Mobile Netw Appl (2014) 19:33–44
DOI 10.1007/s11036-013-0479-2

Continuously Changing Information on a Global Scale and its Impact for the Internet-of-Things

Stefan Forsström · Theo Kanter

Published online: 27 October 2013

Abstract This article analyzes the challenges of supporting continual changes of context information in Internet-of-Things applications. These applications require a constant flow of continuously changing information from sensor based sources in order to ensure a high quality-of-experience. However, an uncontrolled flow between sources and sinks on a global scale wastes resources, such as computational power, communication bandwidth, and battery time. In response to these challenges we present a general approach which focuses on four layers where we provide a proposed solution to each layer. We have realized the general model into a proof-of-concept implementation running on devices with limited resources, where we can moderate the information exchange based on relevance and sought after quality-of-experience by the applications. In conclusion, we evaluate our solution and present a summary of our experiences regarding the impact of continuously changing information on the Internet-of-Things.

Keywords Pervasive · Context awareness · Real-time · Internet-of-things

This work has been supported by grant 2010-00681 of VINNOVA the Swedish Governmental Agency for Innovation Systems, and by grant 00163383 of the EU European Regional Development Fund, Mellersta Norrland, Sweden.

S. Forsström (✉)
Department of Information and Communication Systems,
Mid Sweden University, SE-851 70 Sundsvall, Sweden
e-mail: stefan.forsstrom@miun.se

T. Kanter
Department of Computer and System Sciences,
Stockholm University, SE-164 40 Kista, Sweden

1 Introduction

Today there is a significant proliferation in situation-aware computing, in which different applications and devices can change their behavior depending on the situation of their user. These applications are integrated into people's everyday lives and are made aware of their user's situation and context, in order to change their own application behavior. This opens up a new field of user friendly services, which are able to pervasively adapt per person in order to provide the best possible service for that particular user. These applications do however require a constant feed of continuously changing information from ubiquitous information sources, in order to make real-time critical decisions making. The sources are often based on sensors which can provide helpful information regarding the user's situation, such as information from the environment in the user's surroundings. In detail, this has enabled us to use the features of mobile computing and intelligent reasoning in new forms of applications [1], such as sensing campaigns and social interactions in large-scale populations. This also extends to the vision of an Internet-of-Things [2], where many small devices will be ubiquitously connected and communicating with each other. For example gathering sensor information from smartphones, smart homes, and wireless sensor networks to create new types of context-aware services. The current expectations are of the order of 50 billion connected devices to the Internet by 2020 [3].

Current Internet-of-Things applications focus on quite narrow scenarios, where they are able to manage the communication without scalability issues. This has led to a large proliferation in the area, but it has obscured the real problems associated with large scale deployment. These early successes include, for example, health care solutions [4] which can share simple data such as heart-rate and

blood pressure a few times per second, and location-based services [5] which often claim to be operating in real-time even when they only update the location every few seconds. Smart home solutions such as [6] focus on local exchange and reasoning of contextual information, but have limited potential because they do not share information with all connected entities on a global Internet-of-Things. We foresee that the future will demand better services in the form of faster update frequencies, awareness, and reasoning. Thus, applications will require a continuous stream of sensor and actuator updates for their regular operation. Hence, as our surroundings become more aware by means of an increasing number of sensors, the amount of information on the Internet-of-Things will also increase.

In response to these earlier shortcomings in relation to the handling of this vast amount of information, we are researching a general model that is able to widen the application focus and support continuously changing information from large numbers of sources on the Internet-of-Things. This had led to the identification of the following challenges which must be solved before it is possible to move from the current narrow applications on the Internet-of-Things, to large scale proliferation.

1. The Internet-of-Things is based on the idea of many small connected entities which have quite limited capacity on their own, but which together are able to form a usable collaborative collective knowledge. However, creating this knowledge is expensive in terms of communication and data processing. Therefore the first challenge is to determine mechanisms that adaptively minimize the resource consumption based on application demands, which should ensure a good quality-of-experience for the end user and longevity for the connected entities.
2. This collective knowledge will require data from widely spread sources, which will be globally distributed. Hence, this large and constant flow of information in the system renders traditional approaches unsuitable for the Internet-of-Things. Therefore, the second challenge is to determine mechanisms which are able to manage global data flows in a properly scaling manner, avoiding choke points, and which do not expose any central points of failure.
3. Once the raw data has been acquired, it must be possible to create a higher form of knowledge from it. This demands an information model which is capable of providing knowledge fusion from the data and being simultaneously continuously updated with the latest values. Therefore, the third challenge is in discovering an information model that can create knowledge from continuously changing raw sensor values from the Internet-of-Things.
4. Because of the inconsistency of the knowledge and the continuous feed of raw data from the Internet-of-Things, most knowledge becomes temporal and transient. Because of this, queries related to the data might become invalid even before the query has passed through the whole dataset and returned an answer. Therefore, the fourth challenge is in determining knowledge access models that are able to operate, understand, and make use of the continuously changing knowledge base.

The remainder of this paper is organized in the following order: Section 2 presents the background and important prior work which has been conducted within the area. Section 3 presents our general model for enabling an Internet-of-Things which allows continuously changing information. Section 4 evaluates our approach based on different types of architectures. Section 5 presents our proof-of-concept implementation and Section 6 an evaluation thereof. Section 7 presents our analysis of the impact of continuously changing context information from the Internet-of-Things, and Section 8 presents our conclusions and future work.

