PROACTIVE ELASTIC CONTROL FRAMEWORK

A LOCATION-AWARE SOLUTION FOR LATENCY-INTOLERANT MOBILE EDGE CLOUD APPLICATIONS

LUDOVICO DE SANTIS – 0320460

INTRODUCTION

- Modern MEC applications are latency-sensitive (ex. AR applications).
- Edge Data Centers (EDCs) to front the latency problem, but there are limited/high cost resources, optimization is strictly necessary.
- The prediction of workload is the core-activity for optimization of performances ad resource utilization.
- Allocated resources are not ready to be used immediately, due to actuation delays →focus on correct prediction and system stability.
- The concept of elasticity must be extended with modern, edge native solutions to cope latency.

WHAT IS ELASTICITY?

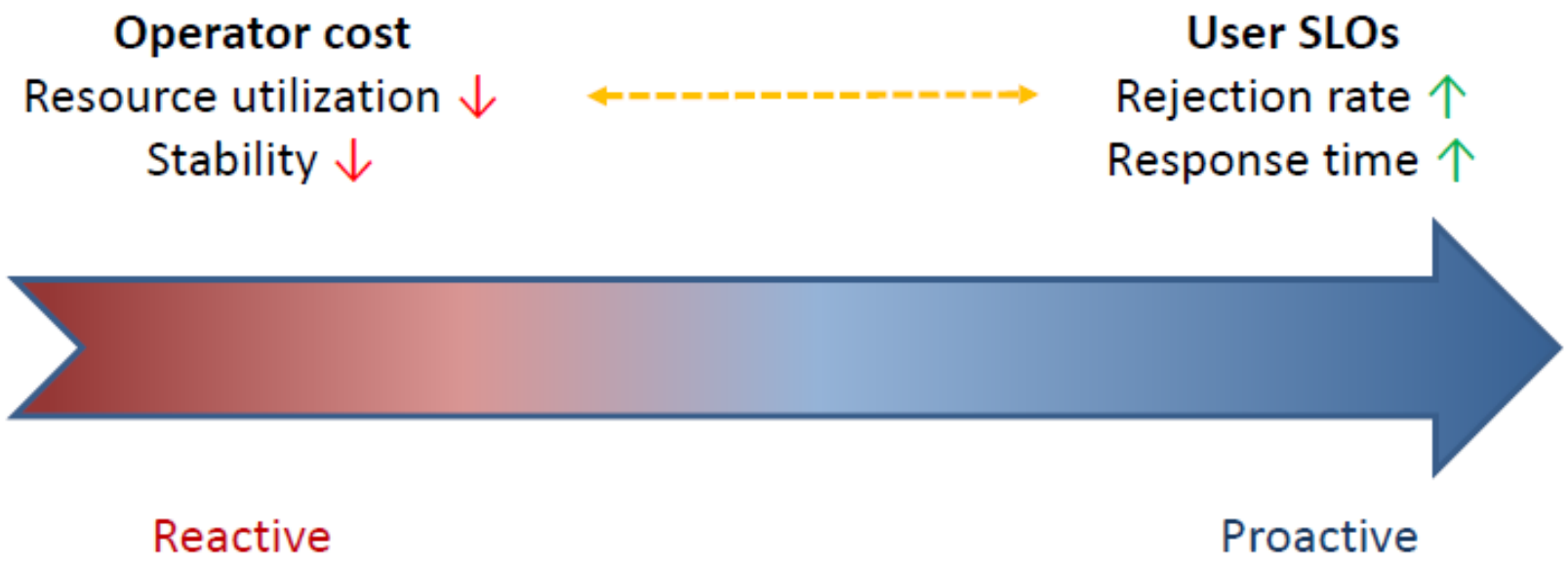
- Elasticity is the ability of a system to automatically adapt resource provisioning as required to handle variation in load.

