Context-Aware Environments for the Internet of Things

Valentin Cristea, Ciprian Dobre, and Florin Pop

Abstract. This chapter discusses the challenges, state of the art, and future trends in context aware environments (infrastructure and services) for the Internet of Things, which is defined as a world-wide network of uniquely identified self-organized and intelligent things. Intelligence means primarily the capability of things to be aware of the context in which they operate (time, geographic location, geographic dimension, situation, etc.) and to inter-cooperate with other things in the environment. The Chapter is structured in three sections. The first section, which frames the issues discussed in the rest of the chapter, is a systematic presentation of the most relevant concepts and aspects related to the infrastructure and services for the Internet of Things. The second section presents relevant research works in the infrastructure, and up to date solutions and results regarding the infrastructure and services. The third section presents future trends and research directions in the domain.

Keywords: Context-Aware Environments, Services, Collective Intelligence, Scalability, High-Performance, Internet of Things.

1 Introduction to Internet of Things Infrastructure and Services

This section, which frames the issues discussed in the rest of the chapter, is a systematic presentation of the most relevant concepts, aspects and main issues related to the context-aware infrastructure and services for the Internet of Things (IoT). For the purpose of this chapter we adopt the IoT definition presented in [5]:

"The Internet of Things (IoT) is an integrated part of the Future Internet and could be defined as a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network. In the IoT, "things" are expected to become active participants in business, information and social processes where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information "sensed" about the

Valentin Cristea · Ciprian Dobre · Florin Pop
University *Politehnica* of Bucharest, Computer Science Department
Splaiul Independentei 313, Bucharest 060042, Romania
e-mail: {valentin.cristea,ciprian.dobre,florin.pop}@cs.pub.ro

environment, while reacting autonomously to the "real/physical world" events and influencing it by running processes that trigger actions and create services with or without direct human intervention. Interfaces in the form of services facilitate interactions with these "smart things" over the Internet, query and change their state and any information associated with them, taking into account security and privacy issues."

As part of the Future Internet, IoT aims to integrate, collect information from-, and offer services to a very diverse spectrum of physical things used in different domains. "Things" are everyday objects for which IoT offers a virtual presence on the Internet, allocates a specific identity and virtual address, and adds capabilities to self-organize and communicate with other things without human intervention. To ensure a high quality of services, additional capabilities can be included such as context awareness, autonomy, and reactivity.

This section starts with an introductory presentation of things and IoT infrastructure and continues with the main functional aspects related to things' intercommunication, the context model for the IoT and the event-driven mechanisms to sense, process, and exchange context data. Non-functional requirements for the IoT infrastructure are described in the end.

Things are very diverse. Simple things, like books, can have Radio Frequency Identification - RFID tags that help tracking them without human intervention. For example, in an electronic commerce system, a RFID sensor network can detect when a thing leaves the warehouse and can trigger specific actions like inventory update or customer rewarding for buying a high end product [1]. In this simple case, RFIDs enable the automatic identification of things, the capture of their context (for example the location) and the execution of corresponding actions if necessary. Sensors and actuators are used to transform real things into *virtual objects* [3] [5] with digital identities. In this way, things may communicate, interfere and collaborate with each other over the Internet [6]. Adding part of application logic to things transforms them into *smart objects* [15], which have additional capabilities to sense, log and understand the events occurring in the physical environment, autonomously react to context changes, and intercommunicate with other things and people. A tool endowed with such capabilities could register when and how the workers used it and produce a financial cost figure. Similarly, smart objects used in the e-health domain could continuously monitor the status of a patient and adapt the therapy according to monitoring results. Smart objects can also be general purpose portable devices like smart phones and tablets, that have processing and storage capabilities, and are endowed with different types of sensors for time, position, temperature, etc. Both specialized and general purpose smart objects have the capability to interact with people.

The IoT includes a hardware, software and services infrastructure for things networking. IoT infrastructure is event-driven and real-time, supporting the context sensing, processing, and exchange with other things and the environment. The infrastructure is very complex due to the huge number (50 to 100 trillion) of heterogeneous, (possibly) mobile things that dynamically join and leave the IoT, generate and consume billions of parallel and simultaneous events geographically distributed all over the world. The complexity is augmented by the difficulty to represent, interpret, process, and predict the diversity of possible contexts. The infrastructure must have important characteristics such as reliability, safety, survivability, security and fault tolerance. Also, it must manage the communication, storage and compute resources.

The IoT infrastructure supports communication among things. This function must be flexible and adapted to the large variety of things, from simple sensors to sophisticated smart objects. More specific, things need a communication infrastructure that is low-data-rate, low power, and low-complexity. Actual solutions are based on short-range radio frequency (RF) transmissions in ad-hoc wireless personal area networks (WPANs). A main concern of IoT infrastructure developers is supporting heterogeneous things [42] by adopting appropriate standards for the physical and media access control (MAC) layers, and for communication protocols. The protocol and compatible interconnection for the simple wireless connectivity with relaxed throughput (2 – 250 kb/s), low range (up to 100 m), moderate latency (10 – 50 ms) requirements and low cost, adapted to devices previously not connected to the Internet were defined in IEEE 802.15.4 [7]. Other similar efforts refer to industrial and vehicular applications - ZigBee [8], open standards for process control in industrial automation and related applications - ISA100.11a [9] and WirelessHART [10], and encapsulating IPv6 datagrams in 802.15.4 frames, neighbor discovery and routing that allow sensors to communicate with Internet hosts - 6LoWPAN [11]. The scope of IoT specialists is the worldwide network of interconnected virtual objects uniquely addressable and communicating through standard protocols.

The IoT architecture supports physical things' integration in Internet and the complex interaction flow of services triggered by event occurrences. Objects identification, sensing and connecting capabilities form the basis for the development of independent cooperative services and applications that address key features of the IoT architecture: Service Orientation, Web-base, distributed processing, easy integration via native XML and SOAP messaging, component-base, open access, N-tiered architecture, support for vertical and horizontal scalability [13]. Specific Web services help the physical objects to "become active participants in business processes" [14]. They interact with the corresponding virtual objects over the Internet, query and change objects' state, and process other associated information. The new key features for the IoT architecture include persistent messaging for the highest availability, complete security and reliability for total control and compliance, platform independence and interoperability (more specific for middleware).

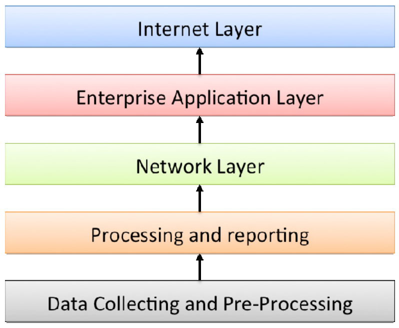

Internet Layer
Enterprise Application Layer
Network Layer
Processing and reporting
Data Collecting and Pre-Processing

Fig. 1 Layered Networking for IoT

IoT infrastructure considers extended process functionality, pathways and layered networks as main components. These layers (see Fig. 1) refer to: real-time Data Collecting and Pre-Processing, which aims to support any available or emerging technology and offer uniform interfaces to upper layers; Processing and reporting environment data according to specific rules and data packing; efficient data transportation over the Network; support for Enterprise Application development; and application exposure and integration in the Internet. The IoT infrastructure supports object-connected technologies for "Human-to-Objects" and "Objects-to-Objects" communications [2] [4]. The communication platforms are heterogeneous, ad-hoc, and opportunistic.

