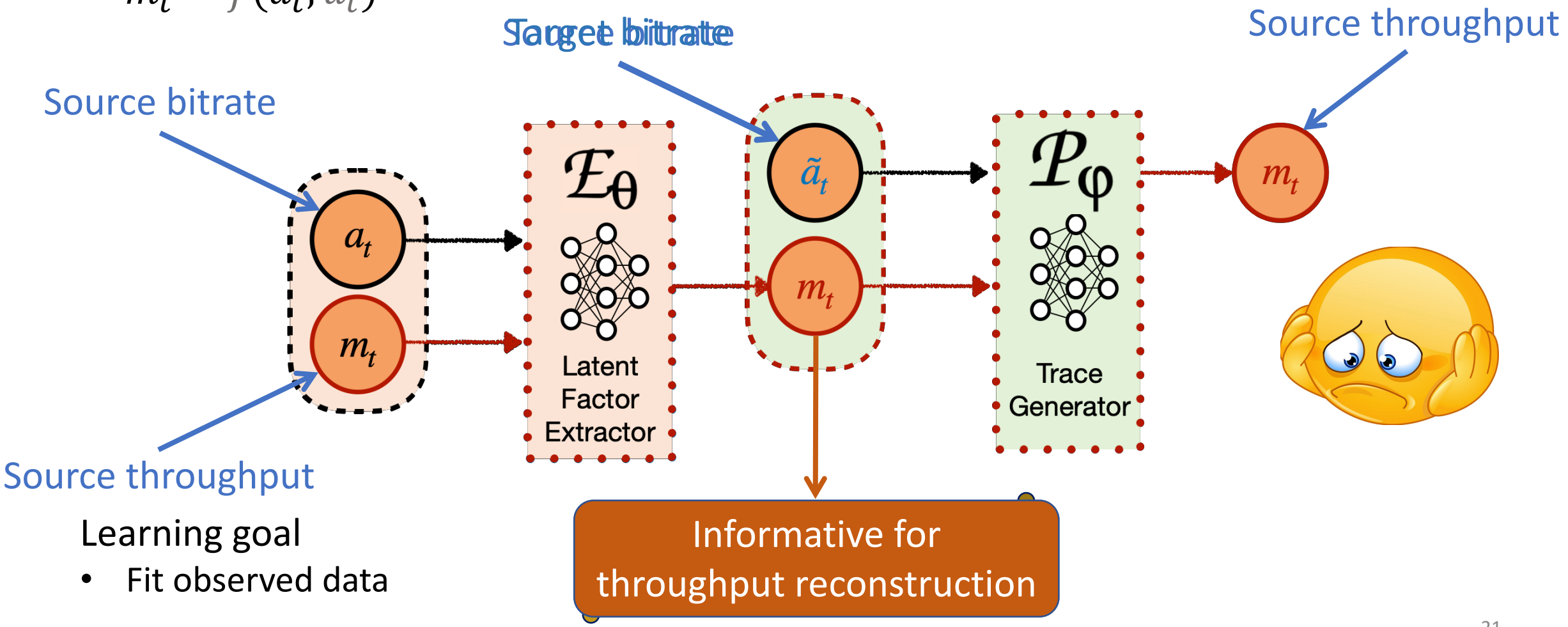
- If u and $f(\cdot)$ were known…

$$\widetilde{m}_t = f(\tilde{a}_t, u_t)$$

simulated bitrate
(counterfactual)

A learning approach

$$m_t = f(a_t, u_t)$$



Learning goal

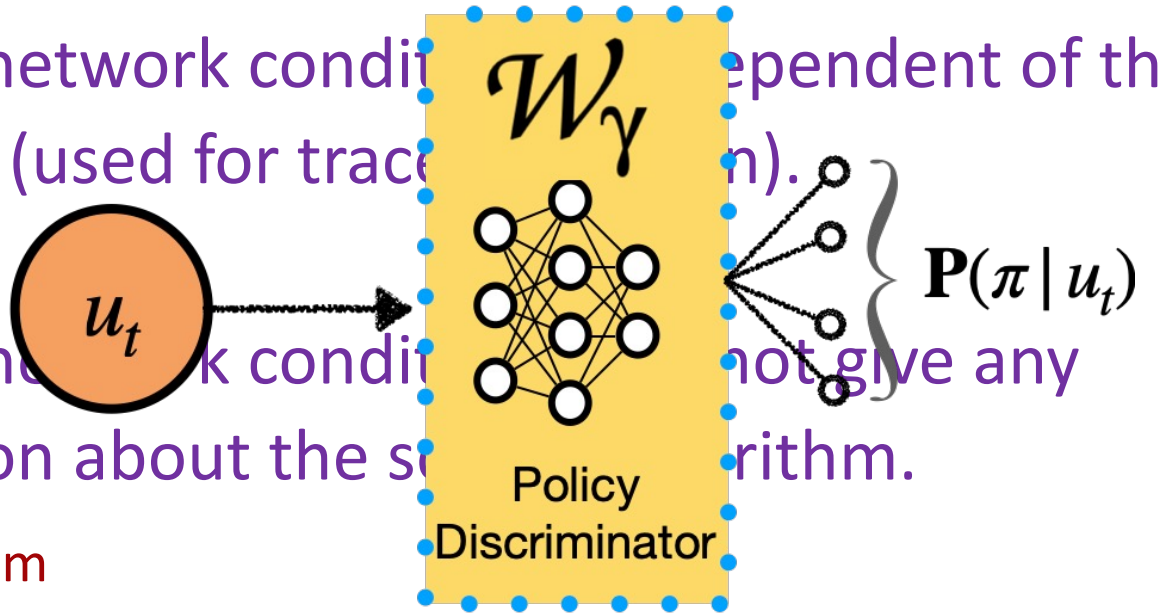
- Fit observed data

RCT to the rescue!

RCT property: Distribution of latent network conditions is the same in trajectories assigned to different algorithms.

↳ Latent network condi[illegible]ependent of the source algorithm (used for trac[illegible]n).

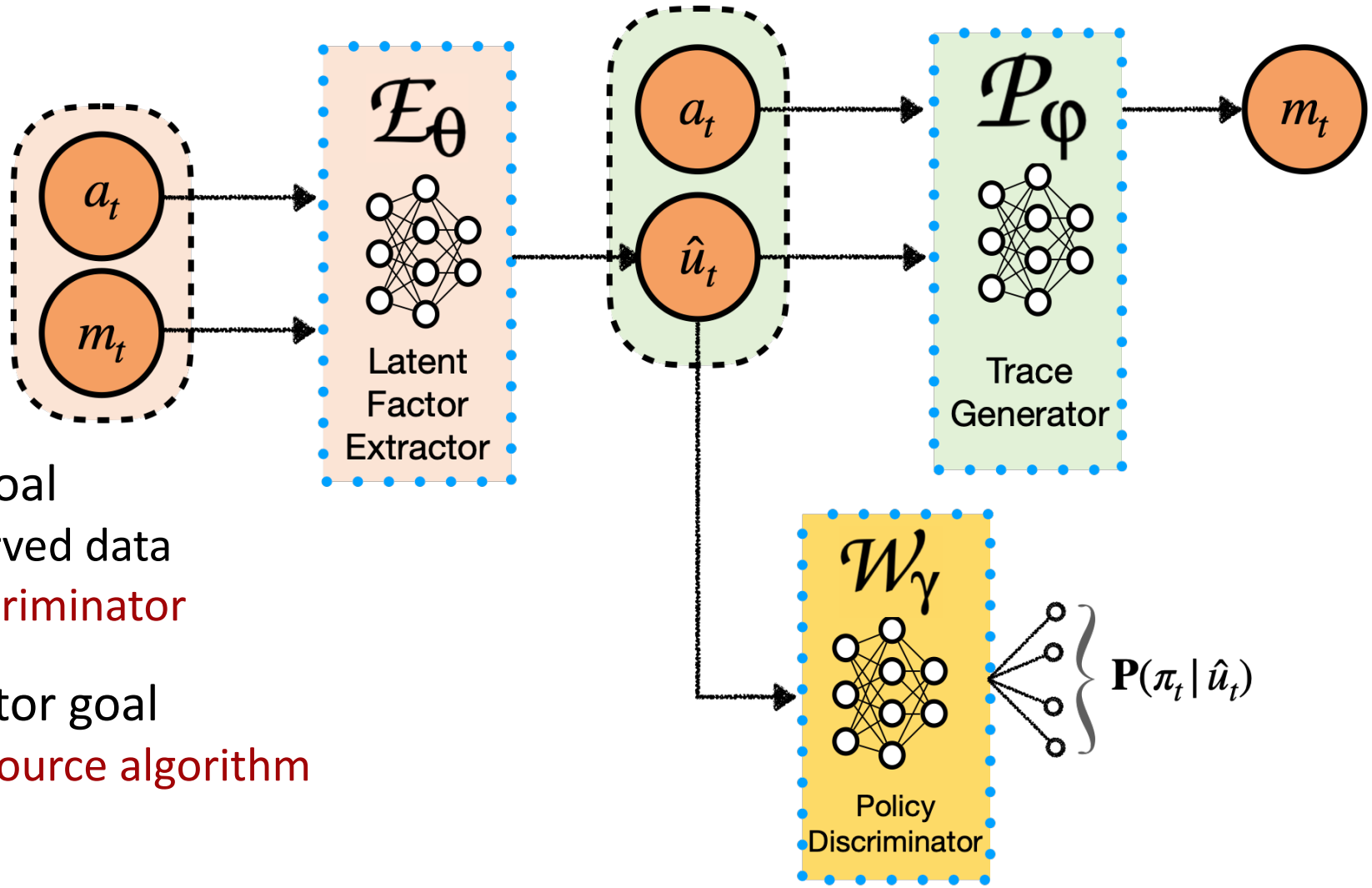
↳ Latent n[illegible]k condi[illegible]not give any information about the s[illegible]rithm.



Discriminator goal

- Predict source algorithm

Exploiting the RCT property



Learning goal

- Fit observed data
- Fool discriminator

Discriminator goal

- Predict source algorithm

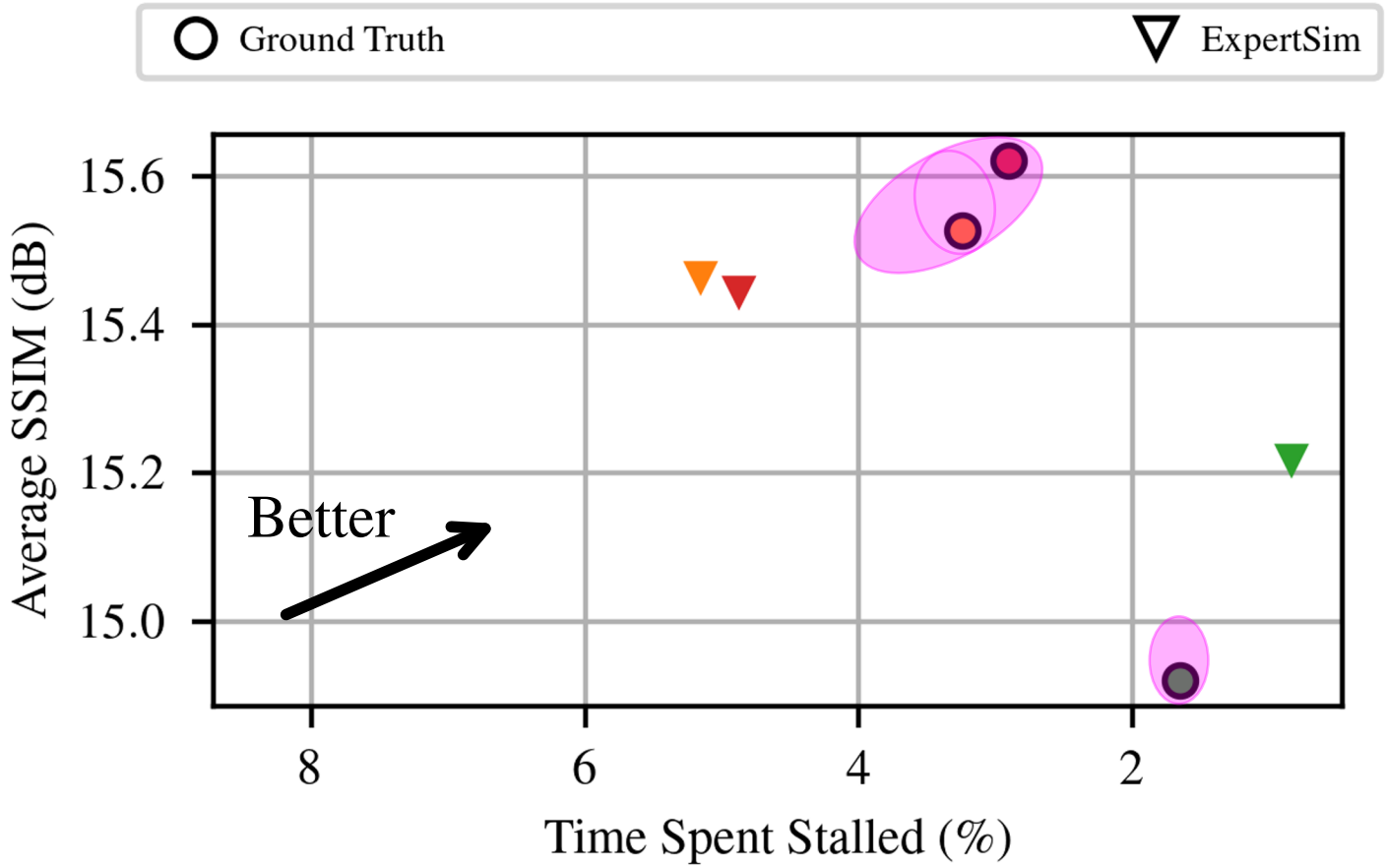
RCT property is sufficient for unbiased simulation

