Smart CAPs for Smart Its – Context Detection for Mobile Users

Florian Michahelles[1] and Michael Samulowitz[2]

[1]Institute for Scientific Computing, ETH, CH-8092 Zurich; [2]Corporate Technology Siemens AG, Otto-Hahn-Ring, Munich, Germany

Abstract: Context detection for mobile users plays a major role for enabling novel, human-centric interfaces. For this, we introduce a context detection scheme applicable in a self-organized sensor network, which is formed of disseminated, computer empowered sensors, referred to as Smart-*Its* [1]. Context-detection takes place without requiring any central point of control, and supports push as well as pull modes. Our solution is based on an in-network composition approach relying on so-called *smart Context-Aware Packets* (*sCAPs*). These packets act as a uniform interchange format, and allow single sensors to share sensed data and to cooperate to build up a meaningful context model from manifold inputs. *sCAPs* travel through the sensor network governed by an enclosed *retrieving plan*, specifying which sensors to visit in order to gain a specific piece of context information. For enhanced flexibility, the retrieving plan itself may be dynamically altered in accordance with past sensor readings.

Keywords: Context awareness; Perceptual computing; Sensor networks; Smart infrastructure

1. Introduction

Processors and sensors are becoming smaller, cheaper, less power consuming and unobtrusive. Consequently, computing resources and devices metamorphose as a matter of course, vanish into the background and blend one into another.

Users wish to access computing resources anytime and anywhere in a ubiquitous [2] manner without restricting their mobility. Thus, emerging computing resources will have to be approached differently to traditional systems. Human computer interfaces have to reflect these facts.

The notion of proactive computing [3] predicts that humans may get out of the interaction loop completely, but may get serviced specifically according to their needs and current situation instead. Users may encounter various different situations and changing environments. Measurable features such as light level, conversations, the proximity to people or other objects, temperature, etc., can be used as clues with which to detect the user's current situation. This information can be used directly as input to the system, which may result in a shift from explicit to implicit human-computer interaction [4]. We argued [5] that services should be aligned to the user's task. In particular, the system should independently discover and execute services considering the user's context [6]. Therefore, services should be offered in a user-centric way. Hinckley calls the inability of a device to detect important events and properties of the physical world missed opportunities [7]. Buxton has even observed that much technological complexity results from forcing the user to explicitly maintain the context of interaction [8].

Addressing these issues of ubiquity and mobility involves a number of research challenges. The challenge of discovering computing services and resources available in the user's current environment has been partly solved by discovery protocols such as JINI [9], IETF SLP [10] or SDP [11]. Another challenge is the selection and execution of services respecting the user's context. As shown in Rodden et al [12], context impacts upon mobile HCI. But how can it be detected? Accordingly, a third challenge is to be solved: the user's current context. However, applying multiple distributed sensors for revealing context is still in its infancy [13].

We believe that augmenting everyday devices with sensors such as *Smart-Its* [1] has the potential to address some of these issues. Advances in wireless networking and microfabrication (e.g. MEMS [13]) facilitate a new generation of large sensor networks [1,14,15].

Personal and Ubiquitous Computing (2002) 6:269–275

Applications of those sensor networks range from tracking and identification to sensing, thus deploying large numbers of sensors in everyday environments becomes feasible. We will use those sensors for context detection of mobile users.

In this paper, we describe an approach towards exploiting manifold sensors available in a future environment for revealing the user's context. We present a uniform communication scheme that allows explicit context detection requests, initiated by the mobile user (pull), as well as event-triggered context evaluation, initiated by sensors' percepts (push). Our document-based approach – *smart Context-Aware Packets* (*sCAPs*) – allows us to capture feature data in self-organized sensor networks without the need for central points of control. Further, *sCAPs* provide adaptive sensor fusion based on an online changeable execution plan and parallel retrieving of various sensors.

The next section motivates the problem of context detection. We present our vision of a sensor network, discuss the benefits to humans, and describe two scenarios. Section 3 gives a discussion and detailed presentation of the general concept of our packet oriented *sCAP* approach. The following section then follows up on that, and shows how *push* and *pull* interaction between user and sensors can be implemented by *sCAPs*. Finally, the *sCAP* approach is summarized and an outlook is given.

