Distributed Performance Management of Internet of Things as a Service for Caregivers

Minkeun Ha
School of Technology and Health
KTH Royal Institute of Technology, Stockholm, Sweden
Email: minkeun.ha@sth.kth.se

Thomas Lindh
School of Technology and Health
KTH Royal Institute of Technology, Stockholm, Sweden
Email: thomas.lindh@sth.kth.se

***Abstract*—This paper presents a multi-layer distributed performance management model for Internet of Things (IoT) as a service for caregivers. Performance monitoring and control of IoT devices and services are vital functions for e-health systems. Cloud-based IoT systems with centralized data centers further underline the need for distributed management. The main contribution of this paper is a model and prototype implementation that combines low-complexity policy-based and conventional network management with cloud computing paradigm, lightweight IoT protocols, and their data models. A case study on monitoring of heart activity demonstrates interworking between OMA Lightweight M2M protocol, Bluetooth low energy GATT services, and Azure IoT hub cloud platform to enable distributed policy-based performance management.**

I. Introduction

Information from wearable and ambient wireless sensors has become increasingly important in e-health systems. Internet of Things (IoT) is today offered as a service for caregivers where the IT backend systems can be cloud-based or traditional servers. There are several initiatives to merge and standardize communication protocols, services and data models for M2M and IoT systems [1]. Some examples are oneM2M, Open Connectivity Foundation (OCF/IoTivity), and Lightweight Machine-to-Machine (LWM2M) from Open Mobile Alliance (OMA). OneM2M is an effort from ETSI and other standards organizations to define a worldwide standard for machine-to-machine communication and IoT [2]. It is supported by industrial consortia e.g. Personal Connected Health Alliance and Home Gateway Initiative. OneM2M is a full and extensive service layer technology, whereas LWM2M is a less complex alternative that works well for many applications. The communication between LWM2M servers and clients (IoT devices) utilizes the Constrained Application Protocol (CoAP). This creates an alternative to HTTP for devices with limited resources to apply the RESTful API methods.

General models for IoT and cloud-based e-health systems have been outlined in several papers [3]-[10], as well as more specialized applications e.g. heart activity monitoring [11], [12]. Mobile edge, cloudlets and fog computing are proposed to offload main cloud data centers and distribute functions closer to network edges for demanding real-time applications with low response times [13], [14].

Healthcare applications, particularly IoT systems with wireless sensor devices, need to monitor whether data is received correctly and in time, and also be able to control and improve the performance levels. A large-scale distributed e-health sensor devices, however, are difficult to manage their performance in real time due to their decentralized and complex nature. Cloud-based services often share resources in large centralized data centers, which are expected to serve potentially thousands of sensor nodes. This further underlines the need for efficient and reliable distributed management. The main contribution of this paper is a model and prototype implementation of distributed performance management of IoT devices and services for caregivers. It combines traditional network management [15], low-complexity policy-based management [16], cloud computing methods, and lightweight protocols and data models for IoT communication.

A general architecture for IoT as a service (IoTaaS) for caregivers is outlined in Section II. A distributed model for management of IoT information systems is described in Section III. Examples of distributed performance policies implemented in the intermediate gateways are given in Section IV. A wearable ECG system is chosen as case study and application of the general model. Results of a prototype implementation of functions that realize performance policies are presented in Section V, and finally the conclusions in Section VI. The paper does not address systems management of the cloud components and data centers.

II. General Architecture of IoT as a Service (IoTaaS) for Caregivers

IoTaaS for caregivers may use a traditional backend server system. However, public or private cloud-based IT systems have today become an increasingly competitive solution. Fig. 1 shows a general architecture for cloud-based IoTaaS offered by a service provider to caregivers. Each caregiver has a number of patients, equipped with a personal gateway (e.g. a mobile phone or a separate modem) and a number of wearable health sensor devices. The links between sensor nodes and a gateway are normally wireless. The connection between a gateway and the cloud system is carried by the cellular mobile network, or a wireless local-area network, until it enters the fixed-line networks. The cloud-based servers implement functions for receiving and processing data from IoT devices such as protocol conversions, data filtering and analytics, storage, presentation and statistics, interfaces to healthcare patient