2 Background

The idea of distributing small sensor motes in our surroundings, in order to pervasively sense our environment has existed for some time. The area has been extensively researched [7], but the focus has mainly been on the short range communication between the motes and the optimization thereof. Thus, the scope of the research area has been quite narrow and focused on the hardware technical aspects. This has, in turn, spawned many sensor mote platforms and operating systems which are particularly aimed at wireless sensor networks, such as Contiki and TinyOS. The research into Machine-to-Machine (M2M) communication has also proliferated from the pervasive sensing area [8]. In this case, the M2M communication focus is on creating intelligence between different devices, without human interaction. For example changing the household heating based solely on different sensor and actuators in the home involves communication between each of them. However, they often rely on different types of sensor gateways or centralized databases to relay the sensor information on the Internet. Thus, this involves hiding the actual sensor motes in order to protect them from true ubiquitous access and global information sharing.

The concept of utilizing information from sensors attached to different entities, in order to provide more personalized, automatized, or even intelligent application behavior can be referred to as the Internet-of-Things [2].

This also includes the ideas of having sensors everywhere in our surroundings, to form a truly pervasive and ubiquitous smart space. The reasoning is that everyday objects will become connected and that they can display intelligent behavior. New forms of applications will also become widespread when all of these objects offer different types of services. These types of Internet-of-Things applications are projected to have a significant impact with regards to how we interact with the world, people, and things in the future. We believe that the main differences between Internet-of-Things applications and traditional sensor systems are the focus on global scale and the integration into everyday objects.

The large number of sensors and actuators on the Internet-of-Things will create even larger amounts of data that is required to be disseminated to applications. Others have also identified this sensor data overload from mobile devices [9], and have indicated the requirement for better scaling infrastructures in the future. Parallels can also be drawn to research conducted within the Big Data area [10], which focuses on the problems associated with managing and querying large amounts of data. In the light of this, we have performed an initial measurement on the mobile devices available today. We tested a typical mobile phone (a Samsung Galaxy Nexus) and it can produce up to 1200 values each second. Even if sensor values are quite small in size this number is staggering large, especially when viewed in the perspective of sharing sensor information on a global scale. This can be seen as an indication of the impending information overload from sensor sources, which will be a problem for the future Internet-of-Things where all devices will be connected and want to ubiquitously exchange sensor information.

3 Proposed general model

From the four challenges in Section 1, we have derived a layered general model of the areas for which the challenges are exposed. This layered general model can be seen in Fig. 1, where we can see that the fourth challenge regarding knowledge access is the topmost layer and is furthest toward the application. This layer will deal with the challenges regarding applications which are querying data and requesting data derived from the Internet-of-Things. The second layer is the information model, which deals with the challenges regarding the organization of data and how to store it in an efficient manner with regards to each connected thing. The third layer deals with the communication architecture, which must organize the connected things and enable communication between the end points in an efficient manner. Finally, the fourth layer deals with the limited resources available and the actual hardware upon which the information is being produced.

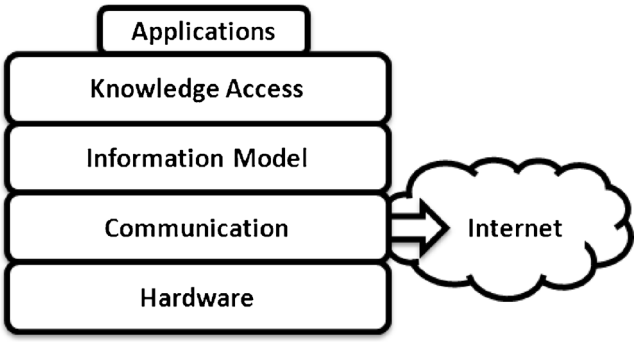

Fig. 1 Proposed general model

3.1 Knowledge access layer

The knowledge access is the topmost layer in our general model, where challenge 4 regarding accessing information is dealt with. This layer should provide query models that can operate, understand, and make use of the continuously changing temporal knowledge that will come from Internet-of-Things sources. As previously stated, query results might become invalid even before the query has been returned from the information model in the layer below. In detail, we have discovered that the knowledge access layer generally deals with five common problems, namely reliability of a query from the information model (that it returns a usable result), rapid response (so that the application will not be kept waiting), searching for data (for finding new data sources), two way data flow (for controlling actuators), and authenticated secured data access (to limit the access of private information).

Applications which are accessing knowledge are often conducted through different Application Programming Interfaces (APIs). Even though there are standardized patterns with regards to how these APIs should be formed, a vast number of different approaches still exists. In distributed environments publish/subscribe methods have traditionally been used to solve application information access. This is because an application has the ability to subscribe to a specific piece of data and receive updates whenever these occur. Lately there have also been a proliferation of cloud computing and mashup services, which are usually HTTP and REST [11] based information access. For relational databases the SQL language has been used for many years in relation to solving application information access, even though different NoSQL [12] databases have gained ground during recent years because of their ability in relation to handling large amounts of distributed data.

3.2 Information model layer

The information model is the structure and organization of the data, thus it must deal with challenge 3 which involves information and knowledge. The application access layer will query the information that an application both desires and requires for its operation. Hence, the information model must solve the demands from the layer above and all of its requirements. The information model is also dependent on the data itself, and thus places demands on the data that it stores. In detail, we have determined that the information model deals with the following common problems, namely, linking data with other data (and thus traversing it), support for fuzzy values and units (because not all sensors provide discrete values and are of the same type), and handling continuously changing data from remote sources (because sensor values are constantly changing).

