Adaptive Strategies for Patient Monitoring in Mobile Health Applications

Mathieu Bagot, Pascale Launay, Frédéric Guidec

Adaptive Strategies for Patient Monitoring in Mobile Health Applications

Mathieu Bagot*, Pascale Launay†, and Frederic Guidec‡
Université Bretagne Sud
Laboratoire IRISA
Vannes, France
Email: *mathieu.bagot@univ-ubs.fr, †pascale.launay@univ-ubs.fr, ‡frederic.guidec@univ-ubs.fr

IRISA, CominLabs, Université Bretagne Sud

***Abstract*—Remotely monitoring the health status of patients in all their activities of daily living is an interesting prospect. Yet an m-Health application devoted to patient monitoring should be able to deal with constrained and unstable operating conditions, such as limited power budget and disruption-prone network connectivity. This paper presents REGAS, a middleware system that makes it possible for a developer to define and enforce adaptive strategies, so their application can continuously adjust its behavior to changing operating conditions. Experimental results confirm that carefully designed strategies can have a significant impact on transmission delays, as well as on the power consumption of an m-Health monitoring application.**

***Index Terms*—mobile health, remote monitoring, context-aware application, middleware system, adaptive monitoring**

I. Introduction

The combination of wearable sensors and cellular networks offers interesting prospects for data acquisition and transmission while people are on the move. In a typical m-Health application, the health status of a heart failure or peripheral artery disease patient can be monitored continuously and remotely, while this patient remains free to live a "normal" life rather than stay at home or in a hospital ward. Yet developing fully operational solutions for m-Health monitoring remains a challenge, because all pathologies do not require the same kind of monitoring. A patient may for example have to wear a sensor providing full-featured ECG (thus producing a continuous flow of samples at a rate of several kBps), while another patient would only need to wear a heart rate meter (producing only a few bytes per second). It could be useful to track the location of a particularly fragile patient, while location tracking would be totally useless —and even overly intrusive— for another patient.

In order to meet the needs of different pathologies, an m-Health monitoring system must therefore be modular, so the wearable sensors can be selected according to the specific condition of each patient, and so the characteristics of the data samples produced by these sensors can also be taken into account at application level. A monitoring system meant to be used while on the move should also account for the variability of the available resources, such as power budget and network connectivity. Indeed, monitoring the health status of a mobile patient requires using battery-powered wearable devices (or energy harvesting devices), so power consumption must be considered with great care in order to prevent unexpected failures of the monitoring system. As for network connectivity, it is often a highly volatile resource when a patient is moving around. On certain occasions a patient may be located in an area covered by several kinds of cellular networks simultaneously (e.g., Wi-Fi, GPRS, EDGE... up to 5G), but on other occasions the same patient may visit a "white area" where no network service is available whatsoever.

In order to deal with such varying constraints, an m-Health monitoring system should be able to change its behavior automatically based on its current conditions of operation, switching seamlessly from one operation mode to another depending on variations observed in the available resources (e.g., battery level, network connectivity), or depending on varying requirements of the application itself (e.g., level of urgency of the data to be transmitted).

In this paper we present REGAS (REsource manaGement with Adaptive Strategies), a context-aware middleware system that has been designed specifically to facilitate the development of adaptive applications, especially those devoted to data acquisition and transmission in resource-constrained environments. This middleware system has been developed in the framework of project SHERPAM[1], whose focus is on m-Health monitoring. Yet REGAS is quite flexible, so it could also be used for other purposes, such as adaptive data collection in the IoT or in VANETs.

With REGAS a developer does not have to implement adaptive strategies directly in the source code of an application, which would make this source code overly complex and hard to maintain. Instead these strategies can be externalized from the application, and be enforced at middleware level. This approach fosters a clear decoupling between the core of the application (i.e., what it does) and its adaptive behavior (i.e., how it does it).

The remainder of this paper is organized as follows. Related work is discussed in Section II. In Section III we present the m-Health monitoring application developed in project SHERPAM as a typical use case, highlighting the need for such an application to adapt its behavior continuously in order to account for the state of the resources available at runtime. In

[1]https://sherpam.cominlabs.u-bretagneloire.fr

Section IV we propose an overview of the REGAS middleware system, and show how this system can help devise and support adaptive strategies in an m-Health monitoring application. In Section V we present experimental and emulation results that confirm the advantage of allowing such an application to be adaptive. Section VI concludes this paper, and proposes directions for future work.

II. Related work

Adaptive context-aware applications are applications that can adapt their behavior automatically according to changes observed in their context of operation. In [1] the context is defined as any piece of information that can be used to characterize the situation of an entity (i.e., a person, a place or an object). Based on this definition, context-awareness consists in using contextual information to optimize a system and/or enhance the user experience in a continuous process as the context changes.