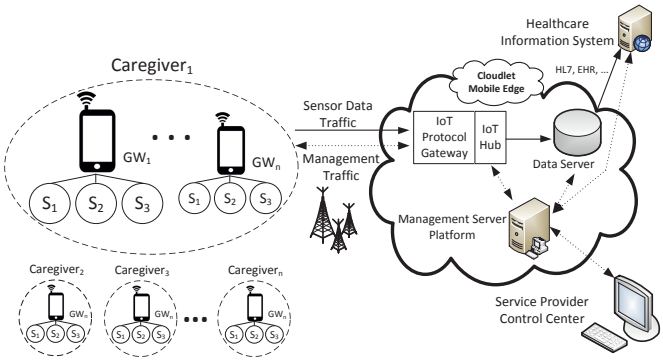

Fig. 1. General architecture of cloud-based IoTaaS supplied by a service provider to caregivers. (S_n = sensor node n, GW_n = gateway n.)

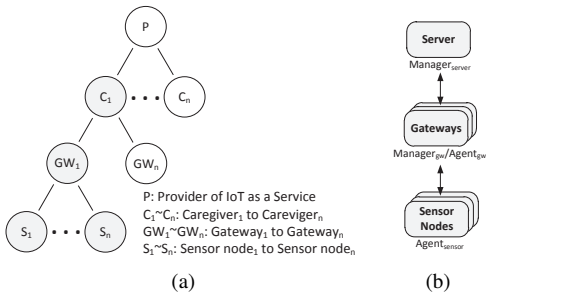

Fig. 2. Multi-layer management architecture. (a) A general multi-layer management architecture for IoTaaS for caregivers. In a cloud-based system two upper layers are parts of the cloud system. This paper is focused on the shaded entities in the figure. (b) The management roles in the sensor nodes, the gateways and the cloud-based management server for one of the caregivers in Fig. 2a. The gateway has a dual role.

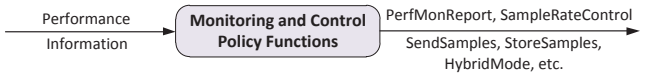

Fig. 3. Input information and output from the monitoring and control functions in gateways that implement the performance management policies.

information systems and caregivers, and also a management platform. The data traffic between sensor nodes, gateways and cloud-based servers is composed of two types: data samples from sensor nodes and gateways to the cloud, and bidirectional management traffic between the entities.

Recently, several network operators and IT companies offer IoT as a cloud-based service, e.g. IBM Bluemix, Cisco Jasper and Microsoft Azure IoT hub. The prototype implementation in Section V uses Microsoft Azure IoT hub[1].

III. General Model for Multi-layer Management of IoTaaS

A multi-layer management architecture for IoTaaS for caregivers is shown in Fig. 2a. A number of caregivers are connected to the service provider. Each of the caregivers has a number of patients, each with a gateway and sensor nodes. Fig. 2b shows the main components in the management system for one of the caregivers: agents in the sensor nodes ($Agent_{sensor}$), a central cloud-based management server ($Manager_{server}$), and gateways acting as intermediate managers ($Manager_{gw}$) as well as agents ($Agent_{gw}$). Intermediate multi-layer management architectures and protocols have been suggested e.g. in SNMP network management [15]. This model distributes functions that implement performance management policies from the central cloud-based management server to the gateways, which in turn monitor and control the sensor node agents. The model is not dependent on any specific communication protocol and data model. OMA Lightweight M2M (LWM2M), a feasible candidate, is applied in the prototype implementation in Section V. In general, the model requires that the gateways have sufficient communication, processing and storing capabilities. Some alternatives for wearable gateways are mobile phones or LTE modems (e.g. Cat M1 or NB-IoT[2]).

IV. Distributed Policy-based Performance Management Functions

A caregiver, or a service provider on its behalf, can deploy distributed policies that realize the performance monitoring and control goals. A policy is implemented as program code (functions and logic) that runs in the gateways. Our low-complexity policy-based management model is inspired by an event-condition-action approach [16]. Fig. 3 shows the input to, and output from, the unit in the gateway that implements the performance policies and functions. A policy is delegated by the cloud-based $Manager_{server}$ to the $Manager_{gw}$ in the gateways using a communication protocol, and related data model. Fig. 4 summarizes the messages between managers and agents in the system. Examples of performance policies are elaborated in Section IV and V: periodic performance reports, automatic sampling rate control and transmission modes.