The information model is the focus of many publications with regards to architectures for situation aware applications, because it solves the majority of challenges based on the stored information. The information model must provide the means in relation to reasoning, linking, searching, relations, comparing, inserting, and the deletion of information. The most common approach within the area of Internet-of-Things is to apply some form of ontology, such as the Web Ontology Language OWL [13]. However, many more exists [14]. In contrast to the more complex information models, quite simple XML schemes such as SensorML [15] also exists, which simply markup the data and allow the applications to create the knowledge from the raw data.

3.3 Communication layer

The communication architecture enables the exchange of information between the connected entities. Hence, the communication layer is used by the information model to retrieve and send information to other devices when they require a particular piece of data. Therefore, this layer must deal with challenge 2, which involves continuous data flows. Because of the inherent structure of the Internet-of-Things we have determined that the communication architecture must deal with the following problems, namely, to avoid unnecessary data exchange (data which is not demanded by the applications), to avoid unnecessary proxying of data (to enable effective distribution), to avoid any central point of failure (to increase system resilience), and to have a low overhead (to save resources overall). The recent advances in the Internet-of-Things area has also produced a large number of different general platforms which are made for creating applications on. These general platforms can be organized into three distinct categories, depending on where they store the actual information. In detail, these three categories are: centralized, semi distributed, and fully distributed architectures.

Centralized systems store the information under a single administrative authority, either in a single large database or replicated in a cloud based manner. Examples of centralized storage include SenseWeb [16], SENSEI [17], and SERENOA [18]. However, all of these centralized systems have scalability issues when the number of updates and queries increases in magnitude. Therefore, they will have problems associated with support for continuously updating data with low latencies. These systems are also prone to failure, because they expose single points of failure.

Semi-distributed systems store the information locally on each entity in a peer-to-peer manner, but still maintain a centralized authority for the exchange between peers. These systems use session-establishment protocols to exchange the context, but under the supervision of a centralized authority. Mobilife [19], CONTEXT [20], ADAMANTIUM [21], and ETSI TISPAN [22] which is based on 3GPP IMS [23], are examples of such systems. These systems will scale in a better manner when compared with the centralized systems, but they still maintain a centralized component. This will become a bottleneck for the exchange, when the entities perform long queries on a large and continually changing dataset, since the centralized component has to administer all the sessions, even if the actual exchange is sometimes performed outside the centralized component.

Fully distributed systems both store and administer the information locally on each entity in a peer-to-peer manner. These systems often utilize distributed hash tables to enable logarithmic scaling when the number of entities increases in magnitude. Examples of such systems are MediaSense [24], SOFIA [25], and COSMOS [26]. Naturally, these systems do not contain any single point of failure and are thus more resilient, even if the distribution itself often requires additional overhead in order to maintain an overlay. The main problem associated with fully distributed systems is that it places a larger responsibility on the end devices. However these end devices can be limited in capacity, e.g. bandwidth and processing power, which will cause there to be issues when sharing context information on a large scale.

3.4 Sensor and device layer

On the Internet-of-Things many different sensors and devices will be connected with different resources and this must be handled in the lowest layer. Currently, a wide range of different hardware exists that are suitable for the Internet-of-Things, for example, environmental sensors, smartphones, wireless sensor motes, and smart dust. However, the idea of the Internet-of-Things is to connect all of these and to enable collaboration between them, even if they have different base conditions which are dependent

on the hardware. One of the vital parts of the Internet-of-Things is the incorporation of sensors and actuators into everyday objects. Hence, the sensor and device layer must deal with challenge 1 regarding the limited resources and heterogeneous hardware. As previously stated, research into this aspect has been conducted for quite some time, namely in the area of Wireless Sensor Networks (WSN) [7]. In WSN, the main focus has always been on the resource limitations of these devices, because of their limited battery capacity and processing power. At the present time quite feature rich but still resource limited platforms exists, such as IPv6 capable TinyOs and Contiki. Many small computer platforms also exists, such as Raspberry Pi [27], which have been used in some Internet-of-Things concepts. It can also be argued that today's smartphone platforms also fit into the same resource limited device category, because of their limited battery capacity and processing power. Although there is a drastic increase in performance as new models of smartphones are being released.

4 Realized model

From the general model and the available related work, we have derived a realized model regarding how we believe that the Internet-of-Things should be realized in order to address the four challenges. Figure 2 presents our realized model which shows our proposed approach to create a device and sensor agnostic architecture, while applying a fully distributed communication architecture, an information model based on contextual entities and application access that provides contextual views of the transient data in the information model. It is important to note that our proposed general model and to some extent our realized model, is similar to the ETSI M2M architecture [28] especially in the lower layers (communication and hardware layer). But our approach and theirs differ, because in our solution we include for example the topmost layers to a larger extent (Application, Knowledge access, and Information model). They also differ in aspects when it comes to the actual realization, we for example propose a solution to limit the exchange of sensor information in order to handle the dissemination of continuously changing sensor values. Furthermore, we propose the usage a of fully distributed communication architecture between the nodes, where the ETSI M2M architecture proposes semi distributed solutions based on IMS.

