Learning-Based Control of Networked Systems

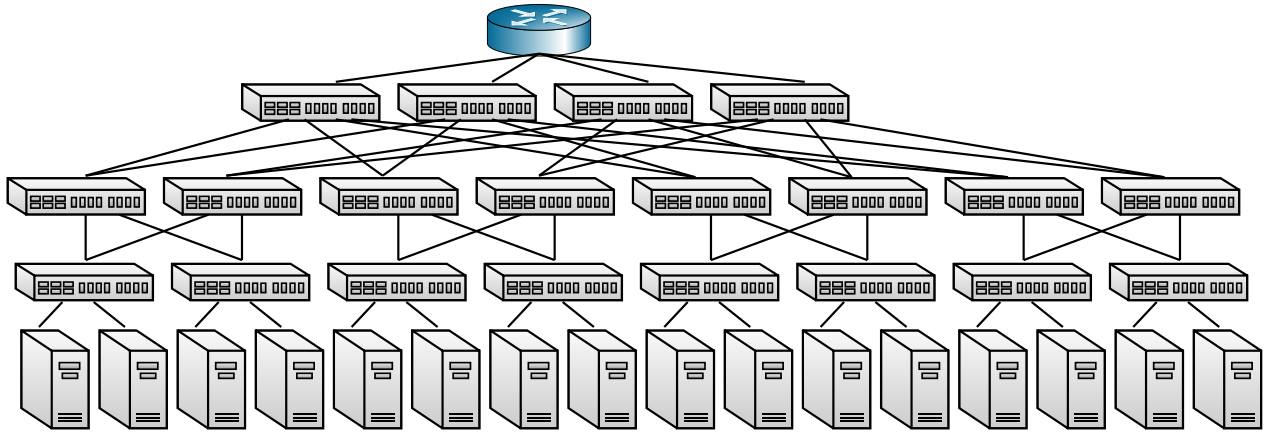
Visit to Université Paris 1 Panthéon-Sorbonne
May 12, 2026

Kim Hammar
Department of Electrical and Electronic Engineering
The University of Melbourne
kim.hammar@unimelb.edu.au

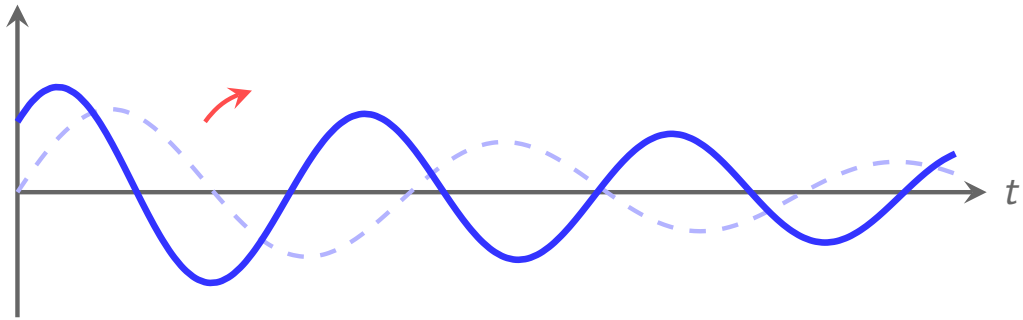
THE UNIVERSITY OF MELBOURNE

Networked Systems

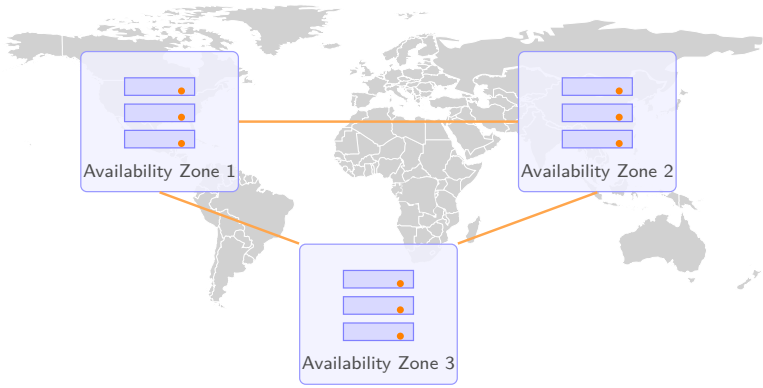
Complex



Dynamic



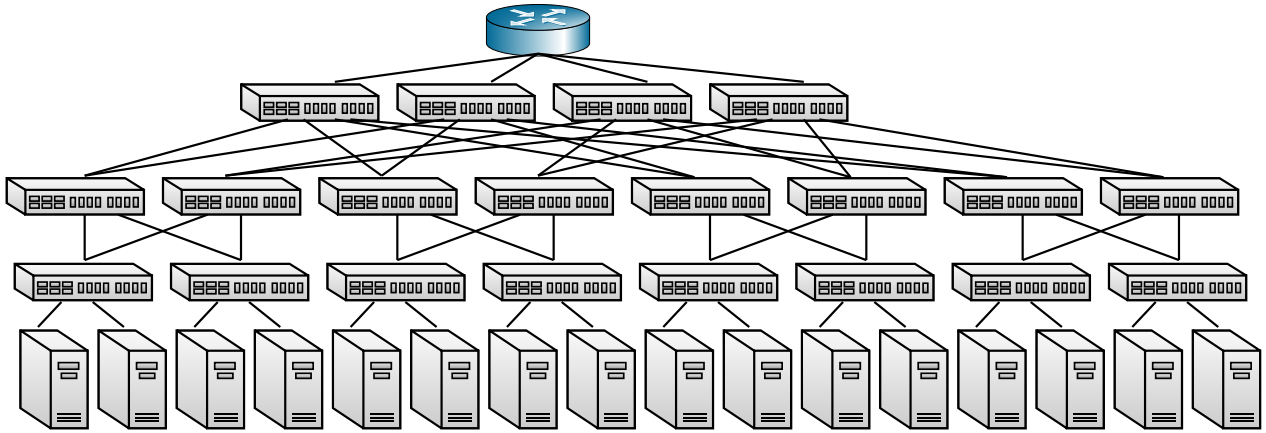
Distributed



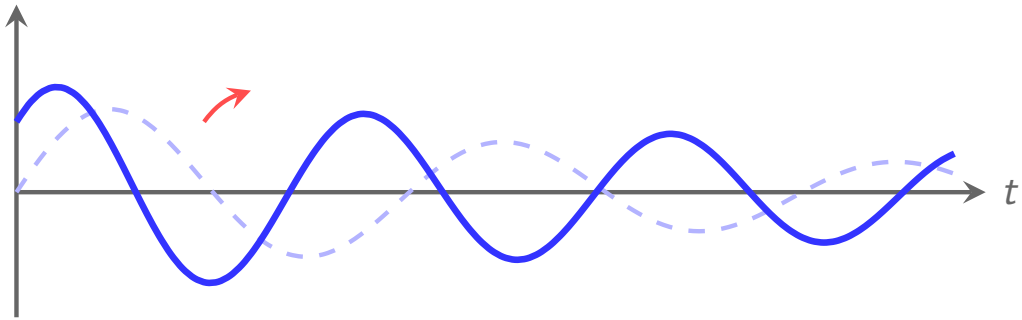
Networked Systems

Complex

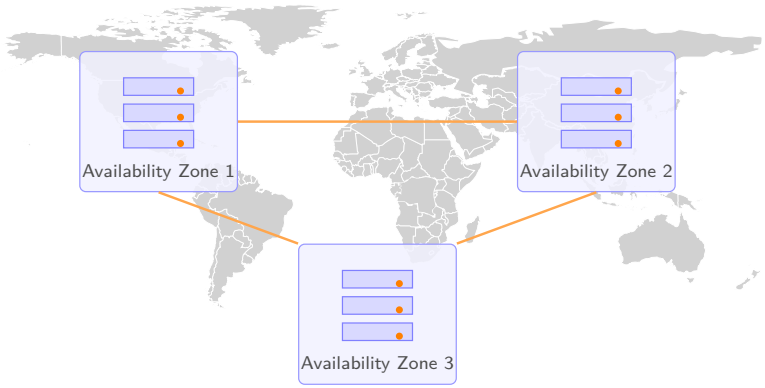
Automation



Dynamic



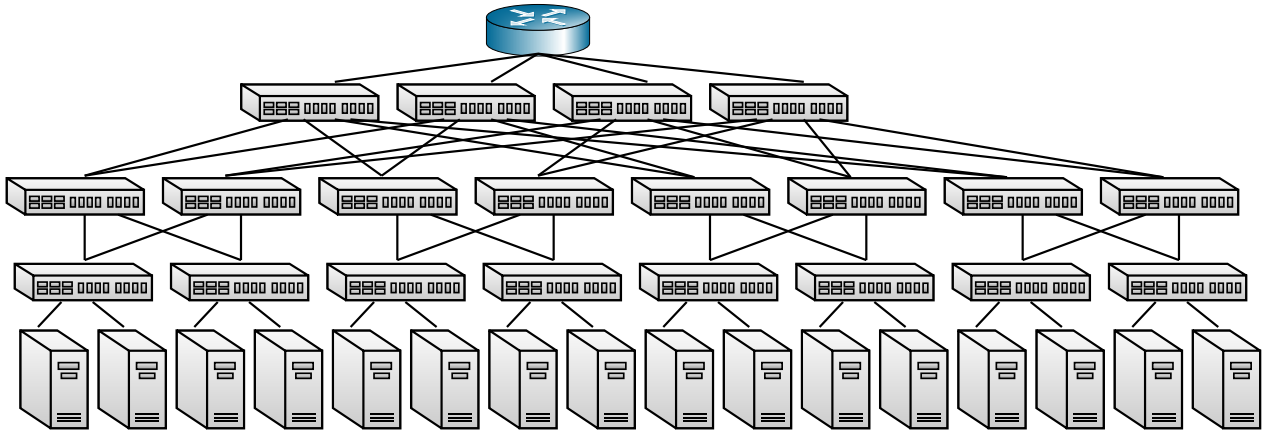
Distributed



Networked Systems

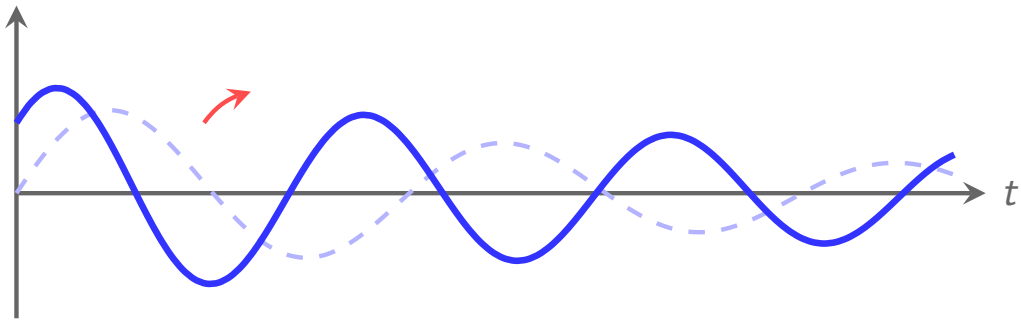
Complex

Automation

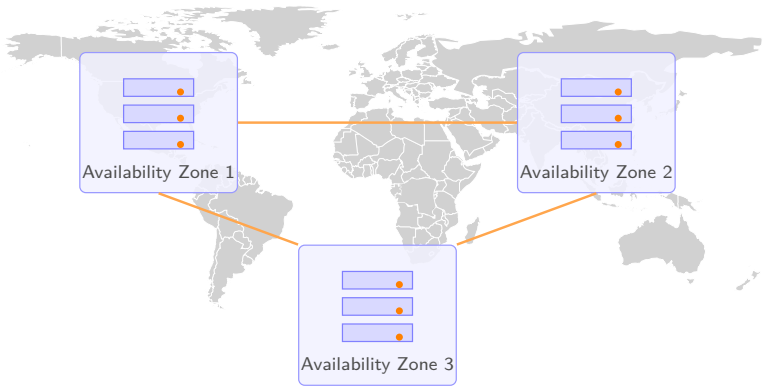


Dynamic

Control



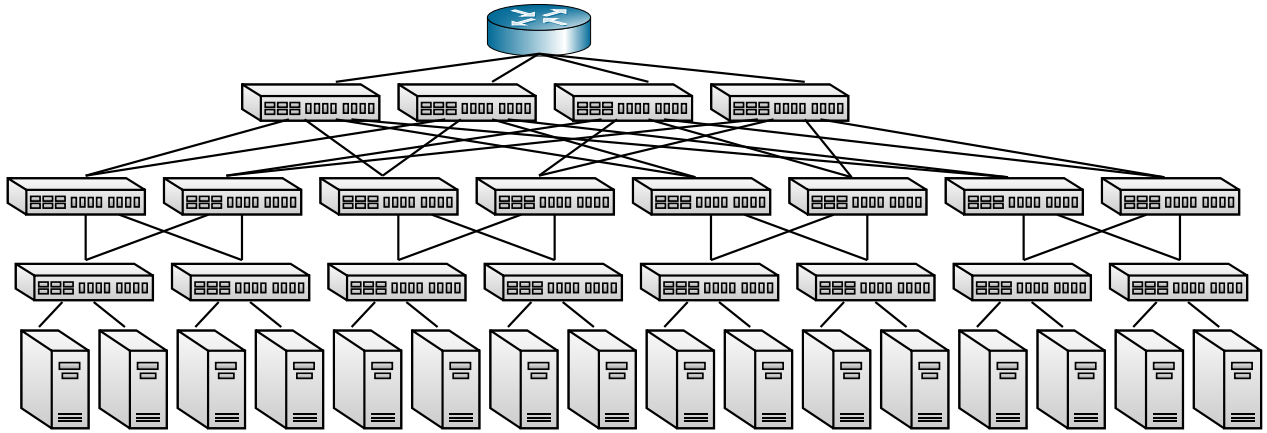
Distributed



Networked Systems

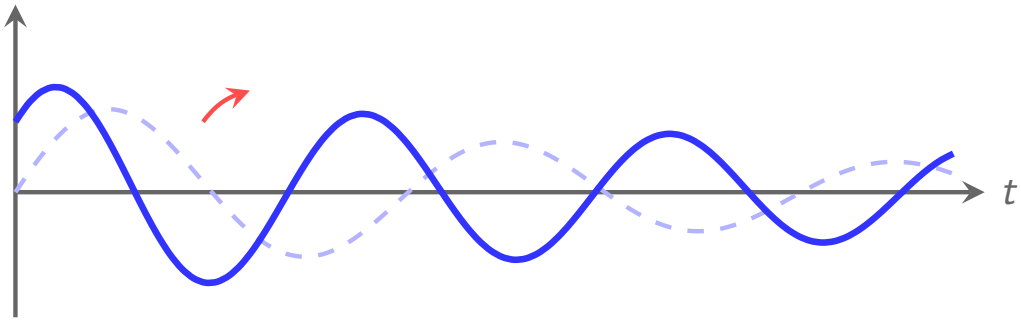
Complex

Automation



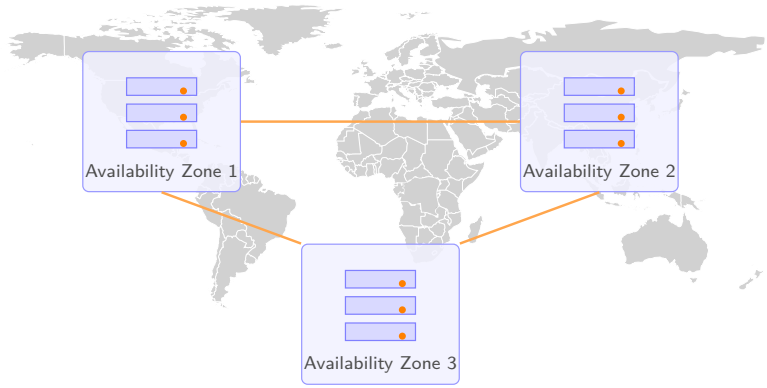
Dynamic

Control

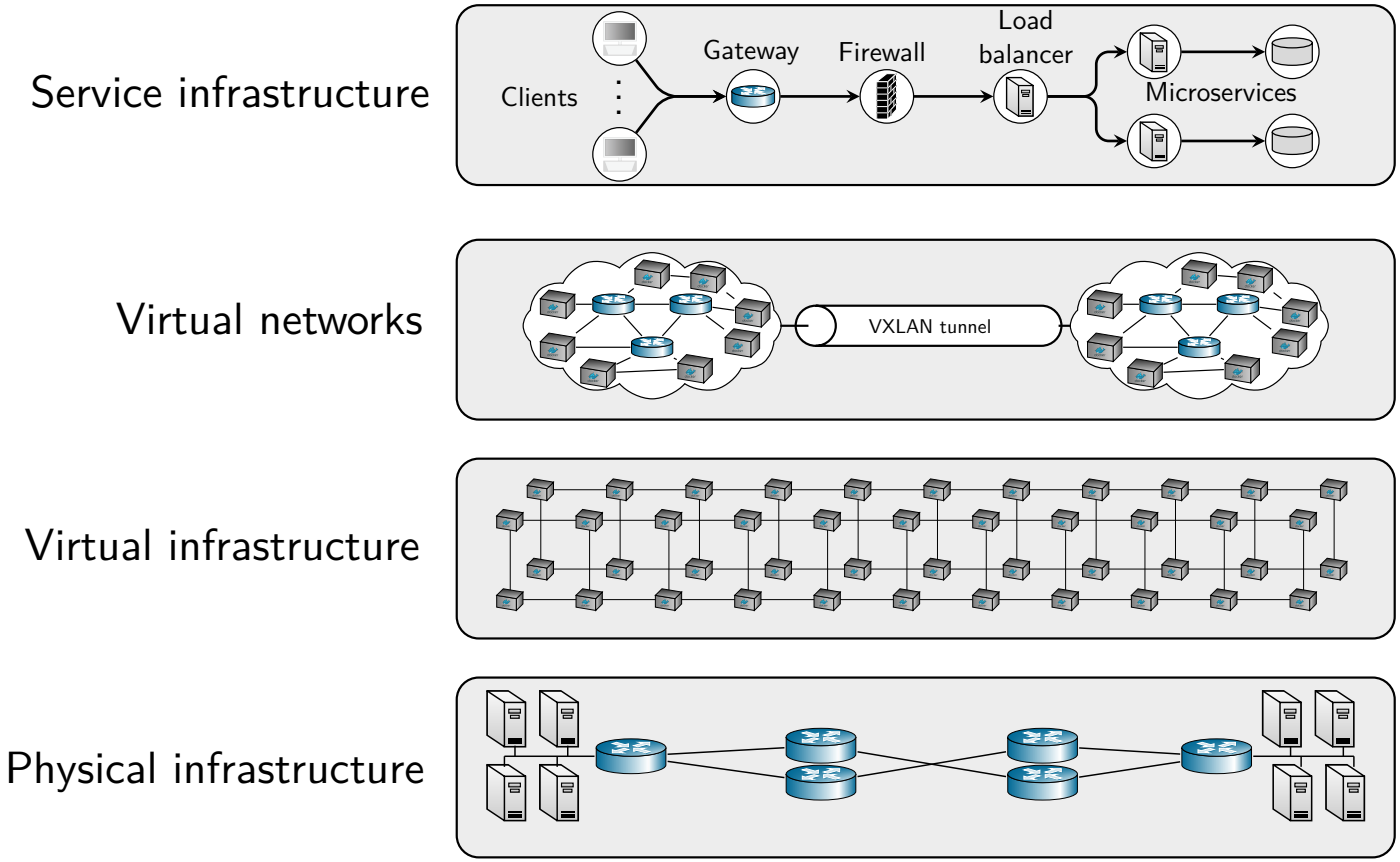


Distributed

Coordination

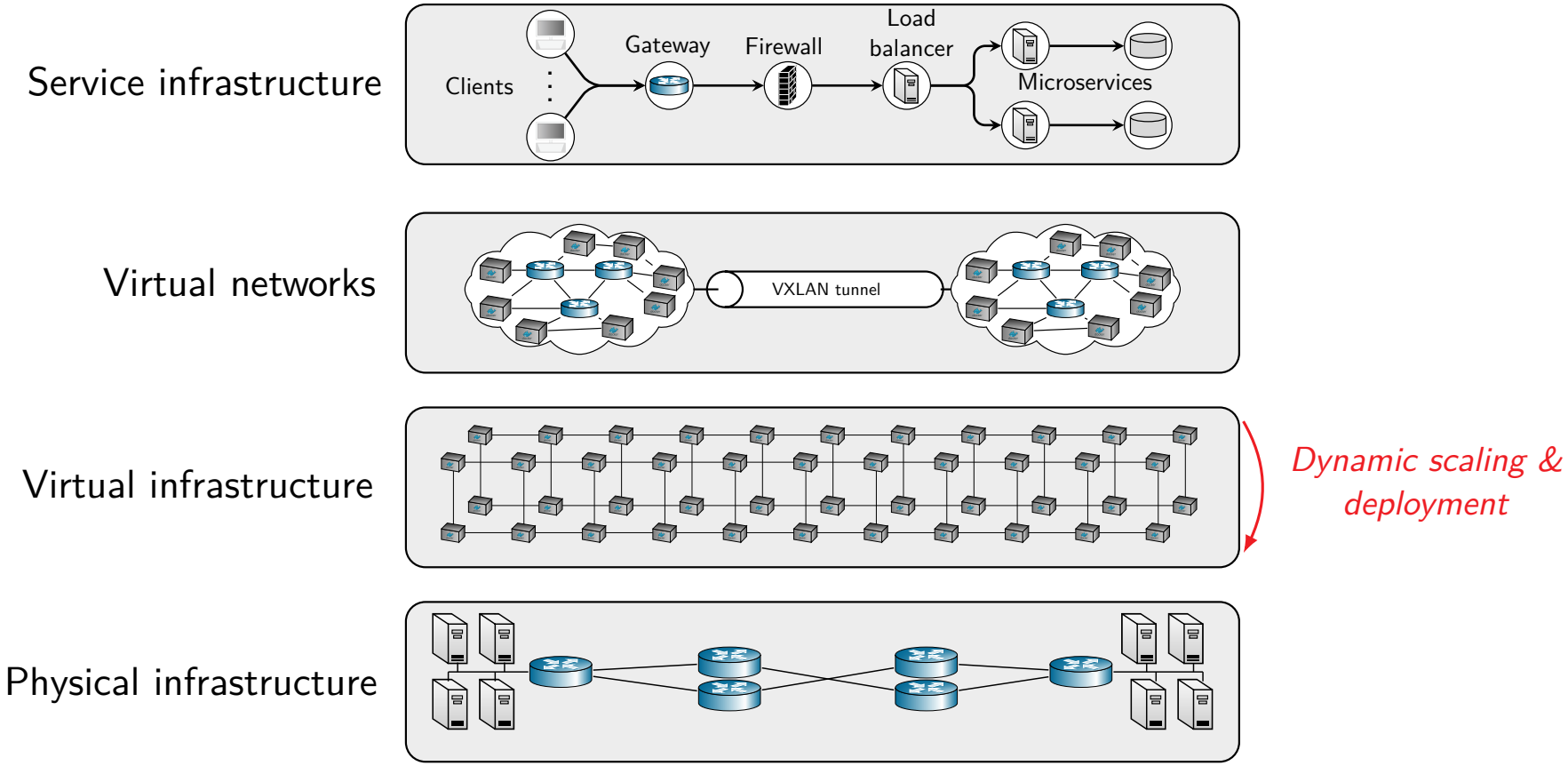


From Manual Configuration to Automatic Control



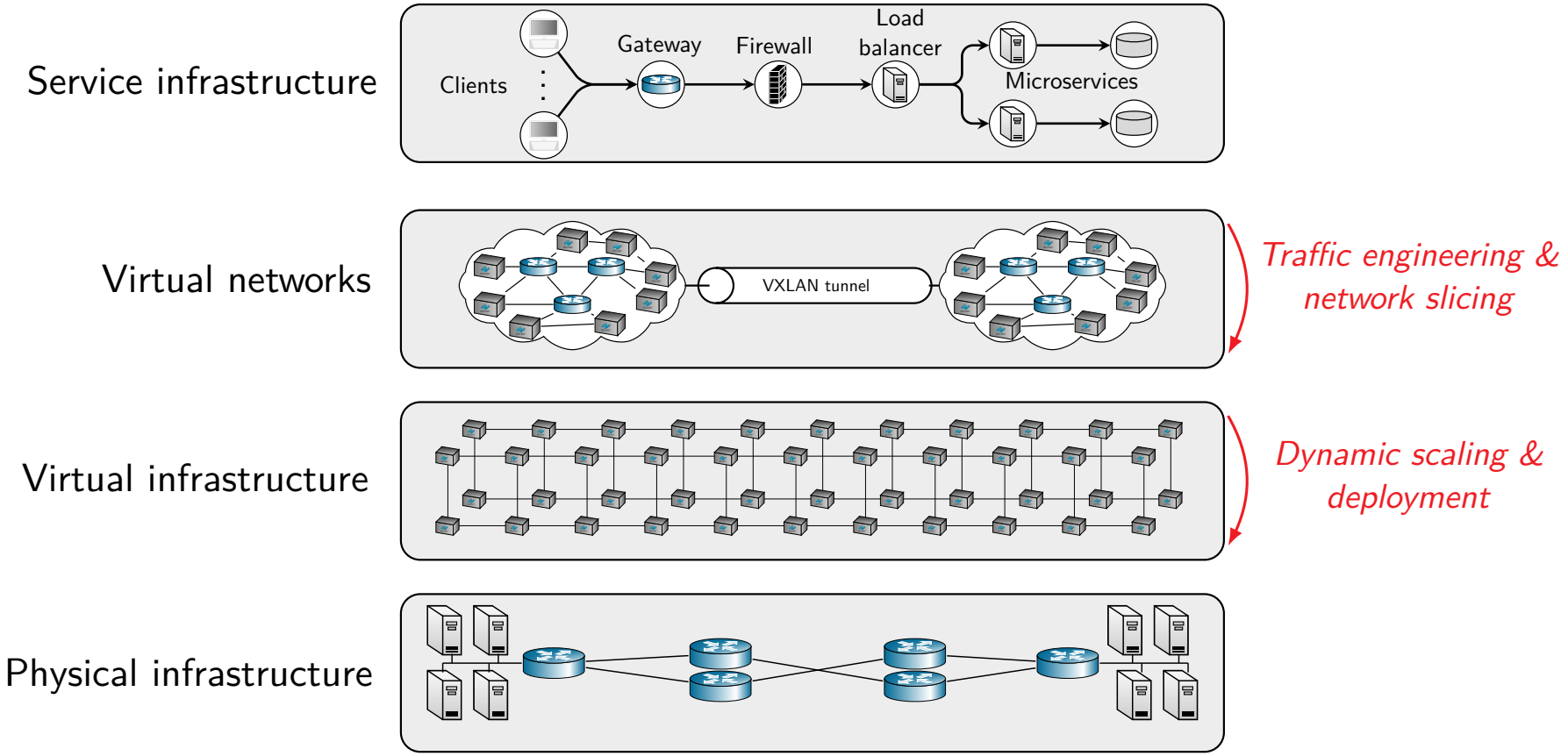
- Networked systems have undergone a shift from hardware-defined architectures to **software-defined** stacks.

From Manual Configuration to Automatic Control



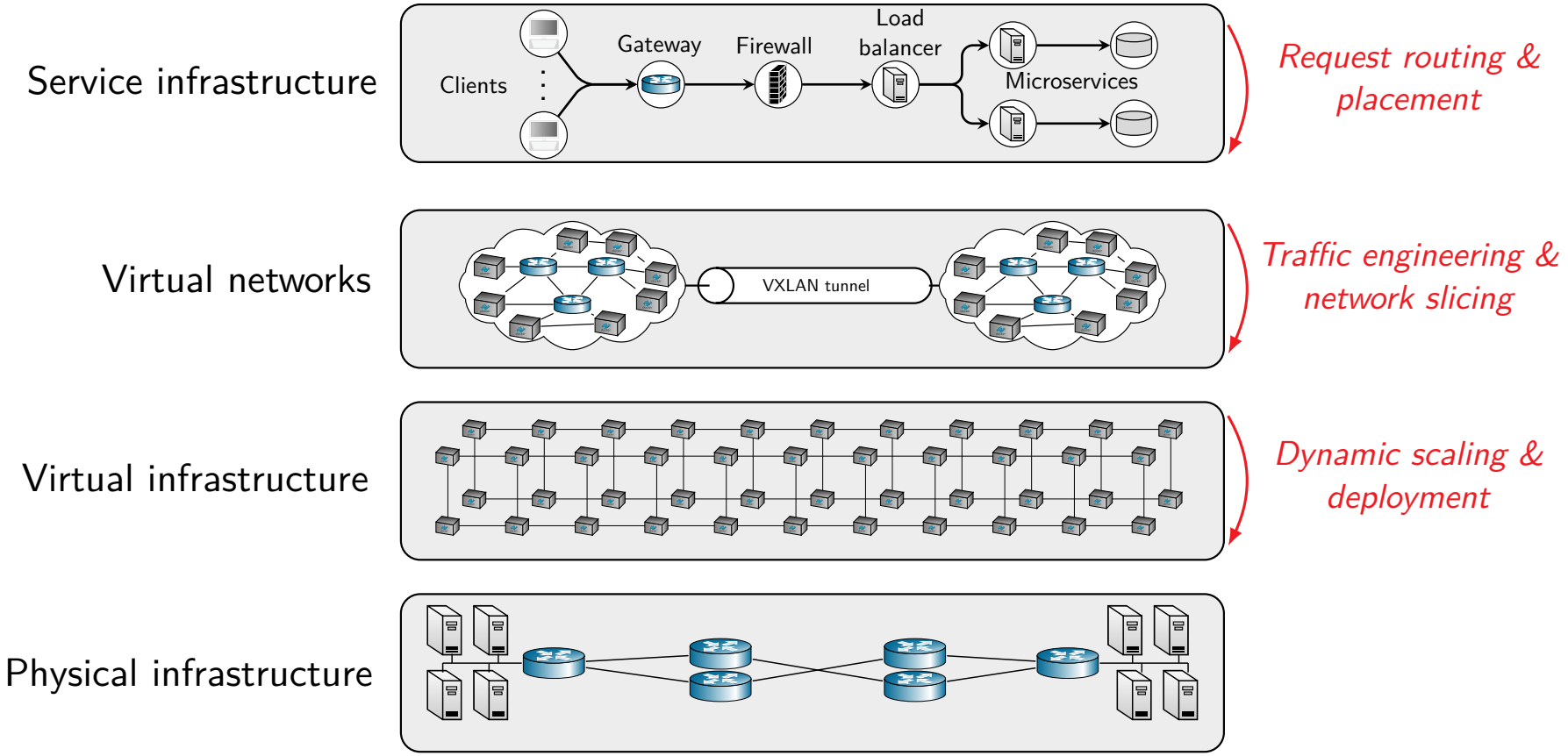
- Networked systems have undergone a shift from hardware-defined architectures to **software-defined** stacks.

From Manual Configuration to Automatic Control



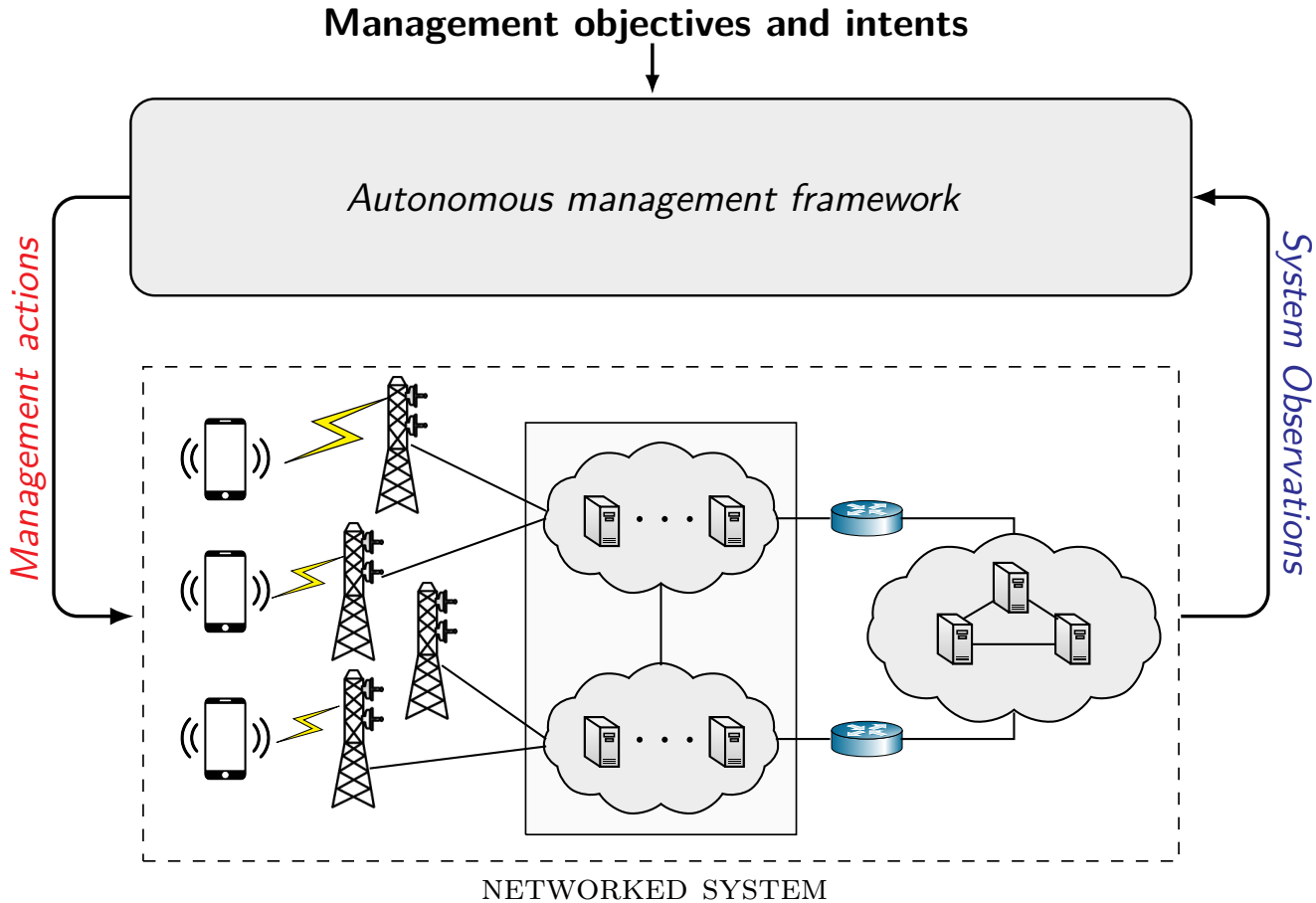
- Networked systems have undergone a shift from hardware-defined architectures to **software-defined** stacks.

From Manual Configuration to Automatic Control



- Networked systems have undergone a shift from hardware-defined architectures to **software-defined** stacks.

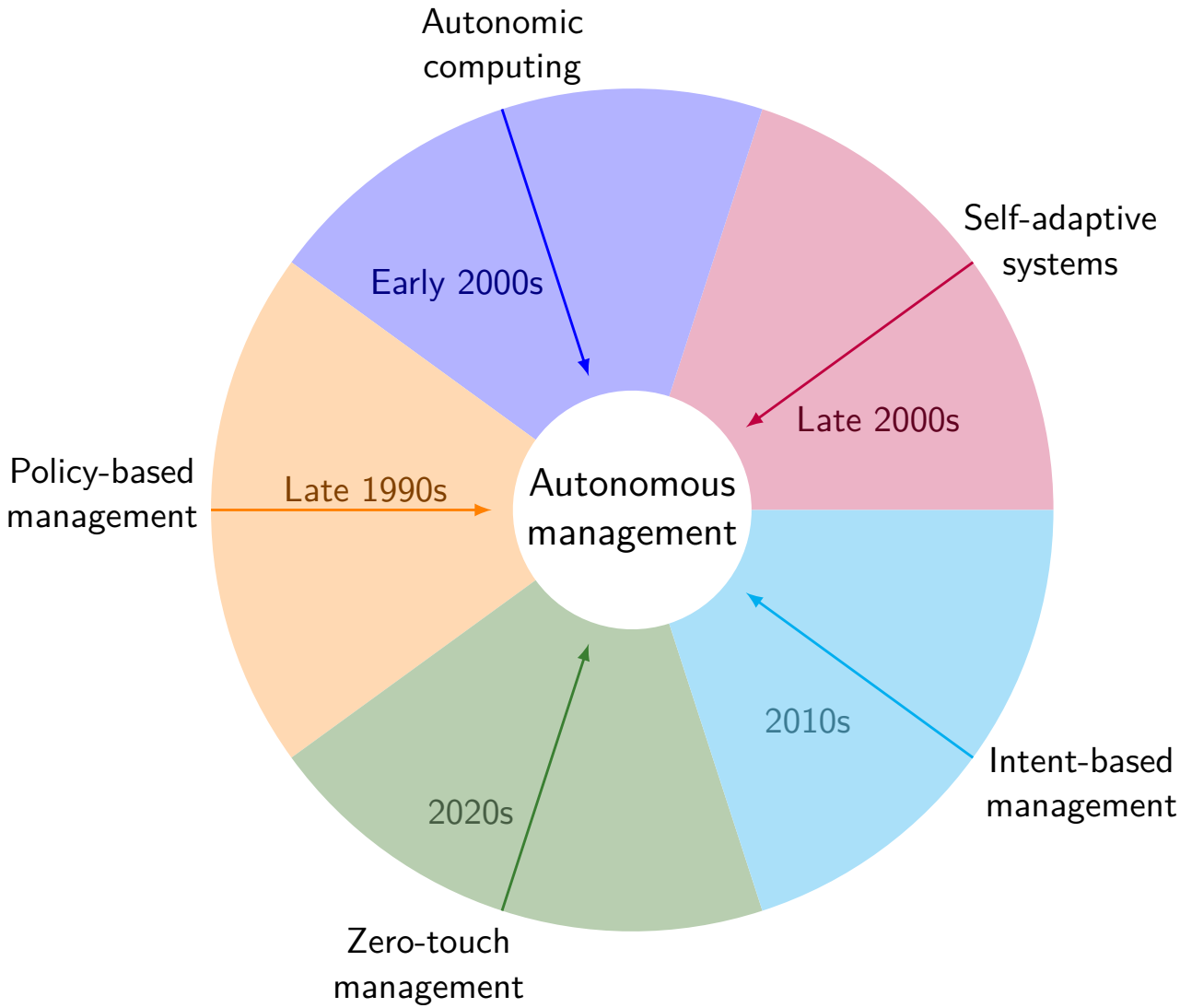
Autonomous Management of Networked Systems



- Autonomous management is needed to cope with the increasing complexity and dynamism of networked systems.
- The programmability and controllability of network and system functions is a prerequisite for autonomous management.

