Diffusing High-level SLOs in Microservice Pipelines

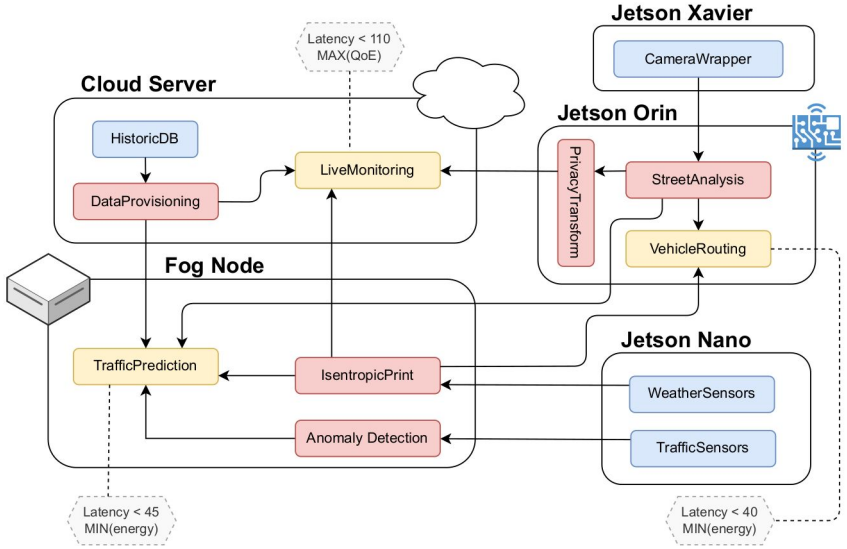
TU Wien: Boris Sedlak, Victor Casamayor Pujol, Praveen Kumar Donta, Schahram Dustdar

Problem Statement

- Microservice pipelines **distributed** over computing infrastructure, i.e., Edge to Cloud, unclear implications of individual services to high-level requirements, i.e., **SLOs**
- Multiple tenant and vendors (= stakeholders) specify **opposing** SLOs, e.g., minimize energy and response time, which results in **contradicting** service configurations
- Stakeholders not aware what their SLO implies for lower level components and services, e.g., **energy** → cpu

---

- Requires multiple layers of SLOs that specify application requirements through **fine-grained** control mechanism, i.e., diffusing high-level SLOs into lower level SLOs

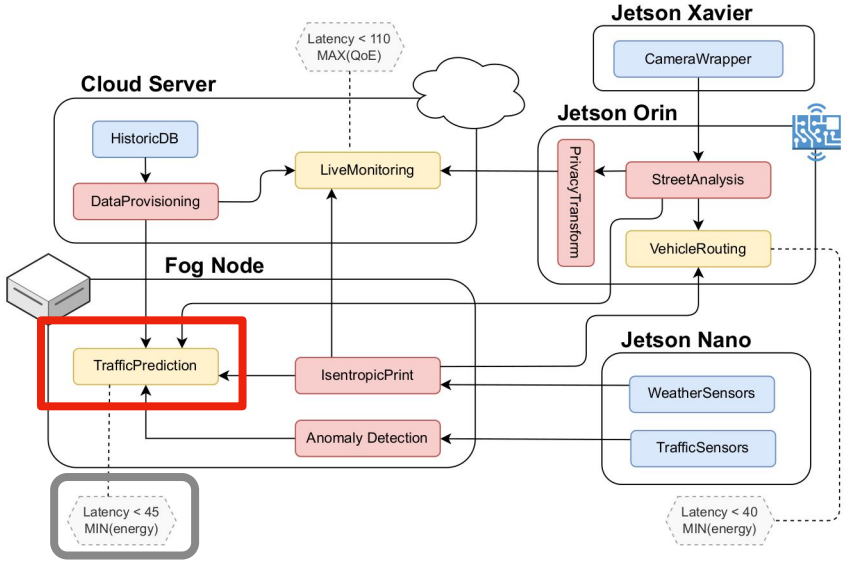


Problem Statement

- Microservice pipelines **distributed** over computing infrastructure, i.e., Edge to Cloud, unclear implications of individual services to high-level requirements, i.e., **SLOs**
- Multiple tenant and vendors (= stakeholders) specify **opposing** SLOs, e.g., minimize energy and response time, which results in **contradicting** service configurations
- Stakeholders not aware what their SLO implies for lower level components and services, e.g., **energy** → cpu

---

- Requires multiple layers of SLOs that specify application requirements through **fine-grained** control mechanism, i.e., diffusing high-level SLOs into lower level SLOs

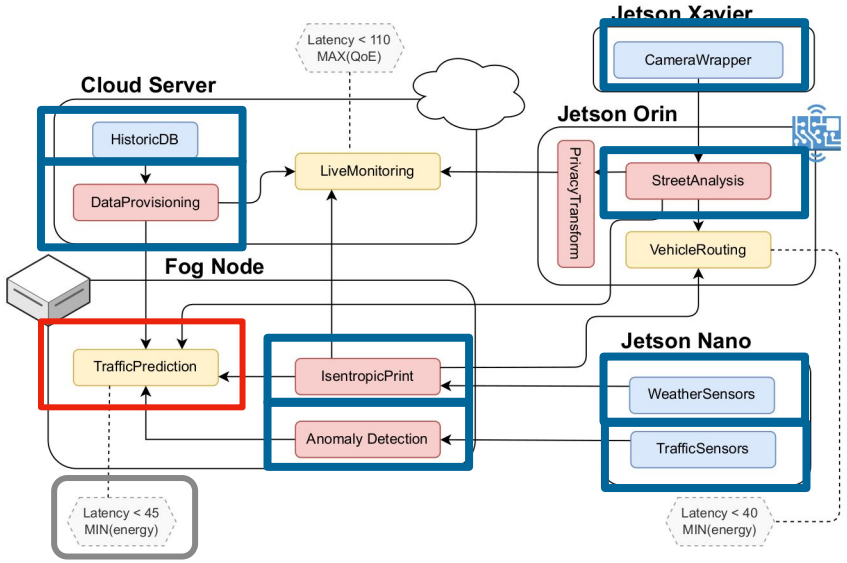


Problem Statement

- Microservice pipelines **distributed** over computing infrastructure, i.e., Edge to Cloud, unclear implications of individual services to high-level requirements, i.e., **SLOs**
- Multiple tenant and vendors (= stakeholders) specify **opposing** SLOs, e.g., minimize energy and response time, which results in **contradicting** service configurations
- Stakeholders not aware what their SLO implies for lower level components and services, e.g., **energy** → cpu

---

- Requires multiple layers of SLOs that specify application requirements through **fine-grained** control mechanism, i.e., diffusing high-level SLOs into lower level SLOs

