Context-aware Edge Process Management for Mobile Thing-to-Fog Environment

Jakob Mass
Mobile & Cloud Lab
Institute of Computer Science
University of Tartu, Estonia
jakob.mass@ut.ee

Chii Chang
Mobile & Cloud Lab
Institute of Computer Science
University of Tartu, Estonia
chii.chang@acm.org

Satish Narayana Srirama
Mobile & Cloud Lab
Institute of Computer Science
University of Tartu, Estonia
srirama@ut.ee

ABSTRACT

Service-oriented Workflow Management Systems (WfMS) have been utilised as an efficient solution to manage the objects and activities involved in Internet of Things (IoT) systems. As researchers have identified the drawbacks of the conventional centralised architecture which relies on the distant, central management server for managing all processes, recent IoT systems architectures have started including mechanisms for distributing computational or networking processes and decision making at the edge network, where the front-end IoT devices are located. We term processes involved in this paradigm as Edge Processes. Managing Edge Processes faces challenges when the involved IoT devices are moving. Specifically, improper timing of executing processes & their subtasks will cause the waste of resources such as battery consumption. Hence, edge processes require adaptive scheduling schemes to increase efficiency and reduce waste of resources. In this paper, we propose an edge process management system architecture together with an adaptive task scheduling scheme which addresses the described issue.

KEYWORDS

Internet of Mobile Things, Task Scheduling, Edge Process Management

ACM Reference Format:
Jakob Mass, Chii Chang, and Satish Narayana Srirama. 2018. Context-aware Edge Process Management for Mobile Thing-to-Fog Environment. In *12th European Conference on Software Architecture: Companion Proceedings (ECSA '18), September 24–28, 2018, Madrid, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3241403.3241449

1 INTRODUCTION

Service-Oriented Architecture (SOA) is a promising approach to improve the interoperability of heterogeneous Internet of Things (IoT) [2], in which systems can integrate IoT devices as atomic services or leverage a group of IoT devices as composite services. For example, when applying SOA to Internet of Mobile Things (IoMT)-based healthcare applications [1], the system may host a microservice server on the patient's mobile device as a gateway, which interconnects the wearable sensors of the patient to the distant IoMT central server. From the perspective of the central server, the microservice[1] hosted on the IoMT device is providing a composite service that consists of the data provided by the servers (e.g. Bluetooth GATT[2] server) hosted on the wearable devices.

Today, Workflow Management Systems (WfMS) [17] have become a major component of service composition in IoT for managing the involved objects and their activities [2]. Specifically, WfMS support IoT systems with self-management mechanisms in numerous IoT applications including but not limited to smart home systems, crowd computing, wireless sensor networks, heating, ventilating, and air conditioning (HVAC) of smart buildings, mobile healthcare and so forth [2].

Meanwhile, the emergent Fog and Edge Computing (FEC) paradigms for IoT [4] emphasise the need of decentralised processes in which the IoT system should distribute certain tasks to the edge network or near the edge network where the front-end IoT devices or clients are located. Specifically, by applying the FEC paradigms, the IoT system can reduce the latency, which relying on distant central servers for data processing and decision-making introduces. In contrast to the conventional centralised architecture, FEC-based IoT systems can react to events much faster, thereby improving the reliability of time-critical applications such as remote healthcare, smart traffic control, disaster recovery etc. Correspondingly, the process management at the edge of IoT leads to a new research theme—Edge Process Management (EPM), which focuses on the design, development, deployment and runtime optimisation of the edge network objects and their activities through WfMS.

EPM systems are promising in supporting self-management at the edge network. However, they also face many mobile and wireless network-related challenges described in the literature study [2]. In particular, dynamic runtime context factors of the IoMT-involved mobile node, such as system workload, signal strength, movement & trajectory can affect task execution performance. For example, consider an IoMT node which needs to execute a task involving a proximity-based wireless fog server. If the IoMT node begins executing the task while the IoMT node is about to leave the coverage area of the fog server, the task can fail and the IoMT node will need to execute it again when it encounters the next fog server. Explicitly, the IoMT node has wasted its resources (e.g. energy) because of the failed task execution or poor signal quality [6], thus raising the question:

ECSA '18, September 24–28, 2018, Madrid, Spain
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6483-6/18/09…$15.00
https://doi.org/10.1145/3241403.3241449

[1]e.g. IETF RFC 7252 CoAP server or HTTP server
[2]https://www.bluetooth.com/specifications/gatt

