

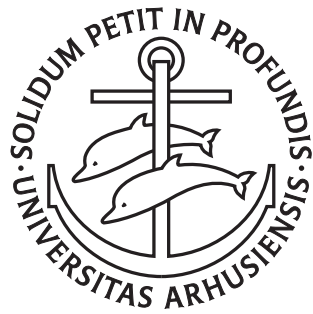
# On Context Awareness in Ubiquitous Computing

Doina Bucur

---

---

PhD Dissertation



Department of Computer Science  
University of Aarhus  
Denmark



# On Context Awareness in Ubiquitous Computing

A Dissertation  
Presented to the Faculty of Science  
of the University of Aarhus  
in Partial Fulfillment of the Requirements for the  
PhD Degree

by  
Doina Bucur  
September 1, 2008



# Abstract

This dissertation contributes to the young field of *ubiquitous computing*, a so-called *third wave of computing* expected to succeed the current Internet era. The paradigm of ubiquitous computing calls for distributing the computation power among the people and objects in the human environment by embedding tiny, sometimes invisible, networked computers into the items of everyday life, so that (i) some of the interactions we engage in on a quotidian basis get redesigned as interactions with networked data-processing systems available everywhere, and (ii) the resulting networks of computers autonomously monitor and control themselves and the environment.

In particular, the concept unifying the contributions of this dissertation is one of the central aspects in ubiquitous computing, *context awareness*—a paradigm calling for the tiny ubiquitous computers to *sense* the change in their computational surroundings and dynamically *adapt* their behaviour to this environmental change. The scientific contributions of this dissertation are of both a theoretical and a practical nature, and lie within a number of areas of computing in context: sensing context (specifically, *sensing location*), discovery of context in large ubiquitous environments (a task known as *service discovery*), *modelling and verification* of context-aware systems, and the porting of traditional security techniques (specifically, *access-control lists*) to work over such mobile, dynamic systems.



# Acknowledgments

The author wishes to thank the many people who had a positive influence upon this dissertation.

Thanks go to the three project advisors and mentors who played a central role in my PhD studies for definite periods of time: Jakob Bardram, whose research group for pervasive computing at DAIMI I was a part of during the first year of my PhD, before his professorship appointment at IT University of Copenhagen; Marta Kwiatkowska, my advisor at Oxford University in the spring of 2008, and Olivier Danvy at BRICS, who kindly looked after the author's research being on a right track, and with whom the author shares the science fiction bug.

Also, thanks go to the Danish and international cast of fellow PhD students and researchers at BRICS during these three years, who, together with the administrative staff members, created our wonderful work environment; special thanks go to Gabi, Gosia and Saurabh for the great amount of patience they put into removing the author from her office after work hours, and have her socialize.

The author is grateful to her family and educators prior to joining the BRICS PhD program, and especially to her Romanian friends and collaborators overseas: Tavi at the author's home school, the University Politehnica of Bucharest, who shared his expertise in operating systems, and Raz and Ral at the Johns Hopkins University in Baltimore, who did the same in the field of sensor networks.

Furthermore, this PhD period yielded more than the author bargained for; thanks go to Henrik for continuous interest and support of my work, and to his family for their graceful acceptance.

Finally, this dissertation wouldn't have existed without the PhD education offered by Mogens Nielsen, my advisor during the entire period. No student could wish for a more supporting, resourceful, passionate and open-minded advisor, and all these in spite of his busiest of schedules. One can only wish to become not only such a good advisor, but also such a kind human being.

*Doina Bucur,  
Århus, September 1, 2008.*



# Preface

This dissertation is structured into two parts: Part I, titled *Overview*, provides introductory matter on the wider subject of this dissertation—the field of ubiquitous computing—then goes into technical detail for those particular topics the author has contributed to, and gives previews of the contributions. In particular, Chapter 1, the *Introduction*, establishes an introduction to ubiquitous computing, giving an overview upon its vision, brief history, challenges and current state of the art. Then, Chapter 2, *Context Awareness. State of the Art and Contribution*, moves away from introductory matters towards a technical view upon context awareness (a central “enabling” concept in ubiquitous computing), seen from a number of perspectives (both practical and scientific) and interspersed with the contribution of this dissertation to the field—all in the historical context set up by Chapter 1.

Finally, Part II, titled *Papers*, lists four published papers and two items of work in progress, grouped into four chapters arranged chronologically, each holding work upon one facet of the topic of context awareness. Two of these chapters are of a practical nature: Chapters 3 and 5 deal with sensing and discovery of context, respectively. The other two chapters cover issues of a more theoretical nature: Chapter 4 deals with modelling context-aware systems with a process calculus and securing them with access-control lists; Chapter 6, a work in progress, aims at building software verification techniques for context-aware C code.



# List of Publications

The following is a list of papers produced during the author's PhD studies. The listing is in reverse chronological order.

- [12] Doina Bucur. Verifying ANSI-C Context-Aware Applications. Published online at <http://www.daimi.au.dk/~doina>. Working paper. 2008.
- [15] Doina Bucur and Mogens Nielsen. A Calculus for Ad Hoc Context Awareness. Published online at <http://www.daimi.au.dk/~doina>. Working paper. 2008.
- [17] Doina Bucur and Mikkel Baun Kjærgaard. GammaSense: Infrastructureless Positioning using Background Radioactivity. In *Proceedings of The Third IEEE European Conference on Smart Sensing and Context (EuroSSC)*. Springer-Verlag, Oct 2008.
- [16] Doina Bucur and Mogens Nielsen. Secure Data Flow in a Calculus for Context Awareness. In *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 439–456. Springer Verlag, June 2008.
- [14] Doina Bucur and Jakob E. Bardram. Resource Discovery in Activity-Based Sensor Networks. In *Mobile Networks and Applications (MONET)*, volume 12, numbers 2–3, pages 129–142. Springer Verlag, June 2007.
- [13] Doina Bucur and Jakob E. Bardram. Resource Discovery in Activity-Based Sensor Networks. In *Proceedings of the First International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–10. Nov 2006.



# Contents

<b>Abstract</b>	<b>v</b>
<b>Acknowledgments</b>	<b>vii</b>
<b>Preface</b>	<b>ix</b>
<b>List of Publications</b>	<b>xi</b>
<b>I Overview</b>	<b>1</b>
<b>1 Ubiquitous Computing. Introduction</b>	<b>3</b>
1.1 Ubiquitous Computing – Brief History and Philosophy . . . . .	3
1.1.1 The Golden Age of Ubiquitous Philosophy: Calm Computing . . . . .	3
1.1.2 Notable Precursors . . . . .	6
1.2 Contemporary Ubiquitous Computing . . . . .	8
1.2.1 A Theory-and-Practice Grand Challenge . . . . .	9
1.2.2 A Definition and Work Plan . . . . .	10
1.2.3 State of the Art and Prototypes . . . . .	13
1.2.4 Venues . . . . .	16
<b>2 Context Awareness. State of the Art and Contribution</b>	<b>17</b>
2.1 A First Axis: from Context to Behaviour . . . . .	17
2.2 A Second Axis: from Practice to Science . . . . .	21
2.3 Sensing Context . . . . .	22
2.3.1 Localization . . . . .	23
2.3.2 GammaSense: Infrastructureless Positioning using Background Radioactivity . . . . .	27
2.4 Service Discovery . . . . .	29
2.4.1 Service Discovery in Activity-Based Sensor Networks . . . . .	32
2.4.2 Modelling Service Discovery . . . . .	34
2.5 Securing Networked Ubiquitous Systems . . . . .	37

2.5.1	Secure Data Flow under Distributed Access Control . . . . .	39
2.6	Contextual Behaviour . . . . .	42
2.6.1	Verifying ANSI-C Context-Aware Applications . . . . .	43
<b>II</b>	<b>Papers</b>	<b>47</b>
<b>3</b>	<b>Resource Discovery in Activity-Based Sensor Networks</b>	<b>49</b>
3.1	Introduction . . . . .	51
3.1.1	ABSN Requirements and Solutions . . . . .	52
3.2	Related Work . . . . .	55
3.2.1	Service Discovery in Ad Hoc Environments . . . . .	56
3.3	ABSN Design . . . . .	58
3.3.1	ZRP and EZRP . . . . .	58
3.3.2	ABSN Discovery Design . . . . .	59
3.3.3	Intra-NC Discovery . . . . .	62
3.3.4	Route and Service Query Solving . . . . .	62
3.4	Discoverability, Optimality and Overhead Analysis . . . . .	63
3.4.1	Neighbourhood Discovery . . . . .	64
3.4.2	Global Discovery . . . . .	66
3.5	Evaluation . . . . .	67
3.6	Conclusions . . . . .	73
<b>4</b>	<b>A Calculus for Context Awareness</b>	<b>75</b>
4.1	Introduction . . . . .	77
4.2	A Calculus for Context Awareness . . . . .	79
4.3	Type Systems and Well-Typedness . . . . .	82
4.3.1	A Type System for Active Code . . . . .	84
4.3.2	A Type System for Inactive Code . . . . .	85
4.4	Operational Semantics and Type Soundness . . . . .	87
4.5	Case Study: Ubiquitous Computing in a Hospital . . . . .	90
4.5.1	The Guessing Visitor . . . . .	90
4.5.2	The Conspiring Nurse . . . . .	91
4.5.3	The Wandering Visitor . . . . .	92
4.6	Related Work, Conclusions and Future Work . . . . .	93
4.7	Appendix . . . . .	95
4.8	An Ad-Hoc Calculus for Context Awareness . . . . .	106
<b>5</b>	<b>GammaSense: Positioning using Background Radioactivity</b>	<b>111</b>
5.1	Introduction . . . . .	113
5.2	Related Work . . . . .	114
5.3	Gamma Radiation . . . . .	115
5.3.1	A Primer on Radioactivity and Ionizing Radiation . . . . .	115

5.3.2	Natural Sources of Radioactivity . . . . .	116
5.3.3	Indoor Variations of Radiation Concentrations . . . . .	117
5.3.4	Experimental Results . . . . .	118
5.4	GammaSense . . . . .	119
5.5	Evaluation . . . . .	121
5.5.1	Data Collection . . . . .	121
5.5.2	Accuracy . . . . .	123
5.6	Conclusions and Future Work . . . . .	126
<b>6</b>	<b>Verifying ANSI C Context-Aware Applications</b>	<b>129</b>
6.1	Introduction . . . . .	131
6.2	Related Work . . . . .	133
6.3	Context-Aware Program Points in SATABS . . . . .	135
6.3.1	SATABS and goto-cc . . . . .	135
6.3.2	Identifying Capps . . . . .	136
6.3.3	Typing the Information Flow . . . . .	138
	<b>Bibliography</b>	<b>141</b>



Part I

Overview



# Chapter 1

## Ubiquitous Computing. Introduction

This chapter visits the history and vision behind the first concept in the title: *ubiquitous computing*. It first looks into the natural evolution of the ubiquitous computing paradigm out of an era of distributed computing and Internet. It visits its notable precursors and state of the art, and lists the technical challenges specific to ubiquitous computing including context awareness, all the while arguing for the need to take a unified, theory-and-practice view upon the matter.

### 1.1 Ubiquitous Computing – Brief History and Philosophy

*The door refused to open. It said, ‘Five cents, please.’  
He searched his pockets. No more coins; nothing. ‘I don’t have to pay you.’  
‘I think otherwise,’ the door said. It sounded smug.  
From the drawer beside the sink Joe Chip got a stainless steel knife; with it  
he began systematically to unscrew the bolt assembly of his apt’s  
money-gulping door.*

*‘I’ll sue you,’ the door said as the first screw fell out.*  
— A cynical passage of futuristic human-computer interaction,  
in Philip K. Dick’s science fiction novel *Ubik*

#### 1.1.1 The Golden Age of Ubiquitous Philosophy: Calm Computing

When one seeks to rediscover the historical origins and motivation behind the computing field now dubbed *ubiquitous computing*, one learns not only the well-known first definition of Mark Weiser [121], a head scientist at the then Xerox Palo Alto Research Center (Xerox PARC [60]), but also the fact that the definition was inspired not only by current computing technologies,

but also by fields of study and sciences inherently non-technical, such as philosophy, anthropology and social sciences.

By the nineties, the research at Xerox PARC had already contributed to treats of modern computing such as the Ethernet and distributed computing, the software for the personal computer workstation, object-oriented programming and graphical user interfaces (as given by their list of “innovation milestones” [61]). Weiser’s heading of Xerox PARC’s Computer Science Laboratory from 1988 through 1994 triggered the creation of “several technologies at the core of the next-generation Internet, among these IPv6, an advanced Internet protocol” [95]. Yet, more than contributing to established technologies, Weiser’s aim was towards novelty, such that their research “continues to contribute to that innovation into the 21st century” [95].

Such proneness to radical innovation, combined with a practical, business-oriented focus, led Weiser’s team to first revisit the computing technology and its eras from the perspective of humans using it; Table 1.1 summarizes that view.

Table 1.1: Mark Weiser’s past and present computing eras, from the point of view of the interaction between humans and technology

Phase I – The <i>Mainframe</i> Era	The term <i>mainframe</i> “[recalls] the relationship people had with computers that were mostly run by experts behind closed doors. Anytime a computer is a scarce resource, and must be negotiated and shared with others, our relationship is that of the mainframe era,” [122] i.e. “from 1940 to about 1980.” [119]
Phase II – The <i>Personal</i> <i>Computer</i> Era	“In 1984 the number of people using personal computers surpassed the number of people using shared computers. The personal computing relationship is personal, even intimate, (...) and you interact directly and deeply with it. The personal computer is most analogous to the automobile – a special, relatively expensive item, that (...) requires considerable attention to operate.” [122]
Transition – The <i>Internet</i> and <i>Distributed</i> <i>Computing</i>	“[People] and their information have become interconnected. [The] Internet brings together elements of the mainframe era and the PC era. It is client-server computing on a massive scale, with web clients the PCs and web servers the mainframes.” [122]

The current Internet technologies are then regarded as precursors of a “third wave” of computing: if the Internet era saw the emerging of “thin clients”, i.e. mobile, lightweight Internet access devices costing only a few hundred dollars, the third era of computing is then supposed to “see the

creation of thin servers, costing only tens of dollars or less, that put a full Internet server into every household appliance and piece of office equipment” [122]. Then, the era of *ubiquitous computing* (informally, *ubicomp*) would be the paradigm of mapping objects to computation (whose cross-over point with the personal computer is optimistically placed before 2020), as quoted in Table 1.2.

Table 1.2: Mark Weiser’s view upon an era of ubiquitous computing

Phase III – The <i>Ubiquitous Computing</i> Era	<p>“[This] era will have (...) computers sharing each of us. Some of these (...) we may access in the course of a few minutes of Internet browsing. Others will be imbedded in walls, chairs, clothing, light switches, cars—in everything. [Ubicomp] is fundamentally characterized by the connection of things in the world with computation. This will take place at a many scales, including the microscopic.” [122]</p> <p>“Activate the world. Provide hundreds of wireless computing devices per person per office, of all scales (...). This has required new work in operating systems, user interfaces, networks, wireless, displays, and many other areas. It is invisible, everywhere computing that does not live on a personal device of any sort, but is in the woodwork everywhere.” [120]</p>
--	--

Such extreme distribution is justly supported by existing technologies, such as IPv6’s design of a large address space (which “can address more than a thousand devices for every atom on the earth’s surface, [and we] will need them all” [122]), microprocessors, the Internet, and artificial intelligence. However, the standard use of these technologies disqualifies them from being ubiquitous: instead of being used one at a time, microprocessor-powered devices should be interconnected, and connected to the Internet. Thus, new scenarios become possible, such as “clocks that find out the correct time after a power failure, microwave ovens that download new recipes, kids toys that are ever refreshed with new software and vocabularies, paint that cleans off dust and notifies you of intruders, walls that selectively dampen sounds” [122]. Also, “doors open only to the right badge wearer, rooms greet people by name, telephone calls can be automatically forwarded to wherever the recipient may be, receptionists actually know where people are, computer terminals retrieve the preferences of whoever is sitting at them” [118].

The motivation behind this activating of everyday objects, called by the ubiquitous paradigm and backed by postmodernism<sup>1</sup> [120], was allegedly

---

<sup>1</sup>*Postmodernism* [86] is a cultural and philosophical term roughly denoting a post-1979 trend calling for creations which are unstructured, lack coherence and consensus of taste, and allow for experimentation with knowledge and rules.

inspired by Xerox PARC’s eclectic environment of social scientists, philosophers and anthropologists; the ubiquitous paradigm focuses not on the power of computing technology itself, but on raising the usability of computing and networking from the human’s point of view, ideally until computing is an information technology as pervasive and comfortably used as written language or the power grid [118, 122]. The constant presence of writing—tagging most environmental objects—allowed humans to retrieve the encoded information, when needed, within moments, without otherwise engaging their attention.

Thus, information technology is expected to move to the background of the human environment as a “tacit dimension”, “periphery” or “calm technology”, and be ideally “so imbedded, so fitting, so natural”, that people use it without being continuously aware of its presence, and without even perceiving the process of the retrieval of that information as complex. [120]. This way, such a technology would allow people to have it easily moved between being the center of attention and the periphery, so that the time gained while not being engaged into the processing of the periphery would be used for higher-level functions.

### 1.1.2 Notable Precursors

Mark Weiser’s 1988 revolutionary idea of calm computing embedded in the woodwork of human environment was a groundbreaking, technically explicit philosophy for the post-desktop computing era. Expectedly, it wasn’t completely unprecedented.

Various research groups did see, if not the all-encompassing vision of calm computing, at least facets of it. Part of the agenda of the Human-Computer Interaction (HCI) research had always overlapped Weiser’s vision of comfortable computing, e.g. the HCI goal of designing computer interfaces which would come closer to the human’s cognitive model of the task at hand, and improve properties such as learnability and efficiency of use. The HCI topic of gesture recognition, now central to ubiquitous computing, was already under research in the sixties, and so was talk on the topic of Computer-Supported Cooperative Work (CSCW), predicting online work participation from people at multiple sites [81]. Later on, the *Things That Think* [31] MIT media lab, founded at the end of 1995 (the year before the majority of Weiser’s papers were published), looked into smart designs for the home of the future from a usability perspective.

The modern Radio-Frequency Identification (RFID) transponder, a central piece of hardware adopted by nowadays’ ubiquitous systems, was perfected as early as 1973, and had predecessors (albeit with a different usage case) in the late forties [74]. Likewise, the earliest barcode (a machine-readable optical representation of data) was patented in the early fifties, and

commercially ubiquitous in the eighties. The first smartphones (i.e. mobile phones with an added functionality close to that of a personal computer, including non-voice communication) were evolved naturally by commercial producers with an aim at everywhere communication.

The technology for micro- and nano-scale sensors and actuators was fairly mature by Weiser’s Xerox PARC tenure; Berkeley’s Center for Microsensors and Microactuators [111] was established in 1986, taking advantage of the progress made in integrated-circuit technology. Sensing technologies, microcontrollers and transceivers in the free ISM bands<sup>2</sup> were eventually combined into modern wireless sensor nodes, which were in development as early as 1998, as part of the SmartDust project [85]; this project, in turn, helped create or influenced nowadays’ main centers for the development of wireless sensor networks [22, 82, 104, 106].

Techniques for positioning (or locating) devices also precede the established definition of ubicomp. The Global Positioning System (GPS [47]) was envisioned as a most needed military navigation system using satellites even during World War II, and the first working satellite was launched in the late seventies. When it was open as a common good to the public, it offered ubiquitous localization (with a complete configuration of 24 satellites in 1994).

To then take a leap also into pre-Weiser science fiction (motivated at least by the fact that science fiction has already developed into science fact on notable occasions), we again find worthy precursors. In Philip K. Dick’s 1969 novel *Ubik* [36], the near-future society lives, unsurprisingly, in environments rich in technology, so that one’s domestic life is virtually run by the likes of news-reading machines, clean-up robots and talking doors. Far from a Weiser utopia though, the technology acts extortionarily at its every use (as this section’s citation recalls). *Ubik* itself is the name given to the entire reality-support, i.e. the universal set of commodities a consumer requires; in this consumerist reality, *Ubik* is ubiquitous and lifted to an almost holy position.

On a lighter note, modern StarTrek’s communication gadgets [59] looked and behaved remarkably like Xerox PARC’s early prototypes supporting the idea of ubiquitous computing (discussed further in Section 1.2.3): a StarTrek communicator badge has the size, localization and basic-application functionality of a ParcTab, while a StarTrek PADD (short for Personal Access Display Device) is PARC’s pad, a post-laptop device as ubiquitous as impersonal sheets of paper.

Arguably, none of these research groups or fiction writers were pursu-

---

<sup>2</sup>The Industrial, Scientific and Medical (ISM) radio bands were originally reserved internationally to be used for purposes other than communication in these fields, but are nowadays shared with licence-free communication protocols such as IEEE 802.11 or Bluetooth.

ing anything precisely on the lines of Weiser’s guidelines, yet they were all foreseeing a post-personal-computer near future, built around a population of mobile, networked devices, which would then become a tool for staying informed about (and acting upon) the human environment.

## 1.2 Contemporary Ubiquitous Computing

“There are many ubiquitous computings”, to quote Greenfield’s Thesis 01 of his 2006 book *Everyware* [2]. Indeed, today’s ubiquitous computing is not the utopia imagined by Mark Weiser (nor, to be fair, the fictional dystopia of Philip K. Dick), but a mesh of loosely-connected fields of study overlapping to various degrees the original, holistic notion of ubiquity.

Furthermore, depending on which aspect of ubiquitous computing registered as central in people’s minds, they called it by various names: to quote only some, “pervasive computing”, “mobile distributed computing”, “global computing”, “the Internet of things”, “context-aware computing”, “ambient intelligence”, “wearable computing”, “tangible media”, “physical computing”, “everyware”, “the disappearing computer”, “calm technology” come to mind, in a rough order from the more abstract, technically inclined (think networking, sensors and artificial intelligence) to the ones closer to users’ experience (think human-computer interaction).

In fact, Greenfield’s *Everyware* [2] makes the point plastically that referring to all these differently named emphases, “one gets reminded time and again of the parable of the six blind men describing an elephant”. In the book’s interpretation, the parable goes like:

“Six wise elders of the village were asked to describe the true nature of the animal that had been brought before them; sadly, age and infirmity had reduced them all to a reliance on the faculty of touch. One sage, trying and failing to wrap his arms around the wrinkled circumference of the beast’s massive leg, replied that it must surely be among the mightiest of trees. Another discerned a great turtle in the curving smoothness of a tusk, while yet another, encountering the elephant’s sinuous, muscular trunk, thought he could hardly have been handling anything other than the king of snakes. None of the six, in fact, could come anywhere close to agreement regarding what it was that they were experiencing.”

Nevertheless, despite their different points of view, these emphases all retain in common the central idea that some of the transactions we engage in during our quotidian deeds (as trivial as the opening of doors or the finding of an ATM) are about to be redesigned as transactions with

networked data-processing systems, distributed at all scales in the environment. Also, despite the distinctions among the technologies employed by various groups—which link themselves to one or another of these working names—for the users of the end system these distinctions disappear. Finally, and sadly, few contributors of these separate directions see all pieces fitting together [2].

In what follows, we argue—sometimes by recalling the parable of the elephant mentioned above—that the final separation of the various facets of ubicomp is unnatural and hinders progress, even if this separation is half-motivated by the fact that the various emphases employ different tools or theories, each with their own technical challenges.

### 1.2.1 A Theory-and-Practice Grand Challenge

Within recent years, the UK’s Computer Research Committee has formulated a number of so-called *Grand Challenges* [30]: a long-term, internationally-scoped topic within Computer Science (i) backed by a consensus of the scientific community, (ii) whose ambition is far greater than what can be achieved by a single research team, and (iii) whose challenge is directed at a revolutionary advance within the topic, rather than at the evolution of legacy, industry-ready products. An opportunity for a *grand* challenge comes up only rarely in science, when a branch of science reaches an adequate level of maturity in order to plan future progress; examples of such challenges are the mapping the human genome (accomplished) or unifying the theories for the four types of forces in physics (under investigation) [30].

The Ubiquitous Computing Grand Challenge [24] is one of currently nine such challenges, formed by merging two previous entries on the theory and engineering of ubiquitous systems, respectively. It preserves the holistic view upon the field (and surrounds our metaphoric elephant on all sides); its manifesto adopts the visionary year 2020, Weiser’s cross-over year between the third wave of computing and the Internet transition era (as described in Section 1.1.1) and addresses not only the engineering and social issues usually focused upon in various research groups, but also “the theory required to underpin the design and analysis of ubiquitous systems which are intrinsically large-scale and complex”.

Questions like the following are the fabric that makes Ubiquitous Computing a Grand Challenge:

“How many computers will you be using, wearing, or have installed in your body, in 2020? How many other computers will they be talking to? What will they be saying about you, doing for you, or to you? (...) Shall we be able to manage such large-scale systems, or even understand them? How do people

interact with them and how does this new pervasive technology affect society? How can non-computing people configure and control them? What tools are needed for design and analysis of these constantly adapting and evolving systems? What theories will help us to understand their behaviour?” [24]

Research is expected to look upon these questions from three distinct perspectives. The *experience* perspective has social scientists investigate people’s interaction with ubiquitous technology, and the way in which a future society will be shaped from a socio-technical perspective. The *engineering* perspective focuses on the design, network infrastructure and dependability challenges brought up by the large scale, heterogeneous and dynamic nature of ubiquitous systems. Finally, the *theoretical* perspective is intended to distinguish rigorous concepts and models which abstract at different levels the behaviour of ubiquitous systems, together with techniques for reasoning upon them at global level.

Among these perspectives’ collective goals are included the need to develop usable interaction paradigms according to individual and societal requirements, the definition of rigorous system design principles (instantiated in a number of running ubiquitous systems with a documented operational history), and the development of concepts, calculi, models, theory and tools for descriptive and predictive analysis.

This holistic manifesto—recognizing the different perspectives upon ubi-comp, while all the while noting their collective goals—has played a mentoring part upon our work. As the opportunities came about, we pursued challenges on all the coordinates of the axis concepts-theory-tools.

### 1.2.2 A Definition and Work Plan

To be able to judge the difference between the actual, registered progress of the field of ubiquitous computing and its initial promise, we first aim at recovering as precise as possible a definition for ubiquitous systems. To this end, the Ubiquitous Computing Grand Challenge [24] characterises a Ubiquitous Computing System (UCS) as being composed of large populations of *entities*, some self-deployed; an entity may be a hardware device, a software agent, for some purposes a human, or any complex of such entities. Entities pair with objects and people in the environment, and as such will share their mobility characteristics.

In order for hardware entities to be embedded into everyday artefacts, clothing and people, *size constraints* step in: ubiquitous computing thus calls for centimeter- and millimeter-scale hardware, complete with processing, sensing (possibly also actuating), and communication capabilities. Then, due to the number, mobility and spatial distribution of these devices, wiring

them to the power supply network is infeasible, and having them battery-powered is usually too short-term a solution. Hence, the challenge in what concerns hardware design switches from focusing on faster processing towards *low-power* devices and alternative, mobile sources of power.

The argument that holds for power also holds for *wireless communication*: in the case of such distributed, mobile systems, radio transmitting takes the place of a stable, wired communication scheme. On the other hand, radio transceiving is power-hungry compared to the other functions of ubiquitous nodes; for example, in the case of the modern Telos wireless sensor node from the Berkeley group [91], the active power of receiving on the radio is more than 12 times the active power of the microcontroller; furthermore, if one factors into consideration the difference in processing speeds between the radio transceiver and the microcontroller (take again the Telos case, which sees a 250kbps radio transceiver versus a 16MHz microcontroller), one gathers that the amount of energy spent by the radio when receiving a single bit is the same as the energy spent by the processor over 800 instruction cycles.

Together with the hardware-issues cited above, at runtime a UCS will exhibit the following overall characteristics (selected from the manifesto [24]):

- *Fluidity*: its structure will evolve in the long term.
- Partial *autonomy*: some of its actions are determined by its purpose and its interactive experience, rather than by invocation from a higher authority. Also, *sustainability*: its components, hardware and software, are designed and built for long life and efficient maintenance.
- *Trustworthiness*; it will behave in a dependable manner and will not adversely affect information, other components of the system or people.
- *Scalability*; its subsystems will differ in size by many orders of magnitude, yet the same design and methods of analysis are applicable.

**Context** To then detail the software behind these runtime characteristics, central to all three perspectives upon ubiquitous computing is the notion of *context*. Repeatedly defined to fit new applications [1, 34, 101] (with some definitions coming from Xerox PARC researchers), the concept of context usually denotes aspects such as where one is, who one is with, what one is doing and what resources are nearby (in other words, location, identity of neighbour entities, activity and the state of the physical environment).

From the experience perspective, when representing and inferring human environment and human activity as computational elements, dealing with

notions such as measurable physical characteristics of objects is straightforward. However, deeper notions such as a person's actual or desired activity in a current context—further nuanced by the person's past experience and individual understanding—is complex to infer.

**Context Awareness** The engineering perspective then focuses on *context awareness*, i.e. the need for ubiquitous systems to *acquire* a measure of context and *adapt* to the context's current values; heavily used contexts include location, neighbouring entities (i.e. devices or humans) and the activities they are currently involved in, or available computational and network resources. In order to acquire such context values, technologies such as *sensing* (and related: sensor data fusion, techniques for inference from sensor data, sensor data history and user input) are central, and work on the matter is partnered with the understanding of context in terms of human activities.

Another intrinsic part of acquiring context (this time network-related) is *service discovery*, the process through which those entities providing the hardware or software resources needed by an entity's application (e.g. a gateway, the nearest printer or anyone able to provide the weather prognosis) are identified in the entity's surroundings.

The design of service discovery protocols in particular is influenced by the *logical topology* of the UCS. While traditional distributed, even mobile, systems are laid out in a *hierarchical* fashion, having the luxury of being attended to by a type of infrastructure or another, e.g. in a built environment, the ubiquitous discovery protocols run also over infrastructureless deployments, such as outdoor major incident sites. In these cases, both protocols (for both plain data communication and service discovery) are designed in an *ad-hoc* fashion: ad-hoc routing works towards replacing a static routing infrastructure, while ad-hoc service discovery replaces the static configuration of resource tables. In turn, this ad-hoc nature of some UCSs allows them to be *open systems*, i.e. to impose few boundaries to system membership, instead allowing incoming entities a level of access to the system.

Adaptation to context then can take the form of *self-configuration*, when the context the system adapts to is its own state. The scale of an UCS dictates that, again, it is infeasible for human users to reconfigure such a number of devices individually; thus, the installation and evolving of software is ideally automatized, and may involve mobile code. As a consequence of the greater amount of power spent in transceiving wireless data compared to the energy spent in computational tasks, software needs to be optimized accordingly for power optimization; in-place data processing thus takes the place of network-wide data gathering (followed by processing).

**Security** On a higher level, open, ad-hoc UCSs present unprecedented security issues. We quote a number of scenarios presenting various types of security attacks from the Ubiquitous Computing Grand Challenge manifesto [24]:

“A ubiquitous application may involve collaborations between ad hoc groups of entities. It may require migration or downloading of code, and may involve people moving and changing the system configuration. New encounters occur, and there are complex issues in knowing what entities to trust. Does a server trust an agent enough to allocate processing resource to it? Does a device trust a neighbour to send message packets for onward routing? (The latter could be a ‘denial of service attack,’ aiming to deplete the device’s battery.) Does a human using the UCS trust a host, a service, a device, or another human communicating and collaborating through the system? Based upon predefined trust, recommendations, risk evaluation and analysis of past interactions, an entity may derive new trust metrics and authorisation policies for what access it will permit to its resources, what services it should refrain from using, or what security mechanisms (...) to use.”

**Theories** The exact central concerns mentioned above would also be addressed by the *theories* for ubiquitous computing. Regarding the mobility and openness of UCSs, one challenge “is to extend existing models, such as process calculi, to accommodate space and mobility; promising candidates already exist, but difficulties of analysis still remain” [24]. Regarding the entities’ awareness of context in a constantly changing context, one added challenge is to come up with models and analyses for the entities’ context acquisition, adaptation and information flow.

These theories will provide the missing assurance of behaviour predictability, of the correctness of system-wide long-term adaptation, and of satisfying security policies, among others. As such, they should then form a scientific foundation for the design and prototyping of future ubiquitous systems.

### 1.2.3 State of the Art and Prototypes

There is currently an accelerated growth in research and prototypes experienced by the field of ubicomp; yet, these are matched by a slight distrust that the paradigm of calm computing will find soon a sustainable, large-scale applicability in reality [2, 96]. In spite of this point being motivated, there are many ways in which to argue that we already live in a ubiquitous IT environment; some of the qualities which define the paradigm of calm computing can already be found in systems used on a large scale today.

The most compelling example is probably the *Octopus smart card* [76], an award-winning debit RFID card, externally-powered by near-field communication, of which 17 million are in operation in Hong Kong. Introduced in 1997 to simplify the paying of fares on public transportation, it made people's life so much easier that it was then extended not only to other purchases, but to non-commercial information transactions such as school and campus access, and personal identification. It is the most extensive contactless smartcard system in the world, with the widest scope of applications, and the highest transaction volume. 95% of the adult population of Hong Kong use it, and by now 25% of transactions are on non-transportation services; two thousand providers or institutions have adapted their interface with the customers to include the Octopus card, and others are continuously being added. All these give Octopus<sup>3</sup> the world's highest number of transactions per card per month, by far.

The card, a Sony FeliCa [33], makes possible this ubiquity of service due to a large data capacity (64KB, compared to 125B on a magnetic stripe card) and high-speed of the near-field communication (212kbps, taking as little as 0.1s per transaction) and fairly wide 3-10cm range. The ease of adding new service providers is due to the card's data being organized in a typical operating-system style, into files and directories (one for each provider, with unique access rights, so that the only action needed to add one is to create a new directory holding the user's data related to that provider). The transactions, if commercial in nature, are store-and-forward, meaning that the transaction data are registered both on the card and by the service provider, but only put through the bank at the end of the day; this eliminates the need for the card readers at the service providers to have realtime round-trip communication capability with the central transaction-clearing network. The card uses encryption for all airborne transactions, with two-way authentication, and has never been cracked.

The Octopus card is the perfect example of a digital artifact which *evolved* from a fairly traditional use into a ubiquitous device. The *ground up* approach is also being taken in the process of building new, *ubiquitous cities*.

The notion of *ubiquitous city*—or *U-city*—has already entered jargon. To quote Greenfield's *Everyware* [2]:

“A ubiquitous city or U-city is a city or region with ubiquitous information technology. All information systems are linked, and virtually everything is linked to an information system through technologies such as wireless networking and RFID tags. The

---

<sup>3</sup>The card's name itself argues for ubiquity of service: the octopus's eight legs point to a Chinese meaning of “many” for the number eight, but also to the common mathematical sign for infinity, i.e a horizontal eight (which also appears on the card's cover design). This was chosen as to point to an infinity of usage possibilities for the card [76].

concept has received most attention in South Korea, which is planning to build some 15 ubiquitous cities. The first U-City, Hwaseong-Dongtan U-City, has been partially completed and operated in 2007. It characterizes diverse U-Services that include U-Traffic, U-Parking, and U-Crime Prevention service. (...) A ubiquitous city is where all major information systems (residential, medical, business, governmental and the like) share data, and computers are built into the houses, streets and office buildings.”

The largest such Korean U-city, New Songdo City [84], is the one which got the most attention, as it is currently built *from the ground up*, as opposed to in an evolving fashion. Its design features public recycling bins that use RFID technology to credit recyclers every time they throw in a bottle; pressure-sensitive floors in the homes of older people that can detect a fall and contact help; cellphones that store health records and can be used to pay for prescriptions. Similar to the Octopus system, in New Songdo a resident’s house key is a smart card which can also be used on the subway or at the cinema. The city holds promise of being the most comprehensive framework in existence for ubiquitous computing [2].

The building of the U-cities today is a far leap from the initial incarnation of ubicomp at Xerox PARC in the early nineties. Their tabs, pads and boards [117,118,120] hit more on a note of computer-supported collaborative work between people in an enterprise, than at large-scale ubiquity.

While the physical world would come in all sizes and shapes, prototyping at Xerox PARC began with three sizes of devices: PARC’s *tab* is a palm-sized mobile computer expanding on existing inch-scale computers, whose purpose it to show that researching new hardware is less important than combining existing components. The tab is supposed to be the tiny computer, analogous to Post-it notes and to the omnipresent displays of words found on book spines, light switches, and hallways. The second size is the *pad*, envisioned not as a personal computer but analogous to scrap paper, with many being used by a person at the same time. The third size is the *board*, a wall-sized interactive surface, analogous to the office whiteboard or bulletin board. Networked in an indoor local area network complete with infrared wireless cells, the applications that these devices would run included locating people (for phone forwarding and meetings) and shared drawing.

Given the impressive scope of the U-city or Octopus system compared to early prototypes, one argues that most components required to build ubiquitous systems already exist, even for some most daring scenarios. Technological factors such as processor speed, storage, displays, wired and wireless communication protocols (and associated addressing schemes) over various physical layers, all exist to a large degree. However, as also seen by the

Ubiquitous Computing Grand Challenge manifesto [24], appropriate design patterns and standard interfaces for interoperability of devices in UCSs as large as an U-city are still missing.

#### 1.2.4 Venues

To date, we have had ten UbiComps (International Conferences on Ubiquitous Computing) and six Pervasives (ditto on Pervasive Computing), and tens of smaller, but similar venues (easily given away by names containing the particles Ubi, Per, Sys, or Mobi), most established events supported by such organizations as IEEE and businesses as Nokia, Intel and IBM, and some highly selective (e.g., the 2006 PerCom boasted an acceptance rate of 8.21%).

There exists the mandatory journal dedicated exclusively to the topic from IEEE, one of Springer's and one from Elsevier, and again a number of less-known ones, not to mention the wealth of venues with topics overlapping both ubiquitous computing and fields like mobile computing, artificial intelligence, networking, formal methods or software engineering.

# Chapter 2

## Context Awareness. State of the Art and Contribution

The preceding chapter has briefly looked into the when, how and why of ubiquitous computing, following its natural evolution out of the Internet era. All the while, we visited the technical, human and scientific facets of the field (or, we dare say, the many sides of the elephant), the current degree of transforming the vision into reality, and the need to take a unified view upon the matter. We have focused in particular on the technical and scientific challenges behind delivering ubicomp—as they are the motivation for our work—and away from issues within the user experience field.

In turn, this chapter moves focus towards the second concept in the title of this dissertation, and the unifying concept behind our work: *context awareness*. It summarizes our technical contributions (upcoming at length in Part II), after first giving some depth to the four technical and scientific topics this contribution falls within, i.e. the topics of localization, service discovery in ad hoc networks, security, and modelling and verification of central concepts in ubicomp. It places these topics of our contribution into a unified framework of thinking about context and context awareness, on a system of coordinates with two axes: one following the concept of context in ubicomp from its sensing to its use in applications, and the other ranging from practical to theoretical.

