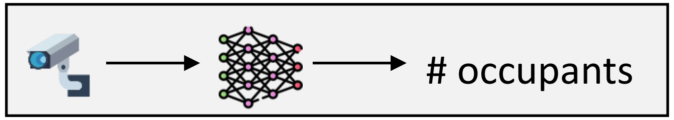
CRAFTER: Causality-based Self-adaptation for Autonomous IoT Systems

Houssam Hajj Hassan[1, 2], Ajay Kattepur[3], Denis Conan[1], Georgios Bouloukakis[1, 4]

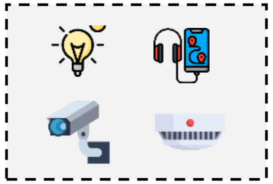
1 Télécom SudParis, Institut Polytechnique de Paris, France
2 Orange Innovation, France
3 Ericsson AI Research, India
4 Department of Electrical and Computer Engineering, University of Patras, Greece

Dynamic Smart Spaces

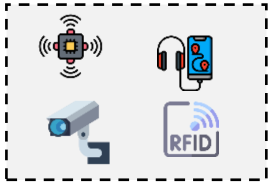
IoT system components



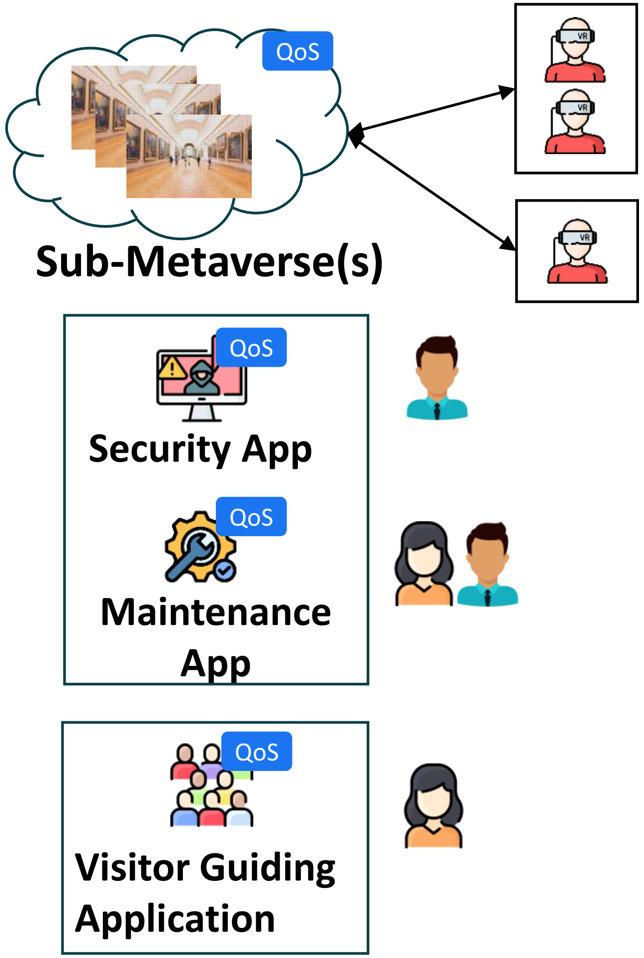
Occupant detection



Apollo Gallery

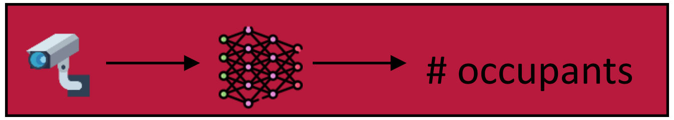


Greek Antiquities

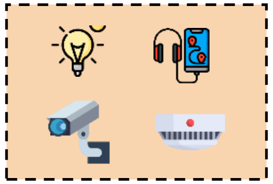


Dynamic Smart Spaces

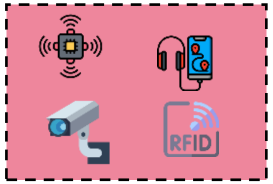
Dynamic workload



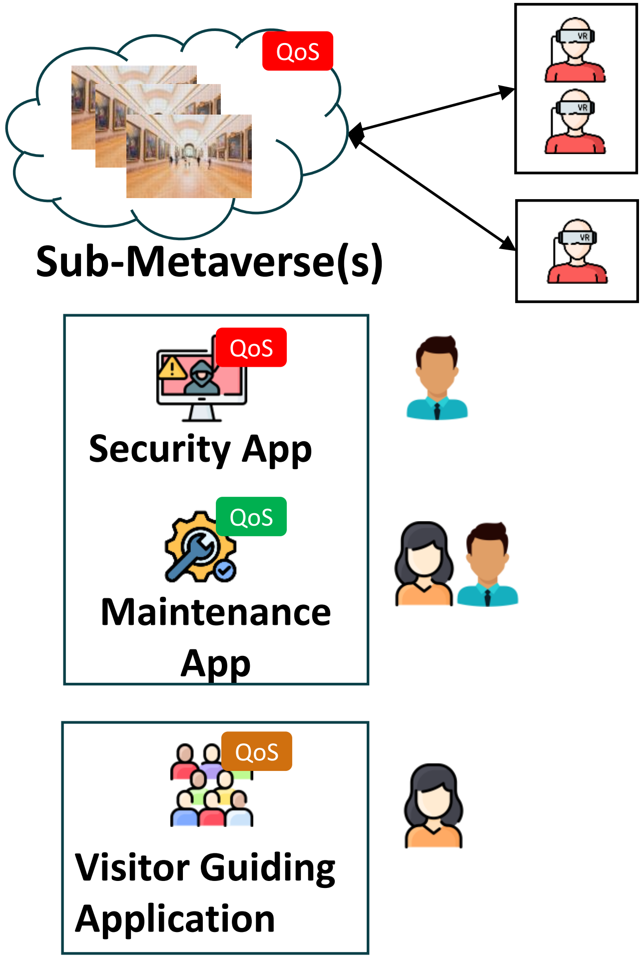
Occupant detection



Apollo Gallery

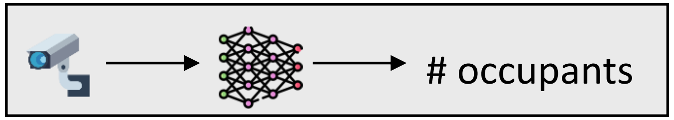


Greek Antiquities

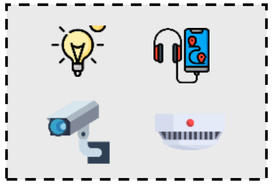


Dynamic Smart Spaces

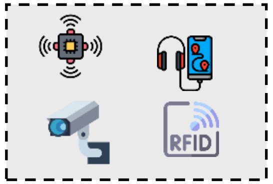
Changing QoS requirements



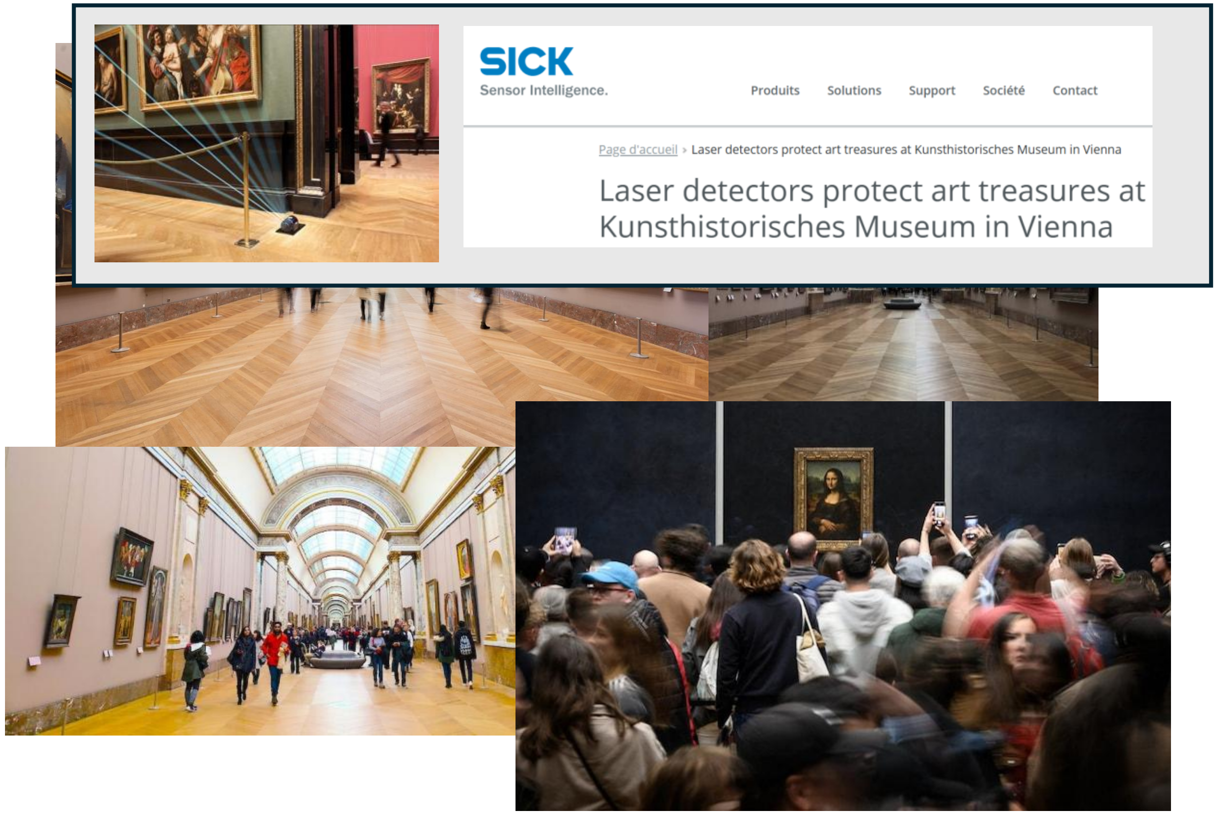
Occupant detection

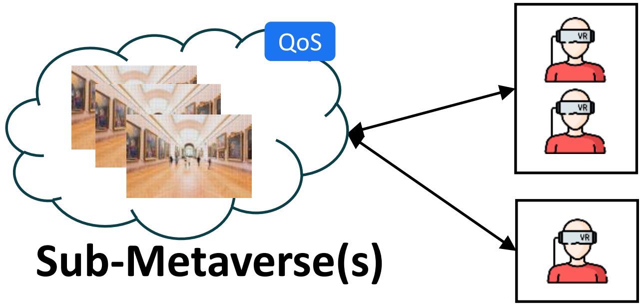


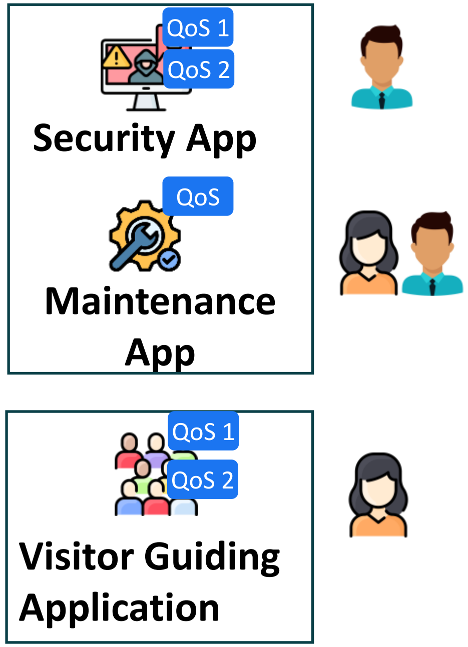
Apollo Gallery



Greek Antiquities







State-of-the-Art Approaches
Self-adaptive IoT

Application-layer adaptation: Dynamically adjusts sensor deployment and data source assignment based on QoS requirements and opportunistic sensing [1, 2]; AI planning further enables adaptive data flow reconfiguration in response to real-time application changes [3, 4].
Goal-driven architectures: Address sudden device unavailability and topology changes [5]; ML-enhanced variants extend this to user activity recognition for smarter deployment decisions [6].