Prior Research

- Efforts towards automating the management of networks and IT systems have been undertaken over the last 30 years.



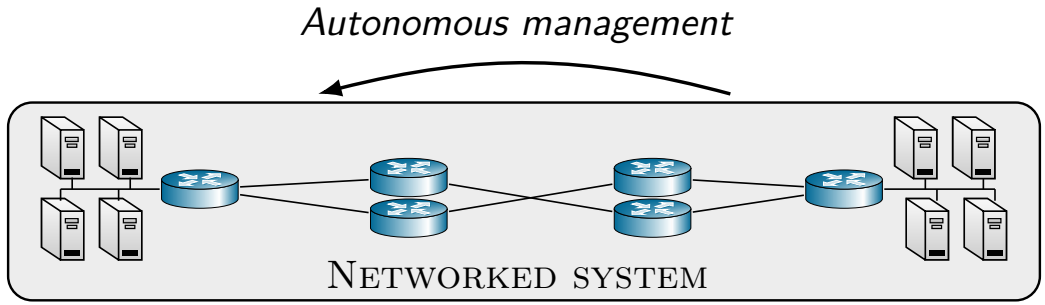
My Research Areas

System modeling.

Game theory, decision theory,
dynamical systems, reliability theory.

Measurement and analysis.

System identification, causal discovery,
traffic characterization, anomaly detection.



Optimization & learning.

Reinforcement learning, dynamic programming,
stochastic approximation, Bayesian learning.

Systems engineering.

Network emulation, real-time monitoring,
attack emulation, load generation.
https://github.com/Kim-Hammar.

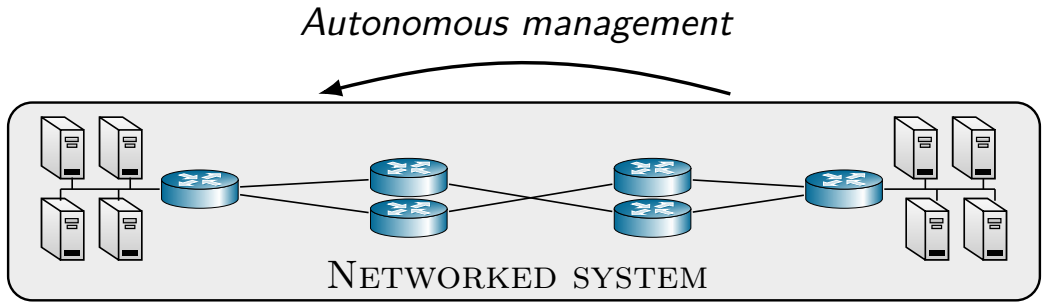
My Research Areas

System modeling.

Game theory, decision theory,
dynamical systems, reliability theory.

Measurement and analysis.

System identification, causal discovery,
traffic characterization, anomaly detection.



Optimization & learning.

Reinforcement learning, dynamic programming,
stochastic approximation, Bayesian learning.

Systems engineering.

Network emulation, real-time monitoring,
attack emulation, load generation.
https://github.com/Kim-Hammar.

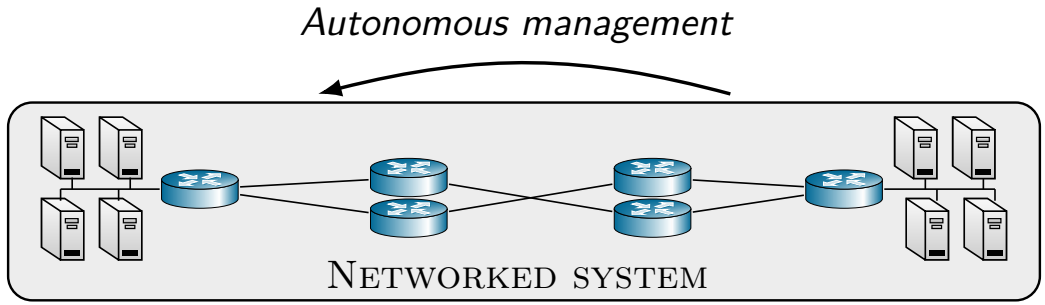
My Research Areas

System modeling.

Game theory, decision theory,
dynamical systems, reliability theory.

Measurement and analysis.

System identification, causal discovery,
traffic characterization, anomaly detection.



Optimization & learning.

Reinforcement learning, dynamic programming,
stochastic approximation, Bayesian learning.

Systems engineering.

Network emulation, real-time monitoring,
attack emulation, load generation.
https://github.com/Kim-Hammar.

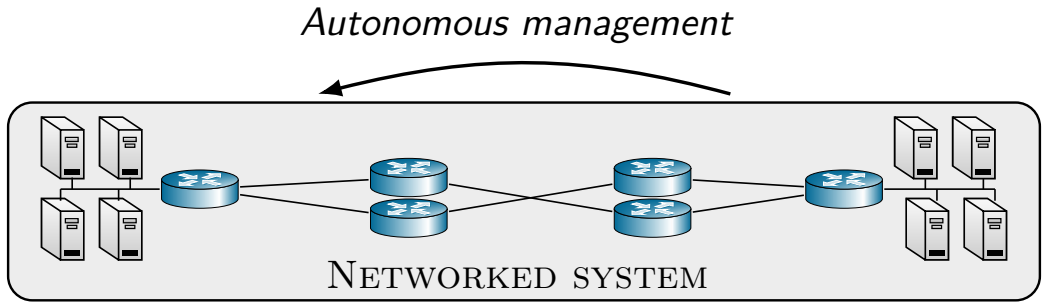
My Research Areas

System modeling.

Game theory, decision theory,
dynamical systems, reliability theory.

Measurement and analysis.

System identification, causal discovery,
traffic characterization, anomaly detection.



Optimization & learning.

Reinforcement learning, dynamic programming,
stochastic approximation, Bayesian learning.

Systems engineering.

Network emulation, real-time monitoring,
attack emulation, load generation.
https://github.com/Kim-Hammar.

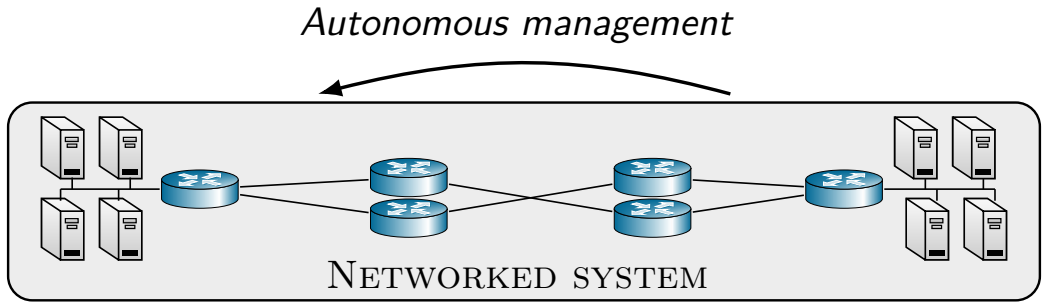
My Research Areas

System modeling.

Game theory, decision theory,
dynamical systems, reliability theory.

Measurement and analysis.

System identification, causal discovery,
traffic characterization, anomaly detection.



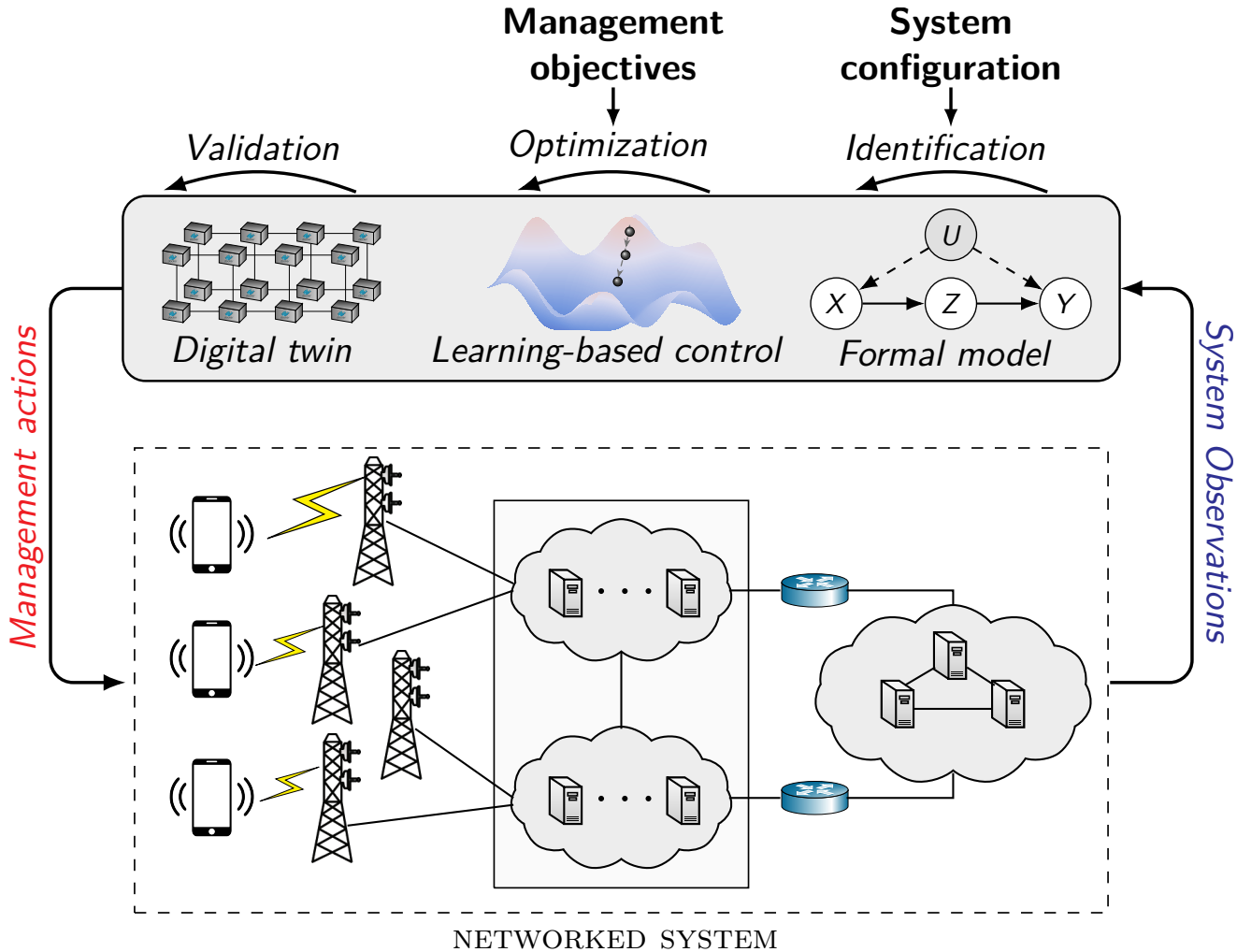
Optimization & learning.

Reinforcement learning, dynamic programming,
stochastic approximation, Bayesian learning.

Systems engineering.

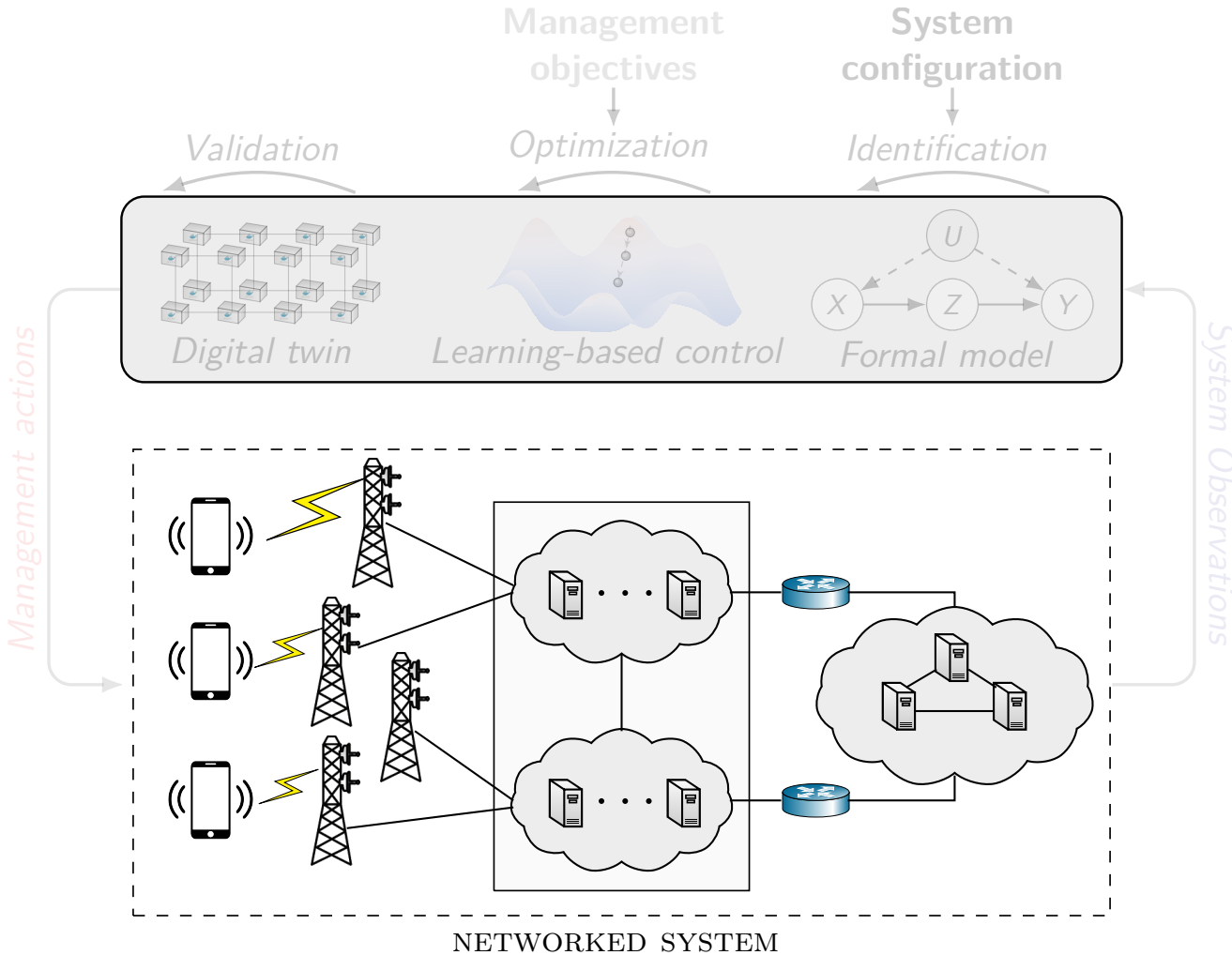
Network emulation, real-time monitoring,
attack emulation, load generation.
https://github.com/Kim-Hammar.

Vision: Learning-Based Control of Networked Systems



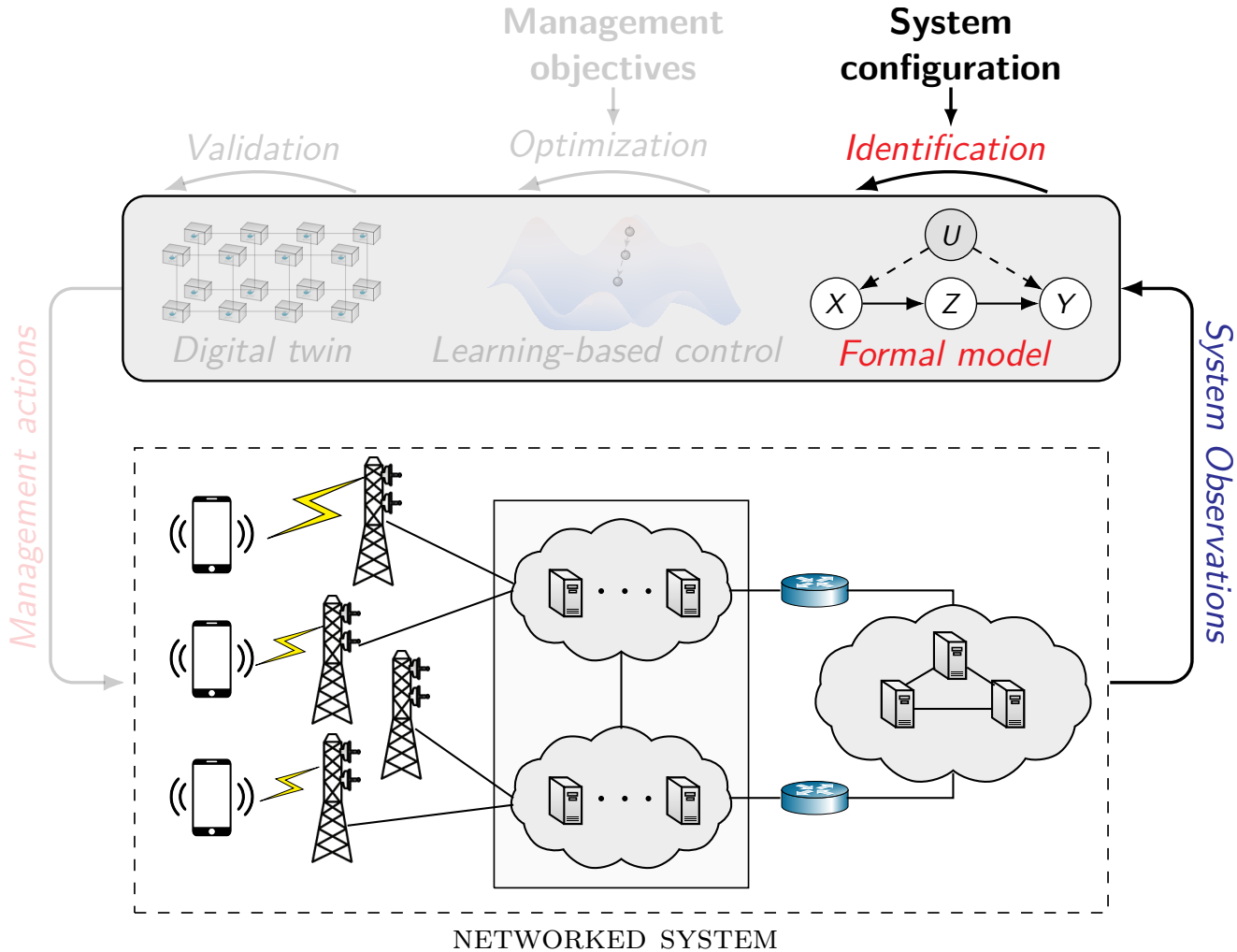
- I advocate for an approach to autonomy that includes:
  - Identification of a formal system model.
  - Learning-based control through simulation of the model.
  - Control strategy validation on a digital twin.

Vision: Learning-Based Control of Networked Systems



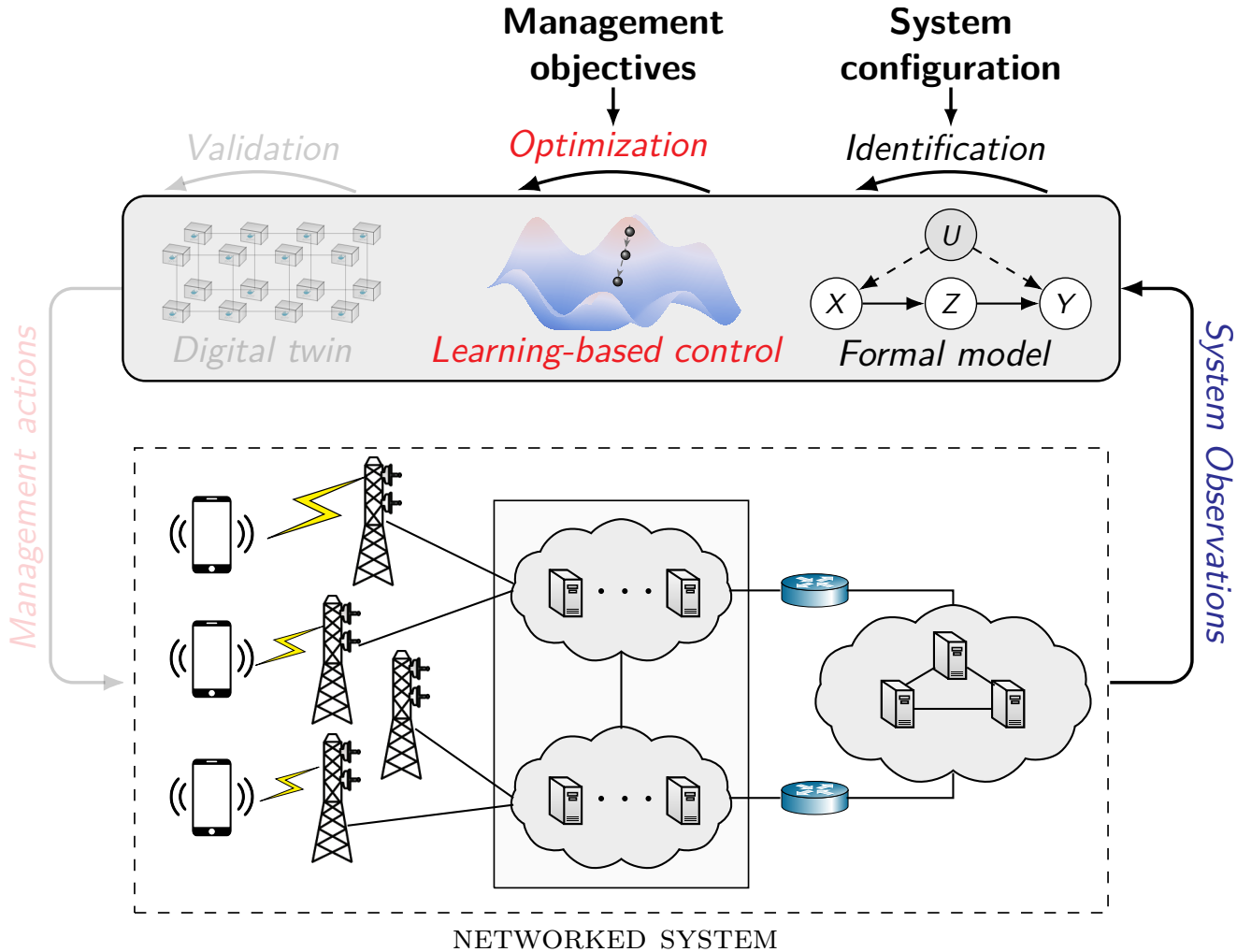
- I advocate for an approach to autonomy that includes:
  - Identification of a formal system model.
  - Learning-based control through simulation of the model.
  - Control strategy validation on a digital twin.

Vision: Learning-Based Control of Networked Systems



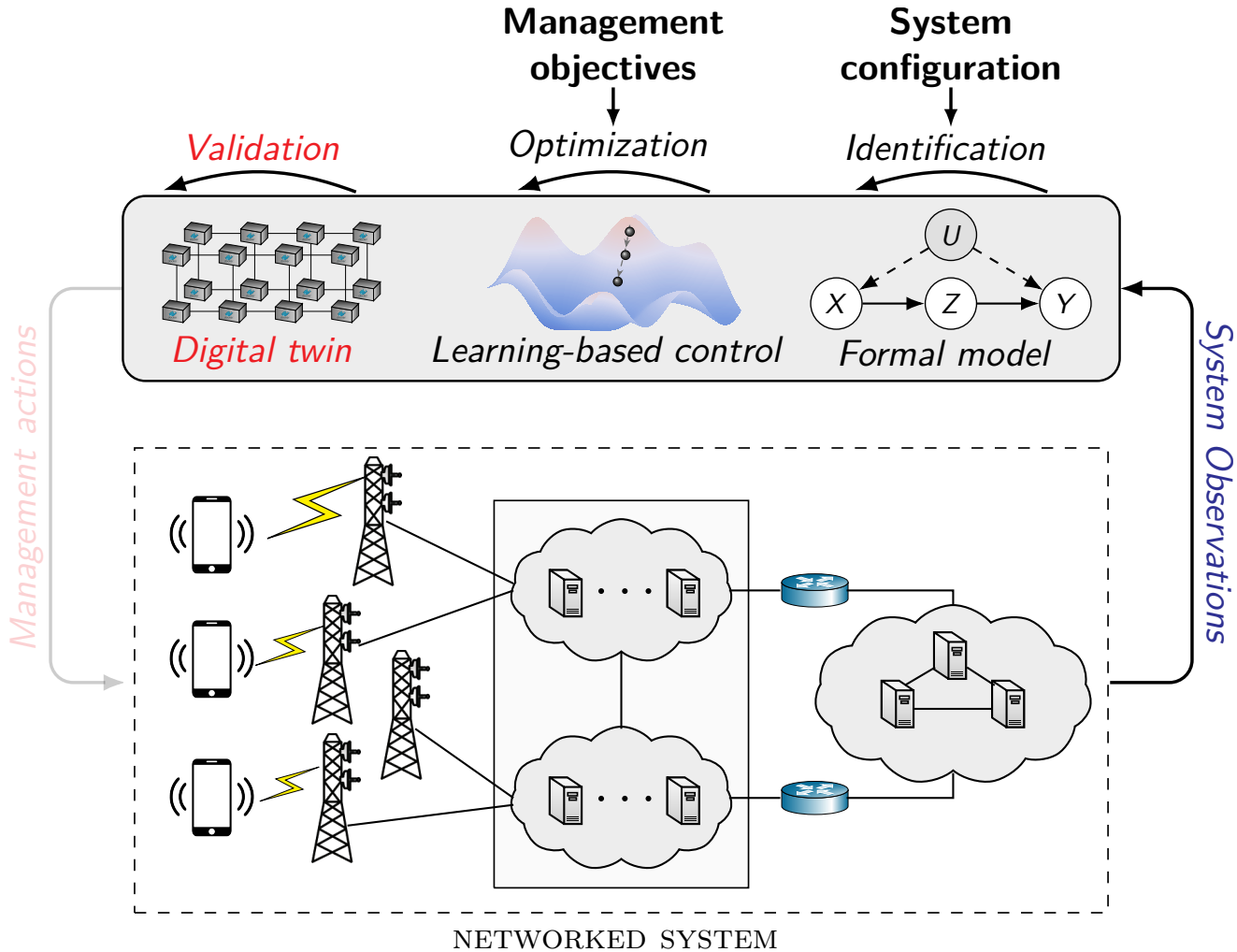
- I advocate for an approach to autonomy that includes:
  - Identification of a formal system model.
  - Learning-based control through simulation of the model.
  - Control strategy validation on a digital twin.

Vision: Learning-Based Control of Networked Systems



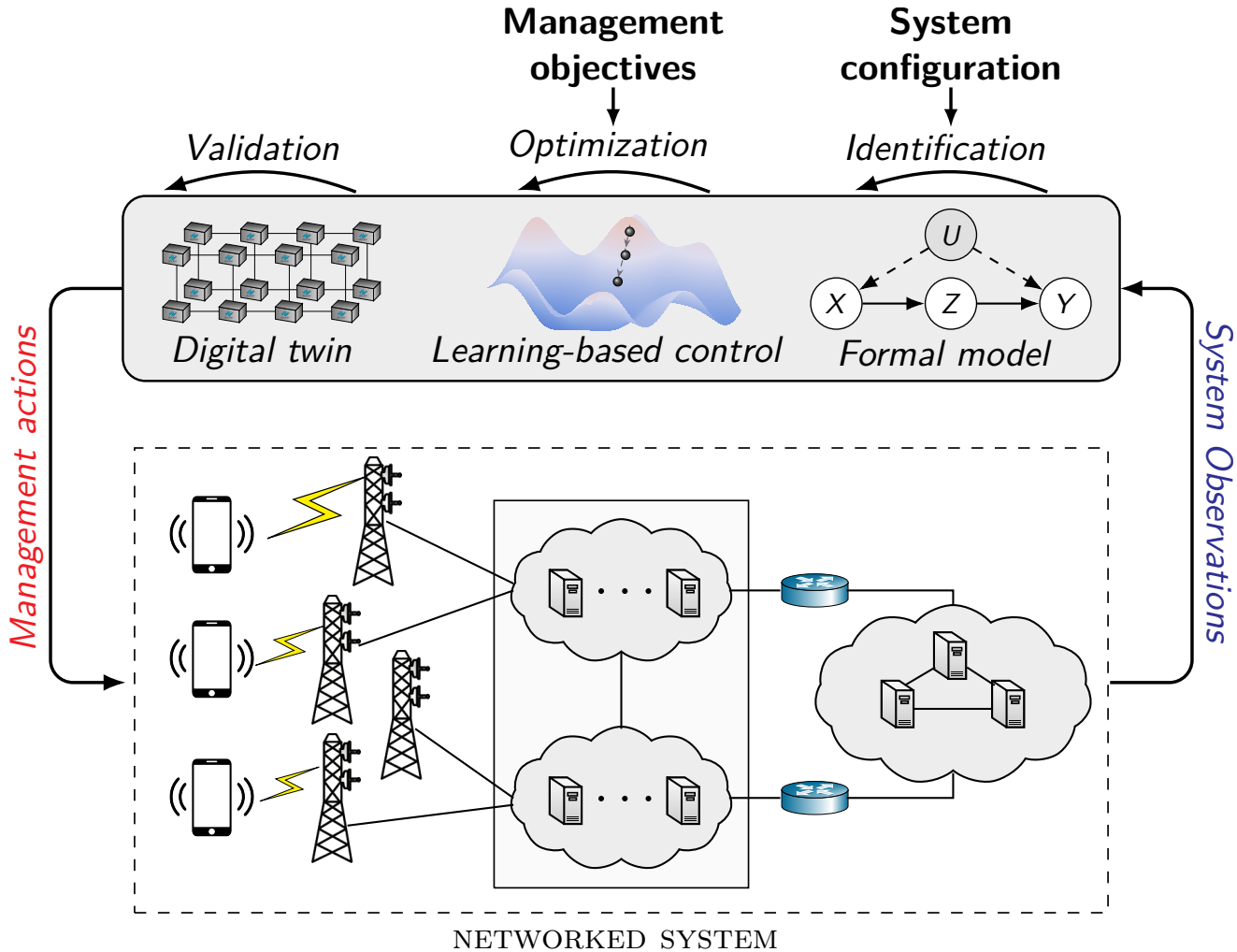
- I advocate for an approach to autonomy that includes:
  - Identification of a formal system model.
  - Learning-based control through simulation of the model.
  - Control strategy validation on a digital twin.

Vision: Learning-Based Control of Networked Systems



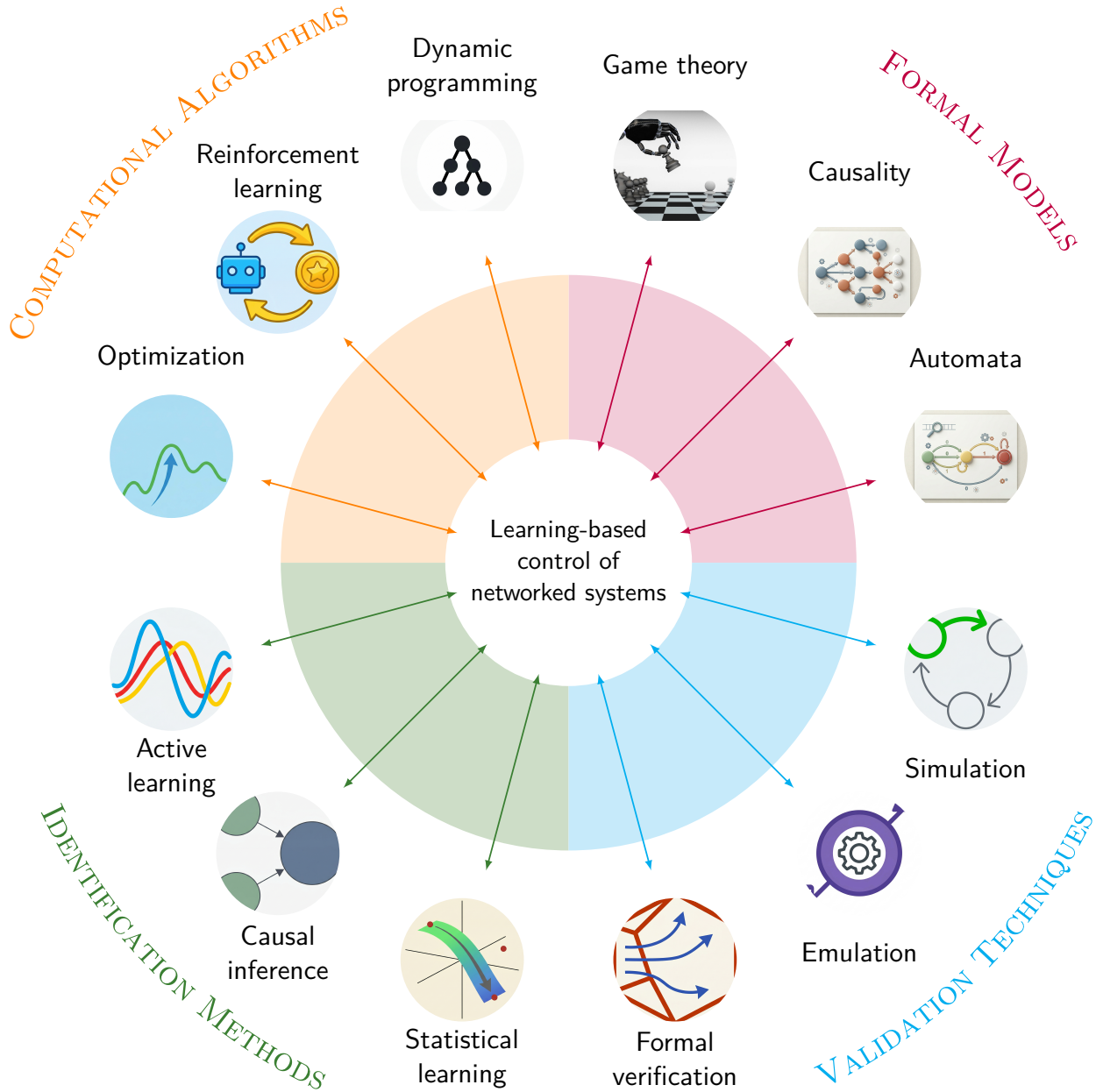
- I advocate for an approach to autonomy that includes:
  - Identification of a formal system model.
  - Learning-based control through simulation of the model.
  - Control strategy validation on a digital twin.

Vision: Learning-Based Control of Networked Systems



- I advocate for an approach to autonomy that includes:
  - Identification of a formal system model.
  - Learning-based control through simulation of the model.
  - Control strategy validation on a digital twin.

