Improving Awareness in Ambient-Assisted Living Systems: Consolidated Data Stream Processing

Koray İnçki[1(✉)] and Mehmet S. Aktaş[2]

[1] Computer Engineering Department,
Adana Science and Technology University, Adana, Turkey
kincki@adanabtu.edu.tr
[2] Computer Engineering Department,
Yıldız Technical University, Istanbul, Turkey
aktas@yildiz.edu.tr

Abstract. Ambient Assisted Living (AAL) aims providing a quality of life to elderly for sustaining their lives without constant supervision. The technology developments enabled devices with more processing power, longer battery life and more advanced sensor capabilities. Internet of Things (IoT) is a phenomenon that allows seamless interconnection of very small devices over Internet; bringing new opportunities for AAL solutions. AAL systems equipped with IoT devices will generate vast amount of data in short time, thus a big data problem to mangle. This study proposes a simulation infrastructure that allows researchers to create their own AAL scenarios without real devices, and a system architecture to tackle the big data problem that is inferred by utilization of IoT in AAL systems. In order to understand feasibility of the architecture we conduct a performance experiment, in which we increase the number of messages that the system can handle per second. The resulst of the experiment are promising.

Keywords: Ambient-Assisted Living · Internet of Things · Big data · Kafka · Complex-event processing · Stream processing

1 Introduction

Internet of Things (IoT) is a new term that implies utilization of a collection of enabling technologies in a seamless and coherent way for interconnection of "things" in our lives. When K. Ashton conied the term in 1999 [1], he suggested to interconnect sensors and actuator devices so that they don't require the intervention of human beings in order to cooperate. The devices that are equipped with IoT capabilities can execute the sensing and acting duties they are predestined to, whilst they cooperate with devices from far away locations through Internet connection capability. These devices are usually resource constraint devices which lack enough processing power, and storage space for running resource-hungry applications, such as processing high volumes of data.

One can notice the time delay between the mass proliferation of the phenomenon and the first time it was introduced in to the literature. We believe that the reason behind this delay was the lack of standardization and open-source community support.

M.U. Ahmed et al. (Eds.): HealthyIoT 2016, LNICST 187, pp. 89–94, 2016.
DOI: 10.1007/978-3-319-51234-1_14

The implementation momentum of the technology has been increasing since the adoption of standardization efforts in certain technology areas, such as communication and networking. Constrained Application Protocol (CoAP) [2] is a soon-to-be standard application layer protocol for Constrained Resource Environments (CoRE) [3]. The resource constraint devices, which are installed with resource-efficient operating systems such Contiki-OS, have more support for CoAP-like network protocols.

AAL is devised to improve quality of life for elderly people [5]. Its main purpose is to enable elderly people to live in their own houses according to their own life styles while still maintaining their well-being and safety. The utilization of IoT technologies is an ongoing discussion in the AAL community, because it opens up new opportunities for various and diverse application scenarios [4]. AAL systems generally rely on powerful backend systems for analyzing data and decision making; and the decisions that such systems make are as reliable and usable as the real-timeliness of the decisions. Increasing number of IoT systems in AAL systems will generate mass streams of data to process in a very short amount of time. Therefore, we need to employ data stream processing approaches to effectively asses such big data on run time. Real-time data stream processing has to meet certain criteria in order to retain the real-time properties of the data. We propose to integrate the complex-event processing technology with real-time big data stream processing technology such that an architectural framework will incorporate best features of both technologies to tackle real-time big data processing problem in AAL-IoT systems.

2 Background

IoT. IoT describes coherent collaboration of a set of enabling technologies. Those enabling technologies are ubiquitous communication through IP-based networking, ambient intelligence through sensor and actuator capabilities, unique identification through URI based naming services [2]. These concepts impact the proliferation of this technology tremendously. Ubiquitous communication allows for accessing the things anytime from anywhere, thus enabling a seamless interoperation between ‘things’. Ambient intelligence capabilities provide the ‘things’ to capture physical phenomenon going on around them through sensor features, and enables them to process the data gathered from their environment. Unique identification feature not only help to authenticate the devices but also promotes heterogeneity in the manufacturing parties of those devices.

AAL. Common features of AAL applications are providing health support, safety, independence, mobility and social interaction. These devices are frequently utilized in closed loop service model, in which data produced by devices are gathered for deducing intelligent conclusions for the well-being of the elderly person (e.g., reminding to take a pill, to alert closest emergency personnel in case of abrupt changes in the person’s vital signs). With the introduction of IoT technologies, those devices are becoming more reliant on service-oriented interfaces, which enable care-givers remotely monitor the person.