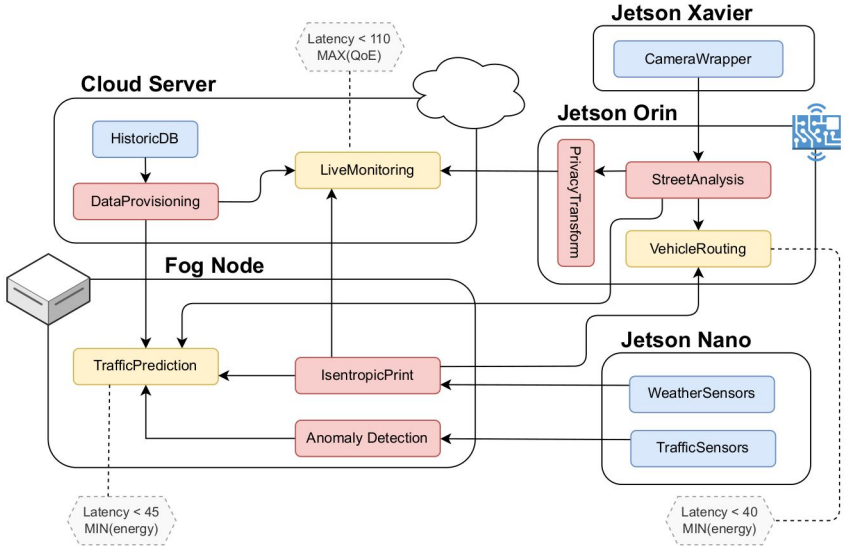


Problem Statement

- Microservice pipelines **distributed** over computing infrastructure, i.e., Edge to Cloud, unclear implications of individual services to high-level requirements, i.e., **SLOs**
- Multiple tenant and vendors (= stakeholders) specify **opposing** SLOs, e.g., minimize energy and response time, which results in **contradicting** service configurations
- Stakeholders not aware what their SLO implies for lower level components and services, e.g., **energy** → cpu

---

- Requires multiple layers of SLOs that specify application requirements through **fine-grained** control mechanism, i.e., diffusing high-level SLOs into lower level SLOs

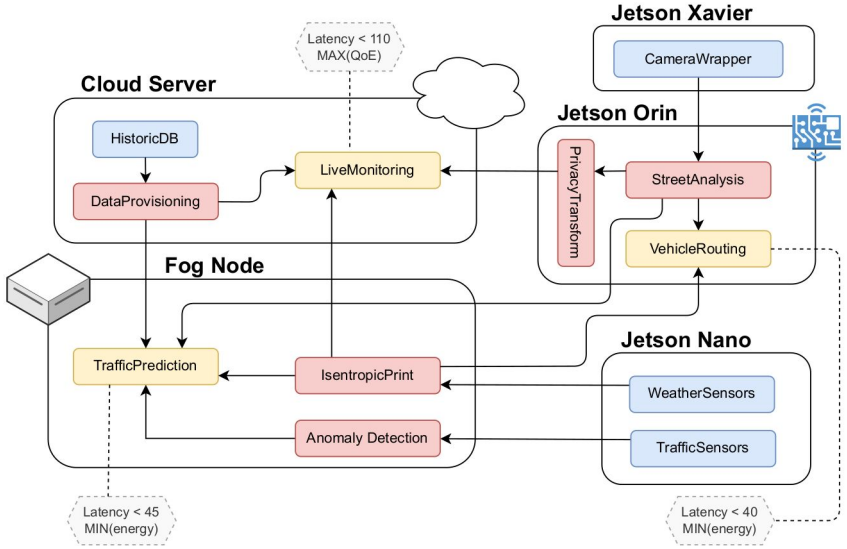


Problem Statement

- Microservice pipelines **distributed** over computing infrastructure, i.e., Edge to Cloud, unclear implications of individual services to high-level requirements, i.e., **SLOs**
- Multiple tenant and vendors (= stakeholders) specify **opposing** SLOs, e.g., minimize energy and response time, which results in **contradicting** service configurations
- Stakeholders not aware what their SLO implies for lower level components and services, e.g., **energy** → cpu

---

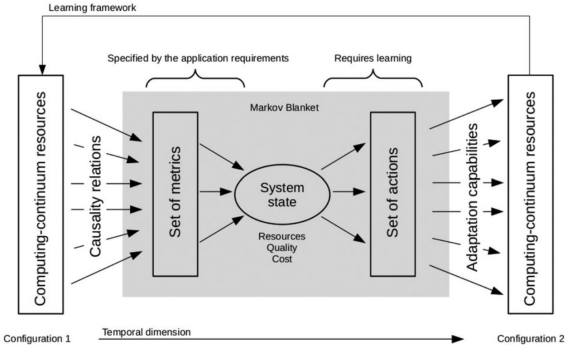
- Requires multiple layers of SLOs that specify application requirements through **fine-grained** control mechanism, i.e., diffusing high-level SLOs into lower level SLOs



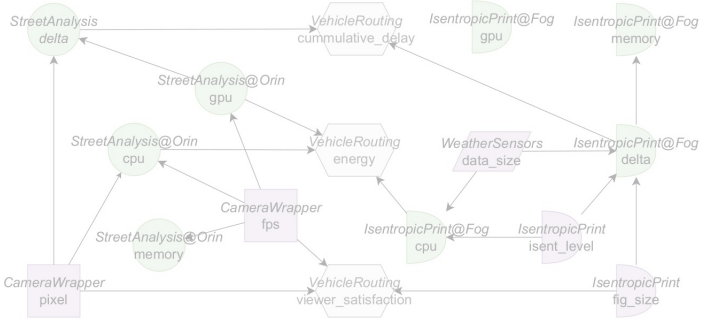
Behavioral Markov Blanket

MBs express how systems interact with their environment through **perception** (e.g., metrics) and **actions** (e.g., scaling)

Model the **behavior** of a system in changing environments; focuses on variables that have an impact on taken actions



Behavioral Markov blanket for a system [1]



Causal dependencies between hierarchical microservices

[1] Dustdar, Casamayor Pujol, and Donta; On Distributed Computing Continuum Systems (2023)

Behavioral Markov Blanket

MBs express how systems interact with their environment through **perception** (e.g., metrics) and **actions** (e.g., scaling)

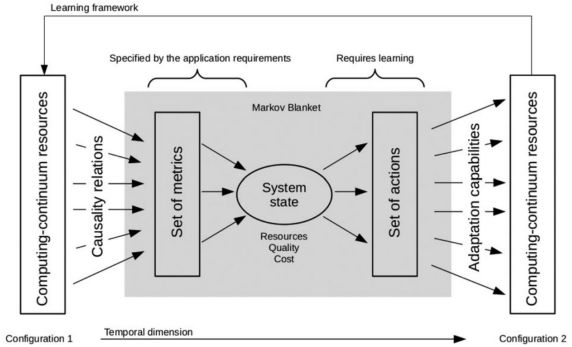
Model the **behavior** of a system in changing environments; focuses on variables that have an impact on taken actions

---

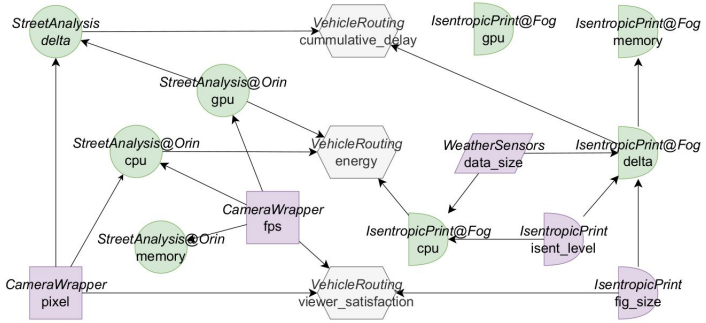
Note two fundamental properties of Bayesian Network (BN)

(1) Variables reflecting high-level SLO are **leaf node**; otherwise constraining childs; always has leaves

(2) Parameter variables always located at **root nodes**; conditional independent of all other nodes



Behavioral Markov blanket for a system [1]



Causal dependencies between hierarchical microservices

[1] Dustdar, Casamayor Pujol, and Donta; On Distributed Computing Continuum Systems (2023)

Behavioral Markov Blanket

MBs express how systems interact with their environment through **perception** (e.g., metrics) and **actions** (e.g., scaling)