4.1 Sensors, actuators, and devices

We believe that the sensors, actuators, and devices will still be very heterogeneous in terms of capacity and resources, but that they will converge toward an all IP based framework with a lightweight REST based protocol such COAP [29] for retrieving sensor values. The first steps toward this are already visible, because both TinyOS and Contiki support IPv6 communication and the COAP protocol for data exchange. However, depending on how powerful the devices are, there will still be a requirement for some type of super device, which can act for the weak sensor mote inside the communication architecture. Thus, we will have aggregation points that are able to communicate the information forward when the resources on the sensors or actuators are limited. In detail, we have realized the hardware layer in two parts, separating devices which can manage the upward layers and devices which are unable

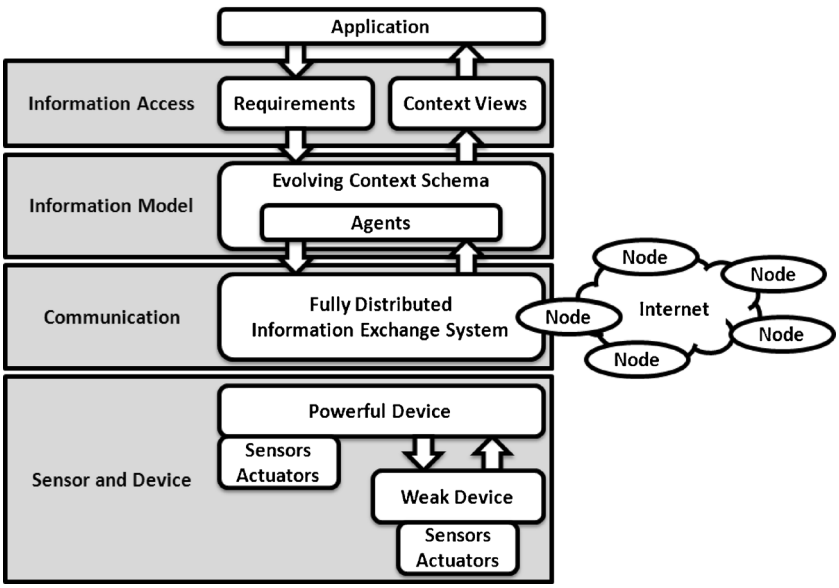

Fig. 2 Realized model

to do so because of resource limitations. In this case, the communication between weak devices and more powerful devices is conducted via IP and COAP.

4.2 Communication architecture

To realize the communication layer, a fully distributed communication platform is perceived as the way forward, but with some type of centralized cloud architecture back-end as a support system. For example, sharing sensor data is performed directly between to entities in a peer-to-peer manner, while storing sensor data for safekeeping and later usage can be performed on a cloud based back-end. The fully distributed architecture should both scale well with billions of globally connected devices, and avoid any central points of failure. However, there are many choices with regards to which type of fully distributed system should be implemented, and at the present time it is not possible to observe a clear long term winner. Therefore we will use the MediaSense platform [24] for now in our realization, as a typical low cost fully distributed peer-to-peer based platform for data exchange. This choice was based on its availability, openness, extensibility, and simple architecture, which fit very well with our other research goals as well.

4.3 Context schemas

To realize the information model layer we will apply a model, which we call context schemas, where each entity on the Internet-of-Things will have its own schematic representation of itself. In detail, we consider any entity that has sensors or actuators attached to itself and which has the ability communicate this information forward, as an entity. All these entities will have a digital representation of their current situation, a form of self perspective, which we call a context schema. This context schema is based on our previous work [30], and it contains all the relevant knowledge for a particular entity's current situation and context. In detail, the schema is the composite set of all local contextual information and all relevant contextual information from other entities. This can be seen in Eq. 1, where C_n is the resulting context schema for an entity n containing a set of information I. I_n^{local} is the set of local information available at entity n, and $I_n^{relevant}$ is the set of all relevant information for entity n from other entities. In detail, Eq. 2 defines I_n^{local} as the set of all information on the local entity n for all different information types k. Eq. 3 shows how we calculate the set of relevant context information from other entities. For all other entities m and for all their k different information types, $i_{m,k}$ is the information being evaluated and $f_k^{relevant}(i_{m,k})$ is the relevance function for determining whether the information of type k from entity m is relevant or not. In this case $x_{n,k}$ is the threshold value for the relevance evaluation, given by the evaluating entity n for the particular information type k. The idea is thus to modify $x_{n,k}$ to fit the desired quality, precision, complexity, etc. for a particular entity's application scenario. To summarize, a context schema is a self representation of an entity which contains all local information and all relevant information from remote sources, where the relevancy is determined on a per application and information type basis.

$$C_n = \{I_n^{local} \cup I_n^{relevant}\} \tag{1}$$

$$I_n^{local} = \forall_k \{i_{n,k}\} \tag{2}$$

$$I_n^{relevant} = \forall_{m \neq n} \forall_k \{i_{m,k} | f_k^{relevant}(i_{m,k}) \leq x_{n,k}\} \tag{3}$$

Traditionally the producers (sensors) have controlled the flow of information to the consumers (applications), using subscription based notifications and their update frequency. This, in turn, forces all entities interested in a particular type of information to constantly receive updates with the latest values. Some solutions do however apply conditional subscriptions to limit this flood of information, but the flow of information from these conditional subscriptions are still similar. Subscriptions can also be sent as multicast messages, but that has proven to be difficult to realize in practice on the public Internet. We approach this problem in relation to controlling the flow of information by combining adaptive subscriptions and autonomous agents, in order to ensure high quality-of-experience for the end user. We will not specify in detail on how the subscriptions will be managed, as it can be either adapted by threshold values, other limits, or even multicasted. In detail, our approach is realized using three different types of agents. A publishing agent, which gathers local sensor information into the context schema and sends subscription updates to other interested entities. A traversing agent, which traverses the system to find new information and to search for a specific piece of information. And lastly, a maintenance agent which maintains the context schema so that it only contain relevant information depending on the sought after quality of service by the applications. Figure 3 shows flowcharts with regards to how these agents operate on the context schema in order to continuously maintain it at an up to date level and to also be relevant for the end application.