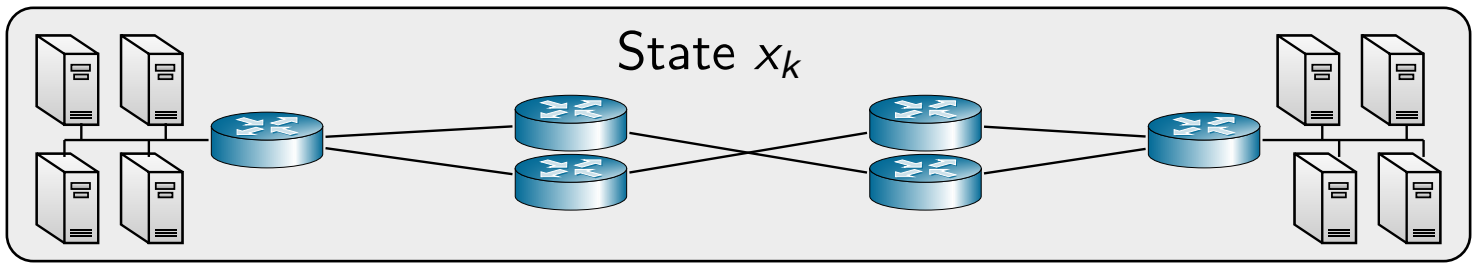
Foundations



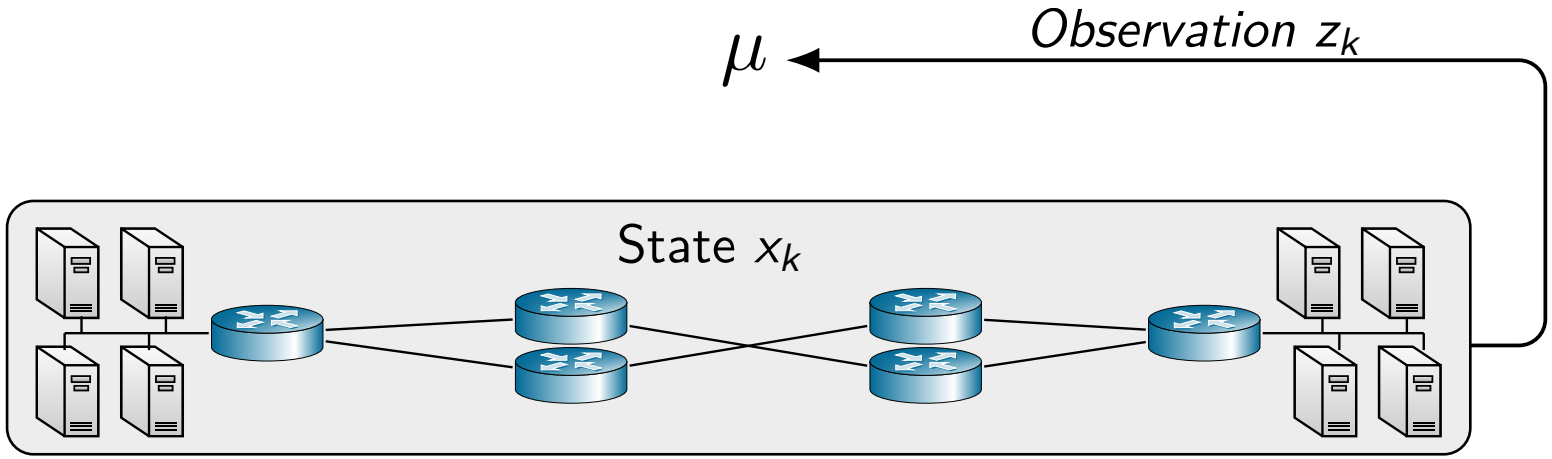
Outline

- **Autonomous management as a control problem.**
  - Problem formulation.
  - Challenges.
- **Causal online learning of safe regions.**
  - Causal inference.
  - Interventional learning.
- **Game-theoretic learning of security strategies.**
  - Game-theoretic model and equilibrium analysis.
  - Conjectural online learning.
- **Multiagent incident response with LLMs.**
  - System architecture.
  - Theoretical analysis.
- **Conclusion.**

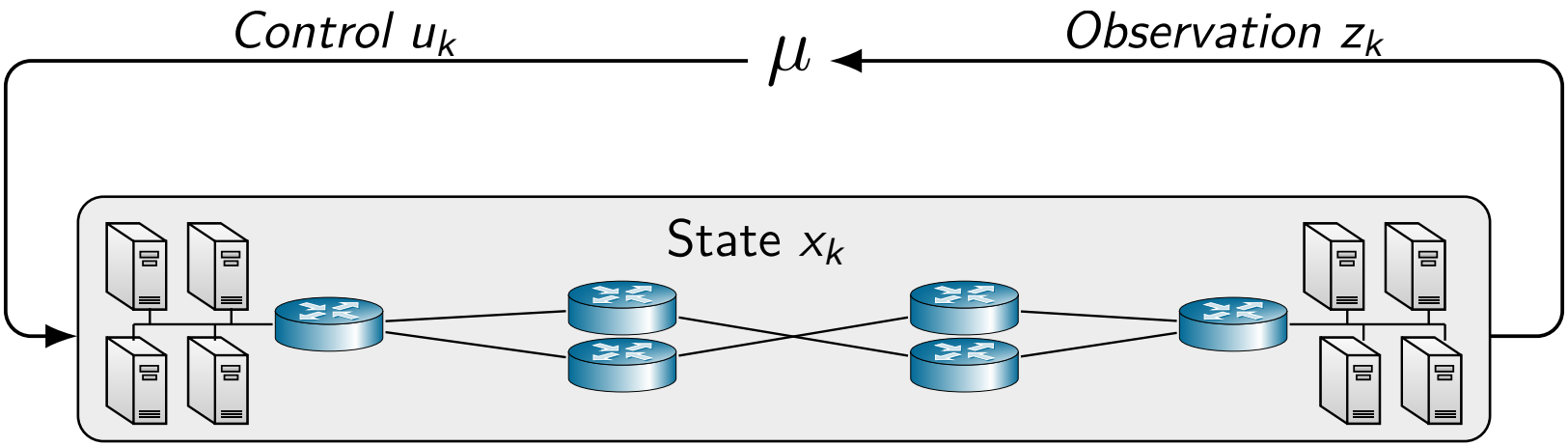
Autonomous Management as a Control Problem



Autonomous Management as a Control Problem

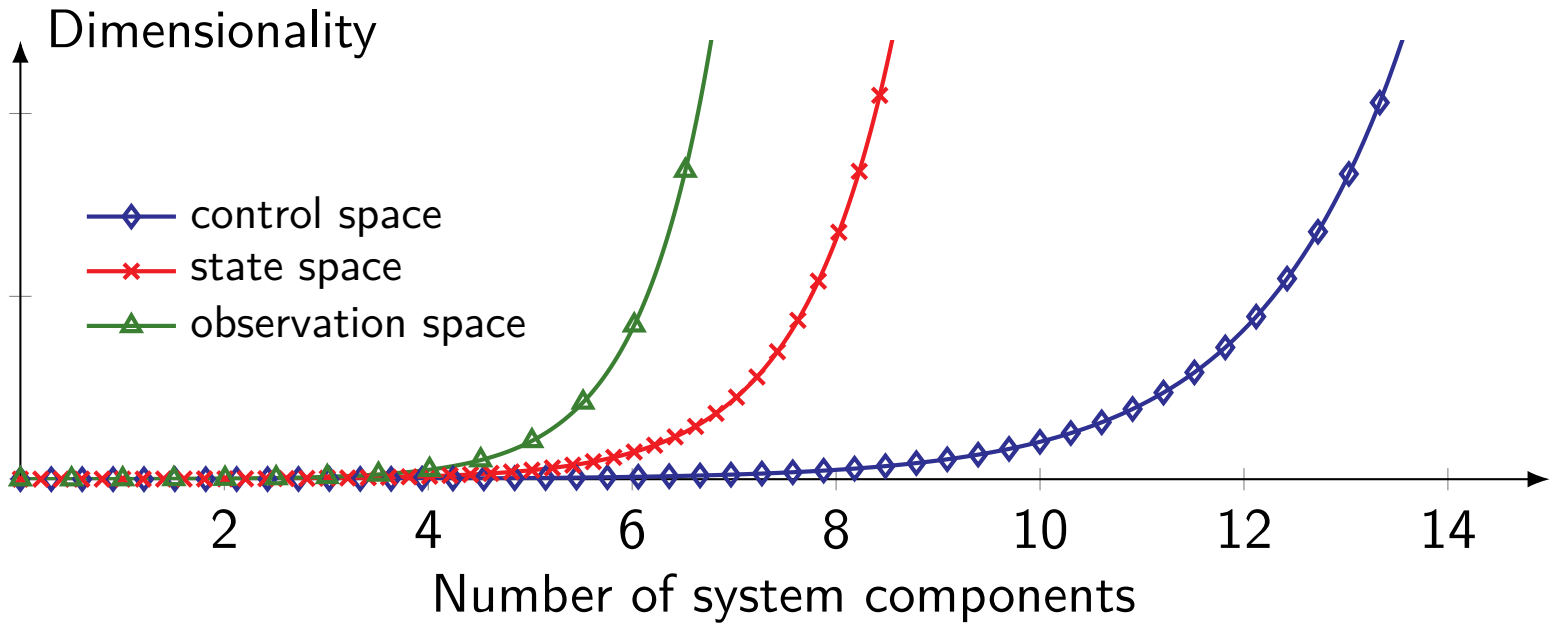


Autonomous Management as a Control Problem



- State x_k (e.g., system load and configuration).
- Observation z_k (e.g., log files and performance metrics).
- Control u_k (e.g., load balancing).
- **Goal**: find a strategy μ that meets management objectives.

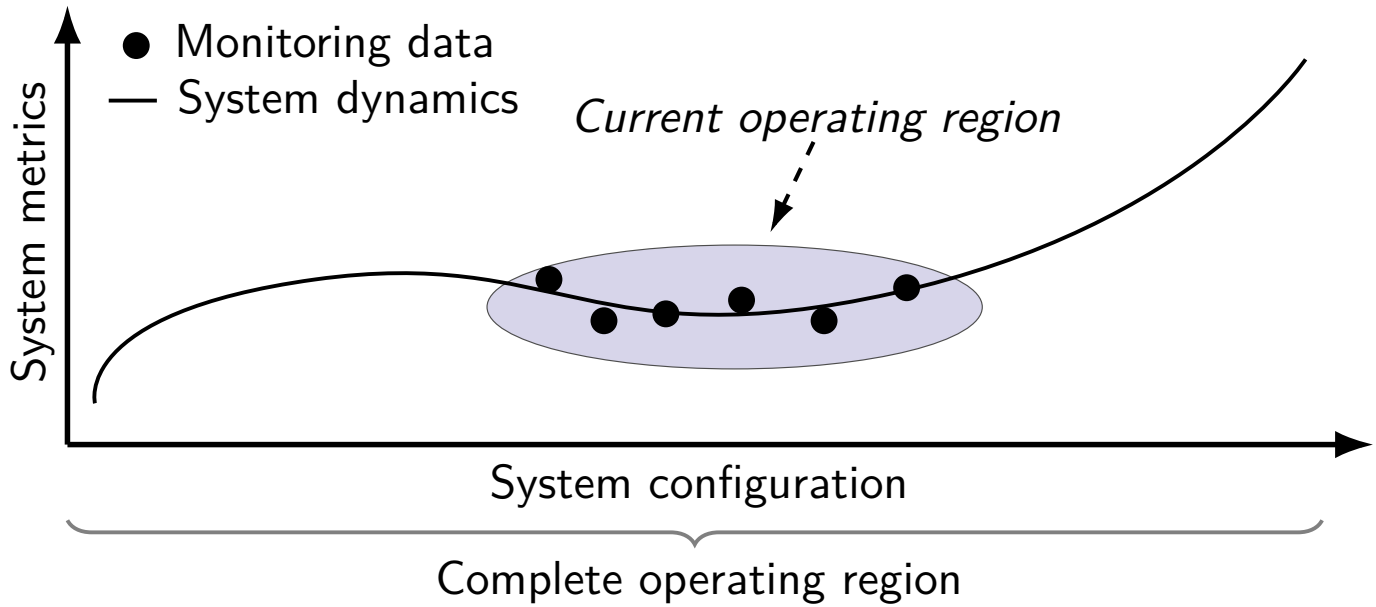
The Scalability Challenge



Curse of dimensionality

Networked system contain thousands of interconnected system variables $\Longrightarrow$ combinatorial explosion.

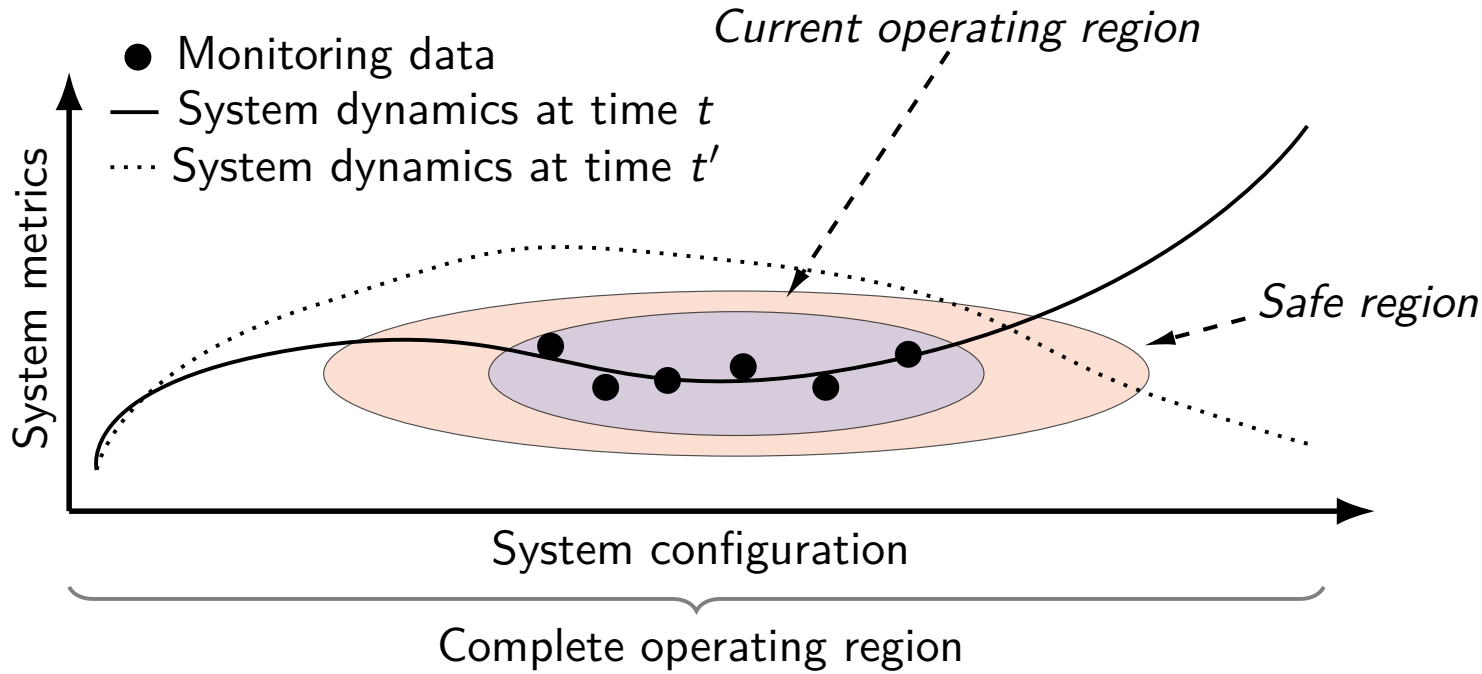
The Identification Challenge



System identification and modeling

There are no equations that describe how a networked system operates. The dynamics must be learned from data.

The Operation Challenge



Operational constraints and events

Networked systems are subject to operational constraints (e.g., performance and availability requirements) and unexpected events (e.g., non-stationarity and cyberattacks).

Summary Of Challenges

Curse of dimensionality

Networked system contain thousands of interconnected system variables $\Longrightarrow$ combinatorial explosion.

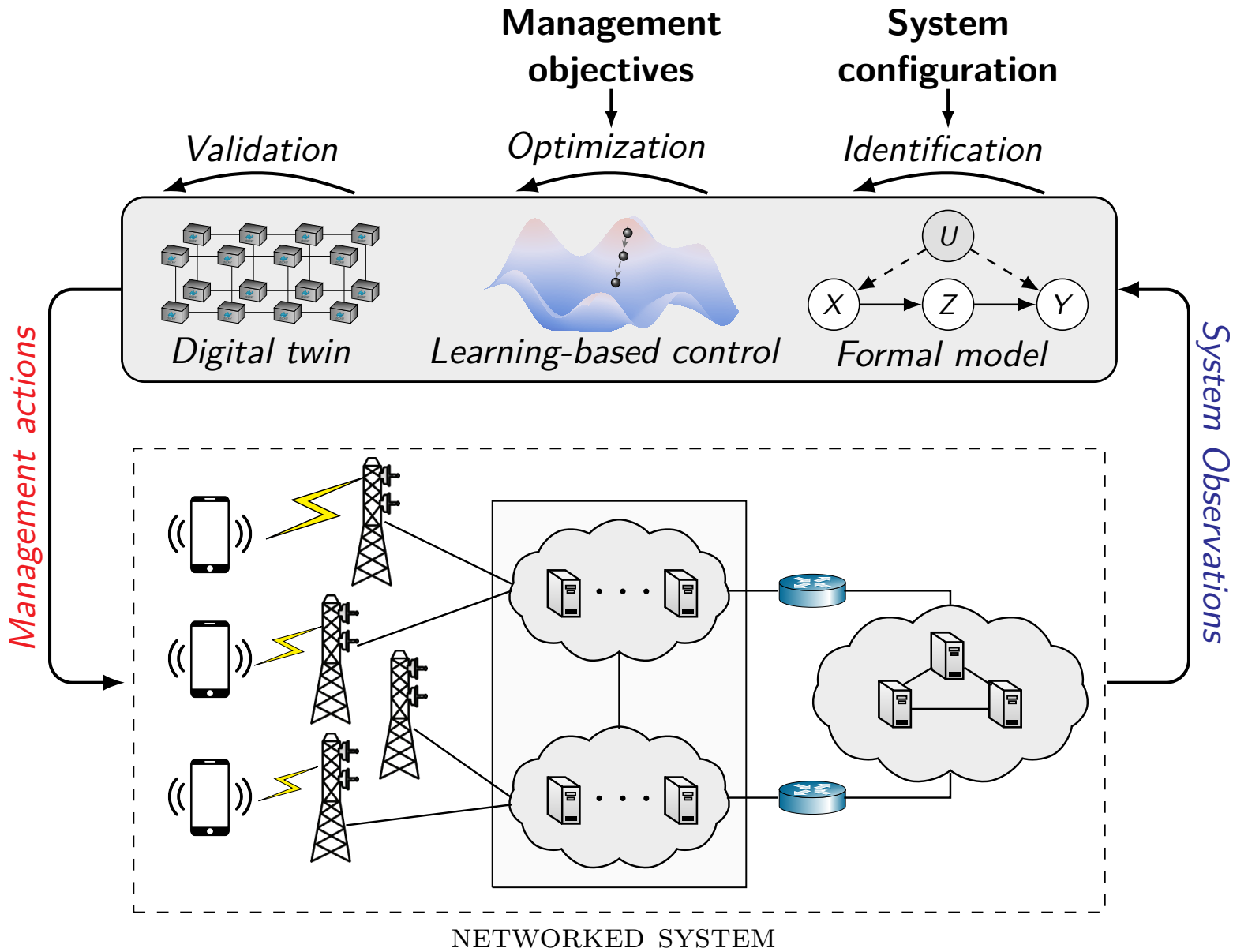
System identification and modeling

There are no equations that describe how a networked system operates. The dynamics must be learned from data.

Operational constraints and events

Networked systems are subject to operational constraints (e.g., performance and availability requirements) and unexpected events (e.g., non-stationarity and cyberattacks).

Method: Learning-Based Control of Networked Systems

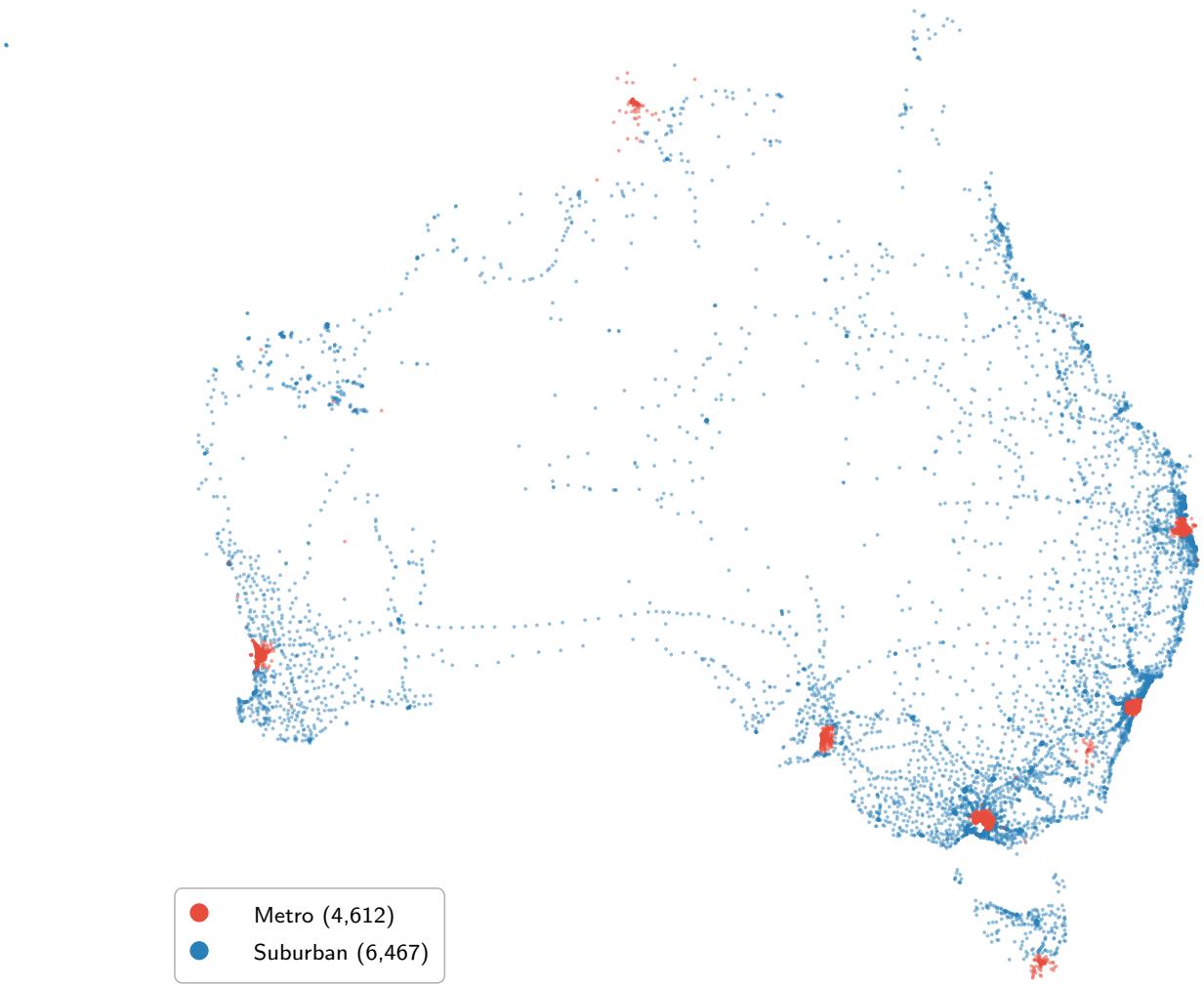


Outline

- **Autonomous management as a control problem.**
  - Problem formulation.
  - Challenges.
- **Causal online learning of safe regions.**
  - Causal inference.
  - Interventional learning.
- **Game-theoretic learning of security strategies.**
  - Game-theoretic model and equilibrium analysis.
  - Conjectural online learning.
- **Multiagent incident response with LLMs.**
  - System architecture.
  - Theoretical analysis.
- **Conclusion.**

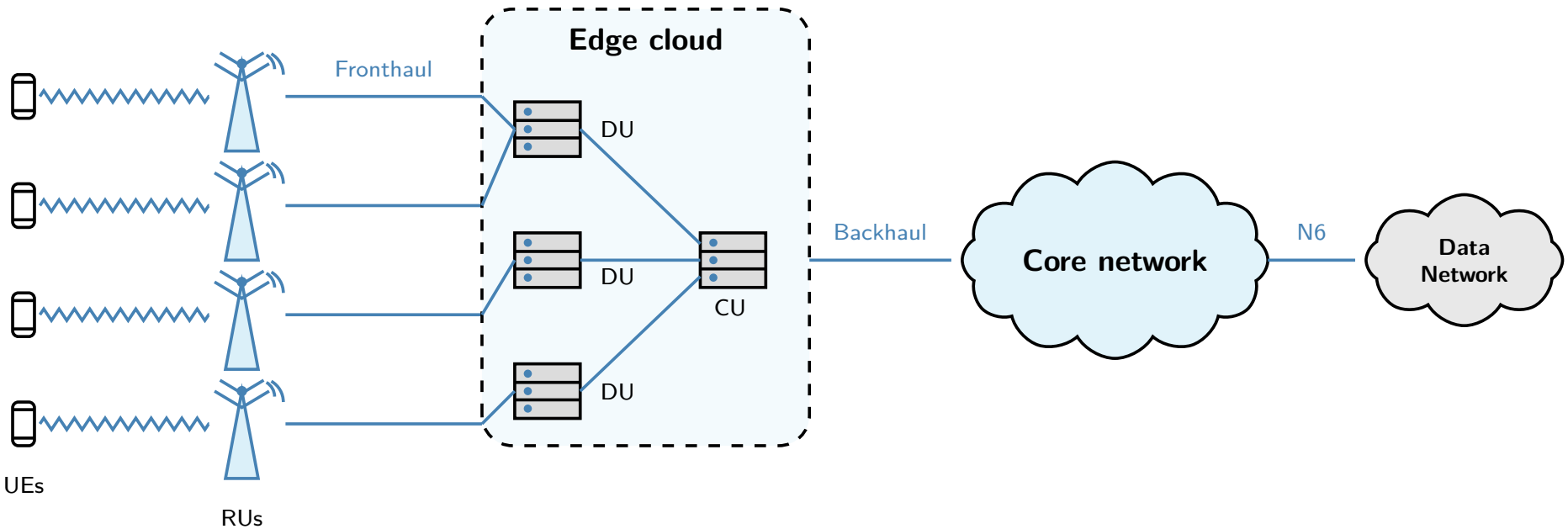
Background

- Use case is based on an **industry collaboration** with Telstra.
- Telstra operates **12,000 sites** across Australia.
- 93 million parameters under management.
- 10 billion control events per day.

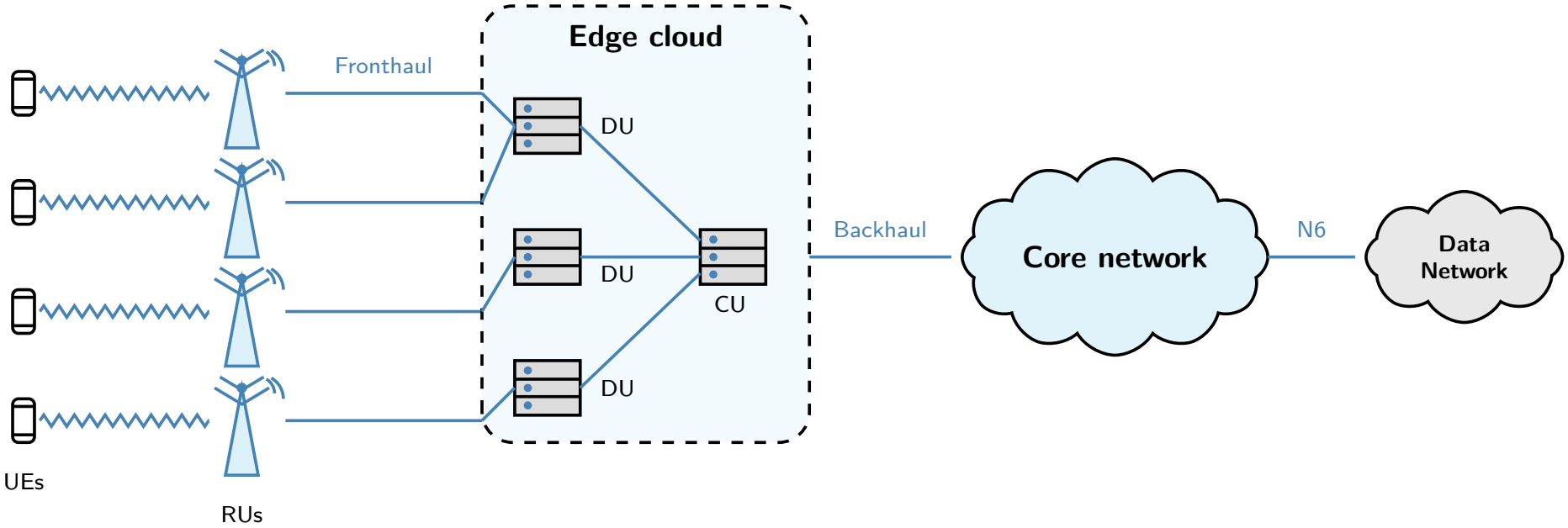


A Base Station Site

Cloud Radio Access Network



Cloud Radio Access Network



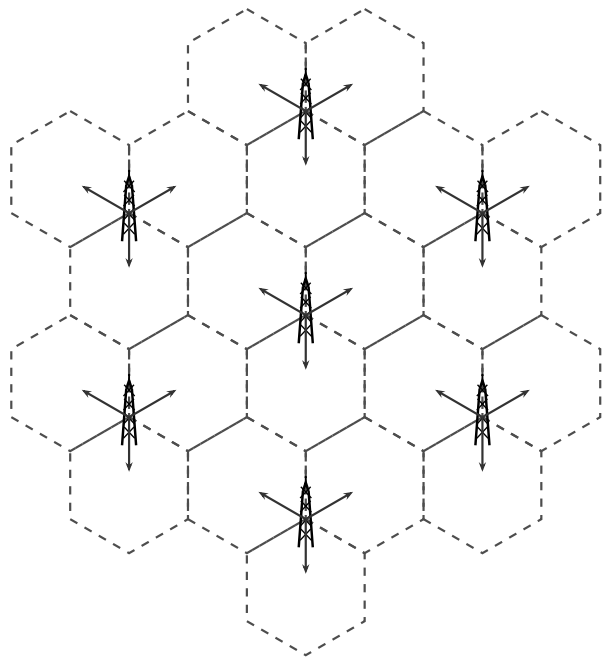
Use Case: Dynamic Resource Allocation

We want to design a control strategy that dynamically allocates resources in the RAN (e.g., CPU, memory, bandwidth) to meet management objectives.

Challenges

1. When optimizing resource allocations, we must ensure that constraints are satisfied.

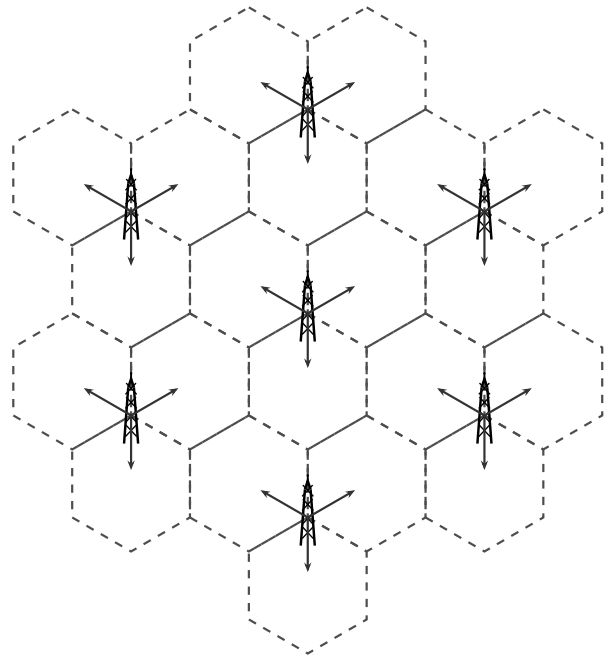
2. While the RAN's configuration is known, the causal effects of control inputs are unknown.



Challenges

1. When optimizing resource allocations, we must ensure that constraints are satisfied.

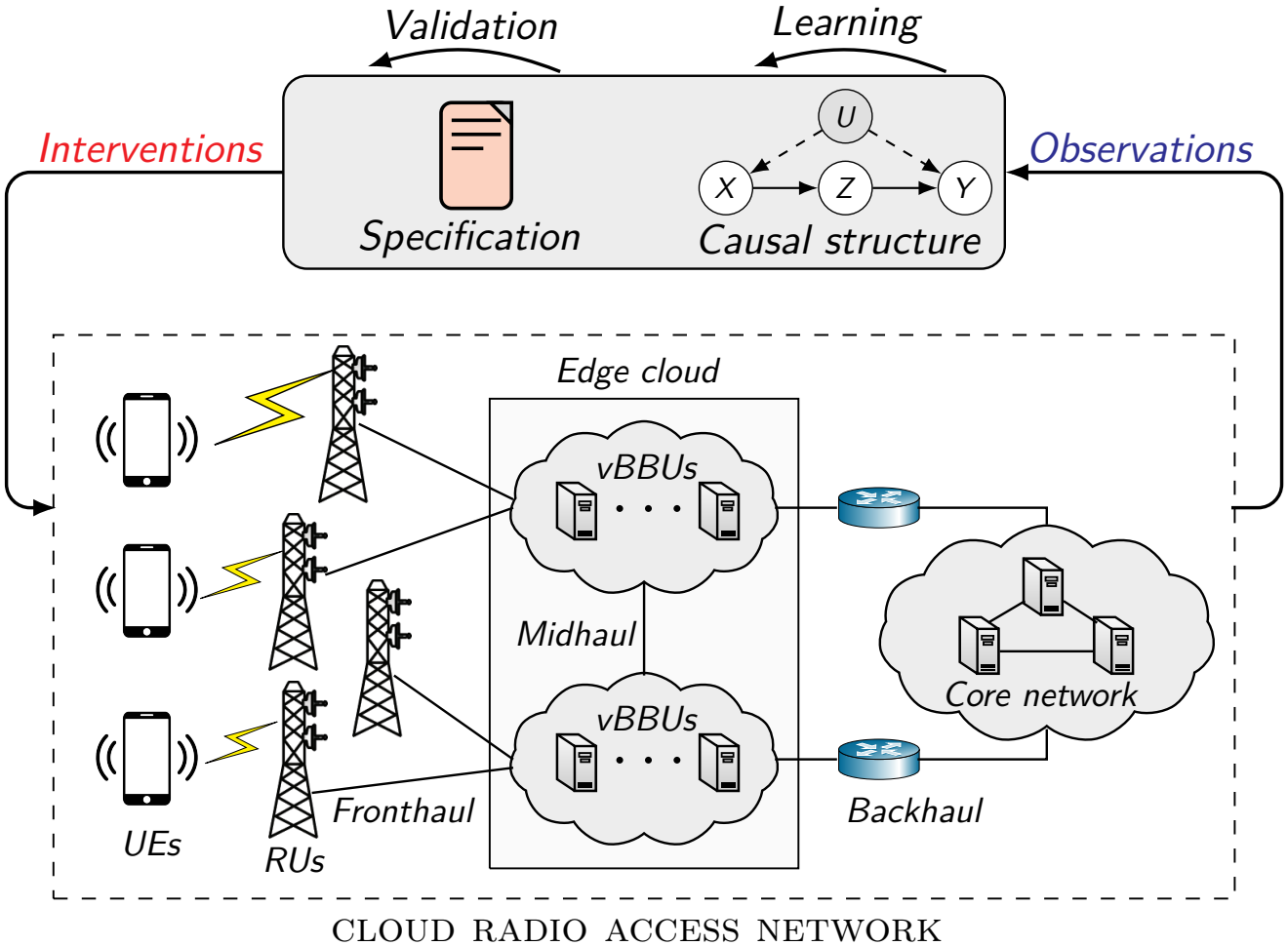
2. While the RAN's configuration is known, the causal effects of control inputs are unknown.



Problem: Identifying the Safe Region

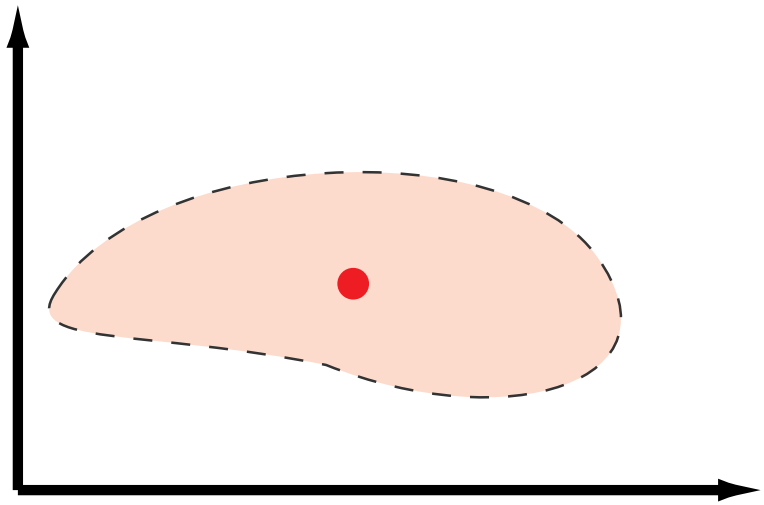
Before optimizing the resource allocation, we need to identify the safe region, i.e., set of resource allocations for which the RAN's constraints are satisfied.

Our Method: **C**ausal **O**nline **L**earning



- Our method (COL) includes four steps:
  1. We define the structure of the RAN in a **causal graph**.
  2. We infer an initial safe region via **causal inference**.
  3. We collect observations through a sequence of **interventions**.
  4. We learn the safe region via **Gaussian process regression**.