A survey on previous work in Ambient Assisted Living (AAL) is available in [2], covering topics such as e-Health in smart homes [3] and behavioural change detection [4]. Most of the works in AAL propose a user-centric adaptation, as they aim at exploiting high-level contextual information such as a patient's usual activities in order to detect anomalies, or to trigger changes in the services offered to that patient. Since AAL is primarily focused on assisting patients at home, most AAL solutions do not address the problem of monitoring patients when they leave their home. Besides, the problem of dealing with resource-constrained wearable devices is usually not considered as being a prime concern in AAL solutions.

In contrast, m-Health applications typically aim at providing e-Health services to mobile patients [5]. Each m-Health project usually focuses on a particular use case, and a dedicated application is developed for this use case. A user-centric approach is often proposed to personalize the application to the patient's profile, as proposed in [6]. Contextual information can also be used to send feedback to the patient, to a caregiver, or to the medical staff when an anomaly is detected. The detection of anomalies can be performed directly on the device worn by the patient via embedded data processing [7], [8], [9], or data can be transferred to the cloud first, where important computing resources can be mobilized for data analysis [10].

For practical reasons the main device worn by a patient is often a smartphone, which serves as a gateway between sensors (also worn by the patient) and remote servers dedicated to data storage and processing. Wi-Fi and mobile data networks can be used for data transmissions to these servers. The management of this smartphone's limited resources — especially the battery— is usually not considered, and most proposals depend entirely on the default network manager implemented in the operating system of the device to drive the transmission of data.

The use of heteregeneous networks for data transmission is investigated in [11], as well as in Release 15 [12] of the 5G specification. This release considers three main features for mobile health applications:

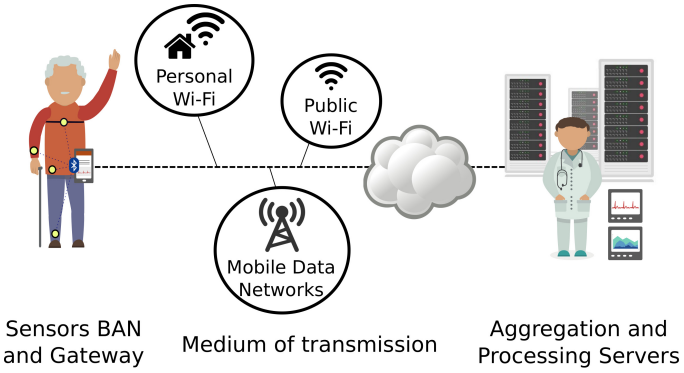

Figure 1. Illustration of the SHERPAM system

- the promise of better network coverage, and higher transmission throughputs.
- an improvement of the handover management based on the QoS required by the applications. For mobile health applications, this QoS is characterized by the need of low latency, and in the case of biosignal acquisition, high uplink throughput.
- the introduction of Mobile Edge Computing that offers interesting prospects to offload some computing processes closer to the patient.

These consideration mostly concern the network infrastructure, while resource management of wearable devices, and most specifically energy-efficiency, are prime concerns when dealing with constrained devices, as observed in [13].

III. Use case

In the remainder of this paper, we will consider the m-Health application developed in the framework of project SHERPAM as a typical m-Health use case. This application is meant to allow the continuous monitoring of cardiopathic and arteriopathic patients during all kinds of indoor and outdoor activities, including sport practise. Wearable sensors are used to acquire data pertaining to a patient's health status (acceleration, heart rate or full-featured ECG, ambient and body, temperature, location, etc.). The data samples produced by these sensors are collected wirelessly (mostly via Bluetooth transmission) by an Android smartphone, which serves as a gateway and is responsible for transmitting the data further toward a remote site where they will be processed (see Figure. 1).

The general objective is that each data sample acquired from a sensor should be processed in "as close to real-time as possible", while ensuring a reasonable autonomy (typically one day) of the monitoring system. A tradeoff must therefore be maintained between sending each data sample to the remote site immediately after it has been produced by the sensor, and differing this transmission for a while in order to preserve the resources involved in data transmissions. Indeed, every developer of mobile applications is aware that maintaining active transmission channels between a smartphone and a remote server is the best way to deplete the smartphone's

battery in a very short time. In contrast, opening transmission channels every now and then to perform short transmission bursts makes it possible to increase the smartphone's autonomy significantly. Android's synchronization mechanism is actually based on that idea, but since REGAS is meant to be portable over several kinds of platforms it cannot simply rely on Android's features. Besides, implementing transmission strategies directly in REGAS makes it possible to define and enforce advanced strategies that could hardly be driven by Android's synchronization mechanism.