Middleware-level adaptation: Manages soft resource allocation for critical microservices [7] and supports configuration adaptation across the cloud-edge continuum in mobile settings [8].
RL-based methods: Dominant paradigm for runtime adaptation; covers traffic forecasting [9], load balancing at the edge [10], resource monitoring with publish/subscribe middleware [11], and safety-aware adaptation for robotic agents [12].
Meta-RL & online learning: Meta-reinforcement learning enables fast policy transfer across environment dynamics [13]; online learning approaches push toward antifragility [14].
Model Predictive Control: Proactive adaptation for non-linear cyber-physical systems using AMPC [15], though primarily targeting autonomous mobile robots.

[1] L. Delouee et al. DEBS 2023.
[2] E. Eryilmaz et al. ACM IoT 2019.
[3] H. Hajj Hassan et al. SEAMS 2023.
[4] H. Hajj Hassan et al. ICSA-4 2024.
[5] F. Alkhabbas et al. ICSA 2020.
[6] F. Alkhabbas et al. ACM IoT 2020.
[7] L. Rosa et al. ACM Middleware 2023.
[8] M. Mendula et al. PerCom 2024.
[9] H. Muccini et al. SmartComp 2020.
[10] Q. Liu et al. IEEE TPDS. 2022,
[11] P. Venkateswaran et al. PMC 2021.
[12] Q. Zhang et al. ACSOS-C 2024,
[13] M. Zhang et al. ACSOS 2021.
[14] V. Scotti et al. ACSOS 2025.
[15] F. Edrisi et al. SEAMS 2025,

State-of-the-Art Approaches

Self-adaptive IoT

Application-layer adaptation: Dynamically adjusts sensor deployment and data source assignment based on QoS requirements and opportunistic sensing [1, 2]; AI planning further enables adaptive data flow reconfiguration in response to real-time application changes [3, 4].