A. Periodic Performance Monitoring Reports

The $Manager_{gw}$ measures and estimates key performance parameters: packet loss, delay variations (jitter) and throughput. The result is used as input to further control actions and for the performance report (*PerfMonReport*) to the $Manager_{server}$. The report indicates if the impairments are located on the link between sensor nodes and a gateway, between gateways and cloud-based systems, or on both links. Based on the report, the $Manager_{server}$ presents statistics and real-time performance information to end-users. A requirement is that the $Agent_{sensor}$ provides sufficient information (*PerfInfo*) like sequence numbers and timestamps along with the transmitted data samples. The *PerfMonReports* are sent with configurable time interval, and alarm notifications are issued without delay.

B. Automatic Sampling Rate Control Policy Function

A caregiver can apply a policy for automatic sampling rate control. The *SamplingRateControl* function enables the $Manager_{gw}$ to instruct the $Agent_{sensor}$ to reduce or increase the sampling frequency. The gateway may set the sampling

[1]MS Azure IoT hub, https://azure.microsoft.com/en-us/services/iot-hub/.
[2]Release-13, 3GPP, 2016.

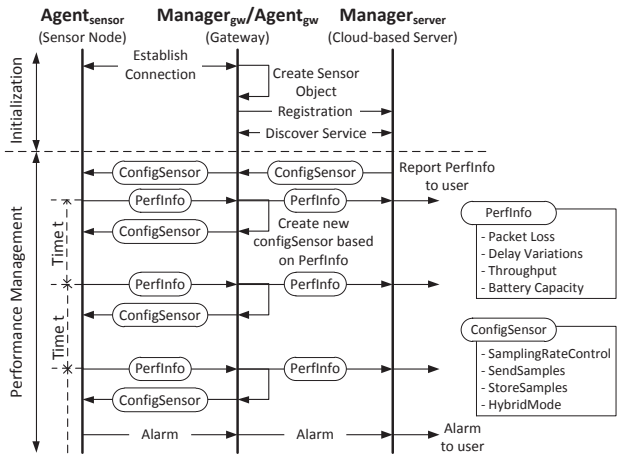

Fig. 4. A message diagram between managers and agents. These messages are mapped to LWM2M methods in the implementation in Section V.

rate to a predefined default frequency, or apply an automatic control algorithm that finds the maximum sampling rate for a specified threshold of packet loss ratio. The algorithm adapts the sensors data sampling rate between a maximum level and a guaranteed level so that the packet loss ratio is kept below the loss ratio threshold. Should the loss ratio exceed the threshold for the guaranteed sampling rate, the gateway then enables the transmission modes *StoreSamples* or *HybridMode* to store samples locally, reduce the packet rate and postpone transmission and more detailed analysis of all data samples (see below).

C. Transmission Mode Policy Functions

The function *SendSamples* instructs the sensor node to transmit data samples to the gateway. The $\text{Manager}_{\text{gw}}$ can configure the $\text{Agent}_{\text{sensor}}$ to transmit data samples according to a traffic pattern set by the caregiver and patient: continuous streaming, store and transmit with certain intervals, or event-driven transmission (e.g. ECG devices activated by patients). The *StoreSamples* function can be activated to store data samples locally during periods of performance deterioration. In case of impairments between the gateway and the cloud-based systems, the gateway stores samples locally; and, in case of impairments on the link between the gateway and the sensor node, the gateway will instruct the sensor node to store samples locally. The maximum storing time is determined by the size of the memory in the gateway and the sensor nodes, and the sampling rate. The hybrid transmission mode function (*HybridMode*) is an option for the caregiver to maintain a basic monitoring function during periods of performance deterioration and postpone more detailed analysis. If the packet loss ratio for the guaranteed sampling rate is above the threshold, the gateway then instructs the sensor node to store samples locally (*StoreSamples*) in its flash memory and to transmit data samples at a lower packet rate, for example that every second, third or fourth sample is transmitted. More details and pseudo-code for policies are given in Section V.

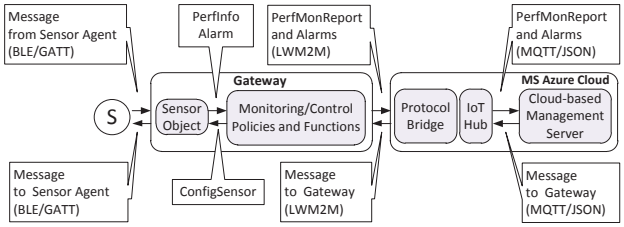

Fig. 5. Block diagram for messages and protocols between a BLE sensor agent (S in the left part), gateway manager and agent (middle), and a protocol bridge and management server in the Azure cloud system (right).