Theorem: The RCT property (independence of u and the algorithm) is sufficient for estimating the counterfactual trace if

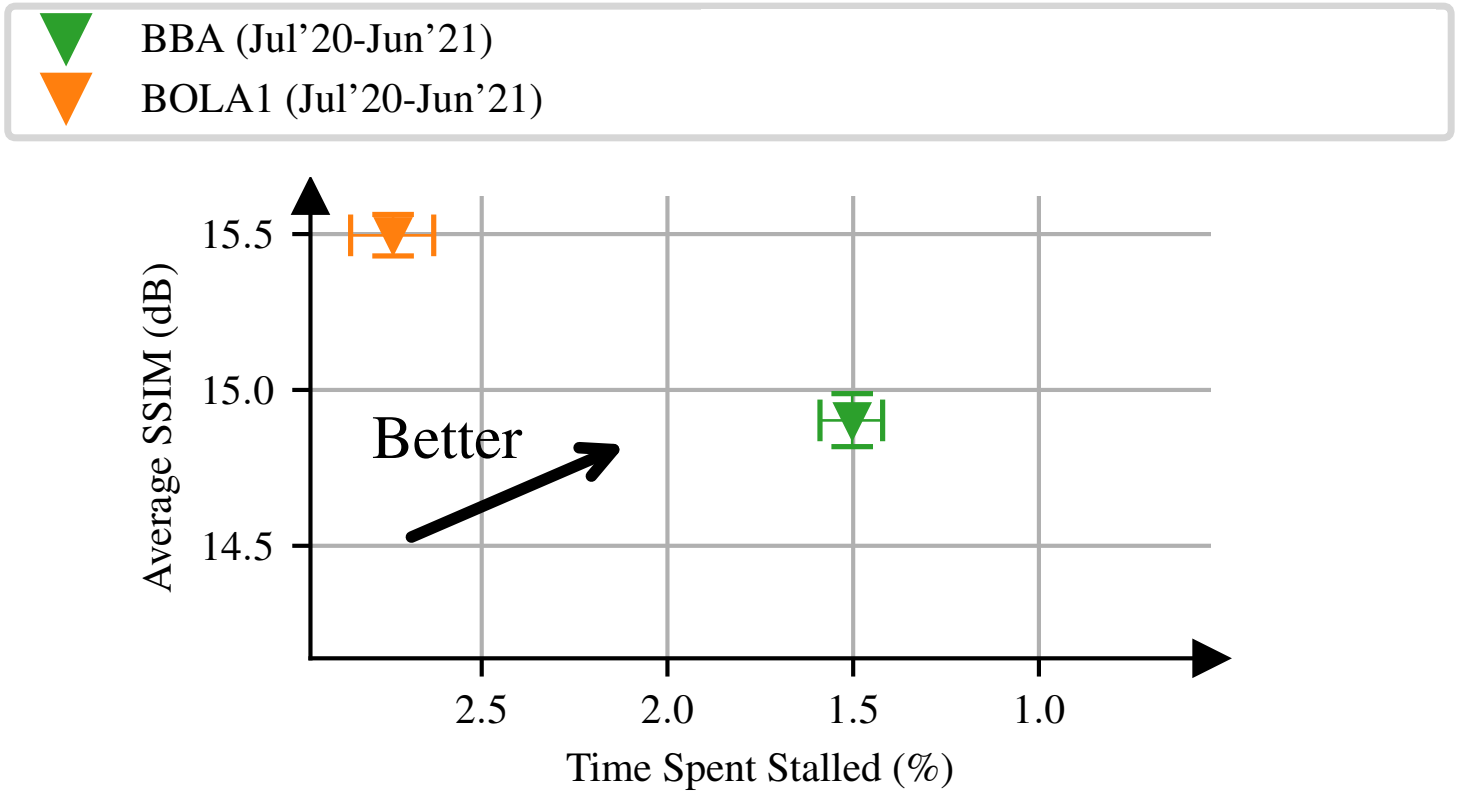
1. (**Invertibility**) $\forall a\colon\ m = f(a, u)$ is invertible.
2. (**Low-rank factorization**) Matrix representation of f has rank r, and $r \leq \dim(\text{trace})$.
3. Traces are collected using sufficient number of **diverse** algorithms (See the paper for the precise statement).

Simulation accuracy

- Source data: The Puffer Dataset with 5 algorithms
- Target algorithm (unseen in training): BBA, BOLA1, BOLA2



Case study: CausalSim in the wild

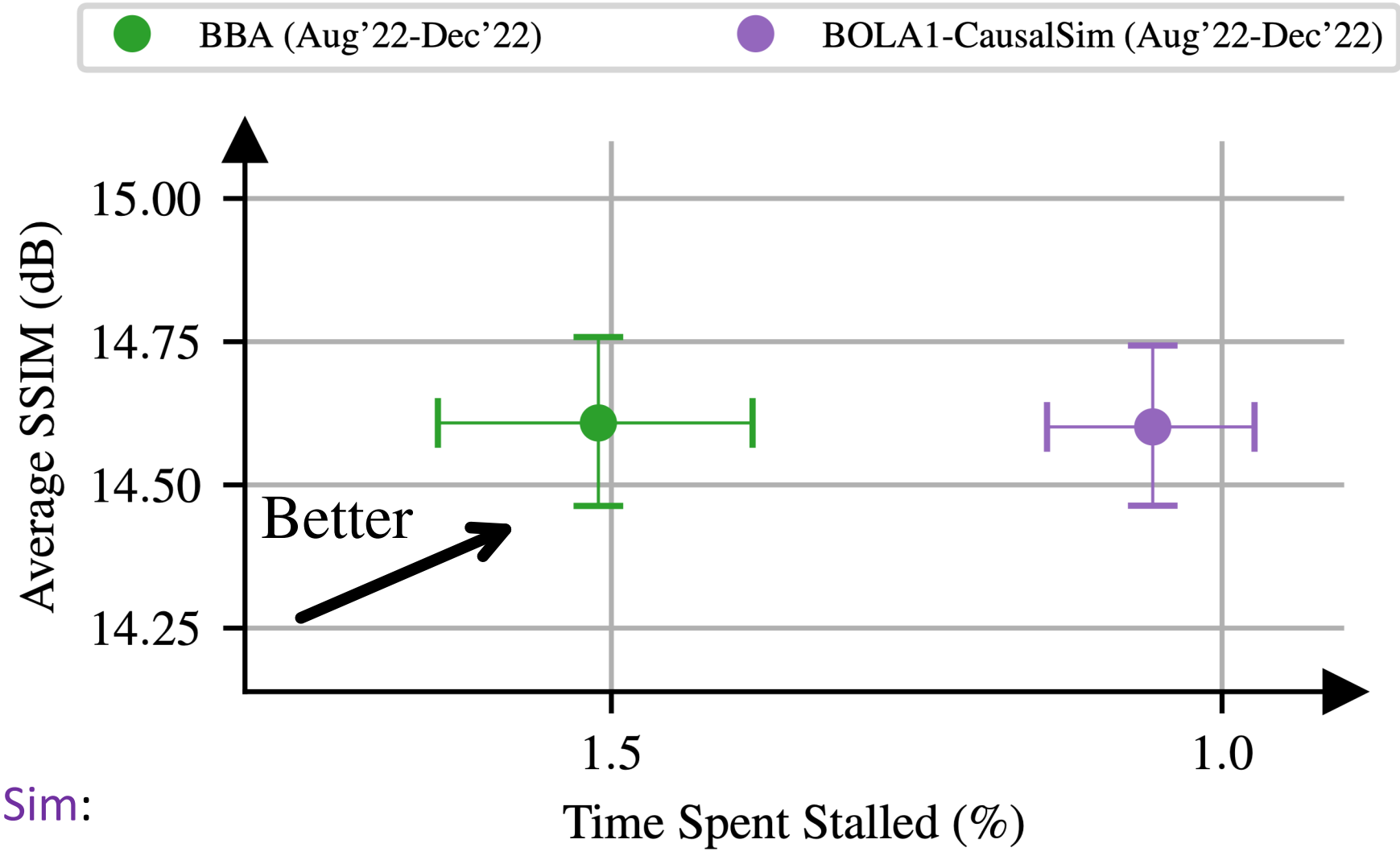


- Can we use CausalSim to improve algorithms?

Case study: CausalSim in the wild

Which one is right?

Case study: CausalSim in the wild



BOLA1-CausalSim:

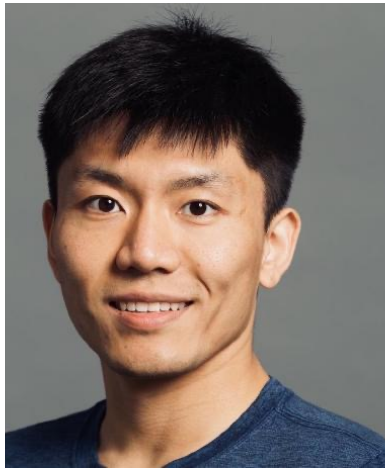
- Deployment: 1.4x less stalling than BBA

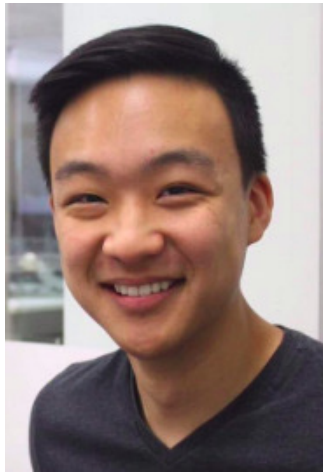
Summary

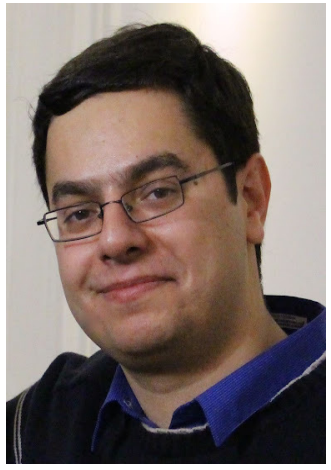
- ❑ The *Exogenous Trace Assumption* causes bias in trace-driven simulation
- ❑ **CausalSim** eliminates bias by modeling the effect of interventions on the trace
- ❑ We can use **CausalSim** to optimize and deploy real-world ABR algorithms