### 2.1 A First Axis: from Context to Behaviour

Since the concept of context awareness has been coined an enabling technology for ubiquitous computing, fairly precise definitions appeared, delimiting the meaning of context and its awareness in ubicomp. Many of these definitions soon after turned up to be too specific, as more types of context and features of context awareness appeared necessary in practice.

In the beginning of this section, we reinterpret and focus these definitions

from a unifying perspective: after settling on a rather recent, and more generic, definition for context awareness, we state that it includes other facets of networking or computing not traditionally thought of as context awareness, such as ad hoc routing.

An initial definition of *context* (Schilit et al. [101]) narrowed its important aspects to be where one is, who one is with, and what resources are nearby. Others, backed by a wealth of ubiquitous computing scenarios, usually focus computing context similarly towards location, identity of neighbour entities or the state of the physical environment. As the number of scenarios computing in context increased, it became clear that instead of having the definition enumerate the types of context, it should instead be more generic, as the enumeration of contextual aspects will vary among situations. Dey’s definition [1,34] delimits the context of an application loosely, as being the *situation of relevant entities*:

“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to [a user’s application], including the user and applications themselves.”

The context of an entity is more or less straightforward to read: Abowd et al. [1] divides context into *primary* and *secondary*: “location, identity, time, and activity are the primary context types for characterizing the situation of a particular entity.” When primary types of context then act as indices into other sources of context, the resulting output is secondary context (e.g., given an entity’s location as primary context, one determines the entity’s neighbouring objects; the latter is secondary context for the initial entity, yet primary context for the neighbouring objects).

Moving on towards pinpointing what makes entities *context-aware*, we find that definitions recognize two common features that bring on awareness; the first is *acquiring* relevant context, followed by *adaptation* to it. Furthermore, one or another set of added features play a role in the various definitions.

The degrees of selectiveness in regard to these other features, and even their naming again vary [1,89,101]. For example, among Pascoe’s [89] set of core generic capabilities used as a vocabulary to identify context awareness are included the above-mentioned acquiring of context and adaptation to it. The acquisition of context is instantiated as the more particular term *sensing*, and contextual sensing is the most basic level of context awareness, defined as the process through which the computing entity simply detects various environmental states and presents them to the application.

To increase precision, we restate that an entity’s context sensing is the acquisition of the entity’s *own* context. This is taken a step further if the

entity can acquire the contexts of other entities, in a process of *resource discovery*, usually achieved not by direct sensing, but by network communication. Given the logical similarities between these capabilities (i.e. context sensing and resource discovery), we unify them into the notion of *context acquisition*.

The next feature of awareness is then *contextual adaptation*, through which “applications leverage this contextual knowledge by adapting their behavior” [89]. The mechanisms for adaptation are then detailed in Schilit’s definition [101], which finds categories of adaptation on two orthogonal axes: one axis distinguishing between running a command and working with data in context, and the other looking whether the adaptation is done manually or automatically (as summarized in Table 2.1). Out of Schilit’s categories, we focus on the three not dependant on the user’s input as being the most challenging ones: the *contextual commands* are calls for data or commands, prewritten in the application’s code, which produce different results according to the context in which they are issued; the *context-triggered actions* and *automatic contextual reconfiguration* are enabled by a set of rules specifying how the application should adapt (like a rule-based expert system), by triggering actions or reconfiguring the system, respectively.

Table 2.1: Schilit’s categories of an application’s adaptation to context [101]

	manual	automatic
command	contextual commands	context-triggered actions
data	-	automatic contextual reconfiguration

In conclusion, in our view, context-aware systems are concerned with, on one hand, context acquisition; this is done on two levels:

- Context sensing: by using sensors, an entity acquires a situation (a primary context, either its own or another entity’s); this includes the fairly complex process of understanding the sensed data (i.e. matching a perceived sensor signal to a human’s or object’s context).
- Service (resource, or context) discovery: through network communication, an entity acquires another’s primary context, or its own secondary context, by network-wide service discovery.

On the other hand, context awareness is adaptation, or in fact the change in the application behaviour based on the previously acquired context<sup>1</sup>.

<sup>1</sup>Concerning the definitions of context awareness, not all definitions require that an

Table 2.2: Features of context awareness

<b>context awareness</b>	<b>context acquisition</b>	sensing: localization, activity recognition service discovery
	<b>contextual behaviour</b> (adaptation)	contextual commands context-triggered actions automatic contextual reconfiguration

Given our concluding definition of context awareness above, we find that this formulation (which we summarize in Table 2.2) unifies a number of related study topics; the principles of awareness to context include those behind ad hoc routing (essentially, an automatic contextual reconfiguration of the routing tables guided by the presence of neighbourhood entities and their configurations) and self-organization.

**Our contribution** Having this view upon context and context awareness (or, in other words, the axis from context to behaviour), we position (as in Table 2.3) our individual contributions on this axis, specifying the section in the current chapter in which the contributions are summarized and placed in context, and the upcoming chapter in which they are presented at length.

Table 2.3: The positioning of our individual contributions on a first axis from context to behaviour

	<b>contribution</b>
<b>context sensing</b>	GammaSense: Infrastructureless Positioning using Background Radioactivity [17] (Section 2.3, Chapter 5)
<b>service discovery</b>	A Calculus for Ad Hoc Context Awareness [15] (Section 2.4, Chapter 4) Secure Data Flow in a Calculus for Context Awareness [16] (Section 2.4, Chapter 4) Resource Discovery in Activity-Based Sensor Networks [13, 14] (Section 2.4, Chapter 3)
<b>contextual behaviour</b>	Verifying ANSI-C Context-Aware Applications [12] (Section 2.6, Chapter 6)

application's behavior be modified for it to be considered context-aware. A more generic definition is possible (which has an application simply present the context to the user, for example) [89]. As before, we move away from the features heavily involving the user, and focus on the automatic adaptations of application behaviour, which are more technically challenging.

## 2.2 A Second Axis: from Practice to Science

*It has long been my personal view that the separation of practical and theoretical work is artificial and injurious. Much of the practical work done in computing is unsound and clumsy because the people who do it have not any clear understanding of the fundamental design principles of their work. Most of the abstract mathematical and theoretical work is sterile because it has no point of contact with real computing.*

— Christopher Strachey, on the founding in 1965 of the Programming Research Group at the Oxford University Computing Laboratory

Besides placing our contribution within the technical topics of research upon context awareness (as in the previous section), we now proceed to placing it on a second axis of coordinates, from engineering to theory.

To quote Tom Henzinger’s input to the Royal Society meeting “From Computers to Ubiquitous Computing, by 2020” [55], among the challenges put forward by ubicomp is the fact that (embedded) systems design is a tale of two cultures: computer science and engineering. In computer science, the temptation is to equal programs, systems and networks with mathematics, and then prove the correctness of a specification. In engineering, on the other hand, the drive is to build reliable systems. One speaks about analytical, guaranteed operation, and the other about best-effort operation.

The same distinction is made by the Ubiquitous Computing Grand Challenge manifesto [24] (as already seen in Sections 1.2.1 and 1.2.2), and both argue for moving the design of systems back to the science area, for the design to become accountable for.

Our work itself covers just as much scientific ground as it does engineering, not only exploring the various technical features of context from Table 2.2—to which we now add the all-encompassing topic of *securing* context-aware systems—but also doing so from alternating, theoretical or engineering points of view. This approach was not predetermined at the start of this work, but evolved naturally as the author became part of both the Pervasive Computing group, and the formal methods group in the Department of Computer Science at the University of Aarhus.

Without going into detail as to the technical aspects of either point of view in our work—as they are discussed in what follows—we update Table 2.3, which places our individual contributions on an axis from context to behaviour, by (i) adding the final coordinate on the first axis, *security*, and (ii) adding a second axis, from practice to science. The result is found in Table 2.4; in the remaining of this chapter, we range through each individual coordinate on the first axis, giving an overview of the field and a summary of our contribution.

Table 2.4: The positioning of our individual contributions on (i) a first axis from context to behaviour, and (ii) a second axis from practice to theory. One of the papers [16] lies within two different areas (service discovery and security), and as such the relevant parts of the paper are summarized in different sections.

	<b>contribution</b>	<b>practice/ theory</b>
<b>context sensing</b>	GammaSense: Infrastructureless Positioning using Background Radioactivity [17] (Section 2.3, Chapter 5)	practice: analysis of sensor data
<b>service discovery</b>	Resource Discovery in Activity-Based Sensor Networks [13, 14] (Section 2.4, Chapter 3) Secure Data Flow in a Calculus for Context Awareness [16] (Section 2.4, Chapter 4) A Calculus for Ad Hoc Context Awareness [15] (Section 2.4, Chapter 4)	practice: discovery protocol theory: modelling
<b>securing ubiquitous systems</b>	Secure Data Flow in a Calculus for Context Awareness [16] (Section 2.5, Chapter 4)	theory: verification
<b>contextual behaviour</b>	Verifying ANSI-C Context-Aware Applications [12] (Section 2.6, Chapter 6)	theory: verification

## 2.3 Sensing Context

*DATA: “Sensors show a temporal signature emanating from the [Borg] Sphere. High concentrations of tachyons...”*

*Picard studies Data’s monitor.*

*PICARD: “And chronometric particles... it’s as though they’re trying to create a temporal vortex...”*

*A beat as Picard makes a shocking realization.*

*PICARD: “Time travel. They’re attempting time travel.”*

— A passage of activity recognition from the analysis of sensor data, in the screenplay of the science fiction movie *Star Trek: First Contact*. Brannon Braga and Ronald D. Moore, 1996

Context sensing is the basic level of context acquisition; it employs sensor mechanisms and infers primary contexts such as environmental conditions (e.g. humidity levels by a humidity sensor), location (e.g. by triangulating incoming signal strengths from a number of access points), nearby persons

(as monitored and recognized by e.g. a video camera) or people’s activity (e.g. by the analysis of the resulting noise intensity and pattern).

In the following, we first give a taxonomy of localization methods and systems for ubiquitous computing. Lengthy such taxonomies and surveys have already been published, some fairly early ones covering the entire field [57,58,80], for others to then focus on one technology or the other (e.g. the use of signal strength [39,70], the fingerprinting of radio signals [68], or the use of active beacons [63]). While we do not aim at replacing these surveys, we set up a succinct framework of localization research, together with a recent survey, as a basis for our contribution.

### 2.3.1 Localization

The topic of positioning an entity in geographic space is not specific to ubicomp. Indeed, the problem of finding a current position or route to a destination stems from historic days, with notable precursors in sea and air navigation. Traditionally, men used reference points as rough guides outdoors, and written signs or maps, which would then require the decoding of a human, indoors [63]. On the other hand, ubicomp requires fine-grained, automatic positioning for mobile vehicles, robots or computing equipment, both outdoors and in indoor environments with various structures and pre-deployed communication infrastructures. Strictly regarding localization in the open outdoor, the problem is widely considered solved by the Global Positioning System (GPS) [47]; indoors and in crowded cityscapes, on the other hand, there is no one perfect solution, and the volume of research on the matter is high.

A set of taxonomy factors differentiate among localization systems. Position measurement is either *absolute* (the majority of systems) or *relative*; if the case is such that the system is covered by a unique set of spatial coordinates into which all positions are given (such as the Earth’s geolocation grid), measurements are absolute. In large networks or sparse systems, on the other hand, positions might be calculated in a relative fashion, as distances or hop counts to (possibly mobile) anchor nodes, which thus act as separate sets of coordinates; examples include locating a vehicle relative to a region’s web of highways. Translation between the two is often done; positioning information for the anchor nodes can be used to translate between absolute and relative location.

Absolute locations can be given either as *physical* positions, i.e. points on an abstract grid (e.g. latitude and longitude) such as the GPS system, or as *symbolic* positions, e.g. landmarks on a logical map, or rooms in buildings. Symbolic location is often more useful to human activity, and easier to compute, even if coarse; it does, however, require additional knowledge about the terrain, such as maps.

As to the physical means of computing an entity's location, the sensed signal most often ranges among the following:

- *Electromagnetic* waves of various frequency, including narrow- or wide-band radio waves, infrared or microwave frequencies, light and even the electromagnetic field of the powerline network;
- *Mechanical* means: inertia, pressure and ultrasound;
- *Physical* properties of the environment.

For example, the Received Signal Strength Indicator (RSSI, or shortly signal strength) of GSM transmissions to a mobile phone from a number of cell base stations gives a potentially accurate indoor localization scheme [88]. A GPS client calculates a set of absolute distances to satellites of known positions by multiplying a constant signal propagation speed with propagation times; the Smart Floor system [87] identifies people based on footstep pressure profile, while one of our contributions, GammaSense [17], roughly pinpoints position by reading the background radioactivity levels indoors.

For wireless localization technologies (by far the most widely researched), each signalling technology is chosen to serve a specific setting, depending on the physical properties of the wave [80] (e.g. low frequency (LF) radio waves have longer range than microwaves). Both localization range and error depend on wave frequency, if the transmission is narrowband, and bandwidth (if wideband). Mostly in indoor locations, some frequencies are prone to the phenomenon of multipath distortion, and some are absorbed by wall materials.

Such signals are emitted as telemetry either by the gadget to be located, or by the infrastructure (or, in rare cases, appear naturally in the gadget's environment). The calculation of the location then happens either at the infrastructure—if telemetry is emitted by the client gadget—or at the client (if telemetry is put out by the infrastructure), or at any of the two, in the case of natural signals from the environment.

The mathematics behind the calculation employs techniques such as *triangulation*, which attempts to pinpoint the client relative to a number of beacons of known positions, given values for either distance measurements to the beacons (time of arrival, *TOA*, or time-difference of arrival, *TDOA*—essentially, the difference between the TOAs of signals from two beacons), or angle of reception (*AOA*, angle of arrival). In three dimensions, for example, a position is calculated using four distance measurements (three, if the ambiguity of height can be solved empirically) as is the case for GPS. Optimizations to the method include the min-max technique (a simplified, computation-efficient triangulation), least squares (a probabilistic version to

use with imprecise distance measurements) and other probabilistic tools, i.e. Bayesian filters and Markov chains.

In ad hoc networks, the *centroid* method has a node locate itself by calculating the geometrical center of a set of beacon nodes that it can hear. *Fingerprinting* is the technique by which—before the positioning system is delivered into use—a set of sensor measurements (fingerprints, e.g. signal strength values) are mapped to actual locations; then, a client’s location is inferred probabilistically by comparing her current sensor measurement with the existing fingerprints.

Finally, the performance of a positioning system is assessed with factors such as:

**Precision** the grain size of the localization method, in units of distance.

It is usually tailored to the specific application (e.g. location can be pinpointed up to room-sized, or over a 0.5-meter grid) and is limited by the physical properties of signal input.

**Accuracy** the frequency (or probability) of positioning with a certain accuracy. There exists a trade-off between precision and accuracy: the coarser the precision, the higher the accuracy of reaching it.

**Portability** the possibility of porting a location system to different sites with little cost.

**Scalability** the physical range an infrastructure can be extended to cover, or the number of clients supported per infrastructure unit.

We then use our succinct set of taxons above to update Hightower and Borriello’s survey [57] with recent localization systems, in Table 2.5.

Table 2.5: Post-2001 indoor localization systems and their parameters.

	Sensed signal	Position	Technique	Computation	Scalability	Precision/ Accuracy	Portability
<b>Ekahau</b> [38] 2002	IEEE 802.11	absolute, symbolic	fingerprinting RSSI	at infras- tructure	limited to Wi-Fi network	1-3m/ 50%	- requires calibration - portable to existing Wi-Fi installations
<b>Ubisense</b> [109] 2004	Ultra- WideBand: 5.8-7.2GHz	absolute, symbolic	triangulation: TDOA, AOA	at infras- tructure	limited to network of Ubisensors	15cm/ 95%	- installs large number of Ubisensors
<b>Horus</b> [124] 2005	IEEE 802.11	absolute, symbolic	fingerprinting RSSI	at client	limited to Wi-Fi network	1.32m/ 90%	- requires calibration - portable to existing Wi-Fi installations
<b>MoteTrack</b> [77] 2005	IEEE 802.15.4	relative	centroid of RSSI fingerprints	collaborative, at beacons	limited to area of deployed beacons	3m/ 80%	- installs beacon nodes - requires calibration
<b>Place Lab</b> [73] 2005	GSM, IEEE 802.11	absolute	fingerprinting beacon IDs	at client	nearly ubiquitous	21.8m/ 50%	- portable to existing GSM, Wi-Fi beacons - requires calibration
<b>GSM Indoor Localization</b> [88] 2005	GSM	absolute, symbolic	wide RSSI fingerprinting	at infras- tructure	ubiquitous	floor/89% 5m/50%	- requires dense (every 1.5m) calibration
<b>Powerline Positioning</b> [90] 2006	powerline tones	absolute, symbolic	fingerprinting amplitude of injected tones	at infras- tructure	ubiquitous indoors	3m/ 87%	- 2 signal injectors at house ends - requires calibration
<b>FiatLux</b> [94] 2007	artificial light indoors	absolute, symbolic	fingerprinting incidence of light	at infras- tructure	ubiquitous indoors	room/ 90%	- zero infrastructure - disallows windows - requires calibration
<b>GammaSense</b> [17] 2008	background radioactivity	absolute, symbolic	particle count fingerprinting	either	ubiquitous	floor/ 50-70%	- zero infrastructure - requires fingerprinting

### 2.3.2 GammaSense: Infrastructureless Positioning using Background Radioactivity

In the following, we give an overview and recall the motivation behind our paper:

- [17] Doina Bucur and Mikkel Baun Kjærgaard. GammaSense: Infrastructureless Positioning using Background Radioactivity. In *Proceedings of The Third IEEE European Conference on Smart Sensing and Context (EuroSSC)*. Springer-Verlag, 2008.

in anticipation of its published form in Chapter 5.

GammaSense was born out of an interdisciplinary effort similar to that of the Powerline Positioning system [90] and FiatLux [94] (from Table 2.5); when looking for sensor signals—other than the well-researched electromagnetic waves—which would yield a degree of localization indoors, we came upon one which was fairly promising in theory, and never researched before as a source of positioning: *natural background radioactivity*. The radiation appears naturally in the environment; it is thus ubiquitous, and more so than any type of communication infrastructure (be it GSM or Wi-Fi), system of powerlines or artificial light.

A relatively rich set of physics studies came with the information that the levels of natural radioactivity vary—naturally—mostly with soil components both indoors and outdoors; more interestingly though, the geometry of the source of radiation indoors is finer than outdoors, due to the construction materials being both a source and a trap for radiation.

Radioactivity is the natural phenomenon in which certain chemical elements emit radiation spontaneously, in the form of either electromagnetic waves or of ions. The cause of this emission is radioactive decay, i.e the spontaneous transmutation of an unstable parent chemical element into a more stable daughter element. One form of radioactive decay is *beta decay*: if a neutron transmutes into a proton within the parent element, a negatively charged electron  $e^-$  with high kinetic energy (a  $\beta^-$  particle) is expelled from the nucleus. This electron is called a *beta particle* or *beta ray*.

Another produce of radioactive decay are *gamma rays*, an electromagnetic radiation having the highest frequency and the shortest wavelength within the electromagnetic spectrum. Neutrons and protons occupy well-defined energy levels in a nucleus, and when either of these particles is excited to a higher unoccupied level, the excited nucleus decays to a lower energy state, and the difference in energy is emitted as gamma radiation. The range of beta particles is short: a 1.9mm sheet of aluminium stops a 1MeV beta particle; on the other hand, gamma rays have long range and high penetration power: to reduce the intensity of a 0.5MeV gamma ray to 0.37 of its initial value, one needs to lay a shield of 4cm of aluminium or

28.6cm of tissue paper [98]. Given these facts, only beta and gamma rays are expected to register on a radiation counter; the range and penetration power of *alpha particles*, another widespread effect of radioactivity, are much shorter.

Surprisingly, radioactive levels indoors were shown to be on a worldwide average 1.4 higher than outdoors [110]. The majority of the beta and gamma concentrations indoors are terrestrial in origin, with the gas radon  $^{222}\text{Rn}$  of terrestrial origin in highest proportion; the earth beneath the building plays the most part, followed by the walls and ceilings. Even more, a small number of physics surveys found differences among the particle counts in various rooms of the same household, due to their different use. Given these results, we expected to be able to consistently harvest indoor radiation levels for positioning.

As sensor, we used a standard Geiger-Müller tube. Our experiments did confirm the expected indoor geometry of the radioactivity source. In one of our testbed buildings (a two-level domestic house), both the mean and the variance of radioactivity readings differ visibly between the two rooms on different floors, as in Fig. 2.1. The mean over a 1-hour-long continuous length of 1-minute counts shows a 10.32% decrease from the ground-floor room (mean 30.79 counts per minute) to the first-floor room (mean 27.61 counts per minute), and a striking difference in standard deviations (a 26.43% decrease, from 5.75 on level zero to 4.23 on the first level). This fact verifies that the major radiation source indoors is the subjacent earth, with its influence being diminished and smoothed with rising levels.

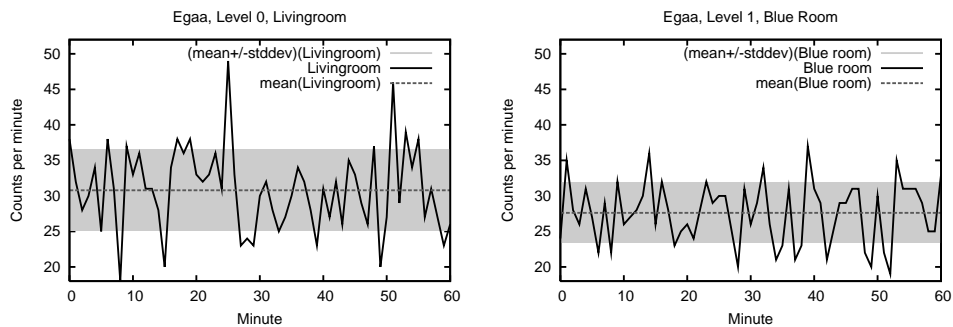


Figure 2.1: The difference in radiation measurements over 1-hour periods between rooms on different floors in a domestic house in Egaa, Denmark

We then used the technique of fingerprinting the particle counts at a number of locations on various floors in four buildings of different structure and materials. A machine-learning algorithm was then able to recognize a client's current particle count on a particular floor with an accuracy varying with the building; our best result was 72% floor recognition in an old

stone-and-brick domestic house. Surprisingly though, we found even higher accuracy for detection among wings of the same floor, in the case of another building of peculiar multistage construction.

This type of site-dependent accuracy counts towards our system, since GammaSense is based on fingerprinting. If the case is such that signal varies in a complex fashion with the site parameters, no mathematical method could easily be created to describe the geometry of the signal consistently. Fingerprinting, on the other hand, creates the site’s unique signal map as a first step.

The great strength of our work is the ubiquity of the natural background radioactivity signal, together with the complete lack of infrastructure installation. This is where the good news end, though: the accuracy of the method is not high enough to employ it on its own, the commercial Geiger-Müller sensor as we used it is not precisely portable and a client’s reading needs to be long for good accuracy of positioning (the reported results above are based on 5-minute counts, and the accuracy keeps increasing up until 8 minutes).

## 2.4 Service Discovery

*“All truths are easy to understand once they are discovered; the point is to discover them.”*  
— Galileo Galilei

*Service discovery* (in other names, *resource* or *context discovery*) is logically the other facet of context acquisition, besides context sensing. It allows an entity to acquire context indirectly: instead of having its own context sensed (with computation taking place either locally at the client or at the infrastructure, as detailed in Section 2.3 on localization), the entity now acquires another player’s own context, with the ubiquitous network as intermediary. Examples of service discovery include locating the nearest printer (i.e. its IP address on the local network), the nearest printer’s capabilities (can it print transparencies?). In an ad hoc setting, an example is locating the identity of all patients currently under life-threatening conditions, at a major emergency site (e.g. a train crash situation, in which a large number of patients are monitored with body sensors at the site).

Just like the other form of context acquisition (i.e. context sensing), service discovery provides an entity with the current value of an environmental parameter, in order for contextual adaptation to then take place.

**The Engineering View** To go back one step in the past, service discovery protocols were created specifically for mobile networking, a precursor of ubicomp: they allow a roaming gadget to automatically discover the identity

and services offered by devices on a newly entered network. The classical protocols, i.e. Sun’s Jini [79] (more recently under the name Apache River), Microsoft’s Universal Plug and Play [42] (UPnP) and Apple’s Bonjour (formerly Rendezvous) are all implementations of a set of techniques—known as Zeroconf, or Zero-Configuration Networking [27]—which create a usable IP network of services without preconfiguration. The implementation which reached the status of proposed standard is the Service Location Protocol [41] (SLP, defined in RFC 2608 under the Standards Track category).

All these versions were tailored for resource-rich, static-network enterprise environments. They rely on stable, LAN-like network connections, are independent of the network layer, are usually centralized—using one or more directories to maintain a correct list of the services on the network—and can employ sophisticated service semantics. Some ubiquitous computing systems do enjoy some, but not all of these advantages: indoor systems are usually resource-rich, but the degree of mobility of entities is high, and with this mobility comes a highly dynamic context.

Fig. 2.2 shows the centralized variant of the SLP protocol, in which a network entity carrying a service (the Service Agent) *registers* to a caching entity (the Directory Agent, a role usually taken by any non-dedicated network entity). These cached advertisements are used by the directory to give *service replies* to users’ *service requests* about the existence and identity of a service, on behalf of the services themselves. For small networks, the same approach can hold without the directory: any user multicasts or broadcasts service requests network-wide, and is sent back a unicast service reply from the entity carrying the service.

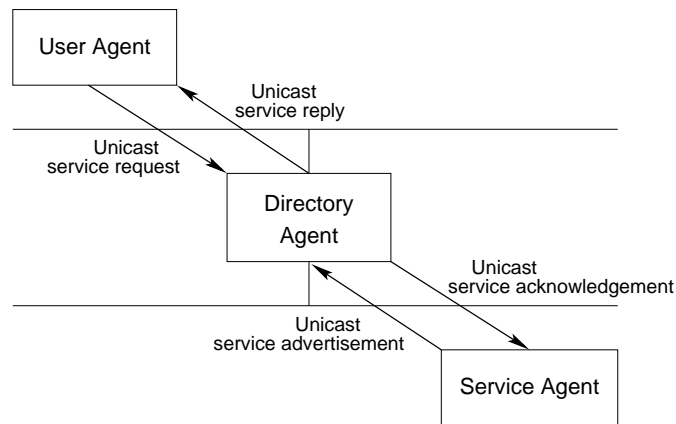


Figure 2.2: Service Location Protocol (SLP [41]) centralized protocol of discovery with a Directory Agent

On the other hand, when catering for large-scale *sensor networks* (such as in the major emergency case mentioned above) sensors form unstable ad

hoc network connections, unlike any LAN devices. Such *Mobile Ad hoc Networks* (MANETs) are autonomous systems of intermittent, energy-bounded nodes collaborating in the absence of any centralized support. The network diameter is large compared to a node's range, any data transport is multi-hop, and the nodes do not have a priori knowledge of the topology of the network. Because of the network diameter ratio to wireless range, routing and power efficiency are central issues, and each node has to act as a router. Due to the size and mobility of such a sensor network, no classical discovery schemes can hold, and the discovery scheme becomes as central as routing and power efficiency. A survey of existing discovery protocols for MANETs (up to year 2006) is included in Chapter 3.

A critical factor is added to the equation, though: in an ad hoc network, resource availability is tightly linked to route determination to that resource; i.e., finding the address of a resource is redundantly followed by finding a route to that address. Hence, separate software layers for routing and resource discovery are redundant, as argued by [69, 114]; of these, the Extended Zone Routing Protocol (EZRP) takes the step forward and designs a service discovery technique by piggybacking service information in routing packets, achieving in fact a power-efficient, embedded routing-and-discovery ad hoc protocol.

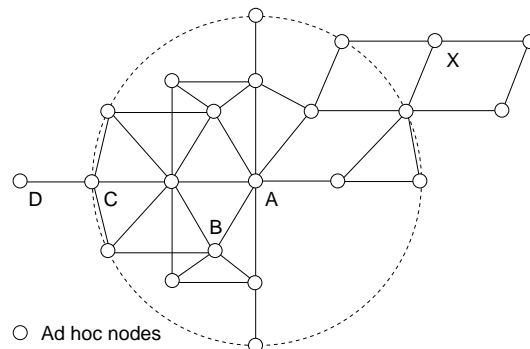


Figure 2.3: A view over an EZRP network: each node (in this example, node *A*) caches the service map of a zone of fixed radius centered at itself.

A view over an EZRP network is given in figure 2.3. Node *A* receives regular hello routing packets from its direct neighbours (complete with a service ID field) and will periodically transmit (routing-and-service) advertisements announcing its list of neighbours and their services. *A* will thus have in its local *cache* a complete view over its *zone* (a region of fixed radius centered at *A*), route- and service-wise, and needs not send service requests if the looked-up service is listed in the cache. In order for *A* to discover services and routes for nodes *D* or *X* (i.e. outside its cached zone), a form of

multicast service request is still used. Mobility is catered for by employing time counters which track the disappearance of advertisements.

**The Theoretical View** There are few direct formal models of context discovery in a mobile ubiquitous network of entities, and in the following we overview those using the formalism of *process calculi*. Some of these abstract the technical, network-related matters of discovery greatly: Braione [11] eliminates the factor of networked discovery completely and concentrates directly on the behaviour of the application; it designs a process calculus creating contextual reactive systems (CRS) upon reactive systems (RS); the difference between a CRS and a RS is the presence of a function which associates elementary rules and their allowed reaction contexts, expressing inhibitor and enabler factors for a reaction (i.e. Schilit’s contextual commands). Instead, Birkedal et al. [8] propose a complex model of context awareness able to model mobility, the dynamicity of, and queries upon context. Kjærgaard and Bunde-Pedersen’s Conawa calculus [67] models the existence of services using several context “trees”, one for each category of context information (e.g., one for location information, one for activities and one for printers); an entity will have a concurrent pointer-like presence in all trees—the current position in one tree determining the set of discoverable resources—and its moves occur in multiple context trees, thus modelling the discovery of resources under mobility.

All of these contributions on the topic, as well as ours, model not only the means of acquiring context by discovery, but also a measure of contextual behaviour (two features of context awareness we made a distinction between, in Table 2.2). However, none of these works is a fully-fledged model for contextual behaviour, and most (ours included) focus on those aspects of the matter related to *concurrency*, i.e. network-wide aspects, such as context availability (i.e. its ability to be discovered); hence, they are included under the current section.

In the remaining of the section, we summarize those of our contributions which fall under the topic of discovery, first on the practical side, then on the theoretical. A note regarding the vocabulary is in order: due to the heritage brought in by these separate views, what is practically denoted as “service” or “resource” becomes the more abstract “contextual information” or even simply “context” in the theoretical work.

### 2.4.1 Service Discovery in Activity-Based Sensor Networks

In the following, we briefly overview and motivate our doubly-published paper (presented in Chapter 3):

- [14] Doina Bucur and Jakob E. Bardram. Resource Discovery in Activity-Based Sensor Networks. In *Mobile Networks and Applications (MONET)*,

volume 12, numbers 2–3, pages 129–142. Springer Verlag, June 2007.

- [13] Doina Bucur and Jakob E. Bardram. Resource Discovery in Activity-Based Sensor Networks. In *Proceedings of the First International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–10. Nov 2006.

The contribution looks into the issue of coming up with a performant service discovery scheme in sensor networks, and preserves the routing-and-discovery piggybacking technique; it caters to an added challenge, though: the human factor brought in by the concept of *Activity-Based Computing* [5].

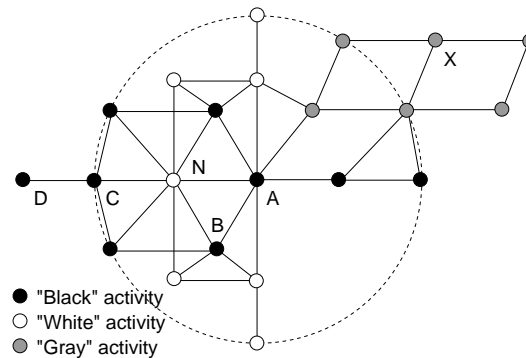


Figure 2.4: The logical view over an activity-based network: each activity is colour-coded and the intra-activity connections are drawn.

The core idea in activity-based computing is to help users organize computational services, data and resources in logical bundles that match their work activity. These bundles are called *computational activities*, or simply *activities*. For example, a healthcare activity would be the monitoring of a patient, which can take the concrete form of the prophylactic monitoring of Mr. Hansen for congestive heart failure by monitoring a combination of parameters like blood pressure, ECG, weight, and pulse. Similarly, at a major accident site, a monitoring activity for each victim could be created by bundling sensors monitoring respiration, pulse, oxygen saturation, temperature, and blood pressure. On a larger scale, a care activity would group the body sensors of all patients classified to be in critical condition after triage. An ABSN network would look, from a logical point of view, as in Fig. 2.4, instead of a “flat” topology.

We thus call by *Activity-Based Sensor Networks* (ABSN) those large-scale sensor networks using the clustering of sensors into logical activities, and basing both data routing and service discovery on each sensor’s knowledge of activity membership. This optimizes the latency of locating a service,

given that at a major incident site, most requests for services happen within the logical bounds of one computational activity (e.g. node  $A$  will more often be transacting with  $B$  and  $C$  than with  $N$ ). We optimize EZRP for use in this activity-driven scenario, by superimposing the high-level, “activity” view upon the sensor network with EZRP’s routing and discovery technique; we leave the more technical details for Chapter 3.

We believe the strength of this paper to be, similarly to the case of the GammaSense localization paper previously presented in Section 2.3.2, its interdisciplinarity: not only we supported the idea of embedding the software layers responsible for routing and service discovery (supported by only a couple of predecessors [69, 114]), but we throw in the mix research from the field of Computer-Supported Cooperative Work, CSCW (namely, the concept of Activity-Based Computing).

### 2.4.2 Modelling Service Discovery

This section overviews our work on modelling service discovery, based on the following papers:

- [16] Doina Bucur and Mogens Nielsen. Secure Data Flow in a Calculus for Context Awareness. In *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 439–456. Springer Verlag, June 2008.
- [15] Doina Bucur and Mogens Nielsen. A Calculus for Ad Hoc Context Awareness. Published online at <http://www.daimi.au.dk/~doina/>. Working paper. 2008.

The papers come up with theoretical models of (i) *context*, also referred to as *contextual information*, and (ii) context (or service) *discovery* in large, mobile, context-aware systems. The latter is, in practice, designed in different fashion over LAN (or any infrastructured) systems than over MANETs, as already argued in this section; thus, we separated the problem in two: the first paper listed above comes up with a model for context discovery in infrastructured environments, while the second, a work in progress, completes the picture with the ad hoc version.

Models are there primarily to focus the questions about the modelled system, and thus the model for infrastructured discovery has as follow-up (in Section 2.5.1) a system for the verification of security policies.

**A Model of Context Awareness in Hierarchical Systems [16]** Our calculi are based on the calculus of Mobile Ambients [21]. In this paper, computing entities in the system are modelled by mobile ambients enclosing processes and other ambients:  $a[P]$ , and the topology of the network evolves

by the ambients' running of in and out movements  $in\ a.P$  and  $out.P$ , as is standard in Mobile Ambients. We model contextual information (i.e. pieces of context, resources or services), distributed through the network over scopes of various sizes and expressed by *named macros* residing at ambients (an idea first found in [125]).

Just as services are *provided* by one entity (e.g. all printers provide a printing service) and *discovered* by another (e.g. the closest printer to the current location), our macros are *defined* and *consumed* by being *called*; a macro definition

$$def\ f \triangleright Q\ in\ P$$

defines macro  $f$  (e.g., “print”) as being the process  $Q$  (e.g., “send to printer T219”) in a floating definition ( $f \triangleright Q$ ) and continues execution with  $P$ , and any call  $f$  for this macro would be replaced by its body  $Q$ ; a simplified semantics for a definition and call are:

$$\text{DEF } def\ f \triangleright Q\ in\ P \longrightarrow (f \triangleright Q)P \quad \text{CALL } (f \triangleright Q) f \longrightarrow Q$$

and a more complete syntax follows (with a number of details still left out):

processes	$P$		$P \mid P'$	parallel composition
			$f^a$	macro call, $f \in \mathcal{F}$
			$def^a D\ in\ P$	macro definition
			$a[P]$	mobile entity, $a \in \mathcal{A}$
			$E P$	public definitions
			$\nu z P$	name restriction, $z \in \mathcal{F} \cup \mathcal{A}$
			$in\ a.P$	movement in
			$out.P$	movement out
definitions	$E$	::=	$(D)^a$	floating definition
	$D$	::=	$f \triangleright P$	macro

Note the “destination” ambient names  $a$  decorating definitions and calls,  $f^a$  and  $def^a$ : definitions and calls of contextual information are not only made by processes to and from their enclosing ambient, but cross multiple ambients' boundaries; process  $def^b f \triangleright Q\ in\ P$  publishes macro  $f$  at any ambient  $b$  in the ancestor ambient line of this process, and process  $f^b$  calls macro  $f$  from any such ambient. The calls and definitions being tagged with the name of a destination ambient fits the infrastructure setting, in which mobile entities have a degree of knowledge about the identities of servers, gateways and about the protocols in the network, at least such that identities of other points of interest can be provided by calling these.

For this, a process defining macro  $f$  to ambient  $b$  evolves into a floating definition destined to  $b$ :

$$\text{DEF } def^b f \triangleright Q \text{ in } P \longrightarrow (f \triangleright Q)^b P$$

for the floating definition to travel upwards to destination in fluid movements of the form

$$(f \triangleright Q)^b P \mid R \equiv (f \triangleright Q)^b (P \mid R)$$

(by a structural congruence rule) and

$$\text{UP } a \left[ (f \triangleright Q)^b P \right] \longrightarrow (f \triangleright Q)^b a [P]$$

(by a semantics rule). When called, a floating definition moves down from its host ambient to the calling process by the inverse of the upward movements above:

$$\text{DOWN } (f \triangleright Q)^b a [P] \longrightarrow a \left[ (f \triangleright Q)^b P \right]$$

for the call to be fulfilled at the calling ambient:

$$\text{CALL } (f \triangleright Q)^b f^b \longrightarrow Q$$

This scope extension for contextual information follows the idea that context is formed by publishing information from a source to a wider locality of users (i.e. upwards in the hierarchical system); it allows for the modelling of context discovery over entire localities of agents.

This scheme also effectively models Schilit's contextual information and commands from [101]: a call for a service has a dynamic interpretation varying with context. Furthermore, the intermediate steps a floating definition takes in order to reach a calling ambient gives that either contextual information or its destination can unexpectedly become unreachable with the ambients' changing of location; this fits the profile of highly dynamic networks.