D. Message Diagram

The diagram in Fig. 4 summarizes messages between agents and managers. These messages in Fig. 4 are mapped to LWM2M methods to implement the policy-based management functions.

E. Block Diagram for Interworking between LWM2M, Bluetooth LE GATT Services, and Azure IoT Hub

The block diagram in Fig. 5 shows messages and protocols between a Bluetooth LE (BLE) sensor node, a gateway with a virtual sensor object and a monitoring and control unit, and a cloud-based protocol bridge and management server. The BLE sensor and gateway communicate via the virtual sensor object using GATT (Generic Attribute) services. The $\text{Manager}_{\text{server}}$ and the $\text{Manager}_{\text{gw}}$ retrieve information and perform control actions through the virtual sensor object. The output from the monitoring and control unit is based on performance data (*PerfInfo*) from the sensor agents. The protocol bridge translates between MQTT/JSON and LWM2M formats.

1) Simplified Policy-based Management Using LWM2M: The prototype described in Section V uses LWM2M to realize policy-based performance management with reduced complexity compared to previous frameworks and platforms [16], [18]. We have extended the LWM2M data model to configure the policy-based functions and to provide sensor data readings. Each of them is mapped to a new LWM2M resource. For example, an ECG sensor is assigned object id 1031, and its resource id 5 is an interface to enable or disable timestamping for each ECG data sample. When a gateway initiates bootstrap, the LWM2M cloud-based server sends the *ConfigSensor* message with parameters to the gateway using the LWM2M Write method. The gateway updates and configures its local resources and sends Write commands to remote sensors with received parameters such as sampling rate, traffic pattern, etc. A caregiver can activate transmission of sensor data by sending a LWM2M Observe message. The gateway lists the requested observations for the specific resources (e.g. ECG data) and registers periodic notifications with a pre-configured sampling rate to its sensor nodes. The sensor nodes will periodically notify the gateway with *PerfInfo*, which can be included in the data message as a performance management option field in order to minimize signaling overhead and battery usage.

2) The Gateways' Dual Role: The gateway has an intermediate dual role, firstly, as a $\text{Manager}_{\text{gw}}$ of the connected

sensors acting as a BLE GATT client and, secondly, as a LWM2M client ($Agent_{gw}$). The gateway collects data from the $Agent_{sensor}$ (BLE GATT server). Caregivers can control the traffic pattern, configure sampling rates, assign priorities to the sensors, set performance and alarm thresholds, performance thresholds, etc. through the backend $Manager_{server}$ applications. In addition, a caregiver can enable the sensor nodes to include performance information (*PerfInfo*), such as sequence numbers and timestamps, in the sensor data packets to the gateway. After powering on, a gateway firstly initializes its LWM2M client stack by registering its existence to the IoT Hub bridge. The bridge node translates and forwards the messages to the Azure IoT Hub cloud server. Upon a successful registration, it scans the nearby sensors and sets up BLE GATT connections to them. It creates LWM2M sensor objects and resources according to the result of GATT service discovery, and maps the GATT service characteristics onto the LWM2M object and resources. According to the configured performance policy-based functions (reports, sampling control etc.), the gateway initializes the sensor service and receives sensor data by registering a notification to the specific GATT characteristics, e.g. ECG measurements. The gateway aggregates and forwards the received data samples to the cloud system.

3) The Cloud System: The chosen cloud system, Microsoft Azure IoT Hub, generally supports MQTT but not natively the OMA LWM2M protocol, which has become a de facto standard for device management. A bridge is therefore needed to translate LWM2M messages to MQTT-based IoT Hub compatible messages or vice versa. It also maintains a map of the LWM2M devices and the IoT Hub device connection string. The cloud-based server collects, stores and analyses data, estimates the patients health condition, and visualizes data. The Azure IoT Hub collects telemetry healthcare data from distributed sensors (via gateways) and provides the current status of connected devices using its Device Twin feature. The Device Twins are JSON documents that store device state information (metadata, configurations and conditions). In addition, its analytics framework is used to visualize the collected data in the performance reports to caregivers. Lastly, we have implemented a backend application for caregivers. Using this application, the caregivers can configure, monitor and control the healthcare devices.