🔗**causalsim**.csail.mit.edu

m3: Accurate Flow-Level Performance Estimation using Machine Learning






Chenning Li*

Arash Nasr-Esfahany*

Kevin Zhao

Kimia Noorbakhsh

Prateesh Goyal

Mohammad Alizadeh

Thomas Anderson

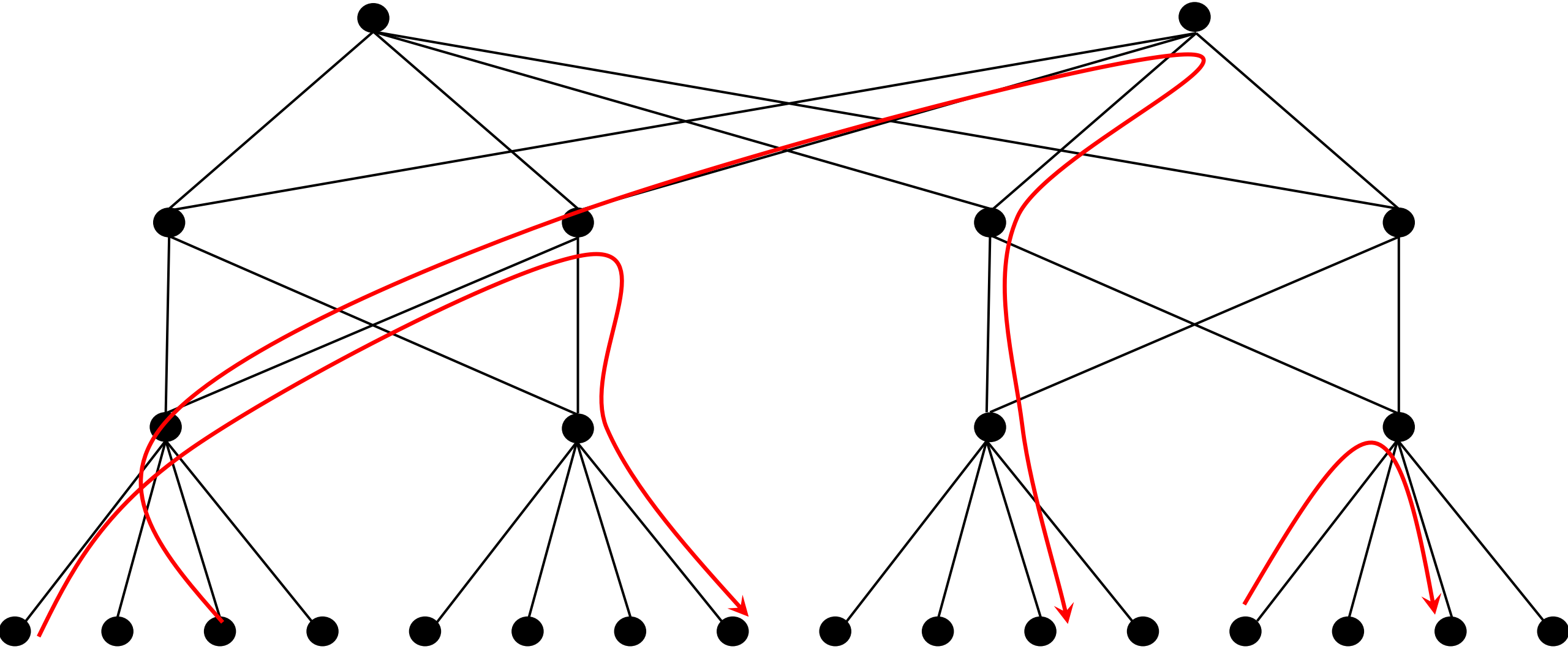
Massachusetts Institute of Technology

UNIVERSITY of WASHINGTON

Microsoft Research

*Equal Contribution

Predicting datacenter network performance

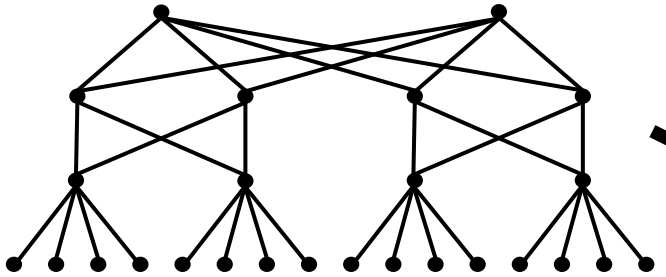


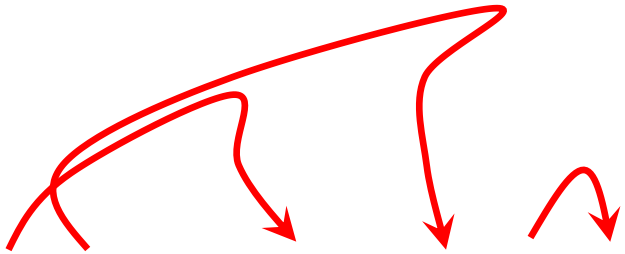
Network topology/configs

Workload

Existing tools for network perf. prediction

Network topology/configs





Workload

ns-3

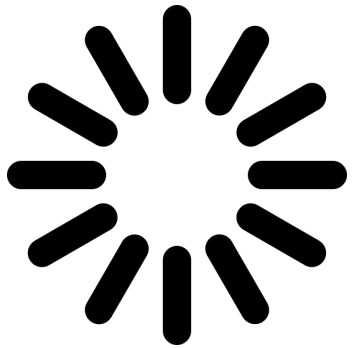
packet loss ratio

buffer occupancies for specific flows at specific times

flow jitter

But ns-3 is very slow…

~6,000 hosts
~ 10 millions of flows



~10 hours for 5s simulated time!

Prior work on faster simulation

- Exact parallel simulation
  - ns-3
  - OMNET++
  - DONS, SIGCOMM'23
- Approximative simulation
  - Parsimon, NSDI'23
  - DeepQueueNet, SIGCOMM'22
  - MimicNet, SIGCOMM'21

All are packet-level simulators → slow for large-scale networks, especially as the network becomes faster

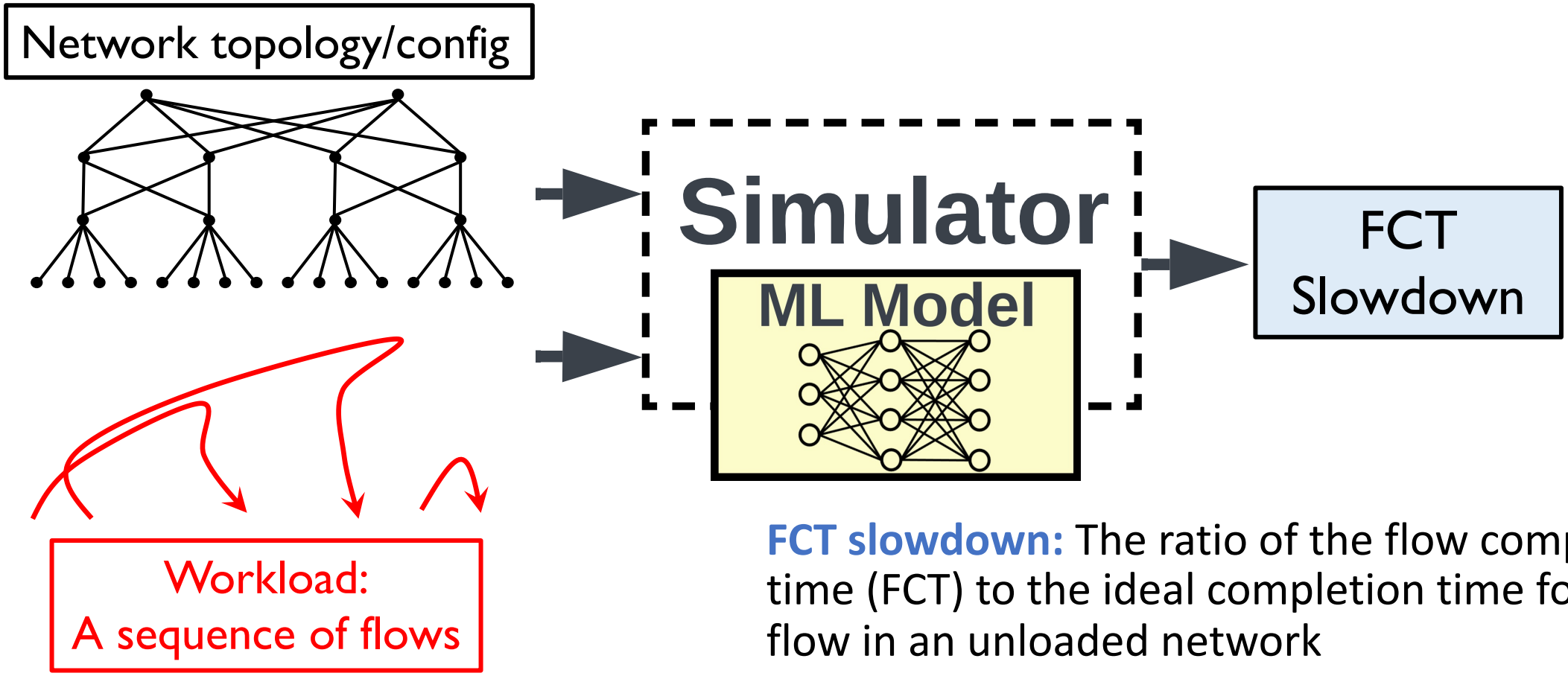
Do we always need packet-level simulation?

- Many questions are about aggregate performance statistics
  - Tail latency
  - Throughput
  - Loss rates
  - Short & long flow completion time

- How would workload/design changes impact these statistics?

Can we elevate the abstraction level of answering such questions without compromising fidelity?

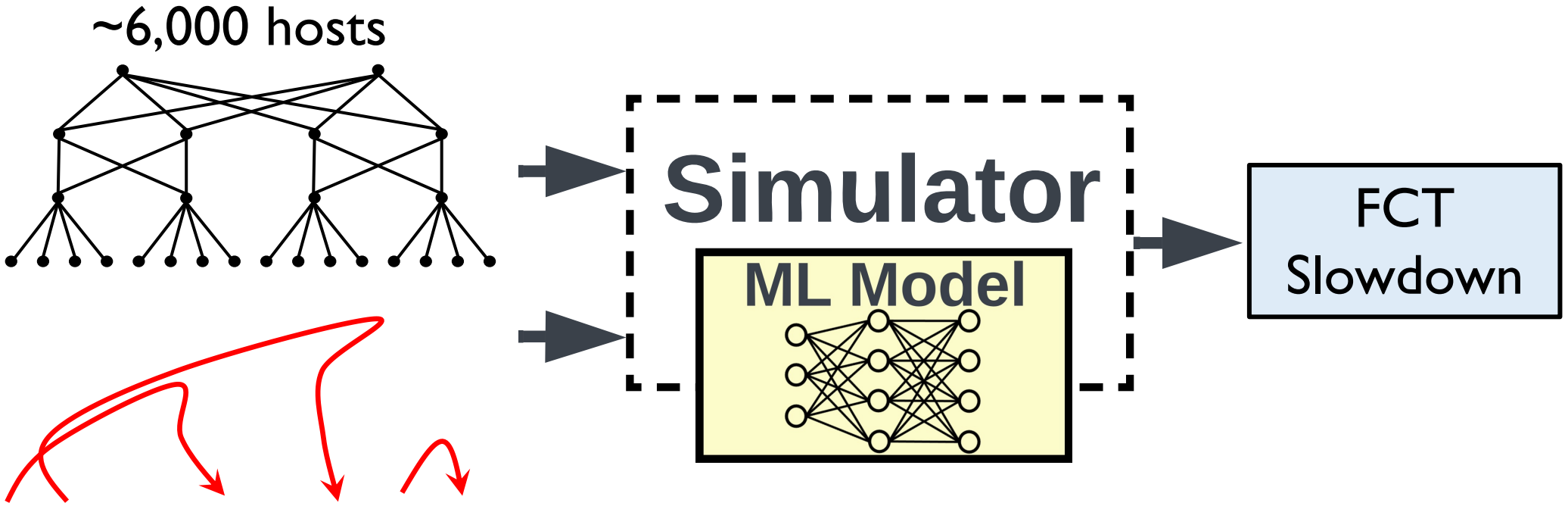
Abstract the network simulator as a function



FCT slowdown: The ratio of the flow completion time (FCT) to the ideal completion time for a flow in an unloaded network

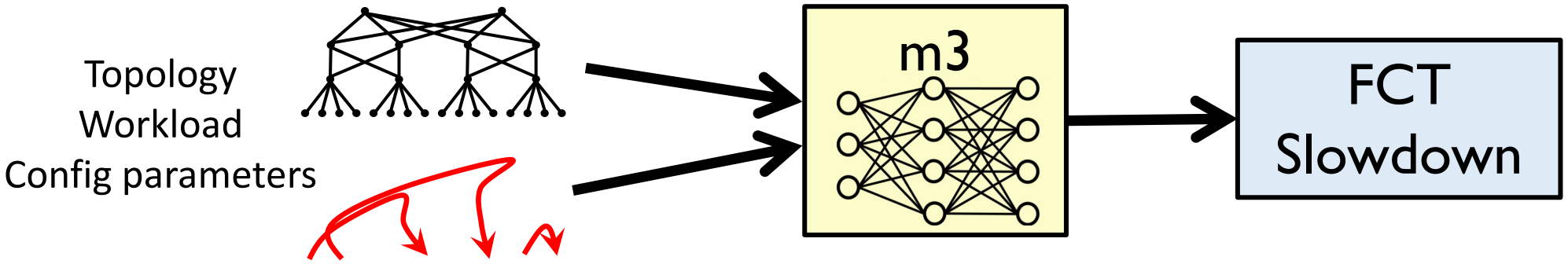
Use supervised learning to learn the end-to-end function?