2. Background

In this section we first define our envisioned sensor environment, and then motivate the role of users by giving some short scenarios.

2.1. The envisioned *Smart-Its* environment

We envision an environment with disseminated computing empowered sensors, so-called *Smart-Its* [1]. A large number of those autonomous units form a wireless sensor network [16]. Due to wireless communication, the sensor network can be easily deployed. Accordingly, single sensors are close to the phenomena, which should be monitored by the network. Further, we assume local awareness as proposed in Savvides et al [17] and time synchronization [18] among *Smart-Its*.

In particular, *Smart-Its* are attached to everyday devices such as cups, tables, chairs, etc.; they can be equipped with various sensors for perceiving temperature, light, audio, co-location, movement, and so on. The sensor units deliver defined abstractions of sensed information as simple feature values (loudness, brightness, speed, etc.).

Further, these tiny devices are supplied with a wireless communication such as RF or Bluetooth [19]. Accordingly, the connectivity of the network is constrained by the reach of wireless communication. Due to the provision of mobile nodes and incremental addition and removal, the communication quality may be weak and encounter intermittent disconnection. The stability of the network is unpredictable.

An onboard micro-controller provides computing power and enables simple feature calculation from the sensor's inputs. These features (e.g. loudness, brightness, speed, temperature, etc.) are described by discrete values. The envisioned *Smart-Its* operate autonomously with no central point of control. Thus, there is no directory service giving information about the sensors available in the current environment. Cooperation among sensor nodes is a general goal, as streamlining the activity of several nodes can increase the performance of the entire network. Each device is self-aware, so that it knows about its own sensing capabilities and can report those, if inquired, to its neighbors.

Finally, there are two different types of globally unique identifiers: one for distinguishing *Smart-It* units, the other for distinguishing types of feature values *Smart-Its* are capable of delivering.

2.2. Bringing the user into the play

A mobile user is given access to services available in the surrounding environment. In our approach we want to gain the user's context from sensors available in the environment for implicit human computer interaction [4]. By making use of the term context, it is important for us to go beyond pure location awareness, and taking more meaningful measures as the semantic proximity hierarchy [20] into account. Context may be used as an invocation context for configuring services the user intends to access. This section gives a brief overview on how (mobile) users may benefit from an environment instrumented with sensors. In all

scenarios system input is gained through sensors from the user's task and situation (context).

Smart Chemical Lab
In today's biology laboratories, information is both created and consumed at the lab bench. As workers are focused on their task at hand, currently it is extremely tedious to interface with computer at the same time. On the other hand, biologists need to access and disseminate information in digital form, which is performed in a traditional office. The vision of a smart chemical lab, such as Labscape [21], will bridge the gap between today's laboratories and traditional offices. In the Smart Chemical Lab, embedded technology is available in a pervasive way, such that the process of experiments can be observed, recorded and triggered in a flexible manner. Facilities in the lab may be equipped with *Smart-Its*, which have the potential to assist biologists in laboratory work and gain user input implicitly from observing his behaviour and work activities.

Smart Kitchen
Another possibility is to apply *Smart-Its* in the everyday kitchen environment. A smart kitchen could be able to detect what food is going to be cooked and then assist the cook appropriately – eliminating any possibility of user error. Similar to *smart chemical lab*, a smart system may observe and support kitchen users in their current task.

3. Context Detection

This section discusses how the context of a mobile user can be detected by *sCAPs*. First we present the general concept of *sCAPs*, and then demonstrate how these packets are governed by a so-called *retrieving plan*.

3.1. The concept of *sCAPs*

In contrast to mobile code concepts [22], ours is more lightweight. *sCAPs* do not feature the mobile code concept, but are passive packets. They act as an interchange format for sharing sensor-features among different sensor units, such as *Smart-Its*. The concept of *sCAP* shares some similarities with *Context-Aware Packets* (*CAP*) [5] based on a document-based approach, because it also uses context for implicit addressing of the packet's receiver. The *sCAPs* can be understood as prepared containers for collecting features from various sensors available in the environment.