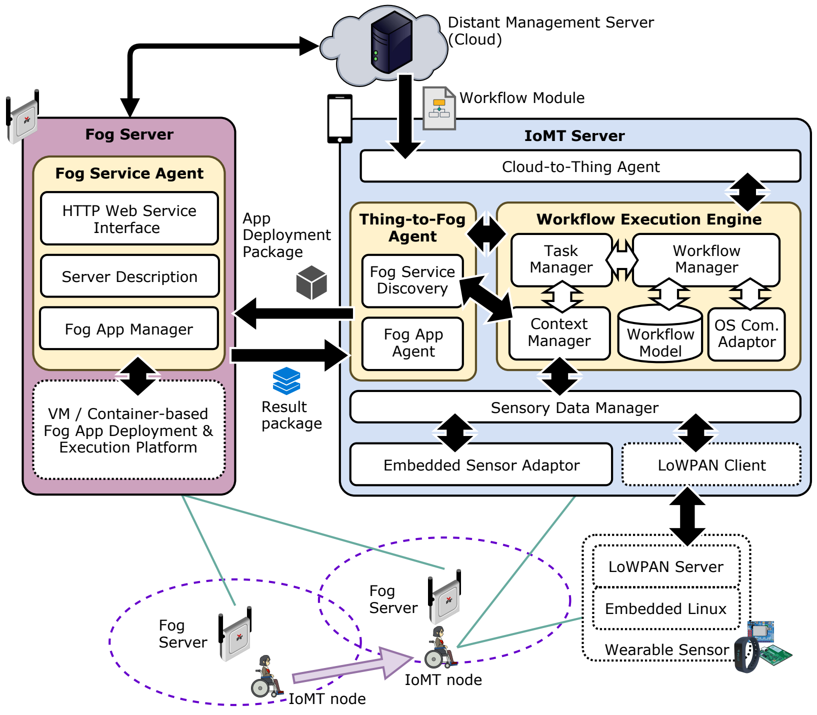


Figure 1: System architecture of the adaptive Mobile Thing-to-Fog EPM system.

How can EPM systems support task execution adaptively in the Mobile Thing-to-Fog environment?

In order to answer the question, in this paper, we propose an adaptive Mobile Thing-to-Fog EPM system which manages its workflow task execution schedules based on runtime context factors.

This paper is organised as follows. Section 2 describes the proposed system architecture and the corresponding components. Section 3 introduces the proposed adaptive workflow task execution scheduling scheme. Section 4 elaborates the experiments of the prototype. Section 5 summarises the related works. This paper is concluded in Section 6 together with future research direction.

2 SYSTEM DESIGN

In this section, we describe the proposed adaptive Mobile Thing-to-Fog EPM system. The system consists of two main edge entities—IoMT server and Fog server.

2.1 IoMT Server

A typical IoMT node consists of two parts— the IoMT server and the co-located sensors such as the wearable sensors attached to the patient in healthcare applications (see Figure 1). In general, the IoMT server contains a Low-power Wireless Personal Area Network (LoWPAN) client-side software which is capable of interacting with the co-located sensors via their server-side interface. For example, the LoWPAN server hosted on the wearable device can provide Bluetooth GATT services that allow the IoMT server to access the sensory data.

The main mechanism of the proposed IoMT server is the capability of executing workflow models sent from a distant management server. This is enabled by the workflow execution engine of the IoMT server. In contrast to the common over-the-air programming in Wireless Sensor Network (WSN) applications [13], which deploy program binaries to the device, workflow model-based deployment only sends the model (e.g. BPMN[3] file) and scripts (e.g. JavaScript) to the device and utilise the inbuilt functions of the workflow execution engine to interact with other internal components of the device (e.g. the API to access the sensor data) to accomplish the workflow tasks.

Below, we describe each component of the IoMT server.

Cloud-to-Thing Agent is the component enabling the communication between the distant management server (i.e. the cloud) and the IoMT server. Specifically, it provides a HTTP Web service interface that allows the cloud to send workflow models to the IoMT server for execution.

Thing-to-Fog (T2F) Agent is responsible for performing a fog server discovery mechanism in the background. In detail, the T2F agent utilises pre-configured functions to browse the proximal area. If the T2F agent discovers a fog server, it will request the fog server for its server description, which contains detailed information about the hardware specification (e.g. CPU, RAM, I/O, network capability etc.), the workload, available fog application (app) deployment platforms and so forth. Further, the T2F agent has a *Fog App Agent* component, which is responsible for packaging request messages (e.g. a computational task that involves scripts and data objects) to an executable fog application and sending the package to the fog server. Afterwards, the T2F agent also handles the response

[3]http://www.bpmn.org/

message from the fog server and converts the response message to the proper format for the Workflow Execution Engine.

