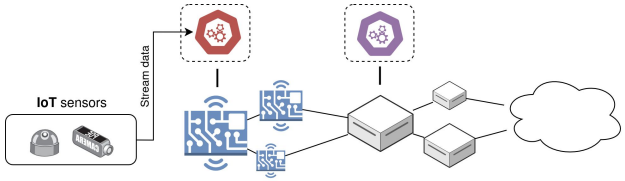
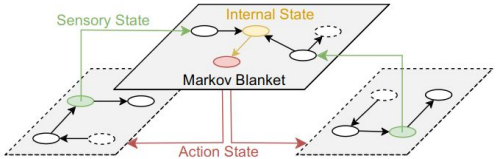
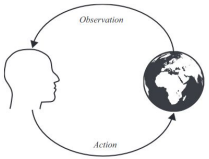
Autonomous Orchestration of Computing Continuum Systems through Active Inference

Boris Sedlak, TU Wien
Supervision: Prof. Schahram Dustdar

Agenda







I – Problem Domain

II – Contributions

III – Methodology

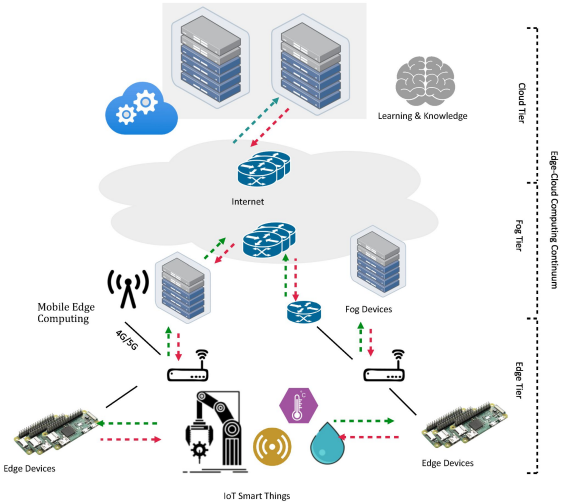
IV – Summary

I – Common Concepts

Computing Continuum (CC) as a composition of multiple processing tiers that stretch from IoT and edge computing, over Fog resources, to distant Cloud centers

Combines the benefits of all its tiers, i.e., low-latency and privacy-protecting computation from Edge, high availability and virtually unlimited processing resources from Cloud

Smart Cities are a common instance of distributed systems, where interconnected services (e.g., traffic surveillance or road surveillance) collaborate based on collected sensor data



Example of a Computing Continuum architecture [1]

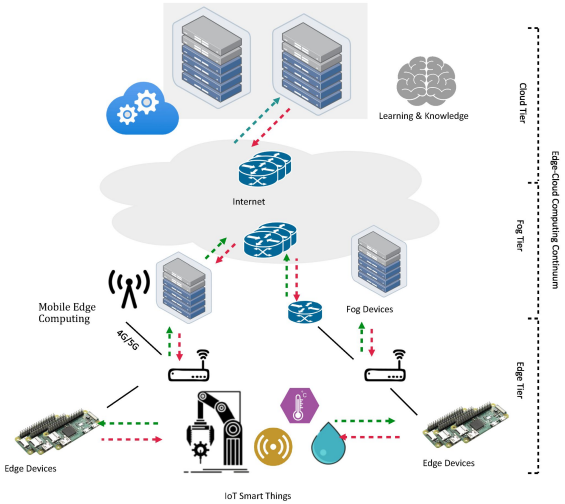
[1] P. Donta, I. Murturi, V. Casamayor, B. Sedlak, and S. Dustdar; **Exploring the Potential of Distributed Computing Continuum Systems** (2023)

I – Common Concepts

Computing Continuum (CC) as a composition of multiple processing tiers that stretch from IoT and edge computing, over Fog resources, to distant Cloud centers

Combines the benefits of all its tiers, i.e., low-latency and privacy-protecting computation from Edge, high availability and virtually unlimited processing resources from Cloud

Smart Cities are a common instance of distributed systems, where interconnected services (e.g., traffic surveillance or road surveillance) collaborate based on collected sensor data



Example of a Computing Continuum architecture [1]

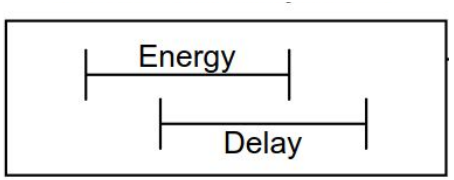
[1] P. Donta, I. Murturi, V. Casamayor, B. Sedlak, and S. Dustdar; **Exploring the Potential of Distributed Computing Continuum Systems** (2023)

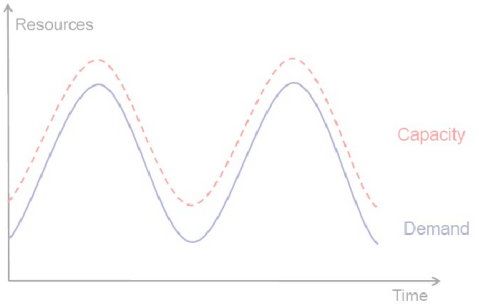
I – Common Concepts (cont.)

Service Level Objectives (SLOs) specify requirements that must be ensured throughout operation (e.g., latency < *t*). Narrow scope on generic performance indicators

Elasticity Strategies scale a system according to current demand; e.g., if performance is insufficient, allocate more resources. However, what if this does not fulfill SLOs?

Service Level Agreements (SLAs) as binding agreement between service provider and consumer. However, very limited support in resource-restricted environments





Elasticity allocates the right amount of resources [2]

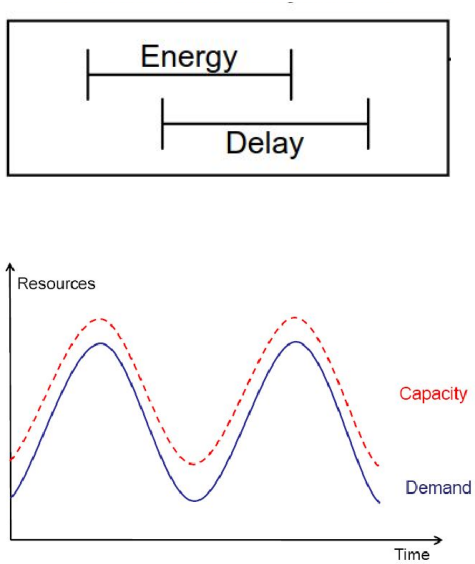
[2] Ricciardi et al., **Saving Energy in Data Center Infrastructures** (2011)

I – Common Concepts (cont.)

Service Level Objectives (SLOs) specify requirements that must be ensured throughout operation (e.g., latency < *t*). Narrow scope on generic performance indicators

Elasticity Strategies scale a system according to current demand; e.g., if performance is insufficient, allocate more resources. However, what if this does not fulfill SLOs?

Service Level Agreements (SLAs) as binding agreement between service provider and consumer. However, very limited support in resource-restricted environments



Elasticity allocates the right amount of resources [2]

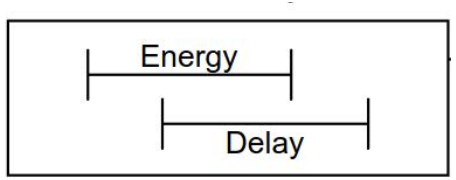
[2] Ricciardi et al., **Saving Energy in Data Center Infrastructures** (2011)

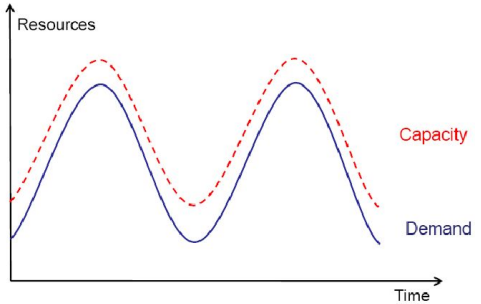
I – Common Concepts (cont.)