**A Hierarchical Example** As an example, take a hospital's ubiquitous system supporting collaboration among mobile employees (inspired by [7]); the hospital network infrastructure is ambient  $hni$ , and a doctor's personal digital assistant  $doc$  currently residing in the operating ward  $ow$  updates the network about his location  $docloc$  of current value  $P$ , so that the tag of any nurse (at any location in the hospital, such as office  $of$ ) to locate him:

$$hni \left[ ow \left[ doc \left[ def^{hni} !docloc \triangleright P \text{ in } 0 \right] \right] \mid of \left[ nurse \left[ docloc^{hni} \right] \right] \right]$$

The nurse's code cannot execute without the  $docloc$  macro available in its context, but after the definition becomes visible the system is:

$$hni \left[ (!docloc \triangleright P)^{hni} \left( ow [doc []] \mid of \left[ nurse \left[ docloc^{hni} \right] \right] \right) \right]$$

and one instance of the definition travels downwards to meet the request:

$$hni \left[ (!docloc \triangleright P)^{hni} \left( ow [doc []] \mid of \left[ nurse \left[ (docloc \triangleright P)^{hni} docloc^{hni} \right] \right] \right) \right]$$

and  $(docloc \triangleright P)^{hni} docloc^{hni}$  reduces to  $P$ .

**A Model of Context Awareness in Ad Hoc Systems [15]** For ad hoc systems, the key concept is in turn not a logically vertical communication of code, but *broadcast* dissemination, and entities have no hierarchical relations. As such, each ambient moves independently of the others among locations  $l$  and  $l'$ :

$$\text{MOVE} \quad a_l[\text{move } l'.P \mid Q] \longrightarrow a_{l'}[P \mid Q]$$

We borrow the well-researched techniques for routing (i.e. route discovery) from the ad hoc routing protocols, to design a service discovery protocol. In ad hoc routing, the process of building one's routing table can be *proactive*: every node in the network maintains updated data about all network entities. For this, they all take initiative in distributing information about their current view of the network. Similarly, service discovery is the proactive broadcast of contextual information  $\text{def}^\odot D$ :

$$\text{DEF} \quad \text{def}^\odot D \longrightarrow (D)^\odot 0$$

$$\text{DEFBCAST} \quad a_l[(D)^\odot P] \mid \prod_{b,l'} b_{l'}[Q] \longrightarrow a_l[P] \mid \prod_{b,l'} b_{l'}[(D)Q]$$

in which all receiving entities  $b$  will have a copy of the definition, and can subsequently call it.

## 2.5 Securing Networked Ubiquitous Systems

*“You’ll see things here that look odd or even antiquated to modern eyes, like phones with cords, awkward manual valves... computers that barely deserve the name. It was all designed to operate against an enemy who could infiltrate and disrupt even the most basic computer [networks]. Galactica is a reminder of the time when we (...) literally looked backward for protection.”*

— Aaron Doral, around year 7300, on the Battlestar Galactica starship just before the Second Cylon War broke out. The virus infiltration tactic was used in the First Cylon War against the humans; in response, they began using simpler technologies: battlestars like Galactica used computers, but didn't network them or tie them to external sensors or communications. This would prove crucial in the Second Cylon war:

because *Galactica* continued its “no networks” tradition, the Cylons could not overcome the battlestar by electronic subversion.  
*Battlestar Galactica* science fiction series, 2003.

Security issues in computing have become acute mostly with the advent of networking, when a safety flaw in any of the network’s nodes would then potentially affect all others. Add to this the fact that a ubiquitous systems is not only networked, but also heavily distributed: many environmental objects are now computers forming an unprecedentedly large, open network population.

Classical security caters for specific issues such as *authentication* (confirming an entity’s identity), *access control* (mechanisms to permit or deny the use of particular resources by particular identities), *trust* (methods to dynamically assign levels of credibility to the members of a community), *delegation* of activity, *integrity* and *non-repudiation*. In ubicomp, on the other hand, some of these issues become critical. A ubiquitous application may involve collaborations between ad hoc groups of entities; no central authority might exist to rely on for large-scale authentication, and no single access control list can predict which resources will be used by whom. It may require migration or downloading of code, and may involve mobile, collaborating entities, in which case the entities responsible for delegation might be able to hide their tracks. Unpredictable encounters occur, and there are complex issues in knowing which entities to trust.

Traditional mechanisms for resource access-control will be inappropriate, as they either rely on single, central authorities, or do not scale to the large set of identities in ubicomp, or are of a too static nature. Many modern distributed systems use a combination of access control lists and user authentication, usually implemented via a public key authentication protocol; a request to perform an action is allowed or not by authenticating the public key, effectively linking the key to a user identity, and then looking up the identity in the access control list to check that the identity is authorised to perform the action.

Blaze et al. [9] argue for a number of reasons why the traditional approach to authorization is inadequate in distributed systems. One is the fact that authentication is establishing identity, and while in traditional static environments such as local networks, the identity of an entity is well-known, in distributed applications this is often not the case: on many occasions, entities are *anonymous*, or not previously encountered. Also, the autonomy of ubicomp entities means that different entities have different relationships with other entities, and thus, different security requirements. They will then specify *local security policies*.

Our contribution to the field is then in designing traditional *access control* in untraditional, distributed environments: having as *resources* the ser-

vices provided by ubiquitous entities, and as *users* the mobile, collaborating, (possibly anonymous) entities in the open system, we give a mechanism of *distributed access control* with any entity enforcing *local policies*, which we summarize in the following.

### 2.5.1 Secure Data Flow under Distributed Access Control

Take the above considerations upon the use of access control under challenging network conditions such as distribution of resources, locality of policies and anonymity of users. Add the challenge of imposing such policies of access control not over flat network topologies, but over *hierarchical* ones (as described in Section 2.4.2); a topology, that is, in which entities (e.g. a doctor and a nurse) located in “sibling” locations (e.g. two hospital wards on different floors) communicate by involving all intelligent entities up until the unique one they have as common parent (e.g. the hospital’s local area network, covering all floors). The message starting from a doctor’s wireless digital assistant  $d$  will be caught by the wireless point of presence in the doctor’s ward, forwarded to the switch on the ward’s floor, from there to the switch on the nurse’s floor, on to the nurse’s ward, and finally to the receiving digital assistant  $n$ .

Ubiquitous systems serving built environments are usually of such hierarchical topology (including examples such as the iconic Xerox PARC prototype system of tabs, pads and boards [117, 118, 120]). Logical partitions are imposed over the network and all communication between mobile entities is mediated by the infrastructure; rooms, floors, buildings, and campuses are such logical cells which act as communication mediators for the entities in their scope.

Have then as resources, in this hierarchical topology, the services offered by an entity and called by another; as users, all entities. The resulting *access control* policies allow or deny the offering or calling of services by certain entities (possibly anonymous); it could require, for example, that visitors to the hospitals are not allowed to be told any doctor’s location (a service offered by a server on the hospital’s network, throughout the hospital); or, a traditional entrance policy could state that visitors are only allowed to enter the first floor of the hospital, where the visitors’ ward is located; or, that anybody on the top floor can query doctors’ location, given another policy already in place only allowing hospital personnel on the top floor. To allow for a *fine grain of policy* (i.e. different access policies for any two smallest, independent locations in the scenario, and crucial for the *locality* of policies), the control should be reinforced at the “lowest” possible level in the hierarchy (e.g., at the ward level).

Finally, take this scenario and multiply its size and numbers (e.g. from a hospital to a town); it then becomes a stronger requirement that access

control policies are allowed to be local, instead of centralized.

The focus of the second part of our paper:

- [16] Doina Bucur and Mogens Nielsen. Secure Data Flow in a Calculus for Context Awareness. In *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 439–456. Springer Verlag, June 2008.

is to verify the satisfying of distributed security policies in such a ubiquitous system, which we summarize in the following.

### Distributed Access Control

Recall the scheme of resource *discovery* (and its dual, *provision*) as modelled in our Calculus for Context Awareness summarized in Section 2.4.2, *Modelling Service Discovery*, and precisely inspired by our definition of context awareness in Section 2.1, *A First Axis: from Context to Behaviour*.

As is natural for Mobile-Ambients-based calculi, policies reside at the ambient membrane. A *subambient* of  $a$  is any ambient enclosed by  $a$  either directly, or at any inner level. Then, we write  $a_G^\tau[P]$  to specify that  $P$  should satisfy the static policies  $\tau$  and  $G$ , where  $G$  denotes an *entry policy* and is a set of ambient names:

$$\text{Entry policy:} \quad G \subseteq \mathcal{A}$$

and where  $\tau$  is a set of rules, each allowing an (anonymous) ambient to have certain *effects* upon another ambient; an effect is any set of definitions or calls (i.e. in practical terms, instances of provision or discovery) for macro names (i.e. service names):

$$\text{Effects:} \quad \mathcal{E} = \bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$$

$$\text{Security policy:} \quad \tau = \mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$$

If  $b \notin G$ , then  $P$  should not have a subambient  $b$  at all, and if  $\tau(b)(c) \not\equiv def(f)$ , then  $P$  should not have a subambient  $b$  directly enclosing a process  $def^c f \triangleright Q$  in  $R$  (and similarly for macro calls).

As an example, the policy  $\tau$  of the hospital network infrastructure  $hni_G^\tau$  could be such that only employees have access to the patient record of a certain VIP, accessible through the protocol (i.e., macro name)  $vip$  directed at  $hni$ ; thus, for any visitor  $vis$  the policy states that  $\tau(vis)(hni) \not\equiv call(vip)$  and applies throughout the hospital system. Supposing that one floor  $floor3_C$  of the hospital is closed to anybody besides the hospital's employees,  $vis \notin C$ , so that no further policies upon  $vis$  are needed inside  $floor3$ .

Given our hierarchical network topology of mobile agents, each hosting a policy, a process sitting in the scope of a set of ambients should comply with the collected policies of those ambients. The initial state of a system is statically checked for compliance with all the system's policies, and then only individual moves of ambients are checked dynamically, i.e. in the operational semantics.

We do not detail the matter of defining a particular syntax for policies, but we assume such a syntax to infer a notion of *free names* of policies  $\tau_G$ , written  $fn(\tau_G)$ . For all non-free combination of names, the policies are *implicit* (either restrictions or allowances); for the free names, policies are *explicit* (either restrictions or allowances), in both cases finitely many.

### Tractable Verification

A dual type system is then put in place to verify that all code (i.e. processes) potentially executing complies with the collected policies of ancestor ambients. A type statement of the form

$$a_G^\tau \vdash P$$

states that  $P$ , running locally to ambient  $a$ , complies with the type  $\tau$ ;  $\tau$  includes not only ambient  $a$ 's local access control policies, but also those of ambients ancestor to  $a$ , and is composed during the top-down *static* type-checking with rules such as:

$$\text{AMB} \quad \frac{b \in G \quad b_{G \cap H}^{\tau \cap \sigma} \vdash P}{a_G^\tau \vdash b_H^\sigma[P]}$$

which allows  $b_H^\sigma[P]$  to be executed under  $a_G^\tau$  if (i) ambient  $b$  is allowed by  $G$  to exist under  $a$  and (ii) the inner process  $P$  is allowed to be executed under  $b$  by the *composed* policies  $G \cap H$  and  $\tau \cap \sigma$ .

With such top-down composition of policies, a piece of *active code* (i.e. executable at any moment) is verified in the fashion of

$$\text{DEF} \quad \frac{\tau(a)(b) \ni \text{def}(f) \quad a_G^\tau \vdash P}{a_G^\tau \vdash \text{def}^b f \triangleright Q \text{ in } P}$$

in which the definition passes the check if the service name being now made public is allowed to be so by all policies above, and if the continuation process  $P$  then passes the same check itself.

The dual facet of the type system concerns *passive* code, i.e.  $Q$  in the DEF rule above; this is mobile, migrating code which is not executable until located at the destination ambient, and explicitly called. This code is again type-checked (partially statically) with another set of rules.

We then define *well-typedness* as being the dual, active and inactive, checking of processes; we define as an *error* the instance in which there

exists a process which breaks at least one policy installed in an ancestor ambient, and prove that if a system is well-typed, no error can ever occur in any further evolution of the system.

As a side comment, the verification caters to *anonymous entities*, e.g. entities which choose to hide their identity. Such an entity is for example  $b$  in the system:

$$a_G^\tau[\nu b \ b[R]]$$

and the verification procedure would then alpha-convert  $b$  to a non-free name in regard to  $\tau_G$ , and pass the process  $b[R]$  through the *implicit* rules in  $\tau$  and  $G$ .

## 2.6 Contextual Behaviour

The last feature of awareness in our definition from Section 2.1 is *contextual adaptation*, the process through which applications adapt their *behaviour* to the context values the application has acquired. Given the dynamicity of ubicomp systems, though, context values, and even the availability of context providers in one's environment is by no means deterministic.

Developing the software for context-aware applications to work under unpredictable execution environments has specific difficulties. On one hand, the unscheduled nature of contextual updates incoming to an application has fueled the development of *asynchronous* languages, which specialize in managing interaction in I/O-heavy systems: an asynchronous function call doesn't start execution at call time, but is instead appended to a task queue, from which a dispatcher pops one task at a time and runs it to completion. Such deferred execution is the basis for event-driven programming languages, in which tasks are callback code prompted into the dispatcher's queue by an input event, and whose execution is delayed until the system is idle. Recent languages centered around a support for asynchrony [43,66] include nesC, a language for writing deeply networked systems and the programming platform for the sensor network operating system TinyOS [106].

On another hand, the opposite approach at tackling the complexity of contextual updates is the development of *middleware*-based applications, in which a specialized middleware software layer takes over the non-trivial process of discovery and interpretation of context updates, thus leaving only the task of adaptation to the application. A middleware can employ the same asynchronous schemes for managing context updates, with the application then being a standard multithreaded program.

The focus has only recently moved towards the analysis of the change in application behaviour with incoming context. Specialized techniques for the *verification* of such contextual behaviour are lacking, and filling this gap

is the intention of our work in progress. We build on the few, and recent techniques for the *validation* of such programs [78, 108, 115].

Of these, Wang, Elbaum and Rosenblum's [115] work on automated generation of tests of context awareness for Java programs sets the initial tone; it improves the test suite of context-aware Java *code* by identifying program points where context changes affect program behaviour, for then to construct sequences of such points as test cases. The other validation techniques [78, 108] use *models* of context-aware applications, of a flavour similar to Schilit's definition of application features (i.e. context-triggered actions [101], as described in Section 2.1). While both approaches (code and model) are legitimate, our work in progress (summarized in the following) takes the approach of verifying actual application code.

### 2.6.1 Verifying ANSI-C Context-Aware Applications

The long-term purpose of our work in progress (still in the early stages):

[12] Doina Bucur. Verifying ANSI-C Context-Aware Applications. Published online at <http://www.daimi.au.dk/~doina/>. Working paper. 2008.

is to provide verification mechanisms which ensure the dependability of applications adapting to context; these applications can be fully aware asynchronous programs, or multithreaded code supported by a middleware layer, as described above. The fundamental difficulty in verifying such adaptive programs is the fact that context updates affect the behaviour of the application involuntarily, at any time during execution: the handlers of asynchronous events preempt the running of the main program or that of any task, or the middleware sprouts threads running context handlers at undetermined times. This can happen for any type of input, but it is particular for context inputs, which are delivered streamingly (e.g. sensor chips feed readings to the application periodically, or all new members of the network neighbourhood broadcast hello messages).

An initial *programming difficulty* resulting from these facts is the need for developers to identify when the streams of context updates impact the behaviour of the application.

We find that, even if the issue of verifying behaviour adaptation is new, the techniques for it should be the specialization of existing analysis and verification procedures, such as the securing of information flow [10], or the generic CounterExample-Guided Abstraction Refinement (CEGAR) technique [29] driving the construction of an increasingly complex abstraction over which to model-check specifications.

For all purposes stated above, we settled on programs written in C-like languages; our intended case studies include nesC applications for sensor networks and C++ industrial programs.

Given the problem of analysing the influence that an external contextual input (e.g. variable `ctx` sent as an argument to the handler `context_handler` from the middleware, as in Fig. 2.5) has upon a multithreaded or event-based program, we settle on identifying the set of *context-aware program points*. This set is composed of program statements which (i) have the side-effect of leaking a measure of the current value of `ctx` to another variable, such as an assignment `power=ctx` where `power` is a global variable, or (ii) have an effect which varies with the current value of `ctx` or a leaked-upon `power`, such as the expression `if(power) E1 else E2`.

---

```

unsigned char power;
unsigned char demo;

void update_power(ctx_t *power_ctx)
{
    power = power_ctx->value;
}

void update_demo(ctx_t *demo_ctx)
{
    if(demo_ctx->value == 1 && power > 30)
        demo = 1;
    else if(demo_ctx->value == 2 && power > 30)
        demo = 2;
}

void* context_handler(void *arg)
{
    if((ctx_t *)arg)->type == TYPE_POWER)
        update_power(arg);
    else if((ctx_t *)arg)->type == TYPE_DEMO)
        update_demo(arg);
    return NULL;
}

```

---

(a) application

(b) middleware

Figure 2.5: The [115] case study on TourApp, a museum guide application translated into C using the `pthread` library

The process of identifying (ii) statements, only for the case in which the leaked-upon variable influencing the statement is strictly a global, is related to the problem of identifying data racing points. However, given the fact that the external contextual input starts always as a variable local to a function (be it an event handler or a thread's main function), all other program points can only be identified by more sophisticated techniques related to information flow.

Given a set of determined context-aware points, as future (and concluding) work for this contribution, we are looking into (i) writing specifications for context-aware applications, and (ii) automatically generating assertions from these specifications, to be verified by existing generic counterexample-based abstraction and refinement procedure.



Part II  
Papers



# Chapter 3

## Resource Discovery in Activity-Based Sensor Networks

The material presented in this chapter is the most recent, journal version of the paper *Resource Discovery in Activity-Based Sensor Networks* [14]. Published earlier as a conference version [13], only the introductory section differs significantly between the two.

[14] Doina Bucur and Jakob E. Bardram. Resource Discovery in Activity-Based Sensor Networks. In *Mobile Networks and Applications (MONET)*, volume 12, numbers 2–3, pages 129–142. Springer Verlag, June 2007.

[13] Doina Bucur and Jakob E. Bardram. Resource Discovery in Activity-Based Sensor Networks. In *Proceedings of the First International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–10. Nov 2006.



# Resource Discovery in Activity-Based Sensor Networks

Doina Bucur

Jakob E. Bardram

## Abstract

This paper proposes a service discovery protocol for sensor networks that is specifically tailored for human-centered pervasive environments and scales well to large sensor networks, such as those deployed for medical care in major incidents and hospitals. It uses the high-level concept of *computational activities* (logical bundles of data and resources) to give sensors in *Activity-Based Sensor Networks* (ABSNs) knowledge about their usage even at the network layer. ABSN re-designs classical service discovery protocols to include a logical structuring of the network for a more applicable discovery scheme. Noting that in practical settings activity-based sensor patches are localized, ABSN designs a fully distributed, hybrid discovery protocol based on *Extended Zone Routing Protocol* (EZRP), proactive in a neighbourhood zone and reactive outside, so that any query among the sensors of one activity is routed through the network with minimum overhead, guided by the bounds of that activity. Compared to EZRP, ABSN lowers the network overhead of the discovery process, while keeping discovery latency close to optimal.

## 3.1 Introduction

Wireless ad hoc sensor networks for medical purposes are playing an increasing role within healthcare. Body Sensor Networks (BSN) are being designed for prophylactic and follow-up monitoring of patients in e.g. their homes, during hospitalization, and in emergencies. For example, a wide range of medical sensor networks has been proposed for post-operative monitoring [3], heart monitoring [48], and follow-up monitoring of e.g. strokes [56]. In the European PalCom project [107] and in the Code Blue project in Boston [37], medical sensors are being developed for patient monitoring in emergencies.

With the arrival of this range of medical sensors, efforts go towards prototyping sensor networks to aid medical care on a large scale, such as patient care in hospitals or that in the emergency medical service field of pre-hospital work in major incidents (e.g. part of the PalCom project [107]).

In the latter scenario, sensors help automate the victims' identification, registration of data, medical assessment and reassessment, triage and overview; if performed entirely by the medical practitioners, especially in the major incident case, such tasks are hindered by the sheer number of victims.

Core research questions in such sizable wireless ad-hoc sensor networks include low-level data routing and service discovery protocols, i.e. the most efficient way – in terms of response time, network bandwidth and power consumption – to route data in the network and to discover and access services within the network. Extensive research within efficient data routing in wireless ad-hoc networks has been done already; however, limited research has been done within the field of resource discovery in such networks, and a majority of these designs service discovery protocols for a heavyweight IP-based network infrastructure. Furthermore, most of this work does not take into account any knowledge about the usage of the network, and hence stays completely general and detached from the application domain.

We propose to use the concept of *Activity-Based Computing* [5] for data routing and service discovery in wireless ad hoc sensor networks. The core idea in activity-based computing is to help users organize computational services, data and resources in logical bundles that match their work activity. We call these bundles *computational activities*, or simply *activities*. For example, a healthcare activity would be the monitoring of a patient, which can take the concrete form of the prophylactic monitoring of Mr. Hansen for congestive heart failure by monitoring a combination of parameters like blood pressure, ECG, weight, and pulse. Similarly, at an incident site, a monitoring activity for each victim could be created by bundling sensors monitoring respiration, pulse, oxygen saturation, temperature, and blood pressure. On a larger scale, at a major incident site, a care activity would group the body sensors of all patients classified to be in critical condition after triage.

The core idea in *Activity-Based Sensor Networks* (ABSN) is to utilize the clustering of sensors into logical activities, and then base data routing and service discovery on each sensor's knowledge of activity membership. Our hypothesis is that data routing and service lookup are more efficient in this protocol design than in a general uniform scheme, since services and their data are mostly relevant within the bounds of each patient activity.

### 3.1.1 ABSN Requirements and Solutions

The trademark usage cases for ABSN are the major incident and hospital care settings; ABSN answers the critical challenges put forward when designing emergency medical service systems for major incidents (after sociological studies by Kristensen [71] under the Palcom project [107]):

- The process of victim *identification* and *registration of data* requires

persistent textual information attached to each patient. During major incidents, these accident cards – which are the tools for the identification of the person, registering of injuries, symptoms and the progress of care on-site – cannot be filled in for lack of time, which leads to administrative overhead and a risk of errors during care. The larger the incident, the less complete data registration tends to be.

- The *categorization* of victims is done during triage according to the patients' severity of injuries, and consists of marking the victim with a coloured card. The status of the victim can easily change and thus needs to be monitored, and more, classification of a victim depends not only on his condition, but also of the other victims. In major incidents, the large number of victims greatly hinders classification efforts.
- Particularly in major incidents, *communication* about the status of victims is mostly verbal, resulting in a lack of crucial information being reported about the patient when the patient is handed in from one care team to another, and over time. Given the lack of recorded textual information, the victims themselves are the objects upon which the care work is organized, requiring multiple assessments from different practitioners. A situational overview of the incident site is also impossible to achieve.
- Technology needs to *scale down* from major incidents to minor ones, rather than the other way around; design should take place at the more complex end of the spectrum, to insure a functional standard and proper familiarity with new technologies in anticipation of the major incidents.

To automate victim identification and data registration, ABSN deploys sensors as patient body monitors, dynamically and explicitly activity-grouped to patients by nurses. An activity ID on each node groups sensors in activity clusters. While patients are moved to ambulances or other locations in the hospital, their body sensors record medical data into memory, and use the knowledge about belonging to a certain activity in order to efficiently aggregate the data history for individual patients and make simple diagnoses. Having patient body sensors deployed in the field, victim categorization and a situational overview can also be automated. Also, ABSN designs for the complex end of the spectrum, having scalability as important a requirement as functionality.

The trademark scenarios that ABSN expects to cover are thus the typical cases encountered in incident and hospital settings:

- The sensors often need to be deployed in dense, relatively localized and connected patches following the location of a patient, contextual

object or group of patients, either in a hospital or at the site of an incident. Because of this, there is a logical structuring of the sensors based on the activity they are a part of; we call this logical grouping an *activity cluster*, AC. Depending on their size and topology, ACs can be singlehop and strictly localized (if they only group body sensors) or multihop and occupying a large area (if they group entire categories of patients).

- Interaction among sensors is, more often than not, bounded inside an activity cluster, but network-wide discovery and data exchange are of no less importance.
- There is a high degree of mobility involving entire activity clusters at a time and sensor unavailability is often a problem (in the example above, sensors might fall off patients while the patients are being moved, and the protocol is required to signal this change to the application layer).

The ABSN discovery protocol allows low-latency data aggregation within the bounds of a patient’s sensor bundle, by locally caching on sensors the neighbouring of same-activity routes and services. For the same purpose, it uses the knowledge about the patient sensor bundles being relatively connected, in order to limit the packet broadcast overhead in the network. This is different from the majority of the ad hoc protocols in the literature – which do not differentiate between nodes in the network – in the fact that ABSN allows the application-level logic to be used at the networking layer, in order to make the network protocols more applicable and efficient in practical scenarios. Also, an ABSN is an ad hoc network that has no reliance on any infrastructure; this approach allows for the seamless adaptation of an ABSN network to work both at an infrastructured site (such as a hospital) and in an ad hoc setting (such as the site of an incident).

To summarize our contribution, ABSN designs a completely distributed discovery scheme, that relies on no directories, but on limited, individual caching of routes and service information on nodes. It employs proactive route and resource discovery within an activity cluster and reactive discovery outside, ensuring low latency for all intra-AC communication.

As is the case in sensor networks, separate software layers for routing and resource discovery are redundant (as argued by [69,114]). This has led us to follow EZRP [114] in embedding routing and discovery into one protocol. We state that it is the binding of these low-level protocols to high-level human activity concepts that makes sensor networks protocols more applicable in pervasive healthcare environments than generic routing or service discovery protocols.

Our main contribution is thus the porting of ABC concepts [5] into pervasive sensor networks for the purpose of redesigning classical networking protocols to make sensors fit for human-centered pervasive settings, with a focus on large-scale networks for use in hospitals and major incidents. Also, to the best of our knowledge, this is the first attempt to practically employ such low-resourced embedded systems like sensors in the field of service discovery in ad hoc environments.

We implemented ABSN on Moteiv's Tmote Sky, TelosB sensors, and tested the protocol on a real-code simulator.

The rest of this paper is organized as follows: section 3.2 gives an overview of other service discovery protocols for ad hoc and transient networks, section 3.3 gives a detailed description of ABSN, while section 3.4 makes an analysis of the performance parameters of discovery in ABSN. At last, section 3.5 gives the evaluation results and section 3.6 summarizes and concludes.

## 3.2 Related Work

In general, for any networked environment, device mobility implies losing connectivity with configured environments. *Service discovery* software then enables a mobile user to take advantage of resources at any new location. For pervasive environments, in which autonomous computing devices interact to achieve intelligence, service discovery protocols permit the adaptation of devices to network composition and context.

“Classical” service discovery protocols for pervasive environments - like Jini, UPnP and others - are tailored for resource-rich, stable-network enterprise-like environments, and as such are not always applicable in pervasive computing, yet they do provide basic protocol design reference points. They rely on stable, LAN-like network connections, do not need to solve network-layer routing issues (since they use IP), are usually centralized - using one or more directories spanning the network - and can employ sophisticated service semantics.

Unlike LAN devices, sensors form unstable ad hoc network connections and are used for extremely mobile applications. Such Mobile Ad hoc Networks (MANETs) are autonomous systems of intermittent, energy-bounded nodes collaborating in the absence of any centralized support. The network diameter is large compared to a node's range, any data transport is multi-hop, and the nodes do not have a priori knowledge of the topology of the network. Because of the network diameter-to-wireless range ratio, routing and power efficiency are central issues, and each node has to act as a router.

Thus, when trying to port “classical” service discovery protocols to MANETs, the problems arising are that in MANETs mobility and resource

limitations disallow static directories, the underlying network is not stable enough to allow centralized, registration-oriented protocols, and the protocols cannot be heavyweight, with respect to bandwidth and power usage.

Due to the large size of a sensor network, the discovery scheme becomes a central issue. *Proactive* discovery schemes maintain routes within the network continuously refreshed – so that a query is answered without latency; they also constantly add overhead on network bandwidth and on a node’s cache memory. *Reactive* protocols do not determine routes before an explicit query, and require a global flood search procedure with long delays and heavyweight traffic at each query. While reactive schemes do not achieve a large degree of performance, proactive schemes – otherwise optimal in performance – do not scale well in large networks; hybrid schemes are thus designed to overcome these limitations.

### 3.2.1 Service Discovery in Ad Hoc Environments

A review of service discovery protocols for transient environments is to be found in Table 3.1, and only the most relevant ones are discussed below. Most of these protocols are intended for resource-rich ad hoc wireless networks, and the others are general proposals, detached from the application domain; some achieve novel designs by allowing application-level policies to dictate discovery schemes.

	Network topology and protocol	Storage of service info	Discovery policy	Software layer	Implementation and/or testing
DEAPspace (2001) [83]	Singlehop, short range 802.11-like transient network	Fully decentralized: all nodes cache all services in network	Proactive: each node adverts all services in network (gossip-like)	-	Simulation
Allia (2002) [93]	IP network over Bluetooth	Fully decentralized: each node might cache the services in vicinity based on policy	Proactive: a node advertises its services; reactive for remote services	IP middle-ware	Applied, mobile commerce on laptops and iPAQs, over Bluetooth
GSD (2002) [23]	As [93]	As [93]	As [93] w/o alliances	-	Simulation (Glo-mosim)
Cheng's (2002) [26]	Multicast IP network	None, discovery is on-demand; only caching of already discovered services	Mainly reactive, with a variant of proactive updated services adverts	IP routing, piggybacked on ODMRP	-
Konark (2003) [51]	Multicast IP network	Fully decentralized: all nodes in network cache all services	Both proactive and reactive	IP middle-ware	Applied, mobile commerce, on iPAQs and phones
SANDMAN (2004) [100]	Any, only a model	Decentralized if nodes awake; centralized, with cluster heads, if nodes asleep	Reactive: nodes form clusters with dynamically elected heads	-	Simulation (ns-2)
Sailhan's (2005) [99]	IP network	Centralized, distributed: dynamically elected directories form backbone	Proactive among directories inside a node's zone, reactive outside	-	Simulation
EZRP (2005) [114]	Multihop, sensor-like network	Fully decentralized: all nodes cache services in their zone	Proactive inside a node's zone, reactive outside	Routing, embedded in ZRP	Simulation (Qual-net)
CARD (2005) [52]	Multihop, sensor-like network	Fully decentralized: all nodes cache services in vicinity	Proactive inside a node's zone, reactive outside	Routing (not a routing protocol)	Simulation (ns-2)

Table 3.1: A review of service discovery protocols in ad hoc networks

*Allia* [93] has participants advertise own services in their vicinity; the advertisements are forwarded within a certain hop diameter and the devices which hear these might passively cache them. The set of devices that a certain node chooses to cache services of forms this node's *alliance*; a request will go to other alliances only if a service is not found in the cache, and a node only knows the participants of its own alliance. The intelligence of the protocol lies in its set of policies (based on application preferences) that rule the frequency and caching of advertisements, and the diameter of the alliance. While still being heavyweight, *Allia* was a path to follow in our design, both because it recognizes the optimal hybrid proactive/reactive approach for discovery, and because it employs application-level policies over the discovery protocol.

Among the lightweight protocols, *GSD* [23] adapts the sophistication of [93] to use in actual MANETs, by replacing policy-based alliances with broadcast vicinities of a certain diameter, so that the network is now uniform, a situation which is not common in networks for medical care. *CARD* [52] keeps with the hybrid approach of proactive local and reactive remote discovery, but is best-effort and only valid as a routing protocol for short flows of data, trading off the optimality of the shortest path method for heavy savings in certain settings. It employs contact nodes instead of ZRP's [49] bordercast, and concentrates on the effort of electing and maintaining the contact nodes for global discovery. The network uniformity in [23] and the limited validity of [52] distances both protocols from our approach: ABSN is intended to make efficient use of a logical structuring of the network, and to be a general service discovery protocol for human-centric pervasive environments.

*EZRP* [114] recognizes that, in an ad hoc network, service availability is tightly linked to route determination to that service: finding the address of a service is redundantly followed by finding a route to that address. Hence, the solution piggybacks service information in routing packets (an idea first found in [69]). It extends ZRP [49] to include service information and has been the basis ABSN was built on, as detailed in section 3.3.

### 3.3 ABSN Design

This section gives a detailed description of both Zone Routing Protocol (ZRP, [49]) and Extended Zone Routing Protocol (EZRP, [114]), to then discuss the design of ABSN.

#### 3.3.1 ZRP and EZRP

ZRP is a MANET routing protocol whose guidelines fit a sensor network's setting by being flat, fully distributed and only limitedly proactive. It

chooses a hybrid, proactive and reactive approach and delimits a zone of a certain number of hops around each node (so that, overall, the zones are heavily overlapped), to then limit the proactive procedure to this zone. For out-of-zone discovery, instead of plainly broadcasting of the query throughout the network, the *bordercast* message distribution scheme directs queries from a source node towards the edges of the network by only forwarding the query to the nodes on the border of the source node's zone. It is important to note that, despite grouping the nodes into zones, ZRP is not a hierarchical protocol, but remains a flat one, since each node has a zone, so that the grain size of this zoning is one node.

The ZRP and its subprotocols' IETF drafts do not impose specific protocols for the proactive and reactive discovery of routes. We choose to adapt these drafts for use in resource-poor sensor networks, as described in subsection 3.3.2.

EZRP extends ZRP for use in service discovery, simply by adding a service ID to the hello messages used by neighbours to let them know each other. This way, neighbouring nodes find the others' service IDs, together with their simple presence. Furthermore, nodes periodically broadcast their set of neighbours and their service IDs throughout their zones, so that intra-zone routes are extended with service information.

A view over an EZRP network is given in figure 3.1. We will denote the ZRP zones by the term *network cluster*, NC, to differentiate them from the activity clusters, ACs. In figure 3.1, if a link-state protocol is used intra-NC (for example, a simplified OSPF), node A will receive hello packets from its direct neighbours (also containing a service ID field) and will periodically transmit advertisements throughout the NC, announcing its list of neighbours and their services. When node A's NC converges (all nodes have a consistent view upon the network, from a routing point of view), A will have in its local cache a complete view over its NC, route- and service-wise. In order for A to discover services and routes for nodes D or X, bordercast is employed.

### 3.3.2 ABSN Discovery Design

As stated in section 3.1, the reflection of high-level activities into the sensor network is the logical grouping of the sensors into *activity clusters*, ACs, which are deployed in multihop, overlapping patches throughout the network. Since data communication is more often than not bound inside an AC, we call for a proactive discovery scheme for intra-AC routes and services and a reactive discovery scheme inter-AC.

Although this might immediately recall EZRP (subsection 3.3.1) as a solution, there is no possibility of identifying ACs with the network clusters in ZRP in a one-to-one fashion. In ZRP, NCs are sets of nodes reachable

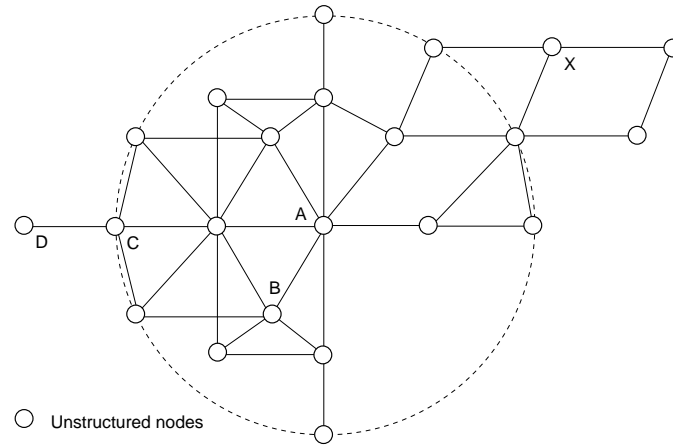


Figure 3.1: A view over an EZRP discovery network: the nodes are not logically grouped and each node keeps a zone (NC) of a 2-hop radius (for this example) centered at itself. The NC for node A is drawn.

within a certain radius (number of hops) from any central node and are flatly distributed across the network (for every node in the network there is a NC, and nodes are logically homogeneous). On the other hand, ACs are unique sets of nodes (for example, there is only one AC with patient Hansen’s ID) and are deployed in irregular, possibly overlapping and dense geometric patterns around the network.

A view over a basic activity-based network is given in figure 3.2. Activity clusters are deployed in a relatively localized, connected fashion throughout the network; they might overlap in the same area, so that any node might have in its range other nodes belonging to activities different than its own.

The main requirements imposed over the ABSN design are protocol scalability in the large networks required in major incidents and low latency at intra-AC discovery: service and route discovery is expected to work throughout the network, yet optimized in such a way that discovery latency is low if the requestor and the service belong to the same activity.

ABSN superimposes the logical AC grouping over the flat EZRP network topology, as in figure 3.3. To achieve scalability, every node in the network still keeps a NC zone centered at itself, in order to limit the proactiveness of the protocol. Since the nodes within this NC belong to different ACs, and since the logical grouping of sensors in ACs implies a lower probability that a service be requested from a node of a different colour, ABSN has a node only cache the service information of same-AC nodes that lie within this node’s NC. Furthermore, a node will proactively cache routing information for all the nodes within its NC.

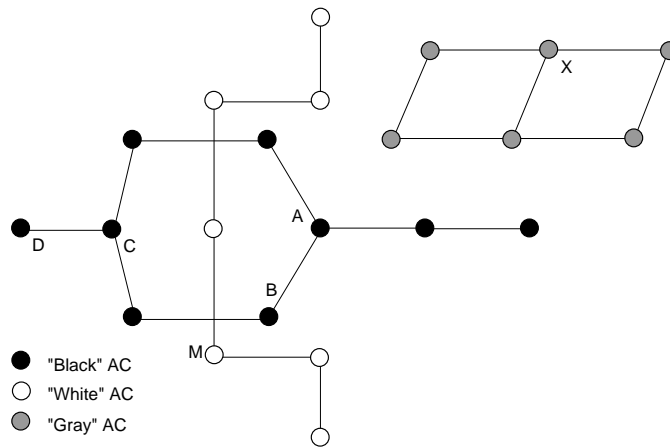


Figure 3.2: The logical view over an activity-based network: each activity is colour-coded and the intra-AC connections are drawn; connectivity also exists among neighbouring nodes of the different, overlapping ACs (not drawn in the figure).

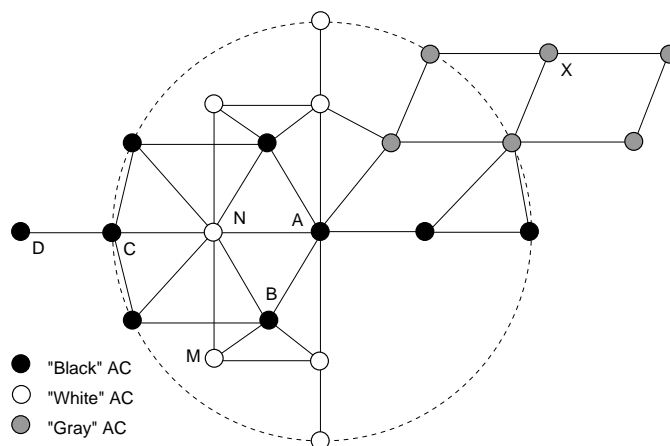


Figure 3.3: A view over a sample ABSN network: in the EZRP fashion, every node, regardless of its AC, keeps a zone (NC) of a 2-hop radius (for this example) centered at itself. The NC for node A is drawn, and it gathers nodes from different ACs, including node A's own AC.