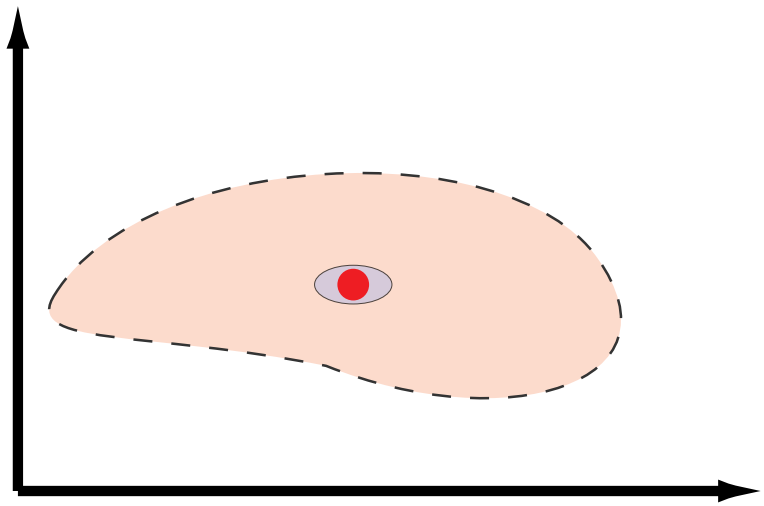
Illustration of **C**ausal **O**nline **L**earning



● *Current operating point* Safe region

- The safe region contains the set of resource allocations for which the operational constraints are satisfied.
- We know that the current operating point is safe, but the **safe region is unknown.**

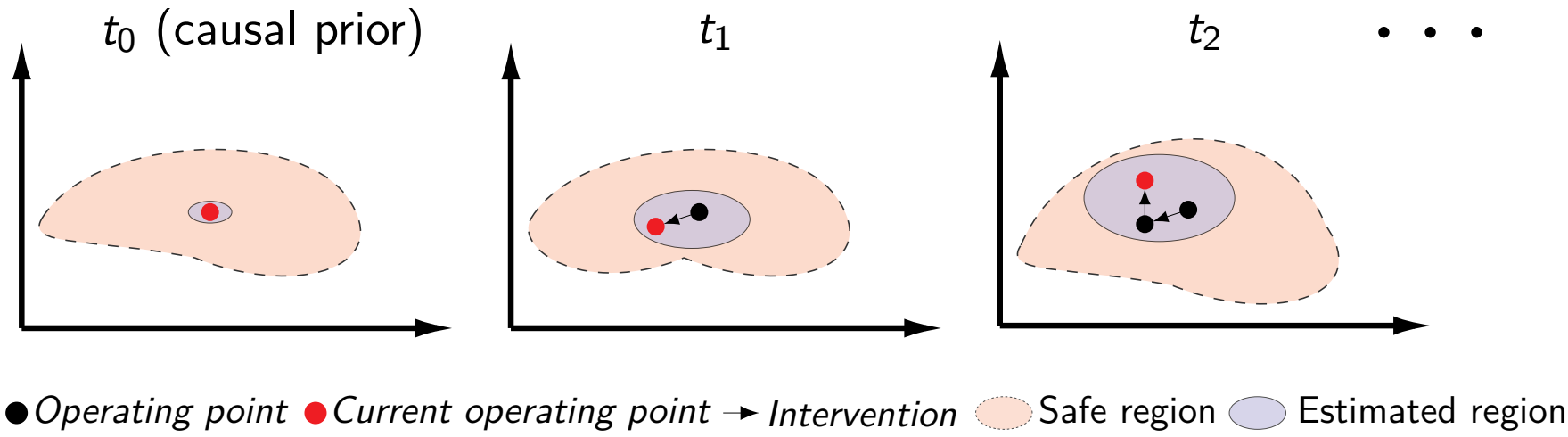
Illustration of **C**ausal **O**nline **L**earning



● *Current operating point* Safe region Estimated region

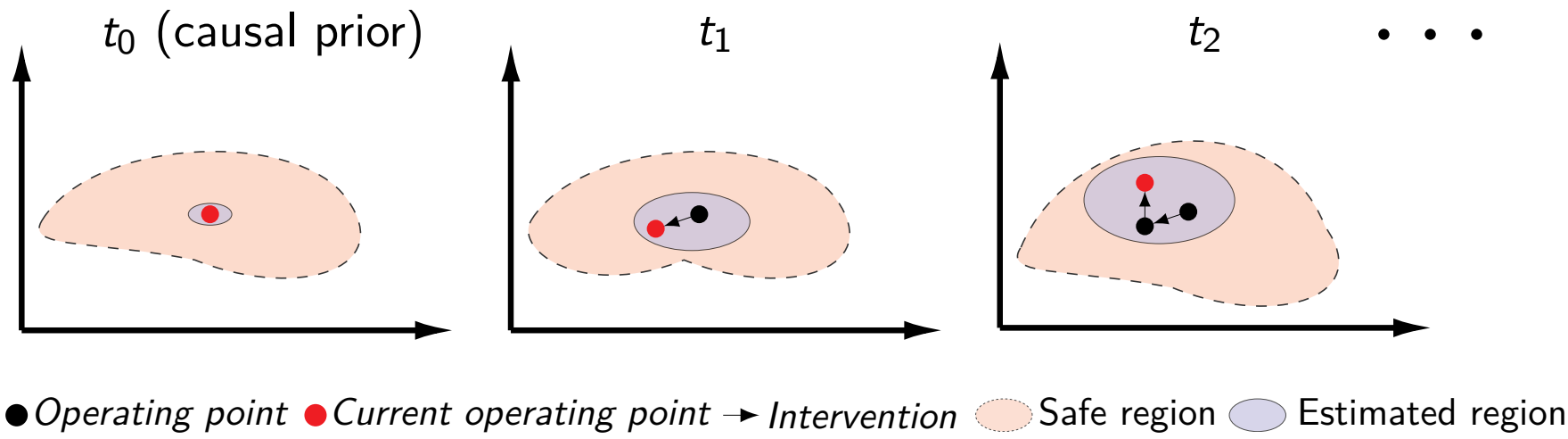
- We initialize the estimated region through causal inference.
- Not all causal effects can be inferred $\Longrightarrow$ initial estimate is typically a small subset of the safe region.

Illustration of **C**ausal **O**nline **L**earning



- We expand the estimated region through a sequence of carefully selected interventions.

Illustration of **C**ausal **O**nline **L**earning



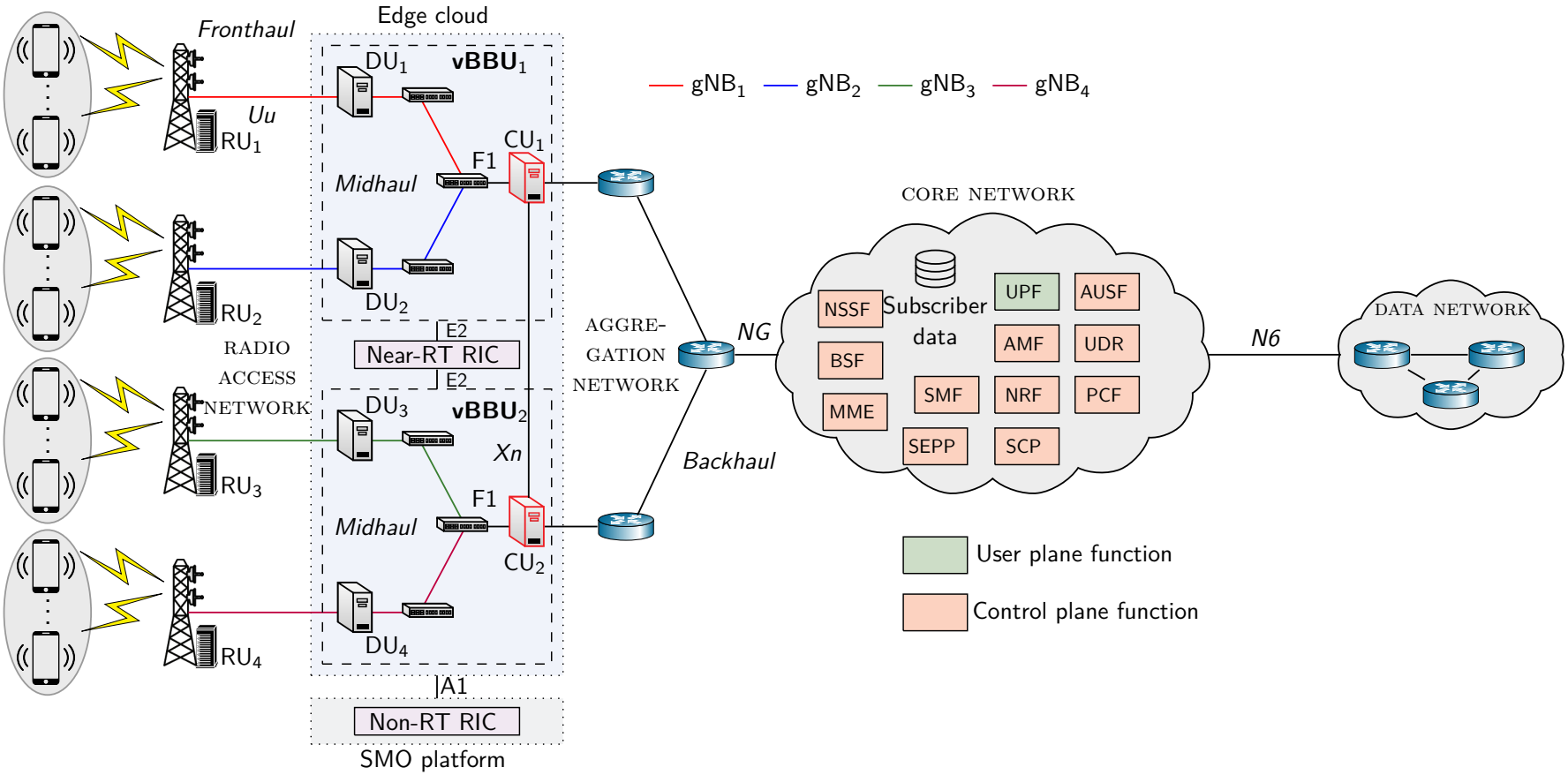
Proposition 1 (Property of our method)

*Given a confidence level $\alpha \in [0, 1)$ and certain regularity conditions, then the estimated safe region is a subset of the true safe region with probability at least α.**

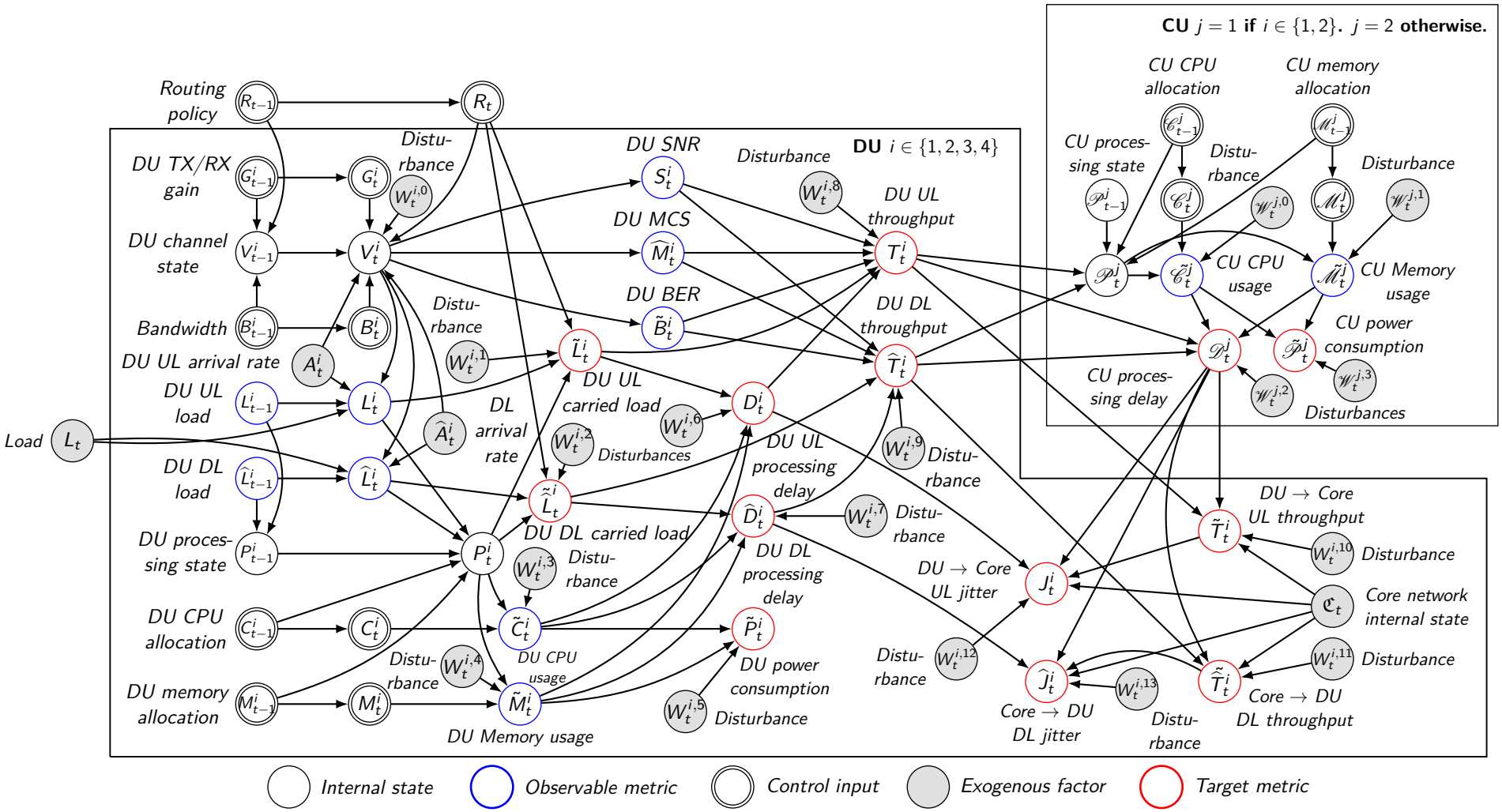
*See our paper for the proof, https://arxiv.org/pdf/2602.05280.

Target RAN for Analysis and Experimental Evaluation

- 5G RAN based on the **O-RAN architecture**.
- Four gNBs distributed into RUs, DUs, and CUs.

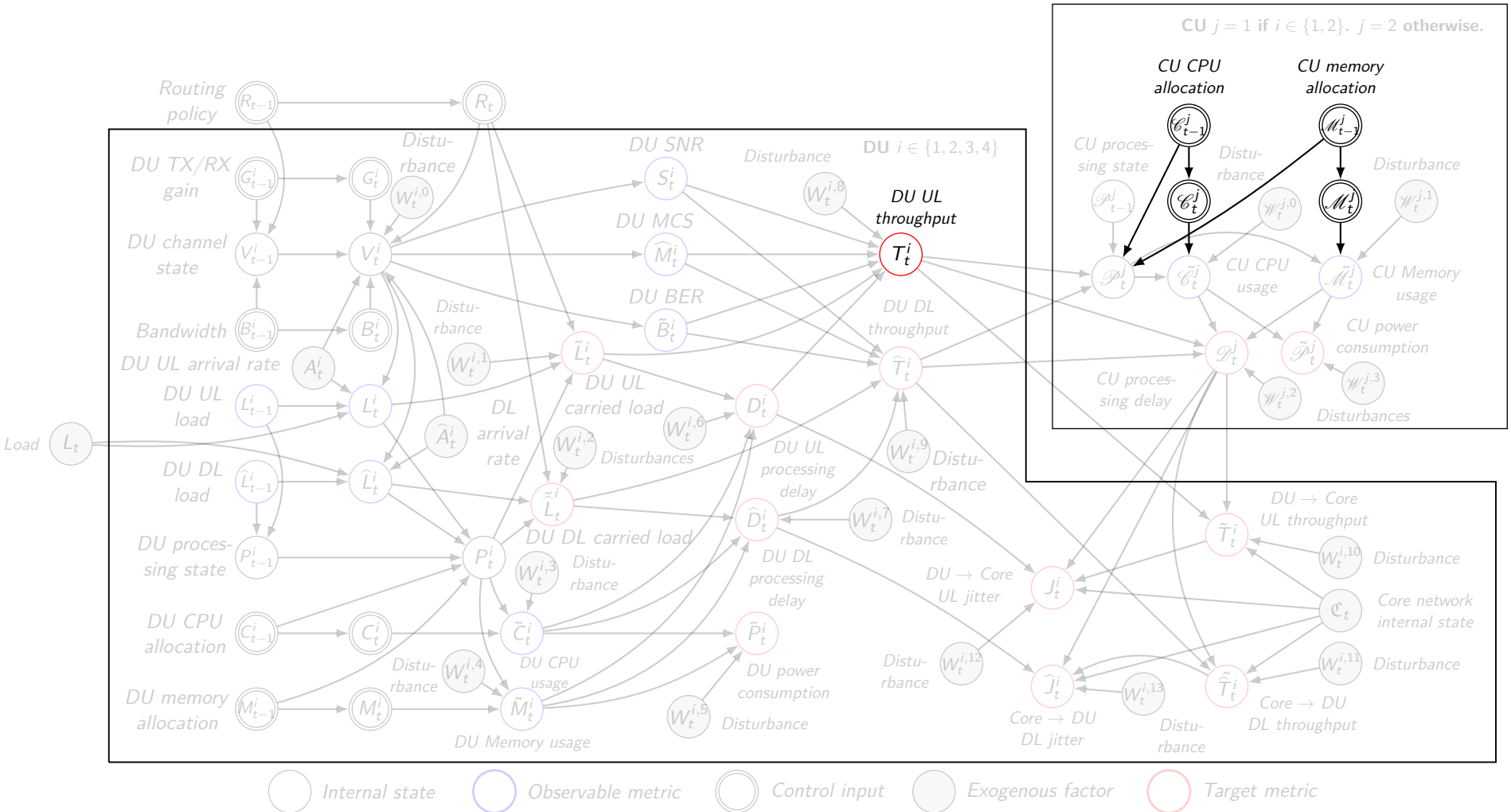


Causal Structure of the Target RAN



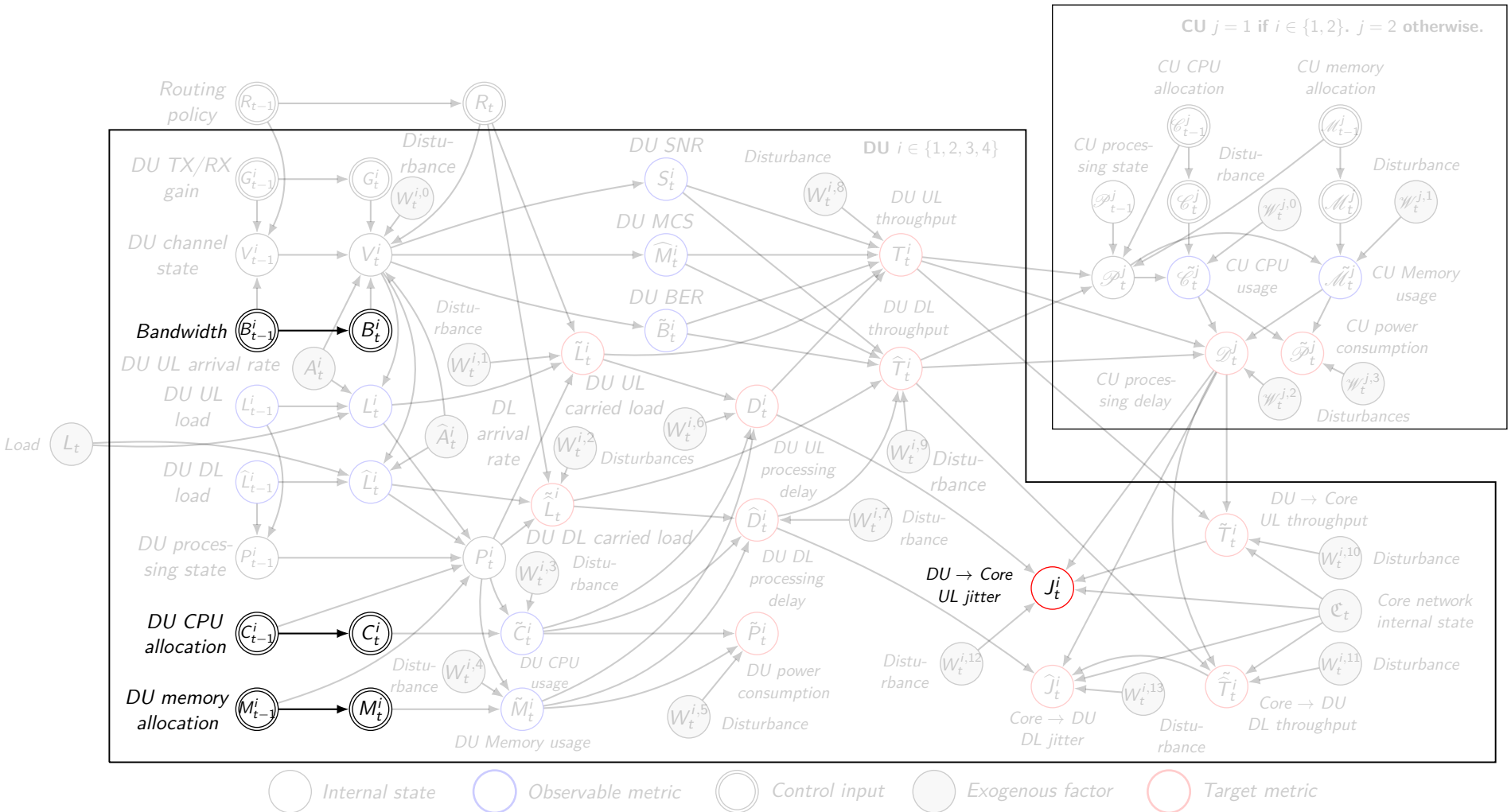
- The graph captures the **causal structure** of the RAN.
  - Specified based on domain knowledge or learned from data.

Causal Structure of the Target RAN



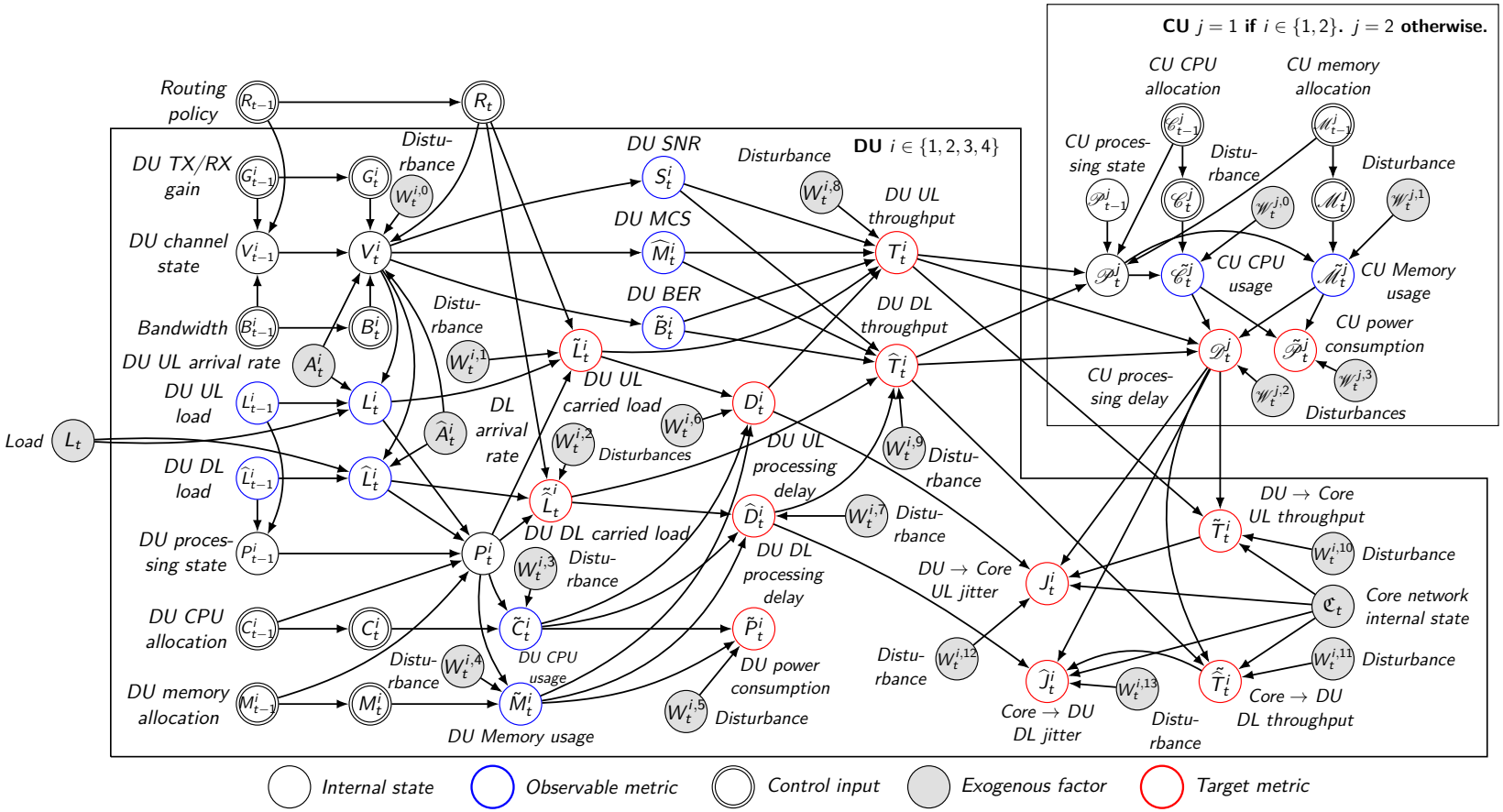
- The graph captures the **causal structure** of the RAN.
  - Specified based on domain knowledge or learned from data.

Causal Structure of the Target RAN



- The graph captures the **causal structure** of the RAN.
  - Specified based on domain knowledge or learned from data.

Causal Structure of the Target RAN



- The **joint probability distribution factorizes** as

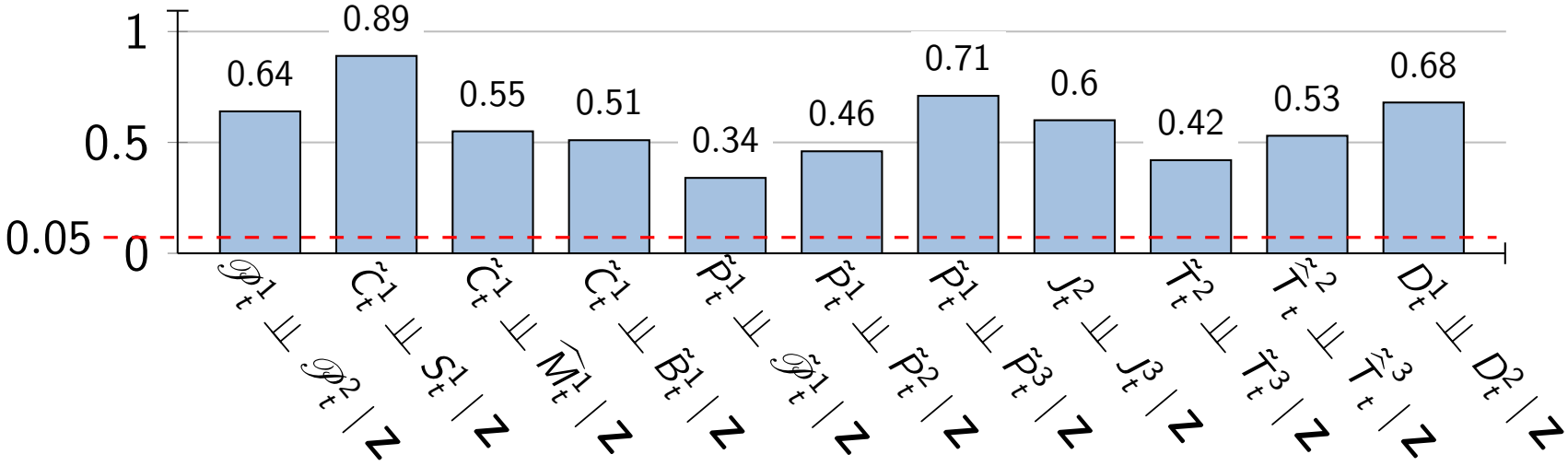
$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{pa}(X_i)),$$

where $\mathrm{pa}(X_i)$ is the set of parents of X_i in the graph.

Validation of the Causal Graph

- The causal graph encodes conditional independence relations.
- We **validate these relations against monitoring data**.

p-value of HSIC independence test based on testbed measurements



- Null hypothesis H_0: the conditional independence holds.
- p-value: $P(\text{data} \mid H_0)$.
- $p > 0.05 \implies$ data is compatible with the causal graph.

Causal Inference

- We leverage the causal graph to infer identifiable causal effects from observational data through **do-calculus**.
- The effect of allocating the CPU $C_t = c_t$ on power consumption P_t can be represented as

$$\mathbb{P}(P_t \mid \mathrm{do}(C_t = c_t)).$$

Causal Inference

- We leverage the causal graph to infer identifiable causal effects from observational data through **do-calculus**.
- The effect of allocating the CPU $C_t = c_t$ on power consumption P_t can be represented as

$$\mathbb{P}(P_t \mid \mathrm{do}(C_t = c_t)).$$

Causal Inference

- We leverage the causal graph to infer identifiable causal effects from observational data through **do-calculus**.
- The effect of allocating the CPU $C_t = c_t$ on power consumption P_t can be represented as

$$\mathbb{P}(P_t \mid \mathrm{do}(C_t = c_t)) \neq \mathbb{P}(P_t \mid C_t = c_t).$$

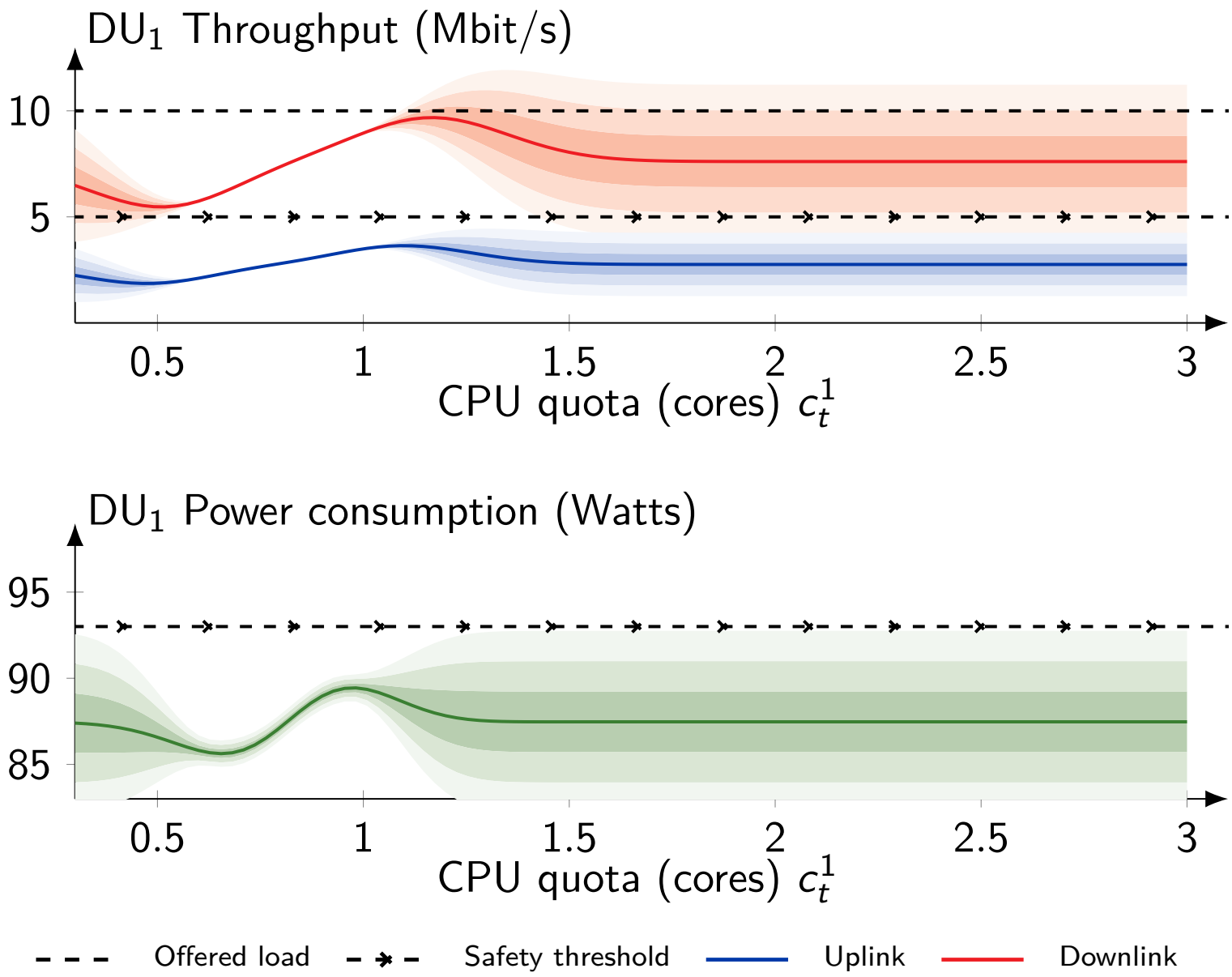
Causal Inference

- We leverage the causal graph to infer identifiable causal effects from observational data through **do-calculus**.

