Network Engineering Department, Universitat Politècnica de Catalunya

An Energy-friendly Scheduler for Edge Computing System

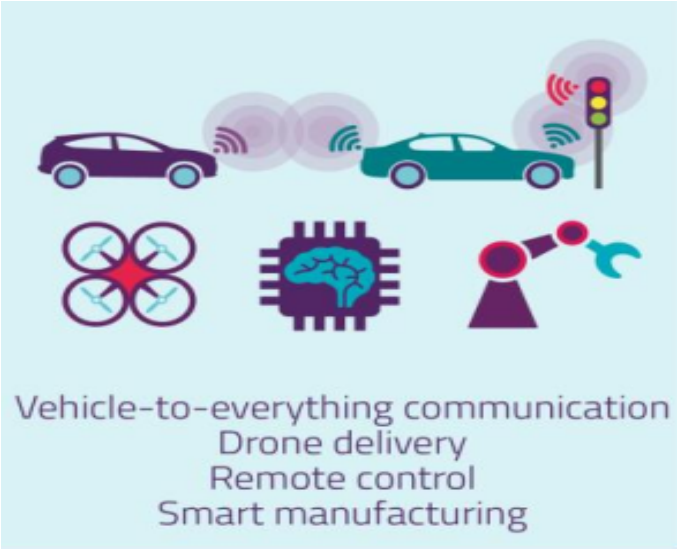
Alejandro Llorens-Carrodeguas
(alejandro.llorens@entel.upc.edu)
Stefanos G. Sagkriotis
Cristina Cervelló-Pastor
Dimitrios P. Pezaros

November 24, 2021

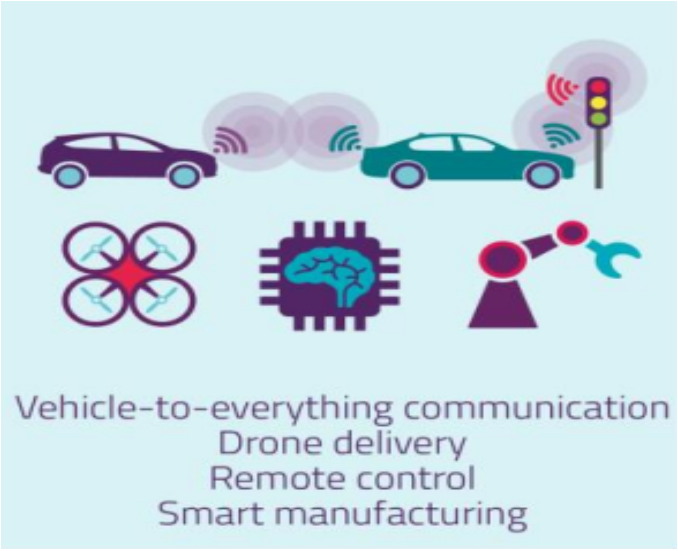
Table of Contents

Table of Contents

Introduction



Introduction



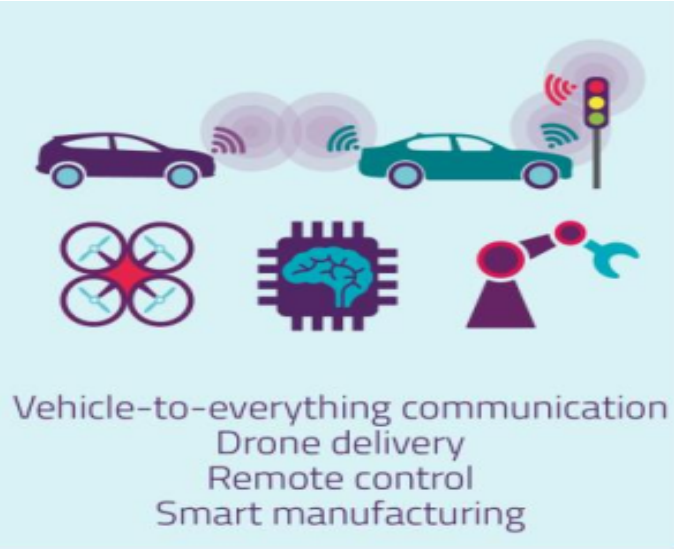
+

Instantaneous response

Robustness

Availability

Introduction



)