Workflow Execution Engine (the engine) is responsible for executing workflow models sent from the cloud. In particular, the engine contains the following sub-components:

- **Workflow Manager** is the core component of the engine, which is capable of (1) utilising the *OS component adaptor* to access the underlying functions of the host OS (e.g. CPU, RAM, battery information of the device); (2) maintaining the workflow model database; (3) creating and assigning Task Managers to each workflow task.
- **Task Manager** manages each instance of a single workflow task. In general, the Task Manager controls the execution schedule of a task based on the run-time context factors, derived from the information provided by the Context Manager. In certain cases, Task Manager may decide to pause the task execution until the best condition occurs. This decision making is based on the proposed adaptive task execution scheduling scheme described in the next section.
- **Context Manager** is a component that interprets raw sensory data (derived from the Sensory Data Manager) and the fog server information (from the Fog Service Discovery component) to a certain specific information such as "the current communication quality level between IoMT node and the fog server is 'very poor', 'poor', 'acceptable', 'good', 'very good' or 'excellent'". Further, the context manager involves real-time information about the user's movement trajectory, which is then used to compare it against fog server locations. To clarify, such information is commonly subjective depending on the administrator's settings. Note that the detail of context interpretation is out of the scope of this paper, we consider it as a future work. In this paper, we use the example of signal strength and user mobility to demonstrate the context factor.

Sensory Data Manager is responsible for managing sensory data derived from the LoWPAN Client and the Embedded Sensor Adaptor, which retrieves data from the embedded sensor (e.g. GPS, Gyro, Accelerometer etc.) of the IoMT device.

2.2 Fog Server

The fog server contains HTTP Web service interface that enables the communication between itself and the requester (e.g. IoMT node). Further, as mentioned previously, fog server provides its information to its clients. Such information is described in a metadata message, which includes the hardware specification of the fog node and the up-to-date workload and states related to its computational and networking performance. Commonly, the fog server expects to receive fog application package in two possible forms: (1) a lightweight Virtual Machine (VM) image, which the fog server can deploy directly without any modification; (2) a container descriptor package (e.g. package containing Docker[4] app description + additional files) . Hence, if an IoMT node intends to perform a task on a fog server, the task needs to be converted to the corresponding package for deployment. Note that fog servers may have other approaches (see [3]) to provide application deployment, such as the script-based platform depending on the service providers' preference.

[4]https://www.docker.com/

3 ADAPTIVE TASK EXECUTION SCHEDULING SCHEME

In this section, we describe the proposed adaptive task execution scheduling scheme using formal Petri-Net model. The scheme is used by the IoMT servers *Task Manager* described in the previous section.

When modeling processes (workflows) using Petri Nets, the execution of a subtask within the process is represented as the *firing of a transition* in Petri Net terminology. Time Petri Nets associate a static 'time window' to each transition in the process model. This window dictates when it is valid for a given transition to fire (i.e. the task execution time). In this section, we describe our extension of the Time Petri Net concept to represent processes where these time windows are based on some external functions whose values change over time: for instance, values which reflect mobility-related context information.

3.1 Preliminary

We use the following conventional definition of a Time Petri Net, lending the notation from [11].

Definition 3.1. **Time Petri Net** A time Petri Net (TPN) is a tuple $\mathcal{Z} = (P, T, F, V, m_0, I)$ where

(1) $(P, T, F, V, m_0) := S(\mathcal{Z})$ is a classic Petri Net with:
 P - the set of places,
 T - the set of transitions,
 F - flow relation with $F \subseteq (P \times T) \cup (T \times P)$,
 $V : F \longrightarrow \mathbb{N}^+$ - is the weight of the elements of F (weight of the arcs),
 m_0 is a total function $m_0 : P \longrightarrow \mathbb{N}$ - called initial marking
(2) $I : T \longrightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$, for each $t \in T$, given $I(t) = (I_1(t), I_2(t))$, it holds that $I_1(t) \leq I_2(t)$,

This above definition assigns an interval $I(t)$ to every transition t of the classic Petri Net S($\mathcal{Z}$). For a given transition t, $I_1(t)$ is the *earliest firing time* of t *(eft(t))* and $I_2(t)$ is the *latest firing time* of t *(lft(t)).*

Definition 3.2. **state of a TPN** Let $\mathcal{Z}$ be a Time Petri Net, $\mathcal{Z} = (P, T, F, V, m_0, I)$. A state in $\mathcal{Z}$ is a pair $z := (m, h)$, such that

$m : P \longrightarrow \mathbb{N}$ is a marking of $\mathcal{Z}$
$h : T \longrightarrow \mathbb{R}_0^+ \cup \{\#\}$

and for every $t \in T$:

(1) $h(t) = \#$, if t is *not enabled* in marking m.
(2) $h(t) \in \mathbb{R}_0^+ \wedge h(t) \leq lft(t)$, if t is *enabled* in marking m.