Goal-driven a … vailability … variants
extend this to … loyment
Middleware- … llocation … s
configuration … um in m
RL-based me … adaptati … ncing at
the edge [10] … ibe midd … for robotic
agents [12].

How to know ***when*** and ***how*** to take adaptation decisions?

How to automate the process of designing self-adaptive applications?

Meta-RL & online learning: Meta-reinforcement learning enables fast policy transfer across environment dynamics [13]; online learning approaches push toward antifragility [14].

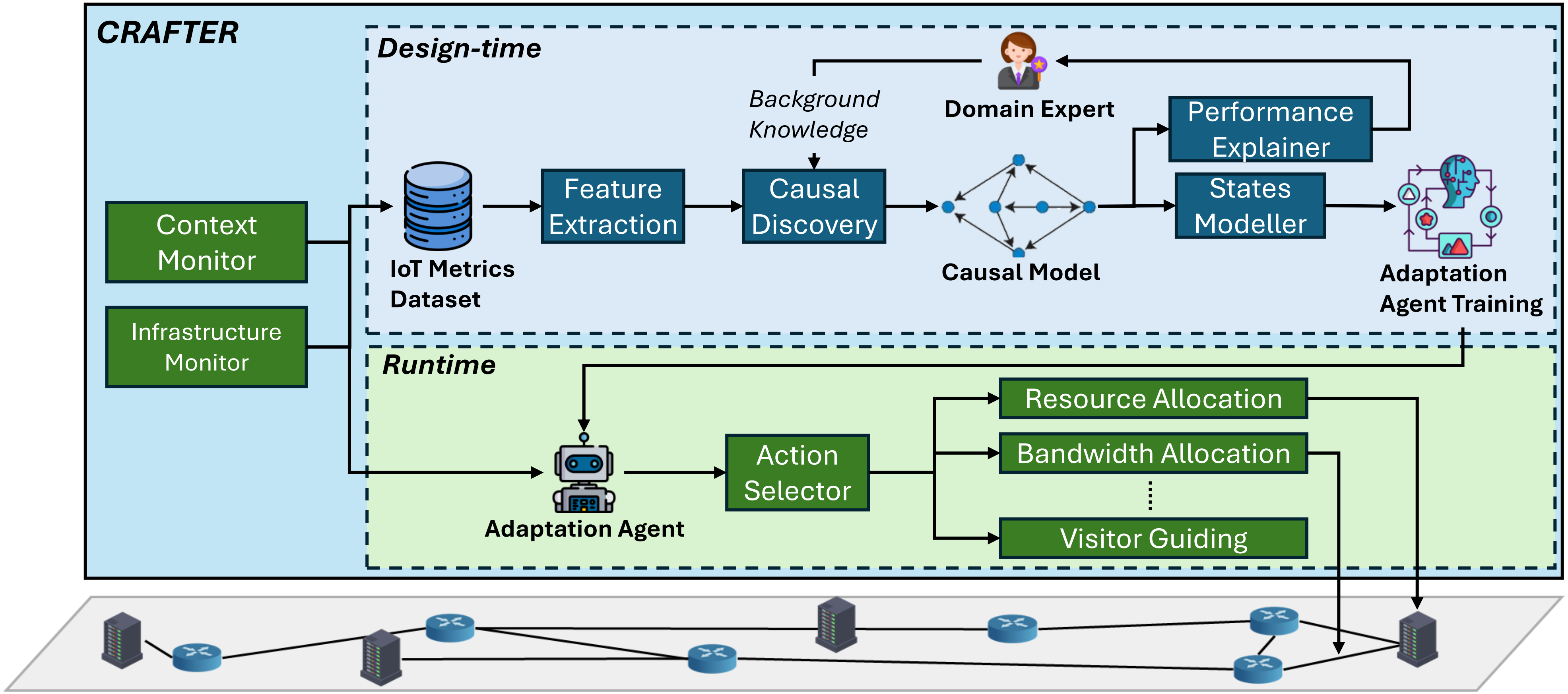
Model Predictive Control — Proactive adaptation for non-linear cyber-physical systems using AMPC [15], though primarily targeting autonomous mobile robots.

[1] L. Delouee et al. DEBS 2023.
[2] E. Eryilmaz et al. ACM IoT 2019.
[3] H. Hajj Hassan et al. SEAMS 2023.
[4] H. Hajj Hassan et al. ICSA-4 2024.
[5] F. Alkhabbas et al. ICSA 2020.
[6] F. Alkhabbas et al. ACM IoT 2020.
[7] L. Rosa et al. ACM Middleware 2023.
[8] M. Mendula et al. PerCom 2024.
[9] H. Muccini et al. SmartComp 2020.
[10] Q. Liu et al. IEEE TPDS. 2022,
[11] P. Venkateswaran et al. PMC 2021.
[12] Q. Zhang et al. ACSOS-C 2024,
[13] M. Zhang et al. ACSOS 2021.
[14] V. Scotti et al. ACSOS 2025.
[15] F. Edrisi et al. SEAMS 2025,

CRAFTER: Causality-based Self-adaptation

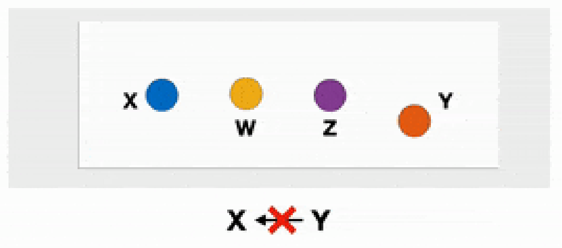
High-level overview

- A semi-automated approach for designing self-adaptive IoT systems across heterogeneous environments.
- An automated Causal RL pipeline to take information adaptation decisions in dynamic environments



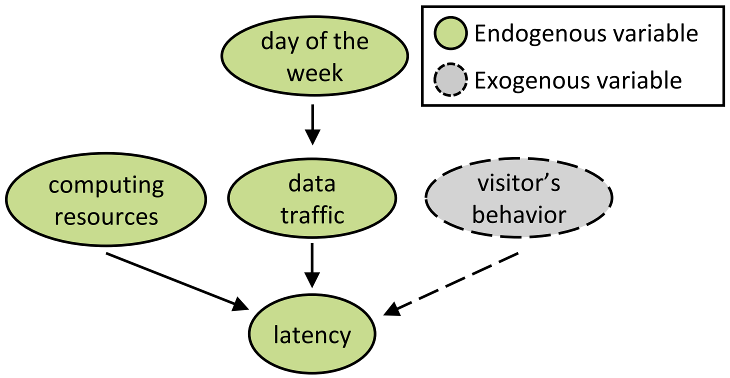
Background on Causality[1] (1)

X causes change in Y



Y does not cause change in X

A causal graph *G* is a DAG that depicts the cause-effect relationships amongst variables in a given system.



data traffic = f1(day of the week)
latency = f_2(computing resources, data traffic, visitor's behavior)

Structural Causal Model (SCM)

An SCM $\mathcal{M}$ is defined by a 4-tuple <$\boldsymbol{V}, \boldsymbol{U}, \mathcal{F}, P(\boldsymbol{U})$>:

- $\boldsymbol{V}$ is a set of endogenous/observable variables
- $\boldsymbol{U}$ is a set of exogeneous/unobservable variables
- $\mathcal{F}$ is a set of functions that determine $\boldsymbol{V}_i$
- P($\boldsymbol{U}$) is the joint probability distribution over $\boldsymbol{U}$

How to retrieve G in practice?