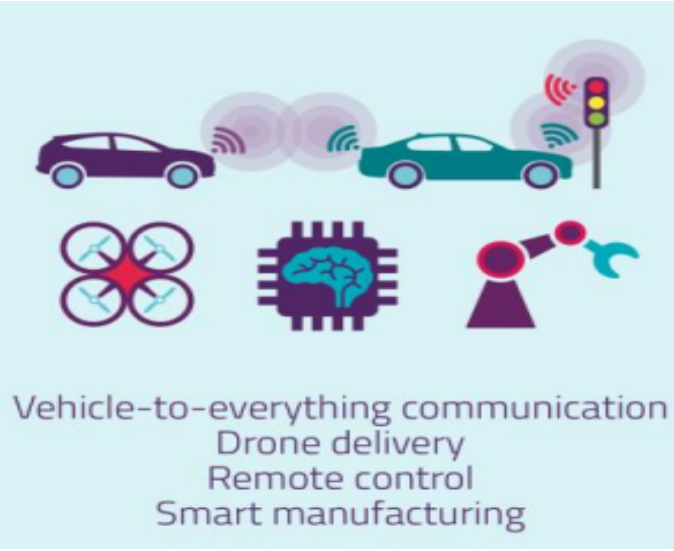
+

Instantaneous response

Robustness

Availability

Introduction



+

Instantaneous response

Robustness

Availability

)

+

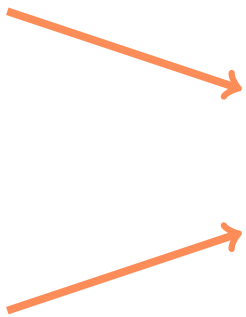
Inter-operative **cluster** compounding that enable failure-recovery and accumulative processing capacity

Network Function Virtualization

Edge and Fog Computing

Network Function Virtualization

Edge and Fog Computing



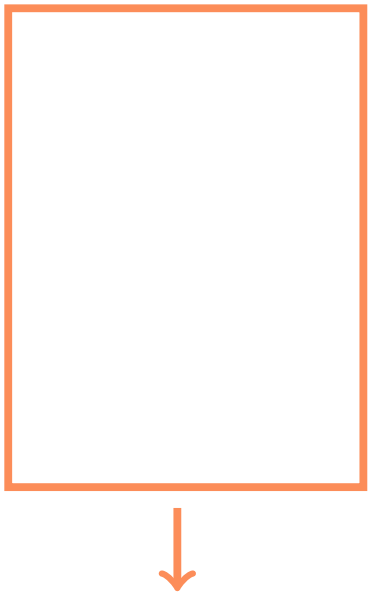
Allow fast deployment of applications

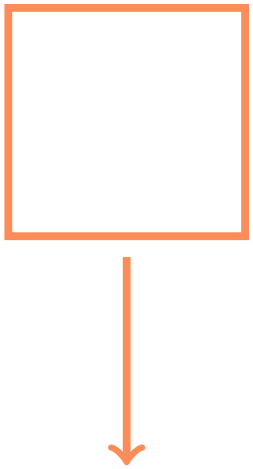
Better resource utilization and scalability