Definition 3.3. A transition t in marking m of a Petri Net S is ***enabled***, if $t^- \leq m$, where t^- describes the no. of tokens removed from each place $p \in P$ upon firing of t :

$$t^-(p) = \begin{cases} V(p,t) & \text{if}(p,t) \in F \\ 0 & \text{if}(p,t) \notin F \end{cases}$$

Thus, the state of a TPN $\mathcal{Z}$ is described by a marking of its places and a function h which assigns a non-negative rational number (or infinity) to each transition. This h can be seen as the *"clock"* of enabled transitions. This clock starts running at the moment a transition becomes enabled. If a transition is not enabled, the value of h is #.

Given some state z in a TPN $\mathcal{Z}$, an enabled transition t in $\mathcal{Z}$ can *fire*, if $h(t) \geq eft(t)$. Then, the transition may be fired at any point in time where $h(t) \leq lft(t)$. The transition must fire no later than $lft(t)$, unless it has become disabled by another transition. Another way to write this possible firing interval is $[eft(t), lft(t)]$.

3.2 Extension

At the general level, the idea is to confine the firing times (earliest and latest firing time) such that they reflect certain context-based time-windows, for example, the availability of a radio signal when moving past a signal source (e.g. fog server).

Definition 3.4. **contextual time-bounding function** Let C denote a finite vector of arbitrary context variables. A function $X : C \longrightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$ is called the contextual time-bounding function. X maps a set of input context variable values to an interval, similar to the TPN I relation, for some $c \in C$, $X(c) = (X_1(c), X_2(c))$

Definition 3.5. **Adaptive TPN** An adaptive TPN (ATPN) $\mathcal{A}$ is a tuple $\mathcal{A} = (\mathcal{Z}, \mathcal{C}, \mathcal{X}, D)$, where

(1) $\mathcal{Z} = (P, T, F, V, m_0, I)$ is a Time Petri Net
(2) $\mathcal{C} : T \longrightarrow C$ maps context variable sets to transitions
(3) $\mathcal{X} : T \longrightarrow X$ maps contextual time-bounding functions to transitions
(4) $D : T \times \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\}) \longrightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\})$

D maps for each $t \in T$ a pair $D(t) = (d_1(t), d_2(t))$, such that:

$$d_1(t) = \begin{cases} I_1(t) & \text{if } eft(t) \geq X_1 \\ X_1^t & \text{otherwise} \end{cases}$$

$$d_2(t) = \begin{cases} I_2(t) & \text{if } lft(t) \leq X_2 \\ X_2^t & \text{otherwise} \end{cases}$$

The above definition introduces a second set of intervals mapped to transitions (D). It stands that for any $t \in T$, for the intervals $D(t)$ and $I(t)$, $D(t) \subseteq I(t)$.

We consider that the context variables C may change values over time, and thereby each time the C is updated, the respective values of D are also updated, hence the adaptive behaviour of the model.

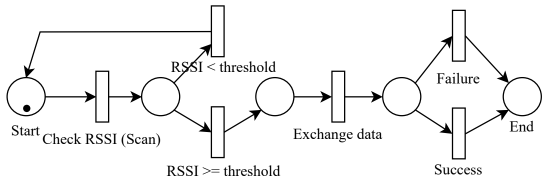
4 EXPERIMENTS

To demonstrate the effect of the proposed system and its adaptive task scheduling, we conducted a case study of scheduling wireless communication tasks. The set-up consists of a static fog server node and a IoMT node, whose moving trajectory passes by the fog server node. The goal is for the nodes to perform communication, however due to the mobility of the IoMT node, the efficiency of this communication is directly influenced by the scheduling of this task. A poor scheduling will start the task when the inter-node distance is large, thus causing poor wireless signal strength.

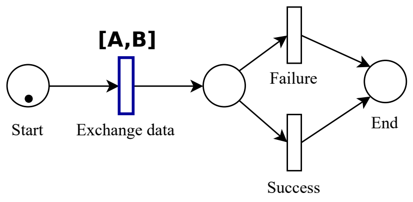
In our experimental study, we compare two approaches for task scheduling in the IoMT servers Task Manager. First, as the baseline, we consider a signal strength-based approach, where the communication is initiated based on a fixed signal strength (RSSI) threshold. The second approach is mobility-based, where communication is initiated based on movement trajectory prediction information and location of the static node. In the next subsections, we denote the signal strength threshold-based scheduling as *SST scheduling* and the scheduling used by the proposed scheme as *MOBI scheduling*.