The simulator is a complex function



- ✗ Hard to represent the function in a compact way
- ✗ Slow to generate the training dataset for the ML model

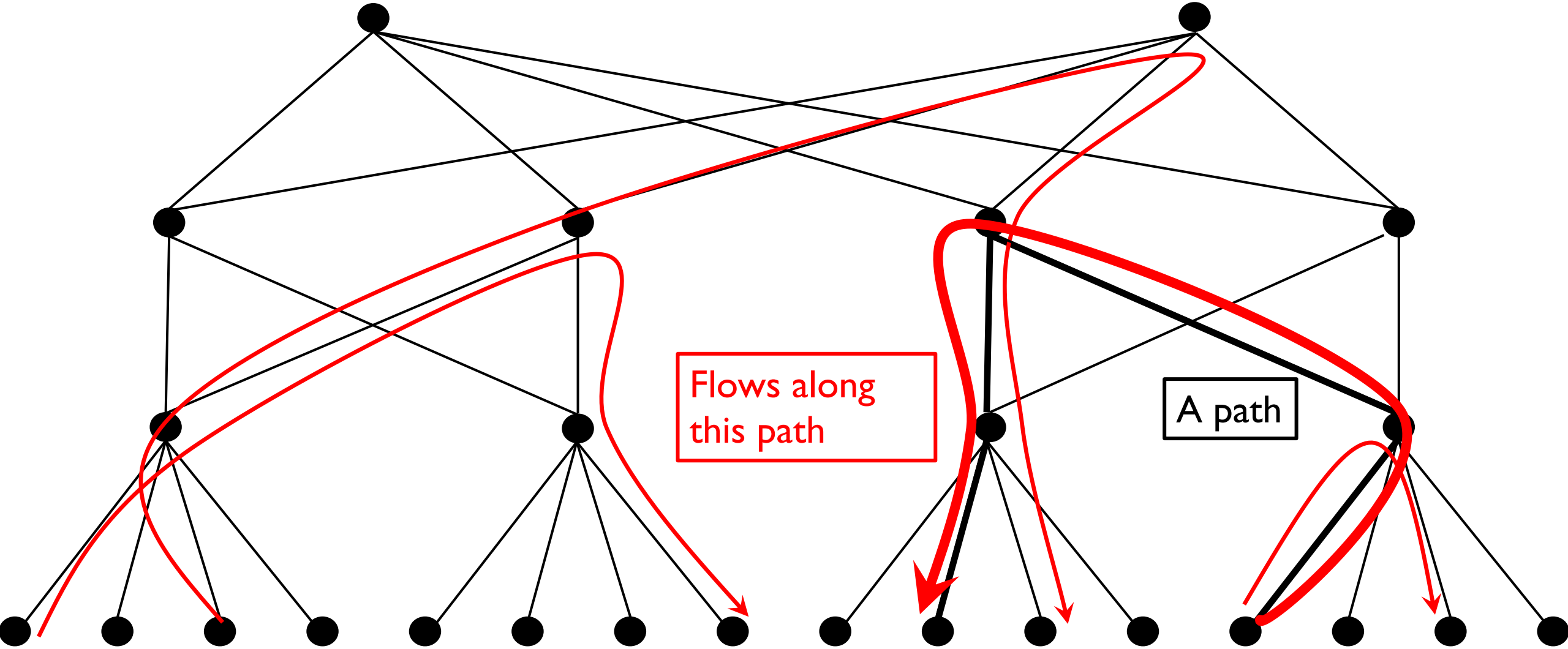
m3: Accurate flow-level performance estimation using machine learning



m3: ~1200X speedup with only ~10% estimation error in a 384-rack, 6144-host topology (vs. ns-3)

From ~10 hours (ns-3) to less than 1 min (m3)

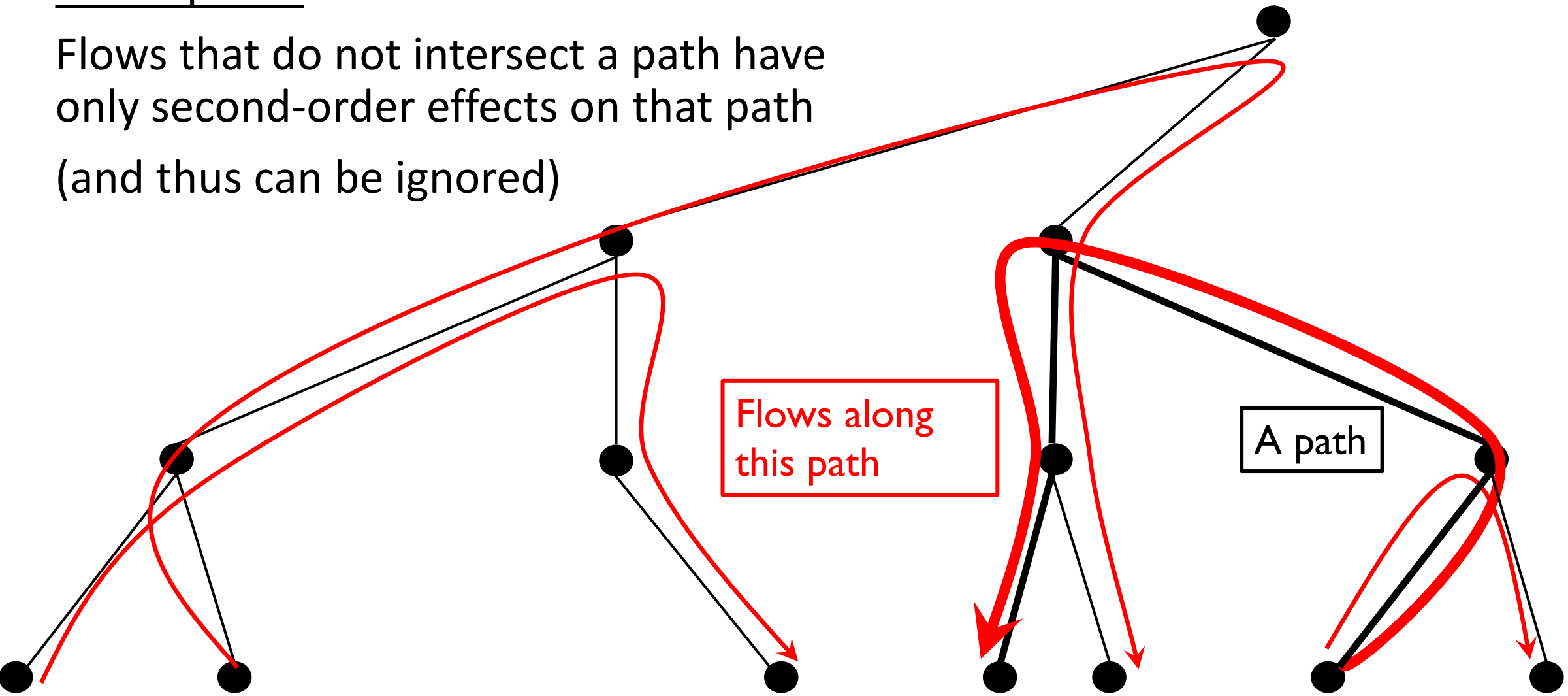
Building block: A path simulation



Assumption:

Flows that do not intersect a path have only second-order effects on that path

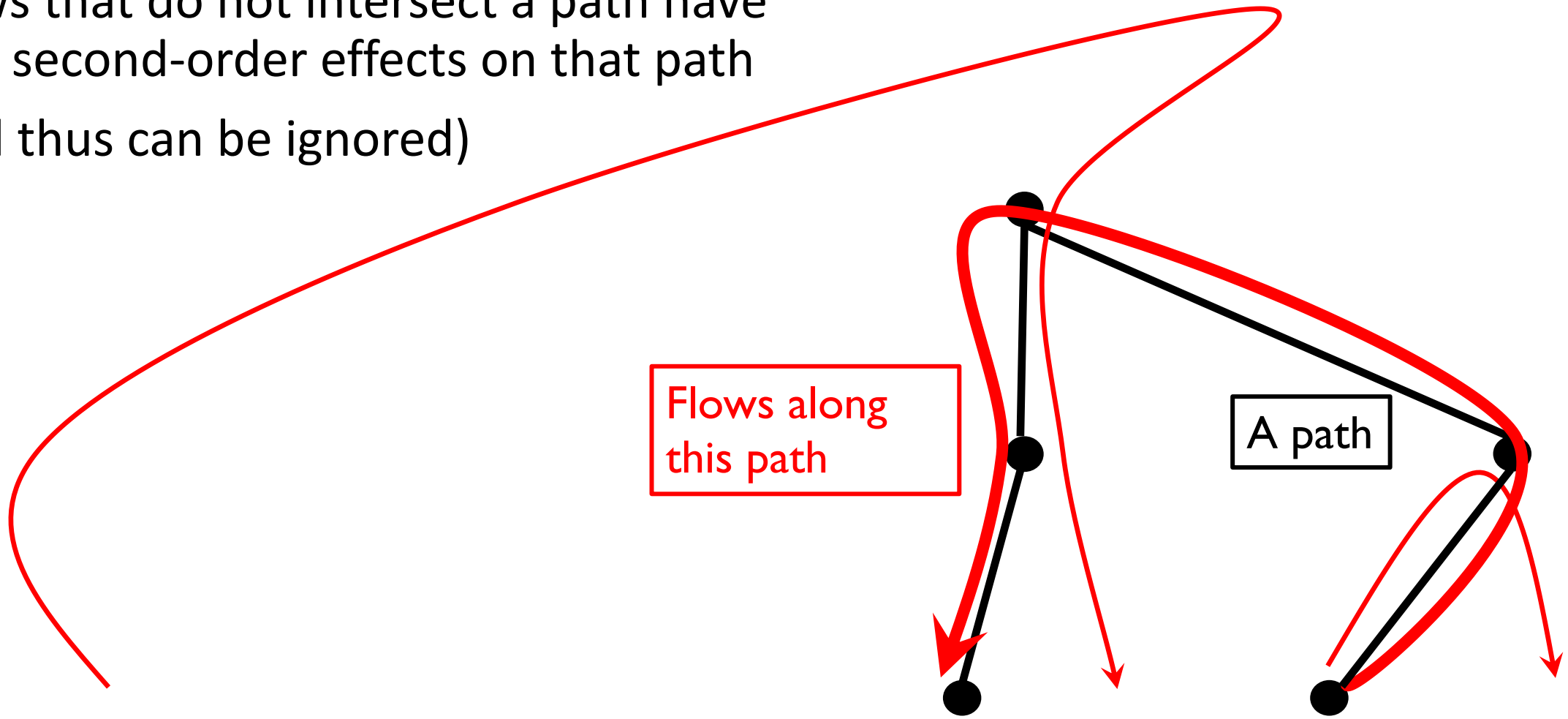
(and thus can be ignored)



Assumption:

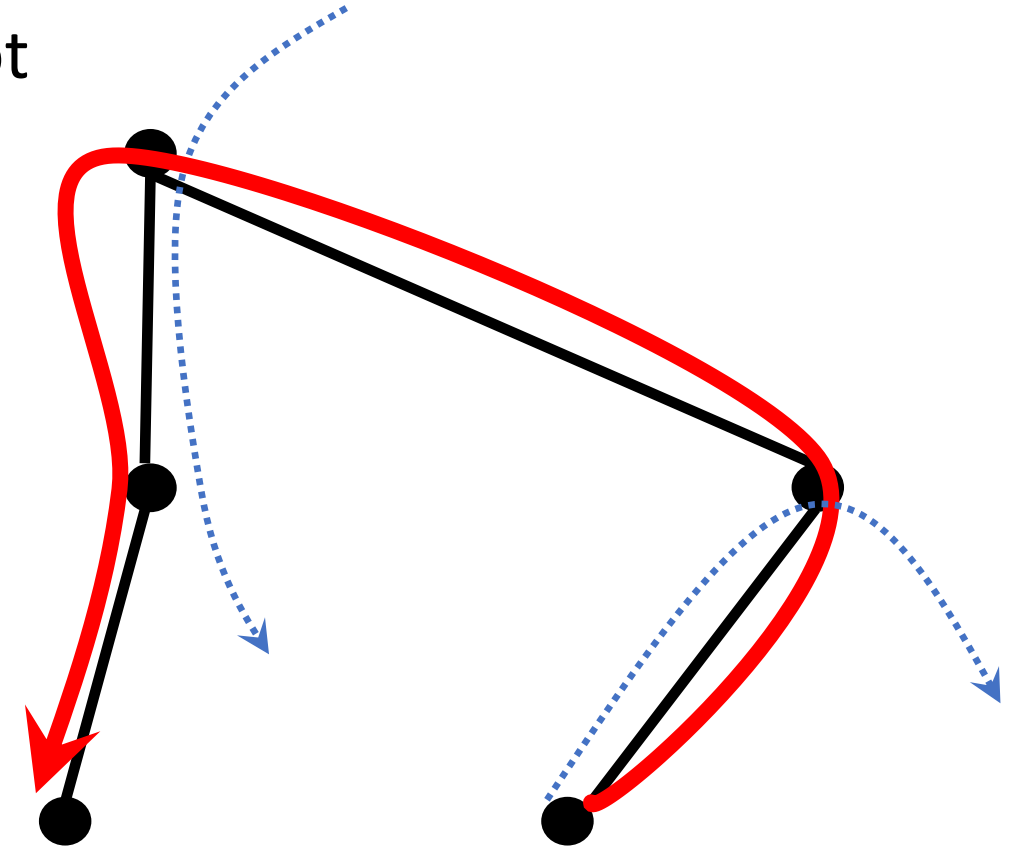
Flows that do not intersect a path have only second-order effects on that path

(and thus can be ignored)