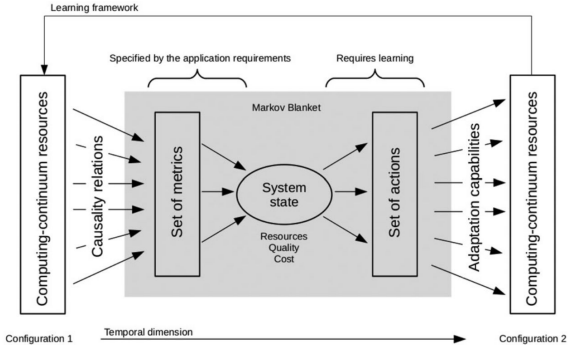
Model the **behavior** of a system in changing environments; focuses on variables that have an impact on taken actions

---

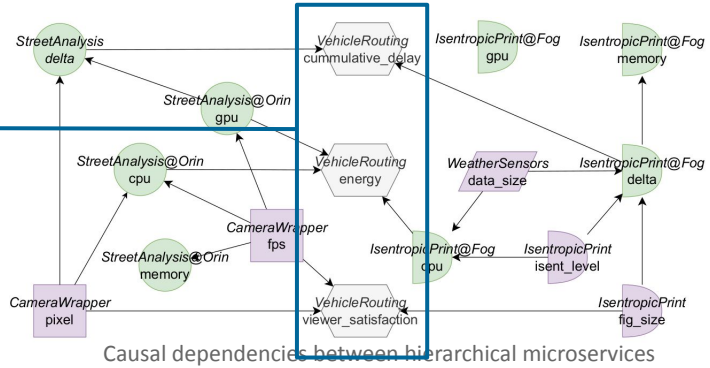
Note two fundamental properties of Bayesian Network (BN)

(1) Variables reflecting high-level SLO are **leaf node**; otherwise constraining childs; always has leaves

(2) Parameter variables always located at **root nodes**; conditional independent of all other nodes



Behavioral Markov blanket for a system [1]



Causal dependencies between hierarchical microservices

[1] Dustdar, Casamayor Pujol, and Donta; On Distributed Computing Continuum Systems (2023)

Behavioral Markov Blanket

MBs express how systems interact with their environment through **perception** (e.g., metrics) and **actions** (e.g., scaling)

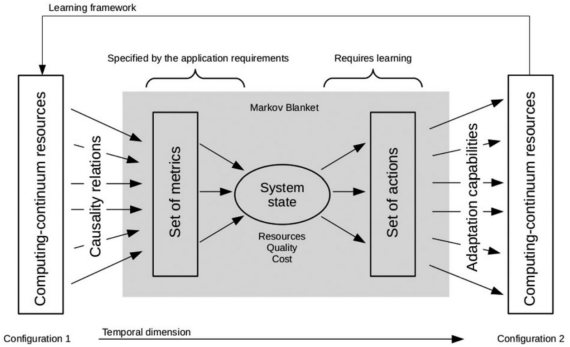
Model the **behavior** of a system in changing environments; focuses on variables that have an impact on taken actions

---

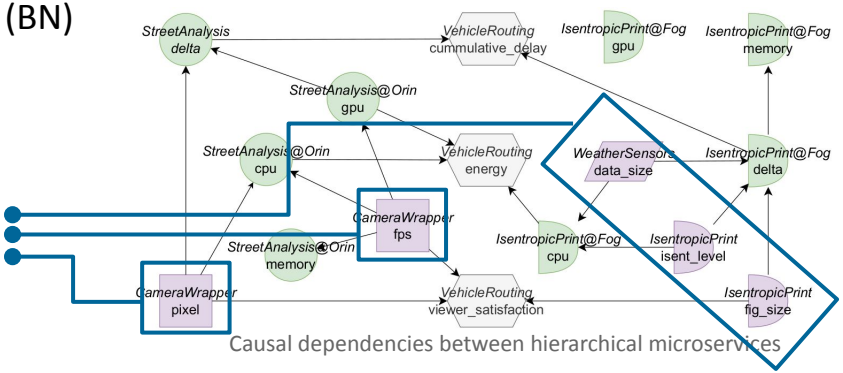
Note two fundamental properties of Bayesian Network (BN)

(1) Variables reflecting high-level SLO are **leaf node**; otherwise constraining childs; always has leaves

(2) Parameter variables always located at **root nodes**; conditional independent of all other nodes



Behavioral Markov blanket for a system [1]



Causal dependencies between hierarchical microservices

[1] Dustdar, Casamayor Pujol, and Donta; On Distributed Computing Continuum Systems (2023)

Methodology – Research Questions

RQ-1) *How can high-level SLOs be translated to lower-level objectives?*

Fulfilling high-level SLOs requires equilibrium among **all** components, hence they which require **clear configurations** to achieve this

RQ-2) *How restrictive should low-level SLOs be?*

Predicting SLO behavior is not black-white, but **continuous**. What are desirable states for lower-level SLO and how to achieve them

RQ-3) *Where do SLO conflicts occur and how can they be resolved?*

Stakeholders cannot maintain a **reasonable** overview over multiple competing SLOs, e.g., energy vs. performance, how to resolve this

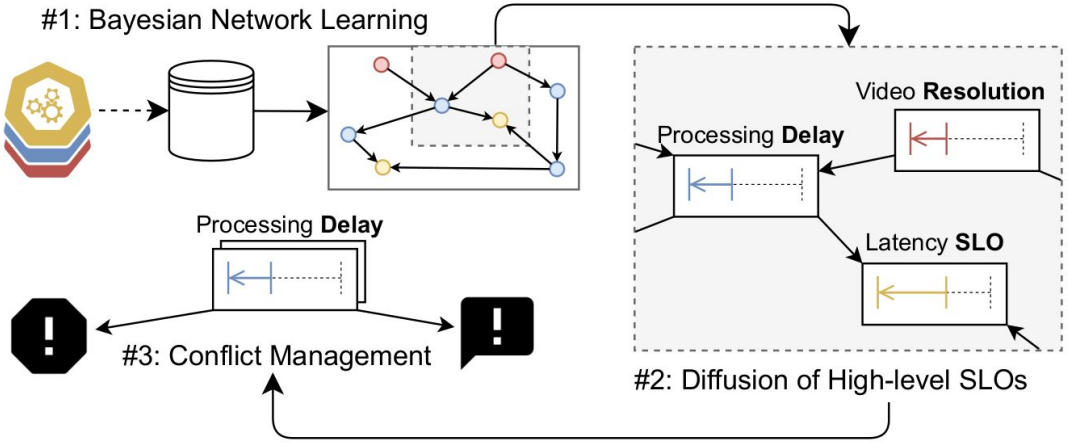
Methodology – Overview

3-Step approach

#1 Extract BN as a probabilistic view into the service execution

#2 diffuse high-level SLOs into lower level ones and param. assignments

#3 identify conflicting variables and resolve them as far as possible

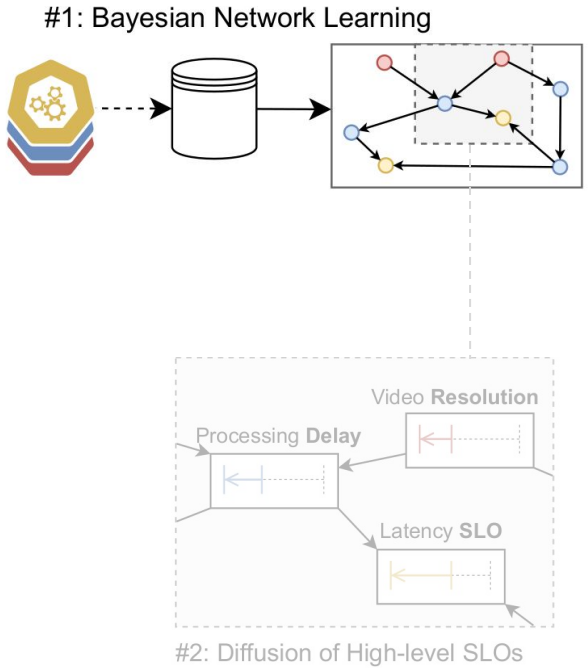


3-Step methodology to diffuse high-level SLOs in a microservice pipeline

#1 Bayesian Network Learning (BNL)