Service Level Objectives (SLOs) specify requirements that must be ensured throughout operation (e.g., latency < *t*). Narrow scope on generic performance indicators

Elasticity Strategies scale a system according to current demand; e.g., if performance is insufficient, allocate more resources. However, what if this does not fulfill SLOs?

Service Level Agreements (SLAs) as binding agreement between service provider and consumer. However, very limited support in resource-restricted environments





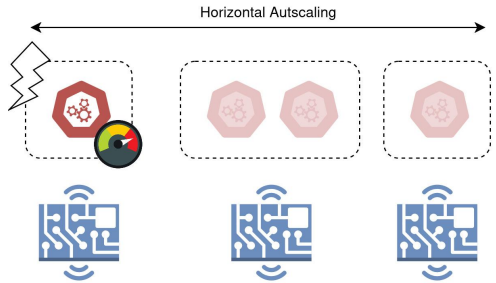
Elasticity allocates the right amount of resources [2]

[2] Ricciardi et al., **Saving Energy in Data Center Infrastructures** (2011)

I – Problem Summary

Naive assumption

Benchmark a service on an Edge device; performance looks fine → deploy it; if we fail our SLOs (e.g., high demand), start a fresh service instance nearby



Naive autoscaling on a set of homogeneous devices

Rigid elasticity models

Resources are scarce at the Edge and its often not possible to offload computation or scale services horizontally/vertically; vulnerable during runtime

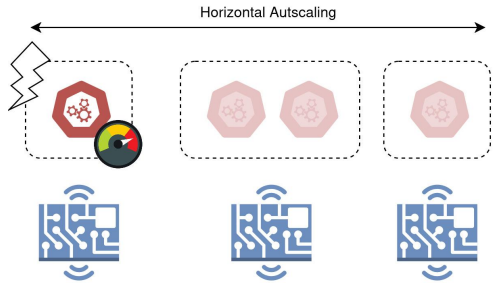
Complex service interactions

Elasticity strategies focus on local service state and don't consider impact on dependent services; actions by one service jeopardize SLO fulfillment of others

I – Problem Summary

Naive assumption

Benchmark a service on an Edge device; performance looks fine → deploy it; if we fail our SLOs (e.g., high demand), start a fresh service instance nearby



Naive autoscaling on a set of homogeneous devices

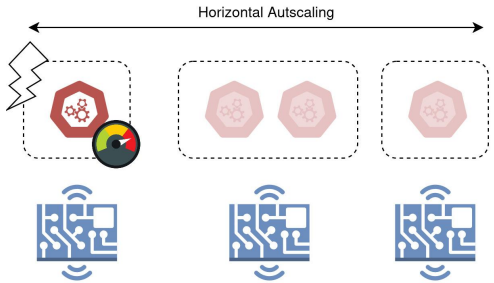
Rigid elasticity models

Resources are scarce at the Edge and it's often not possible to offload computation or scale services horizontally/vertically; vulnerable during runtime

Complex service interactions

Elasticity strategies focus on local service state and don't consider impact on dependent services; actions by one service jeopardize SLO fulfillment of others

I – Problem Summary



Naive autoscaling on a set of homogeneous devices

Naive assumption

Benchmark a service on an Edge device; performance looks fine → deploy it; if we fail our SLOs (e.g., high demand), start a fresh service instance nearby

Rigid elasticity models

Resources are scarce at the Edge and it's often not possible to offload computation or scale services horizontally/vertically; vulnerable during runtime

→

RQ.2 Flexible orchestration

How to quantify the potential impact of elasticity strategies and choose between them according to the current state

Complex service interactions

Elasticity strategies focus on local service state and don't consider impact on dependent services; actions by one service jeopardize SLO fulfillment of others

→

RQ.3 Composable models

How to analyze dependencies between services and use these insights for optimizing global SLO fulfillment

I – Problem Summary (cont.)

Inaccuracy of one-shot training
Train a model (of state transition probabilities) to infer optimal elasticity strategy; however, requires lots of training data; also, variable drifts perturb the model

ML algorithms as blackbox
DNN models are difficult to debug; cannot empirically verify *why* an elasticity strategy was used; leading to low trust in ML-based orchestration mechanisms

I – Problem Summary (cont.)

Inaccuracy of one-shot training
Train a model (of state transition probabilities) to infer optimal elasticity strategy; however, requires lots of training data; also, variable drifts perturb the model

ML algorithms as blackbox
DNN models are difficult to debug; cannot empirically verify *why* an elasticity strategy was used; leading to low trust in ML-based orchestration mechanisms

→

RQ.1 Continuous Learning
How to continuously adjust the model according to new observations without obstructing the inference;

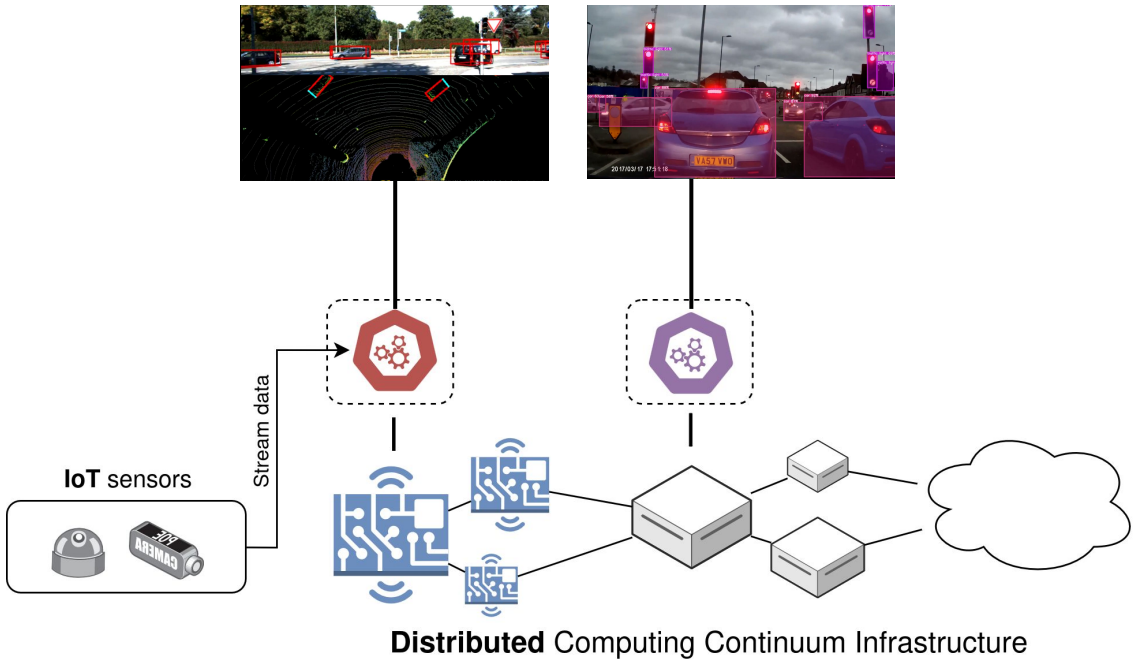
extracting causal patterns within data

II – Overview of Contributions

Distributed processing

– Distributed CC infrastructure composed of various **device types** at different **locations**