As mentioned previously, IoT is a large heterogeneous collection of things, which differ from each other. Even things that have the same nature, construction, and properties can differ from one another by their situation or context. Context means the conditions in which things exist in other words their surrounding world. Since virtual things in IoT are interconnected, the meaning of the data they exchange with other things and people becomes clear only when it is interpreted in the thing's context. This is why the IoT infrastructure runs reliably and permanently to provide the context as a "public utility" to IoT services [31]. For human users, the context is the information that characterizes user's interaction with Internet applications plus the location where this interaction occurs, so that the service can be adapted easily to users' preferences, For things, we need another approach. A very suggestive example is given in [33]. The authors explain the case of a plant that is the target of an automatic watering service. In order to control the watering dosages and frequency, the service has to sense the dryness status of the plant, to use the domain knowledge of plants and find their watering "preferences", and to ask the weather prediction service about the chances of rain in the next days. So, the context of a thing includes information about thing's environment and about the thing itself [33].

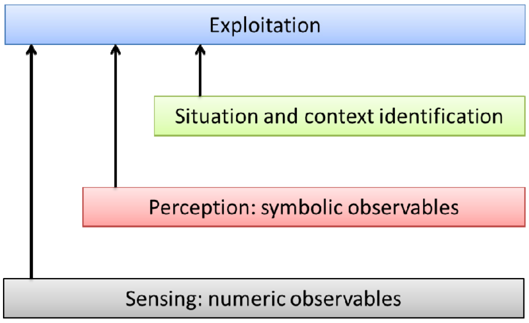

Exploitation
Situation and context identification
Perception: symbolic observables
Sensing: numeric observables

Fig. 2 Context-aware services

Several context modeling and reasoning techniques are known today [34], some of them being based on knowledge representation and description logics. Ontology-based models can describe complex context data, allow context integration among several sources, and can use reasoning tools to recognize more abstract contexts. Ontologies provide a formal specification of the semantics of context data that stay at the base of knowledge sharing among different things in IoT. In addition, ontological reasoning can derive new context. Ontology-based models can be used to organize IoT infrastructure context-aware services as a fabric structured into multiple levels of abstraction (see Fig. 2) starting with collecting information from physical sensors (called low level context), which could be meaningless and consequently not useful to applications. Next, higher-level context is derived by reasoning and interpretation. Finally, context is exploited by triggering specific actions [31].

The IoT infrastructure combines the context model with event-based organization of services that support the collection, transmission, processing, storage and delivery of context information. In the event-driven architecture vocabulary, events are generated by different sources, event producers, when for example a context change is significant and must be propagated to target applications, which are event consumers. Producers and consumers are loosely coupled by the asynchronous transmission and reception of events. They don't have to know and explicitly refer each other. In addition, the producers don't know if the transmitted events are consumed ever. A publish/subscribe mechanism is used to offer the events to the interested consumers. Other components are the event channels for communication, and event processing engines for complex event detection. Components for event specification, event management, and for the integration of the event-driven system with application belong also to the IoT infrastructure.

There are also non-functional requirements associated with IoT infrastructure [1]: large scale integration, interoperability between heterogeneous things, fault tolerance and network disconnections, mobility, energy saving, reliability, safety, survivability, protection of users' personal information (e.g., location and preferences) against security attacks, QoS and overall performance, scalability, self-* properties and transparency.

Development issues for IoT infrastructure are directly related to Service Oriented Architecture (SOA), Collaborative Decision Making (CDM), Cloud Computing, Web 2.0 (and Future Internet) and Semantic Web. The support of 6A connectivity (Anything, Anyone, Anytime, Any Place, Any Service, and Any Network) becomes the most important key feature for adding sense to the Internet of Things [13].

2 Context Aware Internet of Things Infrastructure

This section presents up to date solutions and research results regarding the structure, characteristics, and services for context aware IoT infrastructure.

Sensors in IoT are used to collect various data such as biomedical information, environment temperature, humidity, and ambient noise level. The data provided by such sensors can be used by customized context-aware applications and services, capable to adapting their behavior to their running environment. However, sensor data exhibits high complexity (e.g., due to the huge volumes and interdependency relationships between sources), dynamism (e.g., updates performed in real-time and data that can age until it becomes useless), accuracy, precision and timeliness. An IoT system should not concern itself with the individual pieces of sensor data: rather, the information should be interpreted into a higher, domain-relevant concept. For example, sensors might monitor temperature, humidity, while the information needed by a watering actuator might be that the environment is dry. This higher-level concept is called a situation, which is an abstract state of affairs interesting to applications [19].

2.1 Situational Awareness

Situations are generally representations (simple, human understandable) of sensor data. They shield the applications from the complexities of sensor readings, sensor data noise and inferences activities. However, in large-scale systems there may be tens or hundreds of situations that applications need to recognize and respond to. Underlying these situations will be an even greater number of sensors that are used in situation identification. A system has a significant task of defining and managing these situations. This includes capturing what and how situations are to be recognized from which pieces of contexts, and how different situations are related to each other. The system should know, for example, which situations can or cannot occur: a room hosting a "scientific event" and an "academic class" at the same time); otherwise, inappropriate adaptive behavior may occur. Temporal order between situations is also important, such as the inability of a car to go directly from a situation of 'parked' to 'driving on a highway'. Given the inherent inaccuracy of sensor data and the limitations of inference rules, the detection of situations is imperfect.

The research topics on situation identification for IoT involve several issues [20]. First, *representation* deals with defining logic primitives used to construct a situation's logical specification. In representation, logical primitives should capture features in complex sensor data (e.g., acceleration data), scope knowledge (e.g., a spatial map or social network), and different relationships between situations. Also, an IoT system is assumed to be highly dynamic. New sensors can be introduced, that bring new types of context. Therefore, the logical primitives should be flexibly extensible, such as new primitives to not cause modifications or produce ambiguous meanings to existing ones.

Specification deals with defining the logic behind a particular situation. This can be acquired by experts or learned from training data. It typically relies on a situation model with a priori expert knowledge, on which reasoning is applied based on the input sensor data. For example, in logic programming [37] the key underlying assumption is that knowledge about situations can be modularized or digitized. An example adapted from [21], which defines a situation when a room is detected as being occupied, can be specified as follows:

```
Occupied(room) = ∃t1, t2, event|reservedBy(person, t1, t2)
        •((t1 ≤ timenow() ∧ (timenow() ≤ t2 ∨ isnull(t2)))
        ∨((t1 ≤ timenow() ∨ isnull(t1)) ∧ timenow() ≤ t2))
```

These initial works have been advanced to a more formal approach by Loke et al. [22]. The authors proposed a declarative approach to represent and reason with situations at a higher level of abstraction. The approach uses logical programming in Prolog to embed situations. For example a situation `in_meeting_now` of a user entity E is defined on two situations `with_someone_now` and `has_entry_for_meeting_in_diary` can be defined as:

```
if in_meeting_now(E) then
        with_someone_now(E),
        has_entry_for_meeting_in_diary(E).
if with_someone_now(E) then
        location*(E, L), people_in_room*(L, N) , N > 1.
if has_entry_for_meeting_in_diary(E) then
        current_time*(T1) ,
        diary*(E, 'meeting', entry(StartTime, Duration)),
        within_interval(T1, StartTime, Duration).
```

Each of these situations is defined on sensor predicates. For example, `with_someone_now` refers to two sensor predicates: `location*(E, L)` that returns the location of the entity, and `people_in_room*(L, N)` that returns the number of people in the location. These predicates refer to lower-level context, as detected by various sensors. In this way, situation programs can be made amenable to formal analysis, and the inference procedure of reasoning about situations is decoupled from the acquisition procedure of sensor readings.