In project SHERPAM the data samples produced by the sensors are collected by the gateway (i.e., smartphone), which aggregates these samples in bundles. A bundle is basically a collection of data samples collected over a certain time interval, together with meta-information that characterizes these data (e.g., timestamps, type and format of the data, etc.). Bundles serve as basic transmission units between the gateway and the remote server to which the data should eventually be delivered. A bundle can be stored in the gateway for a while, until an opportunity occurs to upload this bundle to the server. In order to preserve the power budget of the smartphone — thus ensuring the maximum autonomy— the SHERPAM system deliberately switches the Wi-Fi and cellular radio interfaces off whenever possible, and it only switches either of them on when data bundles must be uploaded to the server. Yet, network connectivity (i.e., the availability of a usable Wi-Fi access point or cell tower) is not guaranteed when a radio interface is switched on, so backoff strategies must be enforced in such circumstances.

IV. OVERVIEW OF THE REGAS MIDDLEWARE SYSTEM

REGAS (REsource manaGer using Adaptive Strategies) is a context-aware middleware system that is meant to facilitate the development of context-aware applications, especially those running in resource-constrained environments. REGAS focuses on the management of the data transmission by externalizing adaptive strategies at middleware level instead of implementing them directly in the source code of an application.

The architecture of the REGAS system is presented in Fig. 2. This architecture is roughly similar to the classical architecture of context-aware applications presented in [14]. It includes five main components, which are discussed below.

A. Probes

With REGAS, the whole system is perceived as a set of resources, whose capacity and availability can change over time. Probes are used to gather information about theses resources, whether they are part of the underlying operating system, of the application itself, or of the surrounding environment.

Probes observing the operating system can for example return information about the current amount of available memory, about the CPU load, or about the state of each network interface. When the application must run on a battery-powered host, an important resource is the battery itself, whose maximum capacity is limited, and whose current capacity tends to decrease over time. A battery probe can

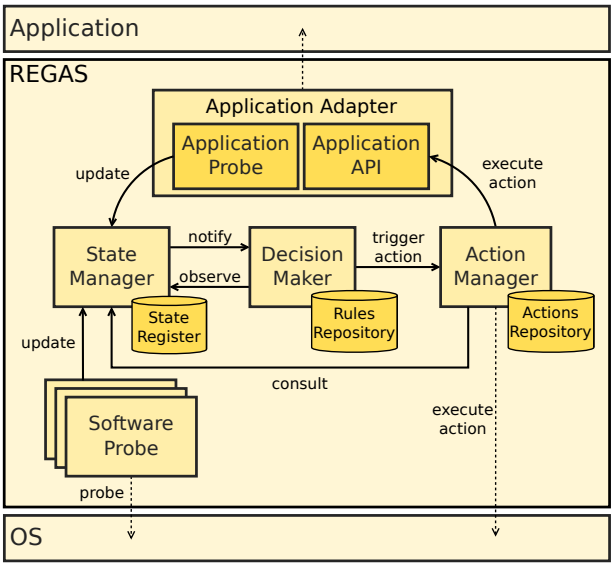

Figure 2. Overall architecture of the REGAS system

therefore provide information about the battery's maximal capacity, about its current level, and about its charging status *(empty/charging/charged/discharging)*.

Probes capable of observing an application must usually be developed specifically for this application, in order to signal any change in its configuration parameters or operating mode. Adaptive strategies enforced by REGAS can thus account for the actual state and needs of an application, rather than considering this application as a black box.

The environment includes any element that is not part of the core of the system. This may include sensors or actuators, networks, or even users. Dedicated probes are therefore required to collect information about such external elements whenever required. When the application must run on a mobile host, for example, the available wireless connectivity is likely to change over time. Dedicated probes can return information about the surrounding Wi-Fi access points or about cell towers.

B. State Manager

The State Manager contains the contextual information adaptive strategies can be based on. This contextual information is mostly provided by the system's probes, but it also pertains to internal elements that are used by REGAS, such as internal timers.

Information maintained in the State Manager is stored in a registry, which is organized as a *(key, value)* map. The maximal capacity of a 3000 mAh battery may for example be stored in this map as *(batt_cap, 3000)*, the current battery level as *(batt_level, 56%)*, and the current charging status as *(batt_charging_status, charged)*.

C. Decision Maker

The adaptive behavior of the system is defined as a set of rules, which must be enforced by the Decision Maker. Whenever a change is observed in the State Manager, this

change is notified to the Decision Maker, which then parses its rules in order to determine if one or several or them are addressed by this change.