– Sensor data continuously streamed from IoT devices to different processing services



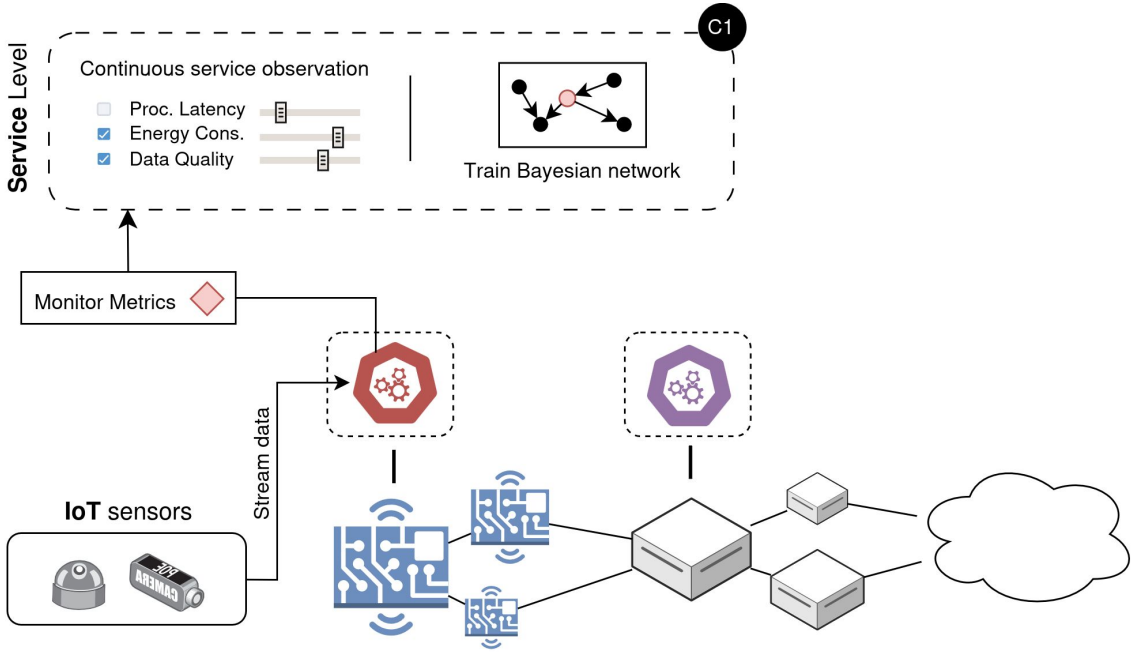
II – Overview of Contributions (C1)

Service interpretation

– Monitor processing across CC by collecting metrics; eval. real-time SLO fulfillment

– Train a model (i.e., BN) for predicting SLO fulfillment of **individual** services in different deployment configurations [7]

– Retrain model according to SLO prediction accuracy; slows down as model improves [9]



[7] Sedlak et al., **Designing Reconfigurable Intelligent Systems with Markov Blankets**, at ICSOC 2023
[9] Sedlak et al., **SLO-Aware Task Offloading Within Collaborative Vehicle Platoons**, at ICSOC 2024

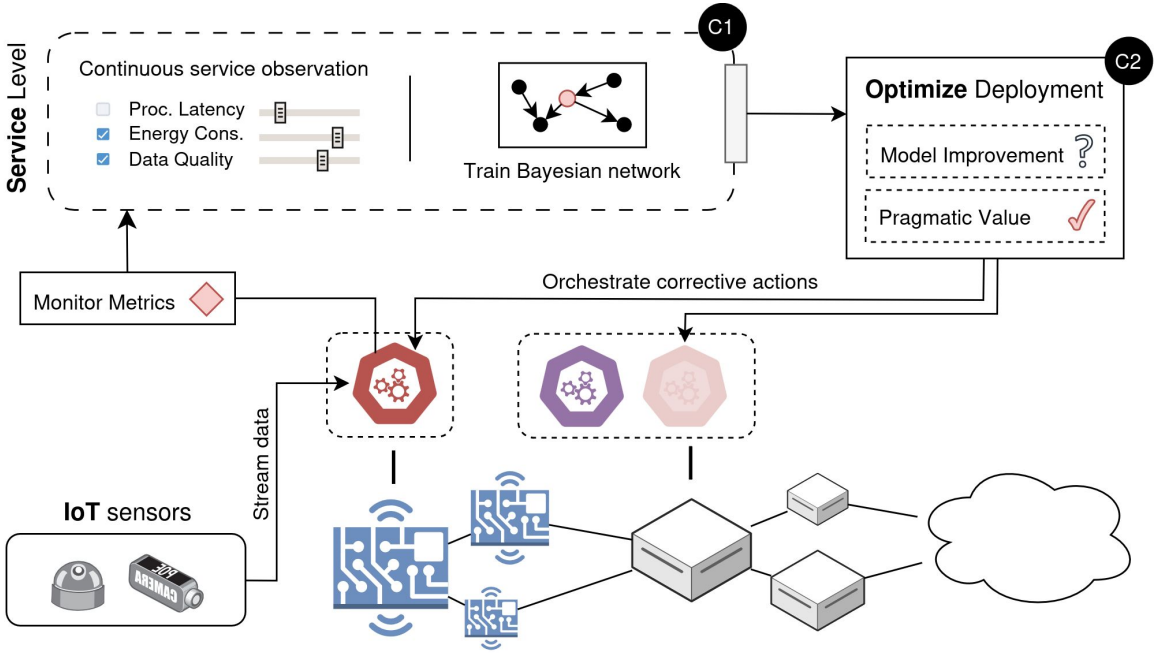
II – Overview of Contributions (C2)

Service adaptation

– Optimize **local SLO fulfillment** by reconfiguring processing services; choose between avail. elasticity strategies [5]

– Escape local optima through continuous exploration; check poss. **model improvement** [12]

– Decisions can be empirically verified and interpreted; useful for non-technical explanation



[5] Sedlak et al., **From Metrics to Multidimensional Elasticity Strategies**, at IEEE Services EDGE 2023
[12] Sedlak et al., **Active Inference on the Edge: A Design Study**, at PerconAI 2024