V. Case Study: Performance Management of Wearable ECG as a Service for Caregivers

There are several solutions for heart activity monitoring outside the clinic. Long-term Holter ECGs register the heart activity continuously and can either store data locally[3] or transmit the samples wirelessly[4]. A handheld ECG[5], often called thumb ECG, is an alternative where the patients themselves activate recording and transmission of data. According to [12], handheld intermittent short ECG recordings can be more efficient than a 24-hour ECG. Coala[6] has introduced a service that combines a handheld ECG and recording of the heart sound for cardiac analysis. A wireless device transmits ECG signals and heart sound via Bluetooth to a mobile phone and further to a cloud-based server. Interviews and discussions with cardiologists and medical staff suggest that additional information from complementary sensors such as GPS, pulse oximeter or IMU should be an option.

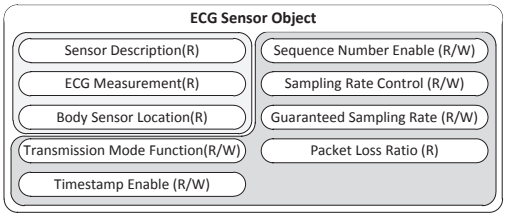

Fig. 6. The Extended ECG GATT Service Object.

A. Prototype Testbed

The testbed is designed in accordance with the block diagram in Fig. 5. It consists of an ECG sensor node, a gateway device, a LWM2M-IoTHub bridge, a cloud-based management server and backend applications. A BLE GATT-based healthcare sensor object, residing on Raspberry Pi 3 (RPi) Model B, has been developed. It can be extended to other network technologies as well, e.g. IEEE 802.15.4 and NB-IoT. The RPi has a 1.2 GHz 64-bit Quad-core ARMv8 Cortex A53 processor (Broadcom BCM2837), 1 GB RAM, one Ethernet port, 802.11n Wireless LAN, and BLE 4.1. The RPi operating system is a Linux-based distribution (Raspbian Jessie 2016-09-23 Release). A low-power wireless sensor application runs on top of Bluez 5.42, which is an open-source Bluetooth library officially accepted in Linux. Each sensor service is exposed to nearby clients, or a gateway device, as a GATT service. The prototype uses the Wakaama[7] implementation of the LWM2M clients and lwm2m-node-lib[8] for the LWM2M server.

B. Extended GATT Healthcare Services

The GATT healthcare services are collections of characteristics and relationships to other services that encapsulate the behavior of a device. In e-health, it is essential to monitor whether received sensor data is correct or not and received in time. Furthermore, caregivers should be able to analyze health data chronologically. However, the pre-defined standard GATT services do not provide ways to associate timestamps and sequence numbers with sensor data. Hence, extended characteristics for GATT healthcare sensor services have been designed and implemented to enable performance reports according to the standard for GATT services. The extensions provide interfaces to configure sensor data report methods, adjust sampling rates, and, enable/disable timestamps and sequence numbers. The extended characteristics can be

[3]Schiller, http://www.schiller.ch.
[4]Kiwok, http://www.kiwok.se.
[5]Zenicor, https://www.zenicor.se.
[6]Coala, https://www.coalalife.se.
[7]Wakaama, http://www.eclipse.org/wakaama/.
[8]lwm2m-node-lib, https://github.com/telefonicaid/lwm2m-node-lib

Algorithm 1 Sampling Rate Control Policy

```
1: HybridMode←off, hybridSeqMod←1, expSeqNum←0
2: prvChkSeq←0, lossCnt←0, lossRatio←0
3: procedure RXSTATISTICS
4:     lossCnt←lossCnt+(rxSeqNum−expSeqNum)/hybridSeqMod
5:     expRxNum←sampleRate∗PerfMgmtTimeItv/hybridSeqMod
6:     lossRatio ← lossCnt / expRxNum
7:     expSeqNum ← rxSeqNum + hybridSeqMod
8:     if PerfMgmtTimeItv expired then
9:         Call SamplingRateControl
10:    end if
11: end procedure
12: procedure SAMPLINGRATECONTROL
13:    if lossRatio > Threshold then
14:        Decrease sampleRate
15:        if sampleRate < guaranteedSampleRate then
16:            sampleRate ← guaranteedSampleRate
17:            HybridMode on, hybridSeqMod++
18:        end if
19:    else if lossRatio < Threshold then
20:        if HybridMode AND hybridSeqMod > 1 then
21:            hybridSeqMod−−
22:            if hybridSeqMod = 1 then
23:                HybridMode off
24:            end if
25:        else Increase sampleRate
26:            if sampleRate > maxSampleRate then
27:                sampleRate ← maxSampleRate
28:            end if
29:        end if
30:    end if
31:    prvChkSeqNum ← rxSeqNum, lossCnt ← 0
32: end procedure
```