### 3.3.3 Intra-NC Discovery

A lightweight link-state proactive protocol is employed intra-NC. This protocol sends periodic one-hop broadcast hello packets from each node, so that – at all times, and with a maximum latency equal to the hello period – every node knows the addresses of its neighbours and, as a metric, the link quality to each of them. Also, a node that detects a drastic change in the metric to at least one of its direct neighbours (including a neighbour’s arrival or disappearance) triggers a limited-range *link-state advertisement* (LSA), announcing the change within the radius of its own NC. LSAs are forwarded in a broadcast manner for only a number of hops equal to the radius of the NC. This way, every node keeps a *routing map* of the NC as a weighted graph of nodes that can be reached in a number of hops equal to or less than the NC diameter.

In order to allow nodes to build a *service map* of the near AC, both hello and LSA packets carry, besides addresses, AC and metric information, the service IDs of the nodes these packets advertise: they will both state the service ID of the source of the packet, and – in addition – a LSA will list the neighbours which changed state and, if available, their service information.

At a “black” node (see figure 3.3), only such incoming packets bearing service IDs for other “black” nodes have their service information cached. In figure 3.3, this leads to the “black” node A having a service table with the service information of all the “black” nodes in its NC: a hello packet originated at node B advertised the service ID of B, and a LSA triggered by node C (and forwarded over one “white” hop) advertised the service ID of C. To save computation time and memory, no service information is cached about nodes of other colours.

### 3.3.4 Route and Service Query Solving

With a routing map and a service map in place at each node, the solving of a service query depends on the parameters of the query. A service of the same colour might be needed, as in the case of the aggregation of data from sensors belonging to the same patient, for automatic diagnosis of the patient. If the gateways to the wired network form an activity cluster with a well-known colour, then the downloading of sensor data from any patient sensor to the wired network will have to be preceded by the (different-AC) discovery of the gateway location.

Given these two cases, if a certain service of the same colour is needed, then

- the local service table is tried; if there is a match, the address is returned;

- if there is no match, the query is bordercast only to the border nodes (or, if none exists, the closest nodes to the border) of the same colour, exploiting the fact that ACs are relatively connected, and thus limiting the network overhead.

If, on the other hand, a service of another colour is needed, then

- the local routing table is checked for the closest node of the searched colour; if such a node (which we call a *gateway* for the entire searched AC) is found, the query is relayed to it, and it will proceed as in the same-colour case above; if more than one gateway is found, ABSN chooses the closest gateway to the query source node;
- if no gateway exists, the query for the gateway is bordercast.

A routing query will be solved exactly as in ZRP, regardless of activity IDs. Routes for all the nodes that can be reached within a number of hops equal to the NC diameter are read from the proactively built network map. Routes for any other nodes are discovered on-demand, through bordercasting.

This design makes discovery either proactive or reactive, depending on whether the unknown is a route or a service, and on the relation between the AC of the requestor and that of the destination. Thus, route discovery is always proactive within the limits of a NC, and reactive outside, regardless of the relation between the colours of the nodes involved. On the other hand, service discovery for same-colour nodes sharing a NC is fully proactive; in all other colour and distance conditions, service discovery is always a mix of proactiveness and reactivity: the query is initially forwarded for certain other nodes to solve, but will eventually reach a node that has discovered the required service information in advance.

### 3.4 Discoverability, Optimality and Overhead Analysis

This section analyzes the performance of ABSN compared to the relevant related work protocols. It defines and uses parameters such as the *discoverability* of that service, the *optimality* and latency of the solving of a query and the network overhead added by a query to assess ABSN's characteristics. Furthermore, this section analyzes the set of network topologies that ABSN serves better than other protocols.

The ability of ABSN to discover any service (including a route), if present in the network, is hence called the *discoverability* of that service. ABSN guarantees discoverability in all network settings, provided that any activity

cluster is only disconnected by a number of hops less than the NC radius; this is like [114] and [23], and unlike [52].

We denote by the *optimality* of a query that query taking the shortest, lowest-latency route in the network. The optimality of ABSN depends on the NC radius and on the network topology of the searched AC. While having low discovery latency in a large number of network and activity topologies, ABSN only has suboptimal latency in few badly-formed activity cluster topologies.

As a routing protocol, ABSN is optimal and guarantees route discoverability, in the fashion of ZRP. The ABSN routing and discovery design following logical activity grouping greatly limits the network and computational overhead that a general-purpose EZRP would imply, if deployed in the pervasive environments ABSN is applicable to.

These statements will be substantiated by the discussion over local and global discovery in subsections 3.4.1 and 3.4.2.

### 3.4.1 Neighbourhood Discovery

In regard to route discovery within a node's NC, it is to note that the proactiveness of the link-state protocol that ABSN uses ensures that changes in the network or in the service provision are propagated in a timely fashion, with a maximum latency directly proportional to the NC radius. Since link-state protocols keep a dynamically updated view over the entire NC and are immune to routing loops, optimality and route discoverability at intra-NC route discovery are ensured. On the other hand, link-state protocols broadcast any LSA throughout the NC. This implies that the radius of each node's NC must be dynamically updated given the node density in the area, in order to keep network and computation overhead down.

In regard to service discovery, any node will announce throughout the NC any change in the states of its direct neighbours by only mentioning known service information: for example, in figure 3.3, when “black” node C enters the network, “white” node N will not cache C's service information provided by C's hellos. Thus, a LSA originated from N and announcing the new “black” node will not provide to A the service information that A could use (since A and C are of the same colour). We call the situation in which the forwarding of service information towards a node is stopped by interposing nodes of another colour by the term “colour shadowing”.

However, within a NC, ABSN is resilient to such apparent “colour shadowing”: even if the “black” AC weren't connected, the new node C will also trigger a LSA, announcing its new link to node N. Since LSAs are forwarded regardless of the colour of their source and contain the complete service information of their source node, node A will receive a first-hand update about the services provided by C from C itself. In general, within an

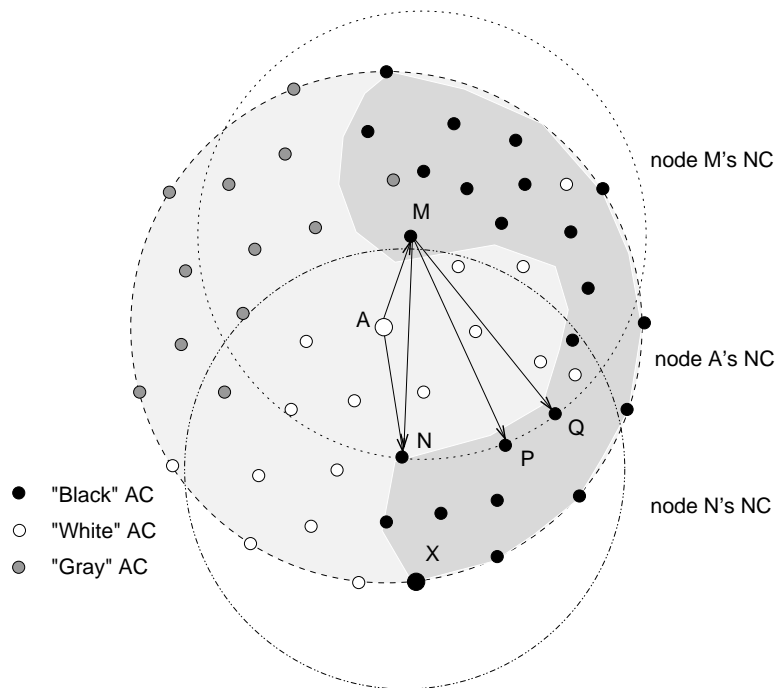


Figure 3.4: For “white” node A to discover the services of “black” node X, A needs to choose a gateway for the “black” AC. The ABSN solution is to choose the best metric (closest) “black” node as a gateway (in this figure’s case, node M). Yet, for M to discover X it needs to do one bordercast step (forwarding the query to N, P and Q). An optimal choice for a “black” gateway would be node N, which can immediately answer it without latency, since X lies in N’s NC.

NC, every node will have a complete image of the same-colour services in the neighbourhood. This resilience is even independent of AC connectivity. Furthermore, still in the example in figure 3.3, if the “black” AC is connected, LSAs with the service information of the new node will redundantly reach node A on all fully “black” paths from C to A. This way, discoverability of same-colour services within an NC is guaranteed and is optimal - this adds to the optimality of the routing of packets intra-NC.

In the case of different-colour service discovery, on the other hand, while it is ensured that a service will be discovered if present (the service discoverability is also fully guaranteed), the optimality of the query depends on the choice of gateway for the searched AC. Yet, in the worst-case scenario in which a gateway is blindly chosen in the exactly opposite direction from the actual service searched, a maximum of 1 bordercast step will have the query solved (as shown in the example in figure 3.4).

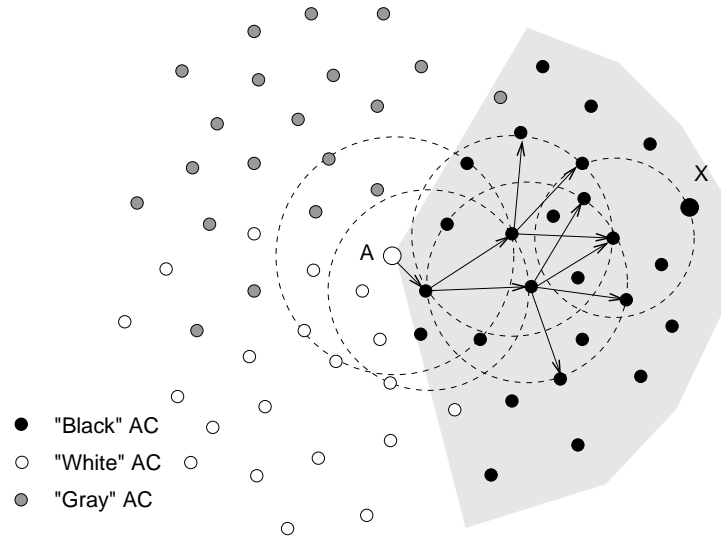


Figure 3.5: For node A to discover the services of “black” X, a service query will be bordercast repeatedly. Unlike EZRP, which would bordercast the query symmetrically around A, ABSN limits the traffic to the bounds of the “black” AC.

### 3.4.2 Global Discovery

At a global level, route discoverability and optimality are secured, by the design of ZRP. An average maximum number of  $\frac{D}{R}$  bordercast steps are employed for the route discovery towards a node which is  $D$  hops away, if the average NC radius of the nodes on the path is  $R$ .

ABSN greatly reduces network overhead by only forwarding service queries for a “black” service within the bounds of the “black” AC. In the case of symmetrical, radially deployed ACs (as in figure 3.5), the network overhead is reduced to a percentage of  $\frac{\text{black network area}}{\text{total network area}}$  of the network overhead in EZRP (here, we denote by the *area* of an AC or a network a qualitative measure that is proportional to the number of nodes in that AC, the nodes’ degree and to the number of hops this AC occupies).

Service discoverability in ABSN relies on the activity clusters only being disconnected by a number of hops less than the NC radius. Given this, ABSN guarantees the discoverability and optimality of a different-AC gateway node search, but only the discoverability of a service is guaranteed: the optimality of the path the query takes towards the service depends on the topology of the searched AC: at a global level, “colour shadowing” is in effect, routing a query according to AC topology. For example, in figure 3.6, a

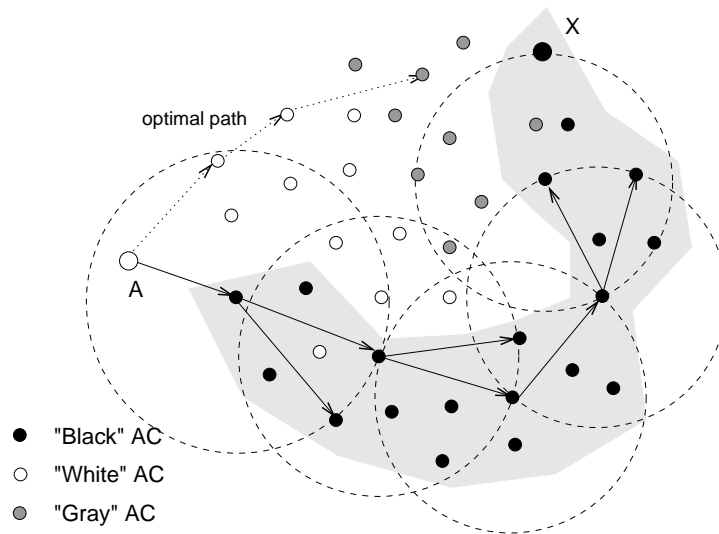


Figure 3.6: The bordercasting of the “black” query will follow the deployed shape of the “black” AC, instead of taking an optimal path across other ACs.

service query takes a sub-optimal path through the overall network. ABSN argues that this is a small price to pay for the reduction in network traffic. It is to note that this sub-optimality is only true for service requests: service replies are treated as regular data packets and, since routing is optimal, the replies take the optimal path back.

Furthermore, as shown in subsection 3.4.1, within the limits of a NC, “colour shadowing” does not limit discoverability. This indicates that global “shadows” or disconnections less than the NC radius in width will be overcome by the protocol. It follows that the preferred AC topologies only exclude disconnections wider than the NC radius.

## 3.5 Evaluation

This section gives an overview of the means of evaluating ABSN. A few typical, trademark real-world ABSN application cases are characterized in some detail; real-code simulated tests are performed to test the increase in optimality and scalability of ABSN versus EZRP in a number of basic scenarios, and finally discussions are had upon the degree to which the test scenarios estimate the compared performance of ABSN in real-world deployments.

As stated in subsection 3.1.1, the trademark usage cases of ABSN are encountered in major incident and hospital settings. To what concerns the

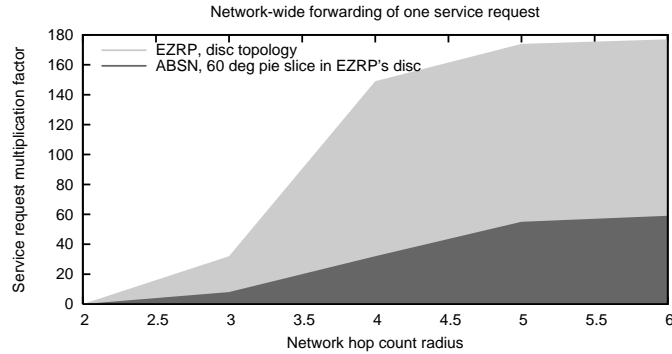


Figure 3.7: Additional network traffic due to the multiplication of a single service query, in number of packets by the radius of the overall network. The network is a 3-degree extended star with varying radius; all nodes keep NCs of 2-hop radius; the query is intra-AC: it starts at the central node and is destined to a same-colour border node. The searched AC is a non-overlapping, 60-degree pie slice in the network.

sensor population, in both settings – when designing at the complex end of the spectrum – it is expected a number of patients upper bounded by a number on the order of a few hundreds, each patient having potentially associated up to five sensors. Furthermore, in a hospital setting, a set of contextual sensors monitor hospital objects (like medicine trays, doors and beds). Due to the size of such a network, in the following, tests are performed upon the improvement in network overhead in an ABSN network compared to EZRP, when one service query is multiplied throughout the network in search of the destination node. Figures 3.7 and 3.8 show the compared network traffic by EZRP and by ABSN, in the same network.

The results in figure 3.7 fit a scenario in which a major incident triage field is divided into areas in which patients have been moved according to their status; roughly, a radial third of the area is reserved to critical patients, which make up one activity cluster for the purpose of getting overviews of the area and of estimating the worst cases at the site. Thus, for this test two ACs occupy the network: the “critical” AC (including the source and the destination of the query) occupies a non-overlapping 60-degree pie slice in the overall network. The results show that ABSN uses only a fraction (approximately equal to the pie slice fraction) of the traffic generated by EZRP in order to find the service, and hence scales to large networks better than EZRP in a direct proportion to the size of the ACs relative to the overall network.

Figure 3.8 fits the scenario in which an improvised sensor infrastructure is deployed at the triage site in a radial, linear topology. In this extreme

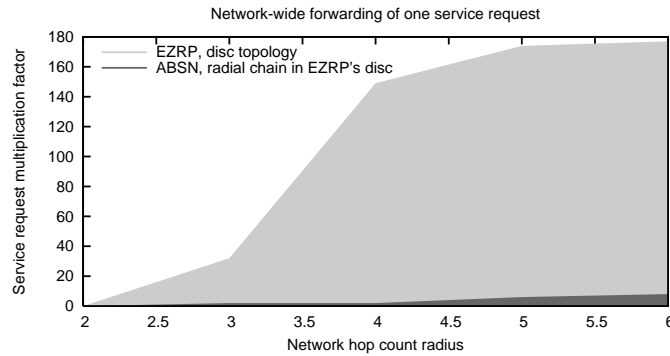


Figure 3.8: Additional network traffic due to the multiplication of a single service query, in number of packets by the radius of the overall network. The network is made up by 3-degree nodes and is an extended star with varying radius; all nodes keep NCs of 2-hop radius; the query is intra-AC: it starts at the central node and is destined to a same-colour border node linked by a chain AC.

case, only a number of packets that is linear to the distance from source to service is forwarded on the ABSN network, compared to EZRP's storm of LSAs.

In both above cases, discovery latency is equal for both EZRP and ABSN, yet ABSN shows a great improvement in network bandwidth usage, by guiding the query along the path of an AC, instead of unconditional bordercasting.

The above tested network scenarios do not make for a complete usage test upon efficiency in traffic overhead in all topologies and AC scenarios, yet they allow for extrapolating the overhead in other topologies, by composing the results above. For example, the more general case of queries originated in one AC and destined to another has a network usage efficiency that is a combination of the performance of EZRP and intra-AC ABSN: the discovery of a gateway for the searched AC performs like EZRP, while the discovery of the service starting at the already found gateway performs like the tests above.

One other important test relates to the scalability of the protocol; as stated above, the size of an ABSN network, when designing for the most complex major incident scenario, reaches hundreds of nodes; also, as stated in subsection 3.4.1, since ABSN is a flat protocol, the main measure of scalability is the node size of the network clusters, and not the overall network size or the AC size. While a large network cluster would improve response times by adding more proactiveness to the overall network, a limit over the NC size is imposed by resource bounds on the nodes and by limited network

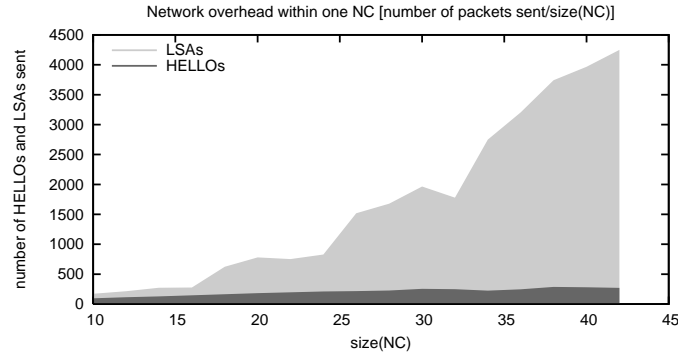


Figure 3.9: Network traffic, in number of packets by number of nodes composing the NC. The hello packets have been sent out within the bounds of one NC by the link-state proactive intra-NC protocol during a period of time equal to 10 times the hello period. The LSA packets have been sent in the initial process of setting up the same network. The NC radius is constant at 2 hops, while the NC size varies. The degree of all nodes is proportional to the NC size. 3 ACs overlap in this NC, and, at every network change, each node sensing the change triggers, on average,  $\frac{2}{3}$  of a LSA packet.

bandwidth.

To test the scalability of the protocol against the NC size, we take a scenario in which at a major incident site patients' conditions have been assessed by practitioners on the spot; the patients have been associated to three activity clusters depending on the severity of their status, but have not yet been moved into separate triage areas. The three ACs thus spread uniformly over the network, and overlap in all NCs.

For this scenario, figure 3.9 shows the network traffic generated by the intra-NC protocol: the figure shows both the maintenance traffic (hello packets are sent by each node even in the absence of change) and network boot traffic (LSA packets are sent at every network change). The large number of LSA packets is due to the process of building up the network, one node at a time; this amount of LSA traffic will only be repeated during normal running of the network in the extraordinary event in which node mobility and network interruptions will make so that no two nodes will continuously find themselves in the same NC they started in (a phenomenon which could be called “complete entropy” of the network).

The number of hello packets that are sent on the network only varies slightly below 10 times the size of the NC, while the number of LSAs varies significantly with the NC size. This LSA overhead variation is recognizable as being proportional to the square of the NC size, which follows the theory: the forwarding of LSAs (a constant number of them originate from each

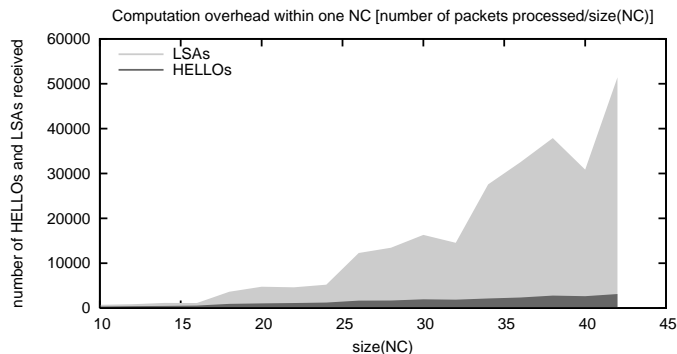


Figure 3.10: Number of receive packet events, summed up for all the nodes in the NC. The hello packets have been sent out within the bounds of one NC by the link-state proactive intra-NC protocol during a period of time equal to 10 times the hello period. The LSA packets have been sent in the initial process of setting up the same network.

node) would trigger their multiplication by a factor proportional to  $N * E$ , where  $N$  is the number of nodes and  $E$  is the number of edges in the network graph (equal to  $\frac{\rho * N}{2}$ , where  $\rho$  is the average degree of the nodes). This gives a multiplication factor proportional to  $N^2$ . Any variation in the test parameters would only change the multiplication factor of forwarded LSAs by a constant.

In the same scenario, figure 3.10 shows the total computational overhead added to the nodes in the network because of the network traffic in figure 3.9, summed up for all the nodes in the network. Only those received LSAs that announce a change to a node for the first time will actually be processed (a number equal to the number of unique LSAs triggered on the network, as in figure 3.9), all the others being duplicates.

ABS<sub>N</sub> was implemented on Moteiv’s IEEE 802.15.4-compatible *Tmote skies*, as a set of TinyOS [106] NesC [43] components. NesC and TinyOS (the de facto standard for programming low-resource sensor networks) were used in order to ground ABS<sub>N</sub> in practice. For simulation results that would be a correct reflection of the running of the code on sensors, we chose a *real-code* simulator composed of two pieces of software: OMNeT++ [105], a general networking discrete-event simulation environment, and NesCT [65], a language translator from the embedded sensor language NesC into OMNeT++’s C++ classes. NesCT already comes with the translation for OMNeT++ of the basic TinyOS components. Thus, the simulations take into consideration all of TinyOS’s features and limitations.

Another measure of the scalability of the protocol is the size of the data structures needed for each node to act as a service router. An overview of

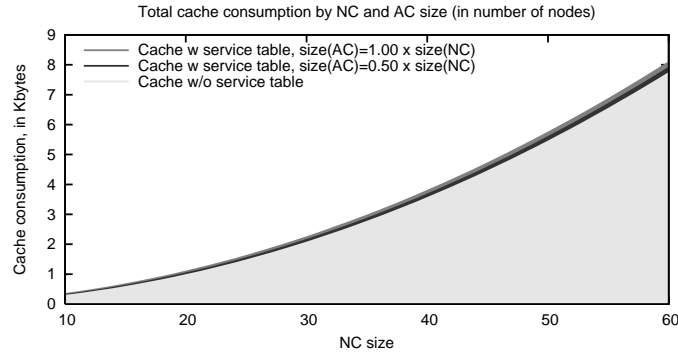


Figure 3.11: An overview over RAM consumption on a sensor node running ABSN. Memory consumption is static in TinyOS (there is no dynamic allocation), so that total occupied size can be found at build time. The graphic leaves out the footprint of the application (which is constant with the NC size) and only considers the size of all data structures involved in routing and service discovery.

the RAM consumption on a TinyOS sensor node is given in figure 3.11. The adding of service discovery capabilities to nodes running a link-state routing protocol is done at an insignificant memory cost. Thus, such proactive service discovery schemes can be a natural, low-cost extension to link-state routing protocols in practice. A *Tmote sky* module with 10kB RAM and 48kB programming flash running ABSN will take 17.750 kB of ROM and will accommodate NCs of as many as 62 nodes, while the sizing down of a NC to a more than sufficient 50 nodes will occupy 7.077 kB of RAM.

Finally, the current primary disadvantage of ABSN is the lack of design for managing activity clusters disconnected by a number of hops greater than the NC size. Devising a solution for this greatly depends on the policies dictated by the usage scenario: in the case of an AC composed of body sensors on a patient, the accidental loss of a sensor would immediately invalidate the data read by that sensor; hence, an energy-consuming scheme for keeping the sensor strictly accounted for is not desirable.

On the other hand, a scenario in which a large AC is accidentally split apart, such as part of a linear sensor infrastructure deployed at an incident site being put out of order, two solutions can be employed. First, the NC size of the nodes composing the AC can be increased until the NC radius becomes greater than the hop size of the gap. Second, an additional *join* mechanism can be designed, following the example of that employed by multicast ad hoc routing protocols for nodes to announce their registration to a multicast group. In the first case, overhead in traffic and resources is only added to the nodes composing that AC; in the second, the added

overhead is network-wide.

### 3.6 Conclusions

To summarize our contribution, ABSN is part of an effort to redesign classical network protocols for a better applicability in pervasive computing: it uses the high-level concept of *computational activities* (as logical bundles of data and resources) to give sensors knowledge about their usage even at the network layer and it makes use of this logical structuring of the network for a more effective service discovery scheme. Noting that in practical settings activity-based sensor patches are localized, ABSN designs a completely distributed, hybrid discovery protocol which is proactive in a neighbourhood zone and reactive outside, tailored so that any query among the sensors of one activity is routed through the network with minimum overhead, guided by the bounds of that activity. ABSN enhances the general-purpose *Extended Zone Routing Protocol* with logical sensor grouping and greatly lowers network overhead during the process of discovery, while keeping discovery latency close to optimal. Future work includes generalizing ABSN for use in networks with arbitrarily disconnected activity clusters.

Furthermore, sensor networks are characterized by a need to carefully integrate functionalities to achieve maximum efficiency (especially with respect to energy consumption). For this, ABSN introduces a close interaction of research from ad hoc networking and human-centered pervasive computing.

Even though our design for ABSN data routing and service discovery emerged from the medical domain, we think that the approach is more generally applicable - there exists a range of other application domains where it is practical to model human activities and use this modeling for data routing and service discovery in a sensor network: fields of application like smart homes or office spaces and ad hoc peer-to-peer connectivity in public places could benefit by the idea of moving some application features into the lower routing layer.



# Chapter 4

## A Calculus for Context Awareness

This chapter presents theoretical work on the matter of modelling and verification for context awareness using process calculi. The major part of the chapter (Sections 4.1 to 4.6) reproduces the recently published paper *Secure Data Flow in a Calculus for Context Awareness* [16] on a calculus for context awareness; Section 4.7 is the paper's Appendix, holding the proofs. The topic is concluded in the last section with a brief overview of a working paper [15] presenting the ad hoc variant of the same calculus.

[16] Doina Bucur and Mogens Nielsen. Secure Data Flow in a Calculus for Context Awareness. In *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 439–456. Springer Verlag, June 2008.

[15] Doina Bucur and Mogens Nielsen. A Calculus for Ad Hoc Context Awareness. Published online at <http://www.daimi.au.dk/~doina>. Working paper. 2008.



# Secure Data Flow in a Calculus for Context Awareness

Doina Bucur      Mogens Nielsen

## Abstract

We present a process calculus based on Mobile Ambients to describe context-aware computing in an infrastructure-based Ubiquitous Computing setting. In our calculus, computing agents can provide and discover contextual information and are owners of security policies. Simple access control to contextual information is not sufficient to insure confidentiality in Global Computing, therefore our security policies regulate agents' rights to the provision and discovery of contextual information over distributed flows of actions. A type system enforcing security policies by a combination of static and dynamic checking of mobile agents is provided, together with its type soundness.

## 4.1 Introduction

The ubiquitous computing systems encourage a constantly changing execution environment for their computing entities. In such settings, context awareness is a computing paradigm in which schemes for context provision and discovery make applications aware of the changes taking place in their computing context, allowing them to gain advantage from context change, instead of employing a middleware layer for hiding the changes from the application.

On the coordinates of the surveys upon context awareness of Chen, Schilit and Abowd [1, 25, 101], *contextual information* or computing context is any piece of information used by an application in order to infer knowledge about other interesting computing entities (objects, persons or places) in its environment. *Primary context* can be divided into (possibly overlapping) categories: resources, such as neighbouring printers or the degree of congestion in the network; user context, meaning an object or person's location or status; physical context, such as temperature or moment in time; history of context, meaning the recording of any primary context over time. More complex conclusions about the environment can be drawn by combining several pieces of primary context into what is denoted by *secondary context*:

a source of information is indexed by one type of context, after which the result gets indexed by another.

In order for applications to be able to access contextual information in a dynamic network, *context provision* is the dissemination of contextual information from the host entity to a network neighbourhood, to enlarge its visibility scope; *context discovery* is the locating of the provided context by interested applications. In large networks, the design for context provision and discovery is complicated by the network size: network-wide provision is not feasible, and provision to a locality of players is used instead.

According to Schilit [101], the basic way in which applications make use of the surrounding context is by *contextual information and commands*, i.e. requests for either data or actions, which produce different results depending on the specific context in which they are issued. A context-aware application can also involve *context-triggered actions*, i.e. rules which specify what commands to be automatically executed, given certain properties which the context fulfills. Finally, applications can employ *automatic contextual re-configuration* to add and remove entire software components in response to certain properties of the context.

Ubiquitous systems are numerous, highly dynamic and their contextual information should be made accessible to a selected subset of the players in the network, features which make such systems difficult to enforce security policies upon. Simple access control upon pieces of contextual information cannot prevent their disclosure by agents collaborating on accessing them; static type checking cannot verify, on its own, that a highly dynamic network respects policies, hence dynamic type checking is called to verify the instances in which agents or code move in the network. Given that such systems are selective, privileges are fine-grained, so that users have different rights upon different pieces of contextual information.

Our calculus models infrastructure-based (as opposed to ad hoc) ubiquitous systems, inherently of hierarchical topology. In such systems, logical partitions are imposed over the network and all communication between mobile entities is mediated by the infrastructure; rooms, floors, buildings, and campuses are such logical cells which act as mediators for context provision, context discovery and general communication for the entities in their scope. Early systems in this category include the Xerox ParcTab [102], Active Badge [116], and GUIDE [28].

Computing entities are modelled by mobile ambients enclosing processes and other ambients, and the topology of the network evolves by the ambients' running of in and out movements. Furthermore, we introduce a modelling of contextual information, distributed through the network over scopes of various sizes and expressed by named macros. The context of an ambient is then the collected contextual information hosted by all ambients enclosing it up until the root ambient, at the ambient's current position.

Context provision is performed by an ambient through defining the named macro up in the network to a specified ambient destination to increase the macro's scope. Context discovery is performed by an ambient calling a macro name from a specified ambient and having the call replaced with the macro body, if such a macro exists in the current context. The context changes whenever ambients move and provide or consume contextual information.

The focus of this paper is to study the fulfilling of distributed security policies, in such a setting in which contextual information crosses boundaries in a hierarchical topology. The type of a process is a pair composed by a function recording the effects (provision or discovery) which ambients inner to the process have on other ambients, and by a set of ambient names which are allowed to reside in the process. Any ambient can host security policies in the form of process types, which all processes residing under this ambient should fulfill. This gives, at any given moment in the evolution of the network, for any process sitting in a context composed by a line of ancestor ambients (each with a policy of its own), that the process must adhere to the composition of all ancestor policies. Static type checking verifies that an initial state of the network is well-typed, for dynamic type checking to then verify the incremental movements of definitions and ambients in the network. We show well-typedness to be verified over any sequence of reductions in the system, define errors and show that they cannot appear in a well-typed system.

The rest of this paper is organised as follows: Section 4.2 presents the syntax and the basic semantics of our calculus, Section 4.3 the type systems, the notion of well-typedness and the subject reduction theorem, and Section 4.4 the operational semantics and the notion of type soundness. Finally, Section 4.5 illustrates a case study modelling a ubiquitous computing infrastructure in a hospital, and Section 4.6 reviews closely related work and concludes.

## 4.2 A Calculus for Context Awareness

In this section we briefly present the syntax and the basic, untyped semantics of the calculus. In the complete syntax from Fig. 4.1, the nil process  $0$ , parallel composition  $P \mid P'$ , ambient  $a[P]$ , name restriction  $\nu z P$  and movement primitives  $in a.P$  and  $out.P$  are all inherited from and have the same meaning as in the Mobile Ambients calculus [21]; as in the Boxed Ambients calculus [18], there is no *open* capability. From Zimmer's calculus for context awareness [125] we borrow the idea of contextual information as macro definitions residing at ambients. A definition of the basic form  $def f \triangleright Q in P$  defines macro  $f$  as being the process  $Q$  in a floating definition ( $f \triangleright Q$ ) and continues execution with  $P$ , and any call  $f$  for this macro would be replaced

processes	$P$	$::=$	$0$	no process
			$P \mid P'$	parallel composition
			$f^a$	macro call, $f \in \mathcal{F}$
			$def^a D \text{ in } P$	macro definition
			$a_G^\tau [P]$	mobile entity, $a \in \mathcal{A}$
			$EP$	public definitions
			$\nu z P$	name restriction, $z \in \mathcal{F} \cup \mathcal{A}$
			$\text{in } a.P$	movement in
			$\text{out}.P$	movement out
		definitions	$E$	$::=$
	$(D)^{a,\tau}$			floating definition, downward
$D$	$::=$		$F \mid !F$	
$F$	$::=$		$f \triangleright P$	macro

Figure 4.1: Syntax

by its body  $Q$ ; a simplified semantics for a definition and call are:

$$\text{DEF} \quad def f \triangleright Q \text{ in } P \longrightarrow (f \triangleright Q) P \quad \text{CALL} \quad (f \triangleright Q) f \longrightarrow Q$$

The decorations  $\tau$  and  $G$  on ambients and floating definitions from Fig. 4.1 and Table 4.1 in the following are security and entry policies, respectively, and their syntax and meaning are detailed in Section 4.3; they are ignored in this section.

Macro definitions come in two flavours: a *one-shot definition*  $f \triangleright Q$  is one which is consumed by that macro being called and is suitable for modelling network packets (if one sees data communication as a simple feature of context); a *permanent definition*  $!f \triangleright Q$  is one which can be instantiated by any number of macro calls, and in fact behaves like an infinite set of one-shot definitions.

Unlike Zimmer's calculus, definitions and calls of contextual information are not only made by processes to and from their enclosing ambient, but cross multiple ambients' boundaries; hence, definitions and calls are tagged with the identity of the destination and source ambient, respectively: process  $def^b f \triangleright Q \text{ in } P$  publishes macro  $f$  at any ambient  $b$  in the ancestor ambient line of this process, and process  $f^b$  calls macro  $f$  from any such ambient. The calls and definitions being tagged with the name of a destination ambient fits the infrastructure setting, in which mobile entities have a degree of knowledge about the identities of servers, gateways and about the protocols in the network, at least such that identities of other points of interest can

be provided by calling these.

For this, a process defining macro  $f$  to ambient  $b$  evolves into a floating definition destined to  $b$ :

$$\text{DEF} \quad def^b f \triangleright Q \text{ in } P \longrightarrow (f \triangleright Q)^b P$$

for the floating definition to travel upwards to destination in fluid movements of the form

$$(f \triangleright Q)^b P \mid R \equiv (f \triangleright Q)^b (P \mid R)$$

(included among the structural congruence rules in Table 4.1) and

$$\text{UP} \quad a \left[ (f \triangleright Q)^b P \right] \longrightarrow (f \triangleright Q)^b a [P]$$

(a semantics rule in Section 4.4). If permanent, a definition at its destination  $b \left[ (f \triangleright Q)^b P \right]$  expands single instances upon calls, with  $(!F) \equiv (F)(!F)$ . When called, a single floating definition moves down from its host ambient to the calling process by the inverse of the upward movements above:

$$\text{DOWN} \quad (f \triangleright Q)^b a [P] \longrightarrow a \left[ (f \triangleright Q)^b P \right]$$

for the call to be fulfilled at the calling ambient:

$$\text{CALL} \quad (f \triangleright Q)^b f^b \longrightarrow Q$$

This scope extension for contextual information follows the idea that context is formed by publishing information from a source to a wider locality of users; it allows for the modelling of context provision and discovery over entire localities of agents, unlike Zimmer's model, which bounds the communication of context within the enclosing ambient of the communicating process. This scheme also effectively models Schilit's contextual information and commands from [101]: a call for a service has a dynamic interpretation varying with context. Furthermore, the intermediate steps a floating definition takes in order to reach a calling ambient gives that either contextual information or its destination can unexpectedly become unreachable with the ambients' changing of location; this fits the profile of highly dynamic networks.

As an example, take a hospital's ubiquitous system supporting collaboration among mobile employees (inspired by [7]); the hospital network infrastructure is ambient  $hni$ , and a doctor's personal digital assistant  $doc$  currently residing in the operating ward  $ow$  updates the network about his location  $docloc$  of current value  $P$ , so that the tag of any nurse (at any location in the hospital, such as office  $of$ ) to locate him:

$$hni \left[ ow \left[ doc \left[ def^{hni} !docloc \triangleright P \text{ in } 0 \right] \right] \mid of \left[ nurse \left[ docloc^{hni} \right] \right] \right]$$

Table 4.1: Structural congruence is the smallest congruence relation satisfying these rules.

$$\begin{array}{ll}
P|0 \equiv P & \nu z 0 \equiv 0 \\
P|Q \equiv Q|P & \nu z \nu w P \equiv \nu w \nu z P \\
(P|Q)|R \equiv P|(Q|R) & \nu z (P|Q) \equiv P|\nu z Q \text{ if } z \notin fn(P) \\
(!F)^{a,\tau} \equiv (F)^{a,\tau} (!F)^{a,\tau} & \nu z (a_G^\tau[P]) \equiv a_G^\tau[\nu z P] \text{ if } z \notin fn(a_G^\tau[]) \\
E_1 E_2 \equiv E_2 E_1 & \nu z E P \equiv E \nu z P \text{ if } z \notin fn(E) \\
E P|Q \equiv E(P|Q) & \alpha\text{-conversion}
\end{array}$$

The nurse's code cannot execute without the *docloc* macro available in its context, but after the definition becomes visible the system is:

$$hni \left[ (!docloc \triangleright P)^{hni} \left( ow [doc []] \mid of \left[ nurse \left[ docloc^{hni} \right] \right] \right) \right]$$

and one instance of the definition travels downwards to meet the request:

$$hni \left[ (!docloc \triangleright P)^{hni} \left( ow [doc []] \mid of \left[ nurse \left[ (docloc \triangleright P)^{hni} docloc^{hni} \right] \right] \right) \right]$$

and  $(docloc \triangleright P)^{hni} docloc^{hni}$  reduces to  $P$ .