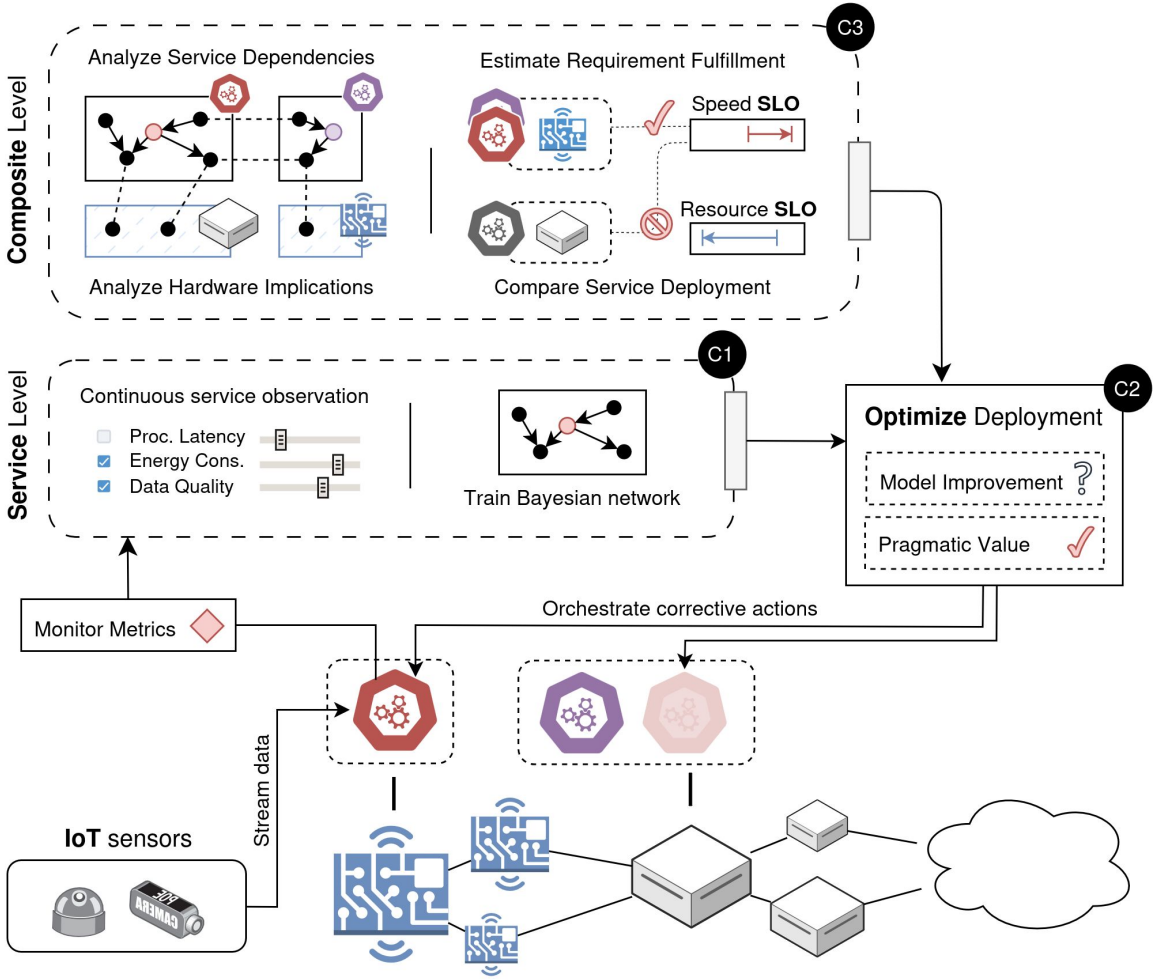
II – Overview of Contributions (C3)

Service collaboration

– Compose individual models to overarching representation; quantify service dependencies and **impact on hardware** [4]

– Collaborative orchestration considers service relations to optimize global SLO fulfillment

– Exchanging and merging BNs between services to speed up the **onboarding** of new service types or devices types [13]



[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Service EDGE 2024
[13] Sedlak et al., **Equilibrium in the Computing Continuum through Active Inference**, at Elsevier FGCS, 2024

III – Methodologies Overview

Central methodologies

– How to quantify dependencies between systems?
– How to align a CC system towards a high-level SLO?

Refined methodologies

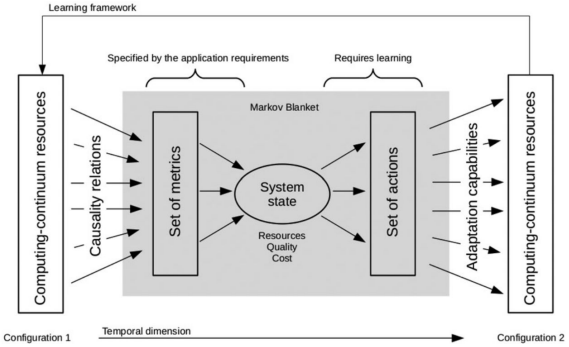
– How to ensure the accuracy of service models?
– How to debug the behavior of scaling agents?

III – Markov Blanket (MB)

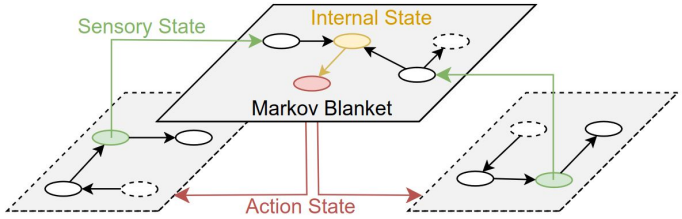
Interactions between **systems** (e.g., human in world) can be expressed through MBs – fulfill Markov property; allow modeling reactive behavioral models for elasticity

Creates **formal boundary** between a system and external states – limits scope of variables that determine **internal state**; discard remaining information to reduce dimension

Provides clear interfaces for **sensory** and action **states**; policy (e.g., scaling) as a mapping between these states



Behavioral Markov blanket of a system [3]



Action-perception cycle between multiple entities [4]

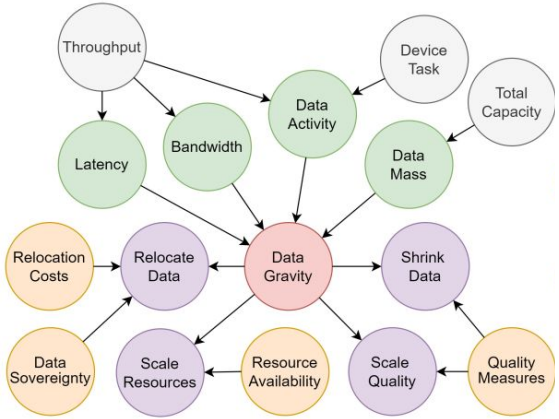
[3] Dustdar et al., **On Distributed Computing Continuum Systems** (2023)
[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services Edge 2024

III – SLOs and Behavioral Models

MB: Expresses how to evaluate a composite SLO and how to react according to the current device context

Behavioral model

Internal state (●) evaluates objectives and how these relate to external sensory inputs (●); can interact with the world through action, i.e., elasticity strategies (●), which are influenced by contextual factors (●)



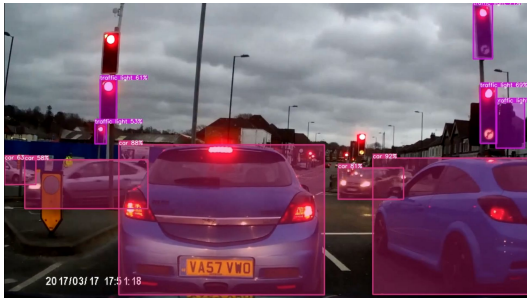
Example of a behavioral model for data gravity [5]

[5] Sedlak et al., **Controlling Data Gravity and Data Friction: From Metrics to Multidimensional Elasticity Strategies**, at IEEE Service EDGE 2023

III – Stream Processing Scenarios

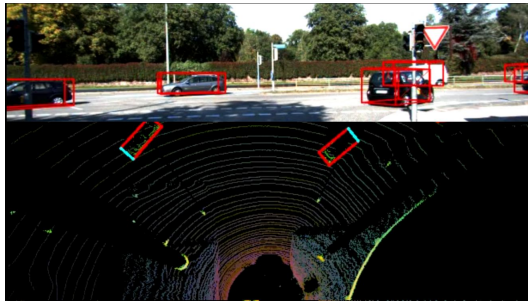
Commonly addressed use cases revolve around continuous **stream processing**; in case **time-critical** adaptations are required, this poses a higher need for sophisticated adaptation mechanisms.

