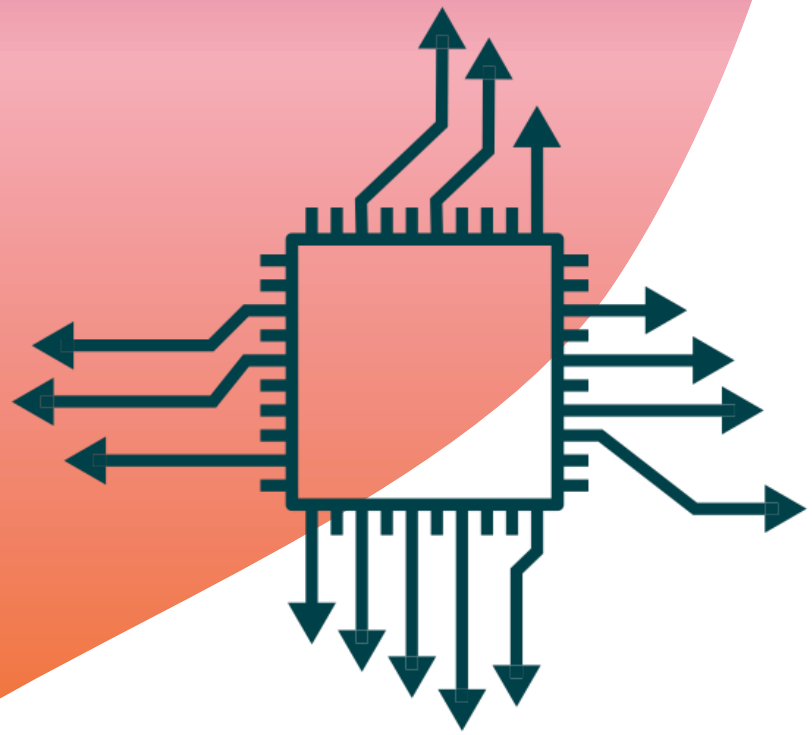
UNIVERSITY OF THESSALY
creative years
1988 — 2018

iPRISM
Intelligent Pervasive Systems Research Group
Dept. of Informatics and Telecommunications

A QoS-aware, Proactive Tasks Offloading Model for Pervasive Applications

- Georgios Boulougaris
  gboulougar@uth.gr
- Kostas Kolomvatsos
  kostasks@uth.gr

Outline



- Introduction
- Preliminaries
- Proposed Approach
- Experimental Evaluation
- Conclusion & Future Work

Introduction

IoT

Numerous Devices

- Become the hosts and 'administrators' of the collected data, while interacting with end users and their environment.
- The goal is to have services and data processing mechanisms fully adaptive to users' demands especially in Pervasive Computing applications.

Centralized Legacy Systems (Cloud)

- Data are transferred across the network to be processed in remote data centers.
- Significant obstacles: increased latency, limited processing control, unnecessary resource consumption, safety and privacy vulnerabilities.
- Difficulties arise in maintaining the desired levels of Quality of Service (QoS).

Edge Computing (EC) Nodes

- EC nodes can be transformed to intelligent, autonomous entities that process the available data and provide responses.
- EC nodes should apply a selective strategy concerning the tasks that will be executed locally or offloaded.

Contribution

- We provide a distributed decision-making mechanism for PC tasks scheduling, taking into consideration multiple criteria/parameters.
- We propose a QoS-aware, proactive tasks offloading model upon the continuous monitoring of the performance of EC nodes.

Preliminaries

01

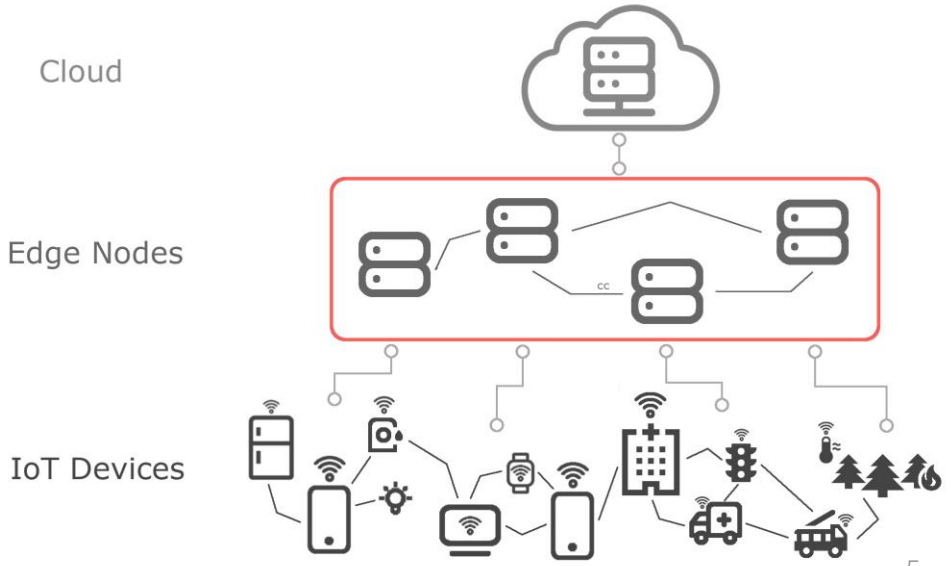
- We consider a set of edge nodes $N = \{n_1, n_2, \ldots, n_{|N|}\}$ which are interconnected and form a graph $G = (N, E)$, where E is the set of edges between them.
- Each $e_{ik} \in E$ connects two nodes and is subject to a communication cost $cc_{ik} > 0$.