CoAP. CoAP is designed to provide HTTP-like communication paradigm for resource-constraint devices [2]. It enables communication between endpoints by using a request/response messaging model. It mainly designed according to RESTful guidelines, so it supports methods like GET, PUT, POST, and DELETE. Its low overhead and simplicity make it suitable for constraint resource environments. Endpoints behave either as client or a server in a request/response message model. Erbium [6] is a C implementation of the standard for Contiki-OS [7], and Californium [8] is a Java implementation for host systems.

CEP (Complex-Event Processing). An event can be described as an occurrence or action that changes the state, so that we need to take reaction. Complex-events can be described as a collection of simple events. Simple events constitute complex events according to various relations, such as timing, location, context, etc. Complex-event processing helps designers of information systems to make more intelligent decisions based on simple events occurring in a system. Esper [10] is an open-source implementation CEP engine, which has a particular programming language to describe simple events and patterns of relations amongst those so as to generate complex-events.

Stream Processing. Results of processing real-time data might affect operation of various applications, and such critical information must not be altered, neither can it be neglected. For example, an alarm generated by a heart-rate monitoring system must be immediately intercepted and care-givers must be notified at real-time. Kafka [9] is a log-based stream processing tool that allows processing real-time data in-memory caches, which allows for very fast processing. It also supports exactly-once processing paradigm, which guarantees processing a data once and only once in the system. Such reliable data processing is crucial for AAL systems in order to provide a reliable service to the elderly people.

3 Improving Awareness Through Stream Processing

Closed-loop service for AAL systems might be sufficient for ultimate well-being of elderly people. In such a system, the system might remind the patient to take some pill. On the other hand, if AAL systems are improved with decision making mechanisms, then human effort and error for acting on certain abnormal conditions would be improved. In certain situations, simple data coming from individual sensor devices can be aggregated to make more informed decisions. For example, increased blood pressure rate and falling data of an elderly might mean that the person is having a heart-attack. Here, we propose our solution architecture for such health related decision scenarios.

Figure 1 shows three sensors; blood-pressure rate, fall sensor, and body temperature sensor; which are implemented as applications that run on Contiki-OS [7]. Each sensor application is loaded onto a mote (a wireless node), and motes are simulated on Cooja simulation environment [7]. Motes are preinstalled with Erbium CoAP library, by which each mote acts as a CoAP server for serving its sensor data. CoAP client is implemented in Java by utilizing Californium open-source library [8]. Border router enables CoAP clients access to servers through IPv6 based unique addresses. CoAP protocol facilitates accessing sensor values as services provided in accordance with

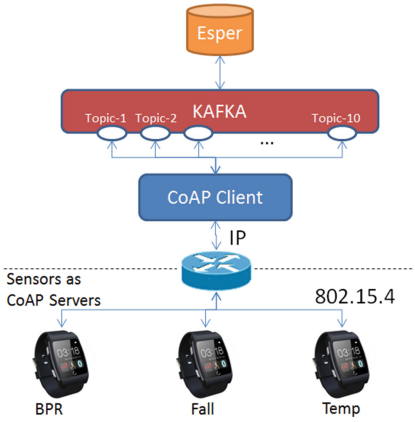

Fig. 1. Architecture for real-time stream processing of health data.

RESTful APIs. This feature of the framework allows us to insert any type of sensor device produced by any manufacturer, provided that the device implements CoAP server capabilities as before.

CoAP continuously updates clients with new sensor data at server devices by OBSERVE method, which allows us to establish a reliable monitoring architecture for AAL systems. Servers send data to clients in a quadruplet that is composed of *{SensorId, Value, Timestamp, SensorType}*. Each sensor might have a particular value range and period of sending update messages might change as well.

CoAP Client receives the sensor data from servers and relays them to appropriate *Topic* in Kafka In-Memory processing system. Having a logging mechanism such as Kafka improves the reliability of our approach, because Kafka guarantees the processing of each data once and only once during its lifetime. Moreover, it performs its operations in cache so that it improves the temporal performance of the overall system. As it can be seen from Fig. 1, CoAP client publishes updates to particular topics in Kafka, and then Esper engine receives those updates through the subscription mechanism provided by Kafka. The data published by CoAP client constitute simple events for the Esper engine.

4 Experiment and Evaluation

In order to understand the performance of proposed solution, we have to conduct an experiment on a sample IOT system. To achieve this, we increased the number of message load on the system to assess the system performance.