Video Processing (Yolo V8)



Object detection in a video stream using Yolo [6]

Mobile Mapping (Lidar)



Creating a mobile map from binaries using Lidar [6]

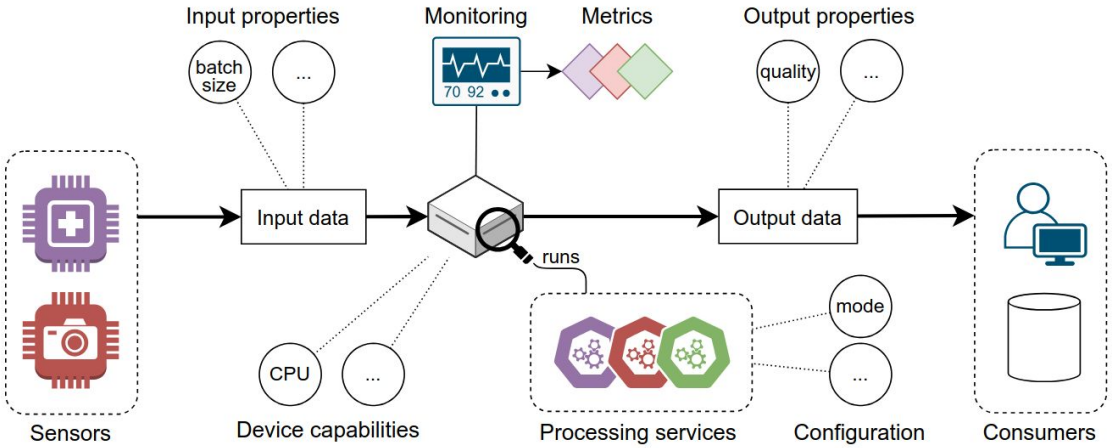
QR Scanner (OpenCV)

QR code scanning in a video using OpenCV [6]

[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

III – Stream Processing Scenarios

Commonly addressed use cases revolve around continuous **stream processing**; in case **time-critical** adaptations are required, this poses a higher need for sophisticated adaptation mechanisms.

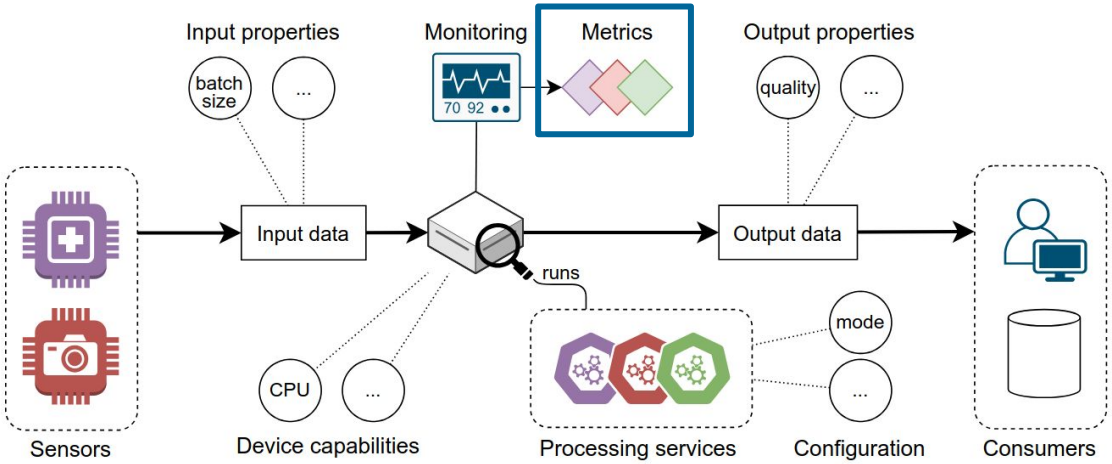


Abstract representation of a monitored stream processing service [6]

[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

III – Stream Processing Scenarios

Commonly addressed use cases revolve around continuous **stream processing**; in case **time-critical** adaptations are required, this poses a higher need for sophisticated adaptation mechanisms.



Abstract representation of a monitored stream processing service [6]

fps	pixel	cores	change_flag	timestamp
31	800	3	False	2024-11-30
32	800	3	False	2024-11-30
32	800	3	False	2024-11-30
32	800	3	False	2024-11-30
32	800	3	False	2024-11-30
30	800	3	False	2024-11-30
32	800	3	False	2024-11-30
32	800	3	False	2024-11-30
32	800	3	False	2024-11-30

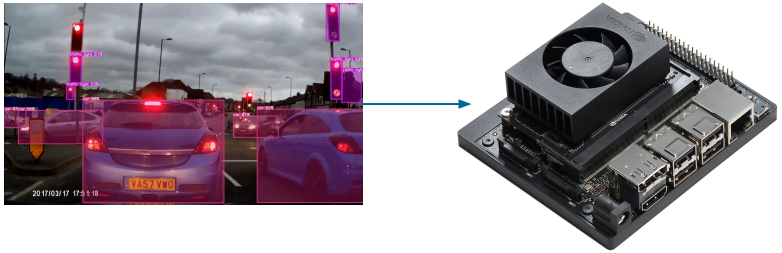
Example of processing metrics as tabular data [6]

[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

III – Multi-dimensional Elasticity

Target: use processing metrics and properties to train a *generative model* that describes the process; create an interpretable representation of service behavior; orchestrate service optimally within current context

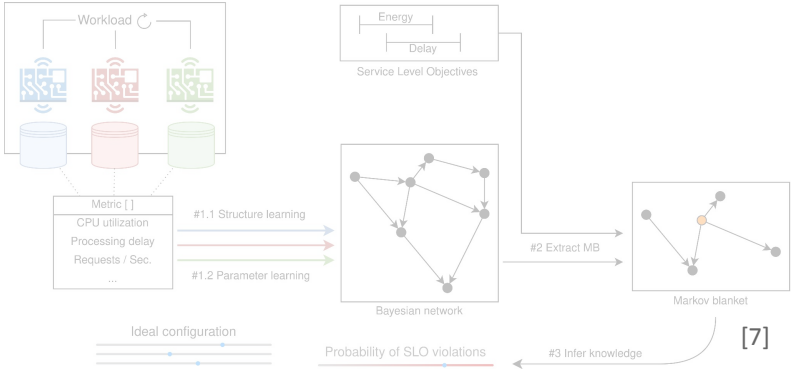
Jetson Xavier [7]



Resulting model contains:

- ❏ Target objectives (i.e., SLOs)
- ❏ Influential system factors
- ❏ Available elasticity strategies

3-Step basic methodology for providing this model through (**1**) Bayesian Network Learning (BNL), (**2**) Markov Blanket (MB) extraction, and (**3**) Inference.