optionally added to any GATT services. As an example, Fig. 6 shows the ECG GATT service with our extensions. This ECG sensor object enables a caregiver to set the transmission modes (*SendSamples*, *StoreSamples*, or *HybridMode*) and sampling rate, enable timestamps and sequence numbers, enable automatic sampling rate control, check the current sampling rate, specify a guaranteed sampling rate, and get performance information like packet loss ratio.

C. Performance Management of ECG as a Service

This subsection presents examples of the performance monitoring report function, the sampling control algorithm and *HybridMode* function for heart activity monitoring.

1) Performance Monitoring Reports: The extended ECG GATT service (Section V-B) means that the gateway manager ($\text{Manager}_{\text{gw}}$) can measure packet loss (based on sequence numbers), and delay variations and throughput (based on timestamps). A caregiver can subscribe to performance reports sent with a specified time interval. The average loss ratio, delay variation and throughput are documented for a chosen time interval. The performance metrics may also be invoked in the displayed sensor data for the end-user. Fig. 7 shows a graph with ECG recordings presented to the end-user (caregiver staff). The medical staff can read details such as missing data samples and sampling rate by hovering the mouse over points of interest.

2) Sampling Rate Control and HybridMode Policy Functions: The pseudo-code for the automatic sampling rate control policy algorithm is shown in Alg. 1 and the result of a test scenario in Fig 8. The algorithm finds the highest

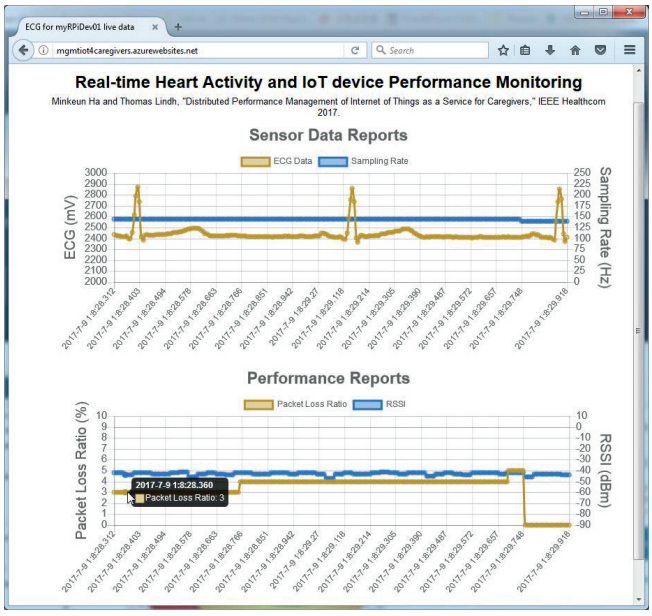

Fig. 7. Display of ECG recordings for caregiver staff. The upper graph shows real-time ECG data readings (yellow) and sampling rate (blue). The lower graph shows packet loss ratio (yellow) and RSSI (blue). The x-axis in both graphs shows timestamps for each ECG data sample. The caregiver staff can read more details by hovering the mouse over a point.

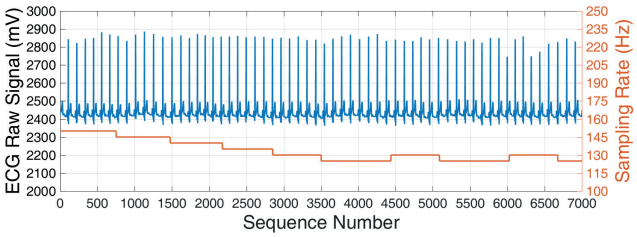

Fig. 8. ECG recordings and sampling rate during the test period. The algorithm adapts the sampling rate to keep the losses below the threshold (0.02 in this case). Here, the time interval for calculating the packet loss ratio (*PerfMgmtTimeItv* in Alg. 1) is set to 5 seconds.

sampling frequency given that the packet loss ratio is below the threshold. Results from previous research show that the stop-and-wait ARQ retransmission procedure in Bluetooth may lead to buffer overflow and increased packet loss during periods of transmission impairments [17]. In our test scenario, we injected random packet drops to show *HybridMode* and automatic sampling rate control. This will be discussed in the following subsection.