Cost reduction

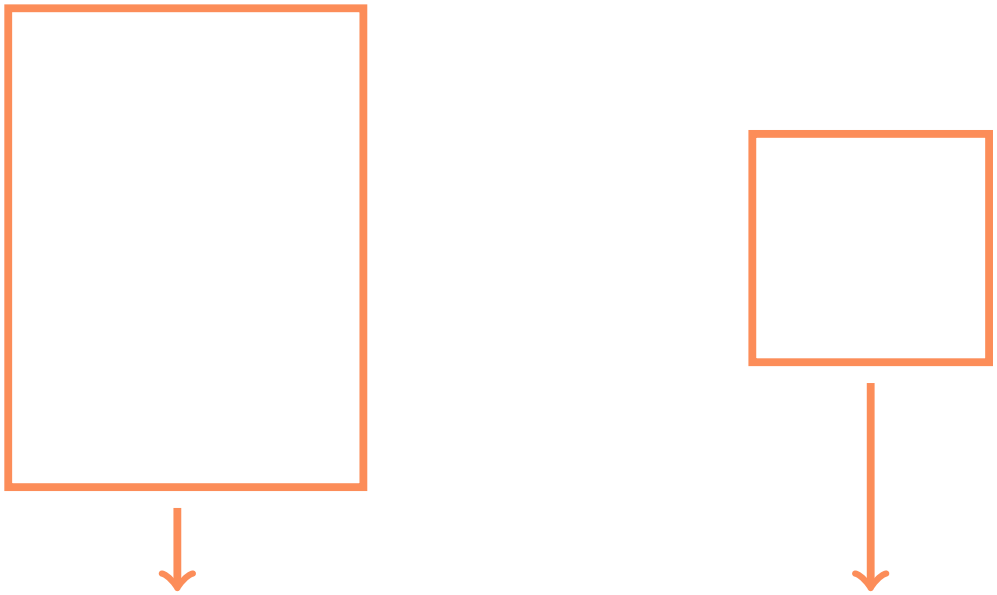
Introduction

Introduction



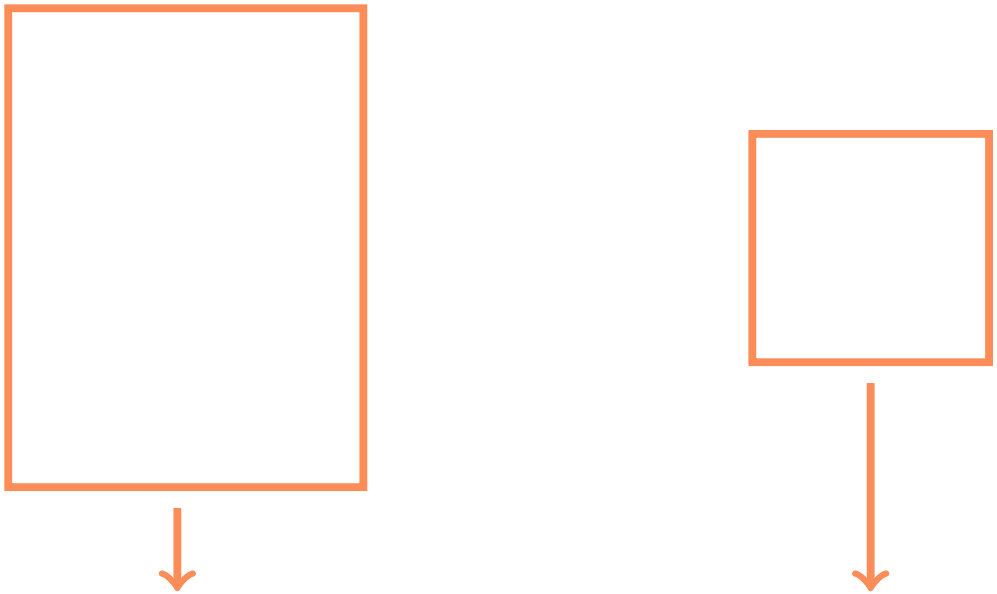


Introduction



Deploying and managing services in a cluster can be a laborious and error-prone task

Introduction



Deploying and managing services in a cluster can be a laborious and error-prone task

An scheduler integrating energy measurements in the placement decisions, and considering all the cluster's nodes is required

Table of Contents

Problem statement

Problem statement

A controller node is deployed alongside a set of computing nodes (N_c) (cluster of SBCs)

Problem statement

A controller node is deployed alongside a set of computing nodes (N_c) (cluster of SBCs)

Incoming requests are scheduled according to their deadline and resources requirements

Problem statement

A controller node is deployed alongside a set of computing nodes (N_c) (cluster of SBCs)

Incoming requests are scheduled according to their deadline and resources requirements

Controller and computing nodes are energy-constrained devices with xed computational resources

Problem statement

A controller node is deployed alongside a set of computing nodes (N_c) (cluster of SBCs)

Incoming requests are scheduled according to their deadline and resources requirements

Controller and computing nodes are energy-constrained devices with xed computational resources

+

The scheduler place events by considering remaining battery of nodes to increase the lifetime and resilience of the cluster

Problem statement

A controller node is deployed alongside a set of computing nodes (N_c) (cluster of SBCs)

Incoming requests are scheduled according to their deadline and resources requirements

Controller and computing nodes are energy-constrained devices with xed computational resources

+

The scheduler place events by considering remaining battery of nodes to increase the lifetime and resilience of the cluster

The controller not only must be capable of coordinating and managing the deployed services and tasks, but also monitor the battery status of each node, including its own battery levels

System model

Notation	Description
P	Set of deployable units of computing nodes
N	Set of physical nodes where events can be scheduled
N_c	Set of computing nodes (N_c N)
S	Network service request arriving to controller node
F	Sequence of VNFs compounding a network service request
T	Task request arriving to controller node
p	Each virtual node created on the physical nodes to run the events
n	Each physical node where virtual nodes are created
p_{f_i}	Indicates the virtual node where function f_i is running
p_T	Indicates the virtual node where task T is running
f_i	Each network function forming part of a network service
t_r	Running time of an event before considering it completed
t_s	Starting time of an event when being processed in the selected node
t_c	Completion time of an event in the selected node
t_e	Execution time of an event in the assigned node
t_a	Arrival time of an event request in the controller node
t_t	Total time of an event in the system
t_w	Waiting time of an event in the priority queue
d	Deadline for processing a given event

$$t_e = t_c \quad t_s$$

$$t_t = t_c \quad t_a$$

$$t_w = \begin{array}{ll} t_s \quad t_{a_{f=1}} & \text{if the event is a service} \\ t_s \quad t_{a_T} & \text{if the event is a task} \end{array}$$