4.1 Scheduling and Modelling differences between the approaches

We wish to emphasise the difference between these scheduling approaches in terms of support for temporal adaptiveness. In case of SST scheduling, modelling a system with Time Petri Nets using this mechanism would involve a loop to reach the task execution (*Exchange data*), as shown in Fig. 2(a). Here the Petri Net model is static, functions associated with the transitions (e.g. *Check RSSI; RSSI < threshold* on Fig. 2(a)) influence the workflow execution.



(a) SST Scheduling.



(b) MOBI Scheduling.

Figure 2: Modelling of the system with different scheduling approaches.

In contrast, modelling the *MOBI scheduling* approach with ATPN is shown in Fig. 2 (b). With ATPN, there is no need for a loop, as the *eft* and *lft*, (A and B on the figure respectively) of the *Exchange data* task are continuously updated. This also means that A and B may change throughout time, while for the *SST approach*, the change of system behaviour in time takes place by iterating the re-try loop for checking RSSI signals.

4.2 Technical configuration

In our practical experimentation, we used a *LG Nexus 5* smartphone as the IoMT node (Fig. 3, "M"), the fog server node consisted of a desktop PC (Fig. 3, "S") and a Wi-Fi router (Fig. 3, "R") with RJ-45 connection. When the task is scheduled, the IoMT node establishes a connection to the fog server node and performs distributed task execution with the fog server.

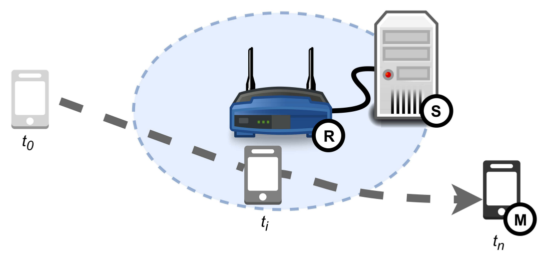


Figure 3: Experiment overview.

The communication protocol for the distributed task execution is based on HTTP. The HTTP server on the fog server is implemented in the Go language, while the IoMT node uses the *OkHttp*[5] library.

The experimentation took place in our office hallways. To better capture the effects of mobility on signal strength in this environment, we adjusted the signal gain (TX strength) on the router (Fig. 3 "R") to a minimum value (we used a value of 1 mW, with the default being 71 mW, possible values in the range 1 - 251 mW) using the open-source router firmware *dd-wrt*[6].

4.2.1 Scheduling implementation of mobility-based approach. A full implementation of the ATPN-based *MOBI scheduling* approach would use some mobility- and trajectory model and information about the routers position to adjust the *eft* and *lft* of the ATPN-s transition.

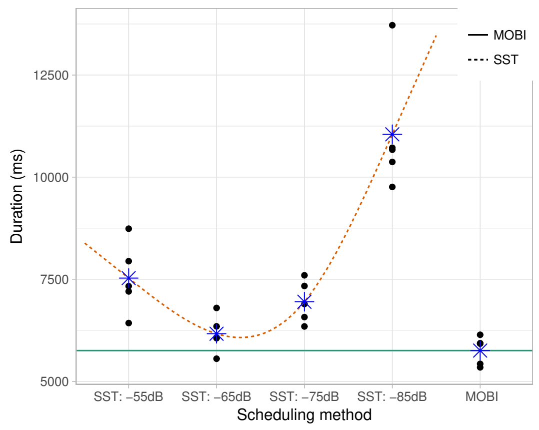
In our experiments, we use a placeholder implementation for this. Namely, our predictor module always assigns a value to the *eft* of the task such that the transition is fired when the smartphone is 1.5 meters away from the router.

4.3 Analysis of results

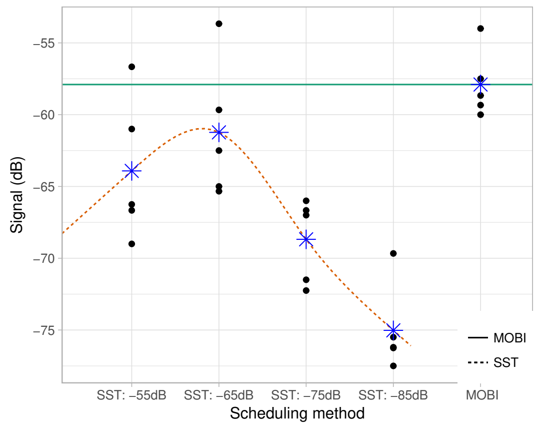
4.3.1 Description of recorded data . Using the above-described setup, we conducted a series of experiments for both approaches. A single experiment instance involved walking with the smartphone from one end of the hallway to the other, with the router located at the center of this trajectory. Note that in a single experiment, one single task is scheduled. More precisely, the task consists of: 1) connecting to the WiFi network, 2) downloading a fixed-size payload of random data.