We use two sets of names in our syntax: macro names  $f$  belong to an infinite set  $\mathcal{F}$ , and ambient names  $a$  belong to an infinite set  $\mathcal{A}$ . A restriction  $\nu z$  can only be made upon any name  $z$  in  $\mathcal{F} \cup \mathcal{A}$ , and  $\alpha$ -conversion substitutes names from  $\mathcal{F} \cup \mathcal{A}$ . We adapt the convention: when considering a process, we assume that the bound names of the process are different from its free names, if necessary, after a number of  $\alpha$ -conversion steps.

Crucially, the restriction operator  $\nu$  is the only binder in our syntax; i.e., a macro name in a macro definition is not a binder. Hence, the bound and free names of a process  $bn(P)$ ,  $fn(P)$ , the substituted process  $P\alpha$  and the extrusion of restrictions are defined as is standard.

### 4.3 Type Systems and Well-Typedness

We introduce a typing system which specifies the effects which a process in our calculus is allowed to have. A type is two-fold, as depicted in Fig. 4.2: on one hand, a security policy  $\tau$  is a function mapping a source ambient name to a destination ambient name and to a set of effects which the source ambient can have upon the destination ambient. An effect is the ability to run either a definition of or a call for a macro name  $f$ . On the other hand, the mere presence of an ambient in a process is a securable feature, thus we also enforce entry policies  $G$ , as being subsets from the set of ambient names.

<i>Entry policy</i>	$G \subseteq \mathcal{A}$
<i>Effects</i>	$\mathcal{E} = \bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$
<i>Security policy</i>	$\tau = \mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$

Figure 4.2: Entry policies and security policies

As is natural for Mobile-Ambients-based calculi, policies reside at the ambient membrane. We call a *subambient* of a any ambient enclosed by  $a$  either directly, or at any inner level. Then, we write  $a_G^\tau[P]$  to specify that  $P$  should satisfy policies  $\tau$  and  $G$ . If  $b \notin G$ , then  $P$  should not have a subambient  $b$  at all. Furthermore, if  $\tau(b)(c) \not\supseteq def(f)$ , then  $P$  should not have a subambient  $b$  directly enclosing a process  $def^c f \triangleright Q$  in  $R$ ; similarly for macro calls.

As an example, the policy  $\tau$  of the hospital network infrastructure  $hni_G^\tau$  is such that only employees have access to the patient record of a certain VIP, accessible through the protocol (i.e., macro name)  $vip$  directed at  $hni$ ; thus, for any visitor  $vis$  the policy states that  $\tau(vis)(hni) \not\supseteq call(vip)$  and applies throughout the hospital system, without the need of it being multiplied (for example, at the lower, ward level). Also, supposing that one floor  $floor3_C^\gamma$  of the hospital is closed to anybody but the hospital's employees,  $vis \notin C$ , so that no further policies upon  $vis$  are needed inside  $floor3$ .

A fully-permissive entry policy is  $\mathcal{A}$ , and a fully-permissive security policy is denoted by  $\omega$ , with

$$\forall b, c \in \mathcal{A} \quad \forall f \in \mathcal{F} \quad \omega(b)(c) \supseteq def(f) \text{ and } \omega(b)(c) \supseteq call(f).$$

Two policies can be composed to denote a policy which satisfies both original policies; the composition of entry policies  $G$  and  $H$  is  $G \cap H$ , while the composition of security policies  $\tau$  and  $\sigma$  is the policy  $\tau \cap \sigma$ , with

$$(\tau \cap \sigma)(b)(c) = \tau(b)(c) \cap \sigma(b)(c).$$

Composition is both commutative and associative, and  $\omega$  is the identity element,  $\omega \cap \tau = \tau$ , for all  $\tau$ . We denote the set of all security policies by  $\mathcal{T}$ .

We do not detail the matter of defining a particular syntax for policies. We only assume such a syntax to infer a notion of *free names* of policies  $\tau_G$ , written  $fn(\tau_G)$ . For all non-free combination of names, the policies are implicitly either fully-permissive or fully-dismissive; then, for the free names, policies specify explicitly either restrictions or allowances, in both cases finitely many.

This notion of free names only needs to satisfy that the set of free ambient names,  $fn(\tau_G) \cap \mathcal{A}$ , and the set of free macro names,  $fn(\tau_G) \cap \mathcal{F}$ , are both finite sets, and that the following conditions are satisfied:

---

NULL	$a_G^\tau \vdash 0$	PAR	$\frac{a_G^\tau \vdash P \quad a_G^\tau \vdash P'}{a_G^\tau \vdash P P'}$	CALL	$\frac{\tau(a)(b) \ni call(f)}{a_G^\tau \vdash f^b}$
		DEF	$\frac{\tau(a)(b) \ni def(f) \quad a_G^\tau \vdash P}{a_G^\tau \vdash def^b ? f \triangleright Q \text{ in } P}$		
		AMB	$\frac{b \in G \quad b_{G \cap H}^{\tau \cap \sigma} \vdash P}{a_G^\tau \vdash b_H^\sigma[P]}$		
MSG	$\frac{a_G^\tau \vdash P}{a_G^\tau \vdash (D)^{b, \sigma} P}$	RES	$\frac{a_G^\tau \vdash P}{a_G^\tau \vdash \nu z P} \quad z \notin fn(\tau_G)$		
	IN	$\frac{a_G^\tau \vdash P}{a_G^\tau \vdash in b.P}$	OUT	$\frac{a_G^\tau \vdash P}{a_G^\tau \vdash out.P}$	

---

Figure 4.3: Type system for active code

- for entry policies, the set of allowed names of subambients  $G$  is either finite or cofinite, relative to the finite set of free ambient names  $fn(\tau_G) \cap \mathcal{A}$ ;
- for security policies, any set of allowed definitions and calls  $\tau(a)(b)$ ,  $\forall a, b \in \mathcal{A}$  is either finite or cofinite relative to the finite set of free macro names  $fn(\tau_G) \cap \mathcal{F}$ , and this former set is uniformly defined for all non-free ambient names.

Given our hierarchical network topology of mobile agents, each hosting a policy, a process sitting in the scope of a set of ambients should comply with the collected policies of those ambients. The initial state of a system is statically checked for compliance with all the system's policies, and then only individual moves of ambients are checked in the operational semantics described in Section 4.4.

### 4.3.1 A Type System for Active Code

There are two interconnected type systems; the one for *active* processes (i.e. all processes not a macro definition body), with type statements of the form  $a_G^\tau \vdash P$  stating that  $P$  running at ambient  $a$  complies with the type  $\tau$ , is depicted in Fig. 4.3. The one for *inactive* processes (i.e. definition bodies, which travel over ambient boundaries before being executed) is discussed subsequently. Finally, Def. 4.1 at the end of the section defines a process to be well-typed if it is both actively and inactively well-typed.

For the compactness of our typing expressions, we write  $?f$  to stand for both permanent  $!f$  and one-shot  $f$  definitions; similarly, we write  $\sigma?$  to stand for both  $\sigma$  and no policy.

The interesting rule is AMB: in a top-down checking fashion, for a subambient  $b_H^\tau[P]$  sitting in an ambient  $a$  to be actively typed, process  $P$  has to be typed with the composed security policy and the intersected entry policy given by the policies of  $b$  and  $a$ ; hence, in any statement of the form  $a_G^\tau \vdash P$ ,  $P$  sits in ambient  $a$ ,  $\tau$  is the security policy composed of all security policies belonging to ambients ancestor to  $P$ , and  $G$  is the intersection of their entry policies.

Then, for a process with a call effect  $f^b$  sitting in  $a_G^\tau$  to be actively typed, the condition in the CALL rule is  $\tau(a)(b) \ni \text{call}(f)$ , for the effect to be allowed by the composed policy  $\tau$ . The same goes for definition effects in the DEF rule, with a further check on the continuation process  $P$  following the definition. Entry policies are checked in the AMB rule, in which a subambient  $b$  is allowed to run at  $a_G^\tau$  if  $b \in G$ .

In the RES rule,  $\nu z P$  is typed with respect to  $a_G^\tau$  if  $z$  is not one of the free names of policies  $\tau_G$ , i.e. those names upon which the policies explicitly specify restrictions or allowances. On the other hand, if  $z$  is such a name,  $z$  is  $\alpha$ -converted to a fresh name. Thus, if e.g. a fresh ambient is created using the restriction operator, at type checking the ambient name is non-free and is checked against implicit policies (e.g. fully-restrictive).

Movement capabilities *in* and *out* are not type checked themselves, but the operational semantics will impose well-typedness conditions for moves, as shown in Section 4.4. Furthermore, we assume that a system starts in a state without floating definitions, which allows us to only check statically definition bodies only once in their defining process (DEF), and not when floating (MSG).

### 4.3.2 A Type System for Inactive Code

For the type checking of definition bodies in Fig. 4.4, given that definitions travel over ambient boundaries, the check is more complex. Intuitively, a defining process  $def^b ?f \triangleright Q \text{ in } P$  has as result the definition  $(?f \triangleright Q)^b$  travelling upwards to ambient  $b$  and then downwards to any calling ambient; in the inactive type system below, it is ensured that when having arrived at  $b$ , the body process  $Q$  complies with the composed policies of  $b$  and its ancestors. Subsequently in Section 4.4, the remaining down movements will be checked dynamically, to ensure that  $Q$  complies with all the policies imposed within  $b$  down to the calling ambient.

The *context function*  $\mathcal{C}$  records the context, in terms of security and entry policies, with which a definition body should comply when travelling upwards, in order to maintain the well-typedness of the system. The func-

---

NULL	$\mathcal{C}, a_G^\tau \vdash 0$	PAR	$\frac{\mathcal{C}, a_G^\tau \vdash P \quad \mathcal{C}, a_G^\tau \vdash P'}{\mathcal{C}, a_G^\tau \vdash P P'}$	CALL	$\mathcal{C}, a_G^\tau \vdash f^b$
DEF	$\frac{\mathcal{C}, a_G^\tau \vdash P \quad a_G^\tau[Q] \text{ well-typed}}{\mathcal{C}, a_G^\tau \vdash def^a ? f \triangleright Q \text{ in } P}$		$\frac{\mathcal{C}, a_G^\tau \vdash P \quad b\mathcal{C}(b)[Q] \text{ well-typed}}{\mathcal{C}, a_G^\tau \vdash def^b ? f \triangleright Q \text{ in } P}$		
		AMB	$\frac{\mathcal{C}[\{a\} \cup G \rightarrow \tau_G], b_{G \cap H}^{\tau \cap \sigma} \vdash P}{\mathcal{C}, a_G^\tau \vdash b_H^\sigma[P]}$		
MSG	$\frac{\mathcal{C}, a_G^\tau \vdash P}{\mathcal{C}, a_G^\tau \vdash (D)^{b, \sigma} P}$	RES	$\frac{\mathcal{C}, a_G^\tau \vdash P}{\mathcal{C}, a_G^\tau \vdash \nu z P} \quad z \notin fn(\mathcal{C}) \cup fn(\tau_G)$		
		IN	$\frac{\mathcal{C}, a_G^\tau \vdash P}{\mathcal{C}, a_G^\tau \vdash in b.P}$	OUT	$\frac{\mathcal{C}, a_G^\tau \vdash P}{\mathcal{C}, a_G^\tau \vdash out.P}$

---

Figure 4.4: Type system for inactive code

tion is totally defined on the ambient names set  $\mathcal{A}$  and takes values in pairs of security policies (from the set  $\mathcal{T}$ ) and entry policies (from  $\mathcal{P}(\mathcal{A})$ ). Formally, the type of this function is (written here using a nonstandard notation reflecting our notation for decorating ambients):

$$\mathcal{C} : \mathcal{A} \longrightarrow \frac{\mathcal{T}}{\mathcal{P}(\mathcal{A})}.$$

Thus, whenever  $\mathcal{C}(b) = \tau_G$ , we have  $a\mathcal{C}(b) = a_G^\tau$ .

The intuition is that, given a definition  $def^b f \triangleright Q \text{ in } P$ , its policy-wise context  $\mathcal{C}(b)$  returns the collected policies of ambients above  $b$ , so that in order for  $Q$  to be run under  $b$ ,  $b\mathcal{C}(b)[Q]$  needs to be well-typed (i.e., both actively and inactively typed against the collected policies  $\mathcal{C}(b)$ ).

As with both security and entry policies,  $\mathcal{C}$  has a fully-permissive instantiation  $\Omega$  with:

$$\Omega(\mathcal{A}) = \overset{\omega}{\mathcal{A}}.$$

We frequently abuse the notation and write  $\mathcal{C}(H) = \tau_G$  to state that the value of  $\mathcal{C}$  for all elements in set  $H$  is  $\tau_G$ . Also, we frequently define an instantiation of such a context function  $\mathcal{C}$  by writing updates upon  $\Omega$ , of the form  $\Omega[H \rightarrow \tau_G]$ .

Inactive type statements have the form  $\mathcal{C}, a_G^\tau \vdash P$ . This intuitively means that all the definition bodies  $Q$  waiting to be activated in processes of the form  $def^b f \triangleright Q \text{ in } R$  inside  $P$  sitting in ambient  $a$  will have to be able to run inside the destination ambient: if the destination is the host ambient  $a$  itself, then the definition bodies have to comply with  $a_G^\tau$ , in which, as in the case of the active type system,  $\tau$  and  $G$  are composed by or intersected

from all ambients above. On the other hand, for a destination  $b \neq a$ , the context function  $\mathcal{C}(b)$  returns the pair of collected policies from the set of ambients ancestor to  $b$  (if  $b$  is in  $P$ 's context) or from the maximal set of ambients which can ever exist above  $b$ , were  $P$  to move under  $b$ .

For consistency, we assume that if one of our systems is not of the form  $a_G^\tau[P]$ , there exists a unique ambient  $world_A^\omega$  with full permissions at the root of the system. The type checking then proceeds top-down, while also collecting contextual policies in a context function  $\mathcal{C}$ : at top level, there exists no restricting context other than the enclosing root ambient  $a_G^\tau$ , and a typing statement looks like  $\Omega, a_G^\tau \vdash P$ ; when another ambient is encountered,  $\Omega, a_G^\tau \vdash b_H^\sigma[Q]$  if  $\Omega[\{a\} \cup G \rightarrow \tau_G], b_{G \cap H}^{\tau \cap \sigma} \vdash Q$  and the context function for ambient  $b$  is the updated  $\Omega[\{a\} \cup G \rightarrow \tau_G]$ , meaning that if a definition in  $Q$  is destined to  $b$ , it should comply with the composed policies of  $a$  and  $b$ ; if destined to  $a$  or any other ambient allowed inside  $a$ , it should comply with the policies of  $a$ . This last fact is to ensure such a well-typedness, that the dynamic checks at *in* movements are simple, as will be detailed in Section 4.4.

We then define well-typedness as being the dual, active and inactive, checking of processes.

**Definition 4.1 (Well-typedness)** *A system  $a_G^\tau[P]$  is well-typed if  $a_G^\tau \vdash P$  and  $\Omega, a_G^\tau \vdash P$ . A system  $P$  without a root ambient is well-typed if  $world_A^\omega \vdash P$  and  $\Omega, world_A^\omega \vdash P$ .*

## 4.4 Operational Semantics and Type Soundness

Our operational semantics from Fig. 4.5 preserves the flavour of reduction rules IN, OUT, STRUCT and CONTEXT from standard Mobile Ambients. Moreover, rules UP and DOWN depict the crossing of ambient borders by definitions floating from their origin to the destination or from the latter to a calling ambient; if these movements are successful, a pair composed by a called definition adjacent to its call reduces to the definition body. In Fig. 4.5, the *active contexts*  $\mathbf{C}$  are the processes with one hole in active locations (i.e, locations in which a process can suffer a reduction immediately), as in the following:

$$\mathbf{C} ::= [\cdot] \mid \mathbf{C} \mid P \mid a_G^\tau[\mathbf{C}] \mid E \mathbf{C} \mid \nu z \mathbf{C}$$

Static checking, discussed in Section 4.3, verifies that an initial state of the system is well-typed. It includes, in the inactive type checking from Fig. 4.4, a scheme to verify that even the bodies of those definitions destined to ambients which are not currently in the context will safely run at those

---


$$\begin{array}{c}
\text{UP} \quad a_G^\tau[(D)^b P] \longrightarrow (D)^b a_G^\tau[P] \quad a_G^\tau[(D)^a P] \longrightarrow a_G^\tau[(D)^{a,\tau} P] \\
\text{DOWN} \quad \frac{a_G^{\sigma \cap \tau}[Q] \text{ well-typed}}{(f \triangleright Q)^{b,\sigma} a_G^\tau[P] \longrightarrow a_G^\tau[(f \triangleright Q)^{b,\sigma \cap \tau} P]} \\
\text{DEF} \quad \text{def}^a D \text{ in } P \longrightarrow (D)^a P \quad \text{CALL} \quad (f \triangleright P)^{a,\tau} f^a \longrightarrow P \\
\text{IN} \quad \frac{b \in G \quad b_G^\tau \vdash Q \mid R \quad \Omega[\{a\} \cup G \rightarrow \tau_G], b_H^\tau \vdash \circ Q \mid R}{a_G^\tau[P] \mid b_H^\sigma[\text{in } a.Q \mid R] \longrightarrow a_G^\tau[P \mid b_H^\sigma[Q \mid R]]} \\
\text{OUT} \quad a_G^\tau[P \mid b_H^\sigma[\text{out}.Q \mid R]] \longrightarrow a_G^\tau[P] \mid b_H^\sigma[Q \mid R] \\
\text{STRUCT} \quad \frac{P \equiv P' \quad P \longrightarrow Q \quad Q \equiv Q'}{P' \longrightarrow Q'} \quad \text{CONTEXT} \quad \frac{P \longrightarrow Q}{\mathbf{C}[P] \longrightarrow \mathbf{C}[Q]}
\end{array}$$


---

Figure 4.5: Operational semantics

ambients, whenever they become present. That verifies, for example, that in the setting

$$c_K^\rho [a_G^\tau[P] \mid b_H^\sigma[\text{in } a.Q \mid R]],$$

given that  $a \in K$ , definitions from  $Q \mid R$  destined to  $a$  are statically checked against the policies of  $c_K^\rho$ . Thus, when  $b$  performs the *in* movement, it is sufficient to dynamically check that  $Q \mid R$  is well-typed against  $a$ 's policies, as shown by rule IN.

Furthermore, since static checking insures that the body of a definition is safe to be run in the destination ambient it reached after moving upwards, all that is left for the DOWN movement to dynamically check is the compliance with the collected policies of each ambient down until the calling one. This is ensured by decorating the floating definition  $(f \triangleright Q)^a$ , when it has reached ambient  $a_G^\tau$ , with a *signature* equal to  $\tau$ ; with every down movement, the signature is composed with the policy of the newly-crossed ambient, and  $Q$  is checked against its signature and the crossed ambient's group policy. This scheme has the flavour of firewall-carrying code.

As an observation, the same static checking scheme, together with the dynamic checking at the DOWN movement, have the side effect that some breakings of policy could be caught by either of these checks (and are caught by the earliest, static checking), if the case is such that the ambient tree in which the definition moves up before reaching the destination also includes the calling ambient. On the other hand, if the upward and downward trees are disjunct, only the dynamic check can verify the definition body.

Furthermore, the reader may have wondered about our less standard approach of modelling the availability of contextual information partially

$$\begin{array}{c}
\text{ERR\_DEF} \quad \frac{(\sigma \cap \tau)(a)(b) \not\equiv \text{def}(f)}{\mathbf{C}_H^\sigma [a_G^\tau [\text{def}^b f \triangleright Q \text{ in } P \mid R]] \longrightarrow \text{err}} \\
\text{ERR\_CALL} \quad \frac{(\sigma \cap \tau)(a)(b) \not\equiv \text{call}(f)}{\mathbf{C}_H^\sigma [a_G^\tau [f^b \mid P]] \longrightarrow \text{err}} \\
\text{ERR\_IN} \quad \frac{H \not\equiv a}{\mathbf{C}_H^\sigma [a_G^\tau [P]] \longrightarrow \text{err}} \quad \text{ERR\_STR} \quad \frac{P \equiv P' \quad P' \longrightarrow \text{err}}{P \longrightarrow \text{err}}
\end{array}$$

Figure 4.6: Errors

Table 4.2: The policies of contexts

$\mathbf{C}_G^\tau$	$\tau$	$G$
$[\cdot]$	$\omega$	$A$
$\mathbf{C}_H^\sigma \mid P$	$\sigma$	$H$
$c_K^\rho [\mathbf{C}_H^\sigma]$	$\sigma \cap \rho$	$K \cap H$
$E \mathbf{C}_H^\sigma$	$\sigma$	$H$
$(\nu z) \mathbf{C}_H^\sigma$	$\sigma$	$H$

through reduction (rules UP, DOWN in Fig. 4.5), instead of structural congruence. Our choice is motivated by the fact that we want to model explicitly the operational behaviour of firewalls in terms of filtering capabilities, in the event of contextual information crossing firewalls during downward moves.

We can now state the subject reduction property, as follows.

**Theorem 4.1 (Subject Reduction)** *If  $P$  is well-typed and  $P \xrightarrow{*} Q$ , then  $Q$  is well-typed.*

In the behaviour of a system it is considered an error,  $P \rightarrow \text{err}$ , if an effect (be it a definition or a call) breaks the security policy of any ambient in its context, or if an ambient's presence breaks the entry policy of any ambient in its context, as in Fig. 4.6. The superscript  $\tau$  and the subscript  $G$  decorating a context  $\mathbf{C}$  are the composition of all security policies above  $[\cdot]$ , and the intersection of all entry policies above  $[\cdot]$ , respectively. The composed  $\tau$  and  $G$  for a context  $\mathbf{C}$  are defined inductively in Table 4.2.

We then state that if a system is well-typed, it can never display an error.

**Theorem 4.2 (Type Soundness)** *If  $P$  is well-typed then  $P \not\rightarrow^* \text{err}$ .*

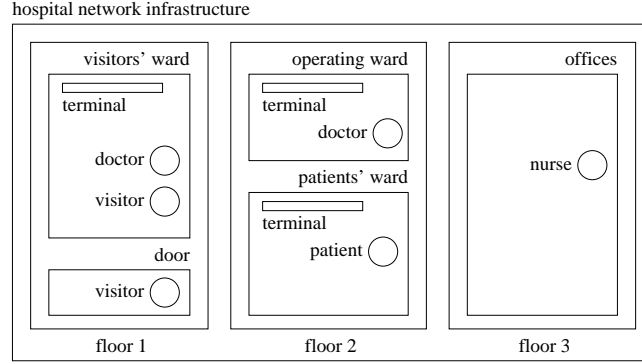


Figure 4.7: The hospital system

## 4.5 Case Study: Ubiquitous Computing in a Hospital

Consider a ubiquitous computing infrastructure in a hospital, inspired by the AWARE project [7], as in Fig. 4.7. The hospital network infrastructure  $hni$  maintains the patient records and keeps track of the location of doctors, nurses, patient and visitors, all of them carrying PDAs, by having their PDAs announce their presence periodically.

The patient records are read and updated by nurses and doctors using either their tabs or the terminals in the operation ward and patients' ward. Patients and visitors have lower degrees of rights upon accessing—with their tabs or at the terminals—patient records and employee locations.

### 4.5.1 The Guessing Visitor

The policy  $\tau$  of the hospital network infrastructure  $hni_G^\tau$  is such that only employees have access to the patient record  $P$  of a certain VIP, accessible through the protocol (i.e., macro name)  $vip$  directed at  $hni$ ; for any visitor  $vis$  the policy states that

$$\tau(vis)(hni) \not\equiv call(vip).$$

An overly-curious visitor  $vis_V^\omega$  who guessed the protocol for accessing the VIP's record would enter through the door  $door_A^\omega$  (i.e., a new ambient  $vis_V^\omega$  would be run as a subambient of  $door$ , after being sprouted from a banged definition resident at  $door$ ). The initial, static type checking determines that the system is not well-typed. As an observation, expression  $\nu h (def^{door} !h \triangleright Q \mid h^{door} \text{ in } h^{door})$  effectively models  $!Q$ .

$$\begin{array}{c}
\frac{\Omega, hni_G^\tau \vdash (vip \triangleright P)^{hni} \text{ door}_A^\omega [\nu h (def^{door} !h \triangleright vis_V^\omega [vip^{hni}] | h^{door} \text{ in } h^{door})]}{\frac{\Omega, hni_G^\tau \vdash \text{ door}_A^\omega [\nu h (def^{door} !h \triangleright vis_V^\omega [vip^{hni}] | h^{door} \text{ in } h^{door})]}{\frac{\Omega[\{hni\} \cup G \rightarrow \bar{G}], \text{ door}_G^\tau \vdash \nu h (def^{door} !h \triangleright vis_V^\omega [vip^{hni}] | h^{door} \text{ in } h^{door})}{h \notin fn(\bar{G}) \quad \frac{\Omega[\{hni\} \cup G \rightarrow \bar{G}], \text{ door}_G^\tau \vdash def^{door} !h \triangleright vis_V^\omega [vip^{hni}] | h^{door} \text{ in } h^{door}}{\text{ door}_G^\tau [vis_V^\omega [vip^{hni}]] \text{ well-typed}}}}} \\
\frac{\text{ door}_G^\tau [vis_V^\omega [vip^{hni}]] \text{ well-typed}}{\frac{\text{ door}_G^\tau \vdash vis_V^\omega [vip^{hni}]}{vis_{G \cap V}^\tau \vdash vip^{hni}}} \\
\tau(vis)(hni) \ni call(vip)
\end{array}$$

Figure 4.8: The derivation tree for the guessing visitor scenario

The system, only focused on the door, is formalised as

$$hni_G^\tau[(vip \triangleright P)^{hni} \text{ door}_A^\omega [\nu h (def^{door} !h \triangleright vis_V^\omega [vip^{hni}] | h^{door} \text{ in } h^{door})]].$$

We show that its well-typedness depends on at least the policy condition which is not met,  $\tau(vis)(hni) \not\ni call(vip)$ ; the other conditions for its well-typedness, possibly satisfied, are not depicted. The system is well-typed if it is also inactively well-typed, as in the derivation tree in Fig. 4.8. Given that the well-typedness of the system depends on a condition which is false, the error is raised at this stage.

### 4.5.2 The Conspiring Nurse

Have a nurse  $nurse_N^\omega$  who agreed to conspire with the overly-curious visitor  $vis_V^\omega$ . They plan on an indirect access scheme to the record  $vip^{hni}$ , for the visitor. The visitor's actions will be inconspicuous: the visitor residing in the visitors' ward  $vw$  and the nurse in her office  $of$  agree upon a new, private service (i.e., macro name)  $key$ , assuming that the nurse is allowed to define  $key$ ,  $key \notin fn(\tau)$ . The nurse makes  $key$  give access to the VIP record's service name:  $def^{hni} key \triangleright vip^{hni} \text{ in } 0$ , for then the visitor to call  $key^{hni}$ .

The hospital system is now:

$$\begin{array}{c}
hni_H^\tau[(!vip \triangleright P)^{hni} \nu key ( \\
\text{ floor } 1_A^\alpha [vw_A^\sigma [vis_V^\omega [key^{hni}]]] | \\
\text{ floor } 3_C^\gamma [of_O^\pi [nurse_N^\omega [def^{hni} key \triangleright vip^{hni} \text{ in } 0]]]]
\end{array}$$

Static checking on this state of the system passes without errors. The dynamic checking at runtime raises the error the moment in which the nurse's

definition is about to enter the visitor's ambient. The reduction steps, up until the raising of the error, are:

$$\begin{aligned}
& \xrightarrow{*} \nu \text{key hni}^\tau_H \left[ \left( \text{key} \triangleright \text{vip}^{\text{hni}} \right)^{\text{hni}} (!\text{vip} \triangleright P)^{\text{hni}} \right. \\
& \quad \text{floor1}^\alpha_A \left[ \text{vw}^\sigma_A \left[ \text{vis}^\omega_V \left[ \text{key}^{\text{hni}} \right] \right] \right] \mid \\
& \quad \text{floor3}^\gamma_C \left[ \text{of}^\pi_O \left[ \text{nurse}^\omega_N \left[ \right] \right] \right] \\
& \xrightarrow{*} \text{hni}^\tau_H \left[ (!\text{vip} \triangleright P)^{\text{hni}} \right. \\
& \quad \text{floor1}^\alpha_A \left[ \text{vw}^\sigma_A \left[ \nu \text{key} \left( \left( \text{key} \triangleright \text{vip}^{\text{hni}} \right)^{\text{hni}, \tau \cap \alpha \cap \sigma} \text{vis}^\omega_V \left[ \text{key}^{\text{hni}} \right] \right) \right] \right] \mid \\
& \quad \text{floor3}^\gamma_C \left[ \text{of}^\pi_O \left[ \text{nurse}^\omega_N \left[ \right] \right] \right]
\end{aligned}$$

in which the down movement

$$\left( \text{key} \triangleright \text{vip}^{\text{hni}} \right)^{\text{hni}, \tau \cap \alpha \cap \sigma} \text{vis}^\omega_V \left[ \text{key}^{\text{hni}} \right] \longrightarrow \text{vis}^\omega_V \left[ \left( \text{key} \triangleright \text{vip}^{\text{hni}} \right)^{\text{hni}, \tau \cap \alpha \cap \sigma} \text{key}^{\text{hni}} \right]$$

is allowed only if  $\text{vis}^{\tau \cap \alpha \cap \sigma}_V \left[ \text{vip}^{\text{hni}} \right]$  is well-typed, a condition which depends on  $\tau(\text{vis})(\text{hni}) \ni \text{call}(\text{vip})$ :

$$\frac{\frac{\text{vis}^{\tau \cap \alpha \cap \sigma}_V \left[ \text{vip}^{\text{hni}} \right] \text{ well-typed}}{\text{vis}^{\alpha \cap \tau}_V \vdash \text{vip}^{\text{hni}}}}{\tau(\text{vis})(\text{hni}) \ni \text{call}(\text{vip})}$$

### 4.5.3 The Wandering Visitor

Have the hospital policies devise a scheme to limit visitors from loading their own services (say, named *key*) throughout the hospital. For this, the first floor  $\text{floor1}^\alpha_A$ , enclosing the door and the visitors' ward, has  $\text{vis} \in A$  and  $\alpha$  allowing visitors to publish services, but only up to the floor's level:  $\alpha(\text{vis})(\text{floor1}) \ni \text{def}(\text{key})$ , but  $\alpha(\text{vis})(\text{hni}) \not\ni \text{def}(\text{key})$ . The second floor still allows patients to be present during visiting hours, but imposes that they shouldn't publish anything while there, to any destination: both  $\beta(\text{vis})(\text{floor2}) \not\ni \text{def}(\text{key})$  and  $\beta(\text{vis})(\text{hni}) \not\ni \text{def}(\text{key})$ . There should never be a visitor on the third floor,  $\text{vis} \notin C$ , hence  $\gamma$  poses no further restrictions upon the visitor's actions.

Have the visitor  $\text{vis}^\omega_V$  planning to tour the hospital's three floors in search of spots to load the system with his *key* service. A try at publishing *key* at *hni* from the visitor's ward:

$$\text{hni}^\tau_H \left[ \text{floor1}^\alpha_A \left[ \text{vw}^\sigma_A \left[ \text{vis}^\omega_V \left[ \text{def}^{\text{hni}} \text{key} \triangleright P \text{ in } 0 \right] \right] \right] \right]$$

is signalled at static checking, since the active well-typedness of  $\text{vis}^\omega_V$  in this context of security policies depends on a security condition which doesn't

hold:

$$\frac{vis_{H \cap V}^{\tau \cap \alpha \cap \sigma} \vdash def^{hni} key \triangleright P \text{ in } 0}{\alpha(vis)(hni) \ni def(key)}$$

The visitor would, however, be able to load the first floor (which acts like a sandbox for his definitions) with his service, if performing  $def^{floor1} key \triangleright P \text{ in } 0$ .

If still undeterred in his trials, he could walk to the second floor having in mind to try defining  $def^{floor2} key \triangleright P \text{ in } 0$ :

$$\begin{aligned} & hni_H^\tau [floor1_A^\alpha [vw_A^\sigma \square] \mid \\ & vis_V^\omega [in\ floor2.def^{floor2} key \triangleright P \text{ in } 0] \mid \\ & floor2_B^\beta \square \end{aligned}$$

In this case, the operational semantics for the *in* movement raises the error when dynamically checking that:

$$\frac{vis_B^\beta \vdash def^{floor2} key \triangleright P \text{ in } 0}{\beta(vis)(floor2) \ni def(key)}$$

and the same happens if moving to the third floor, upon the condition  $vis \notin C$  at the dynamic checking for *in*.

## 4.6 Related Work, Conclusions and Future Work

There are few direct formal models of context awareness in the presence of mobility. Among these, Birkedal et al. [8] propose a complex model of context awareness able to model both the usual reconfigurations of the context, and queries upon context; noting that context queries cannot be naturally modelled with one bigraphical reactive system (BRS), it proposes a solution (called a Plato-graphical model), such that its expressivity suits well sophisticated real-world context-aware systems. Braione [11] builds contextual reactive systems (CRS) upon reactive systems, RS. The difference between a CRS and a RS is the presence of a function which captures an association between elementary rules and their allowed reaction contexts, so that a CRS can express inhibitor and enabler factors for interaction. Both models are yet to achieve full results in studying behavioural equivalences and proving program properties.

Roman, Julien and Payton [64, 97] build a language-based model of context-aware systems under mobility, an interesting feature of which is the fact that agents have as context the exposed (not private) variables of other agents. The work also has a limited associated proof logic, with program properties being expressed as predicate relations whose validity can be

derived. Kjærsgaard and Bunde-Pedersen’s Conawa calculus [67] models context using several context trees, one for each category of context information (e.g., one for location, one for activities and one for printers). An ambient entity will have a pointer-like presence in one or more trees, with the usual in/out ambient capabilities extended for mobility in multiple contexts.

Furthermore, we found inspiration in work on securing information flow in programming languages, such as Boudol’s typing of information flow [10] in a multi-threaded ML-like language, when declassifying information for legal users (a stronger type system with flow policies, also guarding against termination leaks).

A number of type systems were introduced for Mobile Ambients: Cardelli, Ghelli and Gordon [20], Coppo et al. [32], Gorla, Hennessy and Sassone [45], Bugliesi, Castagna and Crafa [19], among others, introduce message exchange types, the typing of capabilities and actions, type-level groups of ambient names (effectively giving policies for crossing, opening, exchanging messages), and various security policies as membranes at ambient boundaries. Of particular interest is Gorla and Pugliese’s enforcing of security policies via types in  $\mu\text{KLAIM}$  [46] for its fine-grained security features, assigning different privileges to users over different resources in a flat topology of networks.

Our calculus aims at capturing a notion of context awareness in infrastructure-based ubiquitous systems over a well-understood and applicable formalisation such as Mobile Ambients; we also put to use our background in designing protocols for ubiquitous systems (Bucur and Bardram, [14]) to ground this work in practice. Unlike the standard ambient communication scheme, we model context and its communication by exposing and calling named macros across ambient boundaries and policies, extending Zimmer’s [125] flat context communication model. We design for a model of dynamic context with contextual information being macros distributed over varying network scopes; we follow the previous work in the field of designing for security with Mobile Ambients and apply security policies at ambients’ membranes, limiting the capabilities enclosed processes can exhibit, to then type processes in regard to the hierarchy of policies enclosing them. Also, we keep the fine-grained policies on the lines of  $\mu\text{KLAIM}$ , only applied over our hierarchical topology of locations.

Our calculus is applicable for modelling and reasoning upon a fraction of the aspects of context-aware computing, in systems deployed over cell-based, hierarchical topologies. The model can aid the understanding of the workings of context in such mobile systems, and guide the implementation of an added software layer for security. Moreover, we feel that the basic ideas of representing contextual information, its communication and its use are applicable to a greater extent; thus, as part of the future work we intend to focus on modelling context awareness in ad hoc ubiquitous systems.

## 4.7 Appendix

**Lemma 4.1 (Preservation by congruence)** *If  $S$  is well-typed and  $S \equiv S'$ , then  $S'$  is well-typed.*

*Proof.* We prove the preservation of well-typedness through structural congruence by visiting the structural congruence definitions and giving detailed structure to both the left-hand process  $S$  and the right-hand processes  $S'$ , to then derive statements about the inner processes of the left-hand process and use them to prove the right-hand process well-typed.