2.2 Other Representation Approaches

Other logic theories, such as situation calculus [38], have also been used to infer situations in IoT systems. Kalyan et al. [23] introduce a multi-level situation theory, where an intermediate level micro situation is introduced between infons and situations. An infon embodies a discrete unit of information for a single entity (e.g., a customer or a product), while a situation makes certain infons factual and thus supports facts. Micro situations are composed of these entity-specific infons, which can be explicitly obtained from queries or implicitly derived from sensors and reasons. Situations are considered as a hierarchical aggregation of micro situations and situations. This work aims to assist information reuse and support ease of retrieving the right kind of information by providing appropriate abstraction of information.

Spatial and temporal logic has also been applied to represent and reason on spatial and temporal features and constraints of context and situations. Augusto et al. [24] introduce the temporal operators ANDlater and ANDsim in Event–Condition–Action rules, upon which temporal knowledge on human activities can be specified. Considering the sensor events `at_kitchen_on` (the activation of the RFID sensors in the kitchen), `tkRK_on` (the activation of the RFID sensor while the user is passing through the door between the kitchen and the reception area), and `no_movement_detected` (sensing of no movement), the following rule specifies a situation of a user 'fainting':

```
   IF at_kitchen_on ANDlater tdRK_on ANDlater no_movement_detected
   THEN assume the occupant has fainted
```

Also, ontologies have increasingly gained attention as a generic, formal and explicit way to capture and specify the domain knowledge with its intrinsic semantics through consensual terminology and formal axioms and constraints. They provide a formal way to represent sensor data, context, and situations into well-structured terminology. Based on the modeled concepts, developers can define logical specifications of situations in rules. An exemplar rule on an activity 'sleeping' is given in [25]:

```
   (?user rdf:type socam:Person),
   (?user, socam:locatedIn, socam:Bedroom),
   (?user, socam:hasPosture, 'LIEDOWN'),
   (socam:Bedroom, socam:lightLevel, 'LOW'),
   (socam:Bedroom, socam:doorStatus, 'CLOSED')
   -> (?user socam:status 'SLEEPING')
```

Ontologies, together with their support for representation formalisms, can support reasoning, including detecting inconsistency, or deriving new knowledge. An ontological reasoner can be used to check consistency in a class hierarchy and consistency between instances, e.g. whether a class is being a subclass of two classes that are declared as disjoint or whether two instances are contradictory to each other (such as a person being detected in two spatially disjoint locations at the same time). Given the current sensor data, the reasoner will derive a new set of statements. In the above 'sleeping' example, if the reasoner is based on a forward-chaining rule engine, it can match the conditions of this rule against the sensor input. If all the conditions are satisfied, the reasoner will infer the conclusion of the rule. The reasoning will terminate if the status of the user is inferred, when the status of the user is set to be the default inference goal in this reasoner.

Other solutions are based on the Dempster–Shafer theory (DST) [39], a mathematical theory of evidence, which propagates uncertainty values and consequently provides an indication of the certainty of inferences. The process of using DST is described as follows. First, developers apply expert knowledge to construct an evidential network that describes how sensors lead to activities. The left-hand side of Fig. 3 below describes that the sensors on the cup and fridge are connected to context information (e.g., 'cup used'). Such context information can be further inferred or composed to higher-level context. The composition of context information points to an activity (e.g., 'Get drink') at the top. Developers can use such an approach to determine the evidence space and degree of belief in an evidence. For example, in Fig. 3, the values on the arrows represent the belief in particular sensor (also called the uncertainty of sensor observations). Generally, in reasoning situations are inferred from a large amount of imperfect sensor data. In reasoning, one of the main processes is called situation identification - deriving a situation by interpreting or fusing several pieces of context in some way. Specifying and identifying situations can have a large variability depending on factors such as time, location, individual users, and working environments [26]. This makes specification-based approaches relying on models of a priori knowledge impractical to use. Machine learning techniques have been widely applied to learning complex associations between situations and sensor data. However, the performance of reasoning is usually undermined by the complexity of the underlying sensor data.

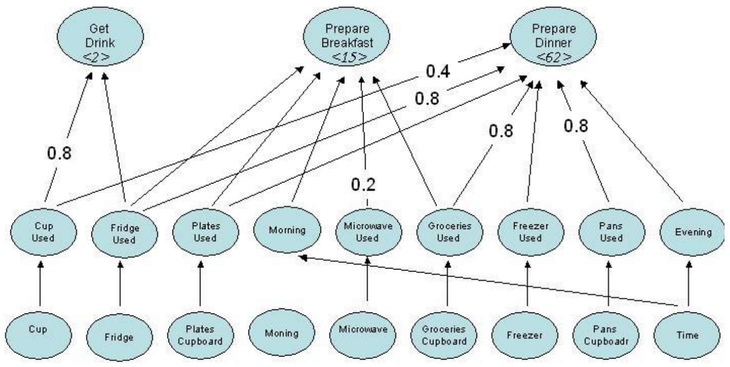


Fig. 3 An example of situation inferring using Dempster-Shafer theory (from [40])

Bayesian networks have been applied in many context-aware systems. For example, in [27] the authors present a solution to infer a user's current activity from sensors within a room, which provide several contexts (e.g., several sensors track people within a house, the light level and noise level in rooms are monitored by other sensors). But, such a Bayesian network cannot model the causal relationship between the status context of one particular user, his/her location, and the status of the micro oven for example, because of the breaking of the independence assumption in naïve Bayes. A better approach consists in the use Hidden Markov Models (HMMs) [28]. HMMs statistical models where a system being modeled is assumed to be a Markov chain that is a sequence of events. A HMM is composed of a finite set of hidden states and observations that are generated from states. For example, a HMM where each state represents a single activity (e.g., 'prepare dinner', 'go to bed', 'take shower', and 'leave house') is presented in [28]. They represent observations in three types of characterized sensor data that are generated in each activity, which are raw sensor data, the change of sensor data, the last observed sensor data, and the combination of them. The HMM is trained to obtain the three probability parameters, where the prior probability of an activity represents the likelihood of the user starting from this activity; the state transition probabilities represent the likelihood of the user changing from one activity to another; and the observation emission probabilities represent the likelihood of the occurrence of a sensor observation when the user is conducting a certain activity.

Unlike this approach, [29] employs the use of neural networks in learning activities (e.g., static activities like 'sitting' and 'working at a PC', and dynamic activities like 'running' and 'vacuuming') from acceleration data. A similar idea was further applied to detect bump holes as cars runs on street, using accelerometer sensors inside the vehicle [41]. The acceleration data is collected from a wireless sensing tri-axial accelerometer module, from which eight features are extracted, including the mean value, the correlation between axes, and the energy that is used to discriminate between activities.

Finally, Support Vector Machines (SVM) [12] is a relatively new method for classifying both linear and nonlinear data. An SVM uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyper-plane that separates the training data of one class from another. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated. SVMs are good at handling large feature spaces since they employ over fitting protection, which does not necessarily depend on the number of features. Kanda et al [30] use SVMs to categorise motion trajectories (such as 'fast', 'idle', and 'stop') based on the velocity, direction, and shape features extracted from various sensors (within a car for example). Different types of sensor data lead to different techniques to analyze them. Numerical data, for example, can be used to infer motions like 'walking' or 'running' from acceleration data. Situation identification at this level is usually performed in learning-based approaches, which uncover complicated associations (e.g., nonlinear) between continuous numerical data and situations by carving up ranges of numerical data (e.g., decision tree) or finding an appropriate algebraic function to satisfy or 'explain' data (e.g., neural networks or SVMs). Specification-based approaches can apply if the association between sensor data and situations are rather explicit and can be represented in logic rules. Situations can also be recognized from categorical features; for example, inferring a room's situation - 'meeting' or 'presentation' — from the number of persons co-located in the room and the applications running in the computer installed in the room. This higher-level of situation identification can be performed using either specification- or learning-based approaches.