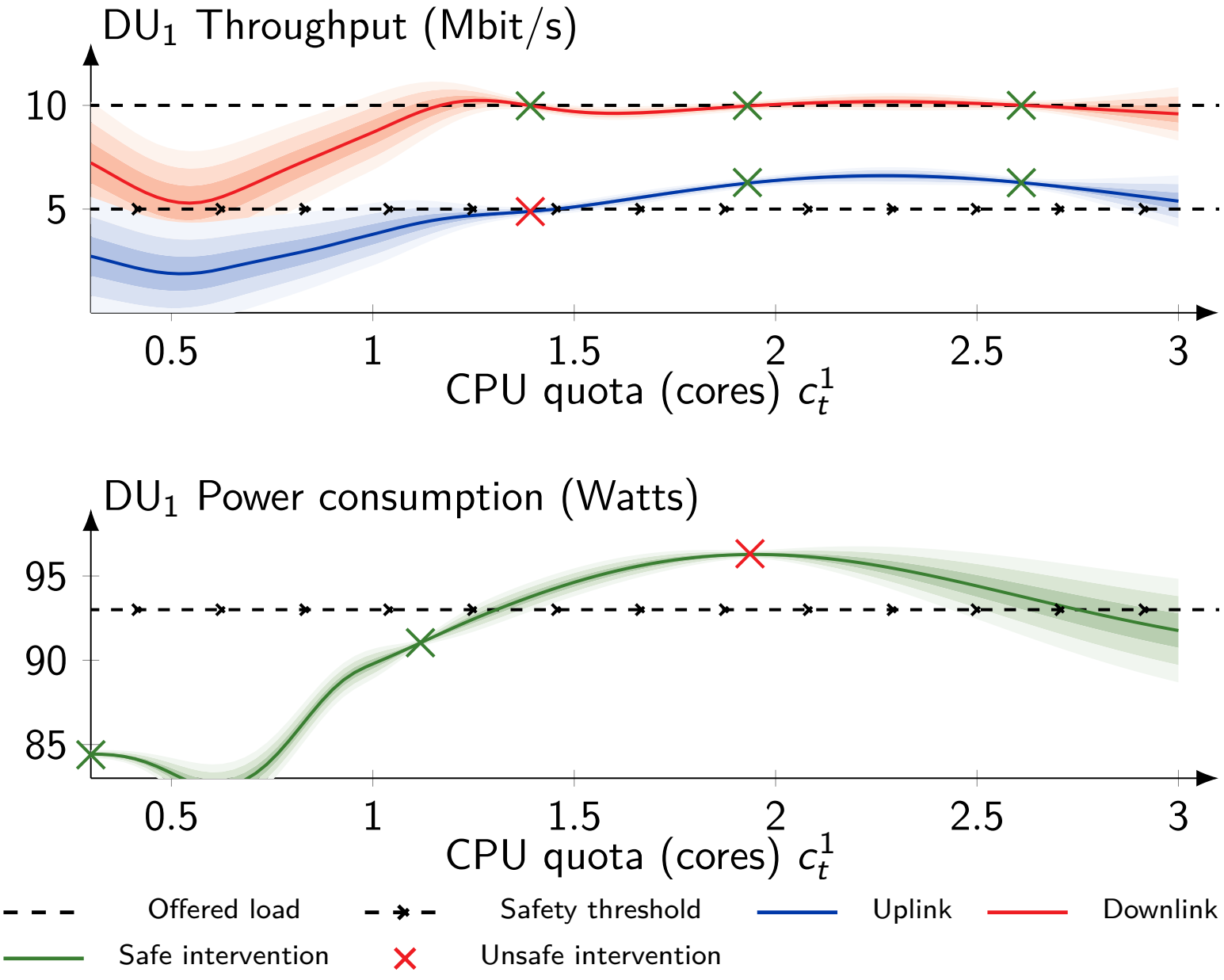
- The effect of allocating the CPU $C_t = c_t$ on power consumption P_t can be derived as

$$
\begin{aligned}
&\mathbb{P}(P_t \mid \mathrm{do}(C_t = c_t)) \\
&= \sum_{c_{t-1}} \mathbb{P}(P_t \mid \mathrm{do}(C_t = c_t), C_{t-1} = c_{t-1}) \mathbb{P}(C_{t-1} = c_{t-1} \mid \mathrm{do}(C_t = c_t)) \\
&= \sum_{c_{t-1}} \mathbb{P}(P_t \mid C_t = c_t, C_{t-1} = c_{t-1}) \mathbb{P}(C_{t-1} = c_{t-1} \mid \mathrm{do}(C_t = c_t)) \\
&= \sum_{c_{t-1}} \mathbb{P}(P_t \mid C_t = c_t, C_{t-1} = c_{t-1}) \mathbb{P}(C_{t-1} = c_{t-1}).
\end{aligned}
$$

Inferred Causal Effects on the Testbed



Interventional Learning of Causal Effects on the Testbed

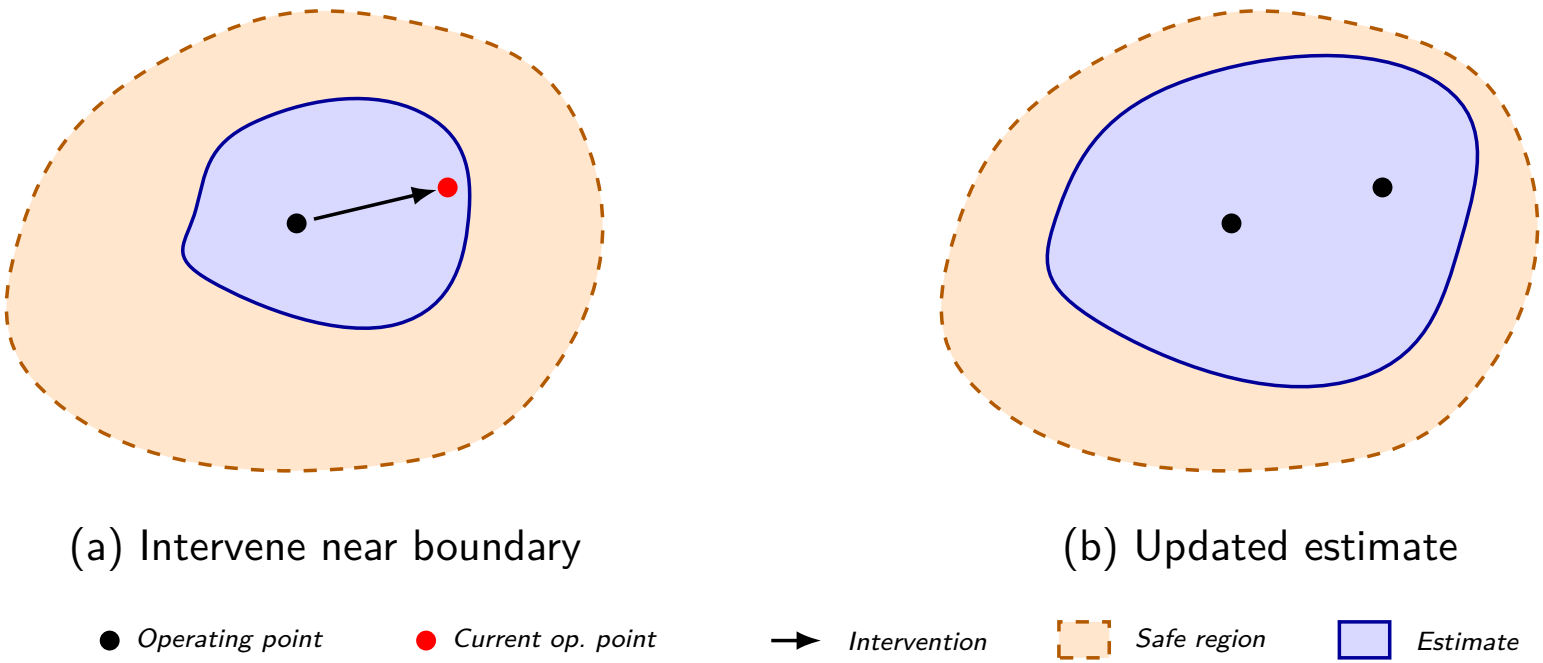


Intervention Policy

- We select interventions according to the policy

$$\pi(\mathcal{S}) \in \arg\max_{u \in \mathcal{S}} \left\{ \frac{\sqrt{k(u,u)}}{c(u)} \right\}, \tag{1}$$

where $\mathcal{S}$ is the current estimate of the safe region, u is a control input, $c(u)$ captures the cost of the intervention, and $k(u,u)$ is the covariance function of the Gaussian process, which captures the information gain of the intervention.



(a) Intervene near boundary

(b) Updated estimate

Intervention Policy

- We select interventions according to the policy

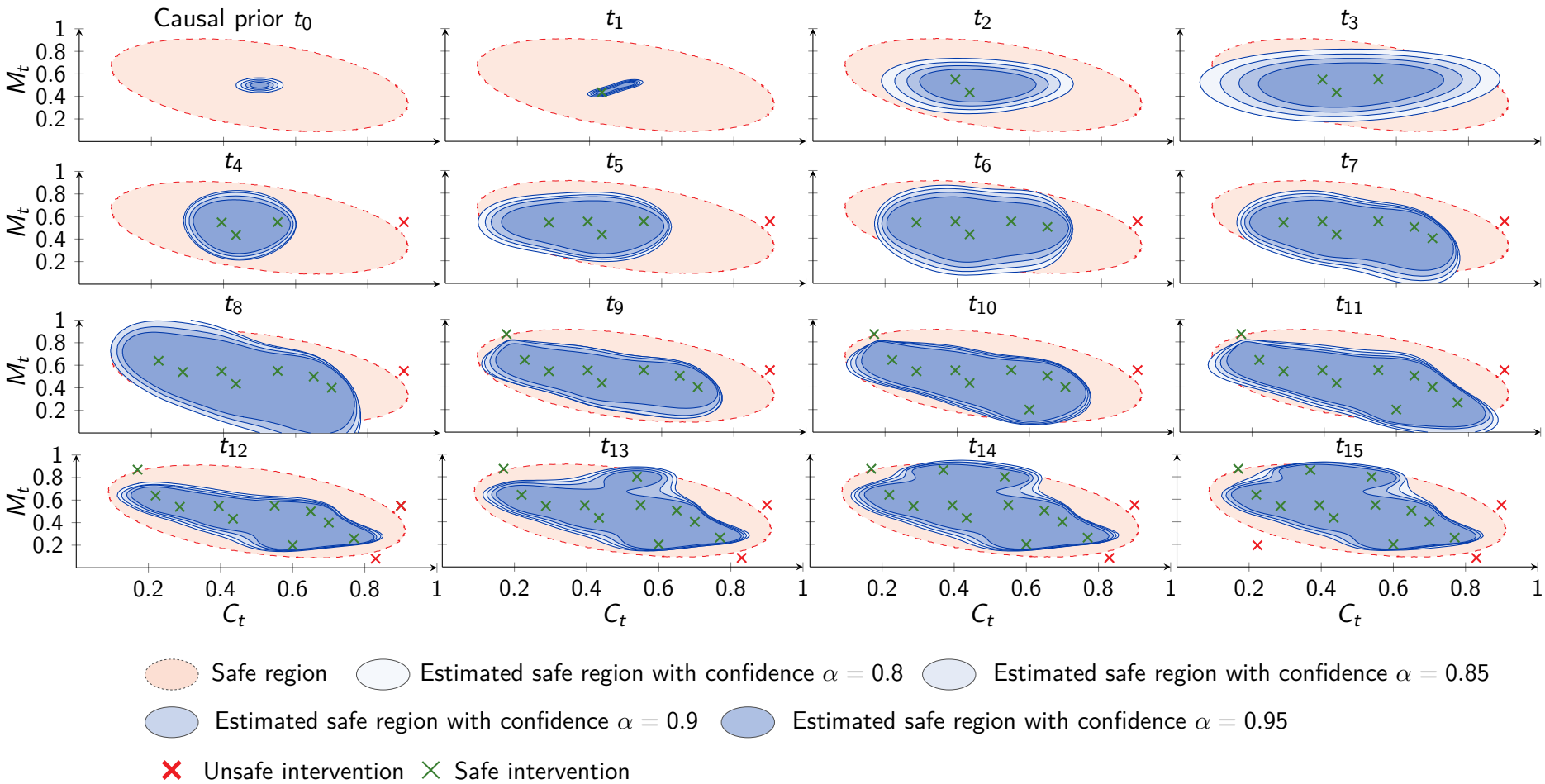
$$\pi(\mathcal{S}) \in \arg\max_{u \in \mathcal{S}} \left\{ \frac{\sqrt{k(u,u)}}{c(u)} \right\}, \tag{2}$$

where $\mathcal{S}$ is the current estimate of the safe region, u is a control input, $c(u)$ captures the cost of the intervention, and $k(u,u)$ is the covariance function of the Gaussian process, which captures the information gain of the intervention.

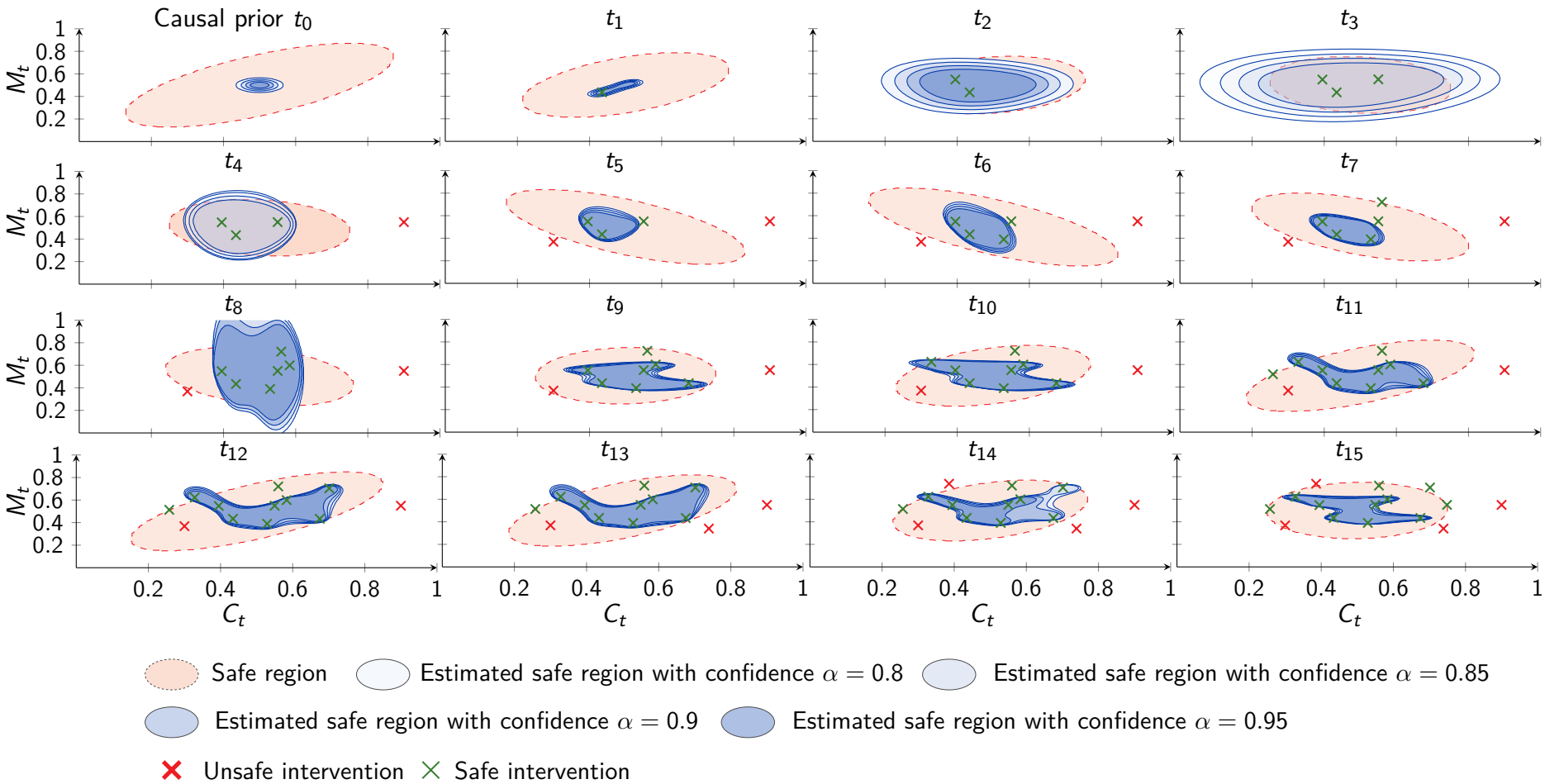
Tradeoff

The intervention policy π captures a tradeoff between safety, exploration, and cost.

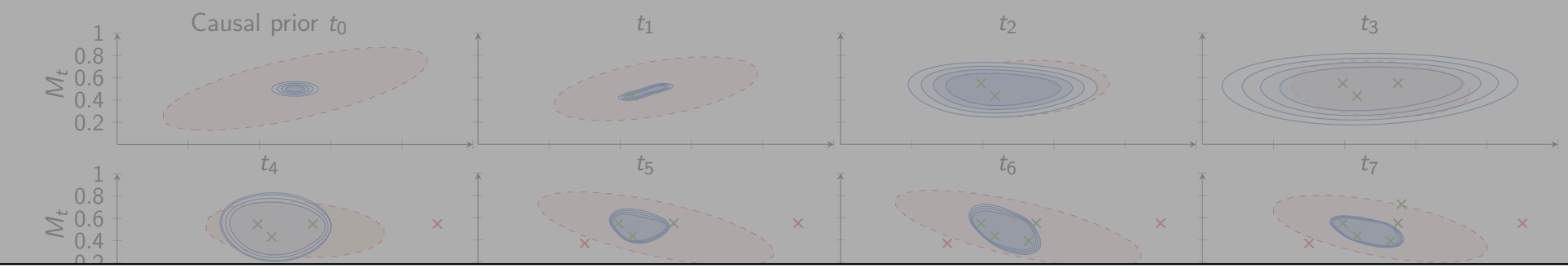
Interventional Learning of a **Static** Safe Region



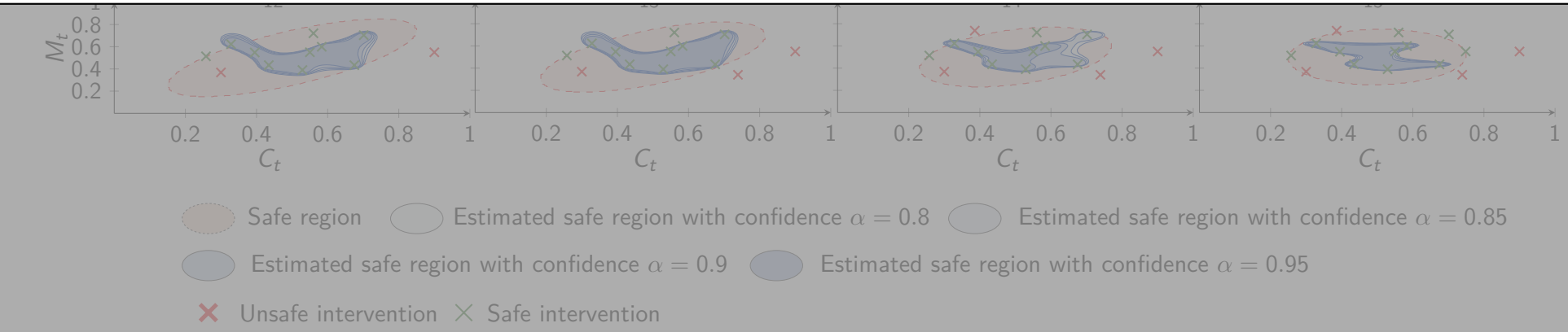
Interventional Learning of a **Dynamic** Safe Region



Interventional Learning of a **Dynamic** Safe Region



For additional results, see https://arxiv.org/pdf/2602.05280



Our Method: Causal Online Learning (COL)

Learn or encode the causal structure of the RAN

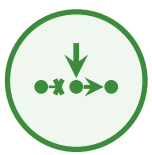
A Cloud RAN is not a black box. The causal graph allows us to encode domain knowledge and engineering intuition.

Validate the causal structure against operational data

The causal graph encodes statistical independence assumptions. These assumptions can be validated against monitoring data.

Infer an initial safe region through causal inference

The causal graph allows us to safely infer the causal effects of some control inputs without having to intervene on the system.



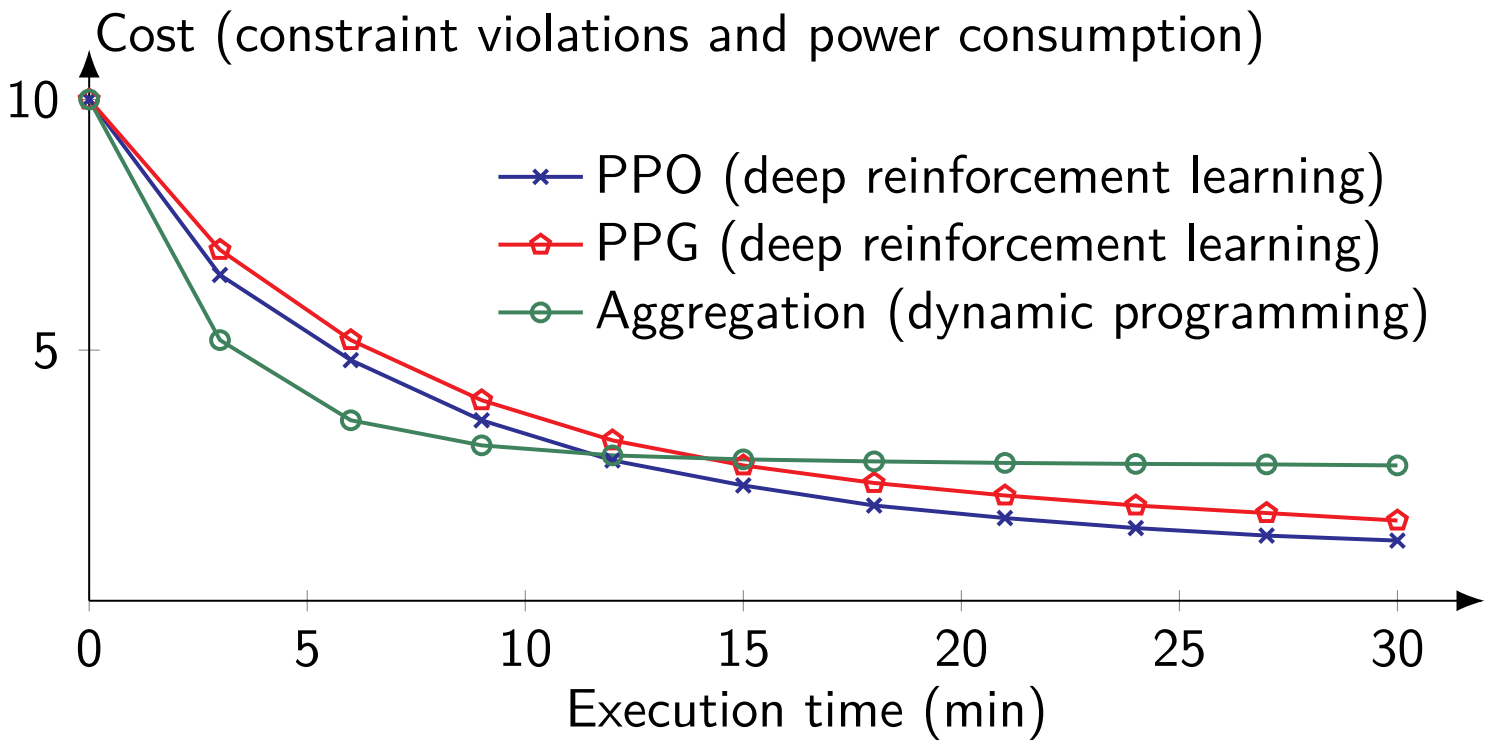
Expand the safe region through interventional learning

We intervene on the system to learn the causal effects of new control inputs, which allows us to refine the safe region through Bayesian learning with Gaussian processes.

Optimization of the Resource Allocation Strategy

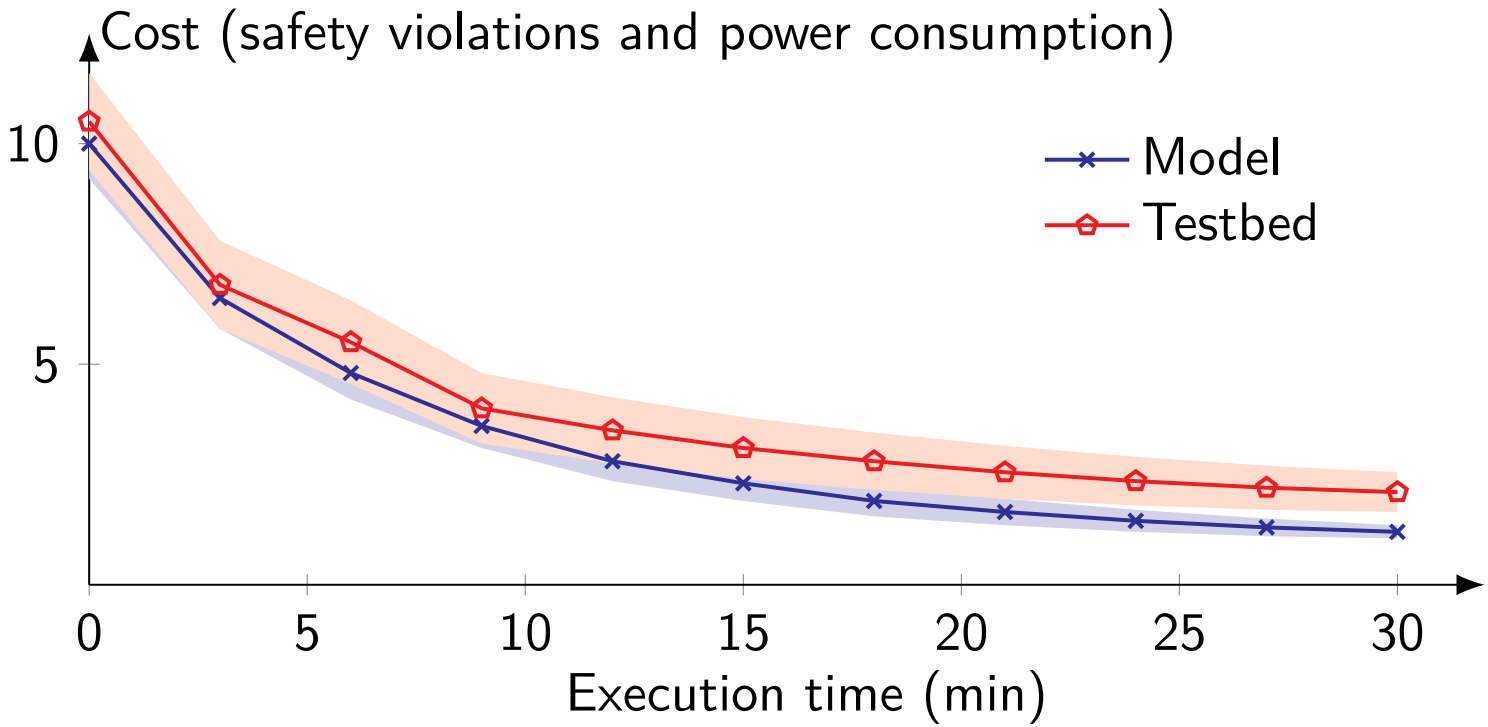
- The learned causal model of the safe region allows us to optimize the resource allocation strategy.

- We design the strategy to to minimize power consumption while keeping the throughput above a threshold.

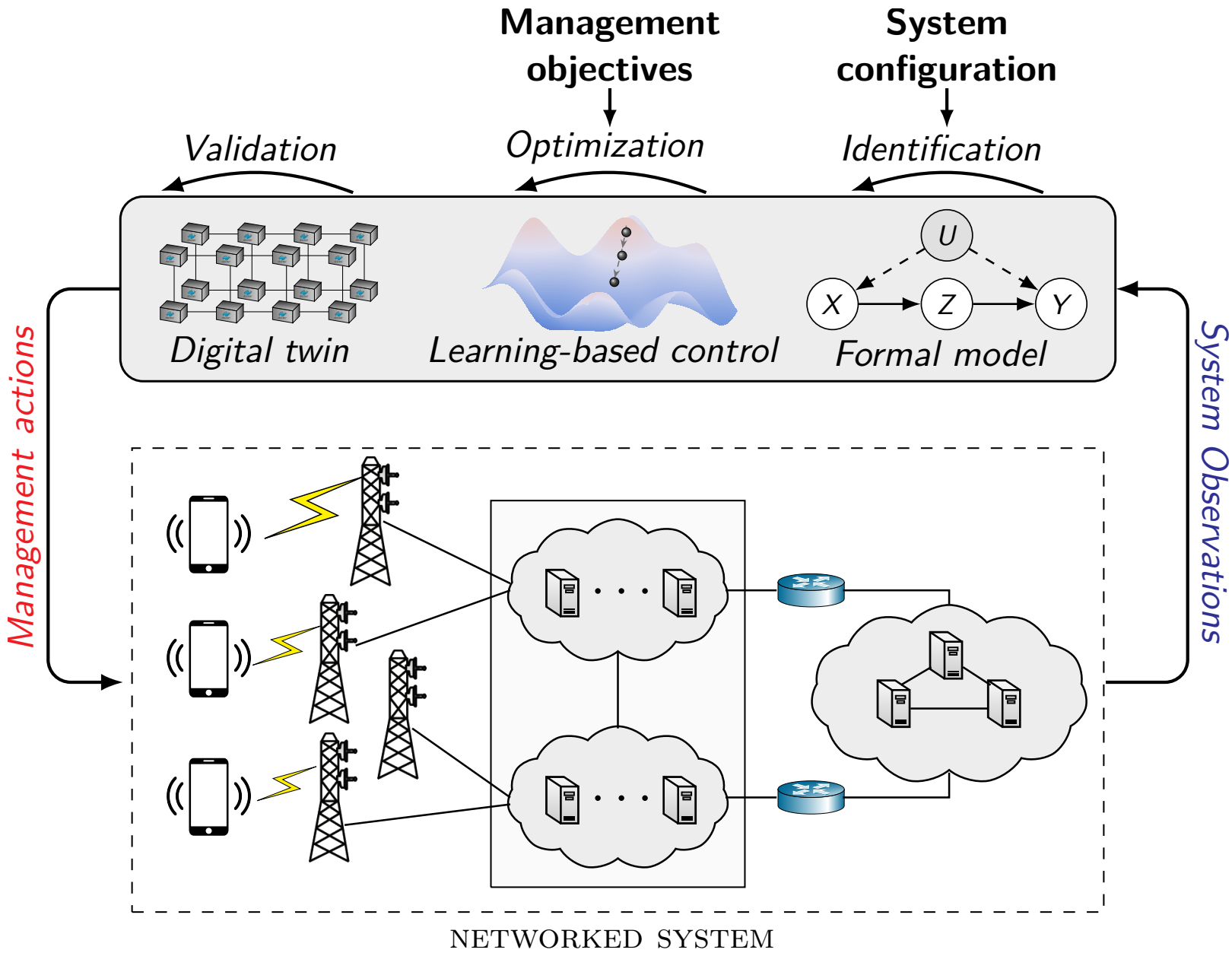


Validating the Control Strategy on the Testbed

- We validate the optimized control strategy on the testbed.
- Curves show the mean and variance from 5 evaluations.



Learning-Based Control of Networked Systems

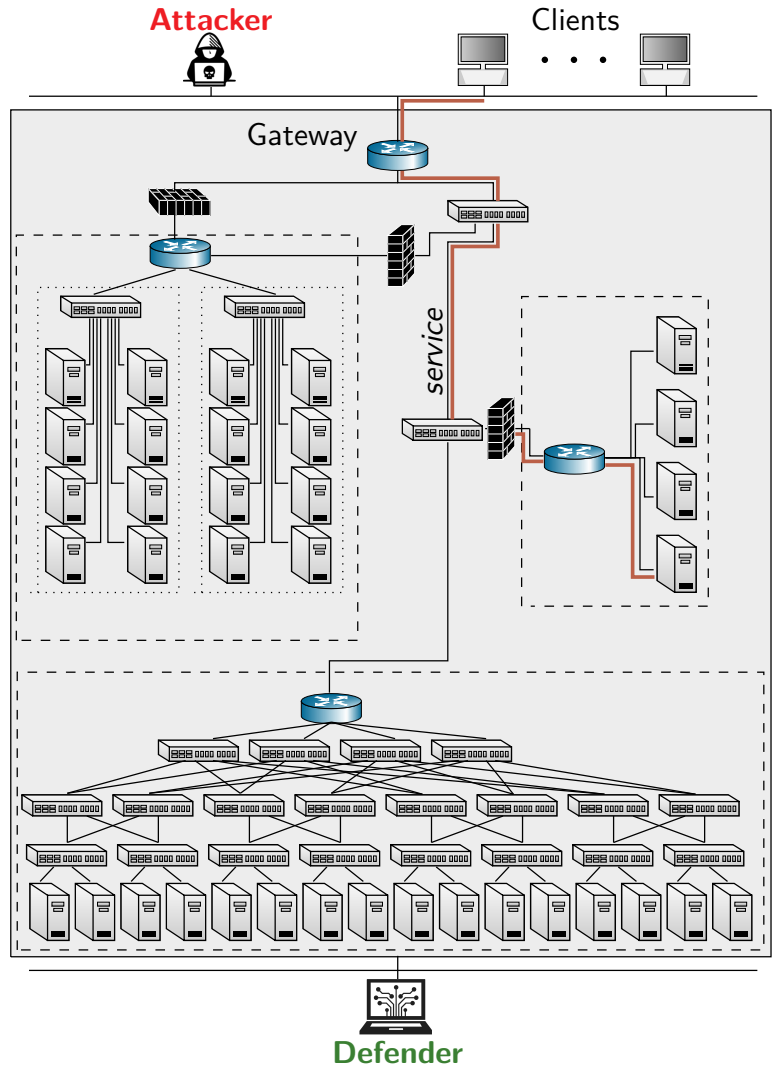


Outline

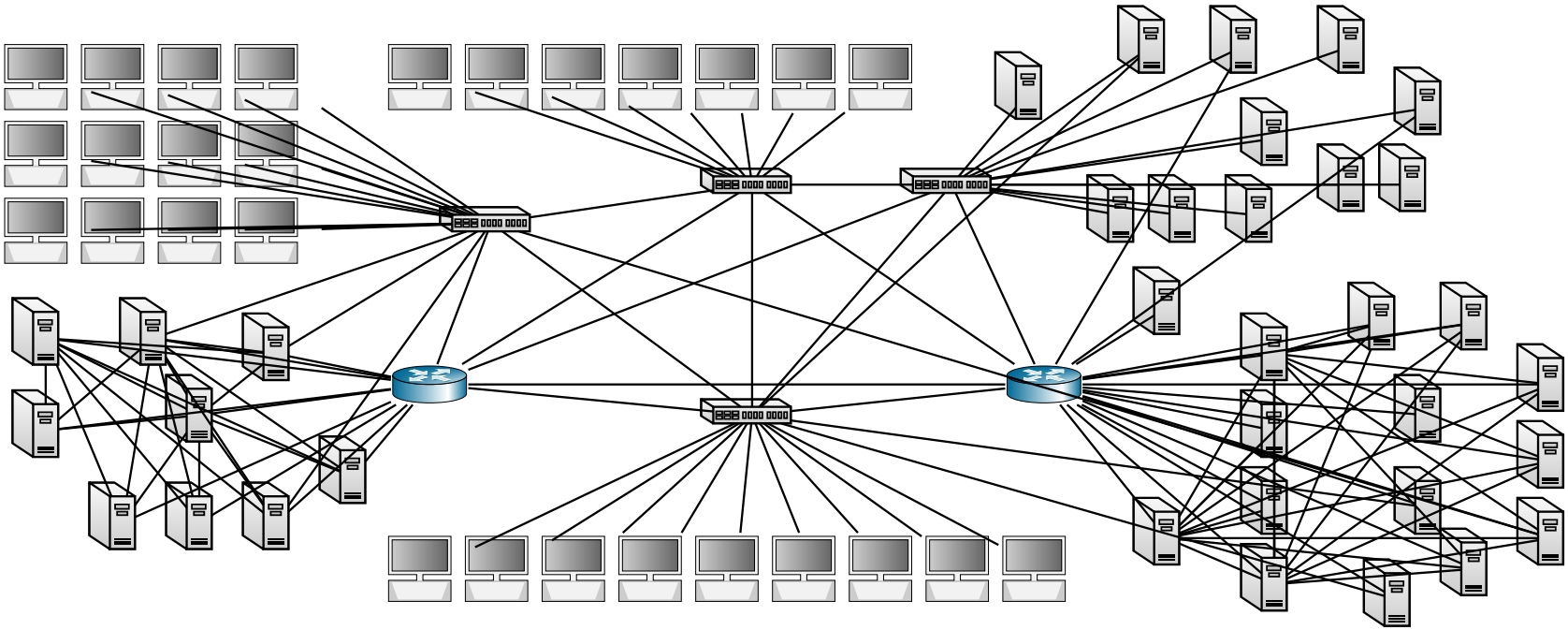
- **Autonomous management as a control problem.**
  - Problem formulation.
  - Challenges.
- **Causal online learning of safe regions.**
  - Causal inference.
  - Interventional learning.
- **Game-theoretic learning of security strategies.**
  - Game-theoretic model and equilibrium analysis.
  - Conjectural online learning.
- **Multiagent incident response with LLMs.**
  - System architecture.
  - Theoretical analysis.
- **Conclusion.**

Use Case: Strategic Defense Against Network Intrusions

- A **defender** owns an infrastructure.
  - Defends the infrastructure by monitoring and response.
  - Has partial observability.
- An **attacker** seeks to intrude on the infrastructure.
  - Wants to compromise specific components.
  - Attacks by reconnaissance, exploitation and pivoting.

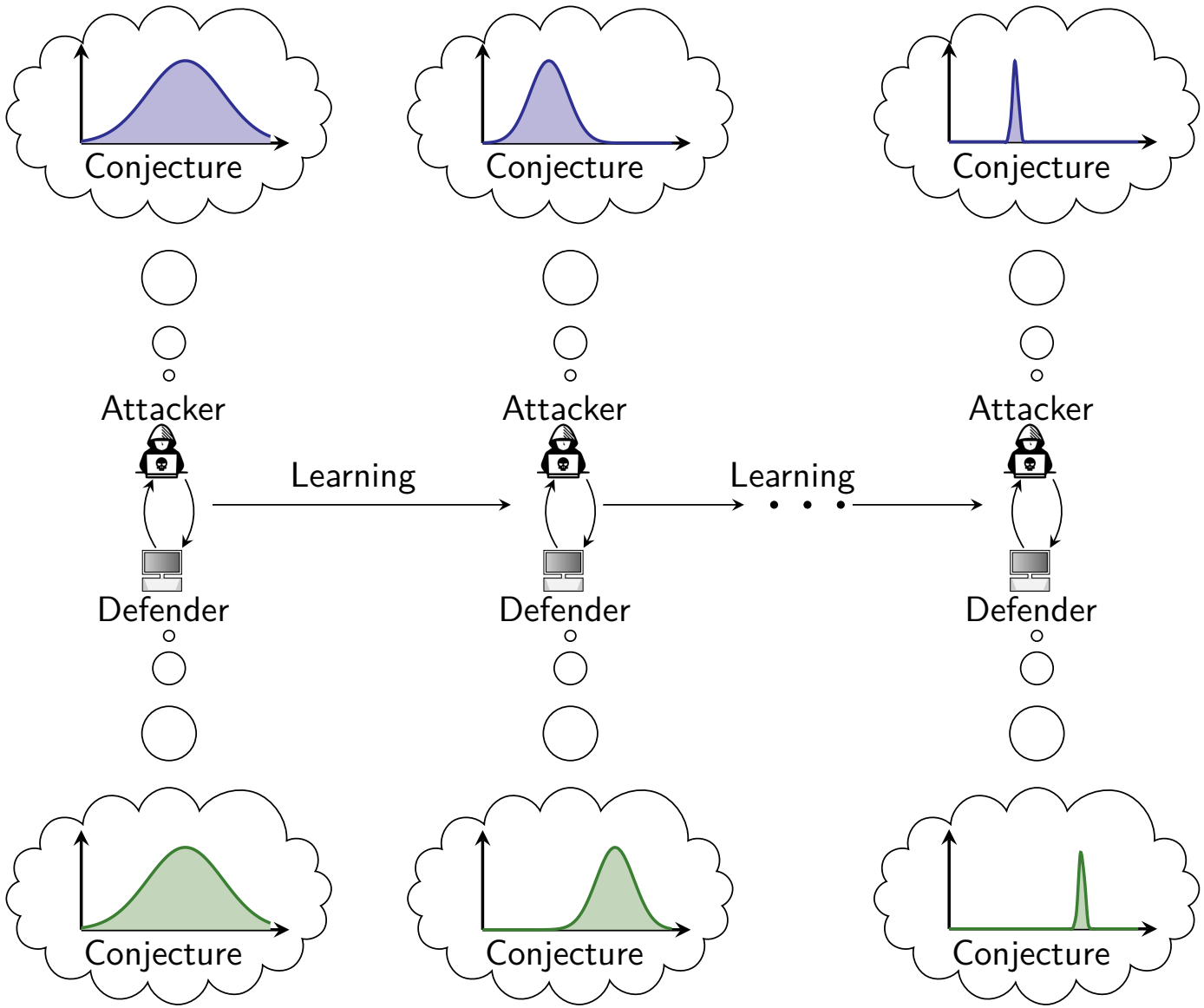


Challenge: Networked Systems are Complex

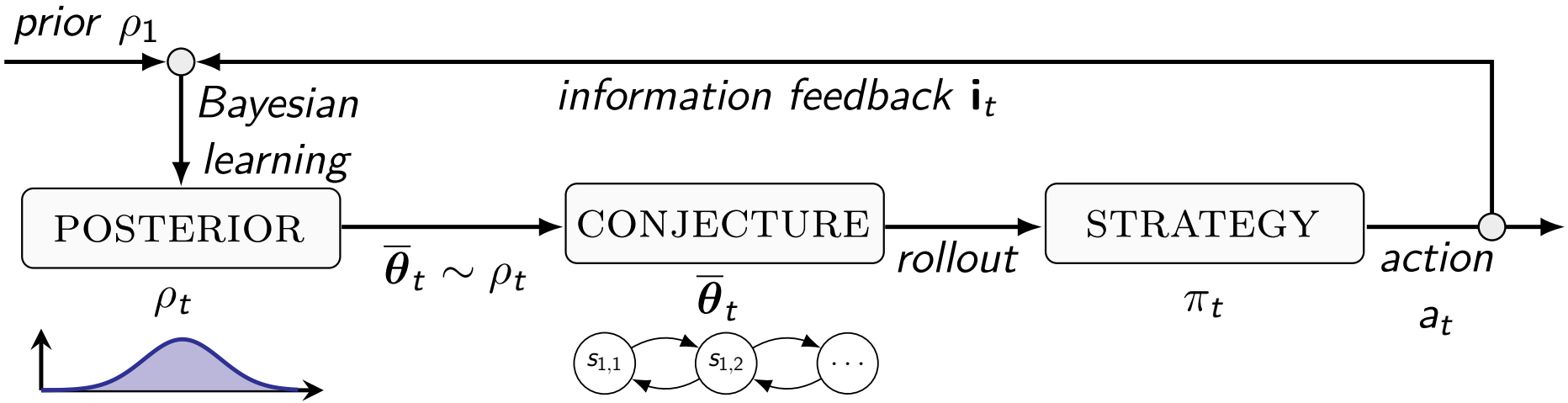


- It is not realistic that any model will capture all the details.
  - $\Longrightarrow$ We have to work with **approximate models**.
  - $\Longrightarrow$ model misspecification.
- How does misspecification affect optimality and convergence?