Each rule is described as a *(trigger, condition, action)* tuple. The *trigger* field is meant to match a key in the State Manager's registry. Whenever the value associated with this key is altered in the registry, the rule is examined by the Decision Maker.

The *condition* field defines the logical conditions for the action to be executed. These conditions can relate to any entry or combination of entries in the registry. The *action* field refers to an action that can be executed by the Action Manager, upon request by the Decision Maker.

As an illustration, consider the following rule:

```
(batt_level,
 batt_level < 20
   AND batt_charging_status == discharging
   AND cell_tx_enabled == true,
 disable_cell_transmissions)
```

This rule is triggered (and thus examined) whenever the value associated with key *batt_level* changes in the registry. Yet the condition field specifies that the rule matches only if the battery level is below 20%, is still decreasing (as the battery is discharging), and if cellular transmissions are still allowed. In that case, function *disable_cell_transmissions* is called upon the Action Manager. Note that the action does not simply consist in changing the value of key *cell_tx_enabled* in the registry, as this would have no effect whatsoever on the current state of wireless interfaces and transmissions. The Action Manager is therefore required to enforce a decision made by the Decision Manager, and enforcing this decision will usually result in new changes in the registry.

D. Action Manager

The Action Manager provides a set of actions that can be executed upon the decision of the Decision Manager. These actions can involve sending commands to the operating system (e.g., enabling or disabling a wireless interface) or to the application via an Application Adapter (e.g., switching to a different operating mode). In some cases, the execution of an action can directly lead to changes in the State Manager's registry. This is for example the case when an action consists in starting a timer, which in turn will trigger another action when the timer completes. In such a case, information must be entered in the registry in order to maintain information about this timer.

E. Application Adapter

The Application Adapter is an optional software module that can be required to allow REGAS to interact with a given application. It defines specific probes that make it possible for REGAS to perceive the application's state, and it provides an API so that functions defined in the application can be called by the Action Manager.

V. Experimentation results

A. Evaluation method

Running real-life experiments involving mobile devices is always a tedious task, and the results thus obtained can hardly compare because it is always difficult —if not entirely impossible— to replay the very same scenario several times. In order to show how REGAS can perform while enforcing different adaptive strategies, we have developed an emulation platform that makes it possible to replay a predefined scenario several times with different adaptive strategies, and to compare the results thus obtained in a fully controlled environment.

When running in real conditions, the REGAS system perceives its environment via real probes, that keep it informed about the state of the underlying platform. When running in emulation mode, REGAS interacts with virtual probes, which are part of the emulation platform, and which present the state of virtual resources whose evolution is driven according to a pre-defined scenario.

Power consumption: The emulation platform implements a power consumption model, whose parameters have been obtained by measuring the actual power consumption observed on a smartphone in real-life conditions. More specifically, we used a Moto G 4G smartphone to measure the impact of enabling or disabling the Wi-Fi and cellular radios, of using any of these radios for sending or receiving data (at different bitrates and using different modulation standards), of using the GPS receiver, of switching the display screen on and off, etc. The parameters thus obtained have been fed into our power consumption model, which allows the emulation platform to closely mimic the power consumption while running an experiment involving REGAS.

Network connectivity: The network connectivity observed by a smartphone when its owner is moving around changes continuously. Our emulation platform can replay a connectivity scenario as many times as needed, which makes it possible to observe how tiny changes in the adaptation rules enforced by REGAS can impact the resulting performance of the system. A connectivity scenario played by the emulation platform can be totally artificial, or it can be a scenario that has first been captured in real life by a user moving around with a smartphone (serving as a recorder) in their pocket.

The experimentation results presented in this paper have been obtained using such a real-life connectivity scenario. A volunteer has carried a smartphone over a timespan of 11 hours, and the available networks detected by the smartphone over this timespan have been recorded. Figure 3 shows the resulting timeline, using a HIGH-LOW presentation for each network technology. It can for example be observed that Wi-Fi connectivity is only available for 1h22 (12% of the time) during the scenario, at the beginning and end of the timespan (as the volunteer was at home at these moments), but only occasionally during the day (as the volunteer was visiting a public library, as well as places with accessible hotspots). LTE connectivity was observed for rather long periods of time, but with a long gap in the middle of the