[1] Pearl, J. et al. Wiley & Sons. 2016.

Background on Causality (2)

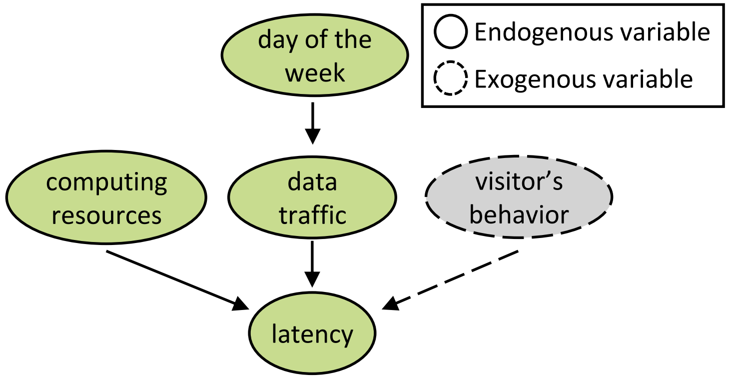
Causal discovery: given observational data, derive a causal model depicting the cause-and-effect relationships that exist within the system.

Day of the week	Computing Resources	Data traffic (Kbps)	Latency (s)
Monday	10%	150	0.8
Wednesday	50%	130	0.6
Saturday	80%	340	1.2
…	…	…	…

?

Causal discovery algorithm

- Peter-Clark (PC) [1]
- Greedy Equivalence Search (GES) [2]
- Fast Causal Inference (FCI) [3]



[1] P. Spirtes et al. MIT Press. 2001.
[2] D. M. Chickering. JMLR. 2002.
[3] E. Strobl et al. Int. J. Data Science and Analytics. 2018.

Causal RL for Self-adaptation

$\mathcal{S}$ = $\mu_{context}$ (retrieved from $\boldsymbol{G}$)

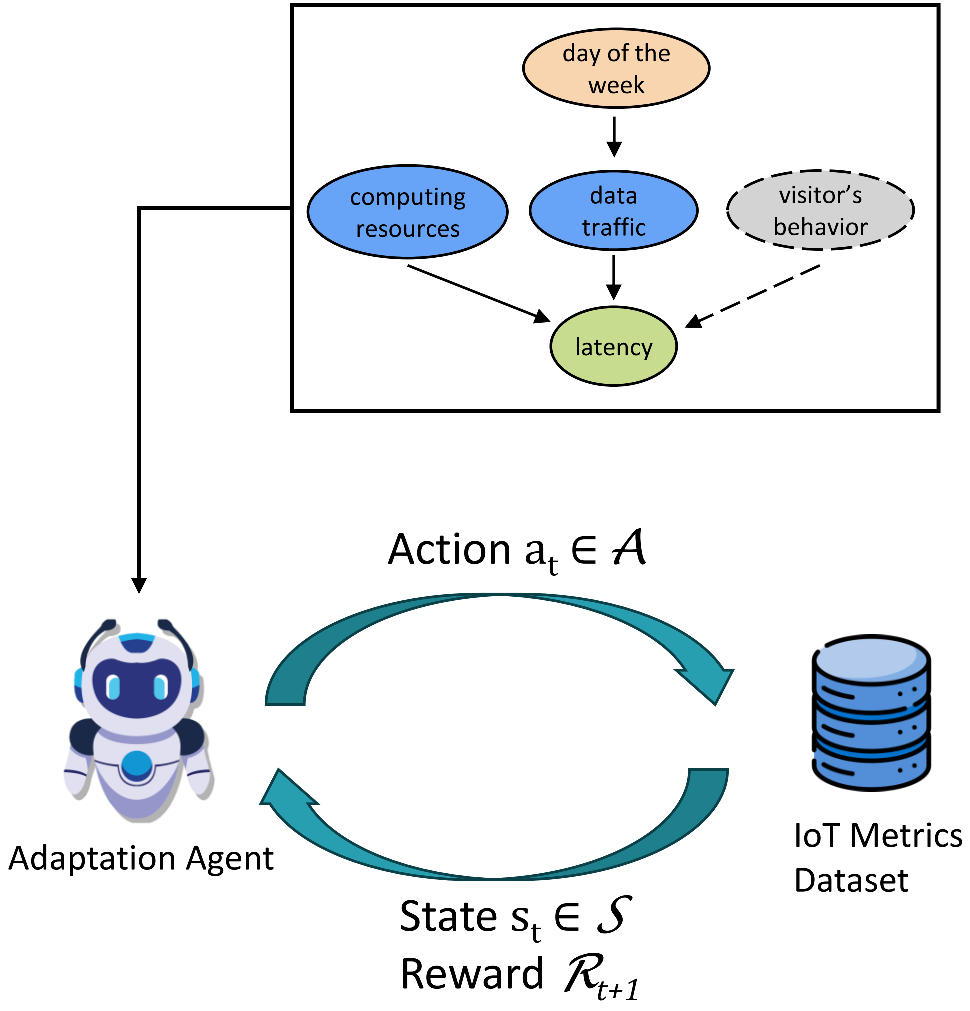
$\mathcal{A}$ = available actions for adaptation (bandwidth/resource allocation)

CRAFTER defines a reward function $\mathcal{R}$ to minimize the deviation between the QoS requirements and the observed performance of the system:

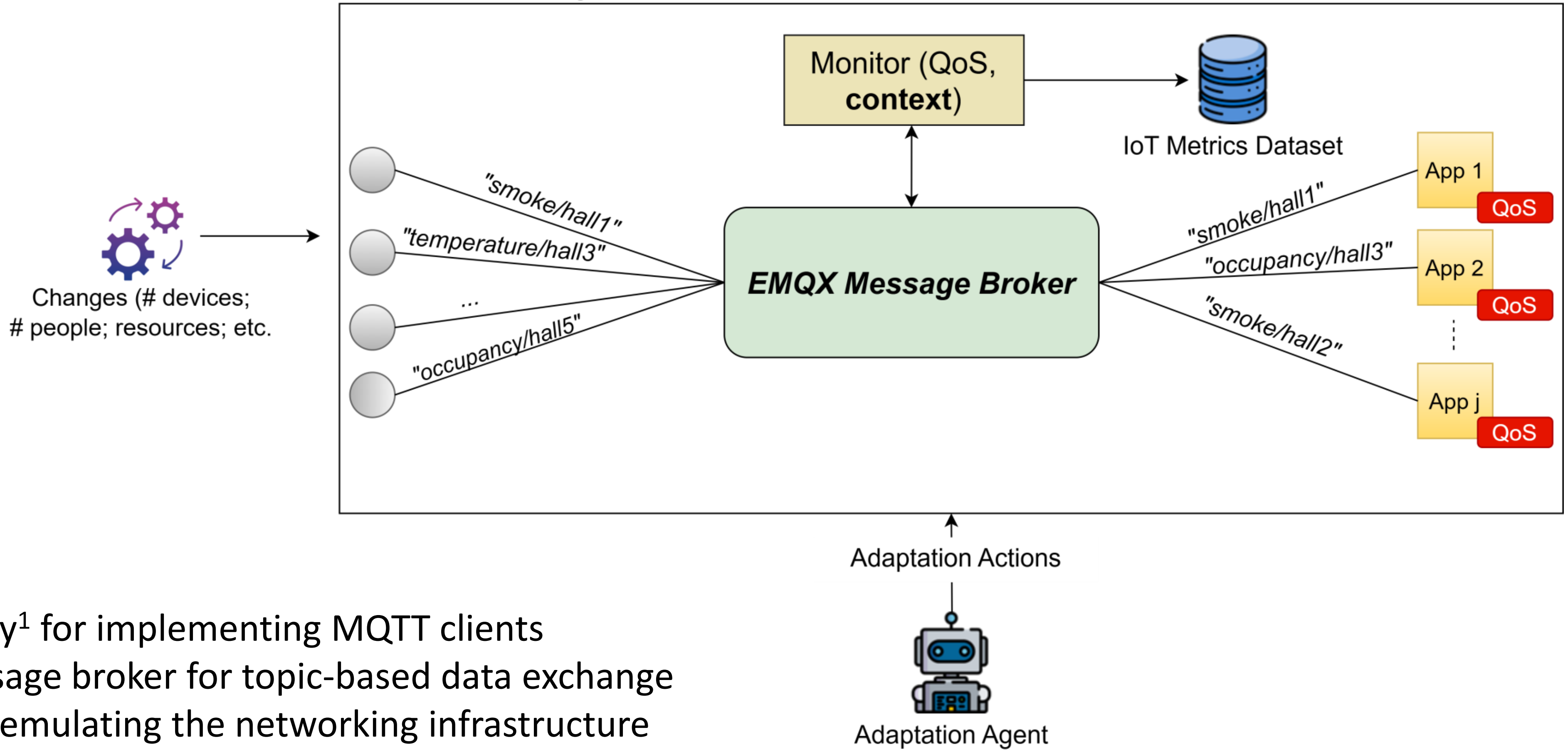
$$\mathcal{R} = -\Big(\sum_{a_i \in A} \underbrace{\boxed{\phi_i}}_{a_i\text{'s priority}} \underbrace{\boxed{\frac{\delta_{a_i} - \delta_{qos}}{\delta_{qos}}}}_{\text{deviation from QoS requirements}} + \underbrace{\boxed{c_1.w_i + c_2.\rho(e_i)}}_{\text{penalty terms}}\Big)$$

This leads the RL agent to learn the optimal policy π^*:

$$\pi^* = \arg\max_{\pi} \sum_t \underbrace{\boxed{\gamma^t}}_{\text{discount term to prioritize immediate rewards}} R_{\pi_{1\ldots t}}$$



Experimental Evaluation
The CRAFTER prototype implementation



- MQTT Paho Library[1] for implementing MQTT clients
- EMQX 5.6.1[2] message broker for topic-based data exchange
- Containernet[3] for emulating the networking infrastructure
- Docker containers to represent multiple locations in a smart space