3.2. Lifecycle

After creation an *sCAP* gets injected into the local sensor network. Each *Smart-It* receiving an *sCAP* contributes to the required sensor features and forwards it to other *Smart-It* devices in its neighborhood. Accordingly, the *sCAP* gets filled with sensed information on its way through the environment. Combining these gained features stored in the *sCAP* allows each *Smart-It* to make an assumption about the current context. Based on this knowledge, it can forward the *sCAP* to an appropriate sensor for further investigation of the context. Thus, there is a permanent in-network recalculation of the context, which allows continuous refinement of the assumption and adaptation of the *sCAP's* path through the sensor network. The context information stored in the packet is used for implicitly addressing sensor units. Accordingly, the *sCAP* is governed by a *retrieving plan* outlined in section 3.2.

3.3. Composition

As Fig. 1 depicts, an *sCAP* document is organized into three parts: *retrieving plan*, *context hypothesis*, *packet trace*. The *retrieving plan* embodies the execution plan determining which sensors should be involved into the context detection. It describes which types of sensors have to be queried for achieving the current context. Due to single sensors percepts, this retrieving plan can be continuously refined at each receiving sensor unit, such that the detection process can adapt to the actual sensor inputs. The representation of the retrieving plan will be discussed in more detail in the subsequent section.

As already mentioned, the *context hypothesis* is represented by the accumulation of feature values retrieved from several sensors. At dedicated more computing empowered nodes, referred to as *Compute-Its*, the *context hypothesis* could be shifted to another level of interpretation by taking the retrieved features into account. The *context hypothesis* is simple represented by a list of feature already perceived. Here, each feature is described by following entries: *Feature ID, Feature value, Sensor type ID, Smart It ID, Sensor location, timestamp*. *Feature ID* determines the type of the

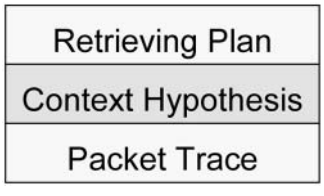

Fig. 1. Smart Context-Aware Packet.

feature, e.g. whether it is loudness, temperature, brightness etc.; *Feature value* is an actual number value; *Sensor type ID* defines the type of sensor the feature was gained from; *Smart It ID* is the uniform identifier for the platform the feature was sensed at; *Sensor location* and *timestamp* give the physical location and time at which the feature was perceived. The context results from these sensor measurements.

The *packet trace* section is organized into two stacks. The first stack maintains a route history of travelled units in the wireless network. The second stack directs an *sCAP* according to a given route. If the second stack is empty the packet just strays the network in order to meet meaningful sensors at random. Both stacks aim at gaining a rough estimate of the current topology in the highly unstable wireless network. This information is used to avoid loops, preventing units from being revisited several times, and to provide knowledge on network topology for transmitting the context to the inquirer (mobile user). The packet trace is core for Smart Stack Routing, as explained in [23].

3.4. Retrieving plan

This section details the semantics and representation of the retrieving plan contained in *sCAPs*. As mentioned above, the retrieving plan defines the sensors to be visited for revealing the user's current context. Initially, the retrieving plan is given by an initial *sCAP* template setup for different context queries. However, what makes our approach promising is the possibility of continuous alteration and adaption and of the retrieving plan to single sensor measurements.

As depicted in Fig. 2, the retrieving plan is represented by three parts: *execution plan*, *rewriting rules* and *inquirer*. The *execution plan* contains the list of sensor units (su) to be involved into the context detection process. In particular, the execution plan maintains a list of types of sensors – it does not identify certain sensors specifically.

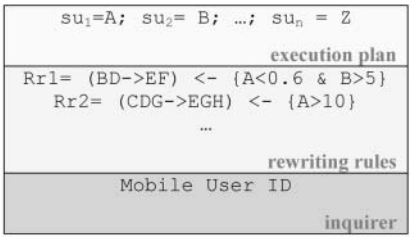

Fig. 2. Retrieving plan.

As the user is mobile, the environment can always change. Accordingly, it seems to be more reasonable to abstract from the actual sensor units, but rather focus on sensor types.