Method: Conjectural Online Learning

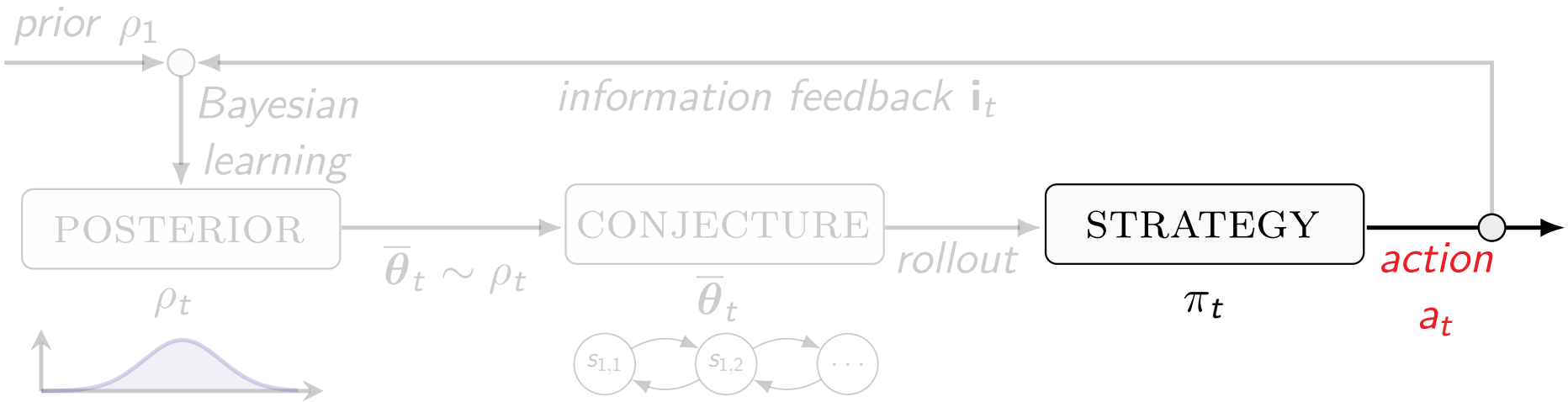


Method: Conjectural Online Learning



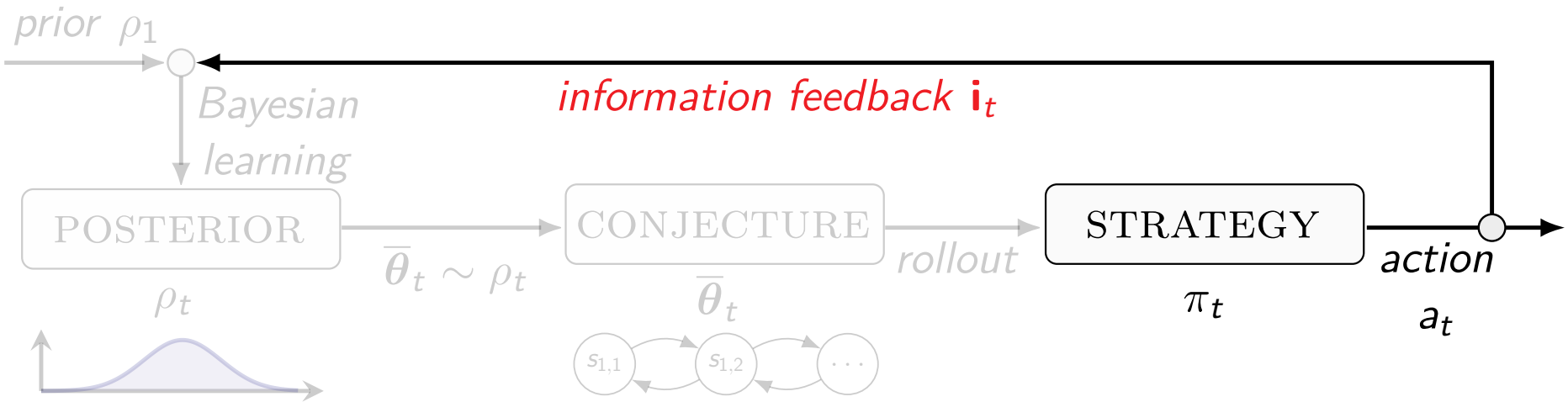
- The **model parameters** are denoted by $\boldsymbol{\theta}$.
- The defender has a conjecture $\overline{\boldsymbol{\theta}} \sim \rho_t \in \Delta(\Theta)$.
- The defender is misspecified if $\boldsymbol{\theta} \notin \Theta$.

Method: Conjectural Online Learning



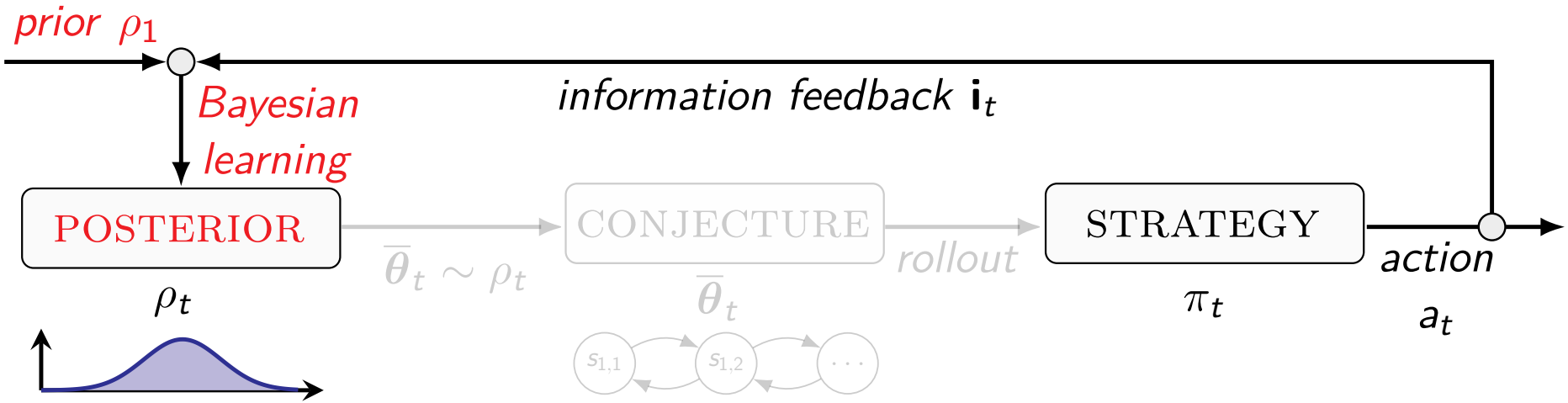
- The **model parameters** are denoted by $\boldsymbol{\theta}$.
- The defender has a conjecture $\overline{\boldsymbol{\theta}} \sim \rho_t \in \Delta(\Theta)$.
- The defender is misspecified if $\boldsymbol{\theta} \notin \Theta$.

Method: Conjectural Online Learning



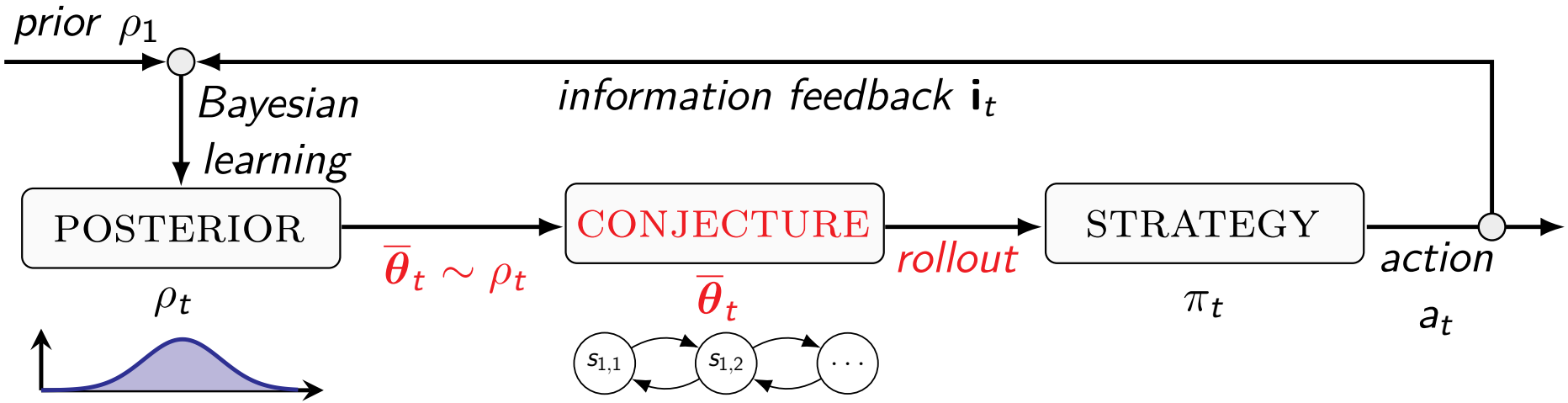
- The **model parameters** are denoted by $\boldsymbol{\theta}$.
- The defender has a conjecture $\overline{\boldsymbol{\theta}} \sim \rho_t \in \Delta(\Theta)$.
- The defender is misspecified if $\boldsymbol{\theta} \notin \Theta$.

Method: Conjectural Online Learning



- The **model parameters** are denoted by $\boldsymbol{\theta}$.
- The defender has a conjecture $\overline{\boldsymbol{\theta}} \sim \rho_t \in \Delta(\Theta)$.
- The defender is misspecified if $\boldsymbol{\theta} \notin \Theta$.

Method: Conjectural Online Learning

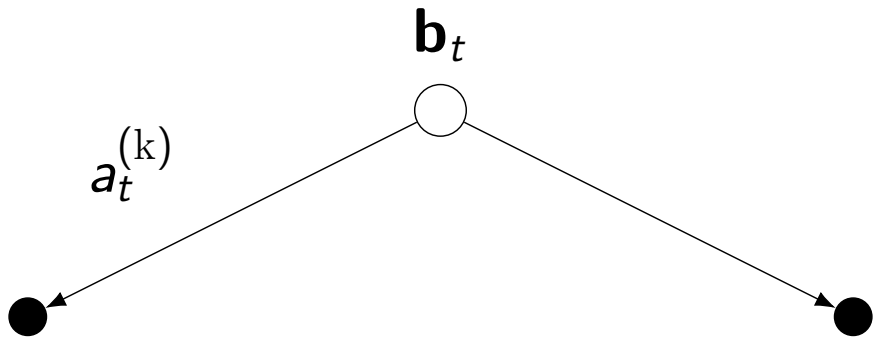


- The **model parameters** are denoted by $\boldsymbol{\theta}$.
- The defender has a conjecture $\overline{\boldsymbol{\theta}} \sim \rho_t \in \Delta(\Theta)$.
- The defender is misspecified if $\boldsymbol{\theta} \notin \Theta$.

Strategy Adaptation through Conjectural Rollout

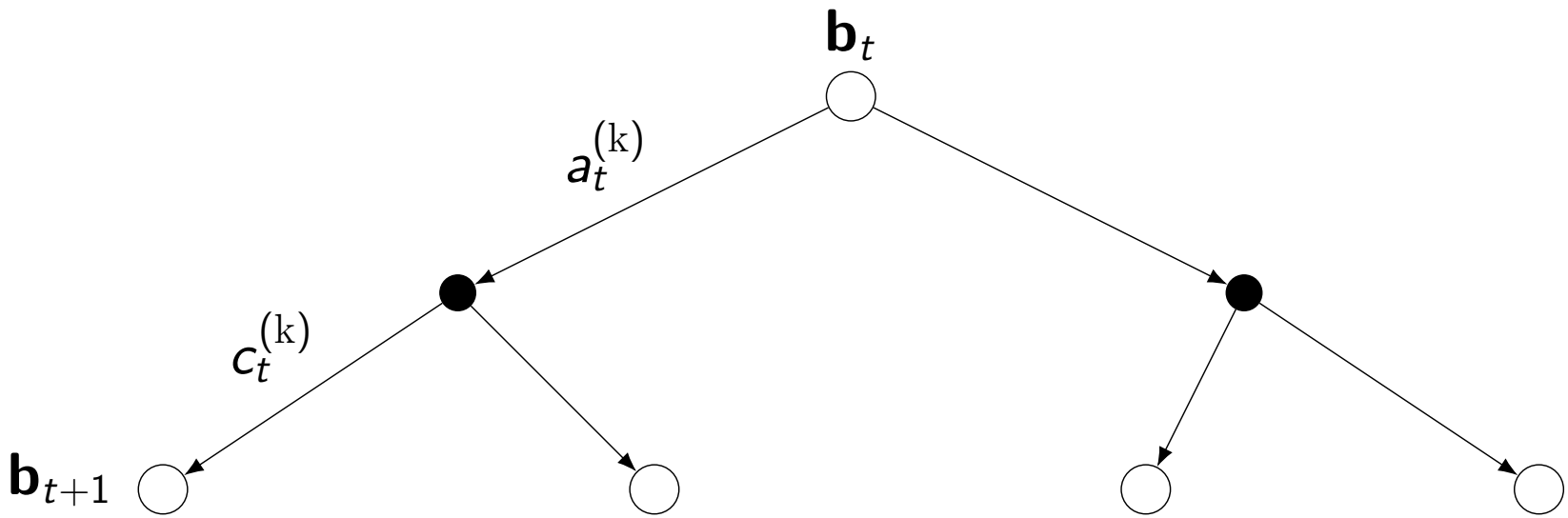
Strategy Adaptation through Conjectural Rollout

Conjectured lookahead tree of player k.

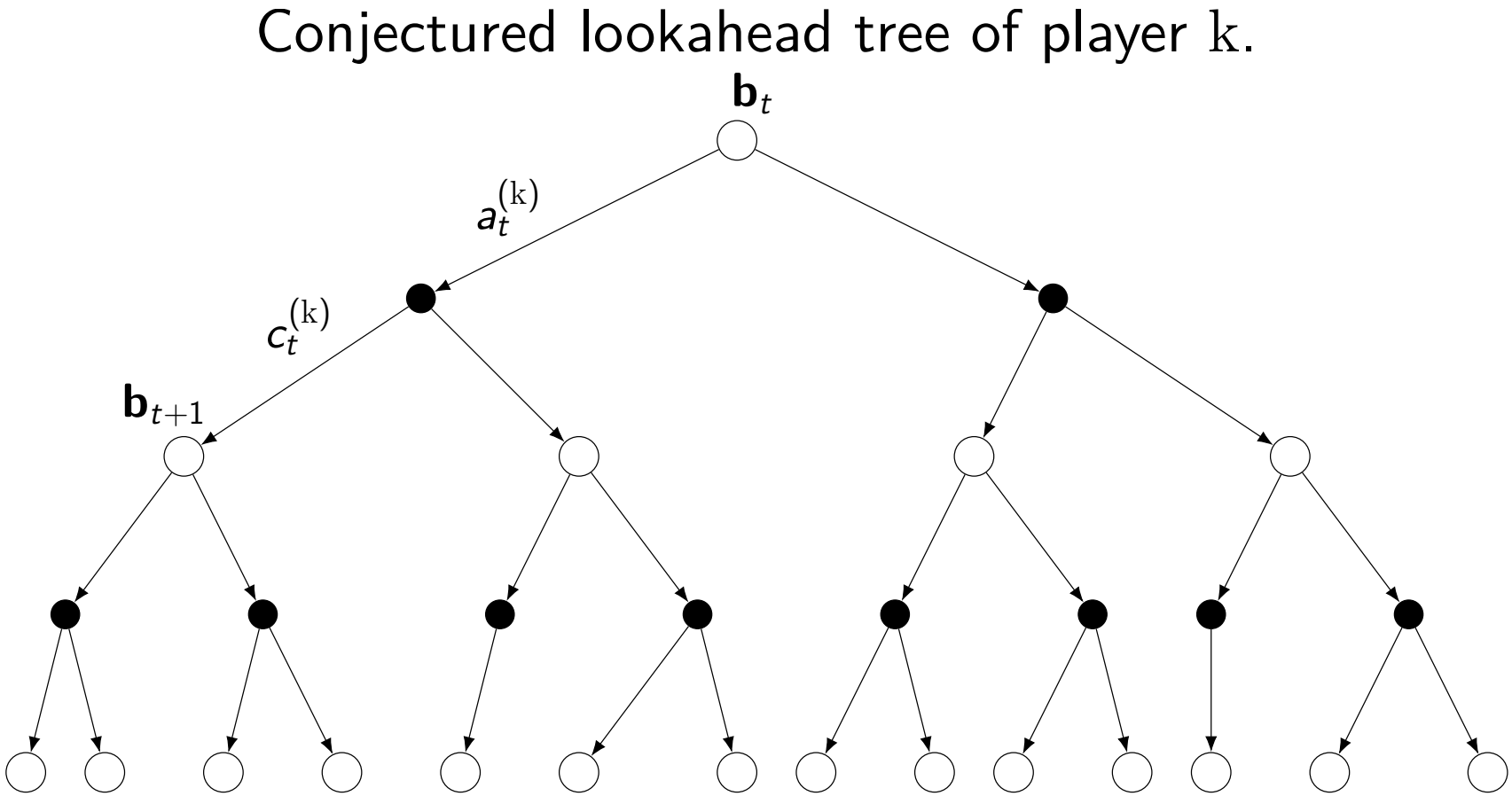


Strategy Adaptation through Conjectural Rollout

Conjectured lookahead tree of player k.

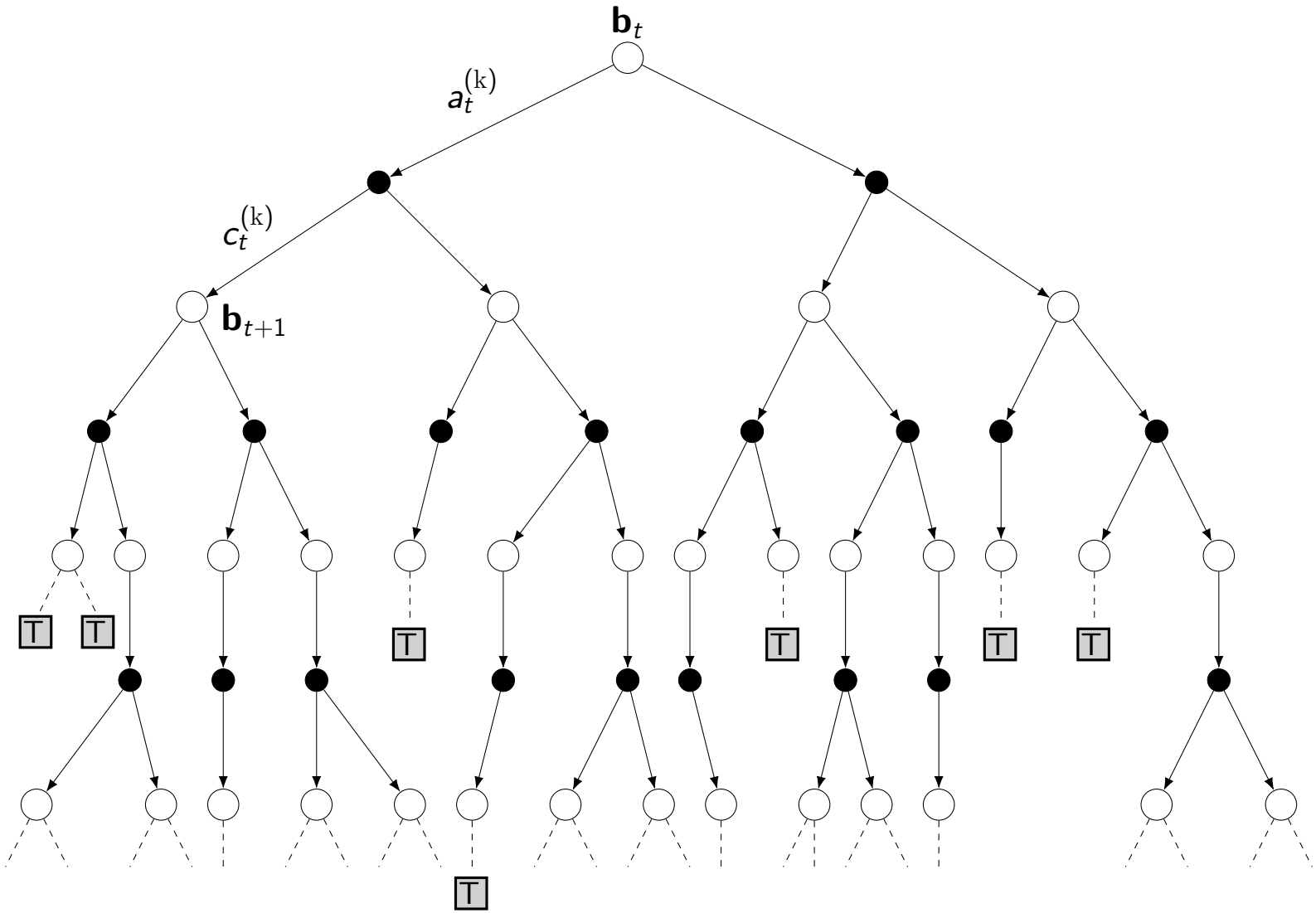


Strategy Adaptation through Conjectural Rollout



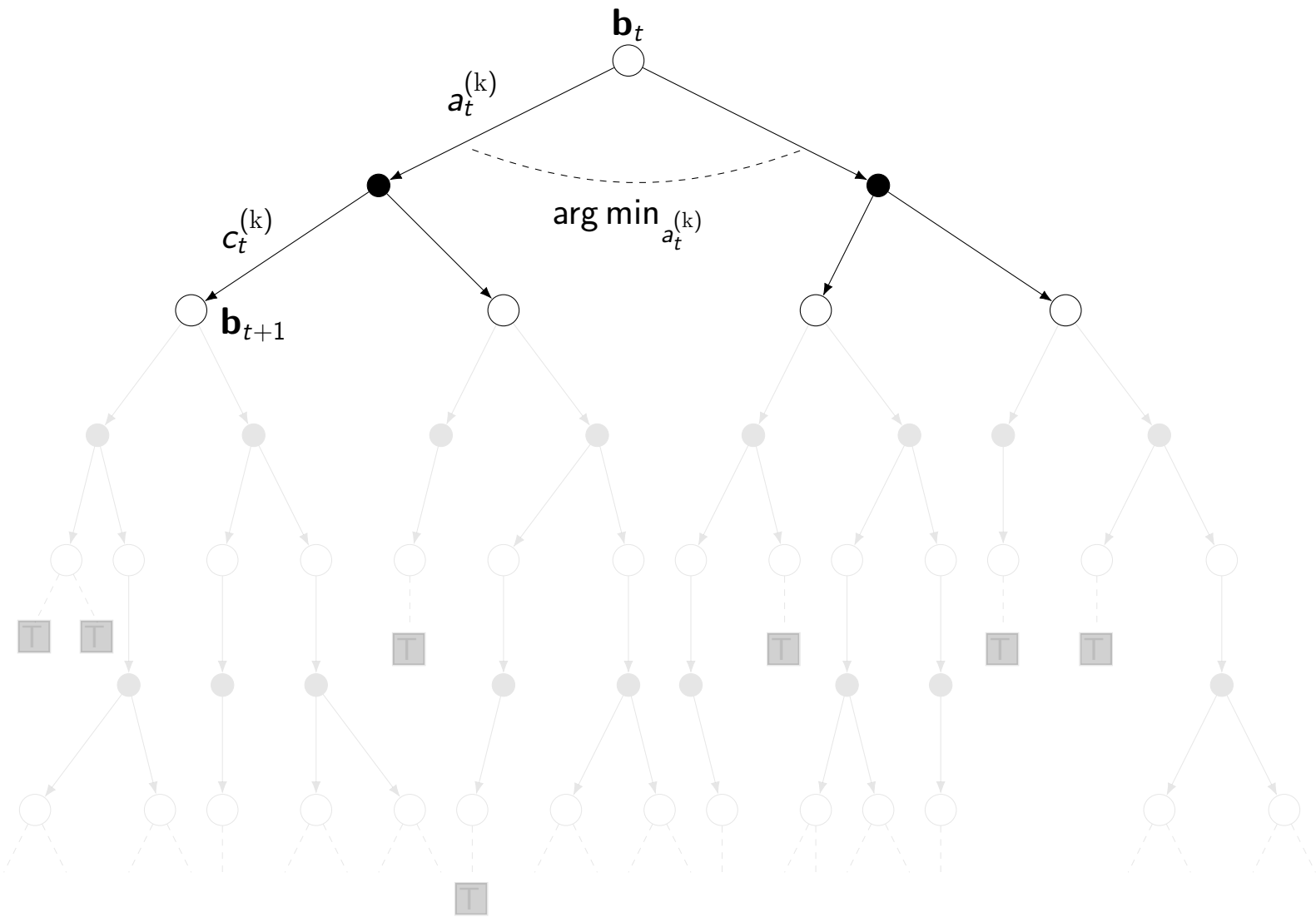
Strategy Adaptation through Conjectural Rollout

Conjectured lookahead tree of player k



Strategy Adaptation through Conjectural Rollout

ℓ_{k}-step rollout.



Theoretical Guarantees of COL (Informal)

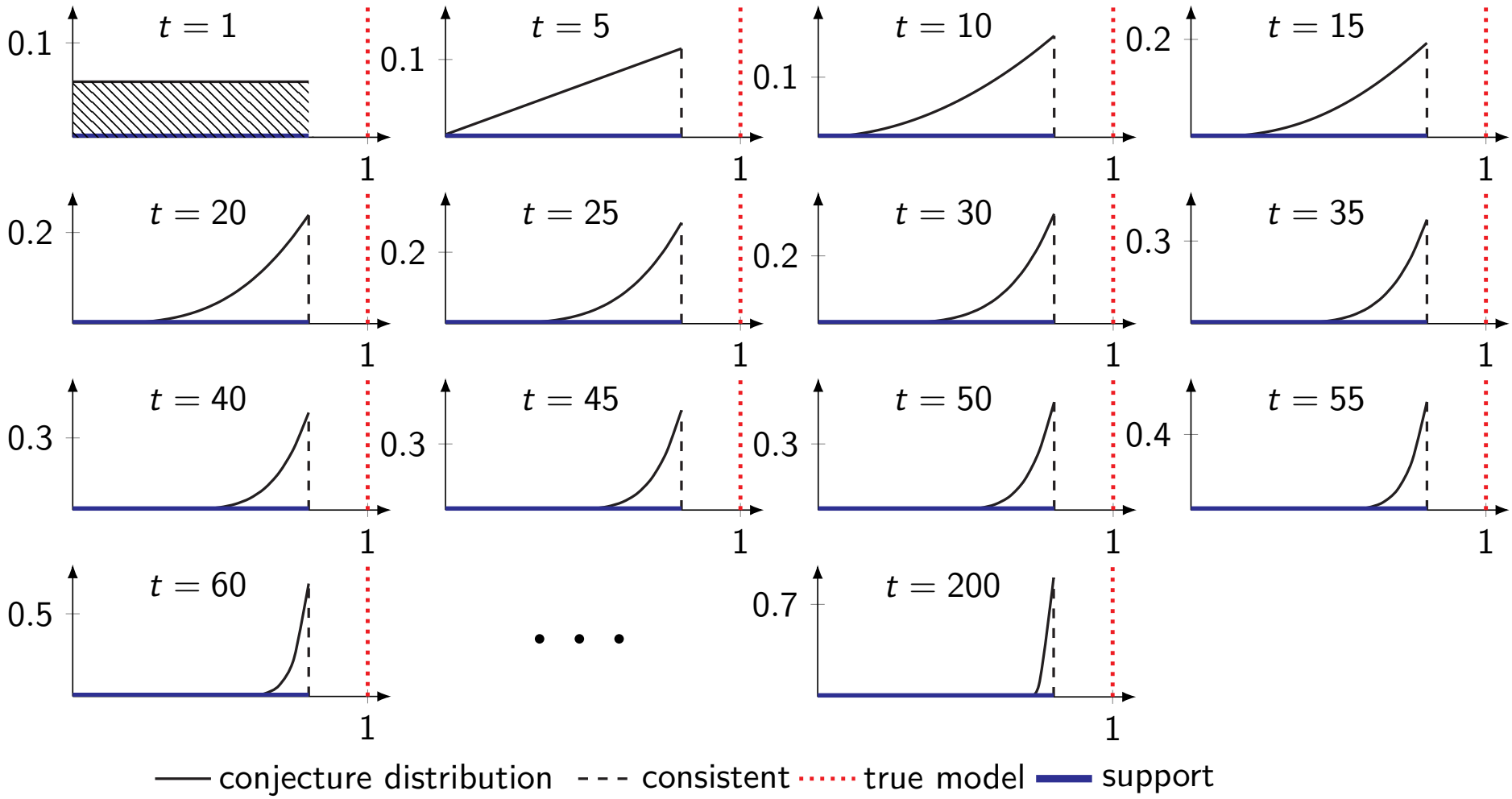
Theorem 1 (Strategy Adaptation)

Under certain technical conditions, the rollout in COL effectively adapts the security strategy to the updated conjecture and improves the upon the base strategy.

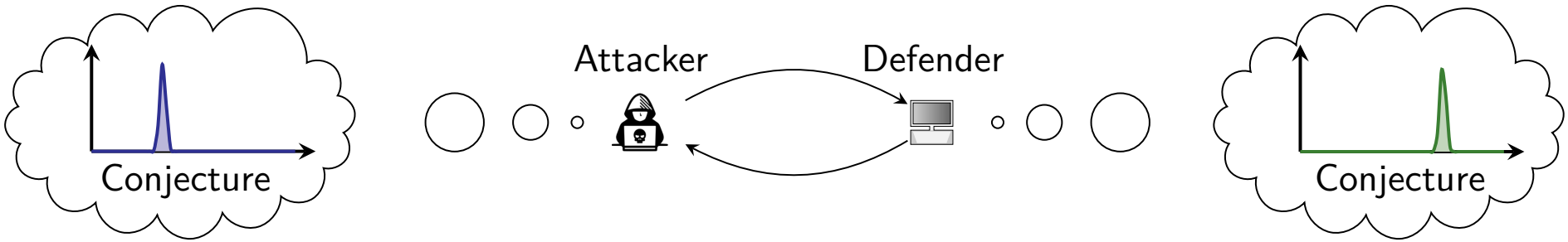
Theorem 2 (Asymptotic Consistency)

COL converges (asymptotically) to conjectures that are consistent with the information feedback from the system.

COL Converges to Consistent Conjectures



The Berk-Nash Equilibrium

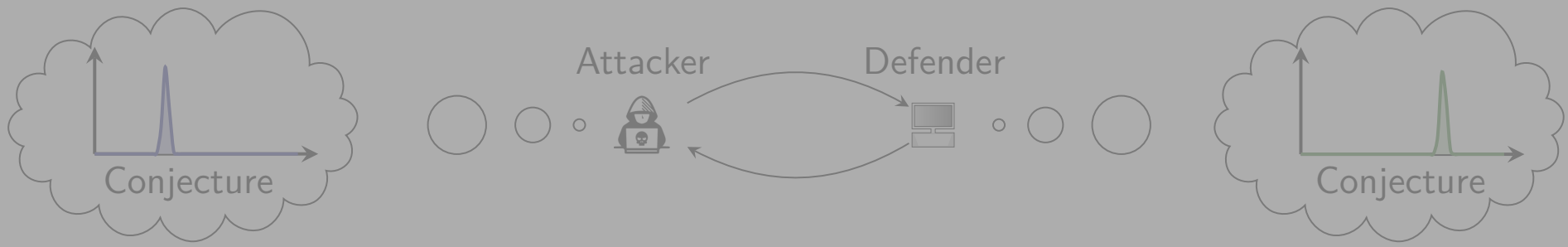


Definition 3 (**Berk-Nash Equilibrium** (Informal))

A strategy profile $\boldsymbol{\pi}$ and an occupancy measure $\nu \in \Delta(\mathcal{B})$ is a Berk-Nash equilibrium iff

1. NASH. π_{k} is a best response against $\pi_{-\mathrm{k}}$.
2. BERK. Each player's conjecture is consistent.
3. STATIONARITY. ν is a limit point of COL.