Uncertainty can also exist in the use of oversimplified rules that are defined in an ad hoc way. In representing uncertainty of rules, Web Ontology Language (OWL), a family of knowledge representation languages for authoring ontologies endorsed by W3C, can be extended with a conditional probabilistic class to encode the probability that an instance belongs to a class respectively given that it belongs to another class. Although good at expressing uncertainty, these qualitative approaches need to be combined with other machine-learning techniques to quantify the uncertainty to be used in situation identification. Learning-based approaches have a stronger capability to resolve uncertainty by training with the real-world data that involves noise. These approaches not only learn associations between sensor data and situations, but also the effect that the uncertainty of sensor data has on the associations. For example, the conditional probabilities learned in Bayes networks include the reliability of sensor data as well as the contribution of the characterized sensor data in identifying a situation.

2.3 Architectural Issues

A popular architectural model for IoT is composed of autonomous physical/ digital objects augmented with sensing, processing, and network capabilities.

Unlike RFID tags, smart objects carry an application logic that let them sense their local situation and interact with the environment through actuators. They sense, log, and interpret what's occurring within themselves and the work, act on their own, intercommunicate with each other, and exchange data [17].

According to the scenarios illustrated in [17], the architectural differences in the way smart objects understand (sense, interpret or react to) events, and interact with their environment in terms of input, output, control and feedback, classify them as either activity-aware objects, policy-aware objects or process-aware objects. A process-aware object represents the most accomplished type, and characterizes: awareness (a process-aware object understands the operational process that is part of and can relate the occurrence of real-work activities and events to these processes), representation (its model consists of a context-aware workflow model that defines timing and ordering or work activities), and interaction (a process-aware object providers workers with context-aware guidance about tasks, deadlines, and decisions).

Adaptable Pervasive Flows [18] is a technology that model applications in a fashion similar to classical service workflows, while being situated in the real world. A flow is a computer-based model that essentially consists of a set of actions, glued together by a plan (or control flow) which defines how the actions should be performed to achieve some goal under a set of constraints. Flows are explicitly tailored (1) to being executed in pervasive environments, and (2) to being adaptable. They are situated in the real world, i.e., they are logically attached to entities, they can move with them through different contexts. While they are carried along, they model the behavior intended for the associated entity, and adapt the entity's environment to this behavior. Thus, when a mobile user carries a flow that specifies his prospective actions, the pervasive computing machinery in his environment will be set up for him by the flow. Since people may change their minds, and since artifacts and people may be subject to changes in the environment, the flow itself may also adapt to reflect such changes. This requires flows to be context-aware. They can take into account the context pertaining to their entity's current environment as well as the entity's actual activities in order to dynamically adapt to changing situations.

A context-aware infrastructure designed to support smart objects could help workers in construction industry by providing just-in-time information about required work activities [17]. To model the organizational process a workflow [18] can be used to define work activities in which the smart object is involved. Such a workflow can contain activities and transitions between activities. Transitions can be annotated with context conditions that refer to sensor or human input. A workflow continues along a transition if input satisfies a condition.

The goal of JCAF [16] is to create a general-purpose, robust, event-based, service-oriented infrastructure and a generic, expressive Java programming framework for the deployment and development of context-aware IoT applications. The infrastructure is composed of a Context-awareness Runtime Infrastructure and a Context-awareness Programming Framework. Each Context Service is a long-lived process analog to a J2EE Application Server. The service's Entity Container manages an Entity with its Context information. An entity is a small Java program that runs within the Context Service and responds to changes in its context. The life cycle of an entity is controlled by the container in which the entity has been added. The entity container handles subscribers to context events and notifies relevant clients on changes to entities.

2.4 CAPIM Infrastructure

An infrastructure that follows these concepts and the hierarchical view of functions is presented in CAPIM [2], a platform for context aware IoT systems [31] that integrates smartphones for students and staff interactions in a university campus. The platform collects and manages a global context (of the surrounding space) by integrating capabilities of various sensors and actuators. Such sensors are aggregated using the sensing, processing and communication capabilities of smart objects, in particular smartphones. Smart objects can support the integration in Internet of parking lots, university restaurants, libraries, classrooms, administrative offices, etc. and can communicate with each other and with smartphones for their exploitation in collaborative e-services dedicated to students and staff. Since smartphones become commodity hardware, used almost everywhere, having more sensing and computing capabilities in every-day situations is attractive for many reasons. Smartphones can sustain next-generation efforts of making the Internet of Things vision a reality – users and devices blend together smoothly in a single virtual world where smartphone, coupled with other sensors and services from the environment, can optimize (e.g. by helping organizing tasks, contacts, etc.) and assist (e.g. with navigation, find information more quickly, access online data, etc.) users in everyday activities. These may refer to finding a vacant parking space in a parking lot that is closer to user's office, classroom or actual car position, assisting people parking and pay for parking, finding the best way towards a specific classroom, getting information about the activity in that room or about the number of people who are actually there, being notified about new publications available in the library or about the actual menu of the preferred restaurant, etc. This is a shift towards developing mobile context-aware services that are capable to recognize and pro-actively react to user's own environment. Such context-aware mobile applications can help things better interact between themselves and with their surrounding environments, and offer high quality information to people. This is the basis for a paradigm where the context is actively used by applications designed to take smarter and automated decisions: start the cooling system when the temperature raises above a specific threshold and there is a meeting in that room, mute the phone of users participating to the meeting, show relevant information for the user's current location, assist the user find its way in the campus, or automatically recommend events based on the user's (possible learned) profile and interests.

CAPIM is designed to support the construction of the next-generation context aware applications. It integrates services designed to collect context data (things' location, profile, etc.). These smart services can be dynamically loaded by mobile things, and make use of the sensing capabilities provided by modern smart objects, including smartphones endowed with additional sensors. The data is collected and aggregated into context instances. This is also possibly augmented with external and inferred data about situations, relations, and events. In addition, the platform includes a workflow engine designed to continuously evaluate the context and take automatically decisions or actions based on customized rules and particular context events.

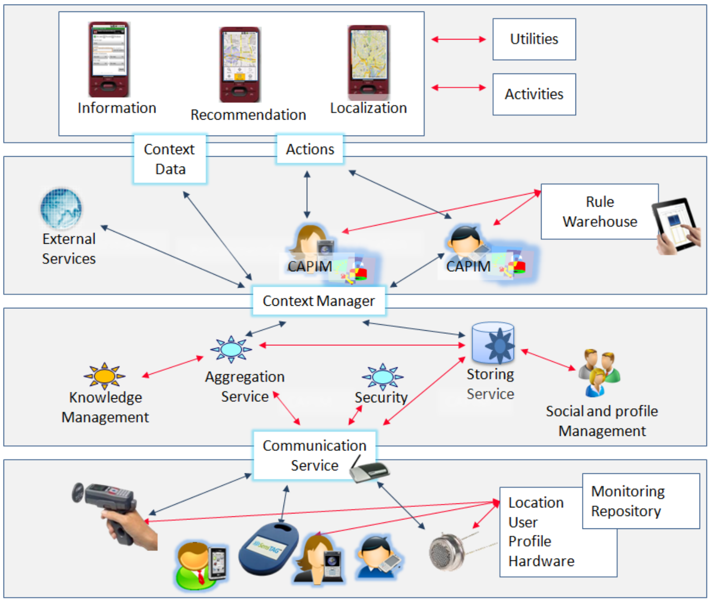


Fig. 4 CAPIMs'architecture

CAPIM's architecture consists of four vertical layers (see Fig. 4). Each layer provides a specific function: (1) collecting context information, (2) storing and aggregation of context information, (3) construction of context-aware execution rules, and (4) visualization and user interaction. Each layer has several components, making the infrastructure suitable for experimenting with a wide range of context-aware things, methods, techniques, algorithms, and technologies. CAPIM can be used to construct context-aware applications using a service-oriented composition approach: load a core container, instruct it to load the necessary context-gathering services, deploy a corresponding context-aware business workflow, and call the actions to be executed when context is met. For example, the monitoring services are dynamically discovered, downloaded as needed, loaded and executed in the container. The first layer includes sets of monitoring services for collecting context data and first-stage storing on the local smart objects.