By applying *rewriting rules* the execution plan can be altered at any *Smart-It* to which the *sCAP* is directed. The rewriting rules incorporate the context hypothesis currently available in this *sCAP*. Consequently, significant sensor inputs and combinations of those trigger rewriting rules to alter, replace, add or delete sensor units from the execution plan. In our current prototype, we do not allow changes of the rewriting rules themselves, but this is feasible for future work. The *inquirer* field finally identifies the mobile user the context has to be delivered to.

4. Interaction Schemes: Pull and Push

This section shows how the *sCAPs* can be applied for context detection. Present day context-awareness research employs sensors in a master-slave manner, in which passive sensors report their results to a superior authority upon request. We call this traditional approach *context pull*, as an inquirer – the user – explicitly requests information und *pulls* it to his destination.

Apart from that conventional interaction with sensors, we envision a *context push* design as a counterpart: sensors observe events of interest and become active, so they *push* the context to their superior authority of the traditional approach. There is no inquirer, only a receiver who shows interest for the information. This section outlines these two interaction schemes: *context pull* and *context push*.

4.1. User request: Context pull

Context pull embodies the active intervention of the user, such that the user (or, respectively, a software agent acting on the user's behalf) explicitly inquires a context update. The user initiates the context detection process by injecting an empty *sCAP* template with a default retrieving plan into the network. Accordingly, the *sCAP* gets round routed throughout the sensor network, being forwarded through Smart Its' neighbourhoods.

The *sCAP* collects data from various sensors according to its retrieving plan. Meanwhile, all visited sensors' identifiers are stored in the

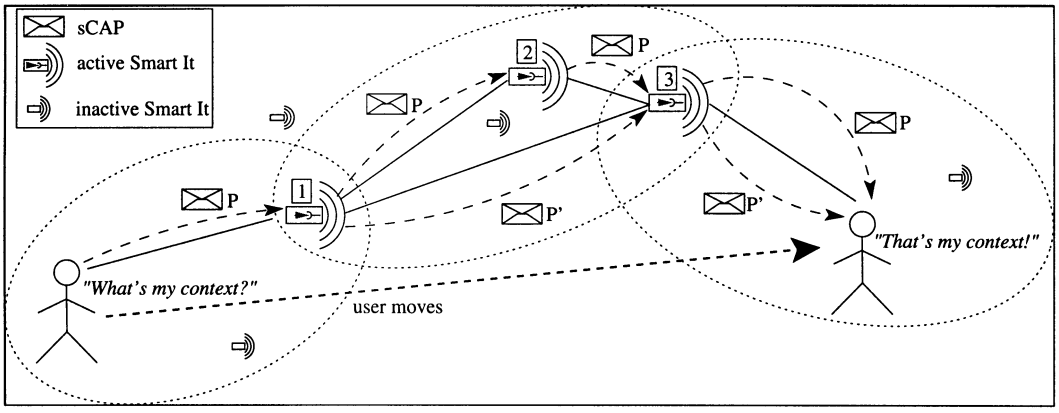

Fig. 3. Context pull.

packet trace section of the *sCAP*. Figure 3 shows an example, where the user broadcasts an initial packet to *Smart-It* 1. According to the packet's retrieving plan, the sensor adds information to the packet (e.g. sound data), and forwards the packet to its neighbours 2 and 3. Consequently, the original packet *P* splits into *P* and *P'*. Nodes 2 and 3 contribute to the packet as well, and forward it to their neighbours. Here, the packet's *packet trace* prevents *P* and *P'* from revisiting node 1 again. The context detection process terminates as soon as the retrieving plan is performed. Assuming that nodes 1 and 3 already fulfil the packet's retrieving plan, node 3 terminates the context detection process. It recognizes the mobile user in its neighbourhood, and returns the packet carrying the requested context information. Generally, its duty is of Smart Stack Routing [23], to locate the mobile user in the network and to trace the packet back to him.