- $P \equiv P \mid 0$ . If  $P$  is well-typed (meaning  $world_{\mathcal{A}}^{\omega} \vdash P$  and  $\Omega, world_{\mathcal{A}}^{\omega} \vdash \circ P$ ), and given that for  $0$  it holds by definition that  $world_{\mathcal{A}}^{\omega} \vdash 0$  and  $\Omega, world_{\mathcal{A}}^{\omega} \vdash \circ 0$ , we infer:

$$\frac{\frac{world_{\mathcal{A}}^{\omega} \vdash P \quad \Omega, world_{\mathcal{A}}^{\omega} \vdash \circ 0}{world_{\mathcal{A}}^{\omega} \vdash P \mid 0} \quad \frac{\Omega, world_{\mathcal{A}}^{\omega} \vdash \circ P \quad world_{\mathcal{A}}^{\omega} \vdash \circ 0}{\Omega, world_{\mathcal{A}}^{\omega} \vdash \circ P \mid 0}}{P \mid 0 \text{ well-typed}}$$

- $P \mid Q \equiv Q \mid P$ . If  $P \mid Q$  is well-typed, we derive:

$$\frac{\frac{P \mid Q \text{ well-typed}}{world_{\mathcal{A}}^{\omega} \vdash P \mid Q} \quad \frac{\Omega, world_{\mathcal{A}}^{\omega} \vdash \circ P \mid Q}{\Omega, world_{\mathcal{A}}^{\omega} \vdash \circ P \quad \Omega, world_{\mathcal{A}}^{\omega} \vdash \circ Q}}{world_{\mathcal{A}}^{\omega} \vdash P \quad world_{\mathcal{A}}^{\omega} \vdash Q}$$

and, with the results, we infer the required:

$$\frac{\frac{world_{\mathcal{A}}^{\omega} \vdash Q \quad world_{\mathcal{A}}^{\omega} \vdash P}{world_{\mathcal{A}}^{\omega} \vdash Q \mid P} \quad \frac{\Omega, world_{\mathcal{A}}^{\omega} \vdash \circ Q \quad \Omega, world_{\mathcal{A}}^{\omega} \vdash \circ P}{\Omega, world_{\mathcal{A}}^{\omega} \vdash \circ Q \mid P}}{Q \mid P \text{ well-typed}}$$

- $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$ . Essentially as above.
- Cases  $(!F)^{a,\tau} P \equiv (F)^{a,\tau} (!F)^{a,\tau} P$ ,  $E_1 E_2 P \equiv E_2 E_1 P$  and  $EP \mid Q \equiv E(P \mid Q)$  proceed simply, since the well-typedness of a term with floating definition does not take into consideration the position or content of the definitions at all, but only boils down to the well-typedness of  $P$  and  $Q$ .
- $\nu z \nu w P \equiv \nu w \nu z P$ . After ensuring that  $z \neq world$  and then  $w \neq world$ , the left-hand side only means that  $P$  (perhaps  $\alpha$ -converted) is well-typed (note that  $fn(\frac{\omega}{\mathcal{A}}) = \emptyset$ ):

$$\frac{\nu z \nu w P \text{ well-typed}}{z \neq world \quad \frac{world_{\mathcal{A}}^{\omega} \vdash \nu z \nu w P}{w \neq world \quad \frac{world_{\mathcal{A}}^{\omega} \vdash \nu w P}{world_{\mathcal{A}}^{\omega} \vdash P}}}$$

and the same inactively:

$$\frac{\nu z \nu w P \text{ well-typed}}{z \neq \text{world} \frac{\Omega, \text{world}_{\mathcal{A}}^{\omega} \vdash \nu z \nu w P}{w \neq \text{world} \frac{\Omega, \text{world}_{\mathcal{A}}^{\omega} \vdash \nu w P}{\Omega, \text{world}_{\mathcal{A}}^{\omega} \vdash P}}}$$

The inference to prove the right-hand side well-typed is only dissimilar in that the two possible substitutions are inverted.

- Cases  $\nu z (P \mid Q) \equiv P \mid \nu z Q$  after  $z \notin \text{fn}(P)$  and  $\nu z E P \equiv E \nu z P$  if  $z \notin \text{fn}(E)$  proceed essentially on the lines above.
- $\nu z (a_G^{\tau}[P]) \equiv a_G^{\tau}[\nu z P]$  if  $z \notin \text{fn}(a_G^{\tau})$ . As before, from the well-typedness of the left-hand side the derivation for the needed results is:

$$\frac{\nu z (a_G^{\tau}[P]) \text{ well-typed}}{z \neq \text{world} \frac{\text{world}_{\mathcal{A}}^{\omega} \vdash \nu z (a_G^{\tau}[P])}{\frac{\text{world}_{\mathcal{A}}^{\omega} \vdash a_G^{\tau}[P]}{a_G^{\tau} \vdash P}}} \quad z \neq \text{world} \frac{\Omega, \text{world}_{\mathcal{A}}^{\omega} \vdash \nu z (a_G^{\tau}[P])}{\frac{\Omega, \text{world}_{\mathcal{A}}^{\omega} \vdash a_G^{\tau}[P]}{\Omega, a_G^{\tau} \vdash P}}}$$

and, with the results, we infer the required:

$$\frac{z \notin \text{fn}(a_G^{\tau}) \frac{a_G^{\tau} \vdash P}{a_G^{\tau} \vdash \nu z P} \quad z \notin \text{fn}(a_G^{\tau}) \frac{\Omega, a_G^{\tau} \vdash P}{\Omega, a_G^{\tau} \vdash \nu z P}}{a_G^{\tau}[\nu z P] \text{ well-typed}}$$

- $P \equiv P\alpha$ . From a standard substitutivity proof.

□

**Lemma 4.2 (Preservation by context)** *If  $S \rightarrow S'$  and  $S, S'$  are well-typed and  $\mathbf{C}[S]$  is well-typed, then  $\mathbf{C}[S']$  is well-typed.*

*Proof.* We prove the lemma for contexts  $\mathbf{C}$  ranging through all elementary contexts:  $R \mid [\cdot]$ ,  $\text{def}^a D \text{ in } [\cdot]$ ,  $c_K^{\rho}[\cdot]$ ,  $(D)^{a, \tau?}[\cdot]$ .

- $R \mid [\cdot]$ .  $R \mid S$  being well-typed derives statements about  $R$ :

$$\frac{\frac{R \mid S \text{ well-typed}}{\text{world}_{\mathcal{A}}^{\omega} \vdash R \mid S}}{\text{world}_{\mathcal{A}}^{\omega} \vdash R} \quad \frac{\Omega, \text{world}_{\mathcal{A}}^{\omega} \vdash R \mid S}{\Omega, \text{world}_{\mathcal{A}}^{\omega} \vdash R}}$$

which, together with  $S'$  being well-typed, gives  $R \mid S'$  well-typed:

$$\frac{\frac{world_{\mathcal{A}}^{\omega} \vdash R \quad \frac{S' \text{ well-typed}}{world_{\mathcal{A}}^{\omega} \vdash S'}}{world_{\mathcal{A}}^{\omega} \vdash R \mid S'} \quad \frac{\Omega, world_{\mathcal{A}}^{\omega} \vdash_{\circ} R \quad \frac{S' \text{ well-typed}}{\Omega, world_{\mathcal{A}}^{\omega} \vdash_{\circ} S'}}{\Omega, world_{\mathcal{A}}^{\omega} \vdash_{\circ} R \mid S'}}{R \mid S' \text{ well-typed}}$$

- $def^a D \text{ in } [\cdot]$ . Again,  $def^a ?f \triangleright R \text{ in } S$  being well-typed derives statements about  $R$  (even though, if  $a \neq world$ , this can never execute):

$$\frac{\frac{def^a ?f \triangleright R \text{ in } S \text{ well-typed}}{\Omega, world_{\mathcal{A}}^{\omega} \vdash_{\circ} def^a ?f \triangleright R \text{ in } S}}{a_{\mathcal{A}}^{\omega}[R] \text{ well-typed}}$$

which, together with  $S'$  being well-typed, gives  $def^a ?f \triangleright R \text{ in } S'$  well-typed:

$$\frac{\frac{\frac{S' \text{ well-typed}}{world_{\mathcal{A}}^{\omega} \vdash S'}}{world_{\mathcal{A}}^{\omega} \vdash def^a ?f \triangleright R \text{ in } S'} \quad \frac{\frac{S' \text{ well-typed}}{\Omega, world_{\mathcal{A}}^{\omega} \vdash_{\circ} S'} \quad a_{\mathcal{A}}^{\omega}[R] \text{ well-typed}}{\Omega, world_{\mathcal{A}}^{\omega} \vdash_{\circ} def^a ?f \triangleright R \text{ in } S'}}{def^a f \triangleright R \text{ in } S' \text{ well-typed}}$$

- $(D)^{a, \tau?}[\cdot]$ .  $S'$  well-typed gives the required result directly:

$$\frac{\frac{\frac{S' \text{ well-typed}}{world_{\mathcal{A}}^{\omega} \vdash S'}}{world_{\mathcal{A}}^{\omega} \vdash (?f \triangleright R)^{a, \tau?} S'} \quad \frac{\frac{S' \text{ well-typed}}{\Omega, world_{\mathcal{A}}^{\omega} \vdash_{\circ} S'}}{\Omega, world_{\mathcal{A}}^{\omega} \vdash_{\circ} (?f \triangleright R)^{a, \tau?} S'}}{(?f \triangleright R)^{a, \tau?} S' \text{ well-typed}}$$

- $\nu z[\cdot]$ . The statement  $\nu z S'$  well-typed holds immediately from a substituted  $S'$  being well-typed.
- $c^{\rho}[\cdot]$ . We know both hand sides of the derivation  $S \rightarrow S'$  in the hypothesis to be well-typed. As above, from having  $c_K^{\rho}[S]$  well-typed, we infer  $c_K^{\rho}[S']$  well-typed. For this, we range through the  $S \rightarrow S'$  derivation cases and give detailed structure to  $S$  and  $S'$ :

UP. The derivation is either  $a_G^{\tau}[(D)^b P] \rightarrow (D)^b a_G^{\tau}[P]$  or  $a_G^{\tau}[(D)^b P] \rightarrow a_G^{\tau}[(!f \triangleright Q)^{b, \tau} P]$ . We know  $c_K^{\rho}[a_G^{\tau}[(D)^b P]]$  well-typed and prove  $c_K^{\rho}[(D)^b a_G^{\tau}[P]]$  and  $c_K^{\rho}[a_G^{\tau}[(D)^{b, \tau} P]]$  well-typed. We derive that the results in the following hold:

$$\frac{\frac{\frac{c_K^{\rho}[a_G^{\tau}[(D)^b P]] \text{ well-typed}}{c_K^{\rho} \vdash a_G^{\tau}[(D)^b P]}{a \in K \quad \frac{a_{K \cap G}^{\rho \cap \tau} \vdash (D)^b P}{a_{K \cap G}^{\rho \cap \tau} \vdash P}}{\Omega \left[ \{c\} \cup K \rightarrow \frac{\rho}{K} \right], a_{K \cap G}^{\rho \cap \tau} \vdash_{\circ} (D)^b P} \quad \frac{\Omega \left[ \{c\} \cup K \rightarrow \frac{\rho}{K} \right], a_{K \cap G}^{\rho \cap \tau} \vdash_{\circ} P}}{\Omega \left[ \{c\} \cup K \rightarrow \frac{\rho}{K} \right], a_{K \cap G}^{\rho \cap \tau} \vdash_{\circ} P}}$$

With these, we infer:

$$\frac{\frac{a \in K \quad a_{K \cap G}^{\rho \cap \tau} \vdash P}{c_K^\rho \vdash a_G^\tau[P]} \quad \frac{\Omega[\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash P}{\Omega, c_K^\rho \vdash a_G^\tau[P]}}{\frac{c_K^\rho \vdash (D)^b a_G^\tau[P]}{\Omega, c_K^\rho \vdash (D)^b a_G^\tau[P]}} \quad \frac{c_K^\rho[(D)^b a_G^\tau[P]] \text{ well-typed}}$$

The well-typedness of  $c_K^\rho[a_G^\tau[(D)^{b,\tau} P]]$  proceeds simply, since the only difference in this term compared to the left-hand side is the signature  $\tau$ , which is irrelevant when type-checking floating messages not about to be consumed.

DOWN. The derivation is  $(f \triangleright Q)^{b,\sigma} a_G^\tau[P] \longrightarrow a_G^\tau[(f \triangleright Q)^{b,\sigma \cap \tau} P]$ , with a condition which is irrelevant in this case. We know  $c_K^\rho[(f \triangleright Q)^{b,\sigma} a_G^\tau[P]]$  well-typed and prove  $c_K^\rho[a_G^\tau[(f \triangleright Q)^{b,\sigma \cap \tau} P]]$  well-typed. The derivation is similar to the inference in the UP case:

$$\frac{\frac{c_K^\rho[(f \triangleright Q)^{b,\sigma} a_G^\tau[P]] \text{ well-typed}}{c_K^\rho \vdash (f \triangleright Q)^{b,\sigma} a_G^\tau[P]} \quad \frac{\Omega, c_K^\rho \vdash (f \triangleright Q)^{b,\sigma} a_G^\tau[P]}{\Omega, c_K^\rho \vdash a_G^\tau[P]}}{\frac{c_K^\rho \vdash a_G^\tau[P]}{a \in K \quad a_{K \cap G}^{\rho \cap \tau} \vdash P} \quad \frac{\Omega[\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash P}}$$

while the inference is similar to the derivation in the DOWN case:

$$\frac{a \in K \quad \frac{a_{K \cap G}^{\rho \cap \tau} \vdash P}{a_{K \cap G}^{\rho \cap \tau} \vdash (f \triangleright Q)^{b,\sigma} P}}{c_K^\rho \vdash a_G^\tau[(f \triangleright Q)^{b,\sigma} P]} \quad \frac{\frac{\Omega[\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash P}{\Omega[\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash (f \triangleright Q)^{b,\sigma} P}}{\Omega, c_K^\rho \vdash a_G^\tau[(f \triangleright Q)^{b,\sigma} P]}}{c_K^\rho[a_G^\tau[(f \triangleright Q)^{b,\sigma} P]] \text{ well-typed}}$$

DEF. The derivation is  $def^a ?f \triangleright Q \text{ in } P \longrightarrow (?f \triangleright Q)^a P$ . We know  $c_K^\rho[def^a ?f \triangleright Q \text{ in } P]$  well-typed and prove  $c_K^\rho[(?f \triangleright Q)^a P]$  well-typed:

$$\frac{\frac{c_K^\rho[def^a ?f \triangleright Q \text{ in } P] \text{ well-typed}}{c_K^\rho \vdash def^a ?f \triangleright Q \text{ in } P} \quad \frac{\Omega, c_K^\rho \vdash def^a ?f \triangleright Q \text{ in } P}}{c_K^\rho \vdash P} \quad \frac{\Omega, c_K^\rho \vdash P}}{\Omega, c_K^\rho \vdash (?f \triangleright Q)^a P}$$

from which it infers (note that  $Q$  is not checked when in a floating message, hence its well-typedness was not derived above):

$$\frac{\frac{c_K^\rho \vdash P}{c_K^\rho \vdash (?f \triangleright Q)^a P} \quad \frac{\Omega, c_K^\rho \vdash P}{\Omega, c_K^\rho \vdash (?f \triangleright Q)^a P}}{c_K^\rho[(?f \triangleright Q)^a P] \text{ well-typed}}$$

CALL. The derivation is  $(f \triangleright Q)^{a,\tau} f^a \longrightarrow Q$ , with these processes residing in  $c_K^\rho[\cdot]$ . We know  $c_K^\rho[(f \triangleright Q)^{a,\tau} f^a]$  well-typed. From the operational semantics, there exist two ways in which this floating definition could reside at  $c_K^\rho[\cdot]$ :

1.  $c \neq a$ .  $(f \triangleright Q)^{a,\tau}$  has been moving down from an ambient  $a \neq c$  sitting above  $c$ . The final DOWN movement into  $c_K^\rho$  has  $\tau$  being a composition including  $\rho$  and the condition  $c_K^\tau[Q]$  well-typed holding. From these, it holds that  $c_K^\rho[Q]$  well-typed.
2.  $c = a$  with  $\tau = \rho$ .  $(f \triangleright Q)^{c,\rho}$  is native to  $c$ . Thus, there existed an ambient  $b$ , with  $b \in K$ , inner to  $c$ , having made the definition:

$$c_K^\rho[\mathbf{C}[b_H^\sigma[def^c f \triangleright Q \text{ in } P \mid R]] \mid f^c]$$

For the system to have been well-typed in this state, it must have held that  $\Omega, c_K^\rho \vdash \mathbf{C}[b_H^\sigma[def^c f \triangleright Q \text{ in } P \mid R]]$ , hence at the very least, if no other ambient  $c$  is allowed to exist in  $\mathbf{C}$ , the definition above was typed against a context containing  $c \rightarrow \rho_K$ :

$$\Omega [\{c\} \cup K \rightarrow \rho_K] \mathcal{L}, b_{K \cap \dots \cap H}^{\rho \cap \dots \cap \sigma} \vdash def^c f \triangleright Q \text{ in } P$$

in which  $\mathcal{L}$  is a list of updates upon  $\Omega$ . If other ambients  $c$  can exist in  $\mathbf{C}$ , other updates in  $\mathcal{L}$  upon the context value for  $c$  can only be equal to or more restrictive than  $[\{c\} \rightarrow \rho_K]$ . From these, it is at least true that  $c_K^\rho[Q]$  well-typed.

IN. The reduction is  $a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R] \longrightarrow a_G^\tau[P \mid b_H^\sigma[Q \mid R]]$ , with the conditions

$$b \in G \tag{4.1}$$

$$b_G^\tau \vdash Q \mid R \tag{4.2}$$

$$\Omega [\{a\} \cup G \rightarrow \tau_G], b_G^\tau \vdash Q \mid R \tag{4.3}$$

true. We know  $c_K^\rho[a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R]]$  well-typed and prove  $c_K^\rho[a_G^\tau[P \mid b_H^\sigma[Q \mid R]]]$  well-typed. We infer other true statements from the left-hand side being well-typed:

$$\frac{\frac{\frac{c_K^\rho[a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R]] \text{ well-typed}}{c_K^\rho \vdash a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R]}}{c_K^\rho \vdash a_G^\tau[P]} \quad \frac{c_K^\rho \vdash b_H^\sigma[in a.Q \mid R]}{b \in K \quad \frac{b_{K \cap H}^{\rho \cap \sigma} \vdash in a.Q \mid R}{b_{K \cap H}^{\rho \cap \sigma} \vdash Q \mid R}}}{a \in K \quad a_{K \cap G}^{\rho \cap \tau} \vdash P} \quad b \in K \quad \frac{b_{K \cap H}^{\rho \cap \sigma} \vdash in a.Q \mid R}{b_{K \cap H}^{\rho \cap \sigma} \vdash Q \mid R}}$$

and also

$$\frac{\frac{\frac{c_K^\rho[a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R]] \text{ well-typed}}{\Omega, c_K^\rho \vdash a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R]}}{\Omega, c_K^\rho \vdash a_G^\tau[P]} \quad \frac{\Omega, c_K^\rho \vdash b_H^\sigma[in a.Q \mid R]}{\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash in a.Q \mid R}}{\Omega [\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash P} \quad \frac{\Omega, c_K^\rho \vdash b_H^\sigma[in a.Q \mid R]}{\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash in a.Q \mid R}}{\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash Q \mid R}}$$

With these results and equations 4.1 and 4.2, we then derive the well-typedness of the right-hand side:

$$a \in K \frac{\frac{a^{\rho \cap \tau} \vdash P \quad \frac{b \in K \text{ eq. 4.1} \quad \frac{b^{\rho \cap \sigma} \vdash Q \mid R \text{ eq. 4.2}}{b \in K \cap G} \quad \frac{b^{\rho \cap \tau \cap \sigma} \vdash Q \mid R}{b_{K \cap G \cap H}^{\rho \cap \tau \cap \sigma} \vdash Q \mid R}}{a_{K \cap G}^{\rho \cap \tau} \vdash b_H^\sigma [Q \mid R]}}{a_{K \cap G}^{\rho \cap \tau} \vdash P \mid b_H^\sigma [Q \mid R]}}{c_K^\rho \vdash a_G^\tau [P \mid b_H^\sigma [Q \mid R]]}$$

and also

$$\begin{aligned} & \Omega [\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash \circ P \quad (4.4) \\ & \text{eq. 4.4} \quad \frac{\frac{\frac{\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash \circ Q \mid R \text{ eq. 4.3}}{\Omega [\{c\} \cup K \rightarrow \rho_K] [\{a\} \cup (K \cap G) \rightarrow \frac{\rho \cap \tau}{K \cap G}], b_{K \cap G \cap H}^{\rho \cap \tau \cap \sigma} \vdash \circ Q \mid R}}{\Omega [\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash \circ b_H^\sigma [Q \mid R]}}{\Omega [\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash \circ P \mid b_H^\sigma [Q \mid R]}}{\Omega, c_K^\rho \vdash a_G^\tau [P \mid b_H^\sigma [Q \mid R]]} \end{aligned}$$

which, combined, give  $c_K^\rho [a_G^\tau [P \mid b_H^\sigma [Q \mid R]]]$  well-typed.

In the final inactive derivation above, the non-trivial subterm

$$\frac{\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash \circ Q \mid R \text{ eq. 4.3}}{\Omega [\{c\} \cup K \rightarrow \rho_K] [\{a\} \cup (K \cap G) \rightarrow \frac{\rho \cap \tau}{K \cap G}], b_{K \cap G \cap H}^{\rho \cap \tau \cap \sigma} \vdash \circ Q \mid R}$$

is proven by ranging through the possible destinations  $x$  of the definitions  $def^x f \triangleright S$  in  $Q \mid R$  and showing that for any of them, the numerator and denominator perform the same checks upon  $S$ :

- If  $x = b$ , then in the numerator, from  $\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash \circ Q \mid R$  it holds that  $b_{K \cap H}^{\rho \cap \sigma} [S]$  is well-typed, and from equation 4.3 it holds that  $b_G^\tau [S]$  is well-typed, or, combined,  $b_{K \cap G \cap H}^{\rho \cap \tau \cap \sigma} [S]$ , as required for the denominator to hold.
- If  $x \in \{a\} \cup (K \cap G)$ , then we write, first knowing  $a \in K$ , and then by the distributive law in the algebra of sets, that:

$$\{a\} \cup (K \cap G) = (K \cap \{a\}) \cup (K \cap G) = K \cap (\{a\} \cup G).$$

Then, for  $x \in K \cap (\{a\} \cup G)$ , in the numerator it holds that  $x_K^\rho [S]$  is well-typed for  $x \in K$  and  $x_G^\tau [S]$  is well-typed for  $x \in \{a\} \cup G$ , or, combined,  $x_{K \cap G}^{\rho \cap \tau} [S]$  is well-typed, as required for the denominator to hold.

- If  $x \in \{c\} \cup K$  or else, the two terms are identical

OUT. The reduction is  $a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]] \longrightarrow a_G^\tau[P] \mid b_H^\sigma[Q \mid R]$ . We know  $c_K^\rho[a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]]]$  well-typed and prove  $c_K^\rho[a_G^\tau[P] \mid b_H^\sigma[Q \mid R]]$  well-typed. We infer that:

$$\frac{\frac{\frac{c_K^\rho[a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]]] \text{ well-typed}}{c_K^\rho \vdash a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]]}}{a \in K \frac{a_{K \cap G}^{\rho \cap \tau} \vdash P \mid b_H^\sigma[out.Q \mid R]}{a_{K \cap G}^{\rho \cap \tau} \vdash P} \frac{a_{K \cap G}^{\rho \cap \tau} \vdash b_H^\sigma[out.Q \mid R]}{b \in K \frac{b_{K \cap H}^{\rho \cap \sigma} \vdash out.Q \mid R}{b_{K \cap H}^{\rho \cap \sigma} \vdash Q \mid R}}}$$

and also

$$\frac{\frac{\frac{c_K^\rho[a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]]] \text{ well-typed}}{\Omega, c_K^\rho \vdash a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]]}}{\Omega [\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash P \mid b_H^\sigma[out.Q \mid R]}}{\Omega [\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash P} \frac{\frac{\Omega [\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash b_H^\sigma[out.Q \mid R]}{\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash out.Q \mid R}}{\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash Q \mid R}}}$$

With these, we derive:

$$\frac{\frac{a \in K \quad a_{K \cap G}^{\rho \cap \tau} \vdash P}{c_K^\rho \vdash a_G^\tau[P]} \quad \frac{b \in K \quad b_{K \cap H}^{\rho \cap \sigma} \vdash Q \mid R}{c_K^\rho \vdash b_H^\sigma[Q \mid R]}}{c_K^\rho \vdash a_G^\tau[P] \mid b_H^\sigma[Q \mid R]}$$

and also

$$\frac{\frac{\Omega [\{c\} \cup K \rightarrow \rho_K], a_{K \cap G}^{\rho \cap \tau} \vdash P}{\Omega, c_K^\rho \vdash a_G^\tau[P]} \quad \frac{\Omega [\{c\} \cup K \rightarrow \rho_K], b_{K \cap H}^{\rho \cap \sigma} \vdash Q \mid R}{\Omega, c_K^\rho \vdash b_H^\sigma[Q \mid R]}}{\Omega, c_K^\rho \vdash a_G^\tau[P] \mid b_H^\sigma[Q \mid R]}$$

which, combined, give  $c_K^\rho[a_G^\tau[P] \mid b_H^\sigma[Q \mid R]]$  well-typed.

□

**Theorem 4.3 (Subject Reduction)** *If  $S$  is well-typed and  $S \xrightarrow{*} S'$ , then  $S'$  is well-typed.*

*Proof.* We give the standard proof by induction on the length of the derivation  $S \xrightarrow{*} S'$ . The base step proves that if  $S$  reduces to  $S'$  in only one derivation step, then  $S'$  is well-typed; the induction step proves that if  $S$

had reduced to a well-typed  $S''$  in a number of derivation steps, any further derivation step from  $S''$  leads to a well-typed  $S'$ .

*Base step.*  $S$  can have only the single-step derivations according to rules DEF, CALL, UP, DOWN, IN and OUT from the operational semantics.

DEF. A well-typed  $def^a ?f \triangleright Q$  in  $P$  evolves into  $(?f \triangleright Q)^a P$ . From the hypothesis that the left-hand side is well-typed we deduce (guided by the definition for well-typedness and the type rules) statements about its inner processes. Note that when the derivation could yield more statements than depicted, they are not needed for the proof of this case.

$$\frac{\frac{def^a ?f \triangleright Q \text{ in } P \text{ well-typed}}{world_{\mathcal{A}}^{\omega} \vdash def^a ?f \triangleright Q \text{ in } P}}{world_{\mathcal{A}}^{\omega} \vdash P} \quad \frac{\Omega, world_{\mathcal{A}}^{\omega} \vdash def^a ?f \triangleright Q \text{ in } P}{\Omega, world_{\mathcal{A}}^{\omega} \vdash P}}$$

With this final statement, using the well-typedness definition and the type rules, we immediately infer the well-typedness of  $(?f \triangleright Q)^a P$ :

$$\frac{\frac{world_{\mathcal{A}}^{\omega} \vdash P}{world_{\mathcal{A}}^{\omega} \vdash (?f \triangleright Q)^a P} \quad \frac{\Omega, world_{\mathcal{A}}^{\omega} \vdash P}{\Omega, world_{\mathcal{A}}^{\omega} \vdash (?f \triangleright Q)^a P}}{(?f \triangleright Q)^a P \text{ well-typed}}$$

CALL. A well-typed process of the form  $(f \triangleright P)^{a,\tau} f^a$  evolves into  $P$ . Similarly to the CALL case in Lemma 4.2, there exists only one way in which this floating definition could reside at  $world_{\mathcal{A}}^{\omega}[\cdot]$ , that is if  $a = world$  with  $\tau = \omega$ . Thus, there existed an ambient  $b$ , inner to  $world$ , having made the definition:

$$world_{\mathcal{A}}^{\omega}[\mathbf{C}[b_H^{\sigma}[def^{world} f \triangleright Q \text{ in } P \mid R]] \mid f^{world}]$$

For the system to have been well-typed in this state, it must have held that  $world_{\mathcal{A}}^{\omega} \vdash \mathbf{C}[b_H^{\sigma}[def^c f \triangleright Q \text{ in } P \mid R]]$ , hence, since no other ambient  $world$  is allowed to exist in  $\mathbf{C}$ , the definition above was typed against a context  $\Omega$ :

$$\Omega \mathcal{L}, b_{A \cap \dots \cap H}^{\omega \cap \dots \cap \sigma} \vdash def^{world} f \triangleright Q \text{ in } P$$

From this, it holds that  $world_{\mathcal{A}}^{\omega}[Q]$  well-typed, or simply  $Q$  well-typed.

UP. A well-typed  $a_G^{\tau}[(D)^b P]$  evolves nondeterministically into either  $(D)^b a_G^{\tau}[P]$  or  $a_G^{\tau}[(D)^{a,\tau} P]$ . From the well-typedness of the left-hand side the following can be derived:

$$\frac{\frac{a_G^{\tau}[(D)^b P] \text{ well-typed}}{a_G^{\tau} \vdash (D)^b P} \quad \frac{\Omega, a_G^{\tau} \vdash (D)^b P}{\Omega, a_G^{\tau} \vdash P}}{a_G^{\tau} \vdash P} \quad \frac{\Omega, a_G^{\tau} \vdash (D)^b P}{\Omega, a_G^{\tau} \vdash P}$$

Given these statements on  $P$ , we infer the well-typedness of the right-hand sides:

$$\frac{\frac{a_G^\tau \vdash P}{world_{\mathcal{A}}^\omega \vdash a_G^\tau[P]} \quad \frac{\Omega, a_G^\tau \vdash \circ P}{\Omega, world_{\mathcal{A}}^\omega \vdash \circ a_G^\tau[P]}}{\frac{world_{\mathcal{A}}^\omega \vdash (D)^b a_G^\tau[P]}{\Omega, world_{\mathcal{A}}^\omega \vdash \circ (D)^b a_G^\tau[P]}}{(D)^b a_G^\tau[P] \text{ well-typed}}$$

or

$$\frac{\frac{a_G^\tau \vdash P}{a_G^\tau \vdash (D)^{a,\tau} P} \quad \frac{\Omega, a_G^\tau \vdash \circ P}{\Omega, a_G^\tau \vdash \circ (D)^{a,\tau} P}}{a_G^\tau[(D)^{a,\tau} P] \text{ well-typed}}$$

DOWN. A well-typed  $(f \triangleright Q)^{b,\sigma} a^\tau[P]$  evolves into  $a^\tau[(f \triangleright Q)^{b,\sigma \cap \tau} P]$  (the side condition is irrelevant in this case). The left-hand side derivation gives:

$$\frac{\frac{\frac{(f \triangleright Q)^{b,\sigma} a_G^\tau[P] \text{ well-typed}}{world_{\mathcal{A}}^\omega \vdash (f \triangleright Q)^{b,\sigma} a_G^\tau[P]} \quad \frac{\Omega, world_{\mathcal{A}}^\omega \vdash \circ (f \triangleright Q)^{b,\sigma} a_G^\tau[P]}}{world_{\mathcal{A}}^\omega \vdash a_G^\tau[P]} \quad \frac{\Omega, world_{\mathcal{A}}^\omega \vdash \circ a_G^\tau[P]}}{a_G^\tau \vdash P} \quad \frac{\Omega, a_G^\tau \vdash P}}{a_G^\tau[(f \triangleright Q)^{b,\sigma \cap \tau} P] \text{ well-typed}}$$

which helps infer:

$$\frac{\frac{a_G^\tau \vdash P}{a_G^\tau \vdash (f \triangleright Q)^{b,\sigma \cap \tau} P} \quad \frac{\Omega, a_G^\tau \vdash \circ P}{\Omega, a_G^\tau \vdash \circ (f \triangleright Q)^{b,\sigma \cap \tau} P}}{a_G^\tau[(f \triangleright Q)^{b,\sigma \cap \tau} P] \text{ well-typed}}$$

IN. A well-typed  $a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R]$  evolves into  $a_G^\tau[P \mid b_H^\sigma[Q \mid R]]$ , with the conditions  $b \in G$ ,  $b_G^\tau \vdash Q \mid R$  and  $\Omega \{a\} \cup G \rightarrow \tau_G$ ,  $b_G^\tau \vdash \circ Q \mid R$  true. Similarly to the IN case in Lemma 4.2, we infer other true statements from the left-hand side being well-typed:

$$\frac{a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R] \text{ well-typed}}{eq. 4.5 \quad eq. 4.6}$$

in which:

$$\frac{\frac{world_{\mathcal{A}}^\omega \vdash a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R]}{world_{\mathcal{A}}^\omega \vdash a_G^\tau[P]} \quad \frac{world_{\mathcal{A}}^\omega \vdash b_H^\sigma[in a.Q \mid R]}{b_H^\sigma \vdash in a.Q \mid R}}{a_G^\tau \vdash P} \quad \frac{b_H^\sigma \vdash in a.Q \mid R}{b_H^\sigma \vdash Q \mid R}}{(4.5)}$$

and

$$\frac{\frac{\Omega, world_{\mathcal{A}}^\omega \vdash \circ a_G^\tau[P] \mid b_H^\sigma[in a.Q \mid R]}{\Omega, world_{\mathcal{A}}^\omega \vdash \circ a_G^\tau[P]} \quad \frac{\Omega, world_{\mathcal{A}}^\omega \vdash \circ b_H^\sigma[in a.Q \mid R]}{\Omega, b_H^\sigma \vdash \circ in a.Q \mid R}}{\Omega, a_G^\tau \vdash \circ P} \quad \frac{\Omega, b_H^\sigma \vdash \circ in a.Q \mid R}{\Omega, b_H^\sigma \vdash \circ Q \mid R}}{(4.6)}$$

We turn all true statements above into initial steps for the derivation:

$$\frac{\frac{a_G^\tau \vdash P \quad \frac{b \in G \quad b_{G \cap H}^{\tau \cap \sigma} \vdash Q \mid R}{a_G^\tau \vdash b_H^\sigma[Q \mid R]}}{a_G^\tau \vdash P \mid b_H^\sigma[Q \mid R]}}{\frac{\Omega, a_G^\tau \vdash P \quad \frac{\Omega[\{a\} \cup G \rightarrow \tau_G], b_{G \cap H}^{\tau \cap \sigma} \vdash Q \mid R}{\Omega, a_G^\tau \vdash b_H^\sigma[Q \mid R]}}{\Omega, a_G^\tau \vdash P \mid b_H^\sigma[Q \mid R]}}{a_G^\tau[P \mid b_H^\sigma[Q \mid R]] \text{ well-typed}}$$

OUT. A well-typed  $a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]]$  evolves into  $a_G^\tau[P] \mid b_H^\sigma[Q \mid R]$  without side conditions. A simple derivation gives:

$$\frac{\frac{\frac{a_G^\tau \vdash P \mid b_H^\sigma[out.Q \mid R]}{a_G^\tau \vdash P} \quad \frac{a_G^\tau \vdash b_H^\sigma[out.Q \mid R]}{\frac{b_H^\sigma \vdash out.Q \mid R}{b_H^\sigma \vdash Q \mid R}}}{a_G^\tau \vdash P \quad \frac{a_G^\tau \vdash b_H^\sigma[out.Q \mid R]}{\frac{b_H^\sigma \vdash out.Q \mid R}{b_H^\sigma \vdash Q \mid R}}} \quad \frac{\frac{\Omega, a_G^\tau \vdash P \mid b_H^\sigma[out.Q \mid R]}{\Omega, a_G^\tau \vdash P} \quad \frac{\Omega, a_G^\tau \vdash b_H^\sigma[out.Q \mid R]}{\frac{\Omega, b_H^\sigma \vdash out.Q \mid R}{\Omega, b_H^\sigma \vdash Q \mid R}}}{\Omega, a_G^\tau \vdash P \quad \frac{\Omega, a_G^\tau \vdash b_H^\sigma[out.Q \mid R]}{\frac{\Omega, b_H^\sigma \vdash out.Q \mid R}{\Omega, b_H^\sigma \vdash Q \mid R}}}{a_G^\tau[P \mid b_H^\sigma[out.Q \mid R]] \text{ well-typed}}$$

which infers the required results:

$$\frac{\frac{a_G^\tau \vdash P}{world_A^\omega \vdash a_G^\tau[P]} \quad \frac{b_H^\sigma \vdash Q \mid R}{world_A^\omega \vdash b_H^\sigma[Q \mid R]}}{world_A^\omega \vdash a_G^\tau[P] \mid b_H^\sigma[Q \mid R]}$$

and

$$\frac{\frac{\Omega, a_G^\tau \vdash P}{\Omega, world_A^\omega \vdash a_G^\tau[P]} \quad \frac{\Omega, b_H^\sigma \vdash Q \mid R}{\Omega, world_A^\omega \vdash b_H^\sigma[Q \mid R]}}{\Omega, world_A^\omega \vdash a_G^\tau[P] \mid b_H^\sigma[Q \mid R]}$$

which, combined, give the required  $a_G^\tau[P] \mid b_H^\sigma[Q \mid R]$  well-typed.

*Induction step.* Rules STRUCT and CONTEXT from the operational semantics add a further derivation step to a previous derivation sequence.

STRUCT. Assume a derivation sequence has taken place, the last step of which is  $P \rightarrow Q$ , with the hypothesis presumed true (both  $P$  and  $Q$  are well-typed). A new step has true that  $P \equiv P'$  and  $Q \equiv Q'$ , so that (due to the STRUCT semantics rule) it holds that  $P' \rightarrow Q'$ . We prove that  $Q'$  is well-typed if  $P'$  is well-typed, by showing that  $\equiv$  preserves well-typedness (refer to Lemma 4.1).

CONTEXT. As above, we assume  $P \rightarrow Q$  with the hypothesis presumed true (both  $P$  and  $Q$  are well-typed). A new step has a context  $\mathbf{C}$  envelop  $P$  and  $Q$ , so that the new derivation step is  $\mathbf{C}[P] \rightarrow \mathbf{C}[Q]$  (due to the CONTEXT semantics rule). We prove that  $\mathbf{C}[Q]$  is well-typed if  $\mathbf{C}[P]$  is well-typed (refer to Lemma 4.2).  $\square$

**Theorem 4.4 (Type Safety)** *If  $S$  is well-typed then  $S \not\rightarrow^* err$ .*

*Proof.* Suppose, by contradiction, that  $S \rightarrow err$ . We prove that  $S$  cannot even be actively well-typed by induction on the length of the derivation.

*Base step.*  $S$  can exhibit an error in a single derivation step in one of the following cases.

ERR\_DEF. A definition  $def^b f$  issued at  $a$  breaks the local security policy of at least one ambient above and including  $a$ .

If the definition breaks the security policy of  $a$  itself, then  $a$  is not actively well-typed, hence the system  $S$  cannot be actively well-typed. Else, let the ambient with a broken security policy be  $c$ , so that  $S$  is of the form

$$\mathbf{C}_{H_1}^{\sigma_1} \left[ c_K^\rho \left[ \mathbf{C}_{H_2}^{\sigma_2} \left[ a_G^\tau \left[ def^b f \triangleright P \text{ in } Q \mid R \right] \right] \right] \right].$$

This means that  $a_K^\rho \not\vdash def^b f \triangleright P \text{ in } Q$  and hence (among others, by the AMB type rule, applied the necessary number of times)

$$c_K^\rho \not\vdash \mathbf{C}^{\sigma_2} \left[ a_G^\tau \left[ def^b f \triangleright P \text{ in } Q \mid R \right] \right]$$

which gives that the system  $S$  cannot be actively well-typed. The same reasoning goes for ERR\_CALL and ERR\_IN.

*Induction step.*  $S$  can evolve into an error by a sequence of steps, the next of which can be one of the following.

ERR\_STR. We have a previous derivation step  $S' \rightarrow err$ , with  $S'$  not well-typed. Similar to the preservation of well-typedness through congruence (in Lemma 4.1), it is straightforward that a process  $S$  with  $S \equiv S'$  cannot be well-typed.  $\square$

## 4.8 An Ad-Hoc Calculus for Context Awareness

This section briefly presents the dual calculus for context awareness: if the syntax and semantics of the process calculus given in the first part of this chapter catered to *hierarchical* networks of entities, we now port the very same concepts to encode an *ad hoc* network topology, inherent of e.g. large-scale sensor networks, such as the one envisioned for major emergencies in one of our previous papers [14]. The calculus encodes *key concepts* about context awareness in ad hoc networks: the lack of predefined knowledge about the network composition, broadcast communication and the fact that every node could act as a communication router in multihop transmissions.