Extract **causal** dependencies between dependent services; training data collected centrally and used for BNL, must be captured in parallel or joined through **interface variables**

Renders a **composite graph** for a services tree starting from the consumer, i.e., restricts the number of variables per case



#2 Diffusion of High-level SLOs

Recursively traverse the **parents** of high-level SLOs and constraint states of lower-level variables and parameters; visiting variables multiple times constrains them further

Evaluates each low-level state's probability to **satisfying** high-level constraints; only considers a state iff $p > max(\boldsymbol{X}) * \boldsymbol{\lambda}$

$\boldsymbol{X}$ = list of all state's probabilities to fulfill SLOs; $\boldsymbol{\lambda}$ = acceptance range

#1 Bayesian Network Learning (BNL)

Extract **causal** dependencies between dependent services; training data collected centrally and used for BNL, must be captured in parallel or joined through **interface variables**

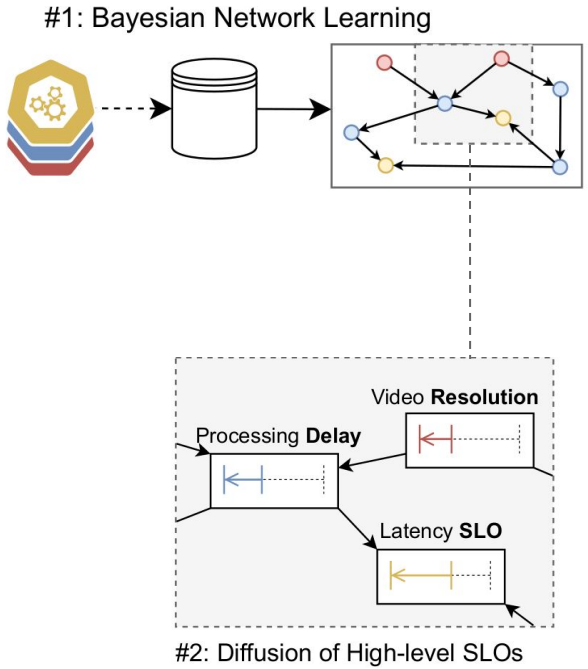
Renders a **composite graph** for a services tree starting from the consumer, i.e., restricts the number of variables per case

#2 Diffusion of High-level SLOs

Recursively traverse the **parents** of high-level SLOs and constraint states of lower-level variables and parameters; visiting variables multiple times constrains them further

Evaluates each low-level state's probability to **satisfying** high-level constraints; only considers a state iff $p > max(\boldsymbol{X}) * \boldsymbol{\lambda}$

$\boldsymbol{X}$ = list of all state's probabilities to fulfill SLOs; $\boldsymbol{\lambda}$ = acceptance range



#1 Bayesian Network Learning (BNL)

Extract **causal** dependencies between dependent services; training data collected centrally and used for BNL, must be captured in parallel or joined through **interface variables**

Renders a **composite graph** for a services tree starting from the consumer, i.e., restricts the number of variables per case

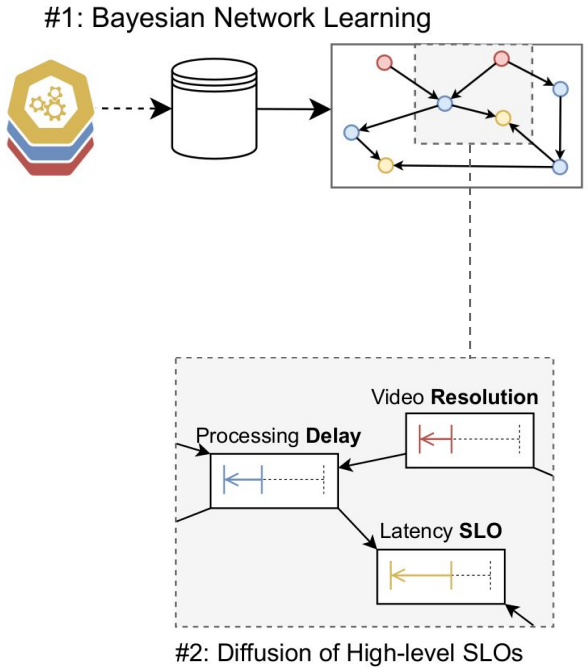
#2 Diffusion of High-level SLOs

Recursively traverse the **parents** of high-level SLOs and constraint s…bles and parameters; visiting vari…trains them further

Complexity grows with the number of states

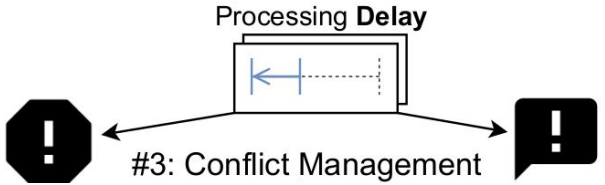
Evaluates e…ability to **satisfying** high-level constraints, only considers a state iff $p > max(\boldsymbol{X}) * \boldsymbol{\lambda}$

$\boldsymbol{X}$ = list of all state's probabilities to fulfill SLOs; $\boldsymbol{\lambda}$ = acceptance range



#3 Conflict Management

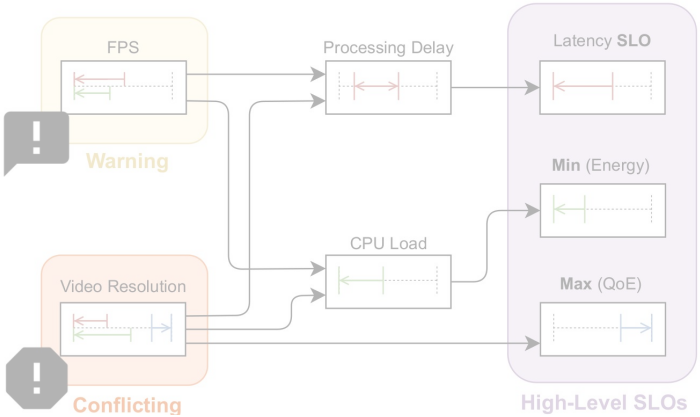
Identify **conflicting variables** and **resolve** them as far as possible; merge the inferred low-level SLOs and try to find an **intersection** among the assignments



$$\text{INTER}(L_v) = \bigcap_{i,j=1;i\neq j}^{n} L_i \cap L_j \neq \varnothing$$

Warning
issued in case that variables were visited and constrained multiple times, but it was possible to merge inferred SLOs

Conflict
highlight to stakeholders which variables are conflicting and which high-level SLO are responsible, i.e., **Max** (QoE)



Detecting conflicting variables when diffusing high-level SLOs

#3 Conflict Management

Identify **conflicting variables** and **resolve** them as far as possible; merge the inferred low-level SLOs and try to find an **intersection** among the assignments