During the experiment, we recorded networking-related metrics such as the RSSI throughout time, linkspeed, and timestamps of events such as *task scheduling start*, *connection successful* and *data exchange finished*. In case of *SST scheduling*, we experimented with various RSSI values as the threshold value to see the impact on performance.

4.3.2 Task execution time . First, we look at the duration of the task, a metric which directly impacts the Quality of Experience (QoE) of applications. Fig. 4(a) shows the total task execution time when downloading a 10 megabyte payload. The plot summarizes 4 different RSSI threshold configurations for *SST scheduling* and



(a) Total time of task.



(b) Average RSSI during download.

Figure 4: Performance with different scheduling configurations.

the result for *MOBI scheduling*. The experiment was recorded 5 times per configuration. The blue asterisks on the figures denote the mean value.

For *SST scheduling* approach, the task duration time generally improves as the RSSI threshold is increased, except for the -55dB case. It is expected to see the task duration improve with a stricter threshold, however, in the -55dB case, the reduced performance stems from the fact -55dB was close to the maximum experienced signal when near the router. Thus, the -55dB threshold causes the task to start only very near the router, meaning that throughout the communication itself, the mobile node is already distancing from the router.

On the other hand, *MOBI scheduling* outperforms the fastest option of the baseline in terms of task duration.

The trend of the performance of the *SST scheduling* approach has been plotted also on Fig. 4(a) with the dashed line, while the solid line on the figure shows the mean result of the mobility-based approach.

[5] http://square.github.io/okhttp/
[6] https://www.dd-wrt.com/

4.3.3 *Signal strength during task .*

Another key metric is the experienced RSSI during communication. From Fig. 4 (b) we can see a result similar to the execution times, where the baseline *SST scheduling* approach yields better results by constraining the threshold, up until a certain value (-55dB), where the threshold becomes too constraining. Meanwhile, the *MOBI scheduling* approach yields better signal strengths while also being more stable.

4.3.4 *Timeseries.*

To gain more intuition of the different configurations, Fig. 5 shows signal strength as a time series. Each sub-figure visualizes a single experiment instance, as opposed to Fig.4, which captured statistics over a series of experiments.

On sub-figure (a), it is visible that due to the low threshold bound of -85dB, the task is scheduled prematurely and communication takes place far before reaching the peak signal. While sub-figure (b) improves upon (a), the scheduling is still a bit off, the best signal is not experienced during the task. Finally, on sub-figure (c), with *MOBI scheduling*, we see that the peak signal falls into the period of task execution.

4.4 Discussion

Based on these results, we could see how a mobility-based scheduling approach can perform better than a simple threshold-based value. However, the mobility-based scheduling is again influenced by the precision of the prediction mechanism, which was out of scope for these experiments. As an example of a prediction mechanism, consider mobile navigation applications such as *Waze*[7].

We also saw, that given an appropriate threshold value, the *SST scheduling* can perform nearly as well as the mobility-based approach. Yet, finding the right threshold value can be a time-consuming task and is dependent upon the used hardware and environment.

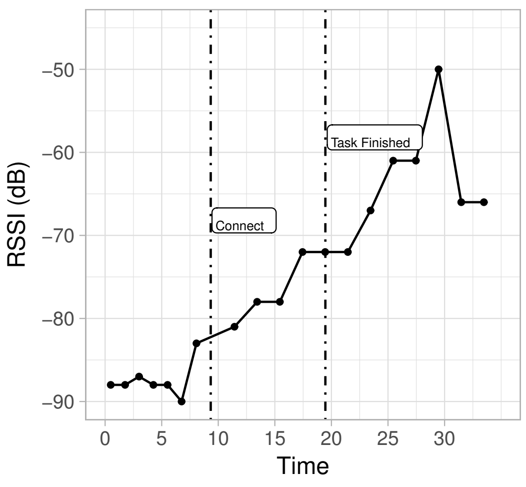
Another arising aspect is the sensitivity towards signal instability. The threshold-based scheduling may be triggered by brief periods of improved signal, which may occur even when the signal source is still far away, while this issue does not apply for the *MOBI scheduling* approach.