In detail, the publishing agents simply retrieves all the latest updates from local sensors, inserts the values into the context schema, and notify any subscribers. Thus, the publishing agents provide the I_n^{local} set in Eq. 1. The sensors do, however, impose their own problems, since they only provide raw values and these often take the form of different formats based on each manufacturer. Thus, the problem

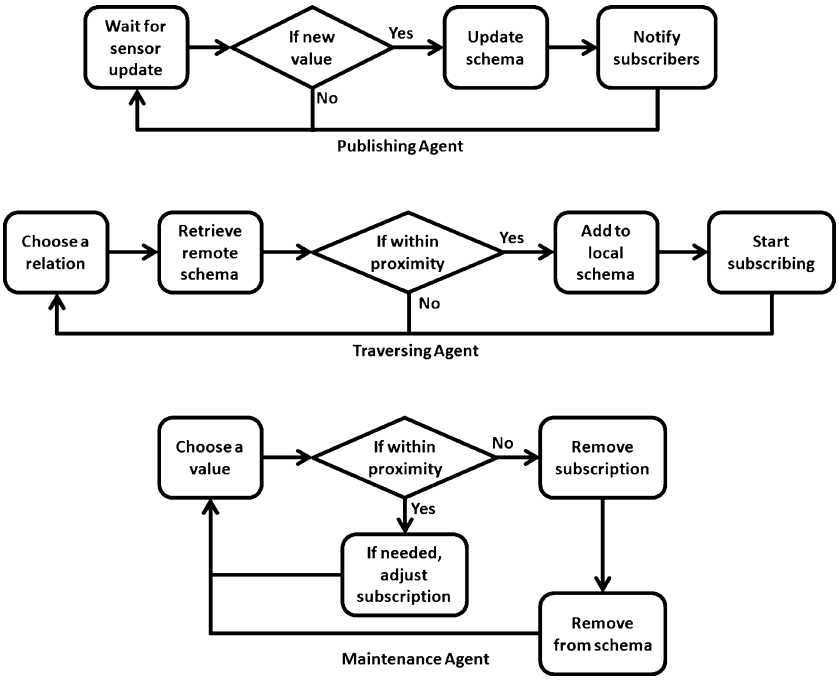

Fig. 3 Flowcharts over the three agents, publishing, traversing, and maintenance

that the publishing agents solve is to create usable information from many different types of sensors and to provide this information upwards to the evolving context schema.

The traversing agents search for new relevant information, evaluating all new context information it finds with the relevance function, and then inserting it into the context schema and starting a subscription. The traversing agents thus solves the problem of finding new relevant context and providing the $I_n^{relevant}$ set in Eq. 1. In detail, the traversing agent browses the local context schema for relations which could be relevant to explore. It is however important to note that although the agent is named traversing it does not physically relocate within the system, it is still running on the end device but searches the system from its own device. To find a new entity, the traversing agent communicates with a known entity from the local context schema, asking for their context schema. To determine whether the values of another entity are relevant, the traversing agent utilizes the context proximity function with a predefined metric. If two values are in context proximity to each other, they are considered relevant and the other entity's relevant context is inserted into the local context schema. The traversing agent can also be used to search for information, but they operate on a best effort system since they are unable to guarantee that what was relevant when it was discovered will still be relevant when it becomes available for the application.

The maintenance agent is responsible for maintaining the relevance of the context schema, thus it is evaluating the relevance of the values and managing the subscriptions in order to have the most relevant and useful values from the other entities. The maintenance agents are required because the traversing agent only finds new relations, they do not keep the context values continuously updated. Hence, the maintenance agents are responsible for keeping the $I_n^{relevant}$ set in Eq. 3 continuously updated and accurate. This is achieved by evaluating the relevance of the values and the desired update frequency by the applications. We will however not specify in our realization how and what this relevance function will be. This because we believe there is currently no commonly accepted way for determining context relevance in a generalized scenario. For example, current implementations ranges all the way from simple Euclidean distances to complex ontologies.

4.4 Knowledge access and context views

There is one major problem associated with using the continuously changing context schema in applications, namely that it is unstable and inconsistent. For example, the values can change between the instant the search is made to that when it is received by the application, or the subscription notification might not yet have arrived even though the value has been updated at the source. We have, for example, measured that an accelerometer can produce over 300 values per second, but such a large amount of data is almost impossible to make use of in context-awareness scenarios if any type of communication is required. Therefore, to realize the application access layer we propose a front-end toward application developers which creates stable contextual views of the otherwise unstable context schema, as well as providing the means of searching the information model.

A context view can be seen as a stable version of the continuously changing context schema, and will be used in applications to retrieve and show an entity's current context. These contextual views are based on requirements and settings made by the application and will be created by taking a snapshot of the context schema at a particular point in time. This type of snapshot operation has a low cost in terms of resources and does not lock the context schema when long queries are required to be made. Furthermore, the snapshot operation allows the application developer to have more control over the quality-of-experience and resource consumption of their own application. For example, the developers can create multiple context views in a real-time manner in order to create continuous monitoring of the context. However, and indeed more importantly, they can opt to only create a new context view when it is actually required, in order to save resources on limited devices. Searching the information model will be solved by instantiating a new traversing agent with the single purpose of finding a specific type of information. This traversing agent will then return answers as they are found and the point in time for which the answer was valid, because the traversing agent can never fully complete the search because the values are changing faster than the speed at which the traversing agent can traverse the whole system.