Table of Contents

Proposed scheduler
SOC and capacity-based scheduler (SOCCS)

Proposed scheduler
SOC and capacity-based scheduler (SOCCS)

Proposed scheduler
SOC and capacity-based scheduler (SOCCS)

Proposed scheduler
Monitor block

Proposed scheduler
Monitor block

Proposed scheduler
Scheduler block (II)

Proposed scheduler
Scheduler block (II)

Proposed scheduler
Scheduler block (I)

$$n_{score} = {}_{1}(SOC = 100) + (\quad 1 \quad {}_{1})(1 \quad E\,CPU = Usage_{CPU_{max}})$$

Proposed scheduler
Scheduler block (I)

$$n_{score} = {}_{1}(SOC = 100) + (\quad 1 \quad {}_{1})(1 \quad E\,CPU = Usage_{CPU_{max}})$$

Proposed scheduler
Scheduler block (I)

$$n_{score} = {}_{1}(SOC = 100) + (\quad 1 \quad {}_{1})\ (1 \quad E\ CPU = Usage_{CPU_{max}})$$

Table of Contents

Evaluation
Testbed setup

Parameter	Values
Number of VNFs in a Service	5 - 10
CPU Capacity per Node (milliCPU)	4000
Memory Capacity per Node (Ki)	7998464
Required CPU per Event (milliCPU)	150 - 250
Required Memory per Event (Ki)	200 - 500
Event Arrival Rates (Events/time-unit)	2 - 12

CPU usage is measured in CPU units and is expressed as an absolute quantity. Thus, 100 milliCPU and 0.1CPU are the same amount of CPU usage in a single-core, dual-core, or 48-core machine.

Results
Utilizing unused controller resources (II)

Results
Utilizing unused controller resources (II)

Remarks:

Number of successfully scheduled events was relatively similar for both criteria

Results
Utilizing unused controller resources (II)

Remarks:

- Number of successfully scheduled events was relatively similar for both criteria
- Noticeable difference in the number of rejections and deadline violations for high-generation rates

Results
Utilizing unused controller resources (II)

Remarks:

- Number of successfully scheduled events was relatively similar for both criteria
- Noticeable difference in the number of rejections and deadline violations for high-generation rates
- Reduction of 50%, 61% and 66% in the amount of rejected events

Results
Utilizing unused controller resources (II)

Remarks:

- Number of successfully scheduled events was relatively similar for both criteria
- Noticeable difference in the number of rejections and deadline violations for high-generation rates
- Reduction of 50%, 61% and 66% in the amount of rejected events
- Reductions are more signi cant than the increments in the deadline violations, which can be up to 16%, 42% and 49%

Results
Utilizing unused controller resources (I)

Results
Utilizing unused controller resources (I)

Results
SOC regression model (II)

Results
SOC regression model (II)

Results
SOC regression model (II)

SOC regression model for any node n in the SBC cluster ($n \in N$, where $N = N_c + 1$):

$$SOC_{pred_{model}} = \beta_0 + \beta_1 pkt_{in_n} ctrl + \beta_2 cpu_n + \sum_{8i \in N_c} (\beta_{3_i} compute_i) cpu_i^2 + \beta_{4_i} compute_i$$

SOC regression model for any node n in the SBC cluster ($n \in N$, where $N = N_c + 1$):

$$SOC_{pred_{model}} = \beta_0 + \beta_1 pkt_{in_n} ctrl + \beta_2 cpu_n + \sum_{8i \in N_c} (\beta_{3_i} compute_i) cpu_i^2 + \beta_{4_i} compute_i$$

Model coef cients ( ) are obtained from the available training dataset

SOC regression model for any node n in the SBC cluster ($n \in N$, where $N = N_c + 1$):

$$SOC_{pred_{model}} = \beta_0 + \beta_1 pkt_{in_n} ctrl + \beta_2 cpu_n + \sum_{8i \in N_c} (\beta_{3_i} compute_i) cpu_i^2 + \beta_{4_i} compute_i$$

Model coef cients ( ) are obtained from the available training dataset

Categorical variables $compute_i$ represents each compute node in the model ($i \in N_c$)