Complex-events generated by Esper engine represent the emergency actions to be taken on abnormal situations in well-being of a person. The scenarios we have implemented are fictional: **Heart Attack Scenario:** When the arteries to the heart get blocked or rupture, this starts affecting the hearth's functioning. This prevents the normal circulation of the blood in the body. We identify the symptoms of heart attack as follows:

(a) Sudden increase in body temperature, (b) Sudden increase in blood rate, (c) Sudden loss of balance or coordination. **Stroke Scenario:** One can think of a stroke like the brain's version of a heart attack. When arteries to the brain become blocked or rupture, they start killing brain cells and sometimes causing permanent brain damage and even paralysis. We identify the symptoms of stoke as follows: (a) Sudden loss of balance or coordination, (b) Sudden shortness of breath and increase in heart rate. Those scenarios can be expressed in Esper engine by using EPL format in Table 1 follows.

Table 1. Formal representation of rules in complex event processing for abnormal health situations

Rule ID	Formal Representation	Description
Rule 1	**ON PATTERN** ((((Body_Temperature (τ)$^{-}$> ζ TEMPRATURE) ∧ (Blood_rate (τ)$^{-}$> ζ BLOOD_RATE) ∧ (HAS_FALLEN node (τ)$^{-}$> ζ 0)) **DO ACTION** (**invoke** heart_attack_procedure)	If disjunction of Body_Temperature, Blood_Rate, and the Fallen Status level exceeds threshold value within a specified time period τ, then we consider this situation as Heart Attack Situation
Rule 2	**ON PATTERN** ((Blood_Rate (τ)$^{-}$> ζ BLOOD_RATE) ∧ (Noise_Level node (τ)$^{-}$> ζ NOISE_LEVEL) ∧ (HAS_FALLEN node (τ)$^{-}$> ζ 0)) **DO ACTION** (**invoke** stroke_procedure)	If disjunction of Blood_Rate, Noise_Level and the Fallen Status level exceeds threshold value within a specified time period τ, then we consider this situation as Stroke Attack Situation

As it can been seen in Table 1, if the certain collection of simple events collected from IoT devices are above predefined threshold values, then the system detects the Complex Events and invokes the relevant procedures.

We performed Load Testing by increasing the message load within a time period to measure system performance. We increased the number of events per second starting from 50 msgs to 700 msgs. We measured the latency of the messages in each component of the system. Up to 600 messages/second, the components of the system showed negligible latency (in the order of seconds). However, the results show that after 600 message load, the latency for the publish-subscribe message bus (Kafka) has increased to a peak value that was over 400 ms. It was clear that additional Kafka node should be included into the system to avoid the bottlenecks in message overload. Based on performance evaluation results, processing overhead of the introduce system was acceptable for our current system.

These results indicate that the performance overhead is negligible until after 600 msgs/sec on the system. If there is need for message load of 600 msgs/sec we recommend that there needs to be another Kafka (Publish/Subscribe) node in the system.

5 Conclusion

Real-time processing of sensitive data as health data requires special solutions, as such a process necessitates a fast and reliable information system to process the data. In this paper, we proposed an architecture that integrates open-source software for complex-event processing and stream processing, which promises to seamlessly analyze real-time data flowing from health sensors from an elderly person. Our experiment on open-source simulation environment demonstrates the validity of our approach based on collected data. We believe using such open-source, community supported software for AAL purposes will promote development of new solution architectures.

References

1. Ashton, K.: That 'Internet of Things' thing. RFID J. **22**(7), 97–114 (2009). http://www.rfidjournal.com/articles/view?4986
2. IEEE Constrained Application Protocol Standard. https://tools.ietf.org/html/rfc7252
3. IETF Constrained RESTful Environments Charter. https://datatracker.ietf.org/wg/core/charter/
4. Gubbi, J., Buyya, R., Marusic, S., Palaniswani, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Elsevier Future Gener. Comput. Syst. J. **29**(7), 1645–1660 (2013)
5. Costa, R., Carneiro, D., Novais, P., et al.: Ambient assisted living. In: Springer 3rd Symposium of Ubiquitous Computing and Ambient Intelligence, Salamanca, Spain (2008)
6. Kovatsch, M., Duquennoy, S., Dunkels, A.: A low-power CoAP for Contiki. In: Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems. IEEE, Valencia (2011)
7. Dunkels, A.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: 29th Annual IEEE International Conference on Local Computer Networks, Florida, USA (2004)
8. Kovatsch, M., Lanter, M., Shelby, Z.: Californium: scalable cloud services for the internet of things with CoAP. In: 4th International Conference on Internet of Things, MA, USA (2014)
9. Kreps, J., Narkhede, N., Rao, J.: Kafka: a distributed messaging system for log processing. In: ACM NetDB 2011, Athens, Greece (2011)
10. Esper Open-Source Complex-Event Processing Engine. http://www.espertech.com/esper/