3) The HybridMode Function: Fig. 9 shows a scenario where the *HybridMode* function is activated. The maximum sampling rate and the guaranteed sampling rate are set to 250 Hz and 125 Hz respectively. The packet loss ratio threshold is 0.02. When the loss ratio exceeds the threshold, the gateway decreases the sampling rate with a pre-defined sampling control step (5 Hz in this case). After a while the sampling rate becomes lower than the specified guaranteed sampling rate (125 Hz). The gateway then enables *HybridMode*, which results in local storage of the data samples at the guaranteed

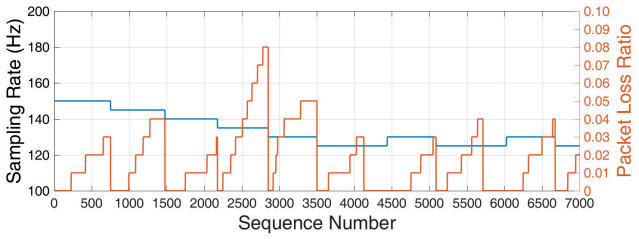

Fig. 9. The figure shows how the sampling rate is adapted to the level of packet loss.

sampling rate and continuous transmission of packets to the gateway at a lower sampling rate. Fig. 9 shows the packet loss ratio and the sampling rate during the test period. The *Hybrid-Mode* function was turned on twice, at sequence number 4127 and 5715. The size of each data sample stored locally when *HybridMode* is activated is 10 byte.

4) Discussion and Future Work: The goal has been to minimize the overhead and energy cost for the added functionality that the system provides. In the prototype, the maximum control signal overhead between the $\text{Agent}_{\text{sensor}}$ and the $\text{Manager}_{\text{gw}}$ is 10 bytes, and 14 bytes between the Agent_{gw} and the $\text{Manager}_{\text{server}}$. However, the requirement for battery lifetimes is lower compared to many other IoT applications. The battery can easily be replaced when a monitoring period is completed. A rough estimate shows that the energy budget in existing systems will not be exceeded. This will be studied in more detail as well as to include an estimate of the remaining battery lifetime in the performance reports.

The prototype system is coded in C and Node.js. Appropriate ways to express a policy description in a high-level language, which is translated to executable code, is part of future work. More efficient software methods to dynamically update, change and test policies and functions are also for future study. Some research is done on methods and tools in Software Defined Networks (SDN) and Network Function Virtualization (NFV) for IoT networks [19], and also using LWM2M and the OpenMTC platform[9] in this context [20]. This is part of future work as well as analysis of mobile edge computing for distributed management to offload the gateways in cloud-based IoT healthcare services.

VI. Conclusion

This paper presents a multi-layer distributed performance management model for IoTaaS for caregivers. The main contribution of the paper is a model and prototype implementation that combines traditional network management, low-complexity policy-based management, cloud computing methods, and lightweight IoT protocols and their data models. A case study on monitoring of heart activity demonstrates interworking between OMA Lightweight M2M protocol, Bluetooth low energy GATT services and Azure IoT hub cloud platform to enable distributed low-complexity policy-based performance management.

[9]OpenMTC platform, http://www.openmtc.org/index.html#openmtc.

Acknowledgment

We thank Microsoft for offering a grant and support to use the Azure IoT platform. We also thank the company Kiwok for valuable discussions and input on caregiver services.

References

[1] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, and D. Boyle, "From Machine-to-Machine to Internet of Things: introduction to a new age of intelligence," in Academic Press, 2014.

[2] E. Kovacs, M. Bauer, J. Kim, J. Yun, F. Le Gall, and M. Zhao, "Standards-Based Worldwide Semantic Interoperability for IoT," *IEEE Comm. Mag.*, vol. 54, no. 12, pp. 40-46, Dec. 2016.