[7]

[7] Sedlak et al., **Designing Reconfigurable Intelligent Systems with Markov Blankets**, at ICSOC 2023

III – Multi-dimensional Elasticity

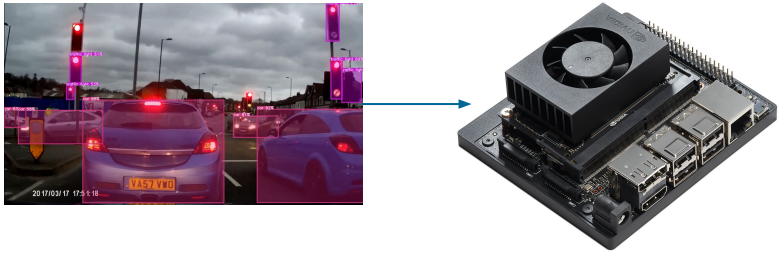
Target: use processing metrics and properties to train a *generative model* that describes the process; create an interpretable representation of service behavior; orchestrate service optimally within current context

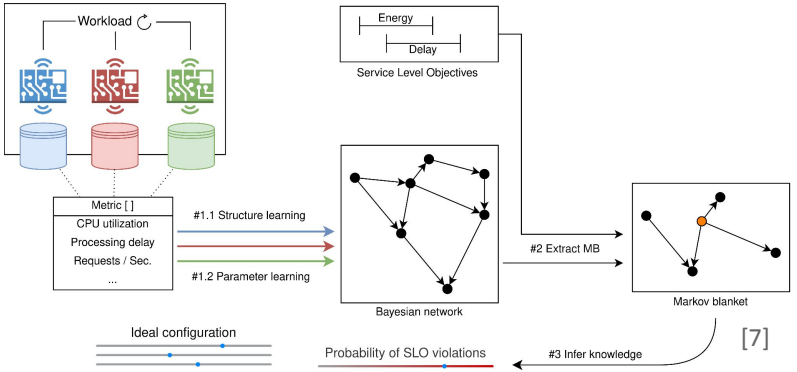
Resulting model contains:

- ❏ Target objectives (i.e., SLOs)
- ❏ Influential system factors
- ❏ Available elasticity strategies

3-Step basic methodology for providing this model through (**1**) Bayesian Network Learning (BNL), (**2**) Markov Blanket (MB) extraction, and (**3**) Inference.

Jetson Xavier [7]

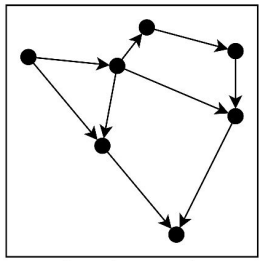




[7] Sedlak et al., **Designing Reconfigurable Intelligent Systems with Markov Blankets**, at ICSOC 2023

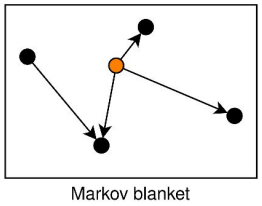
III – Multi-dimensional Elasticity (cont.)

Bayesian Network Learning



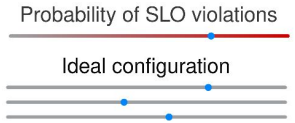
- **Structure Learning**
  Various algorithms (e.g., HCS)
  Directed Acyclic Graph (DAG)
- **Parameter Learning**
  Various algorithms (e.g., MLE)
  Conditional Prob. Table (CPT)

Markov Blanket Selection



- **Causality filter**
  Extract subset of variables that impact SLO fulfillment
- **Behavioral MB**
  MB now contains contextual factors & elasticity strategies

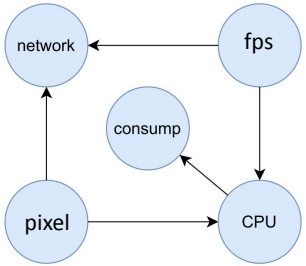
Knowledge Extraction



- **Conditional Inference**
  Estimate impact of different deployment configuration
- **Optimize SLOs**
  Adjust processing services according to optimal policy

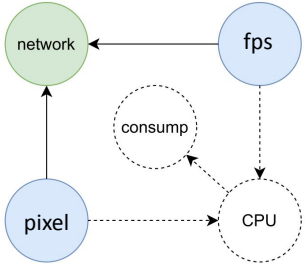
III – Multi-dimensional Elasticity (cont.)

Bayesian Network Learning



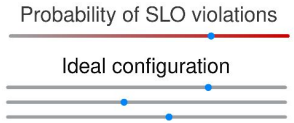
- **Structure Learning**
  Various algorithms (e.g., HCS)
  Directed Acyclic Graph (DAG)
- **Parameter Learning**
  Various algorithms (e.g., MLE)
  Conditional Prob. Table (CPT)

Markov Blanket Selection



- **Causality filter**
  Extract subset of variables that impact SLO fulfillment
- **Behavioral MB**
  MB now contains contextual factors & elasticity strategies

Knowledge Extraction



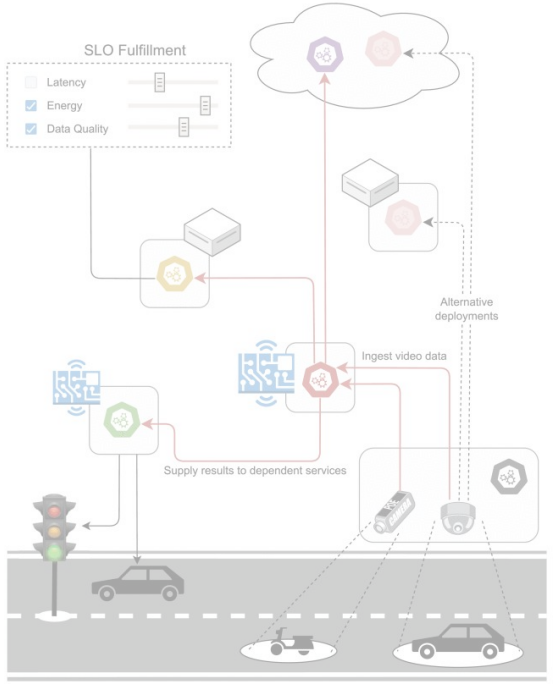
- **Conditional Inference**
  Estimate impact of different deployment configuration
- **Optimize SLOs**
  Adjust processing services according to optimal policy

III – Applying our Approach

Transitive Requirements [4]
SLOs posed by consumers determine the service quality that each “link” has to provide; **compose** MBs of dependent services to find implications and optimize deployment

Spanning CC with SLOs [8]
Microservice architectures composed of various services with SLOs for user-facing layer, e.g., latency or quality; infer lower-level SLOs and parameters for influential services

SLO-Aware Offloading [9]
Offloading a task to a resource-restricted device jeopardizes SLO fulfillment of existing services; estimate the implication to global SLO fulfillment to find suitable device hosts



Optimizing the deployment of microservice pipelines according to the SLOs posed for each service [4]

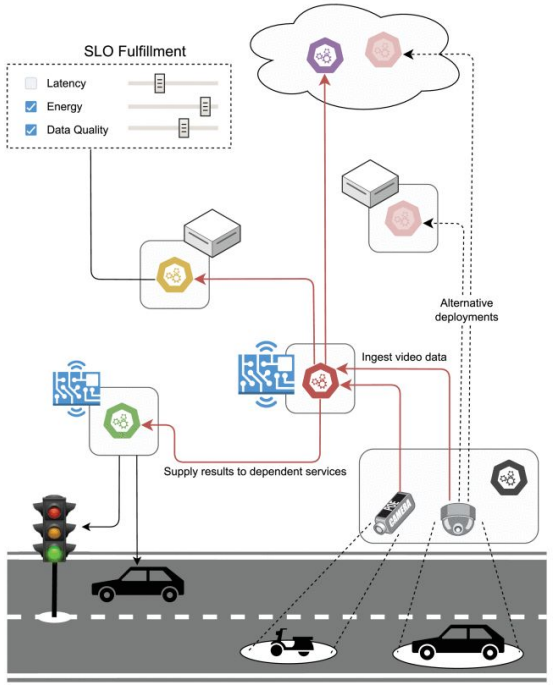
[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024
[8] Sedlak et al., **Diffusing High-level SLO in Microservice Pipelines**, at IEEE SOSE 2024
[9] Sedlak et al., **SLO-Aware Task Offloading Within Collaborative Vehicle Platoons**, at ICSOC 2024

III – Applying our Approach

Transitive Requirements [4]
SLOs posed by consumers determine the service quality that each “link” has to provide; **compose** MBs of dependent services to find implications and optimize deployment