SOC regression model for any node n in the SBC cluster ($n \in N$, where $N = N_c + 1$):

$$SOC_{pred_{model}} = \beta_0 + \beta_1 pkt_{in_n} ctrl + \beta_2 cpu_n + \sum_{\forall i \in N_c} (\beta_{3_i} compute_i) cpu_i^2 + \beta_{4_i} compute_i$$

Model coef cients ( ) are obtained from the available training dataset

Categorical variables $compute_i$ represents each compute node in the model ($i \in N_c$)

Binary indicator ctrl distinguishes if the node is a controller or not

SOC regression model for any node n in the SBC cluster ($n \in N$, where $N = N_c + 1$):

$$SOC_{pred_{model}} = \beta_0 + \beta_1 pkt_{in_n} ctrl + \beta_2 cpu_n + \sum_{\forall i \in N_c} (\beta_{3_i} compute_i) cpu_i^2 + \beta_{4_i} compute_i$$

Model coef cients ( ) are obtained from the available training dataset

Categorical variables $compute_i$ represents each compute node in the model ($i \in N_c$)

Binary indicator $ctrl$ distinguishes if the node is a controller or not

SOC regression model is transformed regarding the node whose SOC is to be predicted (i.e., a linear model with two predictor variables and a second-order polynomial model based on CPU usage)

Results
Comparison among other schedulers (III)

Least Loaded Scheduler (LLS) Kubernetes Scheduler (KS) SOC and Capacity-based Scheduler (SOCCS)

Results
Comparison among other schedulers (III)

Least Loaded Scheduler (LLS) Kubernetes Scheduler (KS) SOC and Capacity-based Scheduler (SOCCS)

Remarks:

For low-value generation rates, KS rejects some events (i.e., around 8 events for 5 events/time-unit)

Results
Comparison among other schedulers (III)

Least Loaded Scheduler (LLS) Kubernetes Scheduler (KS) SOC and Capacity-based Scheduler (SOCCS)

Remarks:

For low-value generation rates, KS rejects some events (i.e., around 8 events for 5 events/time-unit)

SOCCS outperforms the other algorithms in terms of scheduled VNFs for high-value generation rates

Results
Comparison among other schedulers (III)

Least Loaded Scheduler (LLS) Kubernetes Scheduler (KS) SOC and Capacity-based Scheduler (SOCCS)

Remarks:

- For low-value generation rates, KS rejects some events (i.e., around 8 events for 5 events/time-unit)
- SOCCS outperforms the other algorithms in terms of scheduled VNFs for high-value generation rates
- SOCCS reduces the rejected VNFs w.r.t LLS by 11%, 12% and 12% for high generation rates (i.e., 8, 10 and 12)

Results
Comparison among other schedulers (III)

Least Loaded Scheduler (LLS) Kubernetes Scheduler (KS) SOC and Capacity-based Scheduler (SOCCS)

Remarks:

- For low-value generation rates, KS rejects some events (i.e., around 8 events for 5 events/time-unit)
- SOCCS outperforms the other algorithms in terms of scheduled VNFs for high-value generation rates
- SOCCS reduces the rejected VNFs w.r.t LLS by 11%, 12% and 12% for high generation rates (i.e., 8, 10 and 12)
- These reductions are higher in comparison with KS, demonstrated by the values up to 16%, 23% and 18%

Results
Comparison among other schedulers (II)

Results
Comparison among other schedulers (II)

Results
Comparison among other schedulers (I)

Results
Comparison among other schedulers (I)

Remarks:

The greater the event arrival rate, the higher the battery consumption

Results
Comparison among other schedulers (I)

Remarks:

The greater the event arrival rate, the higher the battery consumption

KS has the highest battery consumption for all the generation rates

Results
Comparison among other schedulers (I)

Remarks:

The greater the event arrival rate, the higher the battery consumption

KS has the highest battery consumption for all the generation rates

SOCCS guarantees the lowest battery consumption as it always selected the node with the maximum SOC and minimum CPU usage

Table of Contents

Conclusion

We proposed a scheduling algorithm, i.e. SOCCS, that assigns events to an SBC cluster by considering predicted battery consumption and used resources.

We analyzed the case of using the unused controller resources to deploy certain services and tasks. The obtained results confirmed that such consideration is a convenient criterion in a resource-constrained environment.

The evaluation results revealed that SOCCS outperformed the baseline algorithms with regard to analyzed metrics such as rejected and scheduled events, deadline violations in scheduled events and battery consumption.

Our scheduler reduced the cluster's operation time to process the requested events which indisputably had a lower impact in the battery consumption with regard to the other schedulers.

Thank you very much for your time.

Any question?