02

- Nodes are also connected with a number of IoT devices that report data to them.
- In this ecosystem, nodes exchange the statistical information of the local datasets to inform their peers and facilitate their decision-making.

03

- Apart from vectorial data, nodes receive a number of different, independent, single core and interference free tasks requested by end users or applications.
- Another source that feeds the stream of tasks deals with tasks offloaded from peer nodes.
- All tasks are placed in a queue.



Preliminaries

04

- Assume now that k tasks are present in the queue.
- The 1st task in the queue is going to be executed and 'gets' the resources of the corresponding node.
- The node monitors QoS levels upon multiple parameters and, if needed, decides to select a subset from the k-1 tasks to be offloaded in peer nodes.
- When offloading tasks, the remaining ones will enjoy more quickly the local resources, reducing their total response time.

05

- Nodes should repeatedly follow a set of very simple high level steps and take specific decisions, i.e.,

  Decision A: Select the tasks that will be offloaded

  Decision B: Select the appropriate peer to host every task.
- Decisions A and B are made on condition that QoS is not at acceptable levels.
- The decisions are made based on:
  - the statistical information of local datasets and
  - the communication cost.

Preliminaries

06

In order to quantitatively measure QoS, we study two aspects of node performance, namely

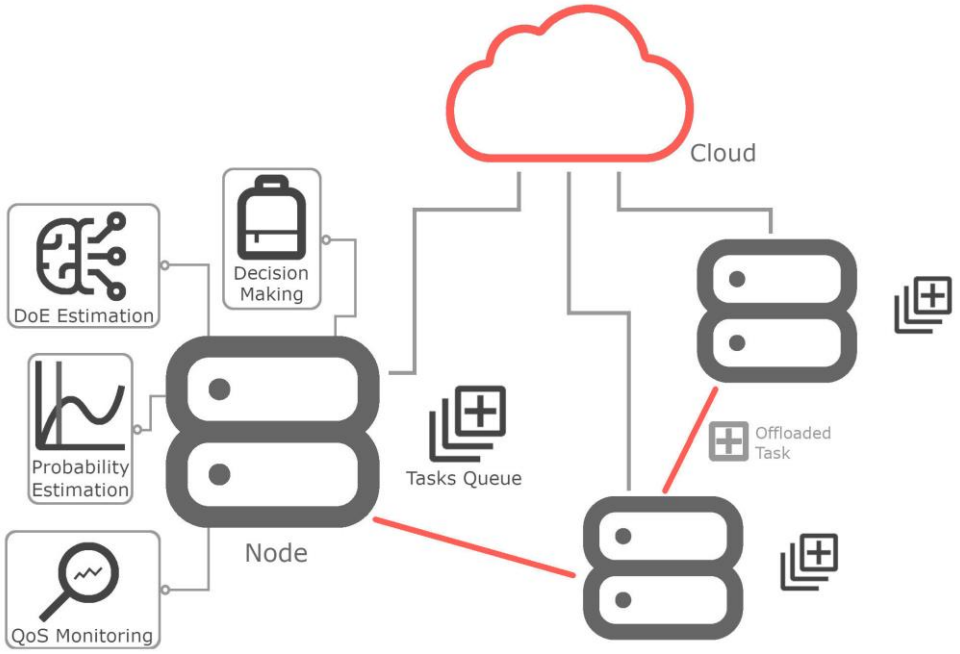
- response time (*RT*) and
- throughput (*TP*).
- $QoS = f(RT, TP)$, $QoS \in [0, 1]$.