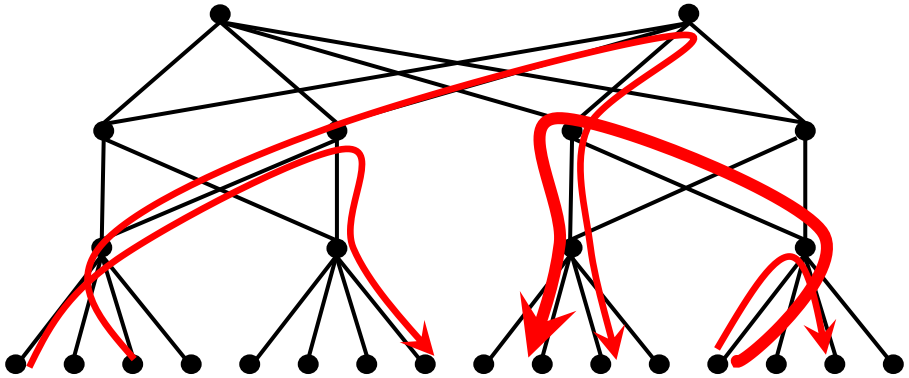


Path-level simulation

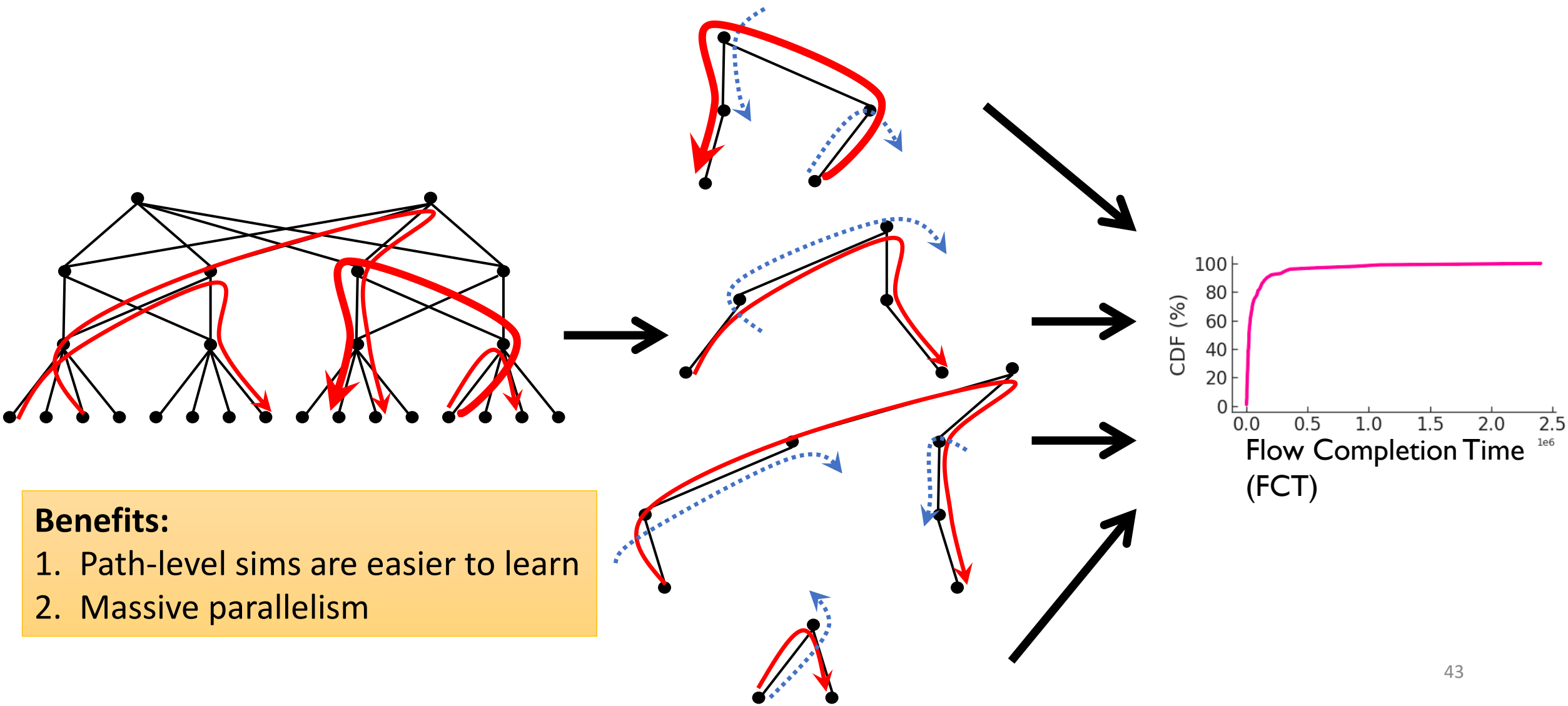
- Select one path and prune flows that do not intersect it
  - Foreground flows follow the entire path
  - Background flows intersect only certain links along this path, not all of them
- Goal: estimate the FCT slowdown distribution of the foreground flows



Path decomposition & aggregation



Path decomposition & aggregation



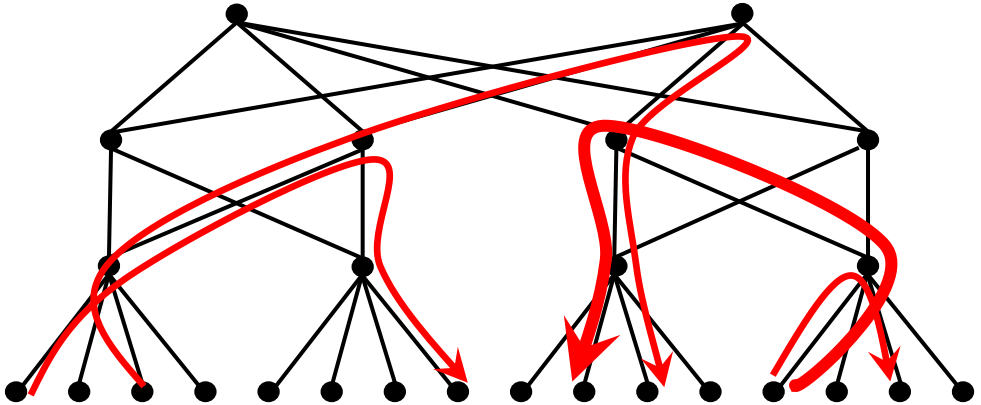
Benefits:

1. Path-level sims are easier to learn
2. Massive parallelism

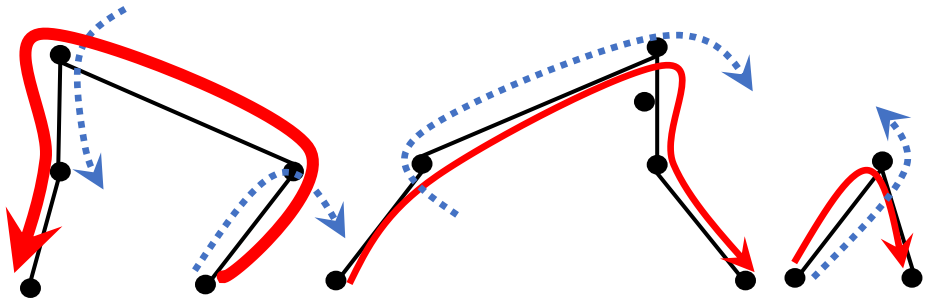
Is path decomposition accurate?

Experiment

- Three scenarios on a 32-rack 256-host topology with different workloads
- Randomly select 500 paths for each scenario
- Compare the tail latency (P99 FCT slowdown) of foreground flows for each path

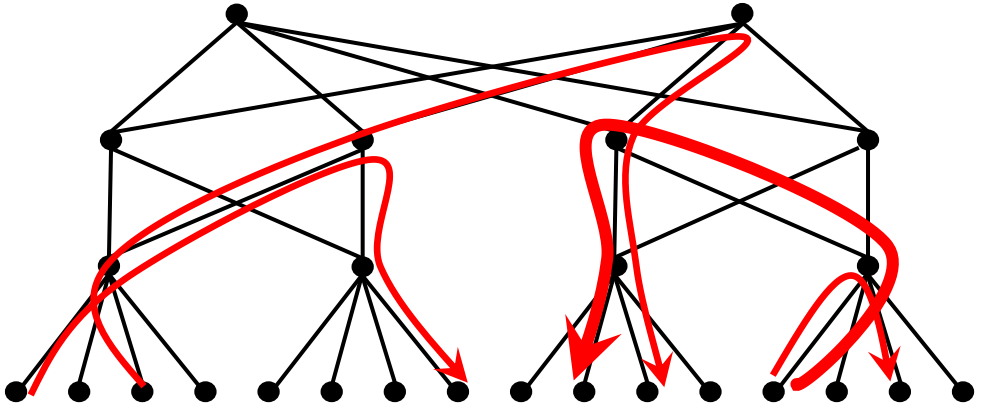


[ns-3]

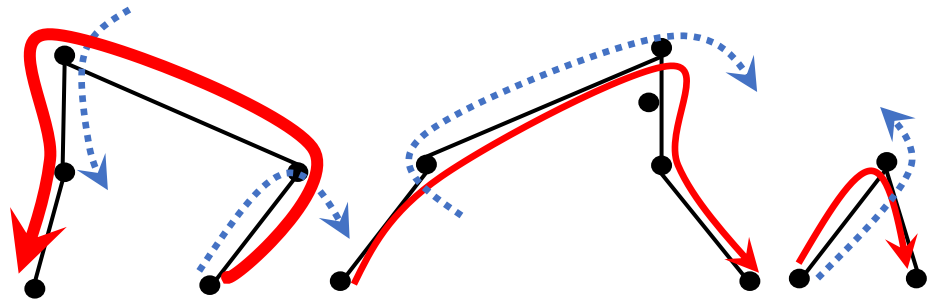


[ns-3-path]

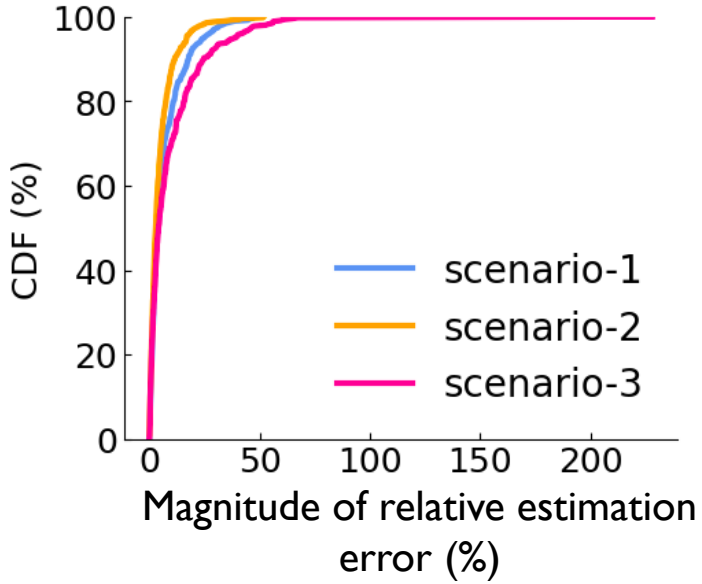
Path-level sim is accurate but still slow



[ns-3]

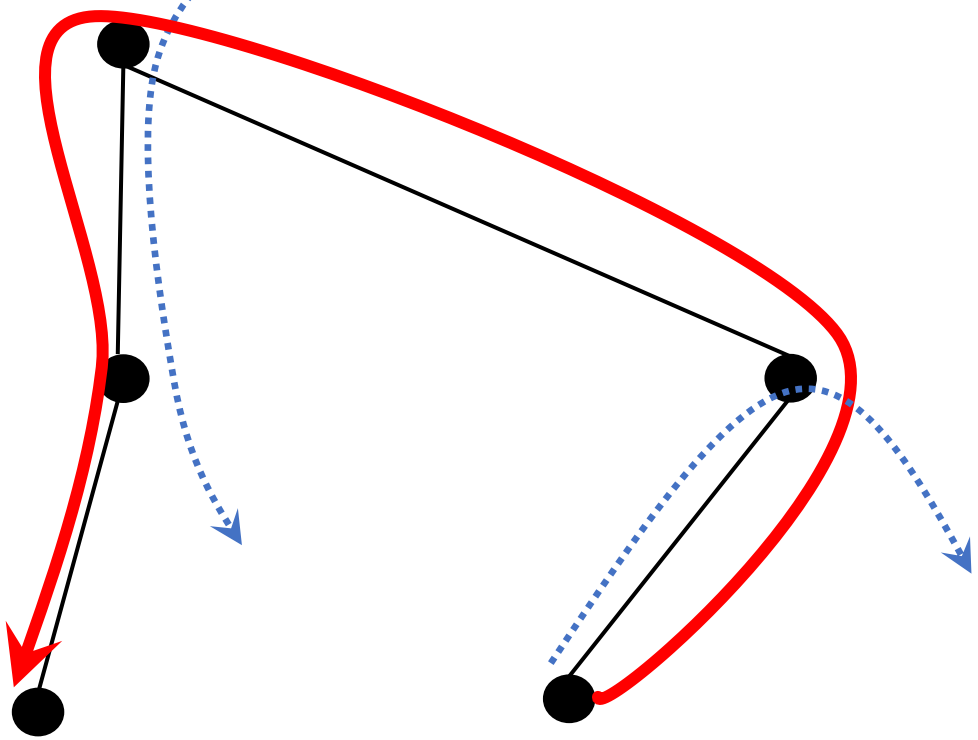


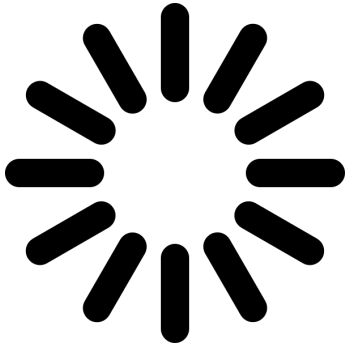
[ns-3-path]



Scenario	Median relative estimation error	Overall speedup
1	-0.83%	3.6X
2	-2.13%	5.4X
3	-5.91%	14.2X