As described in Section 4.1 regarding our previous calculus, computing *entities* are again modelled by mobile ambients; this time though, among these entities exist only “sibling” connections,  $a[P] \mid b[Q]$ . An entity now moves not in relation to parent ambients, but independently from one location  $l$  to another location  $l'$ , encoded as  $a_l[\text{move } l'.P \mid Q] \longrightarrow a_{l'}[P \mid Q]$ .

*Contextual information* is expressed by named macros  $f(\tilde{x}) \triangleright P$  as before, and *contextual commands* (the basic feature of context-aware applications in Schilit’s [101]) are macro calls  $f\langle\tilde{z}\rangle.P$  (both with parameters,  $\tilde{x}, \tilde{z} \in \mathcal{F}$ , for added expressivity).

The dual procedures of service discovery are, on the other hand, of a completely different flavour than previously; called *context provision* and *context discovery*, the former has an entity disseminate contextual information to enlarge its scope, while the latter locates such provided context for use in the application. Previously an entity’s *context* was the collection of all contextual information public at “ancestor” entities; provision meant the “upward” publishing of macro definitions and discovery was a “downward” move of the same definition.

For ad hoc systems, the key concept is in turn *broadcast* dissemination. We borrow the well-researched techniques for routing (i.e. route discovery) from the ad hoc routing protocols, to design a service discovery protocol, on the basis that in ad hoc networks specifically, service discovery works on the same principles as route discovery (an idea pursued before in [14, 69, 114]). In ad hoc routing, the process of building one’s routing table can be *proactive*: every node in the network maintains updated data about all network entities. For this, they all take initiative in distributing information about their current view of the network. Similarly, service discovery is the proactive broadcast of contextual information:

$$\text{def}^{\odot} D$$

in which all receiving entities will have a definition copy, whether they use it or not.

## Syntax and Semantics

The syntax of the calculus is depicted in Fig. 4.9, in which the standard “choice” operator  $+$  is also added.

networks	$N$	$::=$	$a_l[P]$	node at $l, a \in \mathcal{A}, l \in \mathcal{L}$
			$N \mid N'$	parallel composition
processes	$P$	$::=$	$0$	
			$P \mid P'$	parallel composition
			$P + P'$	choice
			$(\nu z)P$	restriction, $z \in \mathcal{F}$
			$f(\tilde{z}).P$	local macro call, $f, \tilde{z} \in \mathcal{F}$
			$\text{def } D$	local macro definition
			$\text{def}^\odot D$	proactive macro definition
			$E P$	public definitions
			$\text{move } l.P$	movement
definitions	$E$	$::=$	$(D)$	locally floating definition
			$(D)^\odot$	floating definition
	$D$	$::=$	$f(\tilde{x}) \triangleright P$	macro, $\tilde{x} \in \mathcal{F}$
			$!f(\tilde{x}) \triangleright P$	permanent macro

Figure 4.9: The syntax of an ad hoc calculus for context awareness

A system consists of:

- A countable set of locations  $\mathcal{L}$ ; a location is  $l \in \mathcal{L}$ . A countable set of ambients  $\mathcal{A}$ ; an ambient is  $a \in \mathcal{A}$ . A countable set of macro names  $\mathcal{F}$ ; a macro is  $f \in \mathcal{F}$ .
- For each ambient  $a \in \mathcal{A}$ , two directed graphs: a *mobility graph*  $G_{mob}^a = (\mathcal{L}, E_{mob}^a)$  and a *communication graph*  $G_{com}^a = (\mathcal{L}, E_{com}^a)$ .

The mobility graph  $G_{mob}^a$  of vertices the set of locations  $\mathcal{L}$  models the entity’s allowed atomic movements: if the directed edge  $(l, l')$  exists in  $G_{mob}^a$ , then the move in  $a_l[\text{move } l.P]$  can execute. This recognizes two practical facts about mobile entities (e.g. people carrying gadgets): (i) some access is restricted, such as locked doors (which would still open from the inside) or physical obstacles, and (ii) long walks are a sequence of short walks of the form  $\text{move } l.\text{move } l'.P$ , of which the short walks are modelled edges in the graph.

Similarly, the communication graph  $G_{com}^a$ —also over vertices from  $\mathcal{L}$ —models the possible singlehop transmissions for a specific entity  $a$  (naturally, the ability to reach farther locations depends on the specific hardware; this graph needs not be directional).

$$\begin{array}{c}
 \text{MOVE} \quad \frac{(l, l') \in G_{mob}^a}{a_l[\text{move } l'.P \mid Q] \longrightarrow a_{l'}[P \mid Q]} \\
 \\
 \text{DEF} \quad \text{def } D \longrightarrow (D)0 \quad \text{def}^\circ D \longrightarrow (D)^\circ 0 \\
 \\
 \text{DEFBCAST} \quad \frac{(l, l') \in G_{com}^a}{a_l[(D)^\circ P] \mid \prod_{b, l'} b_{l'}[Q] \longrightarrow a_l[P] \mid \prod_{b, l'} b_{l'}[(D)Q]} \\
 \\
 \text{CALL} \quad (f(\tilde{x}) \triangleright Q) f\langle \tilde{z} \rangle.P \longrightarrow \{\tilde{z}/\tilde{x}\}Q \mid P
 \end{array}$$

Figure 4.10: Operational semantics

Thus, Fig. 4.10 gives the semantics of the calculus. We omit the complete rules of structural congruence, as they are similar to the ones in Section 4.2, Table 4.1. We do remind that a rule of structural congruence spawns one-shot definitions out of permanent ones:  $(!F) \equiv (F)(!F)$ , where  $F$  is of the form  $f(\tilde{x}) \triangleright P$ . Also, definitions do not “float” out of ambients by structural congruence.

Rule MOVE is straightforward. The multiple rules for definitions and calls work as follows: the DEF rule  $\text{def } D \longrightarrow (D)0$  is the process “locally” defining a macro  $D$ . If the same ambient calls that particular macro name (i.e. needs exactly that service), then rule CALL executes and “consumes” the definition.

On another hand, an entity can broadcast a definitions through the network (a proactive type of context distribution, as mentioned). For this, the DEF rule  $\text{def}^\circ D \longrightarrow (D)^\circ 0$  creates a “floating” definition, which is distributed out of the origin entity by rule DEFBCAST. By this, a subset of the nodes within communication distance of the broadcasting node  $a$  receive a copy of the definition. Again, any of the receiving nodes can then consume the definition with a CALL.

### Expressivity

**Encoding  $!P$**  As a parenthesis, banged processes  $!P$  are easily encoded on the basis of banged definitions,  $def !h \triangleright Q$ :

$$\llbracket !P \rrbracket \triangleq (\nu h) (def !h \triangleright P \mid h) \mid h$$

**Encoding Multihop Broadcasts** More importantly, rule DEFBCAST of semantics in Fig. 4.10 only employs singlehop broadcasts.

Based on the semantics for broadcasting a definition over a single hop, we encode multihop broadcasts, written  $(D)^{\odot n}$ ,  $n \in \mathbb{N}$ . We assume a unique macro name  $fwd$ , known and permanently called (using the encoding for banged processes above) by all agents in the network. The fact that all agents run  $! fwd$  means to say that all agents agree to be a forwarders of definitions (and get a copy of any forwarded definition). The encoding is then as follows:

$$\begin{aligned} \llbracket (D)^{\odot 1} \rrbracket &\triangleq (D)^{\odot} \\ \llbracket (D)^{\odot n} \rrbracket &\triangleq ( fwd \triangleright (D)(D)^{\odot n-1} 0 )^{\odot} \quad \text{for } n > 1 \end{aligned}$$

For example, have a 2-hop definition broadcast, with  $(l, l') \in G_{com}^a$ :

$$\begin{aligned} &a_l[\llbracket (D)^{\odot 2} \rrbracket P \mid b_{l'}[! fwd \mid Q]] \\ &\triangleq a_l[( fwd \triangleright (D)(D)^{\odot 0} )^{\odot} P \mid b_{l'}[! fwd \mid Q]] \\ &\longrightarrow a_l[P \mid b_{l'}[( fwd \triangleright (D)(D)^{\odot 0} )(! fwd \mid Q)]] \\ &\equiv \longrightarrow a_l[P \mid b_{l'}[(D)(D)^{\odot 0} ! fwd \mid Q]] \\ &\equiv a_l[P \mid b_{l'}[(D)(D)^{\odot} ! fwd \mid Q]] \end{aligned}$$

from which  $b$  proceeds with the singlehop broadcast  $(D)^{\odot}$  as usual.



# Chapter 5

## GammaSense: Positioning using Background Radioactivity

The material presented in this chapter reproduces the paper *GammaSense: Infrastructureless Positioning using Background Radioactivity*, under publication as:

- [17] Doina Bucur and Mikkel Baun Kjærgaard. GammaSense: Infrastructureless Positioning using Background Radioactivity. In *Proceedings of The Third IEEE European Conference on Smart Sensing and Context (EuroSSC)*. Springer-Verlag, Oct 2008.



# GammaSense: Infrastructureless Positioning using Background Radioactivity

Doina Bucur and Mikkel Baun Kjærgaard

## Abstract

We introduce the harvesting of natural background radioactivity for positioning. Using a standard Geiger-Müller counter as sensor, we fingerprint the natural levels of gamma radiation with the aim of then roughly pinpointing the position of a client in terms of interfloor, intrafloor, and indoor versus outdoor locations. We find that the performance of a machine-learning algorithm in detecting position varies with the building, and is highest for interfloor detection in the case of an old domestic house, while it is highest for intrafloor detection if the floor spans building segments made from different construction materials.

## 5.1 Introduction

Positioning is an important requirement for many novel applications. However, technologies for positioning often depend on extensive infrastructures, which limit the coverage of these technologies and create breakdowns in the user experience when a user crosses the—often invisible—infrastructure boundaries. One approach for solving this problem is the development of positioning technologies with pervasive coverage which minimizes the dependence on infrastructure.

Prior work on positioning has used the properties of physical phenomena such as hearable sound, ultrasound, and types of non-ionizing radiation (e.g. visible light and radio signals). In this work we consider the use of ionizing radiation (specifically, products of radioactivity: alpha, beta and gamma rays) for positioning. In our environment, the sources of such radiation include natural radioactivity in the soil and building materials, and cosmic rays.

In practice, gamma radiation is most relevant, since alpha and beta radiation have subcentimeter propagation range. The radiation appears naturally in the environment, and the geometry of its sources in a built environment gives more interesting variation patterns than the patterns of other radiation sources, such as visible light.

Harvesting gamma radiation to infer location has several advantages: (i) Gamma radiation is pervasive, therefore coverage is not confined to an area of infrastructure coverage; (ii) The geometry of radiation sources is strikingly different from that of other radiation, which makes gamma readings a desirable sensor input in e. g. sensor fusion; (iii) There exist many devices that sense gamma radiation, and the threat of terror might make them even more widespread. Doing positioning over gamma radiation also has disadvantages: (i) Radiation has no identifier embedded in the signal, so there exists only one signal source to use in location estimation. This is the same drawback as when using visible light. (ii) Compared to other signals, a longer sampling time is needed for detecting relevant statistical properties of received radiation patterns.

We make the following contributions: (i) We show that gamma radiation is a predictable signal for positioning (ii) We analyze which environment properties gamma radiation depends on (iii) We present the first positioning system based on background radioactivity named *GammaSense*, which positions a device using location fingerprinting over gamma readings.

The remainder of this paper is structured as follows: In Section 5.2, we present the relevant related work. Subsequently, we give a primer to gamma radiation and discuss it as a signal source for positioning in Section 5.3. Then the novel *GammaSense* positioning system is introduced in Section 5.4. Evaluation results for *GammaSense* are provided in Section 5.5. Finally, Section 5.6 concludes the paper and provides directions for future work.

## 5.2 Related Work

Our system examines the performance of doing localization on the basis of measuring the levels of background radioactivity (gamma rays) indoors, and employing the technique of location fingerprinting. While—to our knowledge—no other work does infrastructureless localization with background radioactivity, a wealth of existing research does explore the matter of doing localization indoors, based on a variety of technologies and types of infrastructure, such as infrared, ultrasonic and ultra-wideband, and their specific emitters and detectors.

The common drawback of these systems is their reliance on custom infrastructure, a fact which then diminishes their acceptance and easiness of deployment. *GammaSense* takes the very opposite approach and looks into making use of natural signals for indoor positioning, offering a degree of location detection completely independent of any infrastructure.

*GammaSense* uses the technique of *location fingerprinting*, which has already been applied in related work with radio waves, light and sound signals. It is based on the acquiring of a database of prerecorded signal

measurements, denoted as location fingerprints. Clients' locations are then estimated by comparing them with the database of fingerprints [68].

One of the first location fingerprinting systems was the radio-based RADAR [4] system, which applied different deterministic mathematical models to calculate the coordinates of a client's position from WaveLan/IEEE 802.11 signal strengths. Similar methods were applied to GSM signals by Otsason et al. [88]. Unlike RADAR, later systems employed probabilistic models instead of deterministic ones. An example of a probabilistic system which calculates a client's coordinates was published by Youssef et al. [124]; a similar system determining the logical position or cell of a client was published by Haerberlen et al. [50]. A radio-based system named SkyFloor [112] focused on predicting the floor of a client.

The principle of location fingerprinting was also applied to sound signals by Patel et al. [90]; their system uses the electric wiring in a building to generate sound signals on several frequencies, which form distinctive sound patterns throughout the building. A tone detector then picks up these sound signals and uses them as location fingerprints. A positioning system based on location fingerprinting over visible light intensities was also proposed by Ravi et al. [94].

## 5.3 Gamma Radiation

### 5.3.1 A Primer on Radioactivity and Ionizing Radiation

Radioactivity is the natural phenomenon in which certain—possibly artificial—chemical elements emit radiation spontaneously, be it in the form of electromagnetic waves or of charged particles. The cause of this emission is radioactive decay, i.e. the spontaneous transmutation of an unstable parent element into a more stable daughter element; the decay rate is practically expressed with the term "half-life", meaning the span of time required for half the quantity of the radioactive element to transmute.

One form of radioactive decay is the beta decay. A neutron or a proton transform into the other within the parent nucleus, accompanied by the emission of an electron (or its positively charged version, a positron); this electron is called a *beta particle* or beta ray. If a neutron transmutes into a proton, a negatively charged electron  $e^-$  with high kinetic energy (a  $\beta^-$  particle) is expelled from the nucleus. The alternative transmutation with a positron emission (a  $\beta^+$  particle) cannot occur without energy input, because the mass of a neutron is higher than that of a proton. Hence,  $\beta^+$  rays are mostly produced artificially in particle accelerators ([62]).

Another produce of radioactive decay are *gamma rays*, an electromagnetic radiation having the highest frequency and the shortest wavelength within the electromagnetic spectrum. Neutrons and protons occupy well-

defined energy levels in a nucleus, and when either particle is excited to a higher unoccupied level, the excited nucleus decays to a lower energy state, and the difference in energy is emitted as gamma radiation. This energy difference is much larger (in the range of MeV) than in the case of excited electrons, whose similar state mutations release visible, near-visible light or X-rays (with an energy of a few eV).

Beta particles typically have energies from a few KeV to a few MeV and the mass of an electron (atomic mass 1/1836). The range of beta particles is short: a 1.9mm sheet of aluminium stops a 1MeV beta particle ([98]). Because they are relatively light, beta particles do not travel in straight lines but follow a random path through material. Gamma rays are high-energy (0.1 to 3 MeV). Their range is long and penetration power high; their typical means of losing energy is by ejecting an electron from an atom and being scattered from the impact with reduced energy (the Compton effect). To reduce the intensity of a 0.5MeV gamma ray to 0.37 of its initial value, one needs to lay a shield of 4cm of aluminium, 0.59cm of lead or 28.6cm of tissue paper ([98]).

Beta and gamma radiation also accompany *alpha decay* (another omnipresent radioactive decay in which heavy nuclei disintegrate by emitting a positively charged nucleus of helium,  ${}^4_2\text{He}^{2+}$ , with an even shorter range than beta particles, due to their large atomic mass of 4). Many alpha sources are accompanied by beta-emitting radiodaughters, and alpha emission is followed by gamma emission from the remaining negatively charged nucleus.

All three forms of radiation produced by radioactive decay can ionize atoms or molecules in their path (either due to their electric charge, or to their kinetic energy), transforming them into ions by adding or removing charged particles. Hence, along with other rays, they are collectively called *ionizing radiation*.

### 5.3.2 Natural Sources of Radioactivity

Background radiation is omnipresent, and has always existed naturally. Its sources are *cosmic rays* (from outer space and the Sun) and *terrestrial radioactivity* naturally occurring in soil, building materials and in air, water, foods and the human body (as in Table 5.1, after [110]).

Out of the sources of radiation in Table 5.1, the ionizing component of the cosmic radiation, the radioactive elements in soil and construction materials ( ${}^{238}\text{U}$ ,  ${}^{232}\text{Th}$  series and  ${}^{40}\text{K}$ , in approximately equal contributions), and the aerial radioactive gases radon (and, in small concentrations, thoron) are all measurable sources of beta and gamma rays (either directly or indirectly, as a side effect of alpha decay).

The specific concentrations of radioactive elements in soil are related to the types of rock from which the soils originate, which in turn correlate with

Table 5.1: Average worldwide exposure to natural air and soil radiation sources

Source of radiation	Percentage
Cosmic radiation total	18.48
of which ionizing (beta and gamma)	13.74
Terrestrial radiation total	82.46
soil, building materials: $^{238}\text{U}$ , $^{232}\text{Th}$ (alpha), $^{40}\text{K}$ (beta)	22.74
air: radon $^{222}\text{Rn}$ , thoron $^{220}\text{Rn}$ (alpha)	59.24

the concentrations in air. The gas radon (a decay product of radium, with a half-life of 3.8 days) diffuses out of the soil. Radon and its decay products are the most important contributors to human exposure to radiation from natural sources.

Indoor concentration of gamma rays, mainly determined by the materials of construction, is inherently greater than outdoor exposure if earth materials have been used; the geometry of the radiation source changes indoors from a half-space to a surrounding configuration. The indoor to outdoor ratios range from 0.6 to 2.3, with a worldwide ratio of 1.4 ([110]).

### 5.3.3 Indoor Variations of Radiation Concentrations

Some of the radioactive sources are fairly constant and uniform geographically, while others vary widely with location. Naturally, cosmic rays are less intense at lower altitudes, and concentrations of uranium and thorium are elevated in certain soils. More interestingly, the building materials of houses and the design and ventilation systems strongly influence indoor levels of the most important contributors, radon and its decay products.

Advection from the soil is the main factor for high radon entry rates in buildings (as in Table 5.2, after [40, 110]). Radon is driven indoors by the pressure differential between the building and the ground around the foundation, produced by the higher indoor temperatures, ventilation, and to some degree by wind blowing on the building. The effectiveness of this pressure differential is dependent on the permeabilities of the building foundation and the adjacent earth. Also, wind can cause decreases in radon entry concentrations by its flushing effect on radon in soil surrounding the house. Because of differences in the pressure differentials and permeabilities, advection varies greatly from structure to structure, especially in temperate and cold climates. The non-masonry building in Table 5.2 has less accumulation of radon than the masonry one, due to a smaller radiation contribution from walls and ceilings.

Various surveys also find radon concentration variations between rooms

Table 5.2: Representative radon entry rates in low-level residential houses: a masonry house and a wooden house in Finland [110]

Source of radon	Mechanism	Rate ( $Bq/m^3h$ ) and percentage	
		Masonry house	Wooden house
Building elements			
Walls and ceiling	Diffusion	16 (18)	2 (3)
Subjacent earth			
Through gaps	Advection	66 (73)	60 (86)
Through slab	Diffusion	4 (4)	4 (6)
Outdoor air	Infiltration	3 (3)	3 (4)
Water supply	De-emanation	1 (1)	1 (1)
Total		90 (100)	70 (100)

in the same building. Ghany [44] and Sonkawade et al. [103] find that the mean values of radon concentration in bathrooms and kitchens were significantly higher than those in living rooms and bedrooms. The find is motivated by ceramic being a radon source, poor ventilation and the use of underground water and natural gas.

### 5.3.4 Experimental Results

Given the representative surveys in Subsections 5.3.2 and 5.3.3 upon the variations of radioactivity levels indoors, we expect to be able to harvest indoor radiation levels to use in localization. Two facts about the variability of the radiation levels are important when designing a localization algorithms based on fingerprinting: one is the geometric variability of the measurements within the building, and the other is the consistency of the signal levels at any given location. We give a brief account of our findings in the following.

Our experiments did confirm the expected indoor geometry of the radioactivity source. In one of our testbed buildings (a two-level domestic house, as seen later in Subsection 5.5.1), both the mean and the variance of radioactivity readings differ visibly between the two rooms on different floors, as in Fig. 5.1. The mean over a 1-hour-long continuous length of 1-minute counts shows a 10.32% decrease from the ground-floor room (mean 30.79 counts per minute) to the first-floor room (mean 27.61 counts per minute), and a striking difference in standard deviations (a 26.43% decrease, from 5.75 on level zero to 4.23 on the first level). This fact verifies that the major radiation source indoors is the subjacent earth, with its influence being diminished and smoothed with rising levels.

To confirm the stability of the signal, we then refer to Fig. 5.2, which depicts the typical variation in readings in a fixed position over a 1-day-

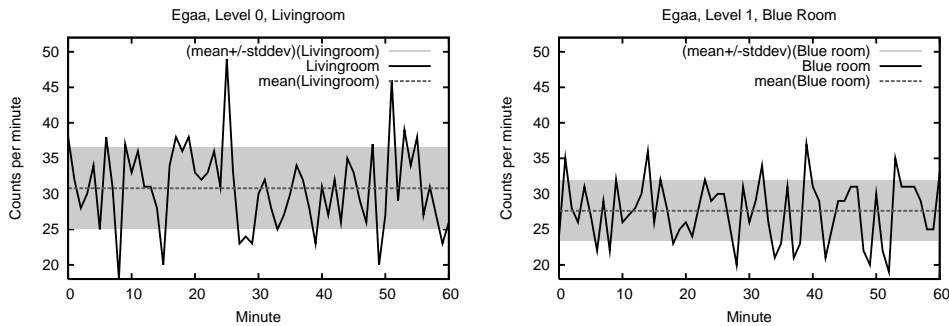


Figure 5.1: The difference in radiation measurements over 1-hour periods between rooms on different floors in a domestic house in Egaa, Denmark

long period of time. In another of our testbeds (a public institution, more on which in Subsection 5.5.1), subjected to a fair level of human activity during a working day, the signal doesn't exhibit significant variation.

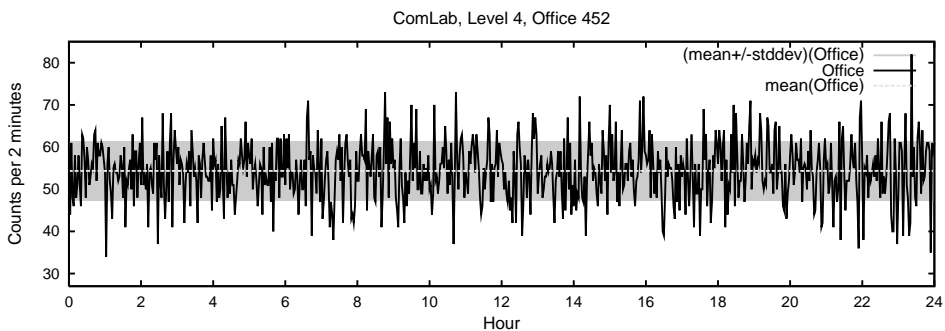


Figure 5.2: Signal stability over a 24-hour period

## 5.4 GammaSense

The sensor we used for measuring radiation levels was a radioactivity meter composed of two devices: a commercially available Geiger-Müller tube and a pulse ratemeter (i. e. counter, either battery- or AC-powered) designed to feed the Geiger-Müller tube with voltage and to handle the pulses delivered by the tube. Both were manufactured by a small local company called *Impo electronic*, for use in education. The setup and the tube's technical specification are depicted in Fig. 5.3.

A Geiger-Müller tube is one of the several radioactivity detectors whose functionality is based on the ionizing capability of the radiation produced through radioactive decay. The tube is filled with a mixture of inert gases, in

which incoming ionizing radiation creates electrons and positively charged ions. The tube wall constitutes the negative electrode or cathode, while a thin central wire is highly charged with positive voltage and is an anode. The strong electric field created by the electrodes accelerates the ion towards the cathode and the electron towards the anode, giving them sufficient energy to ionize further gas molecules through collisions, thus creating an avalanche of charged particles.

The result is a short, intense pulse of current which passes from the negative electrode to the positive electrode and is counted as one ionizing particle. The counter also includes an audio amplifier that produces a beep upon pulse detection. The number of pulses per time unit measures the intensity of the radiation field. Cheap and robust, a Geiger-Müller tube can detect the intensity of radiation (particle frequency), but not particle energy.



(a) Geiger-Müller tube (left) and pulse ratemeter with data log and serial interface (right)

Sensitive to:	<i>beta, gamma rays.</i>
Effective diameter:	27.8mm.
Window material:	<i>mica.</i>
Window thickness:	$[2 \frac{mg}{cm^2}, 3 \frac{mg}{cm^2}]$ .
Gas filling:	<i>neon, argon, halogen.</i>
Plateau:	[450V, 700V].
Recommended voltage:	575V.
Dead time at 575V:	19 $\mu$ S.

(b) The tube's technical specifications. The *plateau* is the voltage interval where counts/sec is least dependent of voltage.

Figure 5.3: The experimental setup: Geiger-Müller tube and counter with technical specifications

We performed all measurements maintaining the anode voltage strictly at the recommended, mid-plateau voltage of 575V (as described in Fig. 5.3(b)), for best independence of counts from voltage supplied, both when powering the device from battery and when plugged into the power supply infrastructure. The counter allows the configuration of a small number of counting parameters, such as different count times (between 1 and 120 seconds) after which the total number of particles is reported, and has the ability to count and report continuously for such subsequent fixed periods.

The results are either reported on the counter's small screen, or sent through the counter's RS-232 interface to a laptop, stored in a volatile memory (which only holds 50 counts), or stored in a non-volatile datalog (with a capacity of 250 counts). In our experiments, we used either the sending

of each count in real time to the laptop, or the downloading of the contents of the datalog—when full—to the laptop; in both cases, the readings were saved in the ratemeter’s format in raw data files.

On the laptop side of the RS-232 interface, we used a Linux Debian with *minicom*, a text-based, GPL-licensed terminal emulator for Unix-based operating systems.

For implementing location fingerprinting using gamma readings we used the machine-learning tool Weka [123]. Weka implements a range of machine-learning algorithms and several GUIs for configuration, experimentation, and visualization. Before selecting a machine-learning algorithm to use, we experimented with different algorithms, and concluded that the *LogitBoost* algorithm was most fit. LogitBoost is based on additive logistic regression, which means that the algorithm—given a simpler algorithm—iteratively constructs a better detector by combining several instances of the simple algorithm. We used a decision stump (i.e. a simple boolean classifier) as the simple algorithm.

In order to feed our data to Weka, we implemented a small Java program that reads the raw data files and outputs ARFF data for the Weka tool. The program also calculates certain features from the raw gamma readings: aggregated 60-second counts, and the mean and variance of the latest five 60-second counts. These features are then written to the ARFF file together with ground truth about whether the data was collected indoor or outdoor, on which floor, and at which horizontal location.

## 5.5 Evaluation

### 5.5.1 Data Collection

We collected the location fingerprints we used in our indoor localization study over an 8-month period ending in June 2008. For the study, we chose four testbed buildings with diverse construction parameters (i. e. number of floors, building materials and age of construction) located in two countries. We give a superficial visual impression of the four testbed buildings in Fig. 5.4, where we also list the identifiers by which we refer to the testbeds in the rest of the paper.

Their construction parameters are then recorded in Table 5.3, in terms of the year the building was finished, the type of construction materials and the number of building levels. For building Egaa, 1829 is the building year, while 1960 is the year when three of the perimeter walls have been renewed. Testbed ComLab is an extensive building composed of a number of smaller constructions, linked together: four old Victorian houses built before 1901 formed the southern wing, to which a identically-styled northern wing was added by 1993, for then a fully-modern segment to be attached by 2006.



(a) an old village house in Egaa, Denmark (Egaa)



(b) the Computing Laboratory at the University of Oxford, UK (ComLab)



(c) the Mathematics Department at the University of Aarhus, Denmark (Math)



(d) the Computer Science Department at the University of Aarhus, Denmark (CS)

Figure 5.4: The testbed buildings, ordered by age of construction; the building identifier is listed in parentheses

Using the setup in Fig. 5.3 detailed in Section 5.4, we collected fingerprints of the background radioactivity at fixed height for any particular building. For Egaa, we took sample counts in each of the four rooms for one hour, while for CS and Math we fixed one and three locations, respectively, on each floor, and took sample counts for 10 to 20 minutes. In ComLab, due to the complexity of the building, we divided the study cases in two: a vertical study aiming at distinguishing among floors collected 20-minutes worth of samples from the same vertical location on each level, while for the horizontal study we took measurements on the fourth floor at six locations divided equally among the three wings of the building.

For an additional indoor-versus-outdoor study, in the case of ComLab we also collected samples from two outdoor locations in the very vicinity of the testbed building. Also, to verify the stability of the signal, we collected long-term counts: one over 5 days in a fixed location in a CS office, and one over 2 days in a ComLab office.

All sample measurements were collected as continuous counts, 10-second

Table 5.3: The testbeds' characteristics

Building	Built by	Building materials	Floors	Use
Egaa	1829, 1960	Stone, brick, wood, straw	2	domestic
ComLab	1901, 1993, 2006	Brick, concrete	5	institution
Math	1967	Brick, concrete, stone, wood	5	institution
CS	2001	Concrete, steel, glass	4	institution

(in the majority of experiments), 1-minute or 2-minute-long; the continuous taking of short counts allowed us to aggregate these short counts into longer counts, as needed for the study.

### 5.5.2 Accuracy

The accuracy of GammaSense was evaluated by emulation on the collected data set. The technique of emulation tests the machine-learning algorithms in an environment emulated by the data set, for the ability to distinguish among indoor horizontal locations, indoor vertical locations and indoor versus outdoor. The emulations were run in the Weka tool (as described in Section 5.4) using fivefold cross-validation for splitting up the data sets into training and test data for the machine-learning algorithms. The features used in the emulations were 60-second gamma counts, and the mean and variance of the last five 60-second counts.

The performance of the localization algorithm is judged in the rest of this section based on quantitative measures, i.e. the *detection accuracy*, the *confusion matrix* and the *kappa statistic*. The detection accuracy is the percentage of correctly classified tests. A confusion matrix shows how many instances have been assigned to each class; the matrix elements show the percentage of test examples whose actual class (i.e. the actual location) is the row and whose predicted class (i.e. the predicted location) is the column.

The kappa statistic quantifies how much a detector is an improvement over a random detector. It can be thought of as the chance-corrected prediction agreement, and possible values range from +1 (perfect agreement between prediction and reality) via 0 (no agreement above that expected by chance) to -1 (complete disagreement).

#### Horizontal

The aim for the indoor horizontal localization was to distinguish among different building segments based on their different gamma patterns. The ComLab building consists of three wings built by 1901, 1993 and 2006, respectively, with the oldest two wings highly similar in style and material. We grouped our ComLab data into three classes, one for each building part. The

emulation for this three-class recognition problem gave a detection accuracy of 67.7% and a kappa statistic of 0.51.

Further analysis of the prediction errors revealed that they were not distributed evenly between the three classes, as shown by the confusion matrix in Table 5.4(b). The highest confusion rates were between the 1901 and 1993 wings, a fact we explain by a high similarity in building materials and style, despite the different ages. Furthermore, if the data is regrouped into a two-class problem for distinguishing between the 1901/1993 and 2006 wings, the accuracy increases to 81.7% and the kappa statistic to 0.60, as summarized in Table 5.4(a). Hence, localization accuracy depends on similarity of construction. We conclude that GammaSense distinguishes between building segments of different construction parameters with moderate performance.

Table 5.4: ComLab horizontal results.

(a) Detection results for localization among the three wings (first row). The same if the oldest two wings are aggregated into a single wing (second row).

	Accuracy	Kappa
ComLab, 3 wings	67.7%	0.51
ComLab, 2 wings	81.7%	0.60

(b) Confusion matrix for the 3-wing study. Elements show the percentage of tests whose row is the actual wing and whose column is the predicted wing.

	1901	1993	2006
1901	15.6%	9.7%	1.1%
1993	8.1%	23.1%	7.5%
2006	1.1%	4.8%	29.0%

## Vertical

For the indoor vertical localization, the target was floor detection. The Egaa, ComLab, Math, and CS data sets were used in this evaluation, and the emulation for these buildings gave the results in Table 5.5(a). Overall, the Egaa domestic house allowed us a high degree of floor prediction agreement, while this decreased for the three public institutions.

From the confusion matrix for the vertical ComLab study (Table 5.5(b)) one identifies that one particular reason for the poor accuracy is the fact that the fourth and basement floors are comparable in gamma counts. While high counts were expected for the basement floor, one explanation for the high counts of the top floor is the fact that either the roof of the building is highly radioactive, or that the poor building ventilation has the radioactive gases accumulate under the roof. The Math building does not suffer from this issue and exhibits higher accuracy and kappa statistic. CS gave the poorest results for the kappa statistic, possibly because the CS building is new, the

shielding between floors and from the subjacent earth is intact, and there exists an active ventilation system.

We then conclude that GammaSense distinguishes among floors in a building, yet there exists a set of parameters which decrease detection accuracy, such as roof accumulation of radon or good ventilation.

Table 5.5: Indoor vertical results.

(a) Results for each testbed quantified by detection accuracy and kappa statistic.			(b) Confusion matrix for ComLab vertical. Elements show the percentage of tests whose row is the actual floor and column the predicted floor.				
	Accuracy	Kappa		0	2	3	4
Egaa	72.2%	0.45	0	10.5%	1.3%	0.0%	9.2%
ComLab	43.4%	0.29	2	0.0%	5.3%	6.6%	9.2%
Math	48.6%	0.35	3	0.0%	5.3%	9.2%	6.6%
CS	49.2%	0.23	4	9.2%	0.0%	0.0%	11.8%

### Indoor versus Outdoor

An additional aim for GammaSense was to detect indoor versus outdoor locations. Indoor and outdoor data from ComLab was used in the evaluation; the outdoor data was collected at two locations: one in the atrium (a small open yard in the core of the building), and the other on the lawn in front. The emulation results were:

accuracy: 91.7% and kappa statistic: 0.55.

We hypothesize that the better "ventilation" outdoor, either in the absence of building materials (which is the case for the lawn location) or even in their presence (the case for the inner atrium, surrounded by walls and paved) resulted in lower counts outdoors.

### Sensitivity Analysis

As a further analysis, we looked into more detail over the localization accuracy of the GammaSense algorithm, in the ComLab horizontal study for detecting among the three building segments, which gave the results on the first row of Table 5.4(a).

Specifically, we tested the variation of the detection accuracy and the kappa statistic as a function of the window size (i.e. the number of the

last 60-second measurements whose mean and variance are given, together with the current reading, to the localization algorithm; the value for all the results reported above was five). The variation is reported in Fig. 5.5. As expected, since radiation counts are fairly unstable in value from one count to the other (as clear in Fig. 5.1 and 5.2), localization accuracy improves by considering more measurements per location; up to a number of eight minute-counts, performance parameters increase.

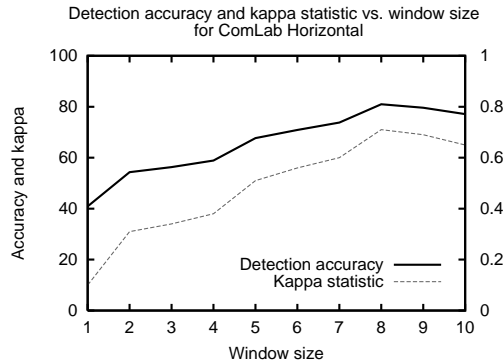


Figure 5.5: The variation of detection accuracy and kappa statistic with window size (i.e. number of 1-minute measurements taken at each location)

## 5.6 Conclusions and Future Work

We investigated the performance of a machine-learning algorithm to pinpoint roughly the interfloor, intrafloor and indoor-versus-outdoor position of a client sensing background radioactivity levels indoors. We found that the accuracy results vary widely with the structure of the fingerprinted building, and report best performance for interfloor detection in a domestic house (a 72.2% detection accuracy) and intrafloor detection in a complex building made up of segments with different construction parameters (a 67.7% accuracy in wing detection). Also, we verify that the indoor-versus-outdoor detection for outdoor locations in the vicinity of the building has good performance (91.7%). Finally, we look into the variation of performance parameters with the window size (i.e. the number of minutes a client has to measure radiation levels at a location to have it detected) and report that performance increases with increasing window size, up to 8 minute-counts.

While we argue that harvesting natural parameters such as background radiation for location detection is worthy of research due to ease of deployment and a complete independence of infrastructure, we recognize that such techniques offer less performance and control than more standard, infrastructure-based techniques for localization. We state that—due to the

geometry of its sources indoors—background radioactivity is a desirable signal for use in location estimation, especially one based exclusively on natural signals such as natural light, sound or the chemical components in the air (a potential future work).

The Geiger-Müller tube and counter that we used in our experiments are bulky for real-world deployments; however, miniaturized versions already exist—for instance, such sensors were integrated into mobile phones for early detection and localization of radioactive threats [113].



# Chapter 6

## Verifying ANSI C Context-Aware Applications

This chapter reproduces the report *Verifying ANSI C Context-Aware Applications*, which presents the initial step in a project aiming at the automatic verification of context-aware code:

- [12] Doina Bucur. Verifying ANSI-C Context-Aware Applications. Published online at <http://www.daimi.au.dk/~doina>. Working paper. 2008.



## Verifying ANSI-C Context-Aware Applications

### Abstract

We report on work in progress upon the verification of context-aware applications written in C-based languages. We recognize that context-aware programs are generally either middleware-based and multithreaded, or driven by asynchronous events, and focus on identifying the program points in which the contextual updates impact the application behaviour. Inheriting from related work on the validation of context-aware applications, we implement a technique for detecting context-aware program points over SATABS, a tool for generic verification of specifications of ANSI C/C++ programs. We then briefly review possible refinements and future work.

### 6.1 Introduction

An extreme distribution of services makes ubiquitous computing an application area in which the behaviour of programs changes with both local and networked events. An added degree of unscheduled device mobility and failure makes this stream of events unpredictable. The *context* of an application is then any such environmental computational information which is relevant to the application behaviour, and an application is *context-aware* if it features actions triggered by context or automatic reconfiguration upon context [1, 25, 101].