1 https://eclipse.dev/paho
2 https://emqx.com
3 https://containernet.github.io/

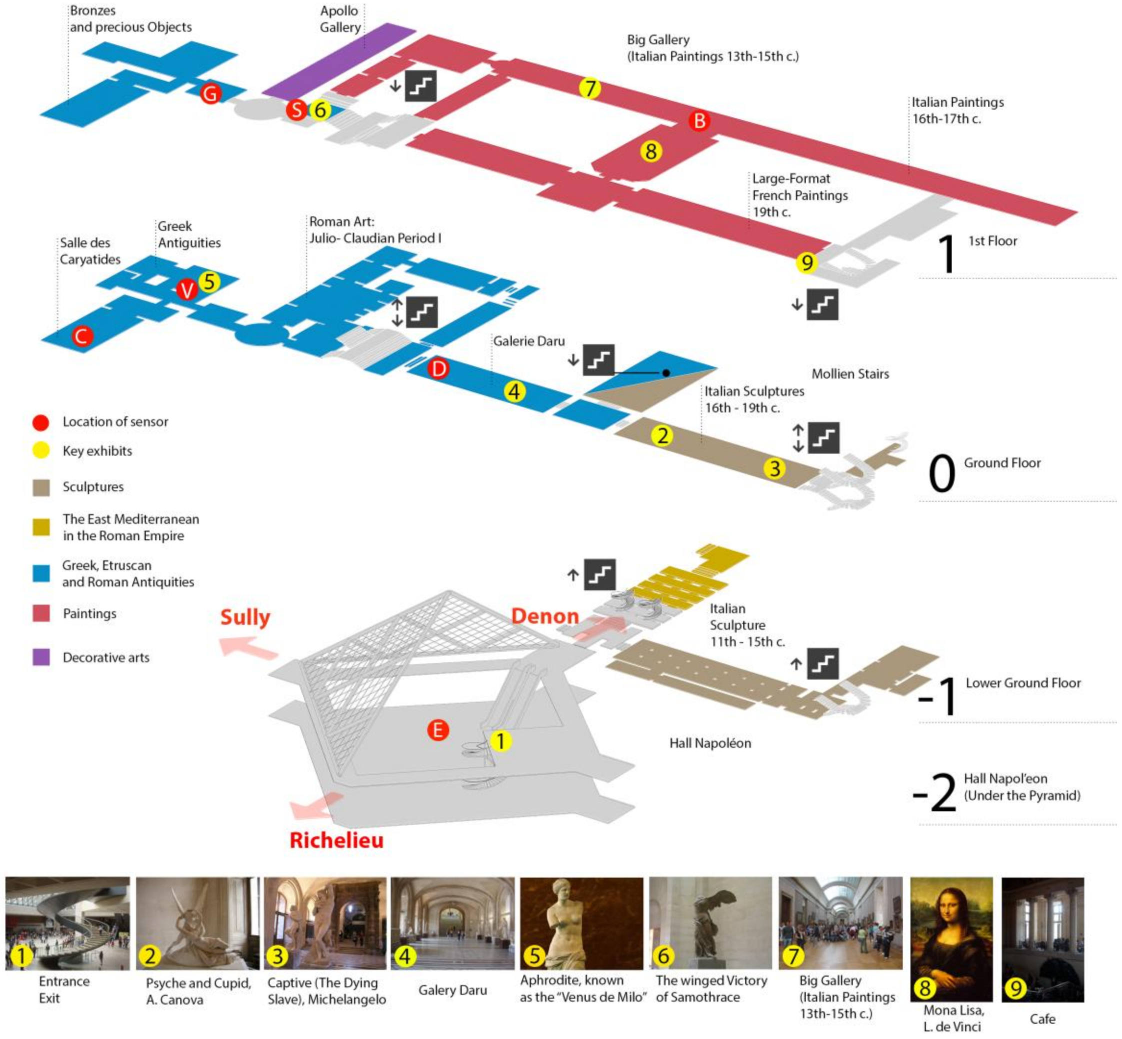
Experimental Evaluation

Experimental setup

Virtual Sensor	Received data	Processing technique
Occupancy Prediction	Camera feed, visitor's location	CNN for people detection, aggregation with visitors' locations
Queue Management	Visitor's location	Estimation of waiting times using linear regression
Security Analysis	Camera feeds, motion, door sensors	Aggregation of data for security threat detection
Predictive Maintenance	Temperature, humidity, air quality, proximity	Estimation of maintenance date using linear regression

Application	Received data	QoS requirements
Le Louvre Sub-metaverse	Lighting, noise, visitors' locations	20 ms
Smart lighting	Lighting	best effort
Visitor guiding	Occupancy, queueing times	400 ms
Artifact maintenance	Maintenance date	2 s
Security	Security alert	1s

7 exhibition halls, 112 devices, 5 application types, 4 virtual sensors, 2 Edge nodes.



Adapted from [1]

[1] Y. Yoshimura, Yuji et al. IEEE Pervasive Computing. 2017.

Experimental Evaluation

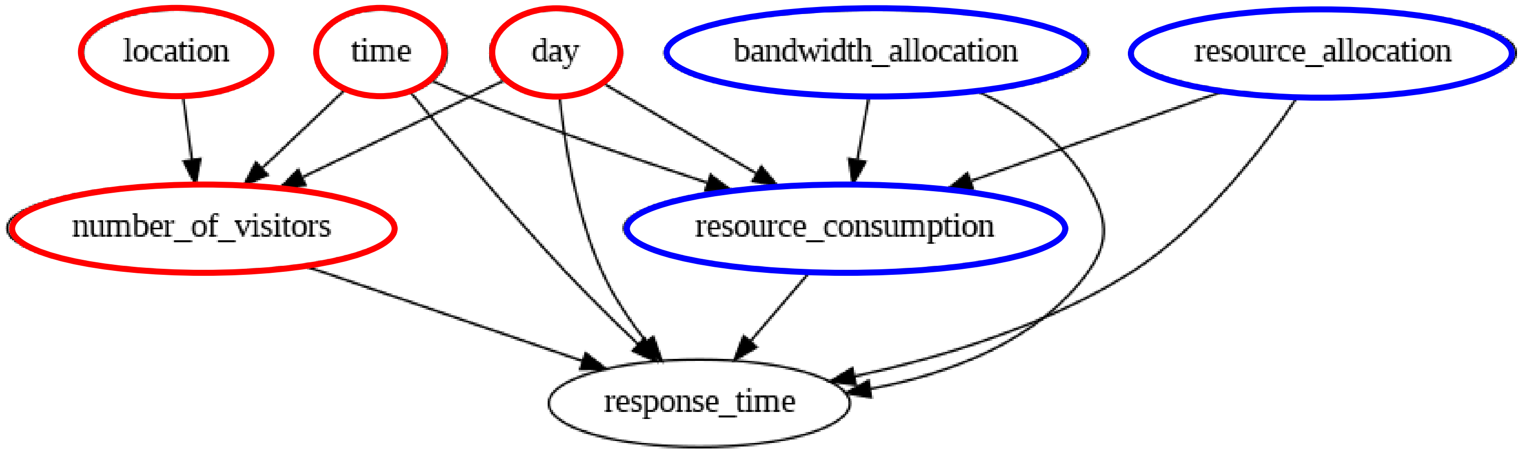
Identifying causal relationships with Causal Discovery

IoT Metrics Dataset $\mathbf{D}$

Days $\in$ {Wednesday $\rightarrow$ Monday}
Time $\in$ {Morning; Afternoon}
$W_{DX} \in$ [20 Mbps; 500 Mbps]
$\rho(e_i) \in [10\%; 100\%]$

1560 configurations

- PC algorithm to retrieve the causal graph.
- μ_{qos} = {bandwidth allocation, resource allocation, resource consumption}
- $\mu_{context}$ = {day of the week, time of the day, number of visitors, location}