Current Demand = Allocated Resource

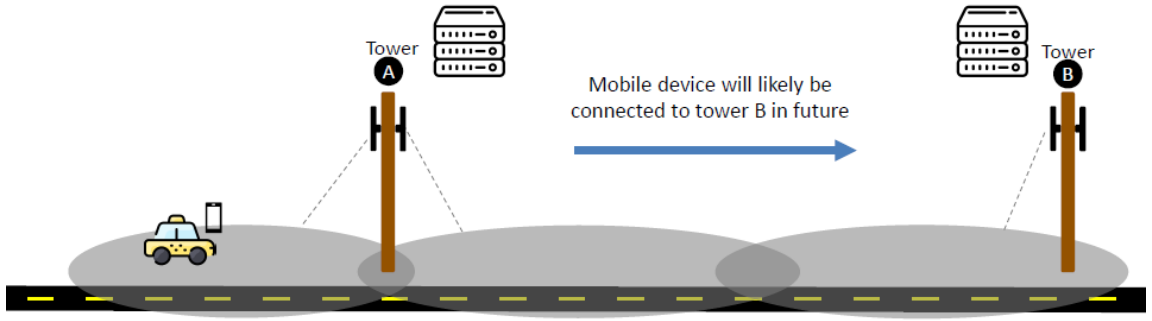
ELASTIC CONTROLLER - GOALS

- Response Time
- Low rejection rate
- Resource optimization
- Stability

TRADITIONAL APPROACHES - A FIRST RAW COMPARISON



BASIC IDEA – LOCATION AWARENESS

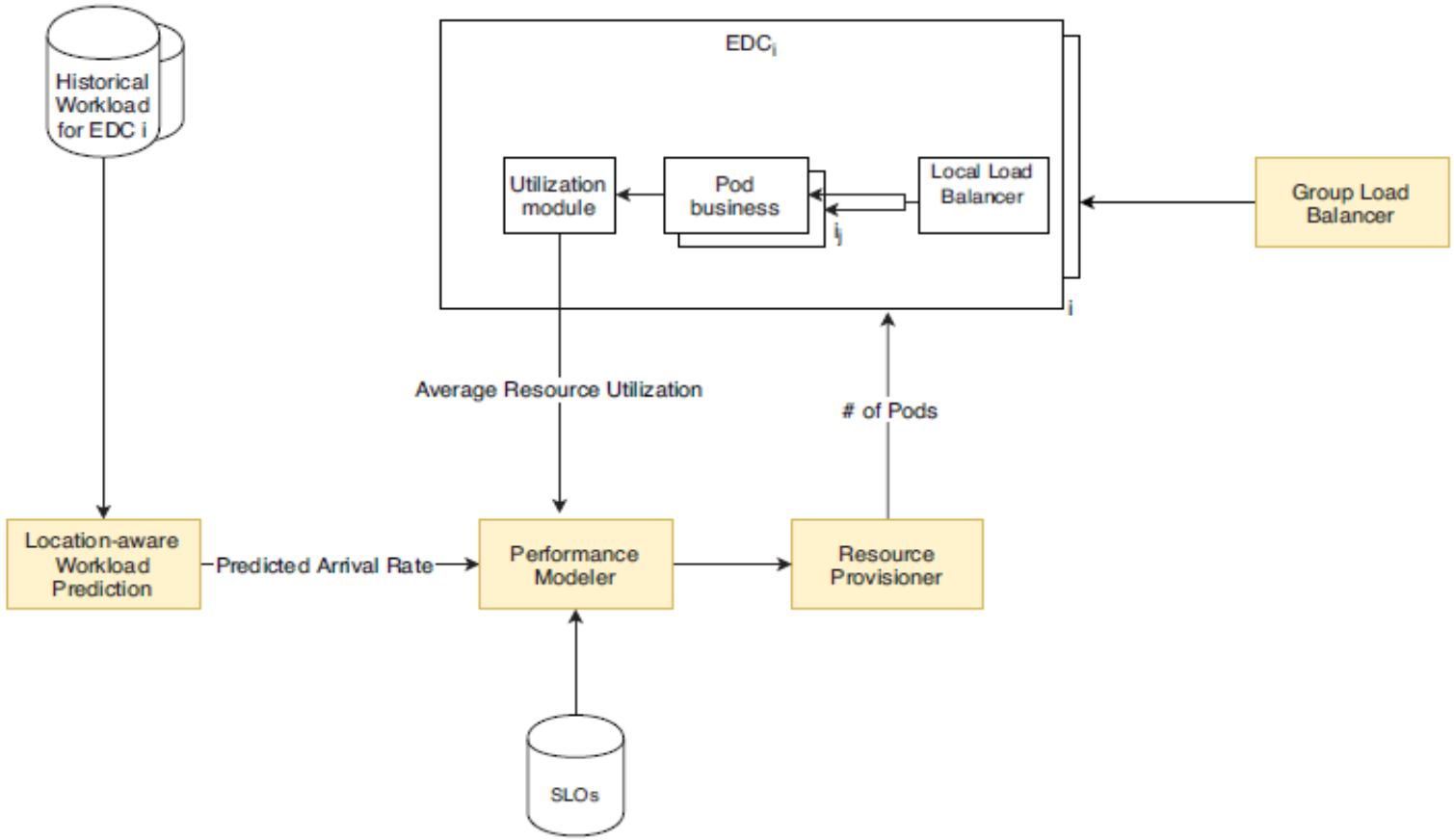


- Proactive scaling based on **location-awareness** to predict workload

→Based on correlation between changes in workload among EDCs located in close physical proximity to predict how the workloads of the EDCs in an MEC will evolve over short periods of time.