[3] S. M. Riazul Islam, D. Kwak, MD. H. Kabir, M. Hossain, and K. Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," *IEEE Access*, vol. 3, pp. 678-708, Jun. 2015.

[4] C. Doukas and I. Maglogiannis, "Bringing IoT and Cloud Computing towards Pervasive Healthcare," *in Proc. Int. Conf. Innov. Mob. and Internet Serv. in Ubiq. Comput. (IMIS)*, Jul. 2012.

[5] Y. Coady, O. Hohlfeld, J. Kempf, R. McGeer, and S. Schmid, "Distributed Cloud Computing: Applications, Status Quo, and Challenges," *ACM SIGCOMM comput. comm. rev.*, vol. 45, no. 2, pp. 38-43, Apr. 2015.

[6] D. B. Hoang and L. Chen, "Mobile Cloud for Assistive Healthcare (MoCAsH)," *in Proc. IEEE Asia-Pacific Serv. Comput. Conf. (APSCC)*, Dec. 2010.

[7] G. Fortino, M. Pathan, and G. Di Fatta, "BodyCloud: Integration of Cloud Computing and body sensor networks," *in Proc. IEEE Int. Conf. Cloud Comput. Tech. and Sci. (CloudCom)*, Dec. 2012.

[8] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, and S. Andreescu, "Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-Based Processing: Opportunities and Challenges," *in Proc. IEEE Int. Conf. Serv. Comput. (SCC)*, Jun. 2015.

[9] J. Cubo, A. Nieto, and E. Pimentel, "A cloud-based Internet of Things platform for ambient assisted living," *Sensors*, vol. 14, no. 8, pp. 14070-14105, Aug. 2014.

[10] E. Kartsakli, A. S. Lalos, A. Antonopoulos, S. Tennina, M. Di Renzo, L. Alonso, and C. Verikoukis, "A Survey on M2M Systems for mHealth: A Wireless Communications Perspective," *Sensors*, vol. 14, no. 10, pp. 18009-18052, Sep. 2014.

[11] Z. Yang, Q. Zhou, L. Lei, K. Zheng, and W. Xiang, "An IoT-cloud Based Wearable ECG Monitoring System for Smart Healthcare," *J. Med. Syst.*, 40:286, Oct. 2016.

[12] T. Hendrikx, M. Rosenqvist, P. Wester, H. Sandström, and R. Hörnsten, "Intermittent short ECG recording is more effective than 24-hour Holter ECG in detection of arrhythmias," *BMC Cardiovascular Disorders*, 2014:41, Apr. 2014.

[13] Y. Li and W. Wang, "Can mobile cloudlets support mobile applications?," *in Proc. IEEE INFOCOM*, Apr. 2014.

[14] M. T. Beck, M. Werner, S. Feld, and T. Schimper, "Mobile Edge Computing: A Taxonomy," *in Proc. Int. Conf. Adv. in Future Internet (AFIN)*, Nov. 2014.

[15] H-G Hegering, S. Abeck, and B. Neumair, "Integrated management of networked systems," in Morgan Kaufmann publishers, Aug. 1999.

[16] K. Twidle, E. Lupu, N. Dulay, and M. Sloman, "Ponder2 - A Policy Environment for Autonomous Pervasive Systems," *in Proc. IEEE Works. Policies for Dist. Syst. and Net. (POLICY)*, Jun. 2008.

[17] J. Wåhlslén and T. Lindh, "Real-time Performance Management of Assisted Living Services for Bluetooth Low Energy Sensor Communication," *in Proc. Int. Works. Protocols, Applications and Platforms for Enhanced Living Environments (PAPELE)*, May 2017.

[18] R. Kamal, M. S. Siddiqui, H. Rim, and C. S. Hong, "A policy based management framework for machine to machine networks and services," *in Proc. Asia-Pacific Net. Oper. and Manag. Symp. (APNOMS)*, Sep. 2011.

[19] N. Bizanis and F. A. Kuipers, "SDN and Virtualization Solutions for the Internet of Things: A Survey," *IEEE Access*, vol. 4, pp. 5591-5606, Sep. 2016.

[20] A. Corici, R. Shrestha, G. Carella, A. Elmangoush, R. Steinke, and T. Magedanz, "A solution for provisioning reliable M2M infrastructures using SDN and device management," *in Proc. Int. Conf. Inform. and Comm. Tech. (ICoICT)*, May 2015.