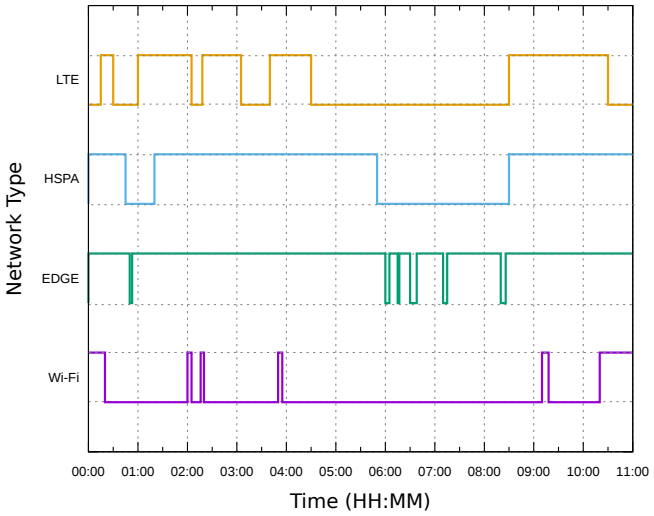

Figure 3. Network connectivity scenario

day. HSPA was also observed quite frequently, but with a similar —though shorter— gap in the middle of the day. As for EDGE connectivity, it was available most of the time, but with occasional connectivity disruptions that each lasted a few minutes (notably between 06:00 and 08:30). It is interesting to notice that during these disruptions no other kind of network connectivity was available, which means that the volunteer carrying the smartphone was actually traversing "white areas", where no transmission is possible, whatever the technology considered.

Several other connectivity scenarios have been recorded by volunteers, using the same approach. These scenarios constitute a valuable traceset with which different adaptive strategies can be examined. They have also been used to validate our power consumption model: each scenario has been replayed with REGAS in emulation mode in order to confirm that the power consumption simulated by the emulation platform is similar to that observed in the corresponding traceset.

Transmission modes: We conducted experiments in order to compare three different transmission modes, referred to as `bundle0`, `bundle1`, and `bundle5` in the remaining of this paper. With mode `bundle0` we assume that the data samples produced by sensors are not assembled in bundles in order to be sent in burst mode to the remote server. Instead they are sent immediately (if the network connectivity permits) to the server. This mode therefore promotes a continuous transmission of data to the server, which clearly yields the least latency, but which also yields higher power consumption. With mode `bundle1` the data samples acquired from sensors are assembled in bundles, each bundle containing the samples acquired over one minute. With mode `bundle5` a similar approach is used, but each bundle contains the samples acquired over five minutes.

Data production: During these experiments we conducted, we assumed that sensors associated with the smartphone produce data samples at a steady rate of 600 bytes per second (which is a realistic bitrate when an ECG sensor is used). These data samples can optionally be assembled in bundles upon being received by the smartphone. Moreover, we assume that high-priority data are produced during two short episodes during the 11 hour timespan considered in our experiments. The first episode occurs from 3h00 to 3h10, and the second episode occurs from 6h55 to 7h15.

Other parameters: The virtual smartphone modeled by the emulation platform is assumed to be powered by a 1500 mAh battery, and we assume that this battery is initially fully charged, and then keeps discharging until being completely depleted. The smartphone can use both Wi-Fi and cellular networking (more precisely 2.5G, 3G, and 4G transmission standards), with an unlimited data plan.

Adaptive strategies: Four adaptive strategies have been defined and implemented with REGAS, for the sake of demonstration. Before describing these strategies it is important to emphasize that our motivation is not to determine which of these strategies is the best one, since each strategy may actually be better suited for a particular combination of application and connectivity scenarios. In this paper our motivation is only to demonstrate that such strategies can actually be defined and enforced with REGAS, and that applying one or the other strategy may yield significant differences in terms of power consumption and transmission delays.

- `"Android-As-Usual" Strategy (AAUS):` this strategy is meant to serve as a baseline in our experiments. It mimics the default behavior observed in an Android platform, assuming that both the Wi-Fi and cellular radios are enabled continuously. In such a case Android's default strategy consists in prefering using Wi-Fi transmissions over cellular transmissions. Besides LTE transmissions are preferred over HSPA transmissions, which themselves are preferred over EDGE transmissions.
- `"Wi-Fi Only" Strategy (WFOS):` this strategy only attempts to use Wi-Fi transmissions to send data. The Wi-Fi radio is thus enabled continuously, while the cellular radio is turned off. Admittedly such a strategy may not be very appropriate with a connectivity scenario such as that shown in Figure 3, but it may prove useful when dealing with a very limited mobile data plan, or when the user mostly moves in an area where Wi-Fi connectivity is almost always guaranteed (e.g., at home, at work, or while moving in an area covered by many public hotspots).
- `"Wi-Fi Preferred" Strategy (WFPS):` with this strategy we assume that the data acquired from the sensors are time-sensitive, and must be sent to the server within a reasonable delay (about 3 minutes in the case considered here). However, since the power budget of the smartphone is considered to be a highly critical resource, we try to foster Wi-Fi transmissions over cellular transmissions. The Wi-Fi radio of the smartphone is therefore enabled continuously, while the cellular radio is disabled most of the time. The cellular