- Not enough: **Group Load Balancer** to redistribute workload between over-provisioned EDCs and under-provisioned EDCs.

FIRST OVERVIEW



SOME DEFINITIONS

- **Location-aware workload Predictor**

It predicts workload arrival rates at each EDC in the group over a configurable interval. Based on historical data integrated with runtime data each configurable time interval.

- **Performance Modeler**

It' s responsible for estimating the number of resources required at each EDC to meet the SLOs. Resources are astracted at Pod. At any given time, there is only one request being served by a Pod, the subsequent requests are queued. Each EDC thus can thus be modeled as a set of *m* **M/M/1/k/FIFO** where *m* is the total number of Pods.

- **Resource Provisioner**

It is responsible for determining the number of Pods to be scaled up/down at each EDC in a group, cross-evaluating the resource requirements of all EDCs in a group and setting a final number of desired resources at each EDC based on the current aggregated number of resources in that particular group.

- **Group Load Balancer**

It redistribute workload between over-provisioned EDCs and under-provisioned EDCs, based on **weight round-robin** approach.

EXPERIMENT – WARM UP

Fig. 4: Distribution of EDCs in San Francisco.

- The MEC is modeled by supposing that one EDC is collocated with each unique tower to provide coverage of the studied area. We assume that the resources of each EDC are virtualized according to the Kubernetes model (Pod abstraction).
- Each end user's request, if admitted, is served by an application instance running on a Pod allocated by EDCs.
- Application is assumed to be an extremely latency-intolerant application.
- Real Taxi Mobility Traces used to simulate Workload arrivals at EDCs.

PREREQUISITES

- SLOs
  - → Avg Utilization 80%
  - →Rejection Rate < 1%
- 3 type of controllers compared:
  - →Re-active AutoScaler: KubernetesHPA
  - →Pro-active AutoScaler
  - →Pro-active AutoScaler+GroupLoadBalancer

MECs	
number of EDC	15
maximum Pods at each EDC	300
minimum Pods at each EDC	5
Pod readiness delay	1 minute
Application	
average response time	1 ms
maximum response time	2.5 ms
Targeted threshold	
avg_utilization_thresh	80%
max_rejectionRate_thresh	1%
Proposed Pro-active auto scaler settings	
number groups	4
look-ahead arrival rate prediction	5 minutes
Re-active auto scaler setting	
downscale-stabilization	5 minutes
periodSeconds	15 seconds
scaleUpLimitFactor	2.0
scaleUpLimitMinimum	4.0
tolerance	10%

METRICS

- Measured metrics
  - → Accuracy
  - →Timeshare
  - →Instability
- Both Accuracy and Timeshare are measured in Under/Over Provisionig scenarios.