07

- Nodes are capable of executing a variable set of tasks $T = \{t_1, t_2, \ldots\}$ depending on the dynamic networked environment and their resource constraints.
- Every task is described by a tuple of characteristics $Ch = \{dd, l, dl\}$ where
  - *dd* is the data dependency,
  - *l* is the load that the task will cause on the host node and
  - *dl* is the task deadline.

08

- Each task is also accompanied by β indicating the offloading hops in the EC ecosystem, till it is finally executed.
- A maximum hops number *B* is imposed to avoid tasks starvation.



Preliminaries

09

- When an offloading action should be present, the nodes check their queues and apply the proposed mechanism for selecting the appropriate tasks.
- They feed an ANN with the values of the characteristics and in turn it infers the *Degree of Execution – DoE* $\in [0, 1]$.
- DoE is a measure that depicts the necessity of local execution.
  - The higher the DoE, the more imperative it is to execute the corresponding task locally.
  - The opposite denotes that local execution is not so critical.
  - *DoE = y(dd, l, dl)*

10

- Once the DoE has been estimated for each of the enqueued tasks, they, thereafter, are sorted in a descending order.
- The EC node, then, decides which of the tasks will be executed locally and which of them should be offloaded.
- It makes use of a suitable mechanism (i.e., the solution of a 0-1 Knapsack problem) which ensures that the resource requirements will be fulfilled, while simultaneously the sum of the DoE is maximized.

Proposed Approach

QoS Modeling and Monitoring

- n_i, estimates the probability of having the QoS less than a pre-defined threshold *Th*, i.e., *P(QoS ≤ Th)*.
- n_i tries to calculate the following probabilities:

$$P_{RT} = P(RT \geq Th_{RT}) = 1 - P(RT \leq Th_{RT})$$

$$P_{TP} = P(TP \leq Th_{TP})$$

- In fact, both RT and TP are continuous random variables and we have to estimate their probability density function (pdf) and cumulative distribution function (cdf).
- We rely on the widely known Kernel Density Estimation (KDE).
- The probabilities produced are then combined with the adoption of the Geometric Mean. The following equation holds true:

$$P_{QoS} = \frac{P_{RT}^{w_{RT}} * P_{TP}^{w_{TP}}}{P_{RT}^{w_{RT}} * P_{TP}^{w_{TP}} + (1 - P_{RT})^{w_{RT}} * (1 - P_{TP})^{w_{TP}}}$$

where w_{RT} and w_{TP} are the weights for QoS parameter.

Proposed Approach

Estimating the Degree of Execution

- In case the QoS probability estimation approaches or falls below a certain threshold an evaluation step of the enqueued tasks takes place.
- DoE ∈ [0, 1] is estimated for each task making use of an ANN.
- We utilize a three-layered feed forward ANN with small needs for computing resources where data related to the realization of *dd*, *l* and *dl* feed the (first) input layer, penetrate the (second) hidden layer and end up in the (third) output layer with the DoE estimation form.

Proposed Approach

Tasks Offloading and Allocation

- After the tasks' DoE estimation, our mechanism concludes the enqueued tasks' execution destination, whether the tasks will be locally executed or will be offloaded.
- The decision-making process is directed by the solution of a 0-1 Knapsack problem.
- The solution of the following optimization problem illustrates the tasks that are preferable to be executed locally.

$$Maximize \quad \sum_{j=1}^{k} DOE_j x_j$$

$$Subject \quad to \quad \sum_{j=1}^{k} l_j x_j \leq L, x_j \in \{0,1\}$$

$$\sum_{j=1}^{k} (1 - x_j) cc_{is} \leq CL$$

$$x_j = 1, \quad if \quad b_j = B$$

- x_j depicts the number (restricted to a binary value) of instances of task j to be included in the knapsack.
- If a task is finally included in the knapsack ($x_j = 1$), its local execution is recommended. Otherwise ($x_j = 0$), the task should be offloaded to a neighboring node or in the Cloud.
- *CL* is an upper limit for the total communication cost for all the offloaded tasks and cc_{is} is the communication cost between n_i and the selected (s) node.