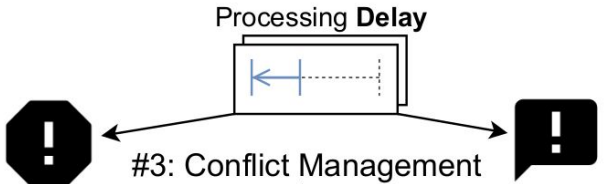
$$\text{INTER}(L_v) = \bigcap_{i,j=1;i\neq j}^{n} L_i \cap L_j \neq \varnothing$$

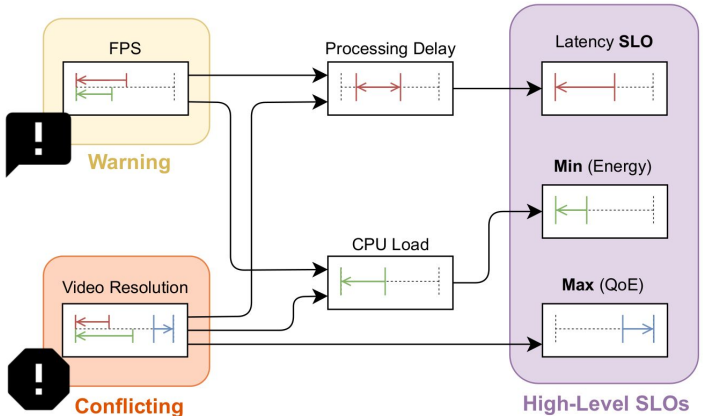
Warning

issued in case that variables were visited and constrained multiple times, but it was possible to merge inferred SLOs

Conflict

highlight to stakeholders which variables are conflicting and which high-level SLO are responsible, i.e., **Max** (QoE)





Detecting conflicting variables when diffusing high-level SLOs

Experimental Setup

Combination of 12 microservices **plugged together**; service implementations all collected in GitHub repository

Microservice pipelines consist of 4 producers, 5 workers, and 3 consumer services; deployed on **different hosts** and feature distinct numbers of **configuration parameters**

Services executed on the physical setup, provide all the metrics required for the 3-step methodology

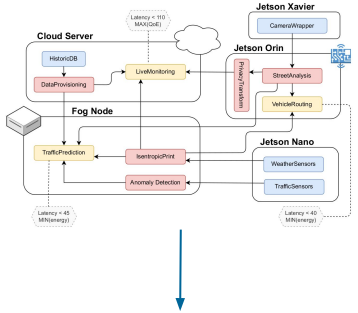
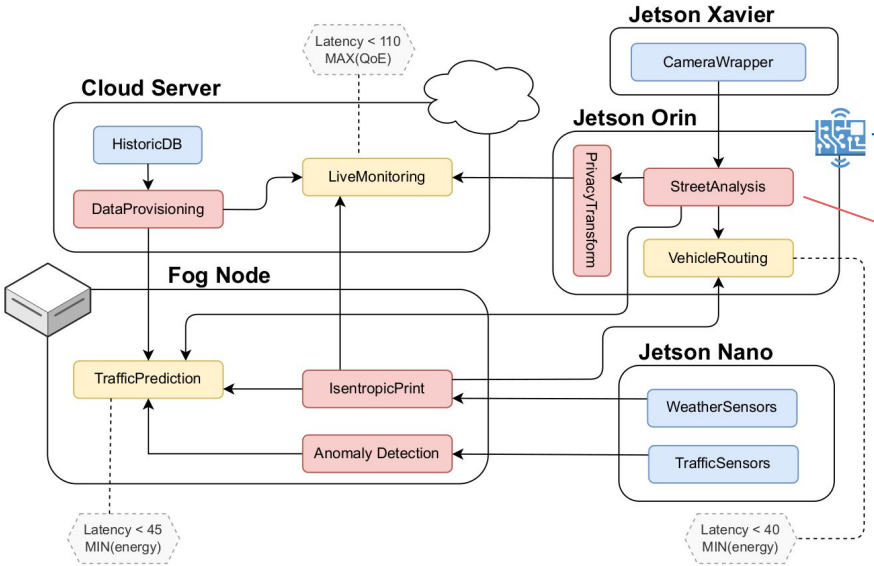


TABLE II: Microsevices available for evaluation

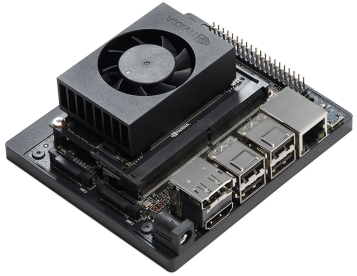
	ID	type	param / var	host
TrafficSensors	[28]	Producer	1 / 1	*Xavier*
HistoricDB		Producer	2 / 2	*Server*
CameraWrapper	[29]	Producer	2 / 2	*Nano*
WeatherSensors	[30]	Producer	1 / 1	*Xavier*
AnomalyDetection	[28]	Worker	0 / 5	*Fog*
HistoricProvision		Worker	2 / 7	*Server*
StreetAnalysis	[31]	Worker	0 / 4	*Orin*
PrivacyTransform	[29]	Worker	0 / 6	*Orin*
IsentropicPrint	[30]	Worker	2 / 6	*Fog*
TrafficPrediction		Consumer	0 / 2	*Fog*
VehicleRouting		Consumer	0 / 3	*Orin*
LiveMonitoring		Consumer	0 / 3	*Server*

https://github.com/borissedlak/deploymentOptimizer/tree/main/SOSE

Experimental Setup (2)



Network of microservice pipelines deployed over the computing continuum

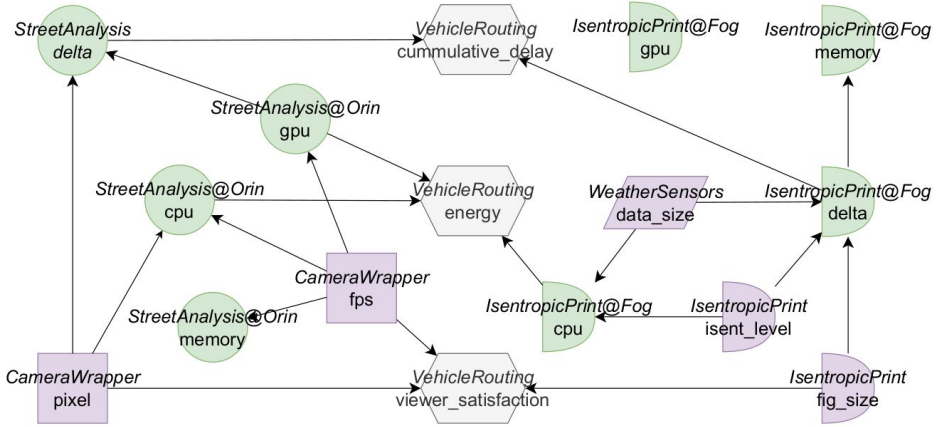


Nvidia Jetson **Orin** board (image from here)

CV Service with Yolov8 running on the produced videos

Results: RQ-1 SLO Diffusion

Given a set of **high-level SLOs** and a **overarching BN**
Diffuse the SLOs to **lower-level variables** and **parameters**