EXPERIMENT - OVERVIEW

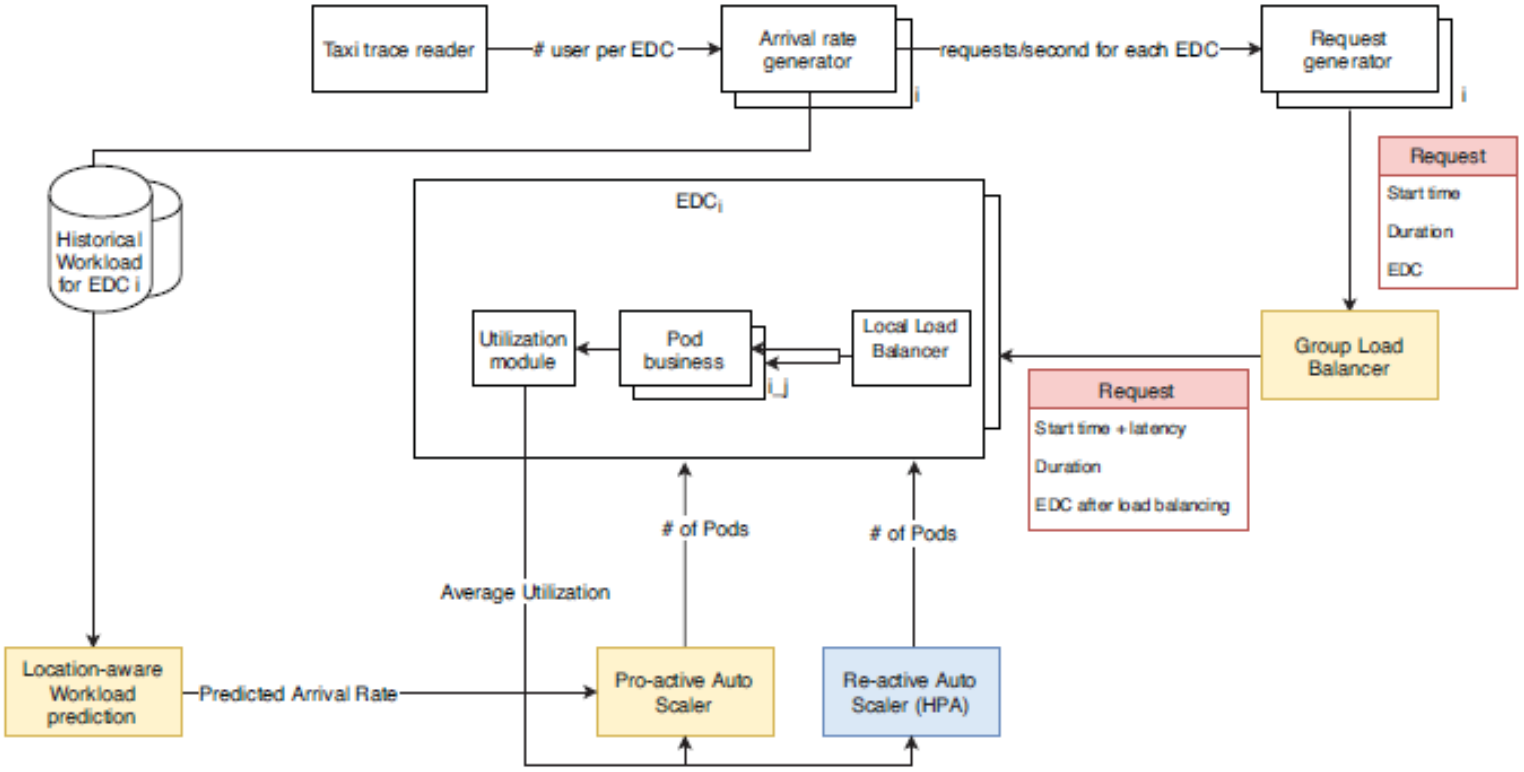
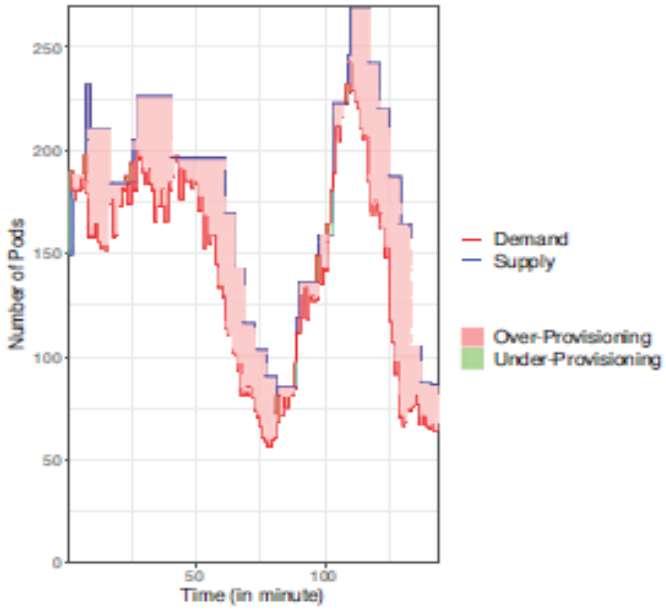
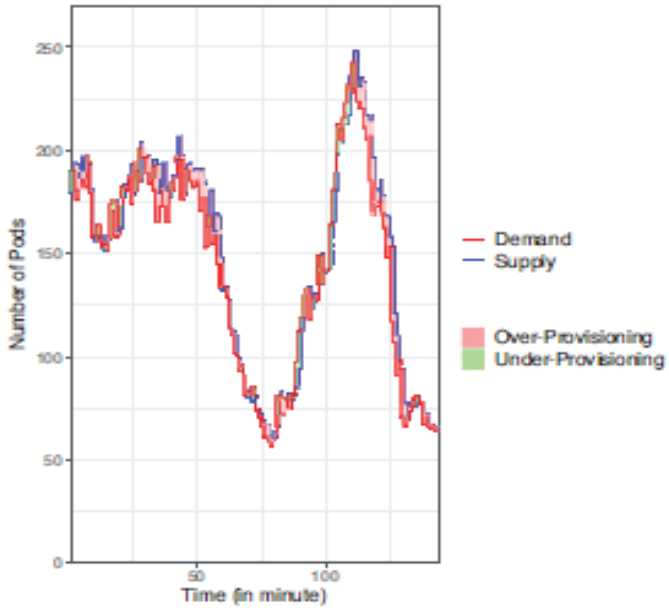


Fig. 3: The experimental simulation.