Each monitoring service is packed in a digitally signed monitoring module. These modules are downloadable from remote repositories, resembling application stores. The monitoring services can be developed/maintained by third party organizations. For example, a manufacturer might build a module to collect data from sensors it is offering on the market, therefore integrating them within Internet.

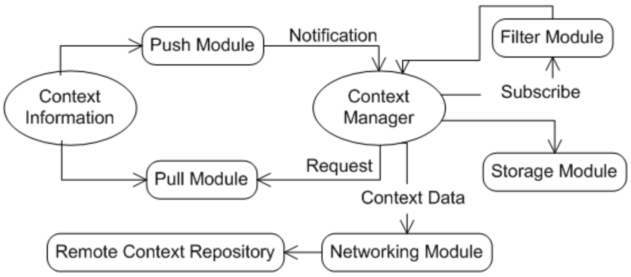


Fig. 5 Flow of monitoring information

Each monitoring service is executed in a separate container. This allows separation of concerns (no service needs to know what other modules are deployed) and fault isolation.

The monitoring flow (see Fig. 5) is under the control of a Context Manager, orchestrating the flow of information between the monitoring services. Depending on the function supported, the monitoring services are grouped in several categories. The Push and Pull monitoring services are directly responsible for collecting context information, usually directly from sensors. The Push service reacts to changes of the context, which in turn triggers notifications to the Context Manager. The Pull service is periodically or on-request interrogated for the current values of the monitoring parameters.

The context information is sent to Filter, Storage and Networking services. The Filter service subscribes to specific context information. The Context Manager forwards the data of interest to the Filter service, which in turns can produce new context information (possible from multiple data sources). Such a construction allows for first-stage aggregation of context information.

The Storage service can keep data locally for better serving the context-execution rules. Finally, the Networking service is responsible for sending the collected context information remotely to aggregation services (the Remote Context Repository component located in the next layer). Here we can experiment with different network protocols and methods of sending data, whilst balancing between costs and energy-consumption.

Each monitoring service is also responsible for a particular type of monitoring information. Thus, these services fall into different categories: location, profile, hardware.

The second layer deals with the aggregation and storing of context data. The components at this layer are running in a server environment, mainly because the aggregation involves collecting data from multiple fixed and mobile sources. Also it involves higher computational capabilities that are available on smart things and user's smartphone without interfering with his/her own activities. The components are distributed, and we envision a scheme where several such servers collect data based on a localization approach.

At this layer the information is received from several context sources. It is further organized based on concepts from a predefined model. At his layer the data is organized according to the proposed Context Model. For example, the data from several sensors (GSM, WiFi, Bluetooth) is aggregated into current Location, and the user can experiment with various location algorithms. The user's characteristics are organized based on a FOAF and semantic technologies [16]. We therefore are able to aggregate data into models describing actual relations between users, inferring information about their interests and activities. In an academic environment this allows defining rules specific to users interested in particular research area, or belonging to particular classes.

This layer also provides an abstraction that can be used by all applications to access context information. The domain described by the model acts as a contract between the middleware system and consumer applications. The information and services offered by the contextualization services are consumed by two sorts of applications. Autonomous applications can use the services directly to access context information. They control entirely the reaction to context changes.

In addition, we define a third layer, which uses context Rule actions. Changes in the context may trigger different actions on the smart things according to a predefined rule set. The rules are expressed in an XML-based format and are stored in a remote repository. Things are therefore able to dynamically load and execute locally specific rules, depending on context. An example of such a rule is presented in Fig. 6, which notifies the interested and available user about an event in the field of Distributed Systems. The main rule consists of two standard rules combined by the logical operator AND. The first rule retrieves context information regarding user agenda or university timetable.

```
<rules-config>
      <rule-definitions>
            <rule-def name="DistributedSystemEventNotification"
                  action="category.EVENT_NOTIFICATION">
                  <rule name="userIsFree" />
                  <operator name="AND" />
                  <rule name="userHasInterest" />
            </rule-def>
      </rule-definitions>
      <rule-implementations>
            <rule-impl name="userIsFree" class="rules.StringFieldEquals">
                  <property name="argField" value="CURRENT_ACTIVITY"/>
      <property name="target" value="free"/>
      </rule-impl>
      </rule-implementations>
      <rule-implementations>
            <rule-impl name="userHasInterest"
                  class="rules.StringFieldContainedInList">
                  <property name="argField" value="INTERESTS"/>
                  <property name="target" value="Distributed Systems"/>
            </rule-impl>
      </rule-implementations>
</rules-config>
```

Fig. 6 Example of a context rule

Finally, the fourth layer is responsible with the applications, expressed as rules and actions, which can be used for orientation, information and recommendation purposes. At this layer there are local utilities that can help with context-triggered actions. There are also the applications that use the context data to improve response to stimulus (an interior or exterior request). An application can react to changes in the current context and take specific actions depending on some predefined rules. For this, conditions are evaluated period as the data is retrieved. Third party applications and services can use the API provided by the context-aware services. They can use functions for obtaining particular context data, using filters, or can subscribe for context data. They can also declare new execution rules for users to install on their mobile devices.

2.5 CAPIM Context Model

The CAPIM's context model (Fig. 7) characterizes the situation of an entity. Entity describes any person, place, or object that is considered relevant to the interaction between the user and the environment. In accordance, the context is the collection of information used in some particular form. Thus, the context model includes external data (relative to the environment of the mobile device executing the application, such as location, proximity) or internal information (to the device, such available disk space, battery, capabilities, etc.). The proposed context model aggregates this information into a unique set of data. The context is build based on all detectable and relevant attributes of the mobile wireless device, the application's user, the device's surrounding environment, and the interaction between mobile devices (creating an ad-hoc social interaction map).

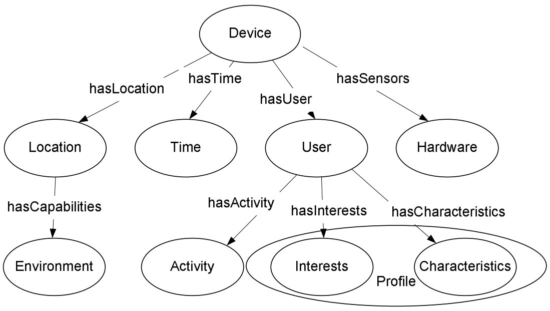


Fig. 7 CAPIM's context model

The context model is hierarchical. On the first layer is the device object (thing) grouping together location, time, identity of a possibly user (in case of smartphones), and the information gathered from various hardware sensors. The device object also provides static information about the device, such as its identifier, operating system and platform, etc.

Location is obtained from several sources. For out-door location we use the GPS or GSM capabilities of the device. For in-door location we combine information received from several sensors, such as GSM cells, Wi-Fi access points, and hardware devices capable of recognizing Bluetooth pairing. The platform also allows experimenting with various in-door locality algorithms and solutions. In this case first the user constructs a module (if one is not already available) for collecting information from sensors. It then aggregates the information into a recognizable form of location data (e.g., the user is in front of a predefined room).