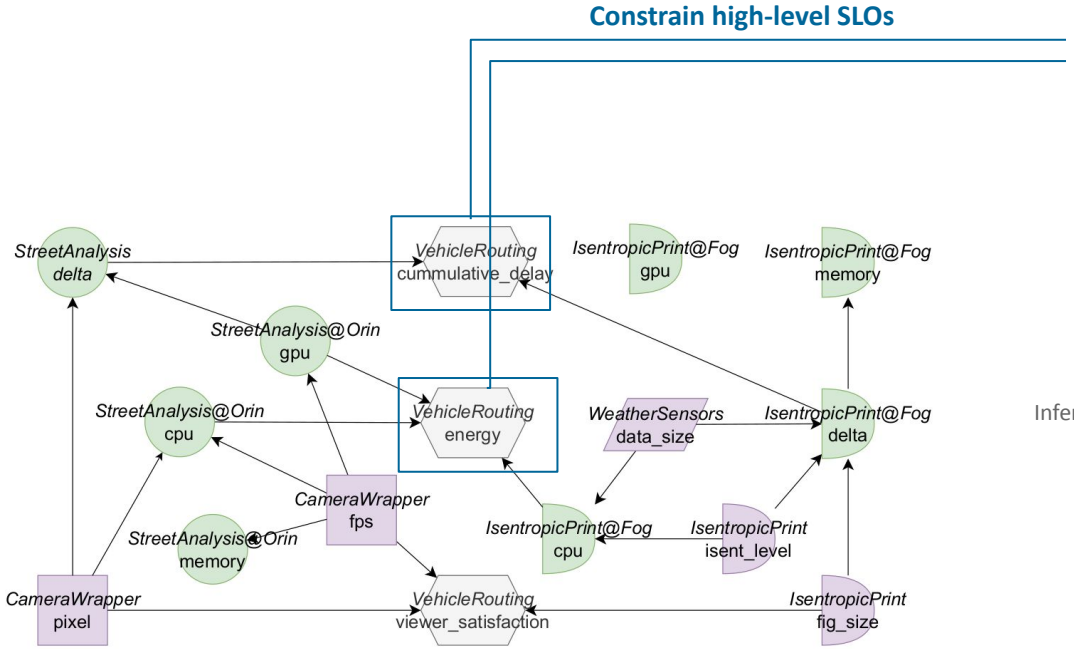
Composite BN for all microservices connected to the *VehicleRouting* consumer

TABLE III: SLOs and parameters inferred for *VehicleRouting*

Microservice	Variable	States	SLO / Param
VehicleRouting	cumm_delay	$\leqslant$ 45 ms	High-level
	energy	$\leqslant$ 19 W	
	viewer_sat	———	
StreetAnalysis	delta	$\leqslant$ 35 ms	Low-level
StreetAnalysis	cpu (Orin)	$\leqslant$ 21 %	
StreetAnalysis	gpu (Orin)	$\leqslant$ 40 %	
IsentropicPrint	delta	$\leqslant$ 37 ms	
IsentropicPrint	cpu (Fog)	$\leqslant$ 17 %	
CameraWrapper	pixel	= 480 p	Parameter
CameraWrapper	fps	= 15 f	
IsentropicPrint	fig_size	$\leqslant$ 50 p	
IsentropicPrint	isent_level	$\leqslant$ 200 k	
WeatherSensors	data_size	$\leqslant$ 30 pi	

Inferred low-level SLOs and parameter assignments for high-level SLOs

Results: RQ-1 SLO Diffusion



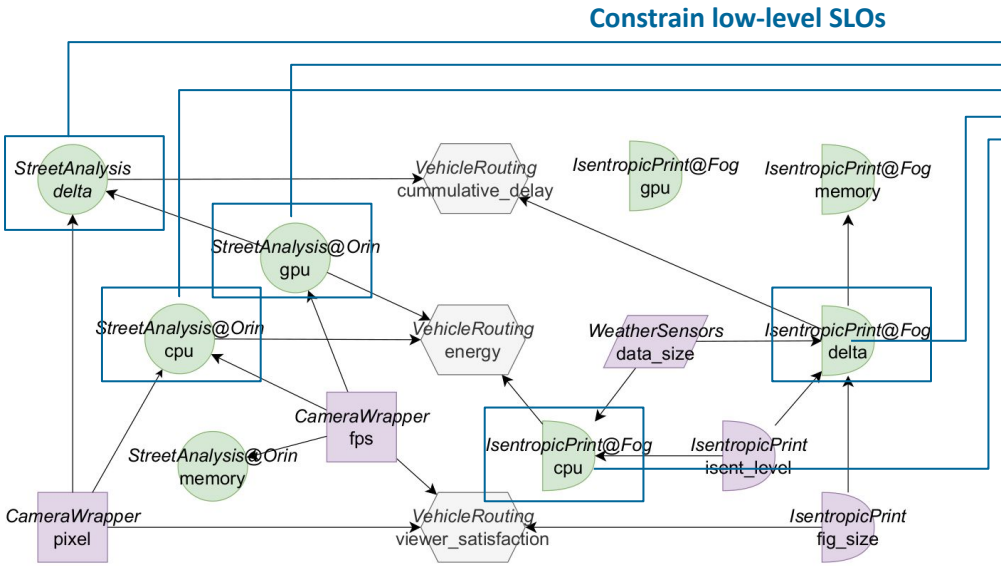
Composite BN for all microservices connected to the *VehicleRouting* consumer

TABLE III: SLOs and parameters inferred for *VehicleRouting*

Microservice	Variable	States	SLO / Param
VehicleRouting	cumm_delay	$\leqslant$ 45 ms	High-level
	energy	$\leqslant$ 19 W	
	viewer_sat	———	
StreetAnalysis	delta	$\leqslant$ 35 ms	Low-level
StreetAnalysis	cpu (Orin)	$\leqslant$ 21 %	
StreetAnalysis	gpu (Orin)	$\leqslant$ 40 %	
IsentropicPrint	delta	$\leqslant$ 37 ms	
IsentropicPrint	cpu (Fog)	$\leqslant$ 17 %	
CameraWrapper	pixel	= 480 p	Parameter
CameraWrapper	fps	= 15 f	
IsentropicPrint	fig_size	$\leqslant$ 50 p	
IsentropicPrint	isent_level	$\leqslant$ 200 k	
WeatherSensors	data_size	$\leqslant$ 30 pi	

Inferred low-level SLOs and parameter assignments for high-level SLOs

Results: RQ-1 SLO Diffusion



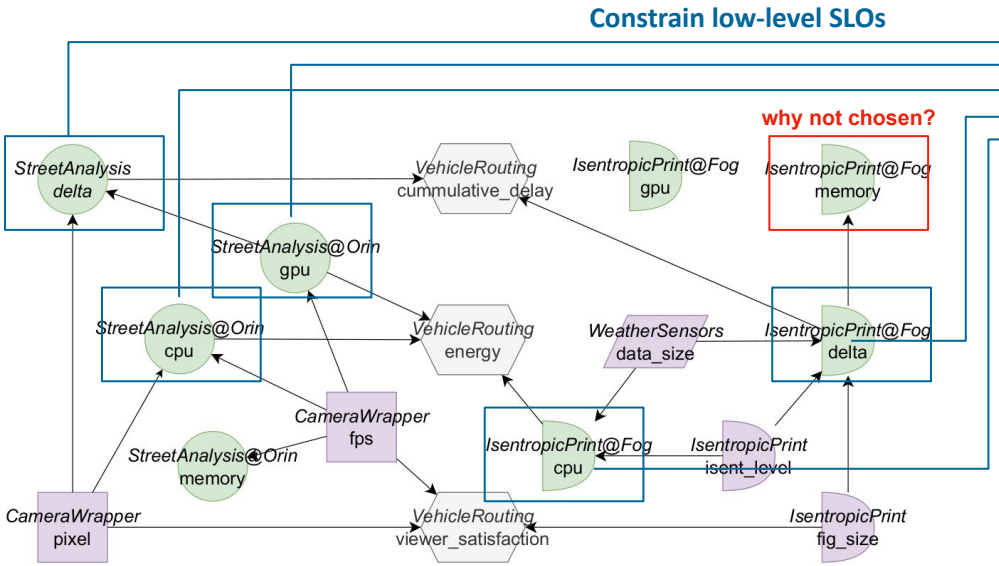
Composite BN for all microservices connected to the *VehicleRouting* consumer