5 Proof-of-concept implementation

We have verified our approach in several stages, starting from simple simulations of the agents in [30]. In those simulations we have shown an entity's evolving context schema, presented as a view containing the other entities that are in context proximity. The view was updated continuously as the entities moved around and as new entities entered or left the context proximity area. We have now adapted these simulations to created a fully functional system for continuously changing context schemes, for a potential real life deployment. The new proof-of-concept application is built using a COAP based wireless sensor network as a weak device, smartphones as powerful devices with attached sensors, the MediaSense platform as the communication architecture, a simplified context schema model with agents and a context view application interface that also supports searching.

In detail, we used Crossbow Telos B motes running TinyOS with COAP capabilities as a weak device and Android based smartphones (Galaxy Nexus) with built in sensors as powerful devices. We used the MediaSense platform [24] which provides an acceptable scaling peer-to-peer system for exchanging the context information, and which will even run on smartphones. In MediaSense all the nodes are connected in a ring based distributed hash table, but communicates in a peer-to-peer manner using a protocol based on reliable UDP. The entities thus form an Internet-of-Things overlay, on which they can exchange information. With this distributed hash table and peer-to-peer communication, the MediaSense platform addresses the requirement of avoiding central points of failure, logarithmic scaling, and peer-to-peer exchange of the actual information exchange. The information model was implemented as a simple object based schema of collected raw sensor values that was considered relevant. In detail, we choose to only take temperature and location under consideration in this proof-of-concept, but the idea is that even more sensors should only increase the accuracy of the relevance determination. From the information model we can query knowledge in the form of context views, the ability to set an actuator, search the whole information space, and provide a visualization of the data based on the location values. Figure 4 shows three screenshots with regards to how this information access layer was visualized on a mobile phone. These screenshots are taken from an actual running instance of the system on a real mobile device. In detail, these are from left to right: A simple context view, a search result listing, and a continuously changing map of the nodes with their values.

6 Evaluation

This section presents a validation and evaluation of the proposed approach, namely, the maintaining and evolving of schemes and adaptive views containing context information. The purpose of this evaluation is to show that the realized system can address the problems stated in Section 1. The methodology of the evaluation will be as follows, first we measure the response times of the system in order to achieve challenge 2 on global distribution and scalability. Following this we will measure the data usage of the applications using the system, to see how they improve in regards to challenge 1 on resource management. Thirdly, we will measure the application delay in order to observe how well the system handles continuously changing sensor values, highlighted in challenge 3. Finally we will study the propagation delay and search time, to evaluate challenge 4 on knowledge access in real-time.

The setup of the experiments is done using the proof-of-concept application with a combination of real and simulated end devices. In detail, the experiment setup contained two computers and one mobile device. One computer was acting as multiple virtual and simulated nodes, between 8 and 498 nodes on the single machine depending on the amount of nodes we decided to use. The second computer and the mobile phone were running a single node each, as a real device would. Giving the experiment setup a total

Fig. 4 Proof-of-Concept screenshots

of 500 connected nodes if all nodes are connected. Each node had three simulated sensor values (latitude, longitude, and temperature) which continuously moved in the range between 62.00 to 62.99, 17.00 to 17.99, and 20.00 to 20.99 respectively. The relevancy threshold value for all the sensor the sensors was set to 0.25. Furthermore, all the nodes both real and virtual, were connected to the same local 802.11g WiFi network. All the experiments were conducted between the mobile device and the computer running a single node, but measured on the mobile device side. Hence, it should be as close to the real world as possible and the only purpose of the other 498 nodes is simply to disturb the system with their overhead and thus to give an indication on the scalability of the system.

As previously stated, our first measurement was to find the time it takes to resolve an entity's name to an IP address and the act to retrieve a value in the system. Where we can observe how well the underlying platform performs, basically the peer-to-peer data exchange and its scalability. These results for 10, 100, 250, and 500 concurrent nodes in the system can be seen in Table 1 below.

Because the information will be stored and replicated across the end devices in the system, the context schemes will use up some of the available memory on the devices. In detail, the amount of stored data on each device in the system has been derived to scale according to Eq. 4.

Table 1 Measured response times for the communication platform

Nodes	10	100	250	500
Resolve time	36 ms	46 ms	53 ms	54 ms
Resolve stdev.	52 ms	67 ms	97 ms	68 ms
Data exchange time	33 ms	37 ms	47 ms	34 ms
Data exchange stdev	52 ms	69 ms	77 ms	42 ms

Where I_{stored} is the total stored information from both I_{local} and the remote information I_{remote} from relevant entities. Hence, the total stored information will, foremost, be based on the amount of remote information considered to be relevant. With 50 nodes in the system which are running the agents we have measured this be 69 entries on average with a standard deviation of 10 values. The amount of entries is however highly dependent on the chosen threshold value of the relevance function and can be optimized for specific applications.

$$I_{stored} = I_{local} + \sum_{n=0}^{relevant\ entities} n * I_{remote} \tag{4}$$

The application delay is the delay to access a snapshot of the evolving context schema. The intention is that if the agents have effectively gathered the relevant information, the sought after information is in a snapshot of the evolving context schema. This means that the delay for acquiring the context view will be very short because no communication over the Internet will be required, which can be seen in Eq. 5. Where the application delay $D_{application}$ is only the delay for making a snapshot of the schema d_{schema}. In detail we have measured this on the mobile device with 50 nodes in the system running agents, and it resulted in a 3.1 ms per snapshot with a standard deviation of 13.1 ms.