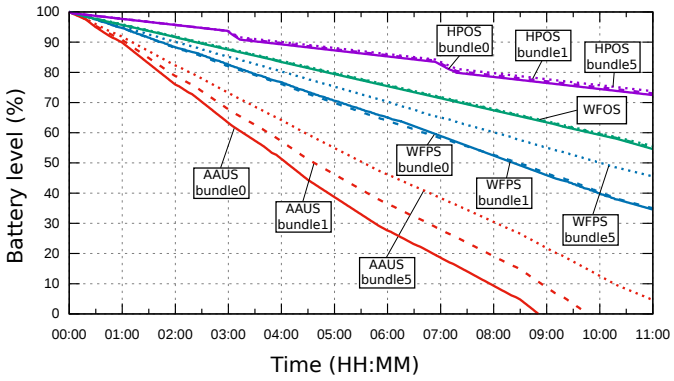

Figure 4. Comparison of the evolution of the battery level

radio is enabled only when no effective transmission to the server has been achieved over the last 3 minutes, or as soon as high-priority data are produced, as these must be sent urgently. The cellular radio remains enabled as long as high-priority data are still waiting to be sent to the server. It is disabled two minutes after the last transmission of high-priority data, and at least 30 seconds after being enabled.

- "`High-Priority Only` Strategy (`HPOS`): with this strategy we focus on high-priority data, and choose to send only these data to the server (assuming low-priority data are only meant to be processed locally by the smartphone). Both the Wi-Fi radio and the cellular radio are enabled as soon as high-priority data have been produced by the sensors, and they are disabled at least 30 seconds later, but only if no priority data are still waiting to be sent.

B. Results

Several emulations runs have be performed using REGAS and the emulation platform, considering each time a combination of one transmission mode (i.e., `bundle0`, `bundle1`, or `bundle5`) and one adaptive strategy (i.e., `AAUS`, `WFOS`, `WFPS`, `HPOS`). Figure 4 presents the evolution of the battery level for each emulation run. It can be observed that when running the `AAUS` strategy, the battery is fully depleted before the end of the 11 hour timespan, except when using the `bundle5` transmission mode. This is because this transmission mode, which consists in assembling data samples in bundles that each contain five minutes' worth of data, allows the radios to enter sleep mode, and to stay in this mode most of the time. In contrast when the radios are sollicited more frequently, the benefit of their built-in power saving mechanism is reduced.

Strategies `WFOS`, `WFPS`, `HPOS` all contribute to reduce significantly the power consumption observed on the smartphone. Of course strategy `HPOS` is the most frugal, as it makes very little use of the radios. Inflection points are actually clearly visible in the figure, as the radios are turned on and off again during the two episodes when high-priority data must be transmitted to the server.

With strategy `WFOS`, the advantage of assembling data samples in bundles (rather than sending then continuously) is negligible. This is because once a Wi-Fi radio is enabled, its power consumption is almost similar whatever the way this radio is used.

With strategy `WFPS`, it can be observed that transmission modes `bundle0` and `bundle1` produce similar results, while mode `bundle5` consumes less power. This is the consequence of the timeout used for enabling the cellular radio: whether the data samples are to be sent individually (`bundle0`) or in bundles containing one minute's worth of data (`bundle1`), the cellular radio will only be switched on after data samples have been available for 3 minutes, waiting to be sent to the server. In contrast, with mode `bundle5` there are times when there is no need to enable the cellular radio, because the next bundle to be sent is not ready yet. This example shows that when defining a strategy to be enforced by REGAS, it is important to make sure that this strategy is consistent with the way data are produced by the application.

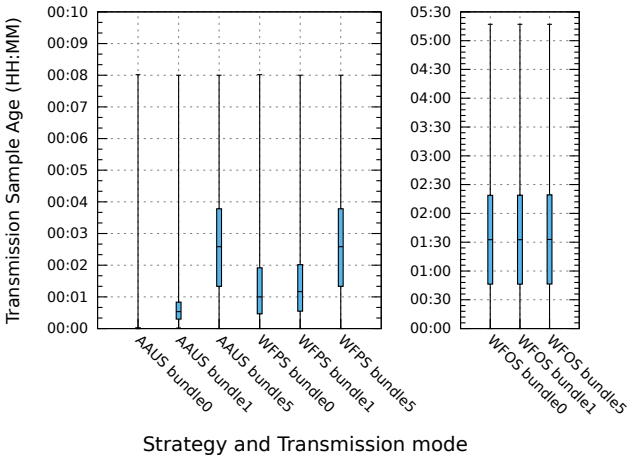

Figure 5. Comparison of the transmission data samples ages (TSA)