Spanning CC with SLOs [8]
Microservice architectures composed of various services with SLOs for user-facing layer, e.g., latency or quality; infer lower-level SLOs and parameters for influential services

SLO-Aware Offloading [9]
Offloading a task to a resource-restricted device jeopardizes SLO fulfillment of existing services; estimate the implication to global SLO fulfillment to find suitable device hosts



Optimizing the deployment of microservice pipelines according to the SLOs posed for each service [4]

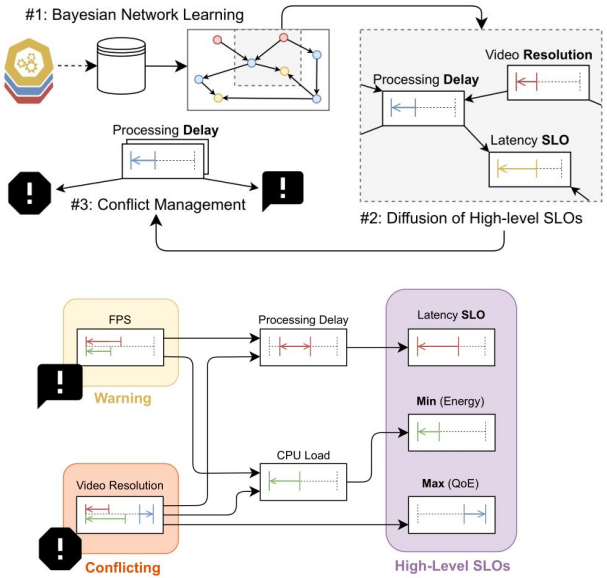
[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024
[8] Sedlak et al., **Diffusing High-level SLO in Microservice Pipelines**, at IEEE SOSE 2024
[9] Sedlak et al., **SLO-Aware Task Offloading Within Collaborative Vehicle Platoons**, at ICSOC 2024

III – Applying our Approach

Transitive Requirements [4]
SLOs by stream consumers determine the service quality that each “link” has to provide; **compose** MBs of dependent services to find implications and optimize deployment

Spanning CC with SLOs [8]
Microservice architectures composed of various services with SLOs for user-facing layer, e.g., latency or quality; infer lower-level SLOs and parameters for influential services

SLO-Aware Offloading [9]
Offloading a task to a resource-restricted device jeopardizes SLO fulfillment of existing services; estimate the implication to global SLO fulfillment to find suitable device hosts



Constraining a CC system by diffusing high-level SLOs to lower-level SLOs; highlights potential conflicts [8]

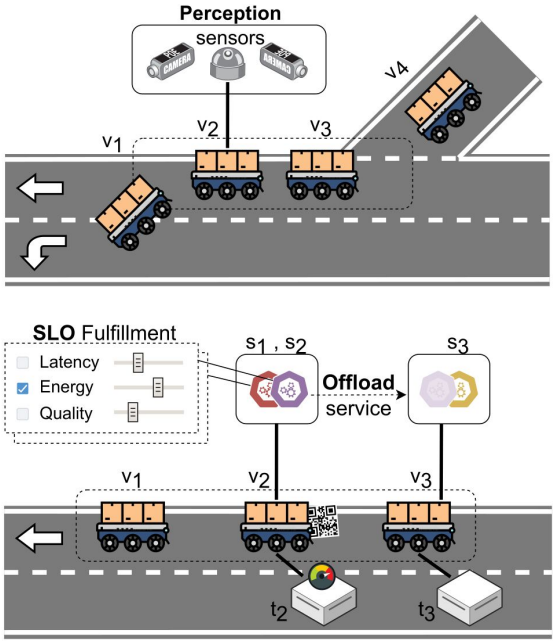
[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024
[8] Sedlak et al., **Diffusing High-level SLO in Microservice Pipelines**, at IEEE SOSE 2024
[9] Sedlak et al., **SLO-Aware Task Offloading Within Collaborative Vehicle Platoons**, at ICSOC 2024

III – Applying our Approach

Transitive Requirements [4]
SLOs by stream consumers determine the service quality that each “link” has to provide; **compose** MBs of dependent services to find implications and optimize deployment

Spanning CC with SLOs [8]
Microservice architectures composed of various services with SLOs for user-facing layer, e.g., latency or quality; infer lower-level SLOs and parameters for influential services

SLO-Aware Offloading [9]
Offloading a task to a resource-restricted device jeopardizes SLO fulfillment of existing services; estimate the implication to global SLO fulfillment to find suitable device hosts



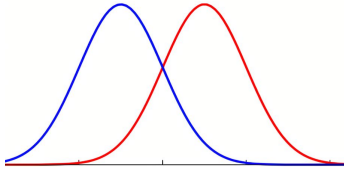
Offload computation between vehicles if this improves global SLO fulfillment [9]

[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024
[8] Sedlak et al., **Diffusing High-level SLO in Microservice Pipelines**, at IEEE SOSE 2024
[9] Sedlak et al., **SLO-Aware Task Offloading Within Collaborative Vehicle Platoons**, at ICSOC 2024

III – Refining our Approach

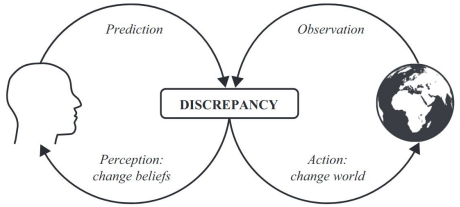
Known Shortcomings

(**1**) BNL requires large amounts of training data upfront;
(**2**) if discrete, must visit all possible states (e.g., scaling actions);
(**3**) over time, models get distorted due to variable drifts



Active Inference

Concept from **neuroscience** developed by Friston et al. [10,11]; allows agents to interact with their environment by learning the underlying **generative models** to persist over time



Action-perception cycle in Active Inference [11]

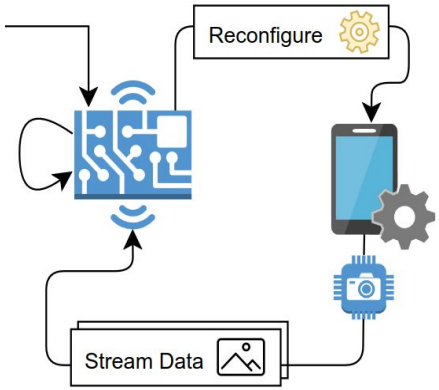
[10] Parr et al., **Active Inference: The Free Energy Principle in Mind, Brain, and Behavior** (2022)
[11] Friston et al., **Designing ecosystems of intelligence from first principles** (2024)

III – Active Inference in CC Systems

Mapping between neuroscience and distributed computing systems [6,12,13]; understanding processing requirements (i.e., SLOs) as a form of **homeostasis,** e.g., cell temperature

Create autonomous components that identify how to ensure requirements and resolve them independently, clear modelling between higher-level and low-level components

Simplify service orchestration in large-scale distributed systems; decentralized decision-making of individual components **avoids** transferring service states to the Cloud



Ensure internal requirements [13]

[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)
[12] Sedlak et al., **Active Inference on the Edge: A Design Study**, at PerconAI 2024
[13] Sedlak et al., **Equilibrium in the Computing Continuum through Active Inference**, Elsevier FGCS (2024)