Experimental Evaluation

$$lc_{ti} = b * l$$

Local Execution Cost

$$oc_{ti} = b * l + 2 * cc_{ik}$$

Offloading Cost

TN

task offloading is correctly chosen when *lc* > *oc*

FP

task local execution is incorrectly chosen when *lc* > *oc*

FN

task offloading is incorrectly chosen when *lc* ≤ *oc*

TP

task local execution is correctly chosen when *lc* ≤ *oc*

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy A

$$P = \frac{TP}{TP + FP}$$

Precision P

$$R = \frac{TP}{TP + FN}$$

Recall R

$$F = \frac{2 * P * R}{(P + R)}$$

F-measure F

Experimental Evaluation

Comparative Analysis

Selection of Appropriate Peer

- Lowest Work Load
- Lowest Task's Offloading Cost

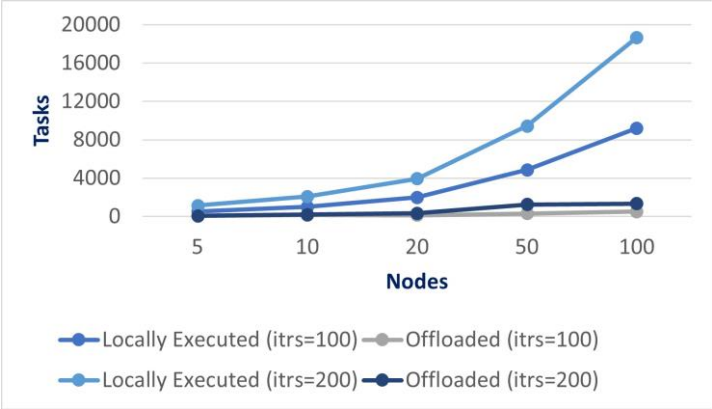
Offloading Method

- Model
- Random
- Last
- Greedy

Parameters	Values
b	0.5
cc	[0, 1]
dd	[0, 1]
dl	[0, 1]
E	100
itrs	{100, 200}
l	[0, 1]
L	[5, 10]
N	{5, 10, 20, 50, 100}
RT	[0, 1]
Th	0.3
TP	[0, 1]
w_{RT}	0.5
w_{TP}	0.5

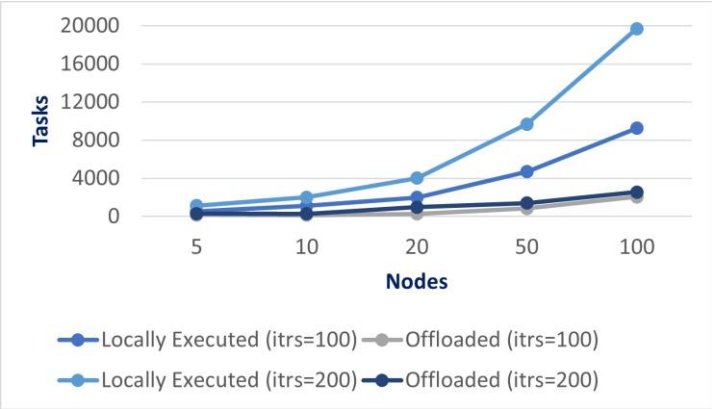
Experimental Evaluation

Selection of Appropriate Peer



Lowest Work Load

	100 itrs				200 itrs			
N	A	P	R	F	A	P	R	F
5	0.789	0.915	0.849	0.881	0.887	0.926	0.954	0.940
10	0.820	0.934	0.868	0.900	0.824	0.873	0.929	0.901
20	0.853	0.903	0.938	0.920	0.860	0.900	0.945	0.922
50	0.851	0.898	0.941	0.919	0.803	0.871	0.903	0.887
100	0.850	0.887	0.952	0.918	0.826	0.861	0.947	0.902



Lowest Task's Offloading Cost

	100 itrs				200 itrs			
N	A	P	R	F	A	P	R	F
5	0.812	0.888	0.870	0.879	0.744	0.868	0.817	0.841
10	0.786	0.794	0.956	0.868	0.814	0.829	0.953	0.887
20	0.738	0.735	0.957	0.832	0.722	0.731	0.905	0.809
50	0.629	0.596	0.945	0.731	0.601	0.559	0.971	0.710
100	0.547	0.478	0.938	0.633	0.492	0.433	0.984	0.602

- The majority of the enqueued tasks are locally executed.
- Metrics values show a remarkable stability in case the necessary task offloading is carried out at the peer node with the lowest load. Our model seems to achieve high levels of efficiency.
- The opposite holds true when the pursuit of the lowest task offloading cost is the criterion.

Experimental Evaluation

Offloading Method

N	Offloaded Tasks (%)
5	14.65%
10	13.22%
20	6.45%
50	6.15%
100	5.19%

- The percentage is getting smaller and smaller as the number of nodes increases.