5 RELATED WORKS

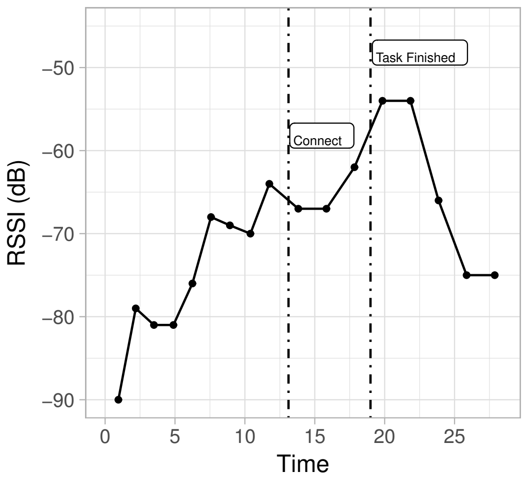
Process- & workflow modelling and management tools, including Petri Nets, have proven to be a significant approach for service composition in IoT [2]. Workflow modelling allows to abstract implementation aspects of the system while providing a formal structure which can be used to verify the model and its properties. Additionally, business process-aware data mining based on the execution logs becomes possible [14]. Various modelling technologies have been utilized in the context of IoT: BPEL[8], BPMN 2.0[9], Petri Nets, Node-RED[10] amongst others.

Scheduling of tasks impacts both energy usage and Quality of Service. Wang et al. [15] formulated the energy-efficiency problem via Integer Linear Programming and proposed heuristic algorithms to solve it. Their work addresses the scheduling problem in the context of Mobile Crowd-Sensing.

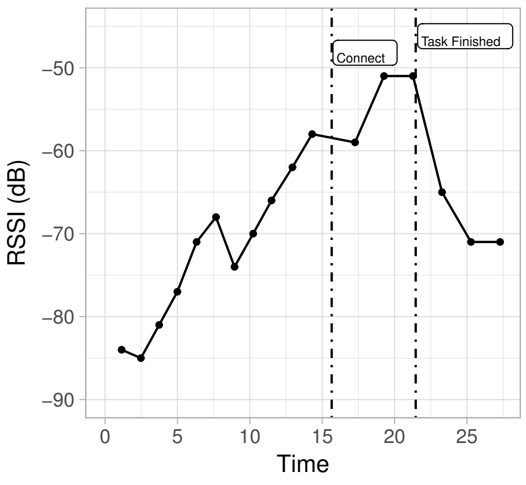
[7]https://www.waze.com
[8]http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[9]http://www.omg.org/spec/BPMN/2.0/
[10]https://nodered.org/



(a) TTS: -85dB.



(b) TTS: -65dB.



(c) MOBI.

Figure 5: Timeseries comparison.

Based on predictions about user mobility trajectory and processing tasks, authors in [7] pre-emptively fetch and process necessary data to reduce latency for spatio-temporal Event Processing in a mobile situation awareness system.

Scheduling of task offloading from mobiles to proximal Mobile Edge Computing servers is considered in [8]. The authors create a task scheduling policy based on Markov decision processes that takes into account power constraints and minimizes delay.

Similarly, Zeng et. al [16] consider the task scheduling between embedded devices and computation servers in the Fog context and use non-linear programming-based algorithm to minimize the task execution time. Like [8], the algorithm involves also deciding whether to execute locally or to offload.

A scheduling scheme for of wireless communication in Vehicle-to-Vehicle networks is proposed in [5], where the markov decision process is based on a stochastic game theory approach.

The authors in [9] proposed an adaptive mobile Complex Event Processing-based vehicular warning system where based on user location, the event processing, which is modelled using operator graphs, is dynamically updated to reduce latency and increase energy efficiency.

Another example of mobility-awareness is *marion* [12], where the agricultural harvesting process which involves cooperating machines is dynamically optimized. The system solves the planning problem of involved machine routes given the set of spatio-temporal constraints using robotics algorithms.

Mobility awareness involves performing geo-location fix acquisition operations. However, continuously tracking the location involves additional energy cost. In [10], the system tries to perform location acquisition tasks sparsely, at the appropriate moment to conserve energy. The acquisitions are scheduled based on user activity recognition, e.g. when the user’s transportation mode changes (from *driving car* to *on foot*).

Although previous works have proposed different scheduling models, they either did not integrate with WfMS or they require collecting historical data in order to design the scheduling. In contrast, the proposed system in this paper has integrated with WfMS and it utilises run-time context factor without additional historical data.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented an architecture design for mobile-based systems that interact in IoT scenarios through the concept of Edge Process Management. The system makes use of Adaptive Time Petri Net modelling to schedule tasks in the edge process, based on temporally changing contextual variables such as movement trajectory. We showed how such ATPN-based scheduling compared against a signal strength treshold-based approach when performing wireless communication tasks. The real-life experiment results indicated improved task execution speeds and higher RSSI values during task execution in the case of the ATPN approach.