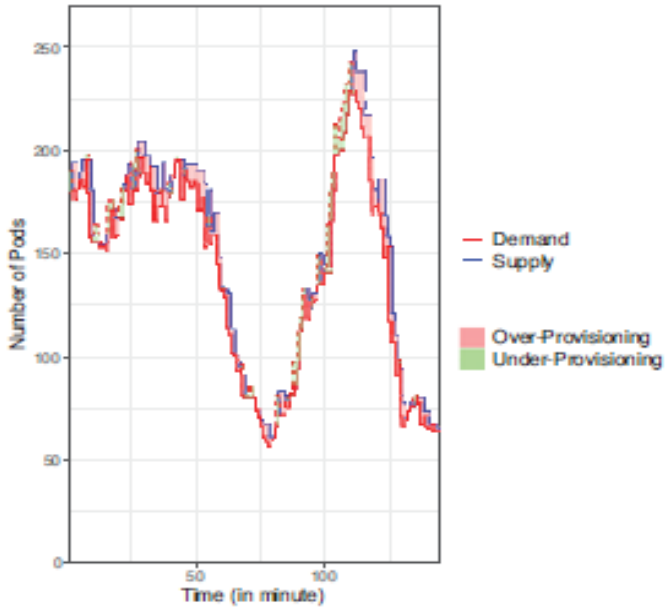
COMPARISON OF RESULTS (1)



(a) Re-active AS



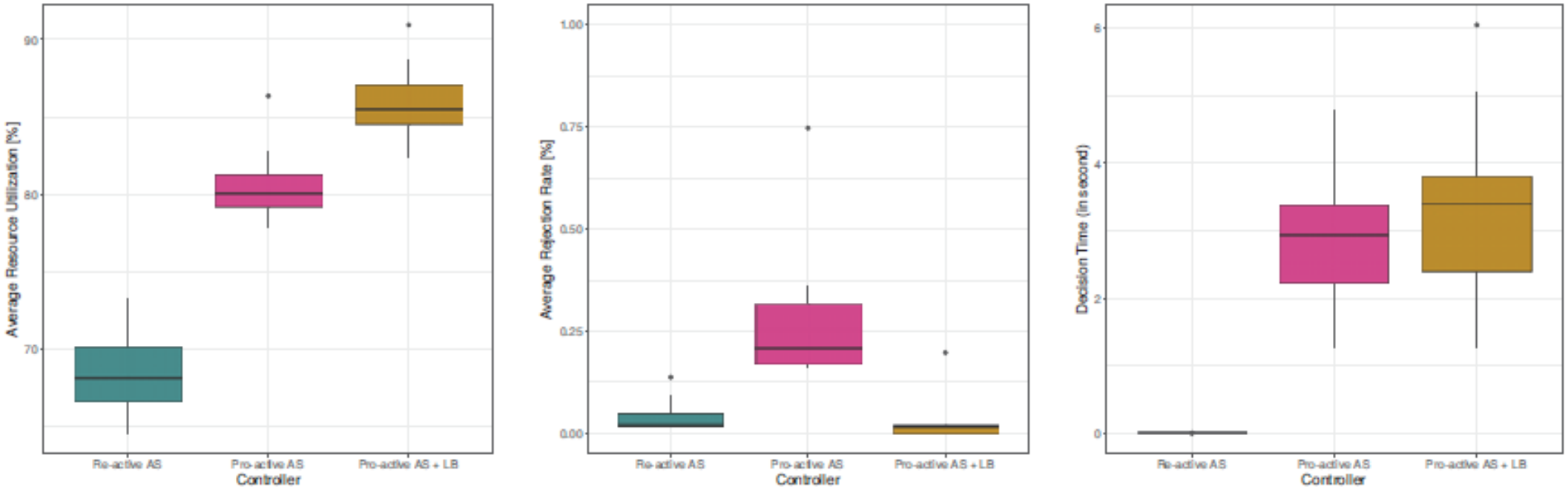
(b) Pro-active AS



(c) Pro-active AS + LB (the red dashed curve: the demand before load balancing; the red curve: the new demand after load balancing)

Fig. 5: The scaling behavior of three controllers on EDC#1 (*avg_utilization_thresh*= 80%, *max_rejectionRate_thresh* = 1%).

COMPARISON OF RESULTS (11)



(a) Average resource utilization.

(b) Average rejection rate.

(c) Average decision time.

Fig. 6: Performance of the three controllers based on the main elasticity metrics (*avg_utilization_thresh*= 80%, *max_rejectionRate_thresh* = 1%).