III – Active Inference Architecture

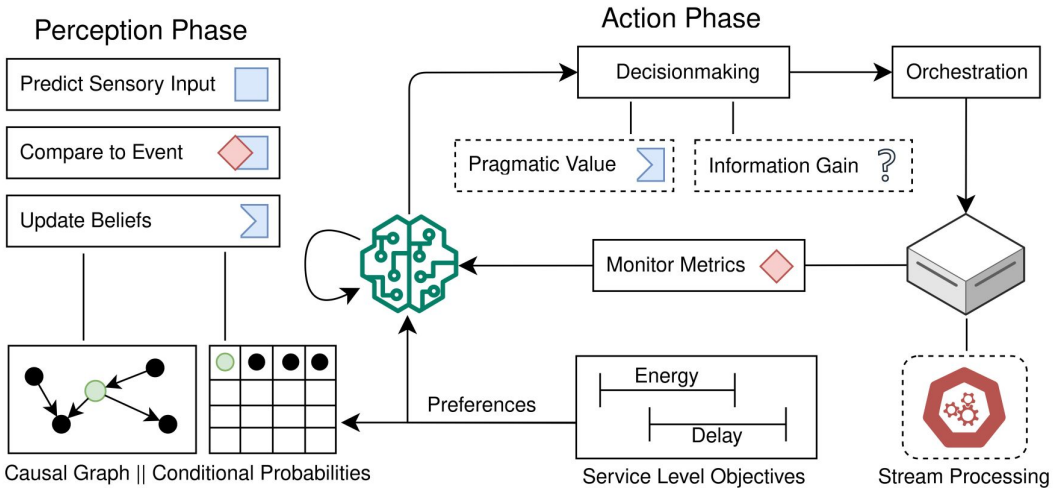
Approach

(1) **Specify** ideal runtime behavior through SLOs

(2) **AIF agents** monitor their environment & collect metrics

(3) **Perception phase** predicts expected SLO fulfillment and adjusts the generative model

(4) **Action phase** orchestrates the processing environment to optimize both SLOs and model



Action and perception cycles performed by the AIF agent to create an accurate model and shape the world [6]

[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

III – Active Inference Architecture (cont.)

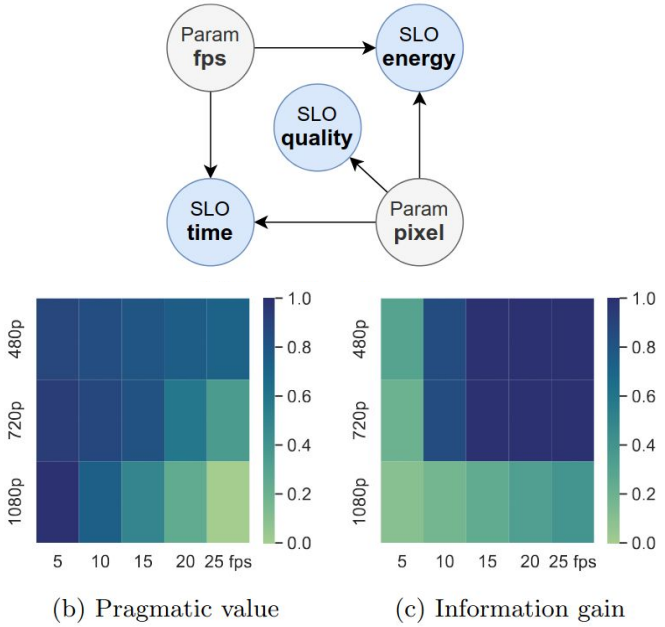
Interpretable behavior [6]

– Empirically verify variable relations in the BNs, e.g., increasing quality (pixel) leads to high energy usage; adjust **parameters** (i.e., pixel & fps) according to SLOs

– Quantified preferences of the agent: (1) expected SLO fulfillment or (2) potential model improvement; determine the behavior of the scaling agent

Continuous composition [4]

– Gradually create increasingly accurate models for individual processing services; continuously compose to estimate the **impact** they have on each other



(b) Pragmatic value (c) Information gain

[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024
[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

III – Active Inference Architecture (cont.)

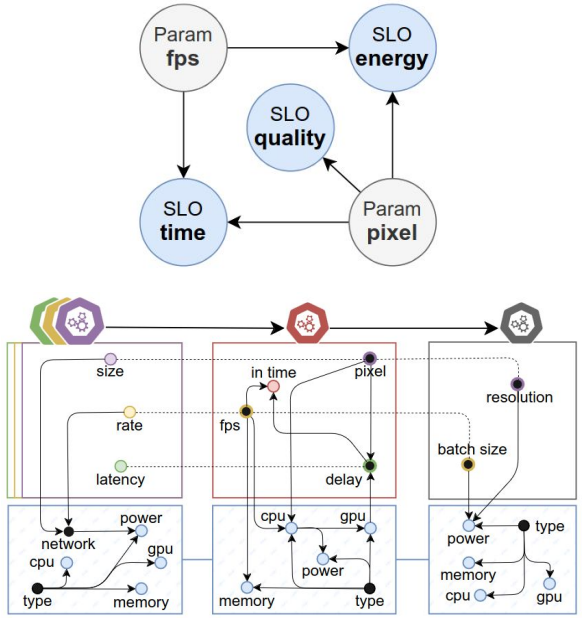
Interpretable behavior [6]

– Empirically verify variable relations in the BNs, e.g., increasing quality (pixel) leads to high energy usage; adjust **parameters** (i.e., pixel & fps) according to SLOs

– Quantified preferences of the agent: (1) expected SLO fulfillment or (2) potential model improvement; determine the behavior of the scaling agent

Continuous composition [4]

– Gradually create increasingly accurate models for individual processing services; continuously compose to estimate the **impact** they have on each other



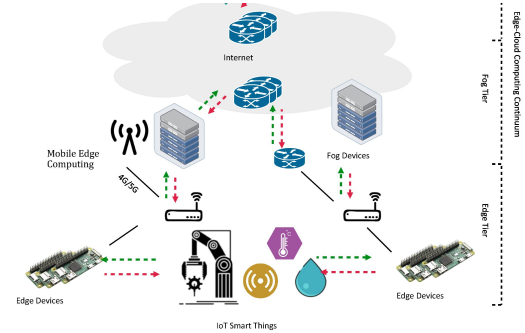
[4] Sedlak et al., **Markov Blanket Composition of SLOs**, at IEEE Services EDGE 2024
[6] Sedlak et al., **Adaptive Stream Processing on Edge Devices through Active Inference** (Scheduled for 2025 at Springer ES)

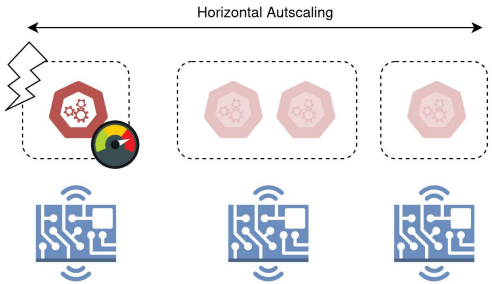
IV – Summary
Contemporary Challenges

IoT & Edge enable large-scale distributed services that optimize our daily routines; CC as underlying infrastructure for supporting these services

Processing SLOs must be continuously ensured to guarantee safe and satisfactory service operation; however, resource limitations and heterogeneity of resources complicate service orchestration

Missing flexible orchestration solutions that choose actions according to current context; policy adjusted continuously according to runtime dynamics





IV – Summary
Contributions & Results

Monitor processing services closely, analyze their behavior, and infer optimal elasticity strategy; train a MB that combines all these factors in one model

Dynamically optimize SLOs fulfillment for a network of processing services and heterogeneous devices; orchestrate services (i.e., placement, configuration, replication, etc) according to current context

Active Inference as a natural fit to train behavioral MBs and keep them accurate; balance continuously between ensuring SLOs and improving the model