4.2. Autonomous sensors: Context push

In contrast to *context pull*, the *context push* scheme empowers sensors to collect data autonomously. Initially, certain sensors are empowered for autonomous *sCAP* creation by their initial configuration. This configuration retains an application's needs, and also contains the address of the user or an entity interested in context detection. In our current prototype, sensors can

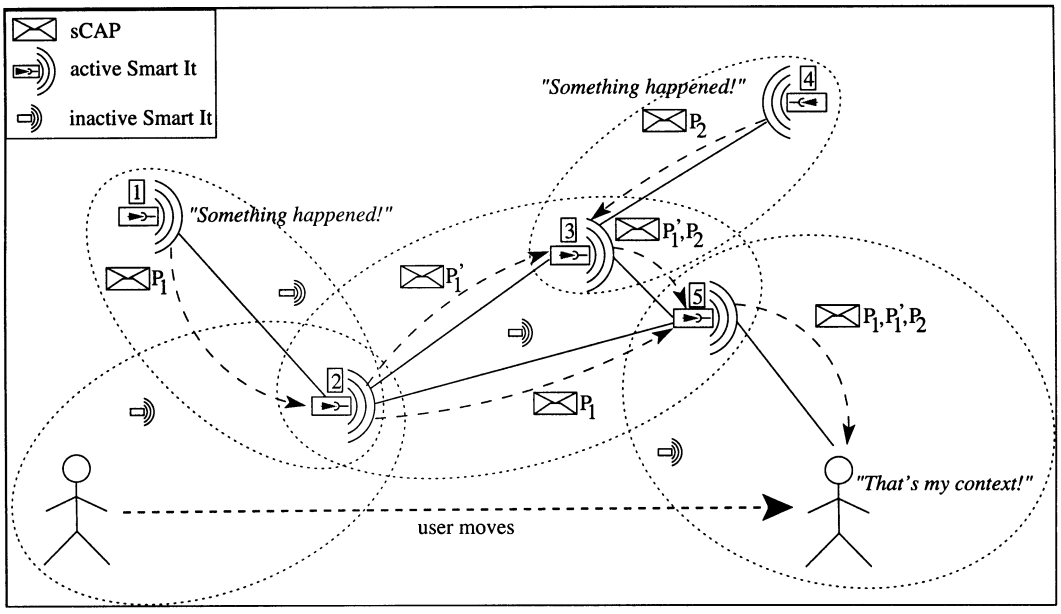

Fig. 4. Context push.

be configured for pushing data on a timely or condition triggered basis.

Due to that configuration, the sensors' precepts are used to trigger the context detection process. As soon as a sensor encounters a certain signal pattern, it can initiate the context detection by injecting an *sCAP* into the sensor network by itself. After that, the detection process elapses as described in section 4.1.

As soon as the context has been revealed, the mobile user is informed on a publish-subscriber basis. Obviously, several sensors can actually perceive different inputs stemming from the same coincidence. As Fig. 4 depicts, both sensor units 1 and 4 detect a certain feature in the environment. Assuming units 1 and 4 carry different sensors (e.g. light and audio), they observe different aspects of the same phenomena. Accordingly, several *sCAPs* can be created, as packets P_1 and P_2 in Fig. 4, at the same time detecting the same context. Packet P_1 is split into P_1 and P_1' which are heading to node 3 and 5. Packet P_2 stemming from node 4 also reaches node 5. Node 5 is the gateway to the user, such that it receives three packets, P_1, P_1', P_2, reporting the same coincidence with input from different sensors. Currently, we do not consolidate those packets, but interpret the occurrence of several packets per coincidence as a quantification of certainty of an observed event. Again, Smart Stack Routing takes care of returning packets back to the user.

5. Conclusions

When envisioning interconnected future environments, several challenges have to be overcome. To provide convenient human-computer interaction in changing environments, the user's context should be taken into account and detected automatically by the system. In this work, we focused on context detection for mobile users in sensor monitored environments. Our work was heavily influenced by the *Smart-Its* project [1] assuming unobtrusively interconnected everyday objects with embedded sensors. We proposed *Smart Context-Aware Packets* as an intercommunication format among *Smart-Its* for revealing mobile users' contexts. Our approach promotes sensor information exchange in sensor networks without central points of control. Further, our *sCAP* documents foster adaptive fusion of feature inputs of diverse sensors.