COMPARISON OF RESULTS (111)

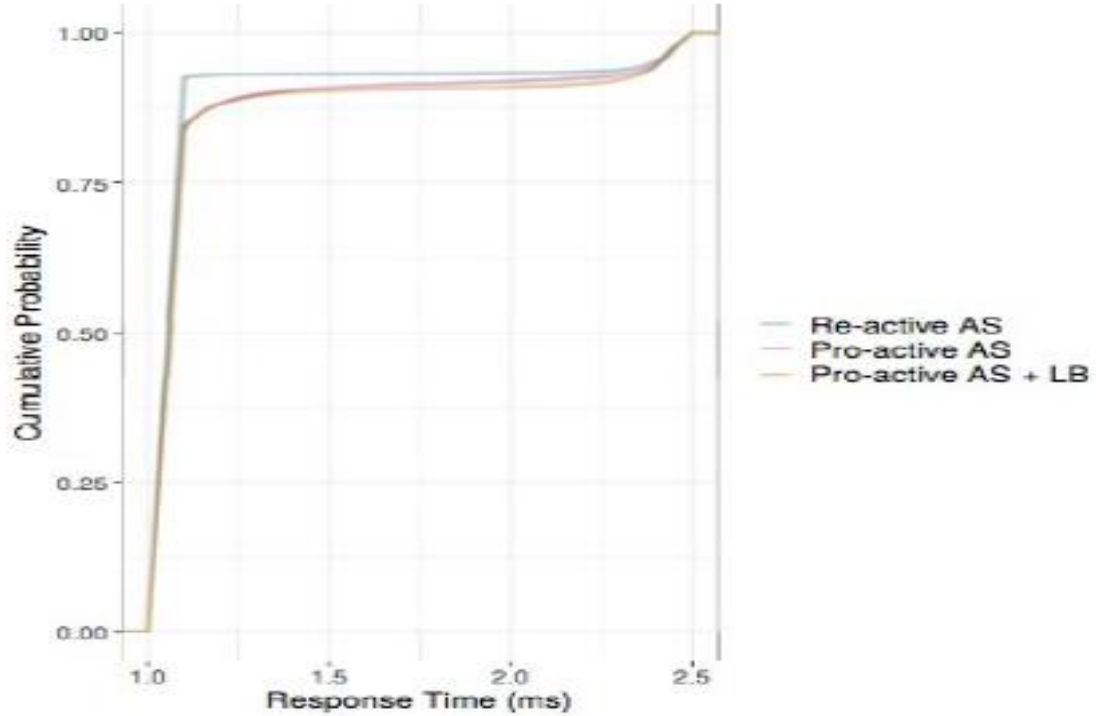


Figure 6: Cumulative density of response times of the application in three elastic controller settings.

COMPARISON OF RESULTS - METRICS

Metric	Pro-active AS + LB	Pro-active AS	Re-active AS
θ_U	13.6	41.2	**5.4**
θ_O	**14.2**	39.5	305.6
τ_U	**4%**	43%	5.3%
τ_O	**2.5%**	46.7%	94.1%
υ	**2.44%**	2.8%	3.9%
Avg. resource utilization	**85.9%**	80.5%	68.4%
Rejection rate	**0.02%**	0.26%	0.04%
total Pods	**3154**	4405	5337
Avg. Pod lifetime (minute)	**73.3**	35.2	29.6

GOING DEEPER…

- Is the location awareness a **true advantage**?

  →The proposed controller estimates the number of Pods desired at each EDC based on the request arrival rate predicted by the workload prediction model. Its performance thus depends strongly on the accuracy of the predictive model.

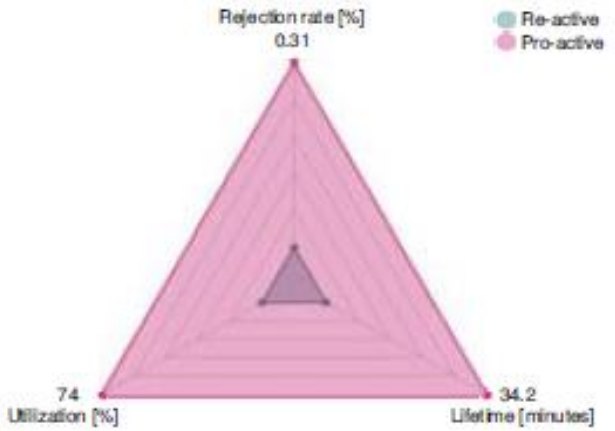
  →The variations in the workloads of neighboring EDCs is correlated, which can be exploited to improve the accuracy of prediction in MECs.