$$D_{application} = d_{schema} \tag{5}$$

The propagation delay is the time between the information changes at the source and when it is received at the sink. Because this is achieved by means of subscriptions in the realized architecture, it can be derived that it will be based according to Eq. 6. Where it will only be based on the transmission delay $d_{transmission}$ and the time to insert it into the schema d_{schema}. In detail, we have calculated from the data exchange and the application delay to be 40 ms with

a standard deviation of 39 ms, with 50 nodes in the system running the agents.

$$D_{propagation} = d_{transmission} + d_{schema} \quad (6)$$

We are also able to study the searching aspect of the information model, conducted by traversing the system. However, because of the continuous changes of the information it can never be determined when a search has been fully completed, thus it is only possible to measure the time between starting the search up to the point of finding the first value. With 50 nodes in the system running agents this was measured to take 1900 ms with a standard deviation of 2100 ms from the start of the search to the time that the first relevant answer has been returned. This because, the keyword that was being searched for only existed temporarily on about 20 % of the entities at any given time.

7 Impact of continuously changing context

In this section we will summarize our findings in relation to the potential impact that continuously changing contextual information will have on the Internet-of-Things and its related research. The early successes of applications based on Internet-of-Things ideas have hidden many of the underlying problems. They have often relied on centralized solutions which can often prove to be a hinder in terms of global and ubiquitous access, hence our initial challenges 1 and 2 back in Section 1. However, in contrast, the benefit of centralization is the simple and quick development process, for which centralized systems are easy to manage. The early successes can be seen as walled gardens or independent silos of potentially useful information, but which are locked down from outside access because of the unwillingness to share information without proper economic compensation. One typical example involves smart home systems of which many exist, but for which cooperation has proved very difficult to achieve because the different smart home systems are built by different manufacturers. This walled garden mindset is not applicable for the future Internet-of-Things for which the desire will be to build cooperative systems in order to take advantage of all the contextual information that exists. It is necessary to determine a method to solve this, in a similar manner to the means by which web services have appeared to address the necessity to create mash-up services between different walled gardens with individual commercial interests.

We expect that some form of cloud based storage will be used in the future, particularly as it is currently the most used solution for global deployment in relation to handling large amounts of users. However, the cloud based solutions can still be considered as a centralized walled garden solution which inherently hinders the flow of information. The continuously changing information from challenge 2 is also inherently not suitable for normal database storage, because it involves more of a flow of information rather than distinct values. The current predictions are of the order of 50 billion connected devices by the year 2020, and thus we have already highlighted the problem which could occur if everyone wanted to share and exchange information between each other. We see this as a potential information overload which will be beyond the capabilities of our communication technologies and resources, especially if we expect to keep costs down and balance it with good application experiences which was stated in challenge 4. We therefore foresee a combination of distributed peer-to-peer based solutions for the optimal exchange of information and cloud powered back ends for persistence and reasoning.

The majority of current Internet-of-Things applications have an extreme focus on location as being one of the most important aspects for defining a situation, but this will probably not be the case for all future applications. However, future applications want to provide good quality-of-experience for their end users, how they obtained the relevant information and how it has been retrieved may be viewed as being of little interest. Therefore we can apply a wide range of techniques for creating the best possible experience, as long as it is hidden from the end user. We are currently researching more details into this quality-of-experience aspects of applications on the Internet-of-Things, such as when an application is being perceived as running in real-time and finding indications on the threshold of certain parameters in regards to quality-of-experience. But we believe there is still much do be done in the this specific area of Internet-of-Things applications. Furthermore, to separate the relevant information from the irrelevant in the information overloaded dataset on the Internet-of-Things will also be a difficult task to solve. We have discovered several approaches for achieving this task, such as intelligent prediction of context values using Markov chains, the ranking of context information in a similar way as search engines, and creating interests groups based on contextual relations. However, one of the most vital question still remains from challenge 3, namely how contextual relevance can be determined and, in particular, how is it possible to evaluate fuzzy and nonlinear values for context.

Our proposed solution is not a universal solution and will not solve all problems, but we believe it is a good start and should be able to solve applications in the following areas: Opportunistic relations applications (ex. vehicle to vehicle), monitoring applications (ex. catastrophes and health-care), crowd sourcing (ex. collaborative sensing), intelligent home (ex. controlling actuators), logistics (ex. intelligent packaging), and more. The major limitations of our current platform is related to managing the spread of private data, the will to share sensor information, and the integrity of

users. Further limitations of our solution is the heightened responsibilities of each end device and thus the impracticalities of persistently storing data. There is also limitations to the currently implemented information model, where the applied relevance function and intelligence algorithm is very simple.

8 Conclusion

This article proposed a method for creating intelligent and pervasive applications using continuously changing context information from the Internet-of-Things. In this regard, we created a realized model which is capable of enabling applications to make use of the continuously changing information. Our system utilizes a per entity unique context object, called context schema, which will be continuously updated, exchanged, and evaluated. These schemes were evaluated using a context relevance function and could limit the exchange based on relevance instead of the update frequency of the information sources, in order to save resources. Our method was also evaluated by creating a proof-of-concept prototype, using continuously changing information from multiple devices. This article has also presented and evaluated the potential impact that continuously changing context values will have for future applications based on the Internet-of-Things.