For smartphone, the context information includes information about the user. User's identity is made available from the certificates installed on the mobile smartphone. When the user’s identity is found, it is augmented with other information, such as the user’s profile and activities. User’s activities are discovered from his/her agenda, or from the user’s academic schedule (if the user is a student, based on his certificate the schedule is discovered by interrogating the university’s data management system). The profile context could include information related to user’s research interests, academic interests, or social interests. For the research interests a special service collects and aggregates data from scientific research databases and provides a set of features including automatic collection of information, guided and focused retrieval of researcher profiles, aggregation and storage of structured data in time, aggregated and personalized view of collected information.

The user’s profile is provided in either a static form (for example, based on the certificate the user’s current academic profile can be easily extracted from the university’s digital record database), or is inferred from social networks. For this, the application uses as data sources several social networks: Facebook, LinkedIn. These sources provide dynamic information about user’s interests for example. But they also provide information about social relations between users. So, instead of asking users to insert their social preferences again, we learn them from the users’ social networks and devise new connections based on the supplementary context information. This allows making queries to the system asking for the whereabouts of the user’s current friends, representing users with current interests situated in the immediate proximity, or finding friends that can serve some specific events.

The context also includes system information, collected from specific sensors for battery level, light intensity, accelerometer, etc. The hardware context includes information gathered from external sensors in the environment.

CAPIM’s vision is to use the context information as part of the processes in which things are involved. The context can support the development of smart applications capable to adapt based on the data relevant to the location, identity, profile, activities, or environment (light, noise, speed, wireless networking capabilities, etc.). We propose the use of a context model that includes these parameters. Based on this model we propose building smart and social environments capable to adapt to context using mainly the sensing and processing capabilities of smart objects.

CAPIM uses a semantic-based context model, but other models are also supported. For example, data is collected as time series, for long-term and near real-time processing guarantees. The semantic model provides a vocabulary to represent knowledge about context and to describe specific situations. The Context Ontology defines a common vocabulary to manage and share context data. The advantage of such an approach is sharing a common understanding of information to users, devices and services, because the ontology includes machine-interpretable definitions of basic concepts and relations.

The aggregation and semantic services are running on server-side because the semantic aggregation involves collecting and aggregating together data from multiple sources. All things send context information to the aggregation service, where it is further managed and semantically organized. The aggregation service is also responsible to infer the stored data and send aggregated information back to things or applications. The aggregated semantic data is kept in a semantic database. CAPIM's repository implementation uses the Jena Semantic Web Toolkit. The framework provides functions to add, remove, query even to infer data on generic RDF models.

The context ontology captures all context information and models the basic concepts of things, interests and activities, describing the relationships between these entities. For example, for the pervasive computing which can be divided in a collection of sub-domains (e.g. home domain, school domain), we composed our own ontology using domain-specific ontologies. Considering its specific characteristics, CAPIM things' characteristics are organized based on the FOAF ontology (Fig. 8). In this way one can describe things, contexts and activities, and relations to other things. To model a paper or a book, CAPIM uses the PUBL ontology, storing and linking in this way information such as authors, publishing company or the release date. To describe events, dates or locations CAPIM uses the ICAL and GEO / WAIL ontologies.

The main benefit of using domain-specific ontologies is that one can dynamically plug and unplug them from the model when the environment has changed. The CAPIM's ontology is based on other, already implemented ones because in this way the redundancy can be avoided and the semantic stored data can be easier linked with other information on the Web.

Using the context ontology in a CAPIM academic scenario, for example, one can query and infer the stored data finding out new useful information easier. To illustrate the modeling concept we can describe a typical scenario: to socialize, in a break, first year computer science student Tom wants to discuss about Semantic Web with his interested mates. For this he just needs to use CAPIM service. It will interrogate the aggregation service, which will send to device the required data. With a relational model, the service should have to iterate through all CAPIM users, to find their locations and their interests. This semantic model has all this data linked, so the result is obtained faster without being affect its validity.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:org="http://www.w3.org/ns/org#">