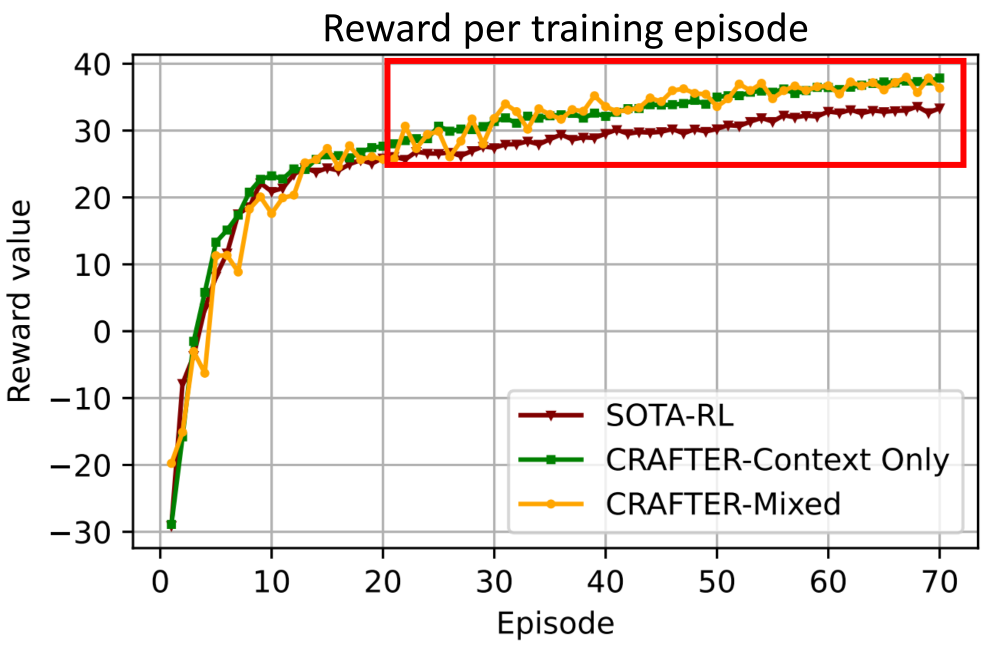


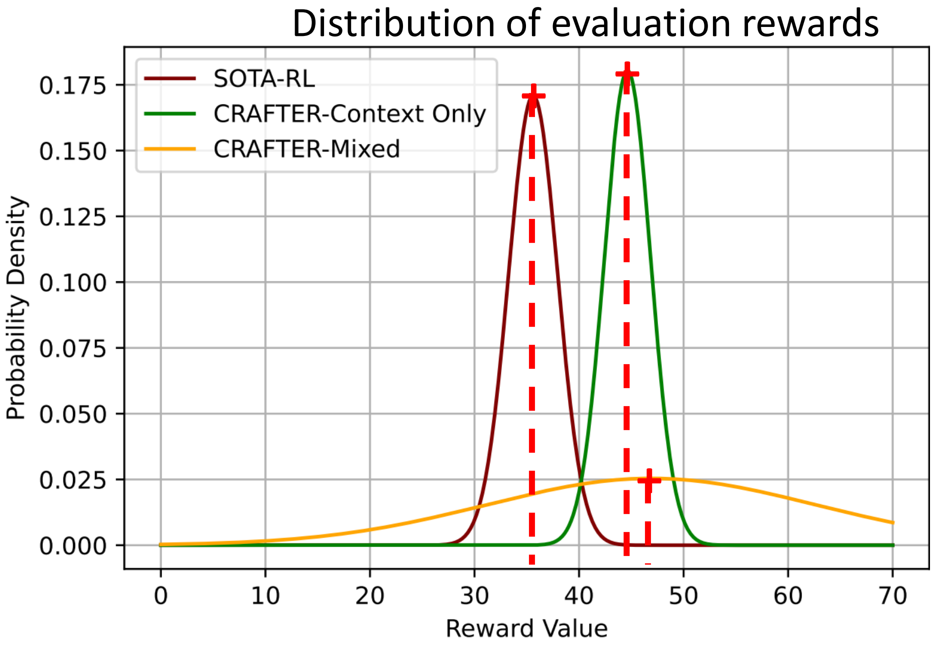
Experimental Evaluation

Learning Self-adaptation with Causal RL

- Adaptation Agents: `CRAFTER-Context Only` (uses $\mu_{context}$) and `CRAFTER-Mixed` (uses $\mu_{qos} \cup \mu_{context}$)
- $\mathcal{A}$ = {Bandwidth allocation W_{DX}, resource allocation $\rho(e_i)$}
- Stable-Baselines3 library[1] to train the RL agents in a Gymnasium[2] environment.
- Agents are trained using the IoT metrics dataset **D** and the PPO algorithm.

Causal RL enables a better training and inference performance over traditional RL