Whereas context *pull* is used for determining the user's context when the user explicitly needs the current context information, the *push* scheme is more appropriate for emerging events and interrupts, such as detecting rapid changes in the environments, states of emergencies or other sporadic coincidences. Up to now, we have not tried to consolidate parallel context detection processes. Accordingly, the user might receive several *sCAPs* reporting the same context. We interpret this as a quantification of the evidence for the reported contexts. However, future work should focus on these issues, and merging strategies for *sCAPs* have to be found to consolidate related detection processes.

6. Outlook

Future directions of the work will focus on the following issues: (1) online reconfiguration of sensors for different push behaviors; (2) merging strategies for related *sCAPs*; (3) spreading strategies of *sCAPs* for parallelizing the context detection process; (4) online alterable rewriting rules; and finally, (5) tackling communication breakdowns [23,24].

Acknowledgements

The Smart-Its project is funded in part by the Commission of the European Union under contract IST-2000-25428, and by the Swiss Federal Office for Education and Science (BBW 00.0281).

References

1. Smart-Its. http://www.smart-its.org
2. Weiser M. The computer for the 21st century. Scientific American 1991; 265(3): 94–104
3. Tennenhouse DL. Proactive computing. Communications of the ACM 2000; 43: 43–50
4. Schmidt A. Implicit human-computer interaction through context. 2nd workshop on human computer interaction with mobile devices, Edinburgh, Scotland, August 1999
5. Samulowitz M, Michahelles F, Linnhoff-Popien C. Adaptive interaction for enabling pervasive services. MobiDE01, Santa Barbara, CA, May 2001
6. Dey A, Abowd GD. Towards a better understanding of context and contex-awareness. Technical Report 22, GeorgiaTech, 1999
7. Hickley K, Pierce J, Sinclair M, Horvitz E. Sensing techniques for mobile interaction. Symposium on User Interface Software and Technology, CHI Letters 2000; 2(2): 91–100
8. Buxton W. Integrating periphery and context. Graphics Interface 1995; 239–246

9. Jini™. 1998. http://java.sun.com/products/jini
10. Perkins C. Service Location Protocol White Paper, May 1997.
11. Czerwinski S, Zhao B, Hodes T, Joseph A, Katz R. An architecture for a secure service discovery service. Proceedings of MobiCom '99, Seattle, WA, August 1999
12. Rodden T, Cheverst K, Davies N, Dix A. Exploiting context in HCI design for mobile systems. Proceedings 1st workshop on human computer interaction for mobile devices, Glasgow, UK, May 1998; 12–17
13. Micro Electro Mechanical Systems (MEMS). http://mems.isi.edu/
14. The Ultra Low Power Wireless Sensors Project. http://www-mtl.mit.edu/~jimg/project_top.html
15. The WINS Project. http://www.janet.ucla.edu/WINS/
16. Pottie JG, Kaiser WJ. Wireless integrated network sensors. Communications of the ACM 2000; 3(5): 51–58
17. Savvides A, Han C-C, Srivastava M. Dynamic fine-grained localization in ad-hoc networks of sensors. Mobicom 2001
18. Elson J, Estrin D. Time synchronization for wireless sensor networks. Workshop on parallel and distributed computing issues in wireless and mobile computing, San Francisco, CA, April 2001
19. The Bluetooth SIG. http://www.bluetooth.com
20. Schiele B, Antifakos S. Beyond position awareness. Proceedings Workshop on Location Modelling at Ubicomp, October 2001
21. Labscape. http://csi.washington.edu/comsystec/labscape/
22. Qi H, Iyengar S, Chakrabarty K. Distributed multi-resolution data integration using mobile agents. Proceedings IEEE Aerospace Conference 2001
23. Michahelles F, Samulowitz M, Schiele B. Detecting context in distributed sensor networks by using Smart Context Aware Packets. International Conference on Architecture of Computing Systems (ARCS), 2002, Karlsruhe, Germany, April 2002.
24. Goff T, Abu-Ghazaleh N, Phatak D, Kahvecioglu R. Preemptive route maintenance in ad hoc networks. Mobicom 2001

Correspondence to: F. Michahelles, Institute for Scientific Computing, ETH Zentrum, CH-8092 Zurich, Switzerland. Email: florian.michahelles@inf.ethz.ch