Estimate path-level perf. with ML





~ 10 K foreground flows
~ 1 million background flows

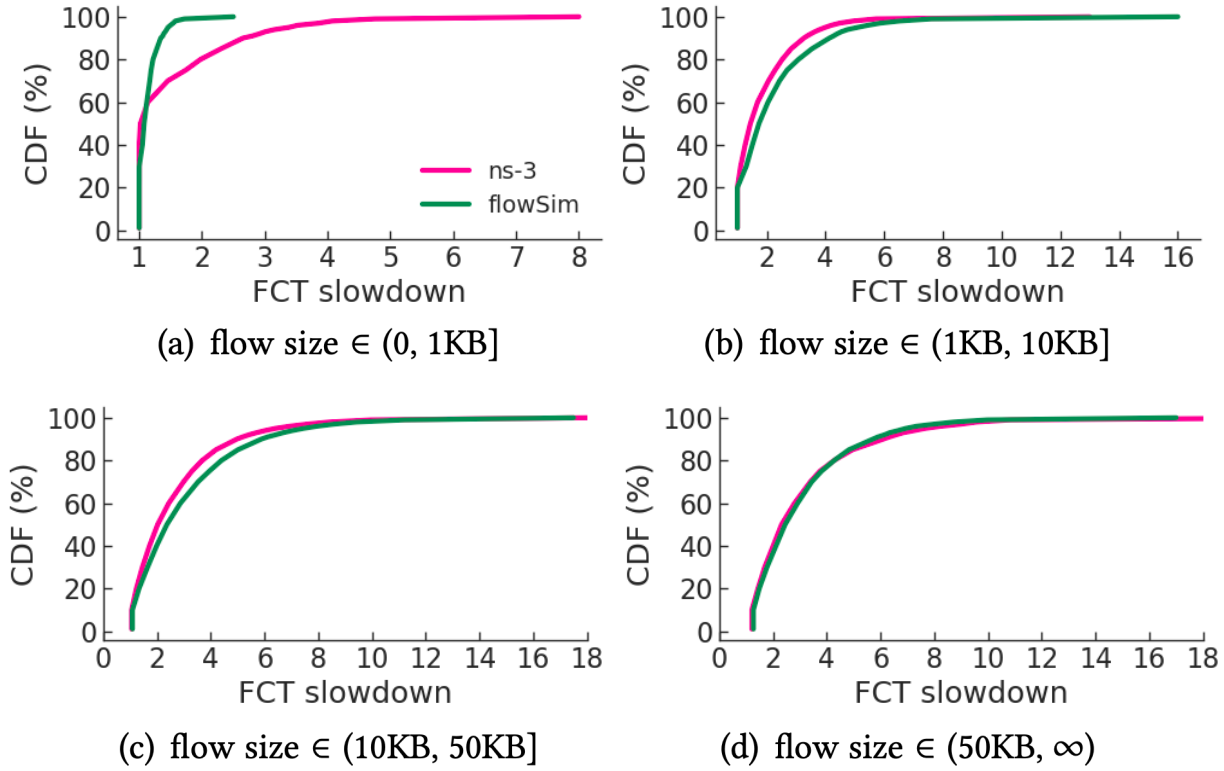
m3 uses a flow-level simulator to extract compact features for its ML model

Flow-level simulator (flowSim)

- Max-min flow-level simulation
  - No packets – A "fluid" flow *i* sending at time-varying rates $r_i(t)$
    - Flows transmit at the max-min fair rates [1] (approximates congestion control)
  - Events: flow arrivals and departures
  - No queues!

Fast: <1 sec for a path-level sim with 1 million flows

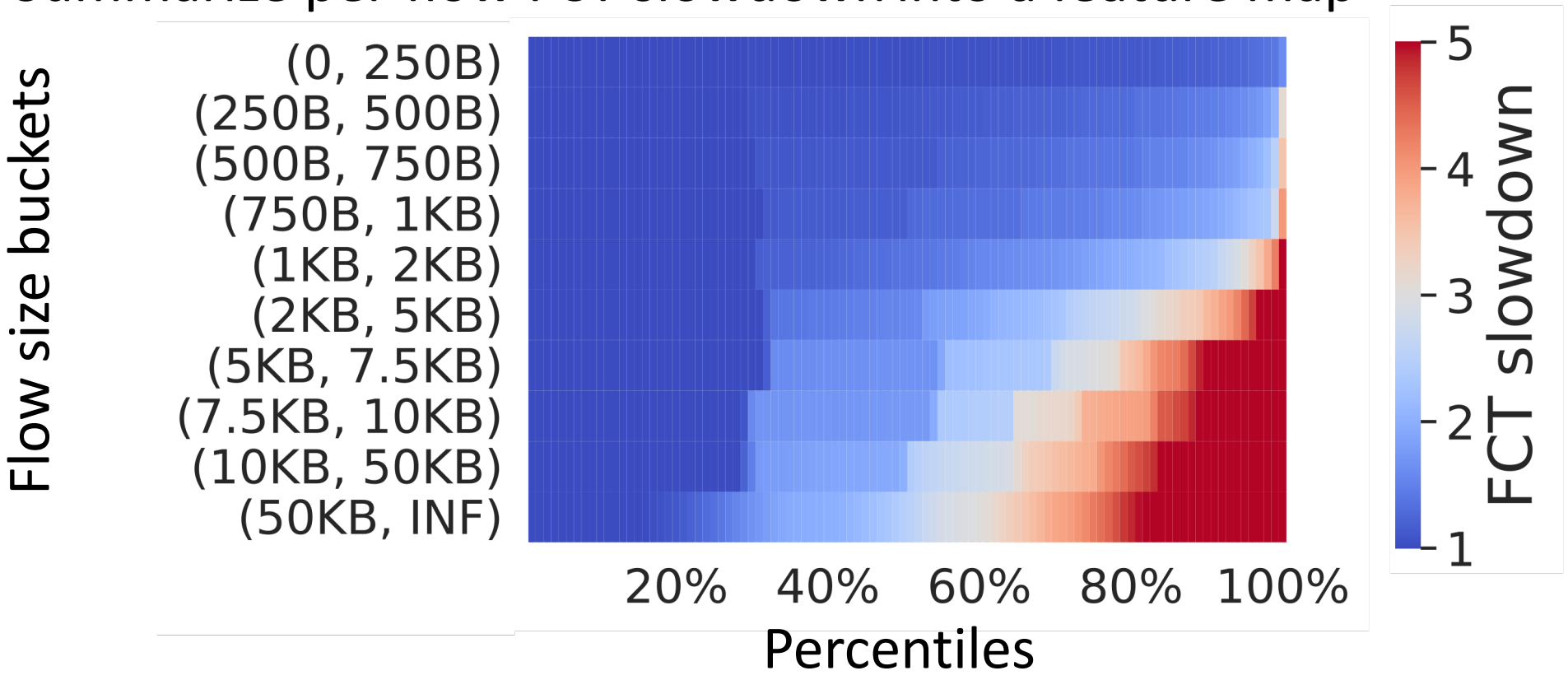
flowSim is fast but inaccurate



Not accurate, esp. for short flows

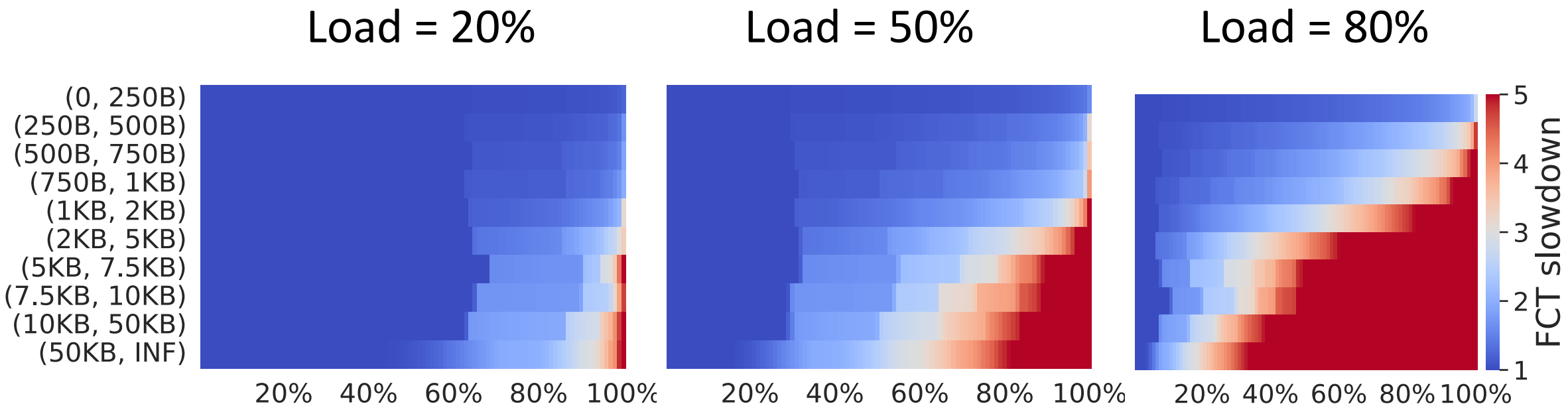
Use flowSim to extract a feature map

- Run path-level simulation on flowSim
- Summarize per-flow FCT slowdown into a feature map



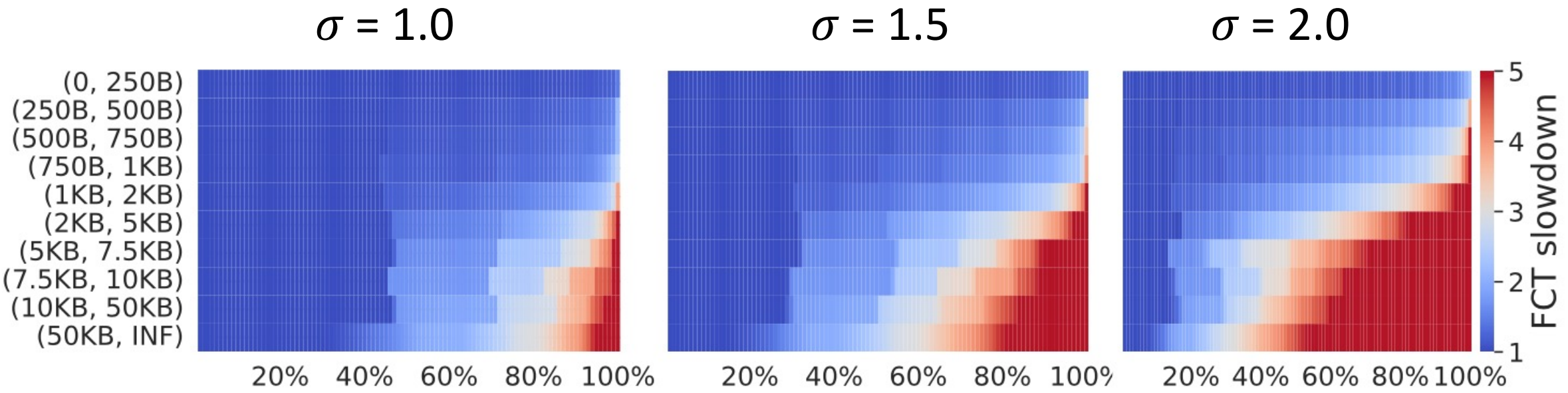
m3's feature map captures workload characteristics

Example: Different levels of load for the simulated network



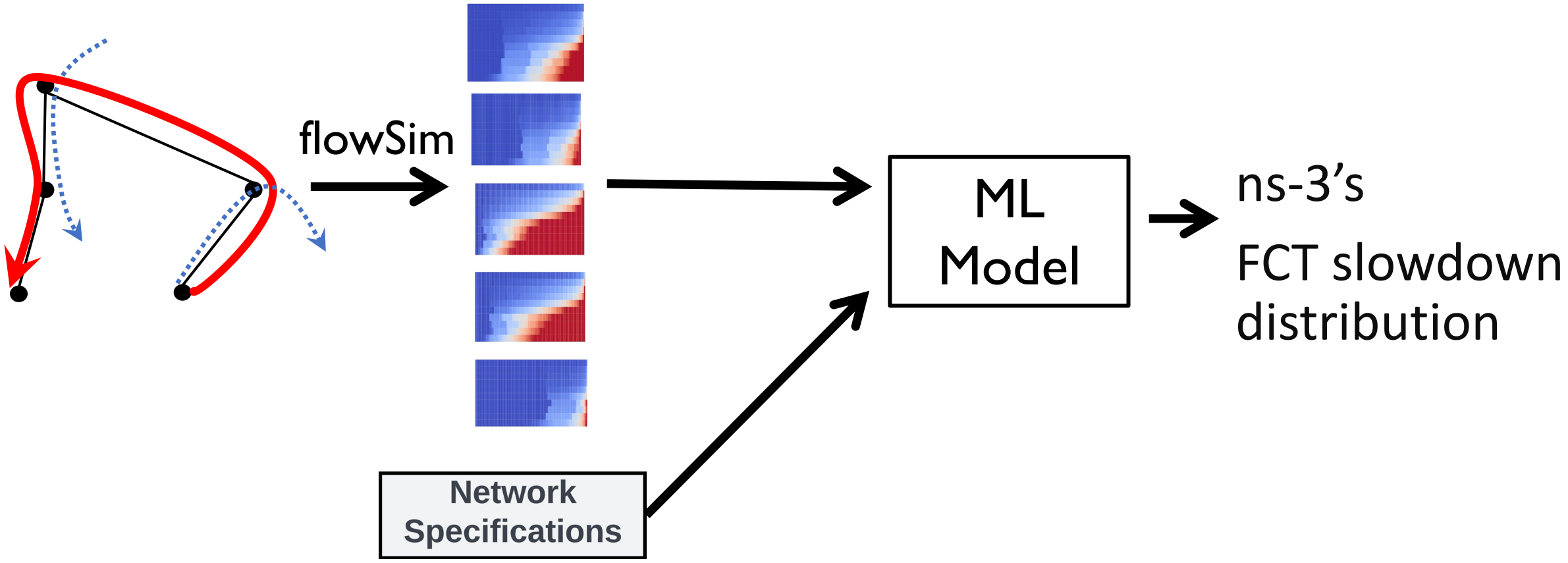
m3's feature map captures workload characteristics