Examples of computational context to an application include sensed input from local sensing hardware on a sensor platform, the presence—announced by a hello packet—of another networked device in the near neighbourhood of a router, the contents of a person’s schedule as retrieved from a server in an office environment, or the current human activity undertaken at a location.

Developing context-aware applications to work under such distributed and unpredictable execution environments has specific difficulties. On one hand, the unscheduled nature of contextual updates incoming to a program fueled the development of *asynchronous* languages. Such languages specialize in managing interaction in I/O-heavy systems: an asynchronous function call doesn’t start execution at call time, but is instead appended to a task

queue, from which a dispatcher pops one task at a time and runs it to completion. Such deferred execution is the basis for event-driven programming languages, in which tasks are callback code prompted into the dispatcher's queue by an input event, and whose execution is delayed until the system is idle. Recent languages centered around a support for asynchrony [43, 66] include nesC, a language for writing deeply networked systems and the programming platform for the sensor network operating system TinyOS [106].

On another hand, the opposite approach at tackling the complexity of contextual updates is the development of *middleware*-based applications, in which a specialized middleware software layer takes over the non-trivial process of discovery, acquisition and interpretation of context updates, thus leaving a lesser degree of awareness to the application. A middleware can employ the same asynchronous schemes for managing context updates, with the application then being a standard multithreaded program. A variety of proposed middleware software for adapting to context, of various design concepts and functionality, range from the classic ContextToolkit [35] to refinements such as JCAF [6].

The purpose of this research is to provide verification mechanisms which ensure the dependability of applications adapting to context; these applications can be fully aware asynchronous programs, or multithreaded code supported by a middleware layer, as described above. The fundamental difficulty in verifying such adaptive programs is the fact that context updates affect the behaviour of the application involuntarily, at any time during execution: the handlers of asynchronous events preempt the running of the main program or that of any task, or the middleware sprouts threads running context handlers at undetermined times. This can happen for any type of input, but it is particular for context inputs, which are delivered streamingly (i.e. sensors feed readings to the application periodically, and all new members of the network neighbourhood broadcast hello messages).

The resulting programming difficulty resides in the need for developers to identify when the streams of context updates impact the behaviour of the application. Since contextual events are asynchronous, contextual changes themselves interfere.

Given the current wealth of development and deployment of such context-adaptive systems, and adding these distinctive features of programming for awareness compared to general programming, we note that specialized techniques for the *verification* of context-aware applications are lacking. We build on the few existing specialized techniques for the *validation* of such programs ([78, 108, 115] described in Section 6.2 on related work) and proceed to improve the context awareness of an existing verification suite (SAT-Based Predicate Abstraction for ANSI-C, SATABS [29]), by identifying the program points in which the data flow of a certain source of context updates influences application behaviour (called context-aware program points, or

*capps*).

We then show the directions for future work, which include a refinement upon the techniques for identifying *capps*, with analyses borrowed from the field of secure information flow [10]. We then state that the final goal is the verification of context-aware programs; for this, we are looking into writing specifications of such programs and automatically generating assertions to be verified with the counterexample-based abstraction and refinement (CEGAR) loop employed by SATABS.

For all purposes stated above, we settled on programs written in C-like languages; our case studies include nesC applications for sensor networks and C++ industrial programs. Since the SATABS suite takes as input ANSI-C programs, we employ translators for flattening nesC and C++ to C, and techniques for handling the remaining features of these languages which are not present in C (e.g. the modeling of nesC's scheduling scheme).

## 6.2 Related Work

Given the problem of analysing the influence that an external contextual input (e.g. variable `ctx` given as an argument to the context's handler, as in Fig. 6.1) has upon a multithreaded or event-based program, we settle on identifying the set of context-aware program points. This set is composed of program statements which (i) have the side-effect of leaking a measure of the current value of `ctx` to another variable, such as an assignment `power=ctx` where `power` is a global variable, or (ii) have an effect which varies with the current value of `ctx` or a leaked-upon `power`, such as the expression `if(power) E1 else E2`.

The process of identifying (ii) statements, only for the case in which the leaked-upon variable influencing the statement is strictly a global, is related to the problem of identifying data racing points. However, given the fact that the external contextual input starts always as a variable local to a function (be it an event handler or a thread's main function), all other program points can only be identified by more sophisticated techniques related to information flow.

In the remaining of this section we review related research, either on the verification of our typical case studies for race conditions, or on the validation of context awareness. To start with, data races are a source of errors in concurrent programs, in a similar way to that of data interference upon program behaviour. Race detection is a safety verification problem for concurrent programs: a race occurs when two threads can access (i.e. read or write) a global variable simultaneously, and at least one of the accesses is a write.

Henzinger et al. [53] recognizes that, while existing race checkers fall into

two categories (dynamic, lockset-based tools and static, type-based tools), programmers write code using less obvious synchronization schemes such as those locking access based on the value of a variable. Such synchronization schemes cause false positives with standard static analysis tools, and the authors develop a more precise analysis of access, tracking the values of variables.

Their tool is implemented in the C model checker BLAST [54], and was run over nesC applications. NesC employs specific synchronization idioms: (i) tasks are nonpreemptible, and as such there can be no races on variables accessed only in tasks, but only between events and events and tasks (ii) there are `atomic` sections. The nesC compiler itself implements a flow-based alias static analysis to catch race conditions on shared variables from event handlers, and reports false positives at certain synchronization methods, as described above. The tool models away the system hosting a nesC application as an arbitrary number of threads, each executing a while-loop triggering hardware interrupts (if interrupts are enabled by a certain global variable) or tasks (if the execution goes idle) nondeterministically, as well as the nesC's task queue and split-phase synchronization.

Moving closer to the point of dealing with contextual input instead of generic concurrent programs, Wang, Elbaum and Rosenblum's [115] work on automated generation of tests of context awareness for Java programs sets the tone for specialized techniques for context awareness. It improves the test suite of a context-aware Java application by identifying program points where context changes affect program behaviour, for then to construct sequences of such points and only select the ones which satisfy certain coverage adequacy criteria.

Their capp identifier takes as inputs the program source and the context object, passed to the application from the middleware as the argument to an instance of the context handler function, running in a newly sprouted thread. The capps are detected by side-effect analysis [75] and escape analysis [92] over Java code, and the outcome is a set of multi-function control flow graphs in which nodes are annotated with capp identifiers.

All subsequent related work moves away from real application code, towards modeling. Other validation techniques [78, 108] stem from either metamorphic testing or data-flow analysis, also trying to measure the comprehensiveness of capp sequences as test sets. They formalize the notion of *context* as a pair of context variable and value, that of *situation* (residing at the middleware level) as a triggering condition over some contexts, which starts an action (residing at the application level) when the trigger holds.

In their middleware-centric model program, context variables are standard data variables enhanced with the ability of being updated from the environment, context-aware program points are computed as those expressions in which a context variable is read or written, either in a situation or in

an action. Based on the above, the program is modeled as a context-aware flow graph: essentially, the standard control flow graphs of situations and actions are enhanced with "phantom" nodes modeling the potential environmental update of context variables. Then, a set of test-suite adequacy criteria are settled, based on the relationship between the context-aware program points and their (situation or action) location.

Other verification schemes for context awareness are based on logic [16, 97]; as expected, they trade practicability for the ability of verifying strong specifications (such as the fulfilling of distributed security firewalls upon types of contexts [16]) over networks of mobile, context-aware entities and mobile code.

## 6.3 Context-Aware Program Points in SATABS

We proceed to add the capp identifier techniques (in Subsection 6.3.2), after first reviewing the existing program analysis and verification features in SATABS (Subsection 6.3.1).

### 6.3.1 SATABS and goto-cc

SATABS is a generic model checking tool targeted at the software industry. Its practicality stems from the fact that, unlike usual model checkers, which take as input programs written in model languages, SATABS analyzes ANSIC programs, and supports a large subset of the language's features (e.g. complex data types, pointer arithmetics, function pointers, dynamic memory and, partially, the `pthread`-library concurrency [72]). Consequently, support for C++ is an expected (and work in progress) extension, while support for C-like languages such as nesC [43] depends only on the existence of a translator from nesC to C<sup>1</sup>.

While this effectively removes the need for developing models of programs as input for the model checker, given the size of typical input applications, the problem of the verification scalability becomes central. For this, SATABS computes an *abstract model* [29] of the input code (now denoted as *concrete model*) automatically: such an abstraction preserves the code's original control flow, but replaces a subset of the original program variables with chosen boolean variables and leaves all others nondetermined. This is a conservative process (i.e., any properties upon variables which are true in

---

<sup>1</sup>`ncc`, the nesC compiler used in building TinyOS, does create a platform-dependent, inlined, non-ANSI `.c` program out of the given nesC application plus the set of TinyOS components the application is wired with. This is then compiled into machine code and deployed on sensors. The effort required to translate this application code into ANSIC is relatively little; nonetheless, in the long term we are aiming towards a fully-automatic, component-based translator from nesC to ANSIC.

the original program remain true in the abstracted program), hence properties of the input code can be proven true based on a small abstract program, by a standard model checker. On the other hand, if the property is false over the abstract program, the counterexample returned by the model checker is checked upon the concrete model; if not spurious, the property is proven false, but otherwise the counterexample-guided abstraction refinement (or CEGAR) technique [29] drives the construction of an increasingly complex abstraction.

Crucially, in the preverification stage, SATABS manipulates the input code for ease of analysis: the rich ANSI C language is translated into a simplified C-like language called *goto-cc*, in which all control flow statements are rewritten as guarded `gotos`, function calls are inlined, and complex statements are broken into simple ones. Over these, SATABS implements preliminary static analyses, such as pointer analysis (i.e. the memory field a pointer points to at a given program statement). The capp identifier itself, presented next, is also a collection of static analyses extending SATABS at this preverification stage.

### 6.3.2 Identifying Capps

To identify the context-aware program points (capps) in a program taking unscheduled environmental events, Wang, Elbaum and Rosenblum’s [115] Java capp identifier employs two known static analysis techniques (side-effect and escape analyses, as in [75, 92]), each giving out a subset of their capp set. We borrowed this idea, and as an initial step extended SATABS [29] with these capp-identifier techniques.

The Java *side-effect analysis* procedure from Le et al. [75]—translated for use over ANSI-C—computes, for every statement  $s$  in the program, sets  $read(s)$  and  $write(s)$  containing all variables (or fields of variables of type structure) read and written by statement  $s$ . This is based on a preliminary pointer analysis, which retrieves the variable a pointer points to, in any given program state; also, the read and write effects of function calls need to be propagated back to their call statement. All variable names are uniquely identified by SATABS based on their scope: `update_demo::ctx` and `update_power::ctx` are a pair of identically named variables `ctx`, arguments of different functions; this extends to same-name functions in different source files.

After the read and write sets are computed for all statements, the adapted escape analysis in [92] defines an approximated interference dependence in a concurrent environment as in Definition 6.1:

**Definition 6.1** *An interference dependence in a concurrent program is a pair of program points  $m$  and  $n$  in distinct threads  $t_m$  and  $t_n$  such that a (field of a) variable is written at  $m$  and read at  $n$ .*

In what follows, we keep Wang, Elbaum and Rosenblum’s [115] case study on TourApp, a middleware-based application running demos on mobile devices at an exhibition, translated into C using the `pthread` library (Fig. 6.1), as a base example.

---

```

unsigned char power;
unsigned char demo;

void update_power(ctxt *power_ctx)
{
    power = power_ctx->value;
}

void update_demo(ctxt *demo_ctx)
{
    if(demo_ctx->value == 1 && power > 30)
        demo = 1;
    else if(demo_ctx->value == 2 && power > 30)
        demo = 2;
}

void* context_handler(void *arg)
{
    if((ctxt *)arg)->type == TYPE_POWER)
        update_power(arg);
    else if((ctxt *)arg)->type == TYPE_DEMO)
        update_demo(arg);
    return NULL;
}

```

---

(a) application

(b) middleware

Figure 6.1: The [115] case study on TourApp, translated into C using the `pthread` library

When applied to the capp identification problem, the techniques above translate into a static analysis which proceeds as follows:

- It starts by identifying the particular program variables which hold the values of incoming environmental contexts; it then adds these to an empty set of variables "tainted" by contextual input,  $tainted_0$ ; in the example code in Fig. 6.1, the middleware passes a pointer to `ctx` of type `ctxt` to the `context_handler(void *arg)` function, which will run as a new thread at application level. Hence,  $tainted_0 = \text{middleware} :: 1 :: \text{ctx}$  and a  $tainted(s)$  is also calculated for each program statement.
- The side effects (i.e. read and write sets) and tainted set of program

statements are calculated for all statements; for all initial thread statements  $s$ ,  $tainted(s) = tainted_0$ , and:

- if at any statement  $s$ ,  $read(s)$  intersects  $tainted(s)$ , then  $s$  is a context-aware program point, or capp;
- for all non-initial statements  $s$ ,  $tainted(s)$  is inherited from the preceding statement in the program flow;
- if the above happens, and additionally  $write(s)$  is not empty, then  $write(s)$  is added to  $tainted(s')$ , for all statements  $s'$  flowing directly after  $s$ .

The capps thus identified are program points at which the program statement accesses the contextual variable `ctx`, or of any other program variable which was assigned an expression containing `ctx`, even indirectly, such as `power` in `power = power_ctx->value` from Fig. 6.1(a). As in [115], for the sample code in Fig. 6.1, the capps identified are depicted in Fig. 6.2, in which the goto-cc statements highlighted in gray are the context-aware program points. These capps serve as the reading program points (at location  $n$ ) in Definition 6.1 for interference dependences.

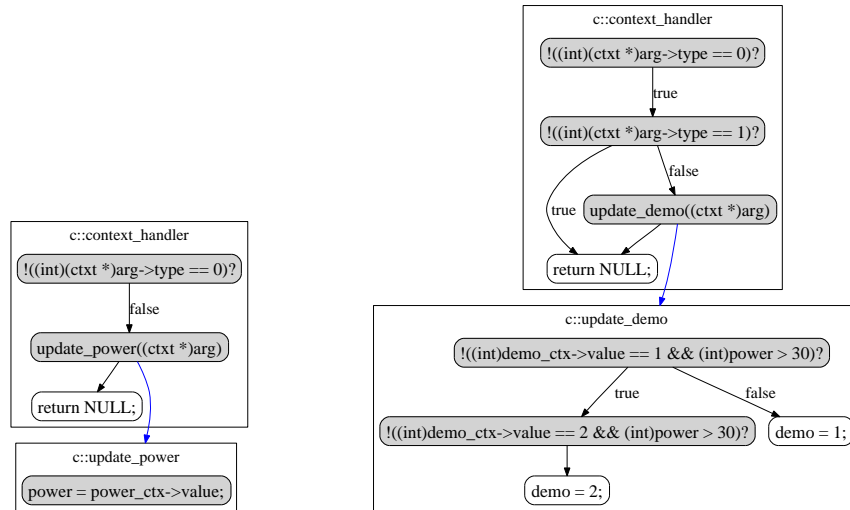


Figure 6.2: Context-aware program points (goto-cc statements in gray) as detected by side-effect analysis in SATABS

### 6.3.3 Typing the Information Flow

The side-effect static analysis in Subsection 6.3.2 has limited performance: take the following fragment from Fig. 6.1:

```
if(demo.ctx->value == 1 && power > 30)
  demo = 1;
else if(demo.ctx->value == 2 && power > 30)
  demo = 2;
```

in which the program flow is controlled based, indirectly, on reading the contextual variable `ctx`. Variable `demo` is then written differently in the resulting two program flows, which effectively allows any subsequent access of `demo` to gain a degree of access upon `ctx`. Thus, we recognize that the technique of detecting side effects from Wang, Elbaum and Rosenblum's [115] overlooks the side effects of a more sophisticated nature.

As future work, we thus plan to add to the capp identifier techniques from formal static analyses in the field of information-flow security, such as Boudol's [10], which comprehensively calculates the flow of information from a confidential location of memory (in our case, a contextual variable) in a multithreaded ML-like toy language.

**Future Work. Verification for Context-Aware Programs** Also as future (and concluding) work for this contribution, we are looking into (i) writing specifications for context-aware applications, and (ii) automatically generating SATABS-style assertions from these specifications, to be verified by the tool's existing generic counterexample-based abstraction and refinement procedure.

## Acknowledgments

This work has developed during the author's stay at the Oxford University Computing Laboratory in early 2008, under the advising of Professor Marta Kwiatkowska; the author wishes to express her gratitude to Marta for the fruitful discussions and the opportunity to be a part of her research group on Quantitative Analysis and Verification.



# Bibliography

- [1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a Better Understanding of Context and Context-Awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, 1999.
- [2] Adam Greenfield. *Everyware: The Dawning Age of Ubiquitous Computing*. New Riders Publishing, March 2006.
- [3] O. Aziz, B. Lo, Guang-Zhong Yang, R. King, and A. Darzi. Pervasive Body Sensor Network: An Approach to Monitoring the Post-operative Surgical Patient. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks, 2006 (BSN 2006)*, pages 13–18. IEEE Press, 2006.
- [4] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, volume 2, pages 775–784, 2000.
- [5] Jakob E. Bardram. Activity-Based Computing: Support for Mobility and Collaboration in Ubiquitous Computing. *Personal and Ubiquitous Computing*, 9(5):312–322, July 2005.
- [6] Jakob E. Bardram. The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In Hans Gellersen, Roy Want, and Albrecht Schmidt, editors, *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, volume 3468 of *Lecture Notes in Computer Science*, pages 98–115. Springer Verlag, May 2005.
- [7] Jakob E. Bardram and Thomas R. Hansen. The AWARE architecture: supporting context-mediated social awareness in mobile cooperation. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 192–201. ACM Press, 2004.

- [8] L. Birkedal, S. Debois, E. Elsborg, T. Hildebrandt, and H. Niss. Bi-graphical Models of Context-aware Systems. In *Foundations of Software Science and Computation Structures (FOSSACS) 2006*, number 3921 in Lecture Notes in Computer Science, pages 187–201, March 2006.
- [9] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The Role of Trust Management in Distributed Systems Security. *Secure Internet programming: security issues for mobile and distributed objects*, pages 185–210, 1999.
- [10] Gerard Boudol. On Typing Information Flow. *Lecture Notes in Computer Science (LNCS)*, 3722:366–380, 2005.
- [11] Pietro Braione. Operational Congruences for Contextual Reactive Systems. Technical Report 2004.33, DEI, Politecnico di Milano.
- [12] Doina Bucur. Verifying ANSI-C Context-Aware Applications. Published online at <http://www.daimi.au.dk/~doina>. Working paper, 2008.
- [13] Doina Bucur and Jakob E. Bardram. Resource Discovery in Activity-Based Sensor Networks. In *Proceedings of the First International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–10, 2006.
- [14] Doina Bucur and Jakob E. Bardram. Resource Discovery in Activity-Based Sensor Networks. *Mobile Networks and Applications (MONET)*, 12(2-3):129–142, 2007.
- [15] Doina Bucur and Mogens Nielsen. A Calculus for Ad Hoc Context Awareness. Published online at <http://www.daimi.au.dk/~doina>. Working paper, 2008.
- [16] Doina Bucur and Mogens Nielsen. Secure Data Flow in a Calculus for Context Awareness. In *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 439–456. Springer, 2008.
- [17] Doina Bucur and Mikkel Baun Kjærgaard. GammaSense: Infrastructureless Positioning using Background Radioactivity. In *Proceedings of the Third European Conference on Smart Sensing and Context (EuroSSC)*, 2008.
- [18] Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Boxed Ambients. In *TACS '01: Proceedings of the 4th International Symposium*

- on Theoretical Aspects of Computer Software*, pages 38–63. Springer-Verlag, 2001.
- [19] Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Reasoning about Security in Mobile Ambients. *Lecture Notes in Computer Science*, 2154:102–120, 2001.
- [20] Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Types for the Ambient Calculus. *Inf. Comput.*, 177(2):160–194, 2002.
- [21] Luca Cardelli and Andrew D. Gordon. Mobile Ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98*. Springer-Verlag, Berlin Germany, 1998.
- [22] UCLA Center for Embedded Networked Sensing. CENS: Center for Embedded Networked Sensing, 2008. <http://research.cens.ucla.edu/>; accessed August 3, 2008.
- [23] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin. GSD: A Novel Group-based Service Discovery Protocol for MANETs. *Mobile and Wireless Communication Networks, Fourth International Workshop*, pages 140–144, 2002.
- [24] Dan Chalmers, Matthew Chalmers, Jon Crowcroft, Marta Kwiatkowska, Robin Milner, Eamonn O’Neill, Tom Rodden, Vladimiro Sassone, and Morris Sloman. Ubiquitous Computing Grand Challenge: Manifesto, February 2006. <http://www-dse.doc.ic.ac.uk/Projects/UbiNet/GC/Manifesto/manifesto.pdf>; version 4, 23-2-06; accessed August 1, 2008.
- [25] Guanling Chen and David Kotz. A Survey of Context-Aware Mobile Computing Research. Dartmouth Computer Science Technical Report TR2000-381, 2000.
- [26] Liang Cheng. Service Advertisement and Discovery in Mobile Ad hoc Networks. *Computer Supported Cooperative Work workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments*, 2002.
- [27] Stuart Cheshire. Zero Configuration Networking (Zeroconf). <http://www.zeroconf.org/>; accessed August 22, 2008.
- [28] Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. The Role of Connectivity in Supporting Context-Sensitive Applications. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 193–207. Springer-Verlag, 1999.

- [29] Edmund Clarke, Daniel Kroening, Natasha Sharygina, and Karen Yorav. SATABS: SAT-based Predicate Abstraction for ANSI-C. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2005)*, volume 3440 of *Lecture Notes in Computer Science*, pages 570–574. Springer Verlag, 2005.
- [30] UK Computing Research Committee. Grand Challenges in Computing Research, 2008. [http://www.ukcrc.org.uk/grand\\_challenges/index.cfm](http://www.ukcrc.org.uk/grand_challenges/index.cfm); accessed August 2, 2008.
- [31] MIT Media Lab: Things That Think Consortium. Things That Think, 2008. <http://ttt.media.mit.edu/>; accessed August 2, 2008.
- [32] Mario Coppo, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Ivano Salvo.  $M^3$ : Mobility Types for Mobile Processes in Mobile Ambients. *Electronic Notes in Theoretical Computer Science*, 78, 2002.
- [33] Sony Corporation. Sony Global – FeliCa, 2008. <http://www.sony.net/Products/felica/abt/dvs.html>; accessed August 13, 2008.
- [34] Anind K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5:4–7, 2001.
- [35] Anind K. Dey, Daniel Salber, and Gregory D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2–4):97–166, 2001.
- [36] Philip K. Dick. *Ubik*. Gollancz S.F. (1 Aug 2006), 1969.
- [37] Division of Engineering and Applied Sciences of Harvard University. CodeBlue: Wireless Sensor Networks for Medical Care. <http://www.eecs.harvard.edu/mdw/proj/codeblue/>; accessed on August 28, 2008.
- [38] Ekahau. Ekahau Real-Time Location System (RTLS), 2008. <http://www.ekahau.com/?id=4200>; accessed August 18, 2008.
- [39] E. Elnahrawy, Xiaoyan Li, and R. P. Martin. The limits of localization using signal strength: a comparative study. In *Proceedings of IEEE Sensor and Ad Hoc Communications and Networks (SECON 2004)*, pages 406–414, 2004.
- [40] European Commission. Commission Recommendation of 21 February 1990 on the protection of the public against indoor exposure to radon. <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31990H0143:EN:NOT>; accessed June 13, 2008, February 1990.

- [41] Request for Comments 2608. Service Location Protocol Version 2. <http://www.openslp.org/doc/rfc/rfc2608.txt>.
- [42] UPnP Forum. PnP Device Architecture. <http://www.upnp.org/>.
- [43] David Gay, Philip Levis, and Robert von Behren. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI)*, 2003.
- [44] H. A. A. Ghany. Variability of Radon Levels in Different Rooms of Egyptian Dwellings. *Indoor and Built Environment*, 15(2):193–196, 2006.
- [45] D. Gorla, M. Hennessy, and V. Sassone. Security policies as membranes in systems for global computing. *Logical Methods in Computer Science*, 1 (1:3):1–23, 2005.
- [46] D. Gorla and R. Pugliese. Enforcing Security Policies via Types. *LNCS Security in Pervasive Computing*, 2802:86–100, 2004.
- [47] U.S. Government. Global Positioning System. <http://www.gps.gov/>; accessed August 20, 2008.
- [48] L. Grajales and I. V. Nicolaescu. Wearable Multisensor Heart Rate Monitor. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks, 2006 (BSN 2006)*, pages 154–157. IEEE Press, 2006.
- [49] Zygmunt J. Haas, Marc R. Pearlm, and Prince Samar. The Zone Routing Protocol (ZRP) for Ad hoc Networks. *IETF MANET Internet Draft*, 2002.
- [50] Andreas Haeberlen, Eliot Flannery, Andrew M. Ladd, Algis Rudys, Dan S. Wallach, and Lydia E. Kavvaki. Practical Robust Localization over Large-Scale 802.11 Wireless Networks. In *Proceedings of the Tenth ACM International Conference on Mobile Computing and Networking*, pages 70–84, 2004.
- [51] S. Helal, N. Desai, V. Verma, and C. Lee. Konark - A Service Discovery and Delivery Protocol for Ad-Hoc Networks. *Wireless Communications and Networking Conference*, 3:2107–2113, 2003.
- [52] A. Helmy, S. Garg, N. Nahata, and P. Pamu. CARD: A Contact-based Architecture for Resource Discovery in Wireless Ad Hoc Networks. *Springer Mobile Networks and Applications*, 10:99–113, 2005.

- [53] Thomas A. Henzinger, Ranjit Jhala, and Rupak Majumdar. Race checking by context inference. In *PLDI*, pages 1–13, 2004.
- [54] Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre. Lazy abstraction. In *POPL*, pages 58–70, 2002.
- [55] Tom Henzinger. Challenges in Embedded Systems Design. In *Proceedings of the discussion meeting From computers to ubiquitous computing, by 2020, March 17–18, 2008*. The Royal Society.
- [56] T. Hester, R. Hughes, D. Sherrill, B. Knorr, M. Akay, and P. Bonato. Using Wearable Sensors to Measure Motor Abilities Following Stroke. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks, 2006 (BSN 2006)*, pages 5–8. IEEE Press, 2006.
- [57] Jeffrey Hightower and Gaetano Borriello. A Survey and Taxonomy of Location Systems for Ubiquitous Computing, Technical Report UW-CSE 01-08-03. Technical report, University of Washington, Computer Science and Engineering, 2001.
- [58] Jeffrey Hightower and Gaetano Borriello. Location Systems for Ubiquitous Computing. *Computer*, 34(8):57–66, 2001.
- [59] CBS Studios Inc. StarTrek.com: Technology, 2007. <http://www.startrek.com/startrek/view/library/technology/>; accessed August 9, 2008.
- [60] Palo Alto Research Center Incorporated. PARC (Palo Alto Research Center, Inc.). <http://www.parc.com/>; accessed July 31, 2008.
- [61] Palo Alto Research Center Incorporated. Innovation Milestones, 2007. <http://www.parc.com/about/history/default.html>; accessed July 31, 2008.
- [62] Ivan G. Draganić and Zorica D. Draganić and Jean-Pierre Adloff. *Radiation and Radioactivity on Earth and Beyond, Second Edition*. CRC Press, 1993.
- [63] A.R. Jimenez, F. Seco, R. Ceres, and L. Calderon. Absolute Localization using Active Beacons: A survey and IAI-CSIC contributions. Technical report, Institute for Industrial Automation, CSIC Madrid, 2004.
- [64] Christine Julien, Jamie Payton, and Gruia-Catalin Roman. Reasoning About Context-Awareness in the Presence of Mobility. *Electr. Notes Theor. Comput. Sci.*, 97:259–276, 2004.

- [65] Omer Sinan Kaya. NesCT: A language translator. <http://nesct.sourceforge.net/>.
- [66] Charles Edwin Killian, James W. Anderson, Ryan Braud, Ranjit Jhala, and Amin M. Vahdat. Mace: Language Support for Building Distributed Systems. In *PLDI '07: Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, pages 179–188. ACM, 2007.
- [67] Mikkel B. Kjærgaard and Jonathan Bunde-Pedersen. Towards a Formal Model of Context Awareness. In *First International Workshop on Combining Theory and Systems Building in Pervasive Computing 2006 (CTSB 2006)*, 2006.
- [68] Mikkel Baun Kjærgaard. A Taxonomy for Radio Location Fingerprinting. In *Proceedings of the Third International Symposium on Location- and Context-Awareness*, pages 139–156, 2007.
- [69] R. Koodli and C. E. Perkins. Service Discovery in On-Demand Ad hoc Networks. *IETF MANET Internet Draft*, 2002.
- [70] A. S. Krishnakumar and P. Krishnan. The Theory and Practice of Signal Strength-Based Location Estimation. In *Proceedings of the First International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2005.
- [71] Margit Kristensen, Morten Kyng, and Leysia Palen. Participatory design in emergency medical service: designing for future practice. In *CHI*, pages 161–170, 2006.
- [72] Daniel Kroening. The CPROVER User Manual. SATABS – Predicate Abstraction with SAT. CBMC – Bounded Model Checking. <http://www.verify.ethz.ch/satabs/download/manual.pdf>; accessed August 28, 2008.
- [73] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proceedings of the Third International Conference on Pervasive Computing*, 2005.
- [74] Jeremy Landt. Shrouds of Time. The history of RFID. An AIM (Association for Automatic Identification and Data Capture Technologies) Publication, October 2001. [http://www.transcore.com/pdf/AIM\\_shrouds\\_of\\_time.pdf](http://www.transcore.com/pdf/AIM_shrouds_of_time.pdf); accessed August 2, 2008.

- [75] Anatole Le, Ondrej Lhoták, and Laurie J. Hendren. Using Inter-Procedural Side-Effect Information in JIT Optimizations. In *CC*, pages 287–304, 2005.
- [76] Octopus Cards Limited. Octopus, 2008. <http://www.octopuscards.com/enindex.jsp>; accessed August 13, 2008.
- [77] Konrad Lorincz and Matt Welsh. MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking. In *Proceedings of the First International Workshop on Location- and Context-Awareness*, 2005.
- [78] Heng Lu, W. K. Chan, and T. H. Tse. Testing context-aware middleware-centric programs: a data flow approach and an RFID-based experimentation. In *SIGSOFT '06/FSE-14: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 242–252. ACM, 2006.
- [79] Sun Microsystems. Jini Technology Architectural Overview. <http://www.sun.com/software/jini/whitepapers/architecture.html>; accessed August 8, 2008.
- [80] Kavitha Muthukrishnan, Maria Lijding, and Paul Havinga. Towards Smart Surroundings: Enabling Techniques and Technologies for Localization. In *Proceedings of the First International Workshop on Location- and Context-Awareness*, 2005.
- [81] Brad A. Myers. A Brief History of Human Computer Interaction Technology. *ACM interactions*, 5(2):44–54, 1998.
- [82] Dust Networks. Dust Networks : Embedded Wireless Sensor Networking for Monitoring and Control, 2008. <http://www.dustnetworks.com/index.shtml>; accessed August 3, 2008.
- [83] Michael Nidd. Timeliness of Service Discovery in DEAPspace. In *ICPP Workshops: International Workshops on Parallel Processing*, pages 73–80, 2000.
- [84] Pamela Licalzi O’Connell. Korea’s High-Tech Utopia, Where Everything Is Observed, October 2005. <http://www.nytimes.com/2005/10/05/technology/techspecial/05oconnell.html>; accessed August 13, 2008.
- [85] University of California at Berkeley: Robotics Lab. SmartDust. Autonomous sensing and communication in a cubic millimeter, 2008. <http://robotics.eecs.berkeley.edu/~pister/SmartDust/>; accessed August 3, 2008.

- [86] Stanford Encyclopedia of Philosophy. Postmodernism, 2005. <http://plato.stanford.edu/entries/postmodernism/>; accessed July 31, 2008.
- [87] Robert J. Orr and Gregory D. Abowd. The smart floor: a mechanism for natural user identification and tracking. In *CHI '00 extended abstracts on Human factors in computing systems*, pages 275–276, New York, NY, USA, 2000. ACM.
- [88] Veljo Otsason, Alex Varshavsky, Anthony La Marca, and Eyal de Lara. Accurate GSM Indoor Localization. In *Proceedings of the Seventh International Conference on Ubiquitous Computing*, September 2005.
- [89] Jason Pascoe. Adding Generic Contextual Capabilities to Wearable Computers. In *Proceedings of the Second International Symposium on Wearable Computers*, pages 92–99, 1998.
- [90] Shwetak Patel, Khai Truong, and Gregory Abowd. PowerLine Positioning: A Practical Sub-Room-Level Indoor Location System for Domestic Use. In *Proceedings of the Eighth International Conference on Ubiquitous Computing*, 2006.
- [91] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling Ultra-Low Power Wireless Research. *Information Processing in Sensor Networks (IPSN) 2005. Fourth International Symposium on*, pages 364–369, April 2005.
- [92] Venkatesh Prasad Ranganath and John Hatcliff. Pruning Interference and Ready Dependence for Slicing Concurrent Java Programs. In *CC*, pages 39–56, 2004.
- [93] Olga Ratsimor, Dipanjan Chakraborty, Anupam Joshi, and Timothy Finin. Allia: Alliance-Based Service Discovery for Ad-hoc Environments. In *Proceedings of the Second ACM International Workshop on Mobile Commerce (WMC-02)*, pages 1–9. ACM Press, September 28 2002.
- [94] Nishkam Ravi and Liviu Iftode. FiatLux: Fingerprinting Rooms Using Light Intensity. In *Adjunct Proceedings of the Fifth International Conference on Pervasive Computing*, 2007.
- [95] Xerox Palo Alto Research Center Press Release. Xerox Names Computing Pioneer As Chief Technologist For Palo Alto Research Center, August 1996. <http://www.ubiq.com/weiser/weiserannc.htm>; accessed July 29, 2008.

- [96] Yvonne Rogers. Moving on from Weiser's Vision of Calm Computing: Engaging UbiComp Experiences. In *UbiComp*, pages 404–421, 2006.
- [97] Gruia-Catalin Roman, Christine Julien, and Jamie Payton. A Formal Treatment of Context-Awareness. In *Proceedings of the 7th International Conference on Fundamental Approaches to Software Engineering (FASE 2004)*, pages 12–36. Lecture Notes in Computer Science 2984, Springer, 2004.
- [98] Royal Society of Chemistry. Essays on Radiochemistry: Alpha, Beta and Gamma Radioactivity. <http://www.rsc.org/pdf/radioactivity/number3.pdf>; accessed June 13, 2008, unknown year.
- [99] Françoise Sailhan and Valérie Issarny. Scalable Service Discovery for MANET. In *PerCom*, pages 235–244. IEEE Computer Society, 2005.
- [100] Gregor Schiele, Christian Becker, and Kurt Rothermel. Energy-Efficient Cluster-based Service Discovery. In *Proceedings of the 11th ACM SIGOPS European Workshop (SIGOPSEW04); Leuven, Belgium, September 20-22, 2004*, Artikel in Tagungsband, pages 75–79. ACM SIGOPS, September 2004.
- [101] Bill Schilit, Norman Adams, and Roy Want. Context-Aware Computing Applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, 1994.
- [102] Bill N. Schilit. *A System Architecture for Context-Aware Mobile Computing*. PhD thesis, 1995.
- [103] R. G. Sonkawade, Rewa Ram, D. K. Kanjilal, and R. C. Ramola. Radon in tube-well drinking water and indoor air. *Indoor and Built Environment*, 13(5):383–385, 2004.
- [104] Crossbow Technology. Crossbow, 2008. <http://www.xbow.com/>; accessed August 3, 2008.
- [105] The Open TinyOS Community. OMNeT++: Discrete Event Simulation System. <http://www.omnetpp.org/>; accessed June 2006.
- [106] The Open TinyOS Community. TinyOS. <http://www.tinyos.net/>; accessed August 8, 2008.
- [107] The PalCom Consortium. Palpable Computing. <http://www.ist-palcom.org/examplesOfWork/accidents.php>; accessed June 2006.
- [108] T. H. Tse, Stephen S. Yau, W. K. Chan, Heng Lu, and T. Y. Chen. Testing Context-Sensitive Middleware-Based Software Applications.

- In *COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, pages 458–466. IEEE Computer Society, 2004.
- [109] Ubisense. Ubisense Platform, 2008. <http://www.ubisense.net/content/8.html>; accessed August 20, 2008.
- [110] United Nations Scientific Committee on the Effects of Atomic Radiation (UNSCEAR). ANNEX B: Exposures from natural radiation sources, subsection IIC2. [www.unscear.org/docs/reports/annexb.pdf](http://www.unscear.org/docs/reports/annexb.pdf), 2000.
- [111] University of California at Davis University of California at Berkeley. The Berkeley Sensor and Actuator Center (BSAC), 2008. <http://www-bsac.eecs.berkeley.edu/>; accessed August 3, 2008.
- [112] Alex Varshavsky, Anthony Lamarca, Jeffrey Hightower, and Eyal de Lara. The SkyLoc Floor Localization System. In *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications*, 2007.
- [113] Emil Venere and Elizabeth K. Gardner. Cell phone sensors detect radiation to thwart nuclear terrorism. <http://www.purdue.edu/UNS/x/2008a/080122FischbachNuclear.html>, 2008.
- [114] Christopher N. Ververidis and George C. Polyzos. Extended ZRP: a Routing Layer Based Service Discovery Protocol for Mobile Ad Hoc Networks. In *MobiQuitous*, pages 65–72. IEEE Computer Society, 2005.
- [115] Zhimin Wang, Sebastian Elbaum, and David S. Rosenblum. Automated Generation of Context-Aware Tests. *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 406–415, May 2007.
- [116] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The Active Badge Location System. Technical Report 92.1, Olivetti Research Ltd. (ORL), 1992.
- [117] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. The ParcTab Ubiquitous Computing Experiment. Technical report, 1995.
- [118] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 1991.

- [119] Mark Weiser. Open House, March 1996. <http://www.ubiq.com/hypertext/weiser/wholehouse.doc>; accessed July 29, 2008.
- [120] Mark Weiser. Ubiquitous Computing, March 1996. <http://www.ubiq.com/hypertext/weiser/UbiHome.html>; accessed July 29, 2008.
- [121] Mark Weiser. Mark Weiser, 1999. <http://www.ubiq.com/weiser.html>; accessed July 31, 2008.
- [122] Mark Weiser and John Seely Brown. The Coming Age of Calm Technology, October 1996. <http://www.ubiq.com/hypertext/weiser/acm-future2endnote.htm>; accessed July 29, 2008.
- [123] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.
- [124] Moustafa Youssef and Ashok Agrawala. The Horus WLAN Location Determination System. In *Proceedings of the Third International Conference on Mobile Systems, Applications, and Services*, 2005.
- [125] Pascal Zimmer. A Calculus for Context-Awareness. Technical Report BRICS Report Series RS-05-27.