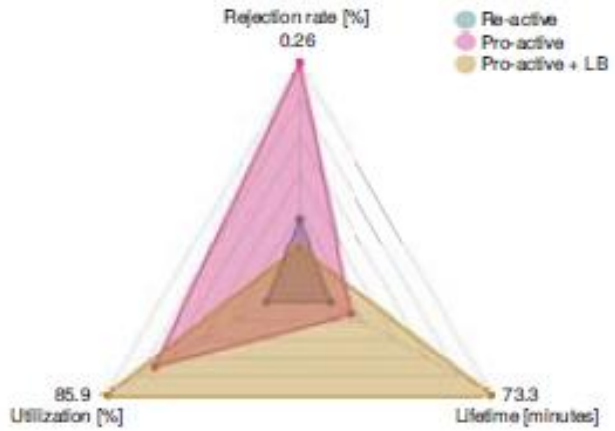
- But…

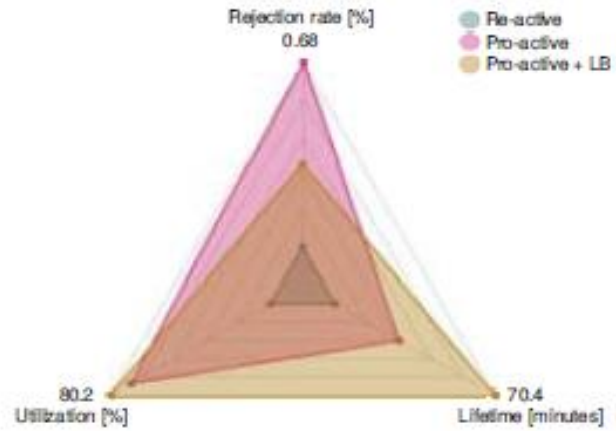
  →the increase in predictive accuracy becomes saturated once the number of neighboring EDCs exceeds a certain threshold.

- **Let's see…**

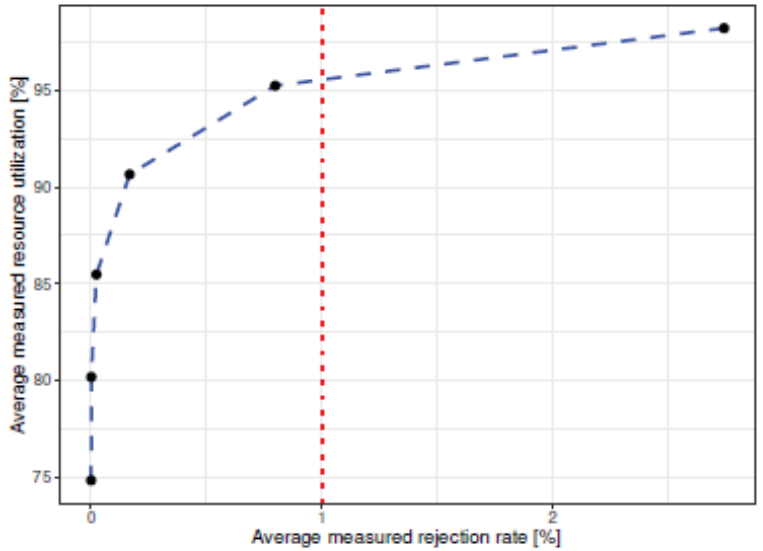
LOCATION AWARENESS - DISCUSSION



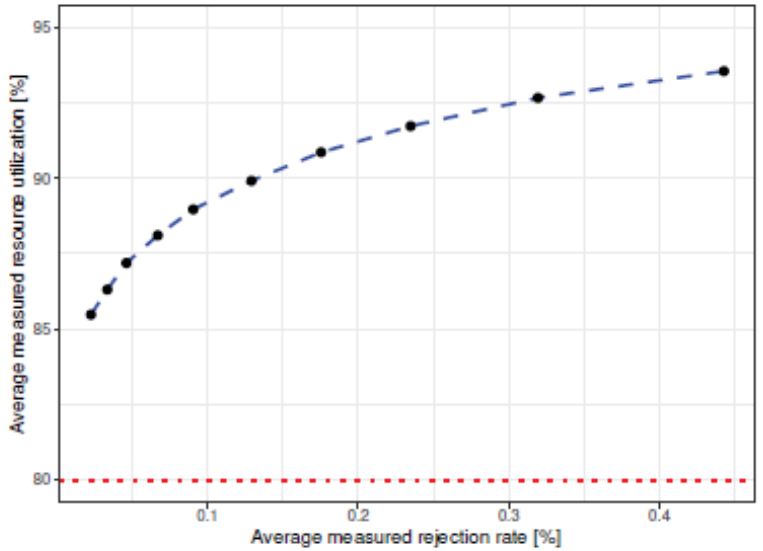




THRESHOLDS - DISCUSSION



(a) The targeted resource utilization is changed, while the targeted rejection rate is held constant at 1%.



(b) The targeted rejection rate is changed, while the targeted resource utilization is held constant at 80%.

Fig. 9: The controller's scaling behavior when varying the threshold settings.

SELF ADAPTATION

- Category: Locus of responsibility
  - (from the application level viewpoint)
- Tactics :
  - *total transparency*
  - *total responsibility*
  - *application-aware*

The application provides informations to the system to drive adaptation, but this informations are managed and processed by the system.