TABLE III: SLOs and parameters inferred for *VehicleRouting*

Microservice	Variable	States	SLO / Param
VehicleRouting	cumm_delay	$\leqslant$ 45 ms	High-level
	energy	$\leqslant$ 19 W	
	viewer_sat	————	
StreetAnalysis	delta	$\leqslant$ 35 ms	Low-level
StreetAnalysis	cpu (Orin)	$\leqslant$ 21 %	
StreetAnalysis	gpu (Orin)	$\leqslant$ 40 %	
IsentropicPrint	delta	$\leqslant$ 37 ms	
IsentropicPrint	cpu (Fog)	$\leqslant$ 17 %	
CameraWrapper	pixel	= 480 p	Parameter
CameraWrapper	fps	= 15 f	
IsentropicPrint	fig_size	$\leqslant$ 50 p	
IsentropicPrint	isent_level	$\leqslant$ 200 k	
WeatherSensors	data_size	$\leqslant$ 30 pi	

Inferred low-level SLOs and parameter assignments for high-level SLOs

Results: RQ-1 SLO Diffusion



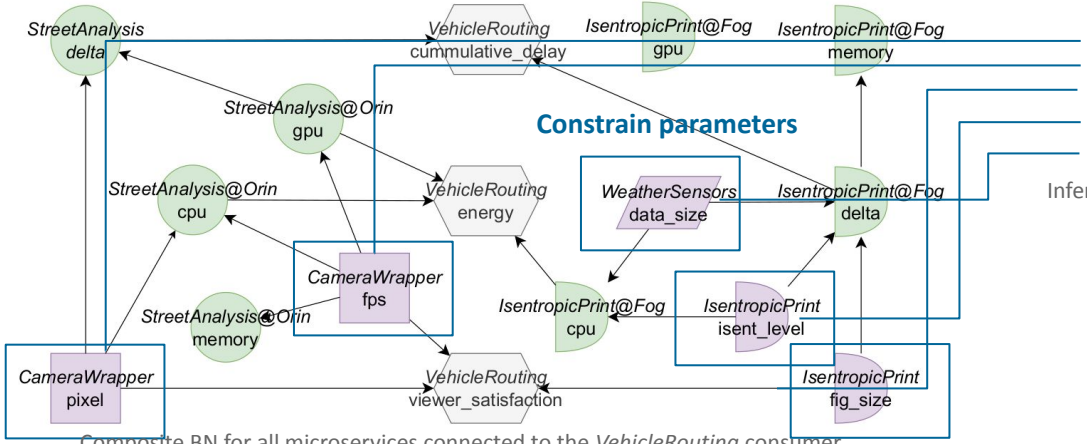
Composite BN for all microservices connected to the *VehicleRouting* consumer

TABLE III: SLOs and parameters inferred for *VehicleRouting*

Microservice	Variable	States	SLO / Param
VehicleRouting	cumm_delay	$\leqslant$ 45 ms	High-level
	energy	$\leqslant$ 19 W	
	viewer_sat	————	
StreetAnalysis	delta	$\leqslant$ 35 ms	Low-level
StreetAnalysis	cpu (Orin)	$\leqslant$ 21 %	
StreetAnalysis	gpu (Orin)	$\leqslant$ 40 %	
IsentropicPrint	delta	$\leqslant$ 37 ms	
IsentropicPrint	cpu (Fog)	$\leqslant$ 17 %	
CameraWrapper	pixel	= 480 p	Parameter
CameraWrapper	fps	= 15 f	
IsentropicPrint	fig_size	$\leqslant$ 50 p	
IsentropicPrint	isent_level	$\leqslant$ 200 k	
WeatherSensors	data_size	$\leqslant$ 30 pi	

Inferred low-level SLOs and parameter assignments for high-level SLOs

Results: RQ-1 SLO Diffusion



Composite BN for all microservices connected to the *VehicleRouting* consumer

TABLE III: SLOs and parameters inferred for *VehicleRouting*

Microservice	Variable	States	SLO / Param
VehicleRouting	cumm_delay	$\leqslant$ 45 ms	High-level
	energy	$\leqslant$ 19 W	
	viewer_sat	————	
StreetAnalysis	delta	$\leqslant$ 35 ms	Low-level
StreetAnalysis	cpu (Orin)	$\leqslant$ 21 %	
StreetAnalysis	gpu (Orin)	$\leqslant$ 40 %	
IsentropicPrint	delta	$\leqslant$ 37 ms	
IsentropicPrint	cpu (Fog)	$\leqslant$ 17 %	
CameraWrapper	pixel	= 480 p	Parameter
CameraWrapper	fps	= 15 f	
IsentropicPrint	fig_size	$\leqslant$ 50 p	
IsentropicPrint	isent_level	$\leqslant$ 200 k	
WeatherSensors	data_size	$\leqslant$ 30 pi	

Inferred low-level SLOs and parameter assignments for high-level SLOs

Results: RQ-1 SLO Diffusion (2)

Setup the evaluation environment

1) **Deploy** microservices over the architecture
2) Set **parameter** according inferred thresholds
3) Measure the **actual** SLO fulfillment
4) Evaluate **alternative** param. configurations
5) **Compare** results and find min / max

What can we report?

- ❏ Inferred configuration close to optimal
- ❏ Discrepancy occur either due to **flexible boundaries**, or conflicts between SLOs

TABLE III: SLOs and parameters inferred for *VehicleRouting*

Microservice	Variable	States	SLO / Param
VehicleRouting	cumm_delay	$\leqslant$ 45 ms	High-level
	energy	$\leqslant$ 19 W	
	viewer_sat	———	
StreetAnalysis	delta	$\leqslant$ 35 ms	Low-level
StreetAnalysis	cpu (Orin)	$\leqslant$ 21 %	
StreetAnalysis	gpu (Orin)	$\leqslant$ 40 %	
IsentropicPrint	delta	$\leqslant$ 37 ms	
IsentropicPrint	cpu (Fog)	$\leqslant$ 17 %	
CameraWrapper	pixel	= 480 p	Parameter
CameraWrapper	fps	= 15 f	
IsentropicPrint	fig_size	$\leqslant$ 50 p	
IsentropicPrint	isent_level	$\leqslant$ 200 k	
WeatherSensors	data_size	$\leqslant$ 30 pi	

TABLE IV: High-level SLO fulfillment of inferred and alternative assignment for all three evaluated applications

Microservice	High-level SLO	% Min	% Fulfill	% Max
VehicleRouting	cumm_delay $\leqslant$ 45	0.00	0.94	1.00
	min(energy)	0.53	0.99	1.00
TrafficPrediction	cumm_delay $\leqslant$ 40	0.00	0.83	0.90
LiveMonitoring	cumm_delay $\leqslant$ 110	0.13	0.93	1.00
	max(viewer_sat.)	0.00	1.00	1.00

Results: RQ-1 SLO Diffusion (2)

Setup the evaluation environment

1) **Deploy** microservices over the architecture
2) Set **parameter** according inferred thresholds
3) Measure the **actual** SLO fulfillment
4) Evaluate **alternative** param. configurations
5) **Compare** results and find min / max

What can we report?