As part of future work, we are interested in creating a prototype with established mobility prediction algorithms to drive the scheduling. Also, we are interested in exploring processes that account for other contextual information such as system load. Further, we are considering adapting the concepts of ATPN to BPMN-based process management tools such as the Camunda BPMN engine[11].

ACKNOWLEDGEMENTS

This work was supported by Telia Eesti AS and the Archimedes Foundation Smart specialisation scholarship.

REFERENCES

[1] Gianluca Aloi, Giuseppe Caliciuri, Giancarlo Fortino, Raffaele Gravina, P Pace, Wilma Russo, and Claudio Savaglio. 2017. Enabling IoT interoperability through opportunistic smartphone-based mobile gateways. *Journal of Network and Computer Applications* 81 (2017), 74–84.

[2] Chii Chang, Satish Narayana Srirama, and Rajkumar Buyya. 2016. Mobile Cloud Business Process Management System for the Internet of Things: A Survey. *ACM Comput. Surv.* 49, 4, Article 70 (Dec. 2016), 42 pages.

[3] Chii Chang, Satish Narayana Srirama, and Rajkumar Buyya. 2017. Indie Fog: An Efficient Fog-Computing Infrastructure for the Internet of Things. *Computer* 50, 9 (2017), 92–98.

[4] Chii Chang, Satish Narayana Srirama, and Rajkumar Buyya. 2018. Internet of Things (IoT) and New Computing Paradigms. In *Fog and Edge Computing: Principles and Paradigms.* Wiley Press, New York, USA, Chapter 1. In Press.

[5] X. Chen, C. Wu, and M. Bennis. 2017. An Oblivious Game-Theoretic Approach for Wireless Scheduling in V2V Communications. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference.* 1–6.

[6] Ning Ding, Daniel Wagner, Xiaomeng Chen, Abhinav Pathak, Y. Charlie Hu, and Andrew Rice. 2013. Characterizing and Modeling the Impact of Wireless Signal Strength on Smartphone Battery Drain. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS ’13).* ACM, New York, NY, USA, 29–40. https://doi.org/10.1145/2465529.2466586

[7] Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. 2013. Opportunistic spatio-temporal event processing for mobile situation awareness. In *Proceedings of the 7th ACM international conference on Distributed event-based systems.* ACM, 195–206.

[8] Juan Liu, Yuyi Mao, Jun Zhang, and Khaled B Letaief. 2016. Delay-optimal computation task scheduling for mobile-edge computing systems. In *Information Theory (ISIT), 2016 IEEE International Symposium on.* IEEE, 1451–1455.

[9] Beate Ottenwälder, Boris Koldehofe, Kurt Rothermel, Kirak Hong, David Lillethun, and Umakishore Ramachandran. 2014. MCEP: a mobility-aware complex event processing system. *ACM Transactions on Internet Technology* 14, 1 (2014), 6.

[10] Thomas Phan. 2013. Intelligent energy-efficient triggering of geolocation fix acquisitions based on transitions between activity recognition states. In *International Conference on Mobile Computing, Applications, and Services.* Springer, 104–121.

[11] Louchka Popova-Zeugmann. 2013. Time Petri Nets. In *Time and Petri Nets.* Springer. https://doi.org/10.1007/978-3-642-41115-1_3

[12] Stephan Scheuren, Stefan Stiene, Ronny Hartanto, Joachim Hertzberg, and Max Reinecke. 2013. Spatio-temporally constrained planning for cooperative vehicles in a harvesting scenario. *KI-Künstliche Intelligenz* 27, 4 (2013), 341–346.

[13] Thanos Stathopoulos, John Heidemann, and Deborah Estrin. 2003. *A remote code update mechanism for wireless sensor networks.* Technical Report. CALIFORNIA UNIV LOS ANGELES CENTER FOR EMBEDDED NETWORKED SENSING.

[14] Wil MP Van der Aalst. 2013. Business process management: a comprehensive survey. *ISRN Software Engineering* 2013 (2013).

[15] Jing Wang, Jian Tang, Guoliang Xue, and Dejun Yang. 2017. Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems. *Computer Networks* 115 (2017), 100–109.

[16] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu. 2016. Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System. *IEEE Trans. Comput.* 65, 12 (Dec 2016), 3702–3712. https://doi.org/10.1109/TC.2016.2536019

[17] Olaf Zukunft. 1997. Adaptation in mobile workflow management systems. *Personal Technologies* 1, 3 (1997), 197–202.

[11]https://camunda.com/