      <foaf:Person rdf:about="andreea.starparu">
            <org:memberOf rdf:resource="Gr341C3"/>
            <foaf:name>Andreea Starparu</foaf:name>
            <foaf:nick>andreea.starparu</foaf:nick>
            <foaf:interest>Semantic_Web</foaf:interest>
            <foaf:interest>Distributed Systems</foaf:interest>
            <rdfs:subClassOf rdf:resource="prezentare_licenta"/>
            <wail:location>
                  <geo:Point>
                    <geo:lat>47.235</geo:lat>
                    <geo:long>25.581</geo:long>
      </geo:Point>
            </wail:location>
      </foaf:Person>
      <ical:vevent rdf:about="prezentare_licenta"/>
         <ical:summary>thesis presentation</ical:summary>
         <ical:dtstart rdf:datatype="xsd:data">2011-07-11</ical:dtstart>
         <ical:dtend rdf:datatype="xsd:data">2011-07-15</ical:dtend>
         <ical:location>
               <geo:Point>
                     <geo:lat>47.235</geo:lat>
                     <geo:long>25.581</geo:long>
      </geo:Point>
         </ical:location>
      </ical:vevent>
</rdf:RDF>
```

Fig. 8 Example of FOAF-based description of context in CAPIM

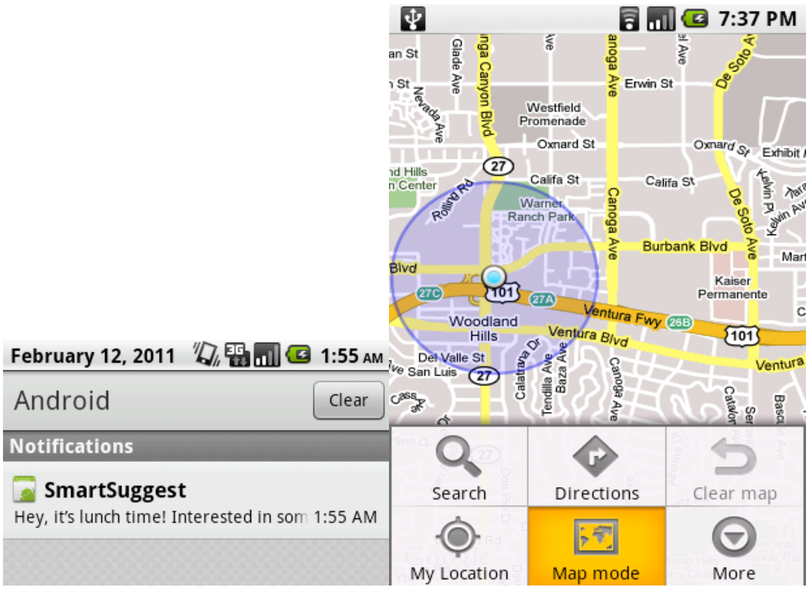
```
<?xml version="1.0" encoding="UTF-8"?>
<rules-config>
      <rule-definitions>
            <rule-def name="showRestaurantSuggestion"
            action="category.PLACE_SUGGESTION"
            parameter="restaurants">
            <rule name="isLunchTime" />
      </rule-def>
</rule-definitions>
<rule-implementations>
      <rule-impl name="isLunchTime" class="rules.IntFieldBetween">
            <property name="argField" value="TIME"/>
            <property name="targetStart"  value="13"/>
            <property name="targetEnd" value="14"/>
      </rule-impl>
</rule-implementations>
</rules-config>
```

Fig. 9 A context-based rule example

2.6 *Use Case*

A possible application of the proposed platform and context services is an automated support for people in an university, who may be endowed with a portable device which reacts to changes of context by (a) providing different information contents based on the different interests/profiles of the visitor (student or professor, having scientific interests in automatic systems or computer science, etc.), and on the room he/she is currently in; (b) learning, from the previous choices formed by the visitor, what information s/he is going to be interested in the next; (c) providing the visitor with appropriate services – to see the user's university records only if appropriate credentials are provided, to use the university's intranet if the user is enrolled as stuff; (d) deriving location information from sensors which monitor the user environment; (e) provide active features within the various areas of the university, which alerts people with hints and stimuli on what is going on in each particular ambient.

The proposed context-aware platform can be used for the experimental evaluation of many solutions. Users can evaluate methods for gathering context information, for aggregating data using semantics, ontologies. In addition, the platform allows experimenting with complementary context-aware solutions. Consider for example a security component designed to offer a session establishment mechanism, along with session verification processes. Services might use it to verify the identity/authorization of the user currently being the possession of the smartphone. A session can be established through HTTPS using certificate authentication. The solution can, for example, allow users to unlock doors within a building without the requirement of using a physical key or any other replacements (smartcards, swipe cards, etc.). All that is required is a smartphone present in the proximity of the door and a valid user X.509 certificate installed within the phone.


February 12, 2011 1:55 AM

Android Clear

Notifications

SmartSuggest

Hey, it's lunch time! Interested in som 1:55 AM

Fig. 10 Expanded notification (up), followed by suggestion of nearby restaurants (down)

Another example is based on the rule described in Fig. 9. In this example the context is evaluated. When it is lunch time (anywhere between 13 and 14 hour), the rules triggers an action which, based on the user's current location and using Internet services, finds all restaurants nearby. A notification is then brought up. If the user is interested he can access more details about the suggested nearby restaurants. In another situation, the application observes that the user is in a free time interval according to his/her agenda and place (location), and also that the weather is sunny (using weather Internet services). According to the user's settings it can suggest parks nearby, or other similar outdoor activities close to the user's current location. An example of an execution of the rule is presented in Fig. 10. As a result of the execution, the user is presented with a notification and restaurants suggestions nearby current location.

3 Future Trends and Research Directions in Internet of Things Infrastructure and Services

Internet of Things is not yet a reality, "but rather a prospective vision of a number of technologies that, combined together, could in the coming 5 to 15 years drastically modify the way our societies function" [13]. The evolution of the IoT on medium and long term unleashed a huge interest and gave rise to many research projects, workshops, and conferences, and to the elaboration of reports and survey papers. In this section we discuss the aspects related to the IoT infrastructure and services with emphasis on the main challenges.

It is estimated [32] that IoT will have to accommodate over 50,000 billion objects of very diverse types and technologies. Standardization and interoperability will be mandatory for interfacing them with the Internet. New media access techniques, communication protocols and standards shall be developed to make thing communicate with each other and people. One approach would be the encapsulation of smart wireless identifiable devices and embedded devices in web services. Some initiatives regarding Web services and things' context [33], interacting with the SOA-Based Internet [35], efficient REST-based communications among embedded systems [36] and others demonstrate the high potential of this solution. They also show enhancing the quality of service aspects like response time, resource consumption, throughput, availability, and reliability is possible. The discovery and use of knowledge about services' availability and of publish/subscribe/notify mechanisms would also contribute to improving the management of complex thing structures.

The huge number of things will make their management a very difficult task. One solution is enhanced monitoring facilities to track things and people, and gather information about their status and situation to support informed decisions. Another one is to enable things' adaptation, autonomous behavior, intelligence, robustness, and reliability. They could be based on new general centralized or distributed organizational architectures or on endowing things with self-* capabilities in various forms: self-organization, self-configuration, self-Healing, self-optimization, and self-protection. As an example, in the BIONETS European project [17], evolutionary techniques are embedded in system components to achieve fully autonomic behavior and to properly solve networking and service management issues.

New services shall be available for persistent distributed knowledge storing and sharing, and new computational resources shall be used for complicated tasks execution. Actual forecasts indicate that in 2015 more than 220 Exabytes of data collected from sensors, tracking systems or generated by smart things will need to be stored [32]. At the same time, optimal distribution of tasks between smart objects with high capabilities and the IoT infrastructure shall be found. Since the volumes and rates of these data are very dynamic, elastic Clouds are the best candidates for storing them. Obviously, Clouds can be also used for rapid processing of information and results delivery to the end user.

New mechanisms and protocols will be needed for privacy and security issues at all IoT levels including the infrastructure. Solutions for stronger security could be based on models employing the context-aware capability of things, and on the capabilities of the wireless channels to ensure security.

New methods are required for energy saving and energy efficient self-sustainable systems. Researchers will look for new power efficient platforms and technologies and will explore the ability of smart objects to harvest energy from their surroundings.

4 Conclusions and Remarks

Actual evolution of the Internet of Things towards connecting every thing on the planet in a very complex and large environment gives raise to high demanding requirements, which challenge the actual and future research. The continuously increasing volume of data collected from and exchanged among things will require highly scalable environments able to support the high resulting network traffic, and offer the necessary storage capacity and computing power for data preservation and transformation. Communication protocols are needed to enable not only the high capacity traffic but also maintain the connectivity between things even in case of transient disconnection of wired or wireless links. Also, new solutions should be found for efficiently store, search and fetch the data manipulated in these environments.