- ❏ Inferred configuration close to optimal
- ❏ Discrepancy occur either due to **flexible boundaries**, or conflicts between SLOs

TABLE III: SLOs and parameters inferred for *VehicleRouting*

Microservice	Variable	States	SLO / Param
VehicleRouting	cumm_delay	$\leqslant$ 45 ms	High-level
	energy	$\leqslant$ 19 W	
	viewer_sat	———	
StreetAnalysis	delta	$\leqslant$ 35 ms	Low-level
StreetAnalysis	cpu (Orin)	$\leqslant$ 21 %	
StreetAnalysis	gpu (Orin)	$\leqslant$ 40 %	
IsentropicPrint	delta	$\leqslant$ 37 ms	
IsentropicPrint	cpu (Fog)	$\leqslant$ 17 %	
CameraWrapper	pixel	= 480 p	Parameter
CameraWrapper	fps	= 15 f	
IsentropicPrint	fig_size	$\leqslant$ 50 p	
IsentropicPrint	isent_level	$\leqslant$ 200 k	
WeatherSensors	data_size	$\leqslant$ 30 pi	

TABLE IV: High-level SLO fulfillment of inferred and alternative assignment for all three evaluated applications

Microservice	High-level SLO	% Min	% Fulfill	% Max
VehicleRouting	cumm_delay $\leqslant$ 45	0.00	0.94	1.00
	min(energy)	0.53	0.99	1.00
TrafficPrediction	cumm_delay $\leqslant$ 40	0.00	0.83	0.90
LiveMonitoring	cumm_delay $\leqslant$ 110	0.13	0.93	1.00
	max(viewer_sat.)	0.00	1.00	1.00

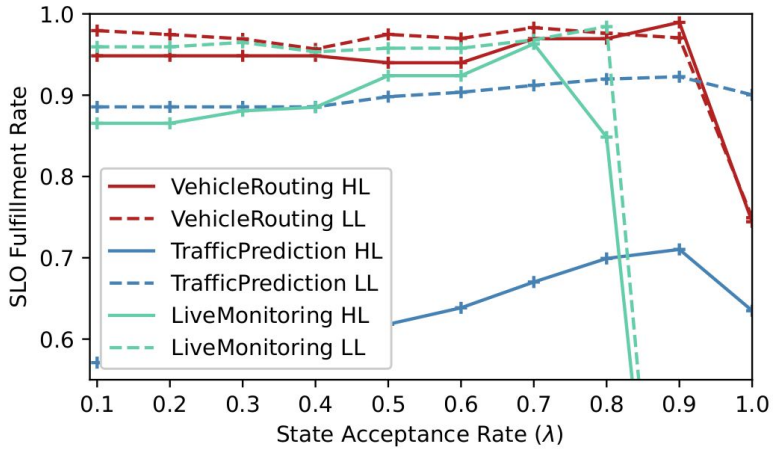
Results: RQ-2 Acceptance Range

When **constraining** lower-level states of variables, how **restrictive** should lower-level boundaries be?

Vary acceptance rate (***λ***) from [0.1,1.0]; means very **loose** or **restrictive** for lower-level SLOs and params; for 3 consumers evaluate SLOs for microservices

---

Lower acceptance rate **improves** SLO fulfillment; when acceptance rate **too narrow**, not possible anymore to find satisfying parameter assignments



Results: RQ-3 Conflicts and Resolution

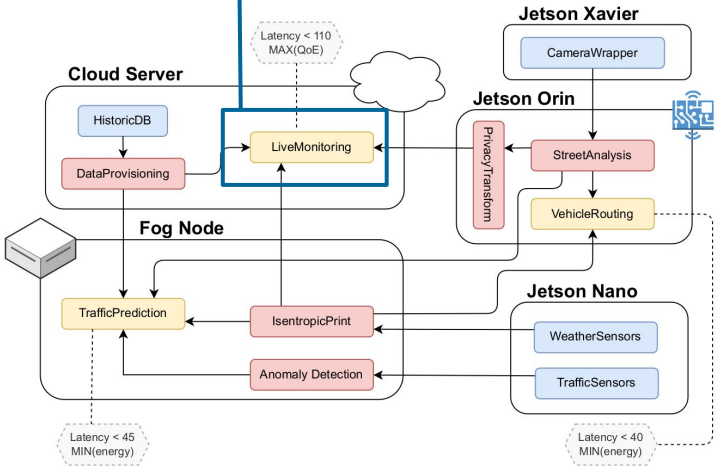
Conflicts between high-level SLOs appear at the lower-level SLOs and parameter assignments

Choose microservices around *LiveMonitoring* and specify increasingly tight performance boundary, extended with SLOs for opposing targets, i.e., **min**(energy) and/or **max**(satisfact.)

Wide boundaries (first row) allow to infer parameter assignments for most cases, but tighter constraints results in most variables finding no intersection

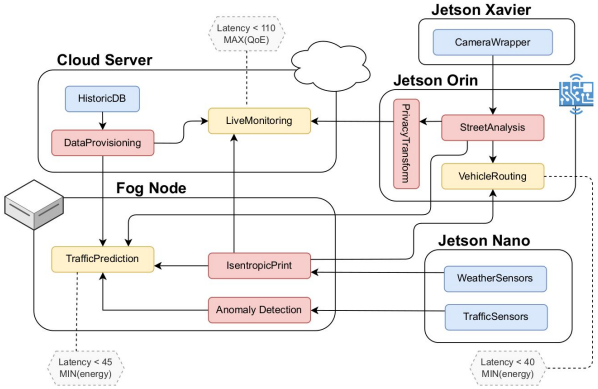
TABLE V: Conflicts among high-level SLOs for *LiveMonitor*

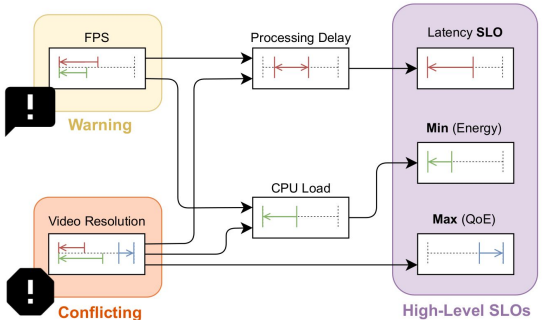
cumm_delay	min(*energy*)	max(*customer_sat*)	both
$\leqslant$ 120 ms	✓	✓	$\lightning\{fps\}$
$\leqslant$ 100 ms	$\lightning\{pixel\}$	✓	$\lightning\{\wedge \cup pixel\}$
$\leqslant$ 50 ms	$\lightning\{\wedge \cup fps\}$	$\lightning\{gpu,\ pixel,\ fps\}$	$\lightning\{\wedge \cup gpu\}$
$\leqslant$ 40 ms	$\lightning\{\wedge \cup batch\}$	$\lightning\{\wedge\}$	$\lightning\{\wedge \cup fig\_size\}$
$\leqslant$ 25 ms	$\lightning\{\wedge\}$	$\lightning\{\wedge \cup cache\_db\}$	$\lightning\{\wedge \cup cache\_db\}$



Summary

- Stakeholders **unaware** of implications of high-level SLOs to lower-level components and parameters and whether specifies SLOs are **conflicting**
- Requires mechanisms to diffuse high-level SLOs to composite microservices pipelines
- Diffusion traverses BN to constrain lower-level variables to states that fulfill higher-level goals
- Evaluation for microservice networks; high-level goals were diffused to service parameters; could highlight conflicting SLOs to stakeholders





Let’s **discuss**!

Please come forward with any **question** you have

@

boris.sedlak@dsg.tuwien.ac.at