In detail, we have addressed the problems defined in Section 1. Challenge 1 regarding adapting resource consumptions based on application demands was addressed to the degree that we have applied low cost protocols, information flow control, and our agent based approach to limit the resource consumption in comparison to uncontrolled sensor sharing. Challenge 2 regarding finding mechanisms in order to manage global data flows, was addressed to the degree that our realized model with a fully distributed peer-to-peer based communication architecture also has the ability to scale well with the number of entities. Challenge 3 regarding finding an information model for the Internet-of-Things was addressed using the context schemes and context proximity evaluations. This works in satisfactory manner for simple reasoning, but problems do still exist in relation to solving the relevance in the general case. Challenge 4 regarding knowledge access was addressed using adaptive views which only contained relevant information based on the situation. These are basically stable views of the continuously changing context schema, thus the problem is solved to the degree that it provides an effective and low cost mean of accessing the continuously changing data.

Our future work includes evaluating how to determine the quality-of-experience for different Internet-of-Things based applications and to further manage the resource limitations based on user satisfaction. We will also continue to develop the algorithm and investigate further into a general approach for optimizing resource consumption to ensure longevity of applications on limited devices. finally, we will explore in a greater depth contextual reasoning, ranking of context information, intelligent grouping, and information prediction.

References

1. Hong J, Suh E, Kim S (2009) Context-aware systems: a literature review and classification. Expert Syst Appl 36(4):8509–8522
2. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805
3. Ericsson (2011) More than 50 billion connected devices. White Paper. [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf
4. Bricon-Souf N, Newman C (2007) Context awareness in health care: a review. Int J Med Inform 76(1):2–12
5. Chan M, Campo E, Estève D, Fourniols J (2009) Smart homes-current features and future perspectives. Maturitas 64(2):90–97
6. Bellavista P, Kupper A, Helal S (2008) Location-based services: back to the future. IEEE Pervasive Comput 7(2):85–89
7. Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E (2002) Wireless sensor networks: a survey. Comput Netw 38(4):393–422
8. Wu G, Talwar S, Johnsson K, Himayat N, Johnson K (2011) M2M: from mobile to embedded internet. IEEE Commun Mag 49(4):36–43
9. Lane N, Miluzzo E, Lu H, Peebles D, Choudhury T, Campbell A (2010) A survey of mobile phone sensing. IEEE Commun Mag 48(9):140–150
10. Bollier D, Firestone C (2010) The promise and peril of big data. Aspen Institute, Communications and Society Program
11. Pautasso C, Zimmermann O, Leymann F (2008) Restful web services vs. big'web services: making the right architectural decision. In: Proceedings of the 17th international conference on World Wide Web, ACM, pp 805–814
12. Han J, Haihong E, Le G, Du J (2011) Survey on nosql database. In: Pervasive computing and applications (ICPCA), 2011 6th international conference on IEEE, pp 363–366
13. Antoniou G, Van Harmelen F (2009) Web ontology language: Owl. In: Handbook on ontologies. Springer, pp 91–110
14. Bettini C, Brdiczka O, Henricksen K, Indulska J, Nicklas D, Ranganathan A, Riboni D (2010) A survey of context modelling and reasoning techniques. Pervasive Mob Comput 6(2):161–180
15. Botts M, Robin A (2007) Opengis sensor model language (sensorml) implementation specification, OpenGIS implementation specification OGC, Tech. Rep.
16. Kansal A, Nath S, Liu J, Zhao F (2007) Senseweb: an infrastructure for shared sensing. IEEE MultiMed 14(4):8–13
17. Presser M, Barnaghi P, Eurich M, Villalonga C (2009) The SENSEI project: integrating the physical world with the digital world of the network of the future. IEEE Commun Mag 47(4):1–4
18. Project Serenoa (2011) Multi-dimensional context-aware adaptation of service front-ends context aware design space and context aware reference framework FP7 ICT 258030 Deliverable 2.1.1
19. Klemettinen M (2007) Enabling technologies for mobile services. The mobiLife book. Wiley
20. Raz D, Juhola A, Serrat-Fernandez J, Galis A (2006) Fast and efficient context-aware services. In: Hutchison D (ed). Wiley, Chichester West Sussex

21. Koumaras H, Negrou D, Liberal F, Arauz J, Kourtis A (2008) ADAMANTIUM project: enhancing IMS with a PQoS-aware multimedia content management system. Int Conf Autom Qual Test Robot 1:358–363
22. TISPAN ETSI. [Online]. Available: http://www.etsi.org/tispan/
23. Camarillo G, Garcia-Martin M-A (2007) The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds. Wiley
24. Kanter T, Forsström S, Kardeby V, Walters J, Jennehag U, Österberg P (2012) Mediasense–an internet of things platform for scalable and decentralized context sharing and control in *ICDT 2012*. In: The seventh international conference digital telecommunications, pp 27–32
25. Toninelli A, Pantsar-Syvniemi S, Bellavista P, Ovaska E (2009) Supporting context awareness in smart environments: a scalable approach to information interoperability. In: Proceedings of the international workshop on middleware for pervasive mobile and embedded computing, ACM, pp 1–4
26. Bellavista P, Montanari R, Tibaldi D (2003) Cosmos: a context-centric access control middleware for mobile environments. In: Mobile agents for telecommunication applications. Springer, pp 77–88
27. Upton E, Halfacree G (2012) Meet the Raspberry Pi. Wiley
28. ETSI (2011) Machine-to-machine communications m2m; functional architecture TS 102 690 V1.1.1, Tech.Rep.
29. Shelby Z, Hartke K, Bormann C (2013) Constrained application protocol (COAP) [Online]. Available: http://tools.ietf.org/html/ietf-core-coap-14.txt
30. Forsström S, Kanter T (2011) Enabling continuously evolving context information in mobile environments by utilizing ubiquitous sensors. In: Mobile networks and management: third international conference