Example: Different levels of burstiness of the workload



(higher σ → more bursty)

m3 uses ML to correct flowSim



Buffer size, initial window size, different CCAs as a one-hot vector, and CCA parameters

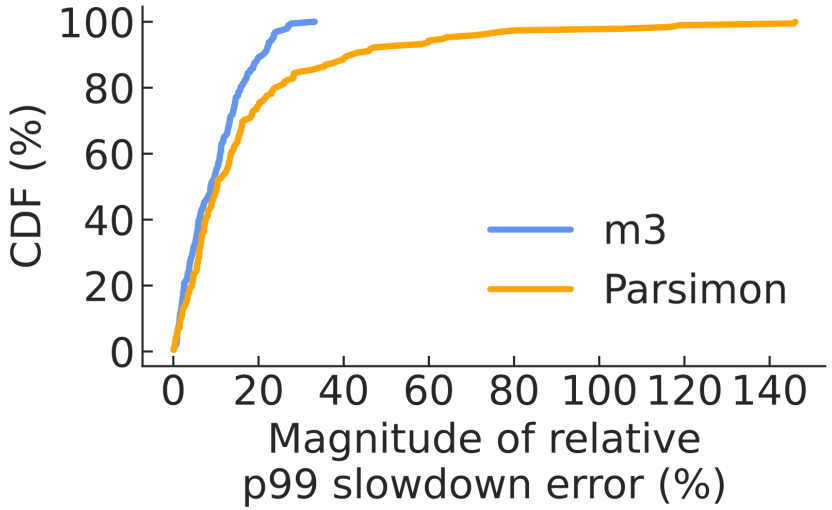
Training

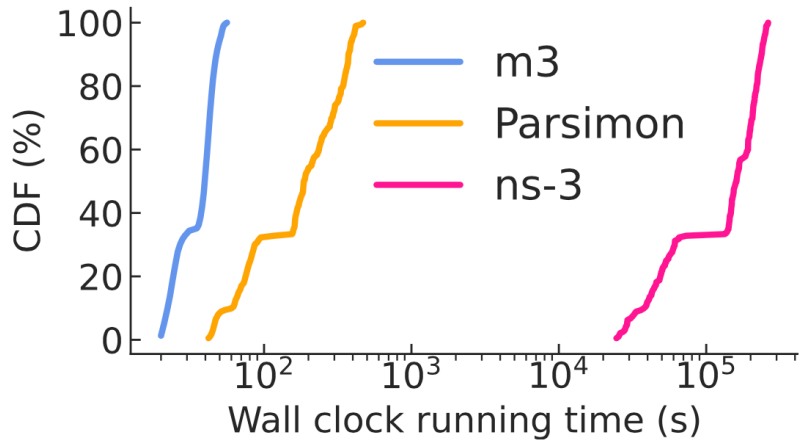
- Train on diverse synthetic scenarios
  - Path topology: 2-hop, 4-hop, 6-hop parking-lot topology
  - Flow size distribution: Pareto, Lognormal, Exponential, Gaussian
  - Flow inter-arrival time distribution: burstiness from low (σ = 1.0) to high (σ = 2.0)
  - Randomly chosen network config: Buffer size, initial window size, CC from DCTCP, TIMELY, DCQCN, HPCC, etc.

- We train the model once and use it directly for inference
  - m3 generalizes to unseen workloads, topologies, and network configurations

Accuracy and run-time evaluation

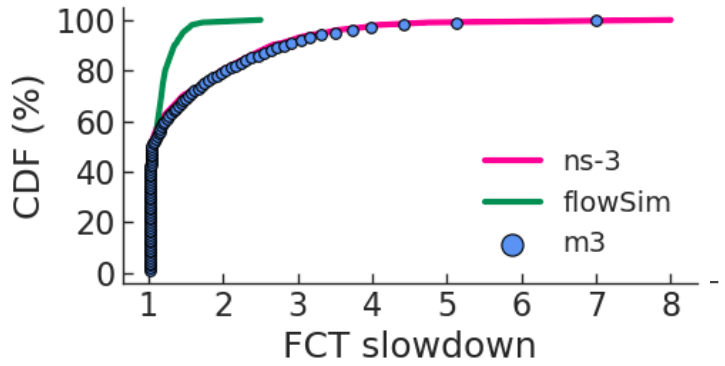
- Various production workloads on a sampled Meta's data center fabric
- Baseline: Parsimon[1]
- m3 reduce Parsimon's mean error from 18.3% to 9.9%
- m3 is fast
  - 5.7× speed-up over Parsimon
  - 3 orders faster than ns-3



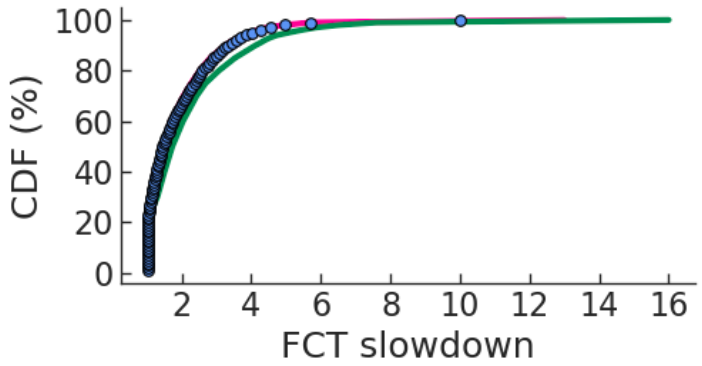


[1] Zhao, Kevin, et al. "Scalable tail latency estimation for data center networks." In proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI). 2023.

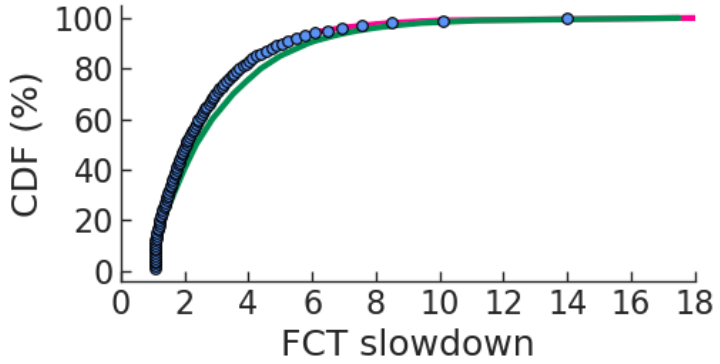
Example Scenario



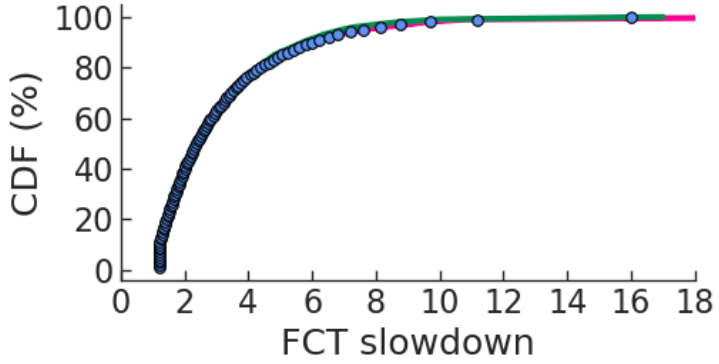
(a) flow size ∈ (0, 1KB]



(b) flow size ∈ (1KB, 10KB]



(c) flow size ∈ (10KB, 50KB]

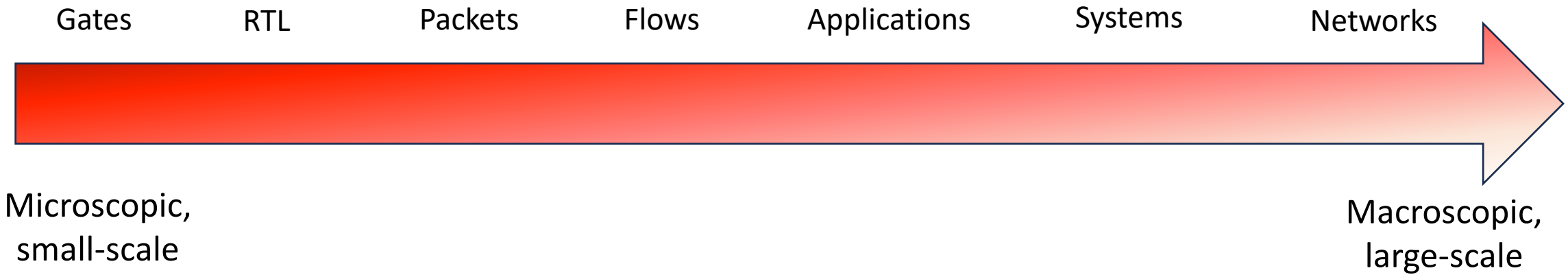


(d) flow size ∈ (50KB, ∞)

Summary

- **m3** is fast and accurate for estimating flow-level performance
- Two key ideas:
  - Path decomposition
  - Workload featurization via flow-level simulation
- **General principle:** Learn to map from simple reference system to real system

🔗https://github.com/netiken/m3



ML can significantly change how we model and simulate network systems (esp. at the large-scale macroscopic level)

- Many challenges:
  - Abstractions for learning system dynamics
  - Handling partial observability & building systems for observability
  - Robustness
  - Interpretability

What m3 does and does not do

- m3 assumes static routes known in advance.
  - This assumption breaks in the case of dynamic routing strategies like packet-spraying or flowlets
- m3 predicts the distribution of FCT slowdown for various size buckets
  - Not per-flow or per-packet performance
- m3 learns the impact of network parameters from training examples
  - cannot predict unseen protocols

Large-scale simulation

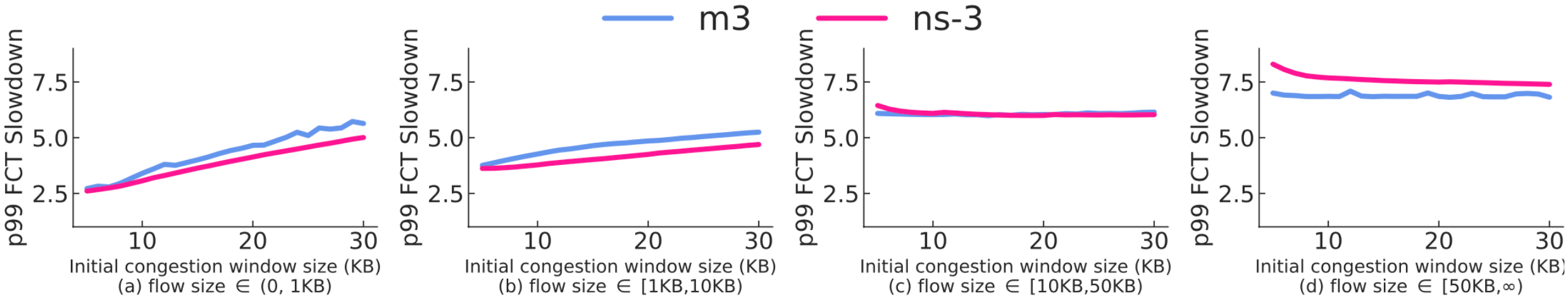
- Meta Production Workload
- A 384-racks, 6,144-host Meta's data center fabric
- Baseline: Parsimon

Init. Window	Methods	p99	Error	Time	Speedup
10KB	ns-3	2.05	-	13.5h	-
	Parsimon	4.29	+109%	1m29s	546×
	m3	2.10	+2.44%	37s	1314×
18KB	ns-3	2.44	-	11.9h	-
	Parsimon	2.73	+11.9%	1m24s	510×
	m3	2.30	-5.74%	40s	1071×

➢ m3 is faster, more accurate, and robust

m3 enables rapid design-space exploration

Example: Effect of initial window size on p99 FCT slowdown for different classes of flows