Several metrics have been computed while running these experiments. In Figure 5 we focus on the age of data samples when they are sent to the remote server. Each data sample is produced by a sensor, transmitted to the gateway, where it is inserted in a bundle, and then sent to the server when the bundle is completed and some connection is available. The transmission data sample age (TSA) represents the time interval between the data sample generation by a sensor and its emission to the server. Strategy `HPOS` is not considered in this figure, since it only concerns high-priority data. For strategy `AAUS`, the TSA represented in this figure does not include the period when the battery is fully depleted and no data can be transmitted anymore.

It can be observed that strategies `AAUS` and `WFPS` present similarities. Both have a maximum TSA value at 8 minutes. This is the longest disconnection time encountered in the scenario, which means that they have been able to send their data as soon as a network has been available. Also, both present similar TSA values for the transmission mode

bundle5. This transmission mode yields a higher standard deviation of TSA than the other modes, since the oldest data samples are already 5 minutes old when the bundles are completed. Strategy AAUS has mostly been able to send the bundles as soon as they were ready, and thus the TSA is only limited by the age of the data samples when the bundles are completed. Yet low latency comes at a price, which is a very high power consumption, as shown above. In fact strategy AAUS makes it possible to send data rapidly for a while as long as the battery is not fully depleted. In contrast strategy WFOS presents the highest TSA values, with a high standard deviation whatever transmission mode used. Again this is no surprise, for this strategy basically consists in waiting passively for Wi-Fi connectivity, which is very rare in the connectivity scenario considered in these experiments. Strategy WFPS lies in-between these two extrema. Although it does not provide the lowest TSA, it still allows to get an average TSA of 1 minute for transmission modes bundle0 and bundle1. The similar results for these two transmissions modes is explained by the choice of parameters in the strategy that enables the cellular interface if no data has been sent in the last 3 minutes, and keeps this interface enabled for at least 30 seconds, when no Wi-Fi transmission is possible, which is the case for 88% of the time in the scenario considered.

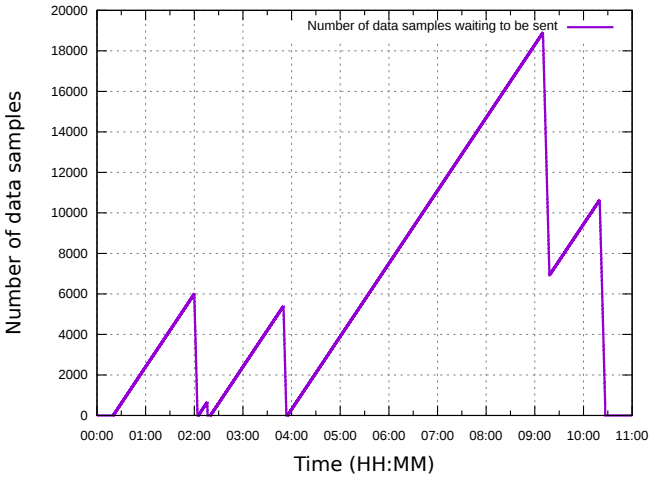

Figure 6. Evolution of the number of data samples waiting to be sent, using strategy WFOS and transmission mode bundle0

Fig 6 shows the amount of data waiting to be sent using strategy WFOS, assuming transmission mode bundle0 (i.e., continuous transmission). We can notice the accumulation of data samples during disconnection periods, with a maximum of 19.000 data samples waiting to be sent around 09:18. Moreover, at 09:20 a disconnection occurs while data samples are being transmitted, which causes the number of waiting data samples to start rising again. Yet at the end of the timespan the amount of waiting data samples is back to 0. This figure demonstrates the ability of strategy WFOS to deliver all the data samples to the server, provided Wi-Fi connectivity is available every now and then. But of course this strategy, which is very frugal as far as power is concerned, also yields high latency in a scenario with very little Wi-Fi connectivity such as that considered here.

In summary, we have observed that REGAS can be used to adapt the behavior of a device according to a set of rules, which can be combined in order to define various strategies. Such strategies can be used to manage critical resources at a fine grain, or to respond to specific application-level requirements.