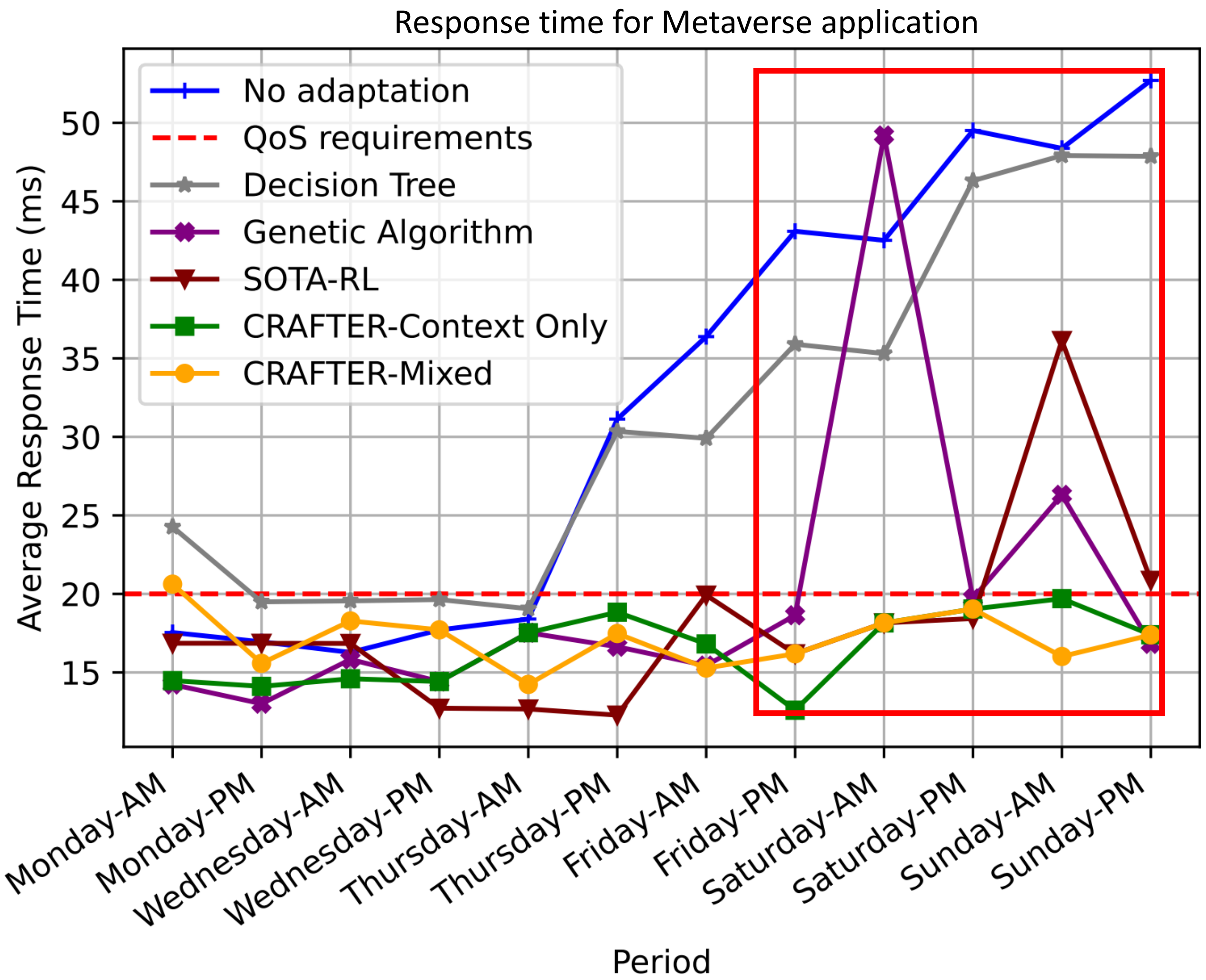


1 https://github.com/DLR-RM/stable-baselines3
2 https://gymnasium.farama.org/

Experimental Evaluation
Evaluating CRAFTER against existing approaches

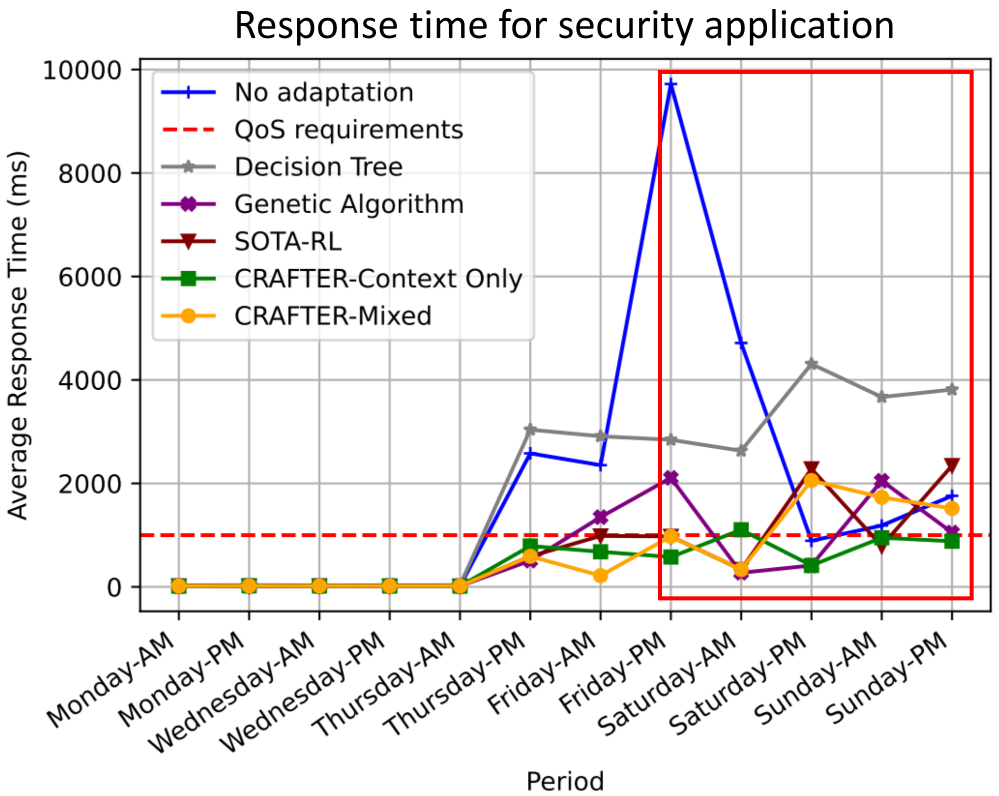
Evaluation against:
- Rule-based approaches (Decision Trees)
- Optimization approaches (Genetic Algorithms)
- SOTA RL approaches

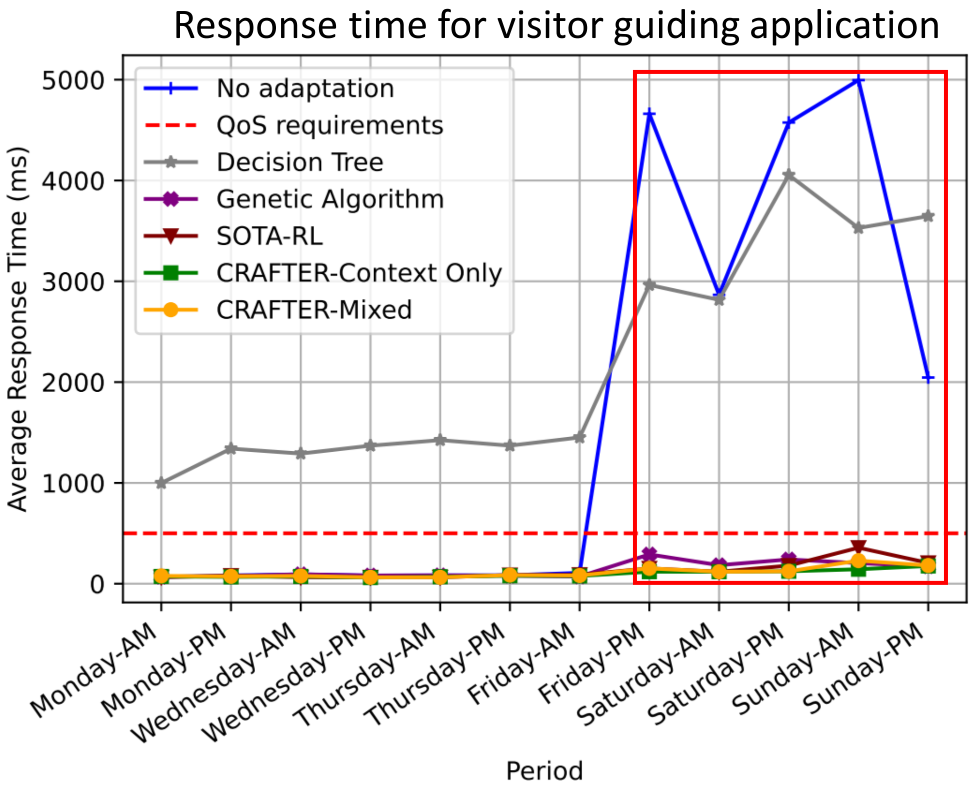
CRAFTER manages to effectively adapt to strict QoS requirements, even in very dynamic environments



Experimental Evaluation

Evaluating CRAFTER against existing approaches





Key Takeaways and Future Directions

- We propose CRAFTER, a causal-based self-adaptation framework for enabling autonomous IoT systems.
- We rely on causal discovery for identifying the causes of performance degradatation in IoT systems.
- We use Causal Reinforcement Learning for adapting to changes in dynamic IoT environments.
- The experimental results shows that CRAFTER achives a performance improvement of 25% over SOTA approaches.
- The CRAFTER code is publicy available on: https://github.com/houssamhh/crafter.
- How to adapt the system in response to *unseen changes?*
- How to detect and respond to causal graph drift when the underlying system dynamics change over time?

Thank You

CRAFTER: Causality-based Self-adaptation for Autonomous IoT Systems

Houssam Hajj Hassan[1, 2], Ajay Kattepur[3], Denis Conan[1], Georgios Bouloukakis[1, 3]

1 Télécom SudParis, Institut Polytechnique de Paris, France
2 Orange Innovation, France
3 Ericsson AI Research, India
4 Department of Electrical and Computer Engineering, University of Patras, Greece