The mean runtimes of each configuration in the plot

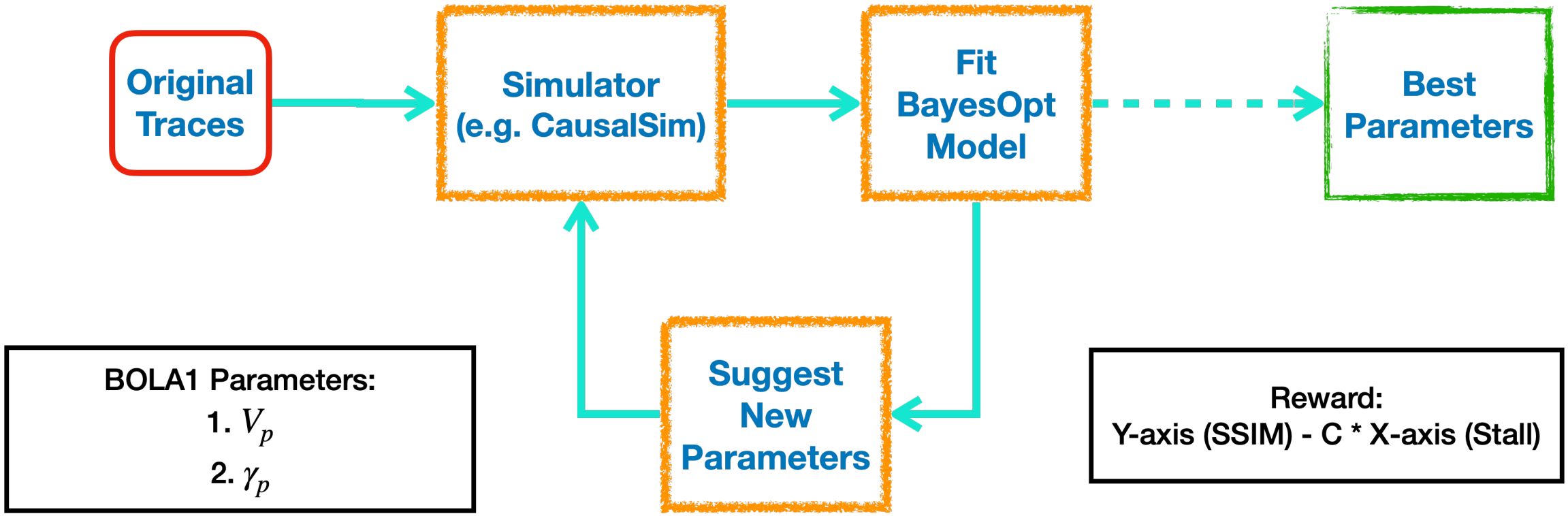
- m3: 25.2 seconds
- Ns-3: 8 hours

Vision

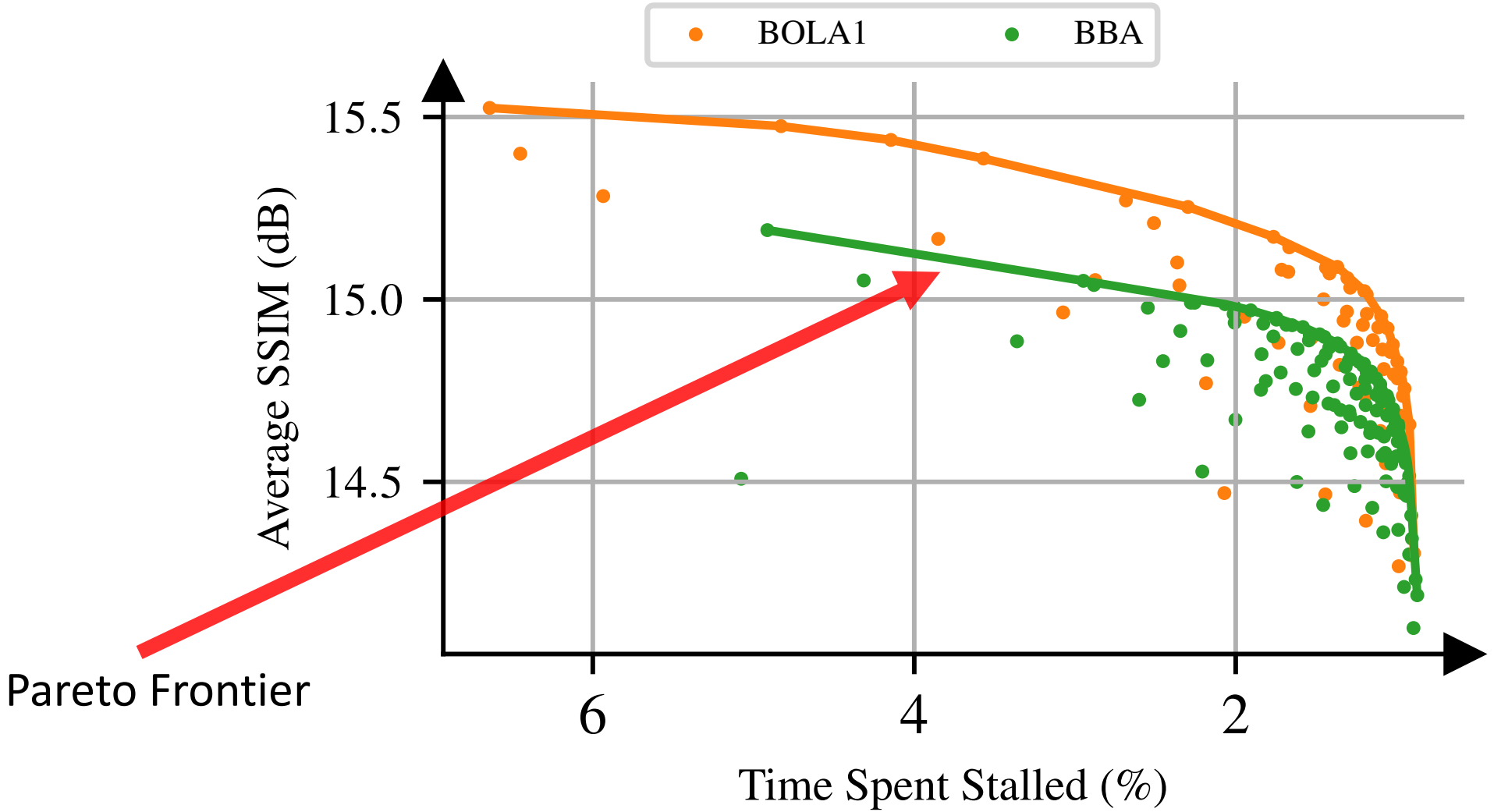
- Networks produce massive amounts of data
- Relatively easy to add instrumentation, run active experiments, …
- Goal: Learn high-level network dynamics from observational data

Case Study: CausalSim in the Wild

We evaluated 300+ different variants in less than 1 day

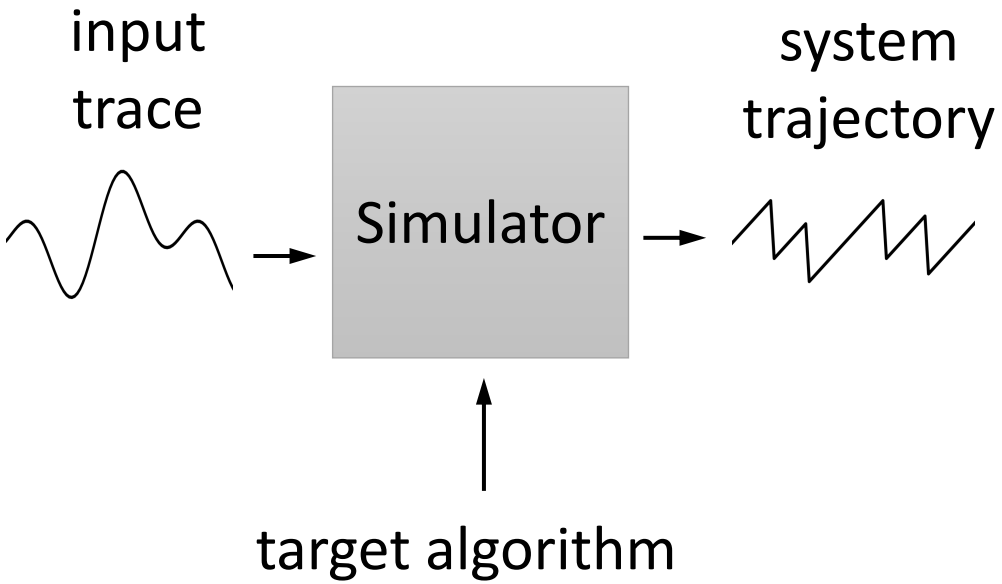


Case Study: CausalSim in the Wild

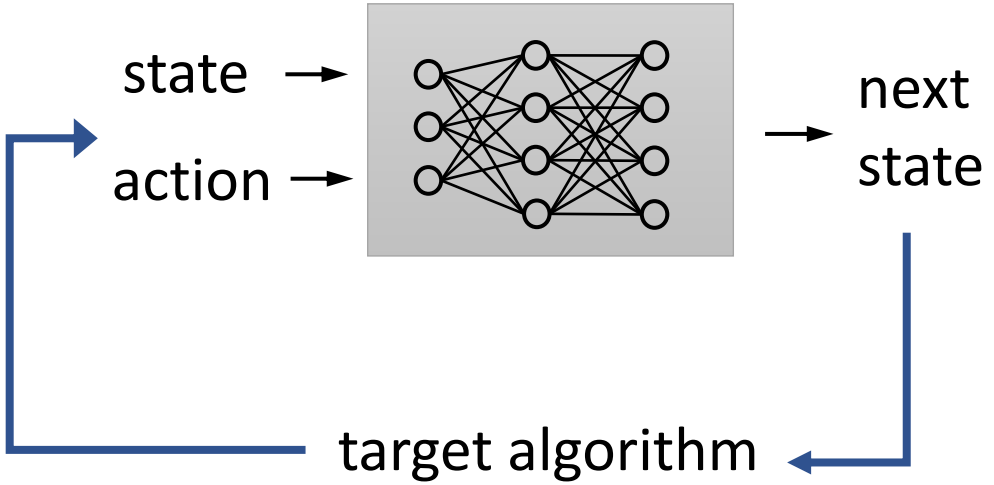


Standard Approaches

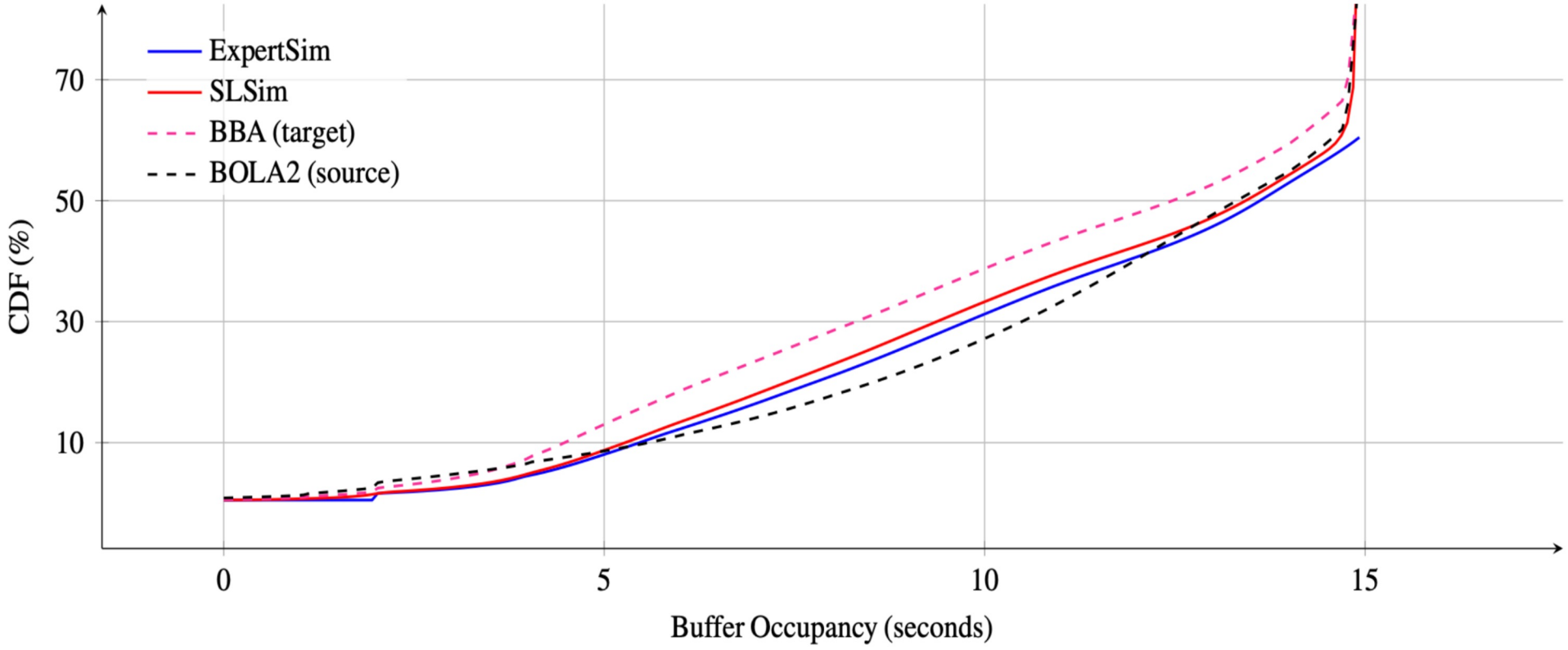
Expert-designed Simulators



Supervised Learning



How accurate is trace-driven simulation?



CausalSim: Unbiased Trace-Driven Simulation