The chapter addresses new research and scientific challenges in context-aware environments for IoT. They refer first to the identification, internal organization, provision of context information, intelligence, self-adaptation, and autonomic behavior of individual things. Then, actual research and main challenges related to IoT infrastructure are discussed, with emphasis on services for context awareness, inter-communication, interoperability, inter-cooperation, self-organization, fault tolerance, energy saving, compute and storage services, and management of things collections and structures. Finally, future trends and research directions for the IoT infrastructure are discussed including performance, monitoring, reliability, safety, survivability, self-healing, transparency, availability, privacy, and others.

Acknowledgments. The work has been co-funded by the Sectorial Operational Program Human Resources Development 2007-2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Agreement, POSDRU/89/1.5/S/62557.

References

1. Cristea, V., Dobre, C., Costan, A., Pop, F.: Middleware and architectures for space-based and situated computing. Int. J. Space-Based and Situated Computing 1(1), 43–58 (2011), doi:10.1504/IJSSC.2011.039106
2. Dobre, C.: CAPIM: A Platform for Context-Aware Computing. In: 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (2011)
3. Dey, K., Salber, D., Abowd, G.D.: A Context-Based Infrastructure for Smart Environments, GVU Technical Report; GIT-GVU-99-39 (1999), http://hdl.handle.net/1853/3406
4. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. 2(4), 263–277 (2007), doi:10.1504/IJAHUC.2007.014070
5. CERP-IoT – Cluster of European Research Projects on the Internet of Things. In: Sundmaeker, H., Guillemin, P., Friess, P., Woelfflé, S. (eds.) Vision and Challenges for Realising the Internet of Things. European Commission - Information Society and Media DG (March 2010)
6. Chen, Helal, S.: A device-centric approach to a safer internet of things. In: Proceedings of the 2011 International Workshop on Networking and Object Memories for the Internet of Things (NoME-IoT 2011), pp. 1–6. ACM, New York (2011), doi:10.1145/2029932.2029934
7. IEEE Computer Society, IEEE Standard 802.15.4, Web page reference (2011), https://standards.ieee.org/findstds/standard/802.15.4-2011.html (accessed on March 2012)
8. Chen, L.-J., Sun, T., Liang, N.-C.: An Evaluation Study of Mobility Support in ZigBee Networks. J. Signal Process. Syst. 59(1), 111–122 (2010), doi:10.1007/s11265-008-0271-x
9. Quoc, N.D., Kim, D.-S.: Performance evaluation of priority CSMA-CA mechanism on ISA100.11a wireless network. Comput. Stand. Interfaces 34(1), 117–123 (2012), doi:10.1016/j.csi.2011.06.001
10. Song, J., Han, S., Zhu, X., Mok, A.K., Chen, D., Nixon, M.: A complete wirelessHART network. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys 2008), pp. 381–382. ACM, New York (2008), doi:10.1145/1460412.1460462
11. Cody-Kenny, B., Guerin, D., Ennis, D., Carbajo, R.S., Huggard, M., Mc Goldrick, C.: Performance evaluation of the 6LoWPAN protocol on MICAz and TelosB motes. In: Proc. of the 4th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks (PM2HW2N 2009), pp. 25–30. ACM, New York (2009), doi:10.1145/1641913.1641917
12. Charalampos, D., Ilias, M., Philippos, T., Dimitris, L., Gregory, Y.: Patient Fall Detection using Support Vector Machines. In: Artificial Intelligence and Innovations 2007: from Theory to Applications. IFIP International Federation for Information Processing, pp. 147–156 (2007)
13. CASAGRAS EU FP7 Project, RFID and the Inclusive Model for the Internet of Things (2012), http://www.grifs-project.eu/data/File/CASAGRAS%20FinalReport%202.pdf (retrieved on March 2012)

14. Haller, S.: Internet of Things: An Integrated Part of the Future Internet, Prague (May 13, 2009), `http://services.future-internet.eu/images/1/16/A4_Things_Haller.pdf` (retrieved on March 2012)
15. Kortuem, G., Kawsar, F., Sundramoorthy, V., Fitton, D.: Smart Objects as Building Blocks for the Internet of Things. IEEE Internet Computing 14(1), 44–51 (2010), doi:10.1109/MIC.2009.143
16. Bardram, J.E.: The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 98–115. Springer, Heidelberg (2005)
17. Miorandi, D., Carreras, I., Altman, E., Yamamoto, L., Chlamtac, I.: Bio-Inspired Approaches for Autonomic Pervasive Computing Systems. In: Liò, P., Yoneki, E., Crowcroft, J., Verma, D.C. (eds.) BIOWIRE 2007. LNCS, vol. 5151, pp. 217–228. Springer, Heidelberg (2008)
18. Herrmann, K., et al.: An Emerging Technology for Pervasive Adaptation. In: Proc. of 2nd IEEE Intl. Conf. Self-Adaptive and Self-Organizing Systems Workshop (SASOW 2008), pp. 108–113 (2008)
19. Costa, P.D., Guizzardi, G., Almeida, J.P.A., Pires, L.F., van Sinderen, M.: Situations in conceptual modeling of context. In: EDOCW 2006: Proceedings of the 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops, pp. 6–16. IEEE Computer Society, Hong Kong (2006)
20. Ye, J., Dobson, S., McKeever, S.: Review: Situation identification techniques in pervasive computing: A review. Pervasive and Mobile Computing 8(1), 36–66 (2012)
21. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: models and approach. Pervasive and Mobile Computing 2(1), 37–64 (2006)
22. Loke, S.W.: Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. Knowledge Engineering Review 19(3), 213–233 (2004)
23. Kalyan, S., Gopalan, V.S.: Hybrid context model based on multilevel situation theory and ontology for contact centers. In: PERCOMW 2005: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 3–7 (2005)
24. Augusto, J.C., Liu, J., McCullagh, P., Wang, H., Yang, J.-B.: Management of uncertainty and spatio-temporal aspects for monitoring and diagnosis in a smart home. International Journal of Computational Intelligence Systems 1(4), 361–378 (2008)
25. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications 28(1), 1–18 (2005)
26. Huynh, T., Fritz, M., Schiele, B.: Discovery of activity patterns using topic models. In: UbiComp 2008: Proceedings of the 10th International Conference on Ubiquitous Computing, pp. 10–19. ACM, New York (2008)
27. Gu, T., Pung, H.K., Zhang, D.Q.: A Bayesian approach for dealing with uncertain contexts. In: Proceedings of Advances in Pervasive Computing, Coexisted with Pervasive 2004, pp. 205–210. Austrian Computer Society (April 2004)
28. van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. In: UbiComp 2008: Proceedings of the 10th International Conference on Ubiquitous Computing, pp. 1–9. ACM, Seoul (September 2008)
29. Yang, J.-Y., Wang, J.-S., Chen, Y.-P.: Using acceleration measurements for activity recognition: an effective learning algorithm for constructing neural classifiers. Pattern Recognition Letter 29(16), 2213–2220 (2008)

30. Kanda, T., Glas, D.F., Shiomi, M., Ishiguro, H., Hagita, N.: Who will be the customer?: a social robot that anticipates people's behavior from their trajectories. In: UbiComp 2008: Proceedings of the 10th International Conference on Ubiquitous Computing, pp. 380–389. ACM, Seoul (2008)
31. Coutaz, J., Crowley, J.L., Dobson, S., Garlan, D.: Context is key. Commun. ACM 48(3), 49–53 (2005)
32. INFSO D.4 Networked Enterprise & RFID, INFSO G.2Micro & Nanosystems, Working Group RFID of the ETP EPOSS. Internet of Things in 2020 (2008) Roadmap for the future, Version 1.1 - May 27, 2008. European Commission - Information Society and Media DG (2009)
33. He, J., Zhang, Y., Huang, G., Cao, J.: A Smart Web Service Based on the Context of Things. ACM Transactions on Internet Technology 11(3), Article 13 (January 2012)
34. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A Survey of Context Modelling and Reasoning Techniques. Preprint submitted to Elsevier (March 27, 2008), `http://www.perada.eu/documents/articles-perspectives/survey-context-modeling-reasoning-techniques.pdf` (retrieved April 2012)
35. Guinard, D., Karnouskos, S., Savio, D.: Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. IEEE Transactions on Services Computing 3(3), 223–235 (2010)
36. Castellani, A.P., Gheda, M., Bui, N., Rossi, M., Zorzi, M.: Web Services for the Internet of Things through CoAP and EXI. In: IEEE International Conference on Communications Workshops (ICC), June 5-9, pp. 1–6 (2011)
37. Christensen, H.B.: Using Logic Programming to Detect Activities in Pervasive Healthcare. In: Stuckey, P.J. (ed.) ICLP 2002. LNCS, vol. 2401, pp. 421–436. Springer, Heidelberg (2002)
38. Yau, S.: Hierarchical situation modeling and reasoning for pervasive computing. In: Proc. of IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, SEUS 2006/WCCIA 2006, Tempe, Arizona, US (2006)
39. Thierry, D.: A k-Nearest Neighbor Classification Rule Based on Dempster-Shafer Theory, Classic Works of the Dempster-Shafer Theory of Belief Functions. In: Studies in Fuzziness and Soft Computing, pp. 737–760. Springer, Heidelberg (2008)
40. McKeever, S., Ye, J., Coyle, L., Dobson, S.: Using Dempster-Shafer Theory of Evidence for Situation Inference. In: Barnaghi, P., Moessner, K., Presser, M., Meissner, S. (eds.) EuroSSC 2009. LNCS, vol. 5741, pp. 149–162. Springer, Heidelberg (2009)
41. Sutter, J.D.: Street Bump app detects potholes, tells city officials (2012), `http://whatsnext.blogs.cnn.com/2012/02/16/street-bump-app-detects-potholes-tells-city-officials/` (retrieved March 26, 2012)
42. Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., Jubert, I.S., Mazura, M., Harrison, M., Eisenhauer, M., Doody, P.: Internet of Things Strategic Research Roadmap. In: Internet of Things - Global Technological and Societal Trends, pp. 9–52. River Publishers (2009)