VI. Conclusion and Future Work

In this paper, we have presented REGAS (REsource manaGer using Adaptive Strategies), a context-aware middleware system that makes it easy for a developer to define adaptive strategies, so their application can continuously adapt its behavior to changing operating conditions. REGAS is notably being used in the framework of project SHERPAM, in order to develop a smartphone-hosted m-Health application devoted to the monitoring of cardiopathic and arteriopathic patients.

Several basic strategies have been considered in this paper in order to demonstrate that REGAS can indeed enforce such strategies, and that the choice of one or the other strategy can have a significant impact on transmission delays and power consumption. More complex strategies should be devised and experimented in the near future. Meta-strategies may for example be defined by combining basic ones, and determining how and when to switch between them. Motion detection (based on the built-in GPS receiver or accelerometers) may be used to disable network scanning when the smartphone is stationary. Indeed, once REGAS has determined which surrounding networks —if any— are available for data transmission, there is no benefit to keep looking for other networks, unless the smartphone is moving. Machine learning may also be an interesting approach to let REGAS discover recurrent network connectivity patterns, so as to predict when and where the smartphone's radios should be switched on and off.

Acknowledgment

This work is part of the SHERPAM project (2014-2019). As such it has received a French government support granted to the COMIN Labs excellence laboratory, which is managed by the National Research Agency in the *"Investing for the Future"* program under reference ANR-10-LABX-07-01. Further information about this project can be found at http://www.sherpam.cominlabs.ueb.eu/.

References

[1] A. K. Dey, "Providing Architectural Support for Building Context-Aware Applications," Ph.D. dissertation, 2000.

[2] E. Zavala, X. Franch, and J. Marco, "Adaptive Monitoring: A Systematic Mapping," *Information and Software Technology*, vol. 105, pp. 161–189, January 2019.

[3] M. Alirezaie, J. Renoux, U. Köckemann, A. Kristoffersson, L. Karlsson, E. Blomqvist, N. Tsiftes, T. Voigt, and A. Loutfi, "An Ontology-Based Context-Aware System for Smart Homes: E-care@home," *Sensors*, vol. 17, no. 7, p. 1586, July 2017.

[4] A. R. M. Forkan, I. Khalil, Z. Tari, S. Foufou, and A. Bouras, "A Context-Aware Approach for Long-term Behavioural Change Detection and Abnormality Prediction in Ambient Assisted Living," *Pattern Recognition*, vol. 48, no. 3, pp. 628–641, March 2015.

[5] B. M. Silva, J. J. Rodrigues, I. de la Torre Díez, M. López-Coronado, and K. Saleem, "Mobile-Health: A Review of Current State in 2015," *Journal of Biomedical Informatics*, vol. 56, pp. 265–272, August 2015.

[6] M. Klein, A. Manzoor, and J. Mollee, "Active2Gether: A Personalized m-Health Intervention to Encourage Physical Activity," *Sensors*, vol. 17, no. 6, p. 1436, June 2017.

[7] P. Kakria, N. Tripathi, and P. Kitipawang, "A Real-Time Health Monitoring System for Remote Cardiac Patients using Smartphone and Wearable Sensors," *International journal of Telemedicine and Applications*, vol. 2015, p. 8, January 2015.

[8] M. Esposito, A. Minutolo, R. Megna, M. Forastiere, M. Magliulo, and G. De Pietro, "A Smart Mobile, Self-Configuring, Context-Aware Architecture for Personal Health Monitoring," *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 136–156, January 2018.

[9] I. Miralles and C. Granell, "Considerations for Designing Context-Aware Mobile Apps for Mental Health Interventions," *International Journal of Environmental Research and Public Health*, vol. 16, no. 7, p. 1197, April 2019.

[10] J. Santos, J. J. Rodrigues, B. M. Silva, J. Casal, K. Saleem, and V. Denisov, "An IoT-Based Mobile Gateway for Intelligent Personal Assistants on Mobile Health Environments," *Journal of Network and Computer Applications*, vol. 71, pp. 194–204, August 2016.

[11] D. Niyato, E. Hossain, and S. Camorlinga, "Remote Patient Monitoring Service using Heterogeneous Wireless Access Networks: Architecture and Optimization," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 4, pp. 412–423, May 2009.

[12] 3GPP, "3GPP specification : Release 15," https://www.3gpp.org/release-15, April 2019, [Online; accessed 22-July-2019].

[13] R. Mahapatra, Y. Nijsure, G. Kaddoum, N. U. Hassan, and C. Yuen, "Energy Efficiency Tradeoff Mechanism Towards Wireless Green Communication: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 686–705, October 2015.

[14] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A Survey of Context Data Distribution for Mobile Ubiquitous Systems," *ACM Computing Surveys (CSUR)*, vol. 44, no. 4, p. 24, August 2012.