The Berk-Nash Equilibrium

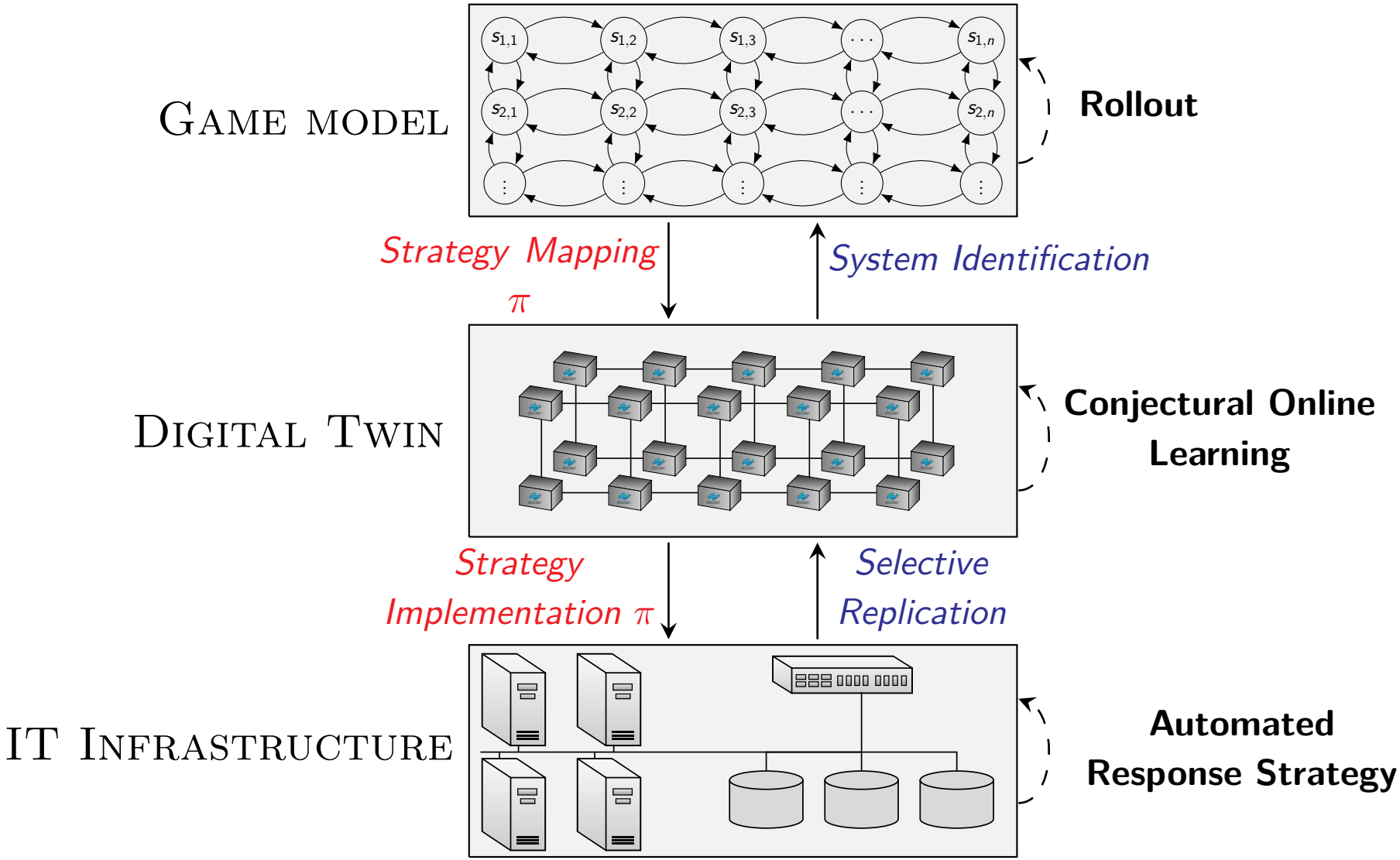


For details, see https://ieeexplore.ieee.org/document/10955193

A strategy profile $\boldsymbol{\pi}$ and an occupancy measure $\nu \in \Delta(\mathcal{B})$ is a Berk-Nash equilibrium iff

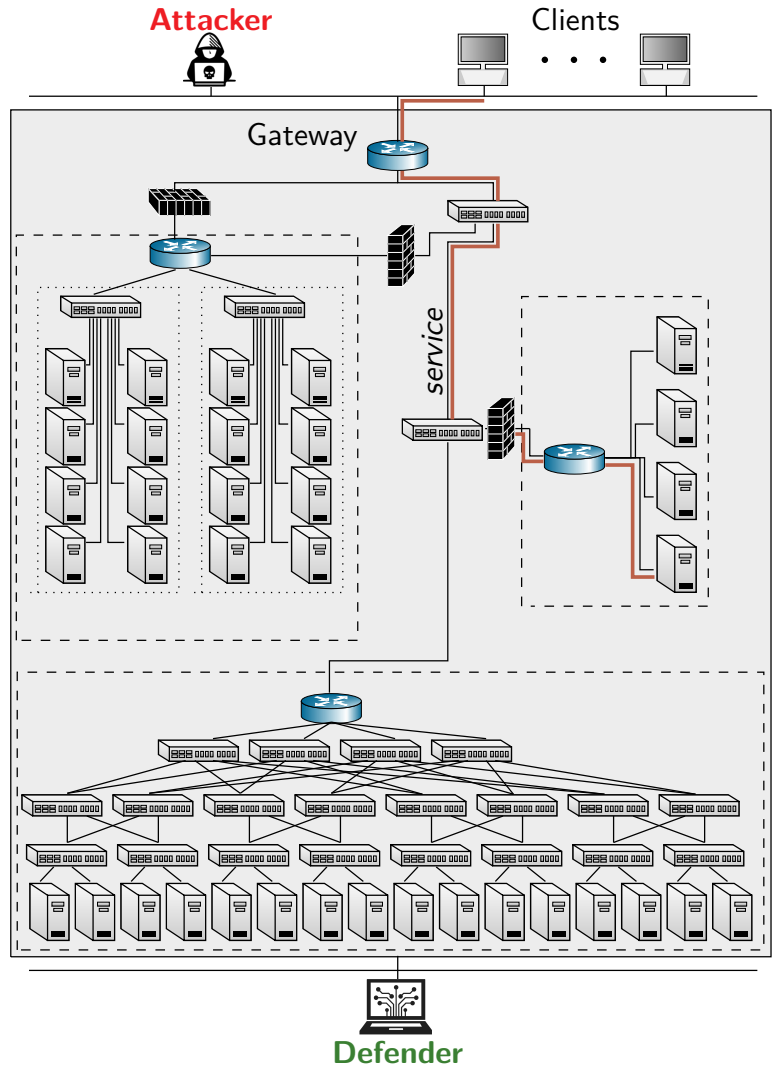
1. NASH. π_{k} is a best response against $\pi_{-\mathrm{k}}$.
2. BERK. Each player's conjecture is consistent.
3. STATIONARITY. ν is a limit point of COL.

Integrating COL in The General Methodology



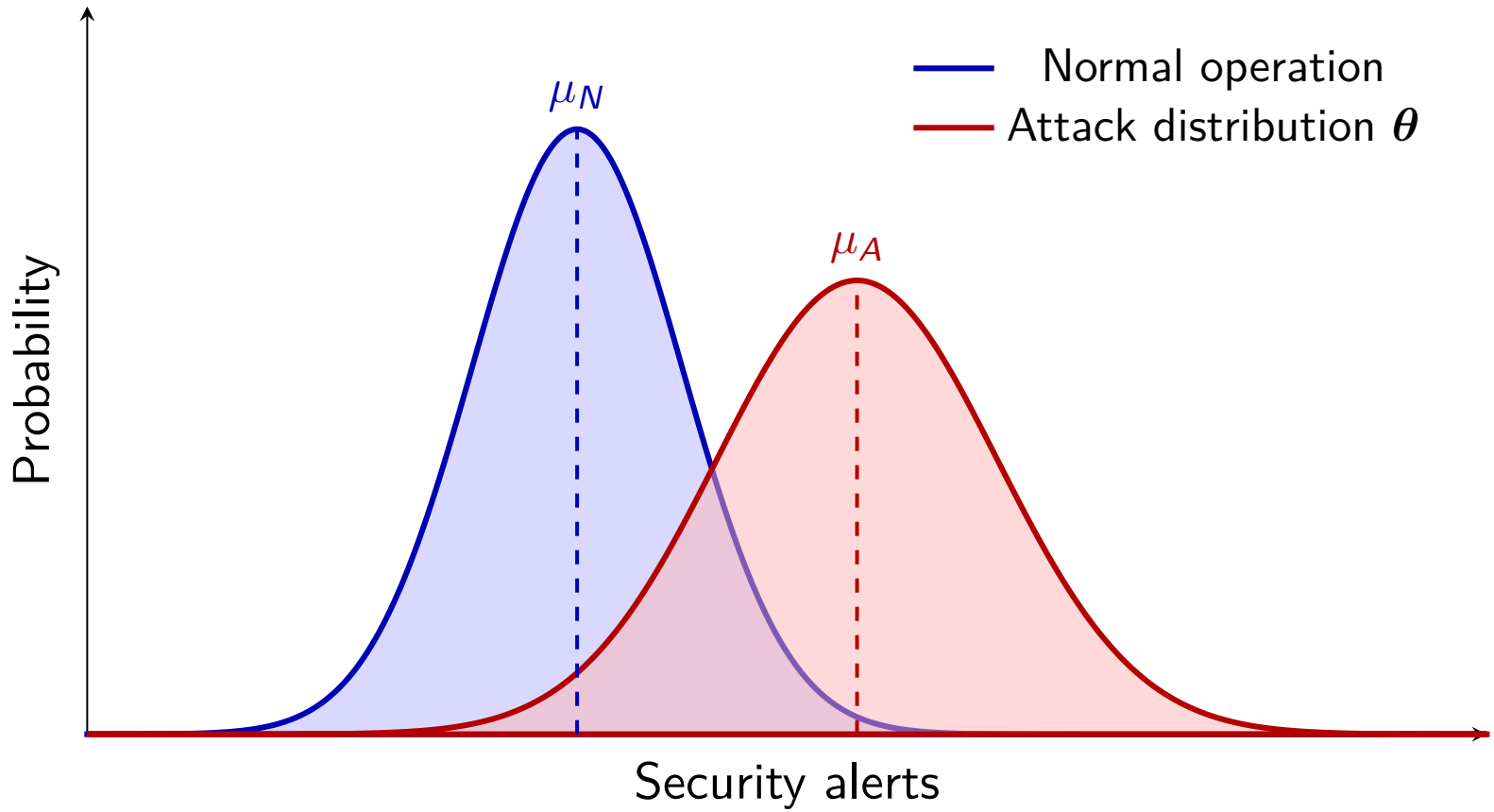
Experimental Evaluation

- We apply our method to **adapt a security response strategy for the IT infrastructure to the right**.
- The defender observes security alerts and can block network flows.



Experimental Evaluation

- The defender knows the normal distribution of alerts.
- The defender has to conjecture the distribution of security alerts during an attack θ.

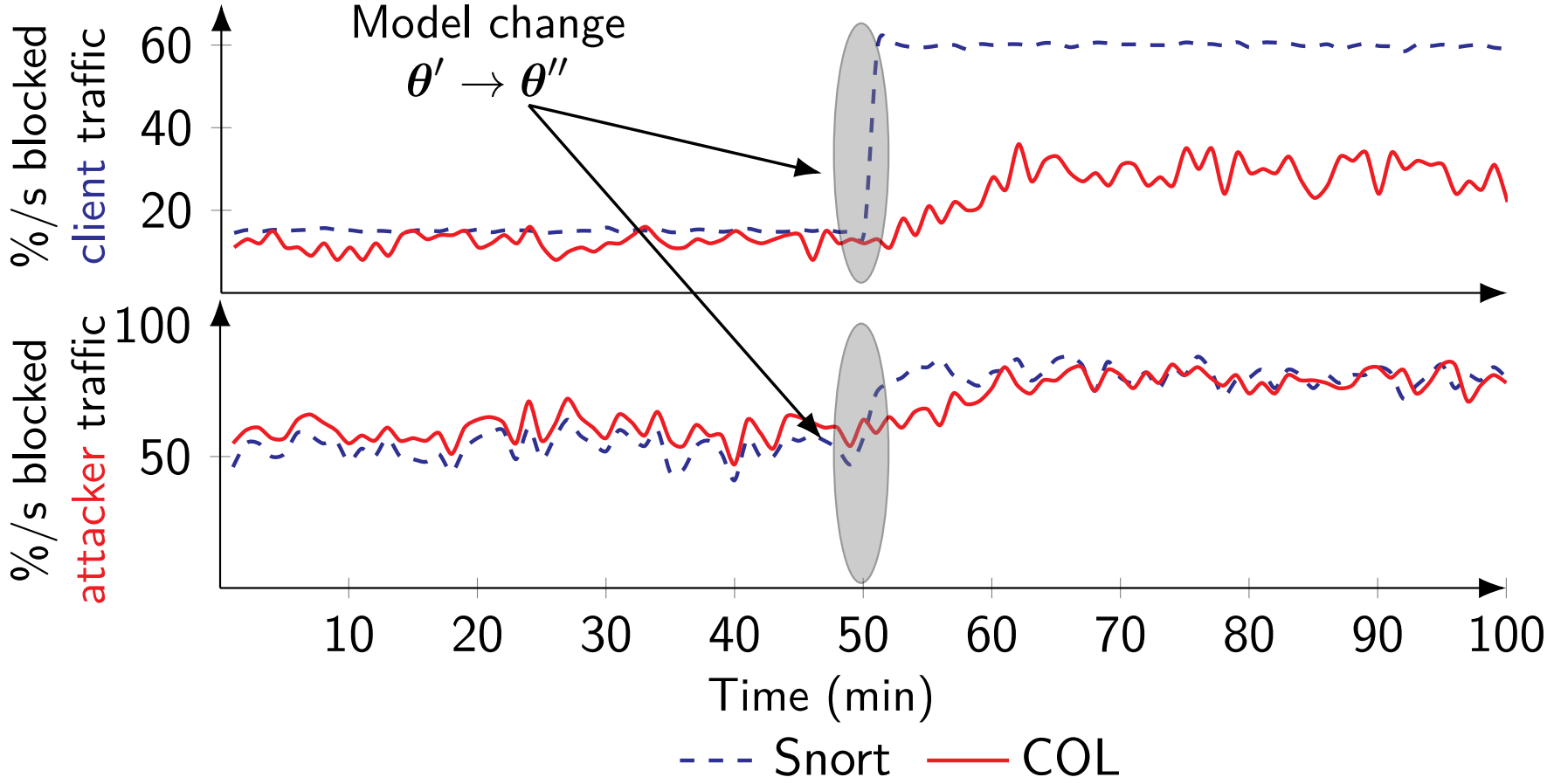


Evaluation Results: Convergence

Method	*Fixed point*	*Time (min)*
COL, $\lvert\Theta_k\rvert = 1$	Berk-Nash equilibrium	6.7 ± 0.7.
COL, $\lvert\Theta_k\rvert = 2$	Berk-Nash equilibrium	12.9 ± 0.9.
COL, $\lvert\Theta_k\rvert = 32$	Berk-Nash equilibrium	17.9 ± 1.0.
COL, $\lvert\Theta_k\rvert = 192$	Berk-Nash equilibrium	29.9 ± 1.2.
COL, $\lvert\Theta_k\rvert = 192$	Berk-Nash equilibrium	194.6 ± 3.7.
Best-response dynamics	ϵ-Nash equilibrium	DNC.
HSVI	ϵ-Nash equilibrium	DNC.
NFSP	ϵ-Nash equilibrium	$\approx$ 919 min.
Fictitious play	ϵ-Nash equilibrium	$\approx$ 4800 min.
PPO	Best response	9.2 ± 0.4 min.

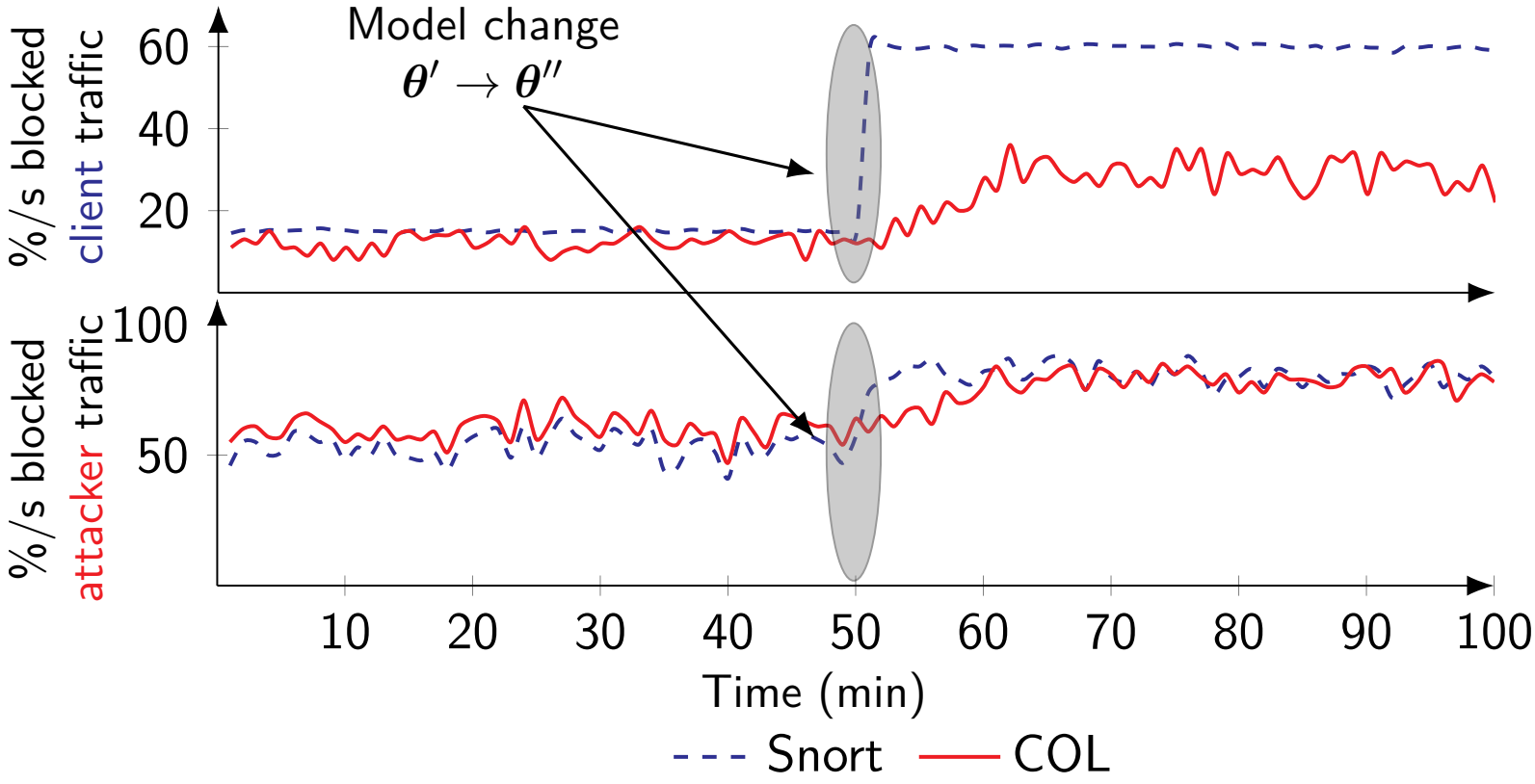
Table 1: Comparison with baseline methods in terms of speed of convergence; DNC is short for "does not converge"; "$\approx$" means that the algorithm nearly converges; numbers indicate the mean and the standard deviation from evaluations with 3 random seeds. $\lvert\Theta_k\rvert$ is the number of conjectures with positive probability initially.

Evaluation Results: Comparison with the Snort IDPS



- θ represents distributions of security alerts.

Evaluation Results: Comparison with the Snort IDPS



Takeaway

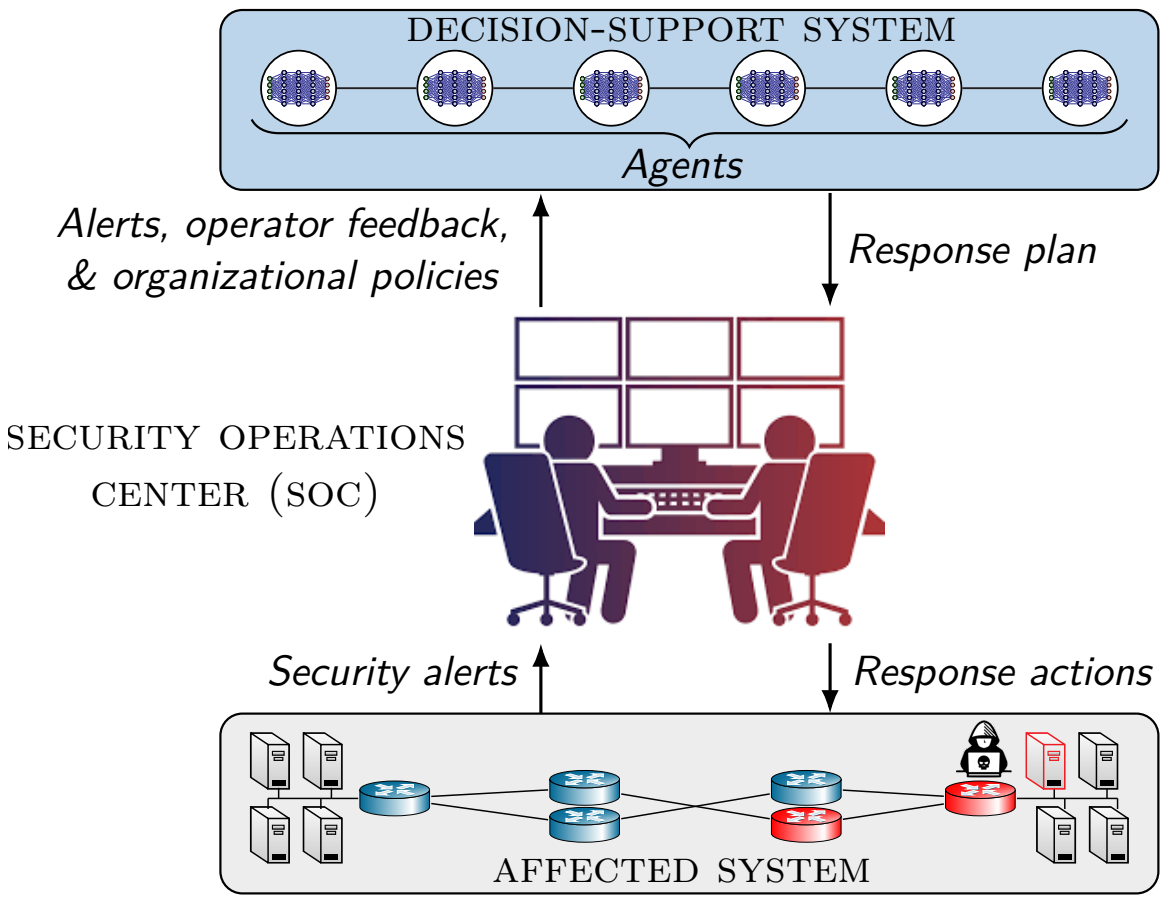
COL relaxes traditional game-theoretic assumptions and provides a practical way to adapt security strategies online.

Outline

- **Autonomous management as a control problem.**
  - Problem formulation.
  - Challenges.
- **Causal online learning of safe regions.**
  - Causal inference.
  - Interventional learning.
- **Game-theoretic learning of security strategies.**
  - Game-theoretic model and equilibrium analysis.
  - Conjectural online learning.
- **Multiagent incident response with LLMs.**
  - System architecture.
  - Theoretical analysis.
- **Conclusion.**

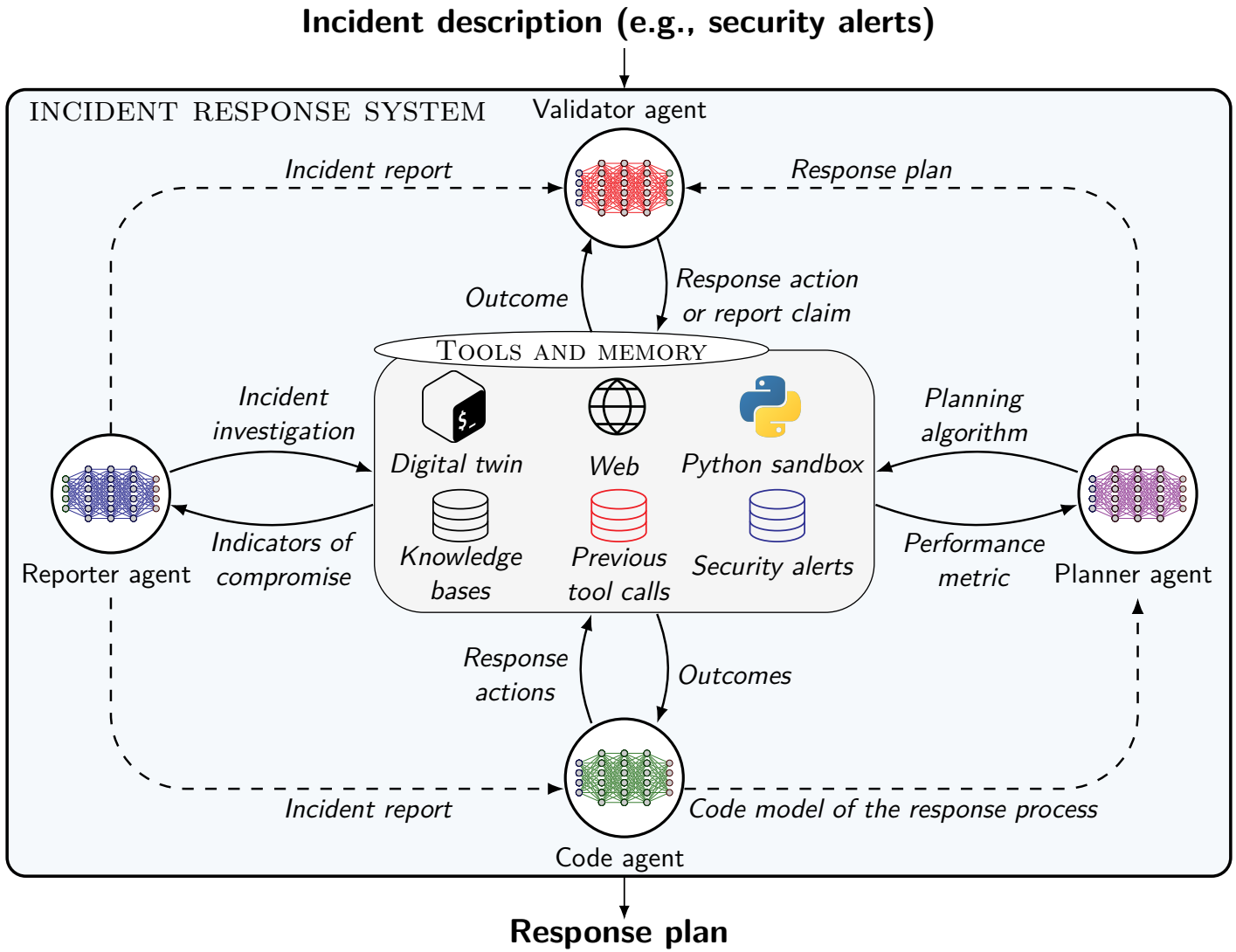
Use Case: Incident Response Planning

- First-line operators have peformed initial triaging and escalated a security incident for further investigation.
- Given the security alerts, the problem is to investigate the incident and generate a response plan.



Multiagent Incident Response Planning with Code Models

- We develop a **multiagent incident response system**.
- LLM-based agents collaborate to investigate an incident.



Design Principles for the Multiagent System

Design Principles for the Multiagent System

Principle 1: Verification

To mitigate the unreliability of LLMs, the system verifies its outputs against a digital twin of the affected system.

Design Principles for the Multiagent System

Principle 1: Verification

To mitigate the unreliability of LLMs, the system verifies its outputs against a digital twin of the affected system.

Principle 2: External planning

To avoid the limitations of LLM-based planning, we use a code model that enables principled planning methods.

Design Principles for the Multiagent System

Principle 1: Verification

To mitigate the unreliability of LLMs, the system verifies its outputs against a digital twin of the affected system.

Principle 2: External planning

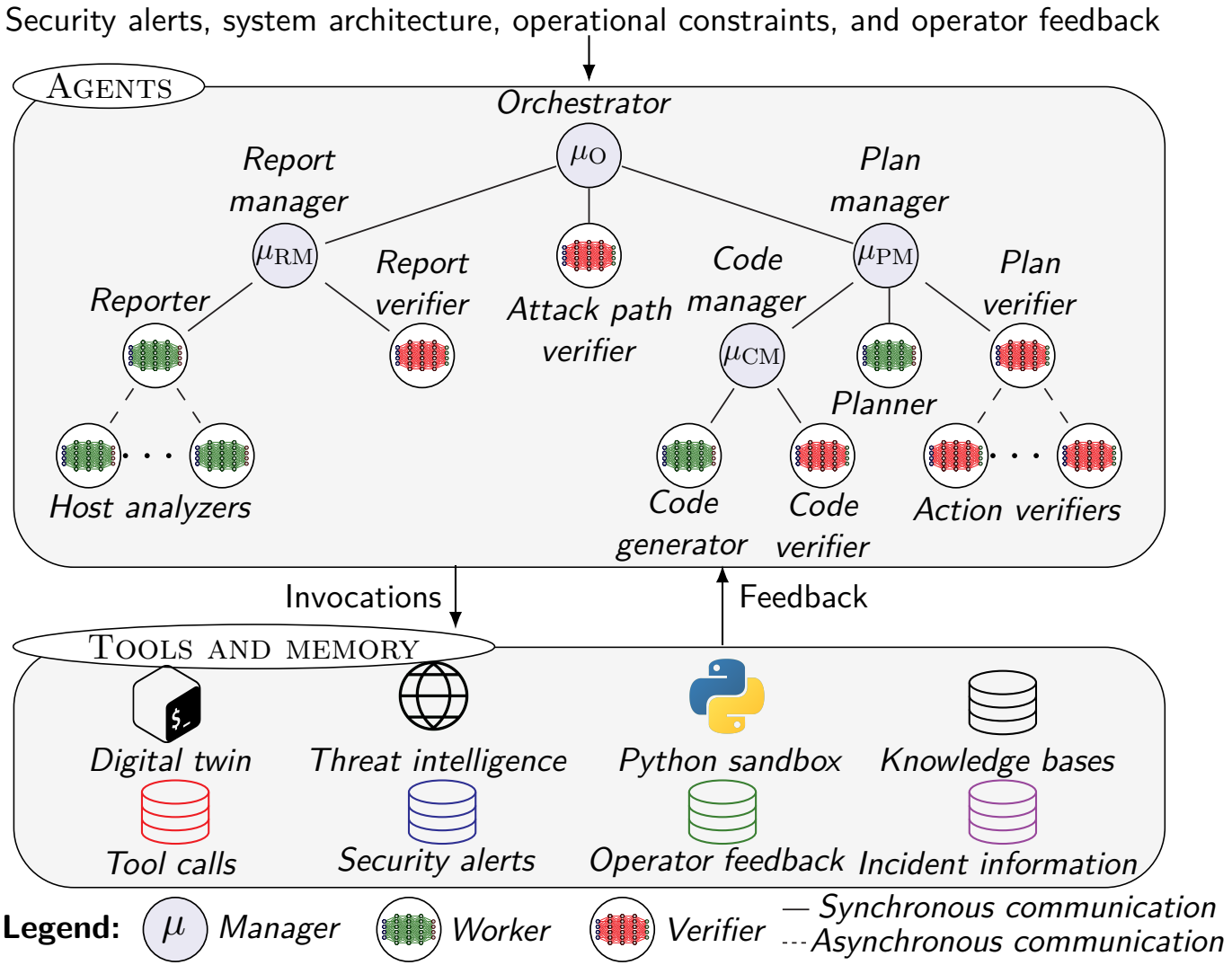
To avoid the limitations of LLM-based planning, we use a code model that enables principled planning methods.

Principle 3: Decomposition into subtasks

To manage the limited context windows of LLMs, we decompose complex tasks into subtasks that are handled by specialized agents in parallel.

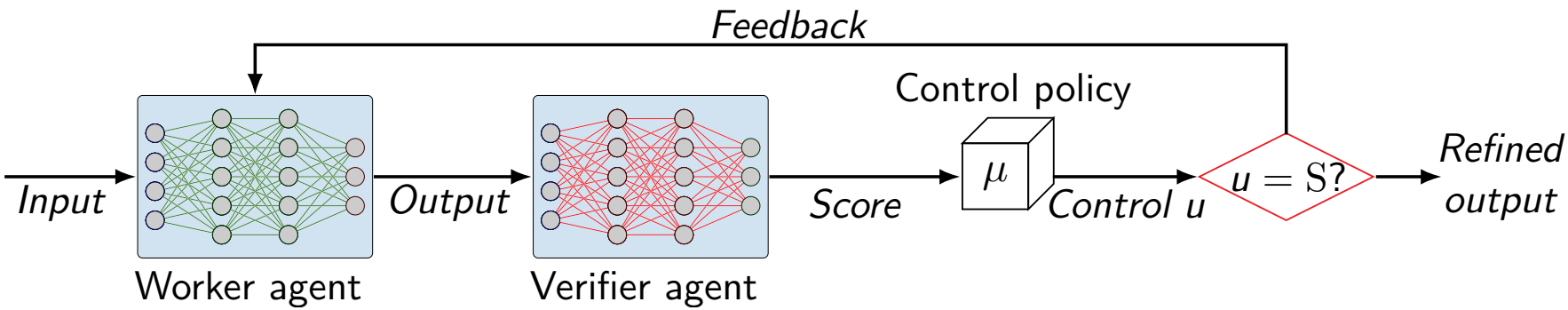
System Architecture

- The system comprises 13 agent-types organized in a tree.
- Agents use LLMs for processing logs, generating outputs, and invoking tools (e.g., executing commands in a digital twin).



Verification Loops

- The multiaget system includes several verification loops where
  - Worker agent generates an output (e.g., a response plan).
  - Verifier agent validates the output (e.g., using the digital twin).
  - Control policy decides the number of verifications.



Theoretical Results (Informal)

Theoretical Results (Informal)

Proposition 2 (Structure of an optimal stopping policy)

Under certain technical conditions, there exists an optimal stopping policy $\mu^\star$ for each verification loop that has the form:

$$\mu^\star(x) = \begin{cases} (\mathrm{S})top & \text{if } x \geq \alpha, \\ (\mathrm{C})ontinue & \text{if } x < \alpha. \end{cases} \tag{3}$$

Theoretical Results (Informal)

Proposition 3 (Structure of an optimal stopping policy)

Under certain technical conditions, there exists an optimal stopping policy $\mu^\star$ for each verification loop that has the form:

$$\mu^\star(x) = \begin{cases} (\mathrm{S})top & \text{if } x \geq \alpha, \\ (\mathrm{C})ontinue & \text{if } x < \alpha. \end{cases} \tag{4}$$

Proposition 4 (Theoretical performance guarantee)

Let $y \in [0, 1]$ be the quality of the response plan and $1 - \delta$ a confidence level. We can tune the stopping threshold α such that

$$P(y \geq \kappa) \geq 1 - \delta, \tag{5}$$

for a desired quality requirement $\kappa \in [0, 1]$.

Theoretical Results (Informal)

Proposition 5 (Structure of an optimal stopping policy)

Under certain technical conditions, there exists an optimal stopping policy $\mu^\star$ for each verification loop that has the form:

$$\mu^\star(x) = \begin{cases} (\mathbb{S})top & \text{if } x \geq \alpha, \\ (\mathbb{C})ontinue & \text{if } x < \alpha \end{cases} \tag{6}$$

A preprint with all details will be available soon.

Proposition 6 (Theoretical performance guarantee)

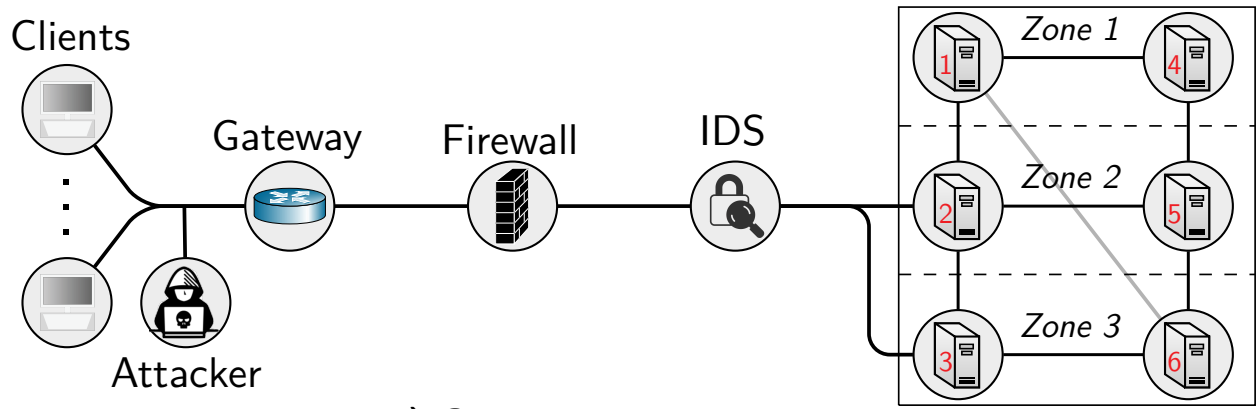
Let $y \in [0, 1]$ be the quality of the response plan and $1 - \delta$ a confidence level. We can tune the stopping threshold α such that

$$P(y \geq \kappa) \geq 1 - \delta, \tag{7}$$

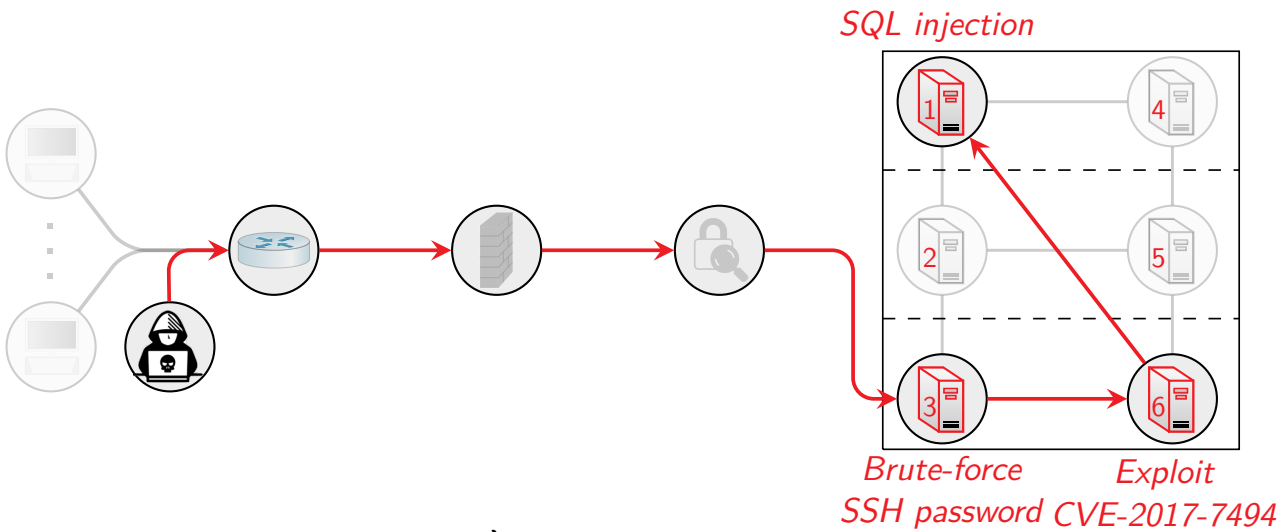
for a desired quality requirement $\kappa \in [0, 1]$.