SELF ADAPTATION

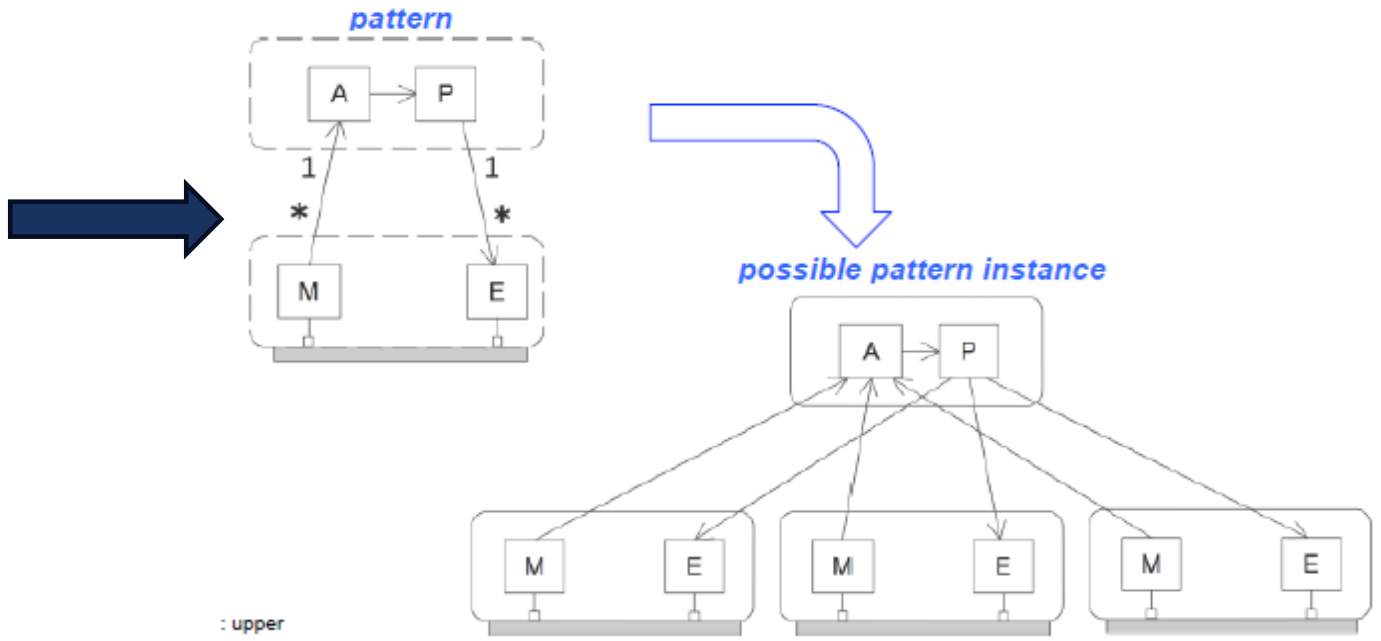
- Category: Type of control
- Tactics :
  - *Top-down (explicit feedback loop)*
  - *Bottom-up (emergent behavior)*

Explicit feedback loop →Mape-K

MAPE-K → MASTER/WORKER PATTERN

- Category: Control architecture
- Tactics :
  - *decentralized control*
    - *hierarchical control*
  - *centralized control*

ıpper



SELF ADAPTATION

- Category: Malleability
- Tactics :
  - *variable data fidelity*
  - *loosely coupled architecture*
    - *loose connectors*
    - *loose deployment/binding*

Runtime Binding

CONCLUSIONS

- In this paper, we introduce a local-aware pro-active controller for dynamic provisioning of resources to services running in MECs. The proposed controller takes advantage of the correlation of workload variation in physically neighboring EDCs to predict the request arrival rate at EDCs.
- These predictions are then used as inputs to estimate service demand and the number of resources that will be desired at each EDC.
- Additionally, to minimize the request rejection rate, we develop a group-level load balancer to redirect requests among EDCs during runtime.
- This controller has better scaling behavior than a state-of-the-art re-active controller and also improves the efficiency of resource provisioning, satisfying SLOs while maintaining system stability.

COMPANIES POV – A CRITICAL ANALYSIS

- The paper focus on performances, not mentioning a crucial factor for the providers' point of view: the cost.
- The Proactive approach, in this case used in combination with group load balancer, provides the best performances, furtherly proved by the experiments showed before. But this approach is really expensive, lot of computations are needed, it generates comunicative overhead, with occupation of bandwidth.
- A cost-based analysis is necessary to integrate this job in a more realistic scenario, meeting the companies' necessity to maximize profits.
- An Hybrid solution between Proactive and Reactive approach could be a solution to reduce costs, maintaning an acceptable level of performance.

THANK YOU!