	Model			
N	A	P	R	F
5	0.789	0.915	0.849	0.881
10	0.820	0.934	0.868	0.900
20	0.853	0.903	0.938	0.920
50	0.851	0.898	0.941	0.919
100	0.850	0.887	0.952	0.918

Average Improvement (10%)			
A	P	R	F
-6.047%	-4.360%	-2.564%	-3.425%
0.202%	0.430%	-0.045%	0.189%
5.677%	-1.025%	7.505%	3.168%
3.794%	-1.255%	5.657%	2.127%
5.401%	-2.609%	9.024%	3.010%

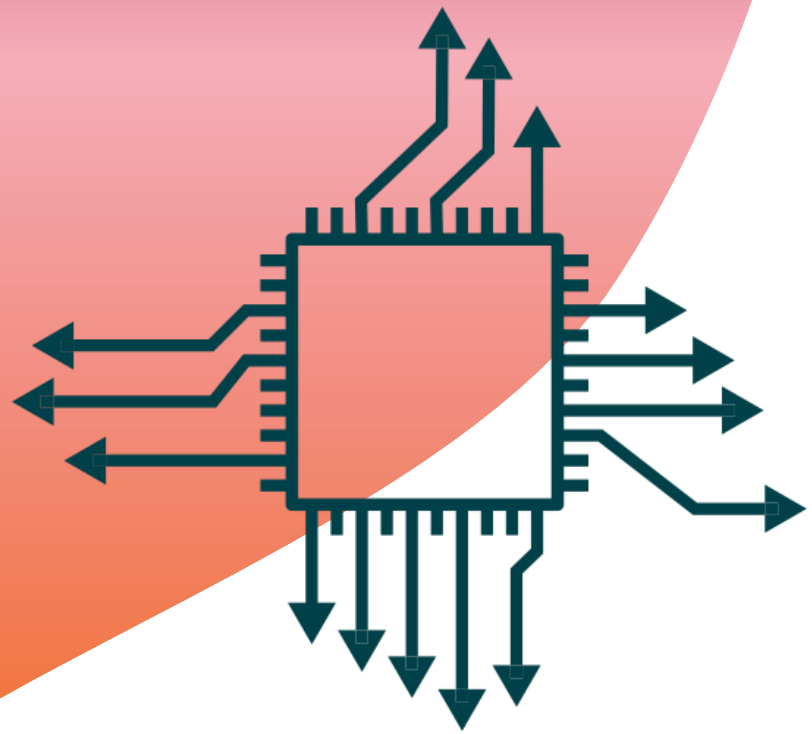
	Random (10%)				Last (10%)				Greedy (10%)			
N	A	P	R	F	A	P	R	F	A	P	R	F
5	0.823	0.948	0.861	0.903	0.848	0.953	0.883	0.917	0.847	0.968	0.871	0.917
10	0.793	0.916	0.851	0.882	0.851	0.942	0.896	0.919	0.810	0.933	0.857	0.894
20	0.807	0.906	0.876	0.891	0.794	0.917	0.852	0.883	0.822	0.913	0.890	0.901
50	0.817	0.903	0.893	0.898	0.820	0.903	0.896	0.899	0.823	0.923	0.882	0.902
100	0.819	0.906	0.892	0.899	0.790	0.910	0.853	0.881	0.812	0.915	0.876	0.895

- The metrics values support the claim that our model constitutes an effective classification method.

	Random (5%)				Last (5%)				Greedy (5%)			
N	A	P	R	F	A	P	R	F	A	P	R	F
5	0.874	0.897	0.970	0.932	0.900	0.926	0.970	0.947	0.891	0.921	0.964	0.942
10	0.835	0.848	0.980	0.909	0.881	0.899	0.976	0.936	0.836	0.859	0.968	0.910
20	0.832	0.855	0.966	0.907	0.847	0.864	0.975	0.916	0.855	0.878	0.969	0.921
50	0.839	0.849	0.984	0.912	0.836	0.854	0.973	0.910	0.845	0.863	0.974	0.915
100	0.846	0.858	0.982	0.916	0.825	0.838	0.980	0.903	0.839	0.859	0.972	0.912

- The proposed model seems to achieve similarly good results compared to the alternative approaches.

Conclusion & Future Work



- We propose a distributed and intelligent tasks offloading model which aims to eliminate the tasks migration to the Cloud, while satisfying high QoS levels.
- Each EC node, operating autonomously, systematically observes its performance and it is proactively possible to select some tasks to be offloaded to neighbors or to Cloud, based on their multiple characteristics.
- The experimental evaluation shows that the proposed model effectively concludes the right decision-making which ensures that resource constraints are met.
- Future research involves the estimation of the overhead for task monitoring, decision-making and comparison with some ‘task level’ models.

iPRISM

THANK YOU

Georgios Boulougaris
gboulougar@uth.gr

http://www.iprism.eu