Example Incident for Evaluation

- **Multi-stage attack** on a cloud infrastructure.



a) System architecture



b) Attack path

Security Alerts

- The system is only given the following about the incident:

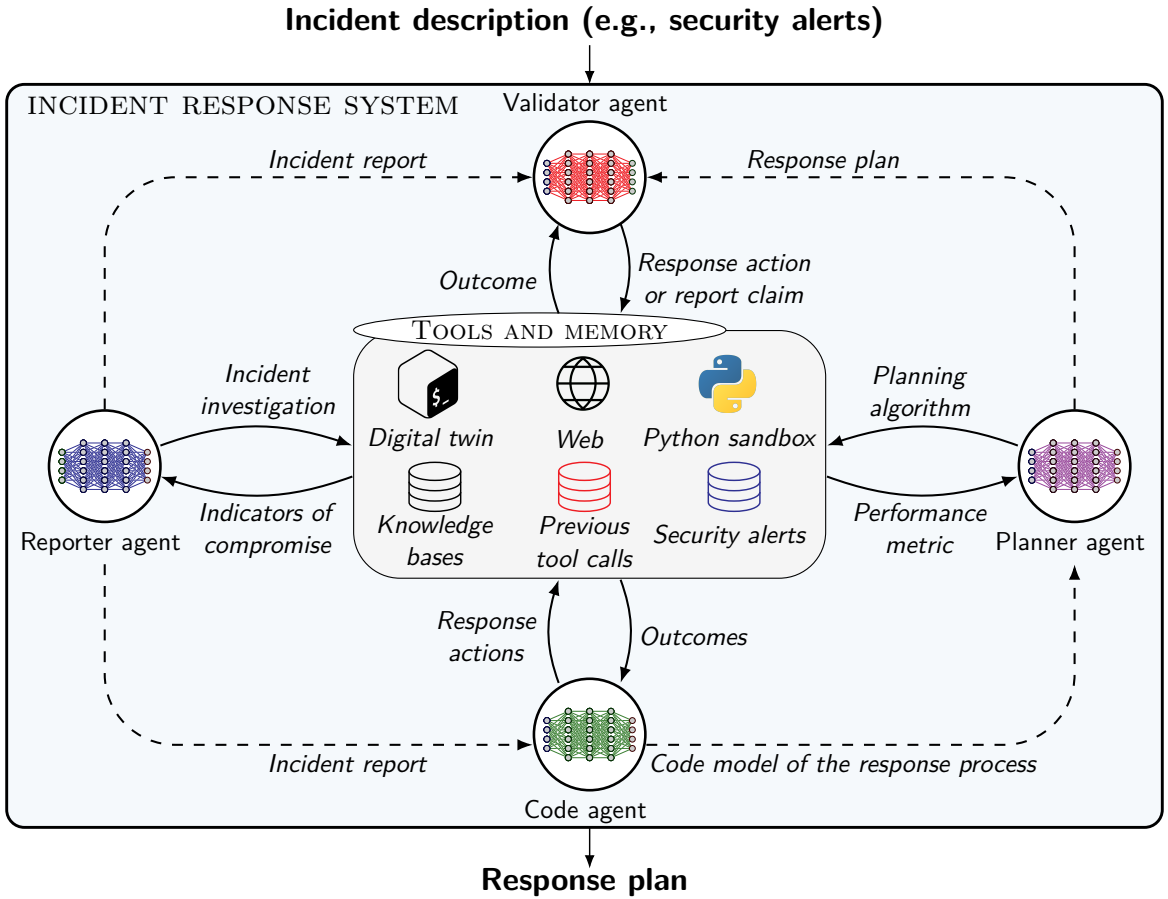
Security alerts

```
02/06-10:15:22.341201 [**] [1:2006546:3] ET SCAN SSH Brute Force Login Attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
{TCP} 192.168.1.50:44321 -> 10.0.3.3:22

02/06-10:42:15.889102 [**] [1:2014473:3] ET EXPLOIT Possible SQL Injection Attempt
 (UNION SELECT) [**] [Classification: Attempted Administrator Privilege Gain]
[Priority: 1] {TCP} 10.0.3.3:55210 -> 10.0.2.1:80
```

Instiation of the Multiagent System

- We instantiate each agent with Gemini 3.1 Pro as the LLM.
- The **agents have access to following tools**:
  - The digital twin, a Python sandbox, the NIST vulnerability database, the MITRE ATT&CK taxonomy, the VirusTotal security API, the AbuseIPDB API, the OTX threat intelligence API, and the Tavily search API.

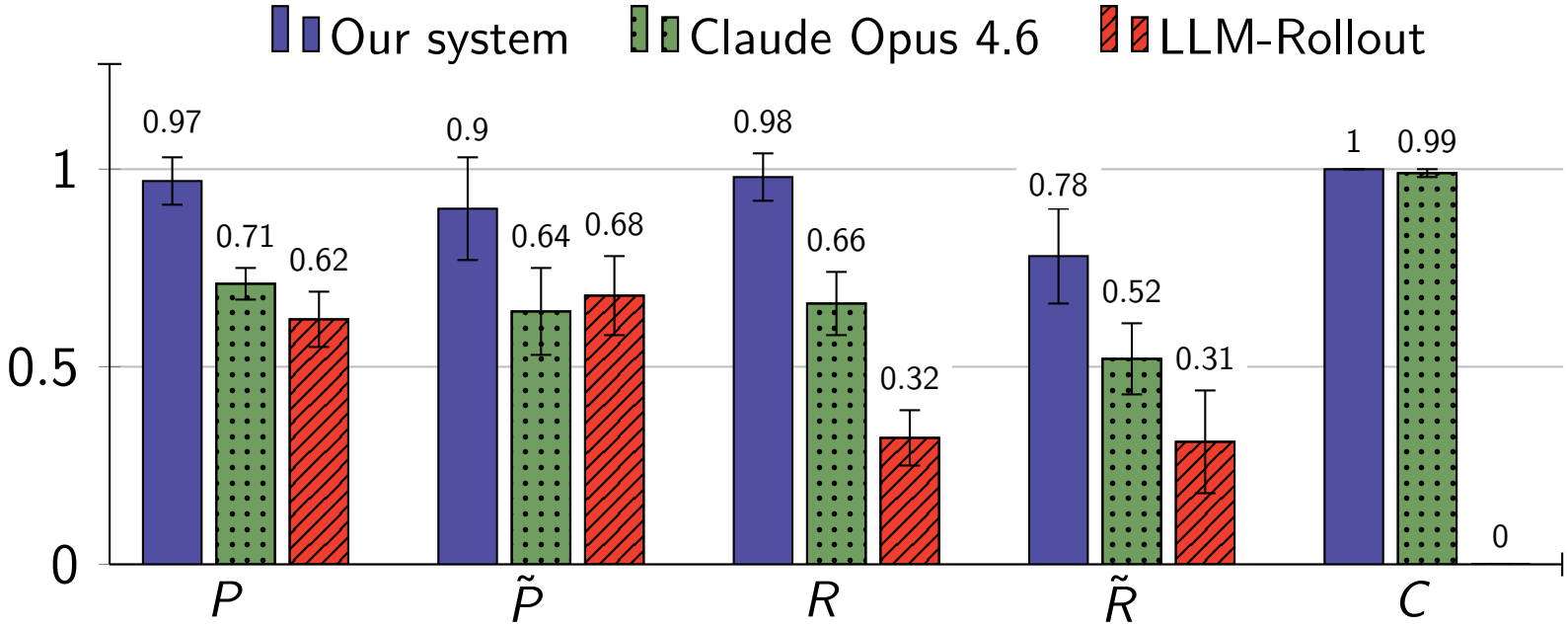


Evaluation Metrics

Metric	*Description*
Report precision P	Fraction of correct incident report claims.
Plan precision $\tilde{P}$	Fraction of correct response actions.
Report recall R	Fraction of incident details in the incident report.
Plan recall $\tilde{R}$	Fraction of response stages completed.
Constraint satisfaction C	Fraction of satisfied constraints after the response.
Token cost T	Total tokens processed.
Planning time $\tilde{T}$	Total processing time.

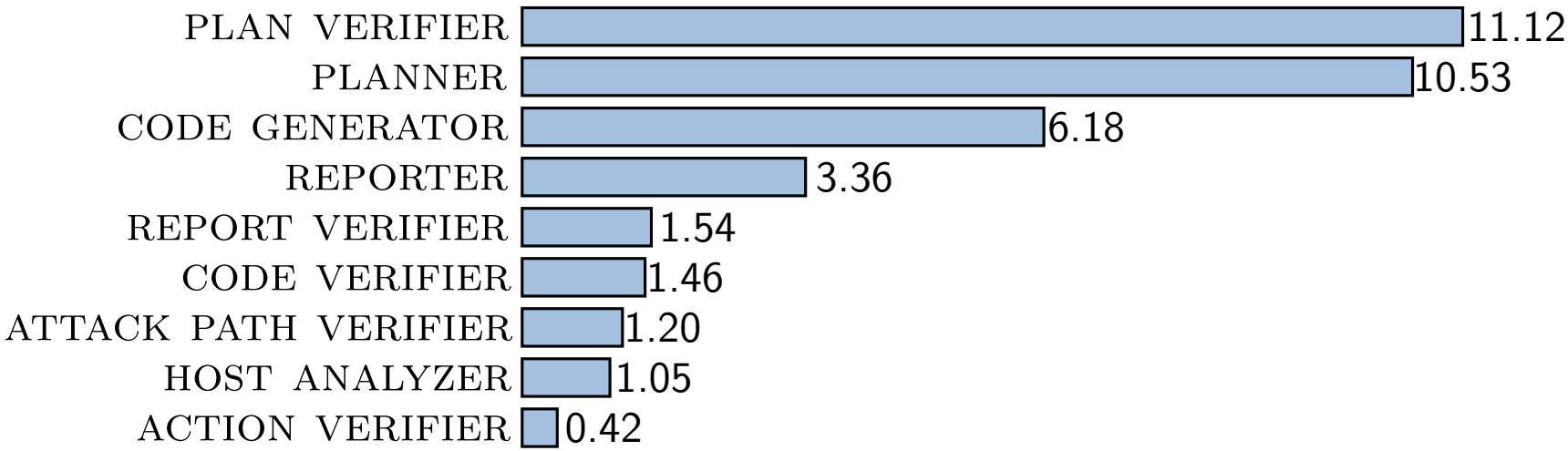
Table 2: Metrics to evaluate the performance of the system.

Evaluation Results: Comparison with Baselines



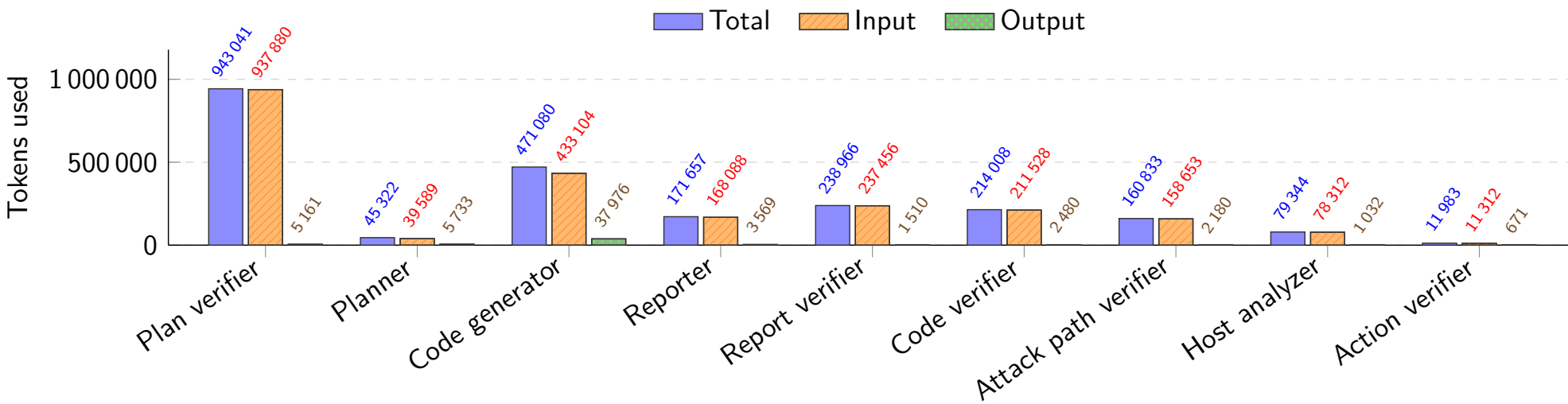
Performance of the multiagent system when applied to the example incident. Bar groups relate to performance metrics (↑ better). Values are averaged across five executions; error bars indicate standard deviations.

Evaluation Results: Agent Execution Times



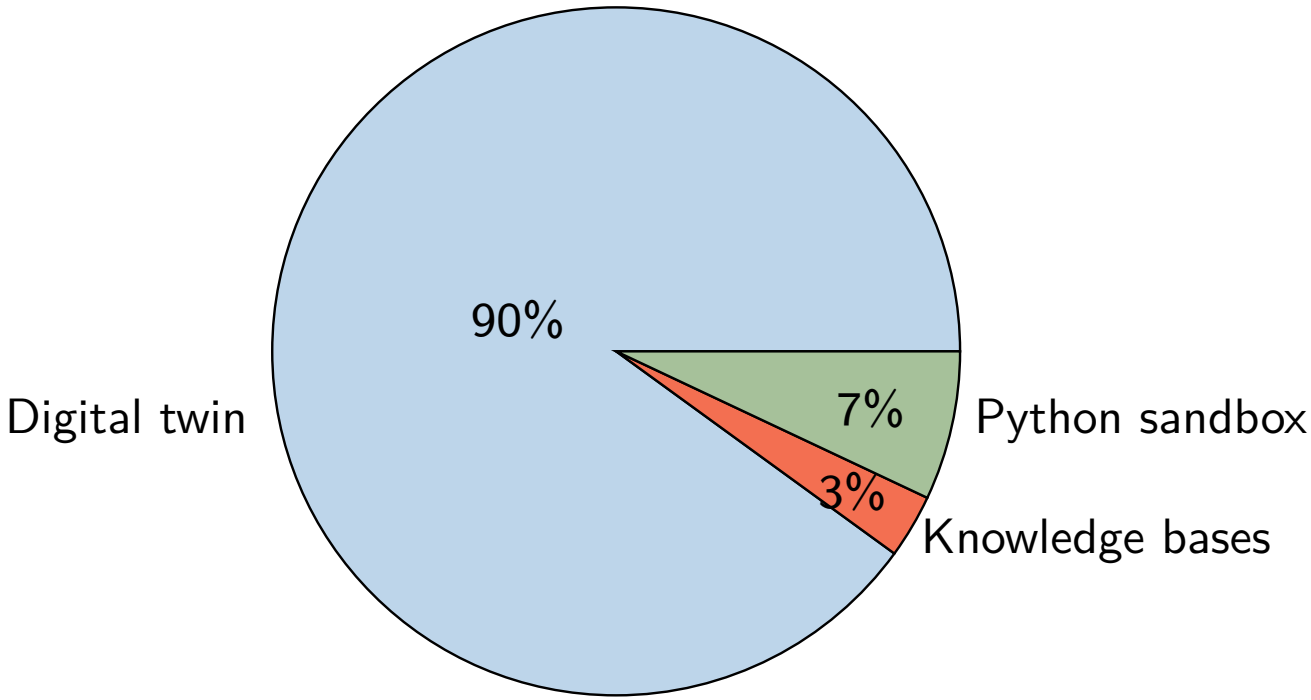
Agent execution times (min) when applied to the example incident.

Evaluation Results: Token Usage



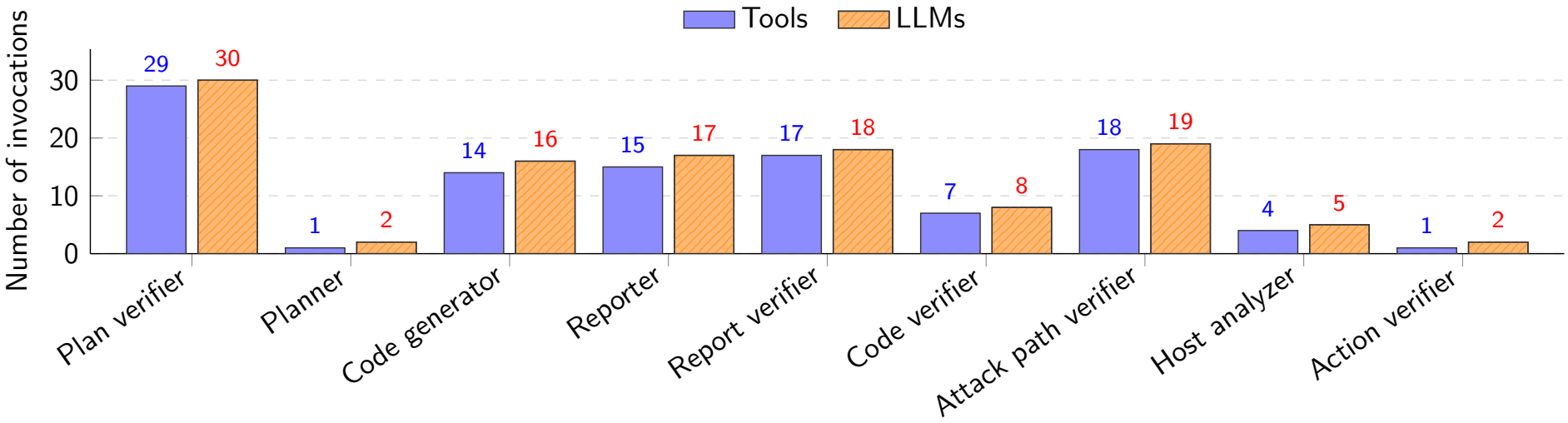
Number of tokens used per agent. Bars indicate average values across 5 executions of the multiagent system when applied to the example incident.

Evaluation Results: Tool Usage (1/2)



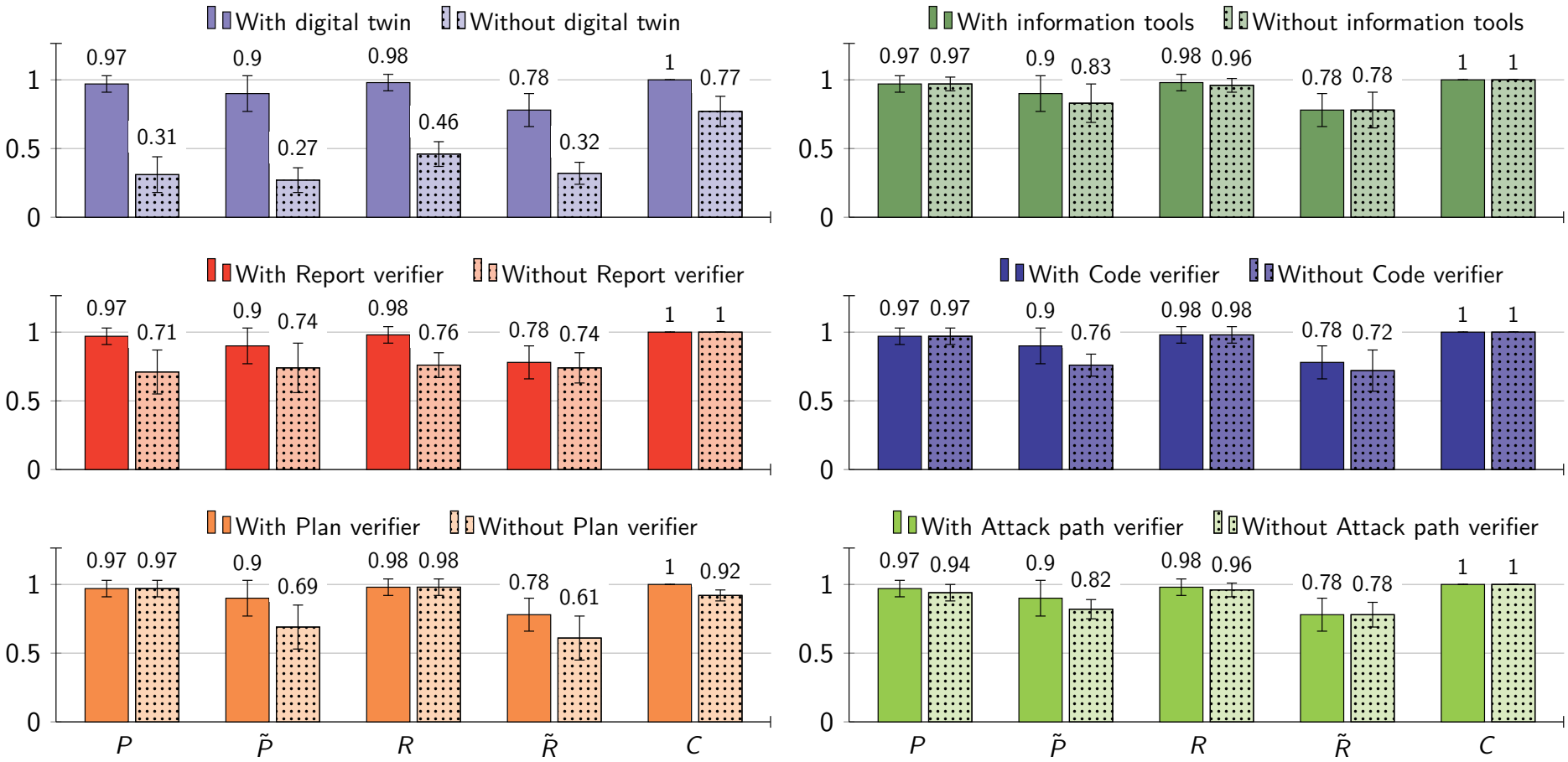
Fraction of tool calls of the multiagent system when applied to the example incident. The fractions represent tool calls that relate to the digital twin, external knowledge bases, and the Python sandbox.

Evaluation Results: Tool Usage (2/2)



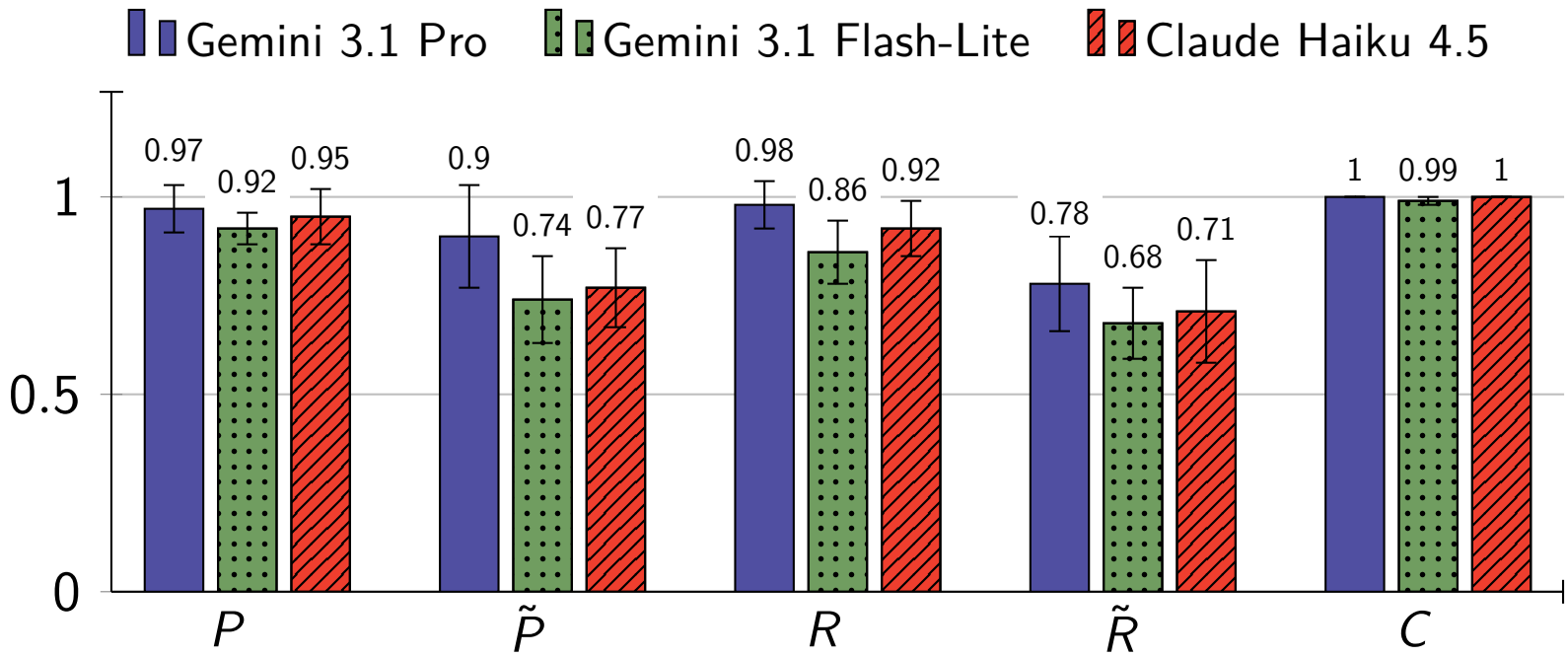
Number of invocations of tools and calls to LLMs per agent when applying the multiagent system to the example incident.

Evaluation Results: Ablation Study (1/2)



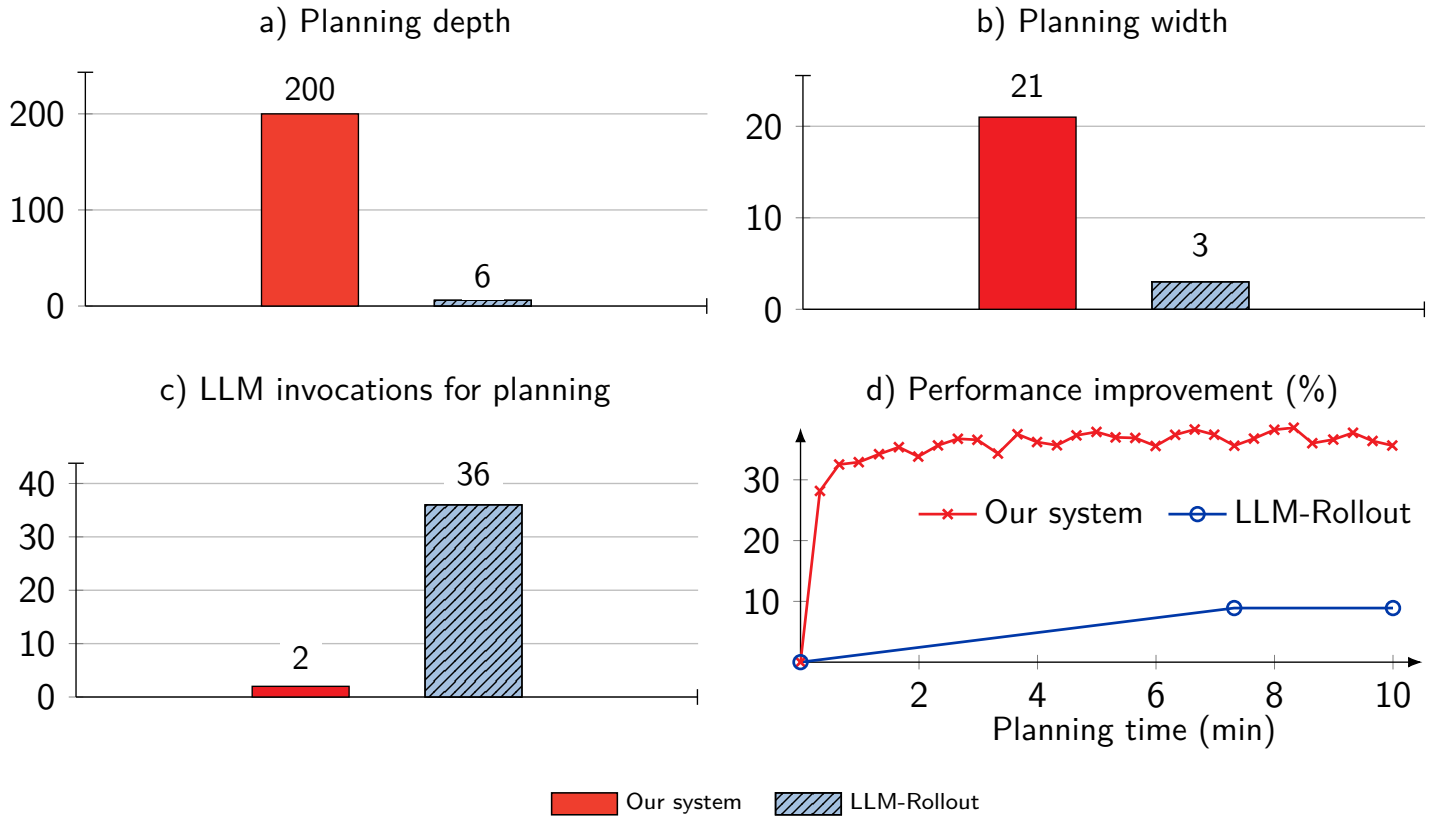
Performance of the multiagent system with and without different components when applied to the example incident. Bar groups relate to performance metrics (↑ better). Values are averaged across five executions; error bars indicate standard deviations.

Evaluation Results: Ablation Study (2/2)



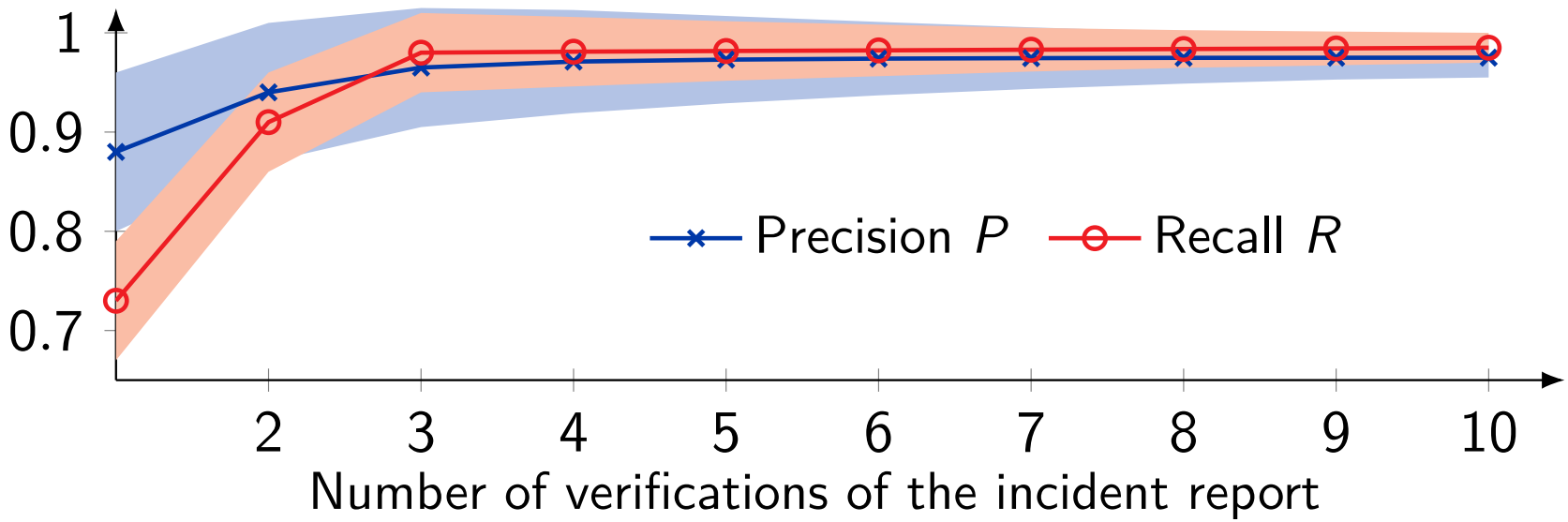
Performance of the multiagent system when instantiated with different LLMs and applied to the example incident. Bar groups relate to peformance metrics (↑ better). Values are averaged across five executions; error bars indicate standard deviations.

Evaluation Results: Planning Performance



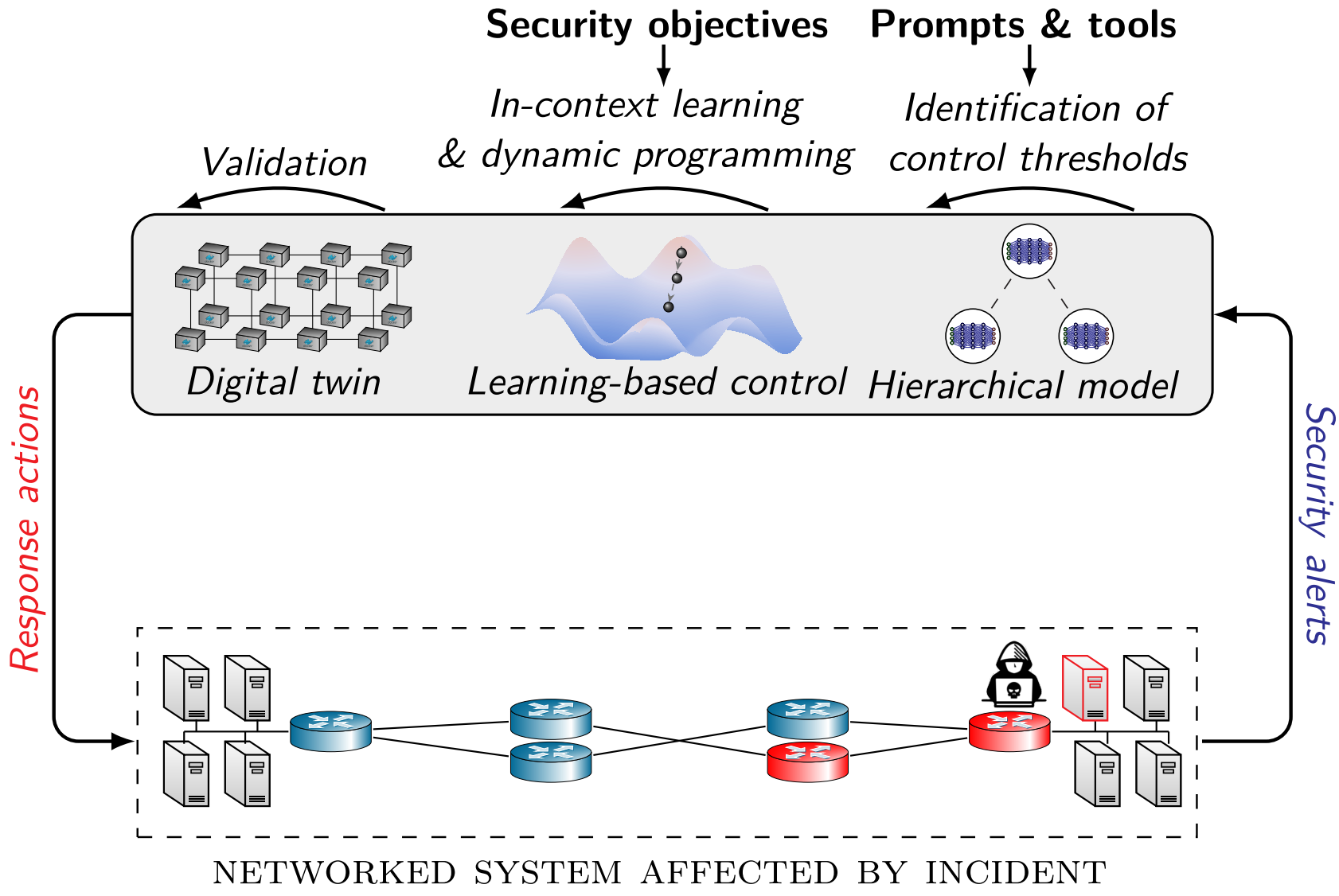
Comparison between our system and LLM-Rollout in terms of planning depth (lookahead horizon of the planning), planning width (number of response actions considered at each planning step), number of LLM invocations, and performance when applied to the example incident.

Evaluation Results: Iterative Refinement



Evolution of the precision and recall of the incident report in function of the number of verifications. Curves indicate the mean values across five executions; shaded regions indicate standard deviations.

The Multiagent System in the General Methodology



Outline

- **Autonomous management as a control problem.**
  - Problem formulation.
  - Challenges.
- **Causal online learning of safe regions.**
  - Causal inference.
  - Interventional learning.
- **Game-theoretic learning of security strategies.**
  - Game-theoretic model and equilibrium analysis.
  - Conjectural online learning.
- **Multiagent incident response with LLMs.**
  - System architecture.
  - Theoretical analysis.
- **Conclusion.**

Conclusion

- Networked systems grow increasingly complex and dynamic.
  - Require autonomous management and control.

- I advocate for a **learning-based control methodology**.
  - Identification of a system model.
  - Control optimization through learning-based methods.
  - Validation on a digital twin.