uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Dineshbalu Balarishnan
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

An Infrastructure for Context Dissemination in Ambient Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Amiya Nayak
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

Luiz Fernando Capretz (U.W.O.)

Paola Flocchini

Nicola Santoro

Jiying Zhao

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

An Infrastructure for Context Dissemination in Ambient Networks

By

Dineshbalu Balakrishnan

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy
in Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa
Ottawa, Ontario, Canada
March 2010

Your file Votre référence
ISBN: 978-0-494-65566-5
Our file Notre référence
ISBN: 978-0-494-65566-5

Abstract

Context awareness is the ability to use contextual information in support of operations and decision making. The *context dissemination* process, an important component to enable context awareness, is essential for the smooth operation of various demanding and self-organizing context-based systems in *Ambient Networks*. Ambient Networks enable co-operation between heterogeneous networks, on demand, in order to provide seamless connectivity to end-users. We argue that incorporating contexts in a scalable and flexible fashion is a challenging task as it poses overhead and complexity due to non-automated service convergence, and is more difficult to maintain and interoperate successfully. We deal with this challenge and propose a meliorated context dissemination system.

We tend our work towards an overlay design, which acts as the key to assemble and interconnect context nodes and hide the heterogeneity and variability of the underlying networks. We propose a context-specific *Pure Overlay Space* construction and maintenance architecture on top of overlays to disseminate context information in Ambient Networks with respect to *ContextWare* (a model for context processing which acquires and distributes context data among interested context entities). We propose a *Context Dissemination Processor* for processing the Pure Overlay Space's contextual nodes by handling heterogeneous contexts and their respective profiles as agents to attain network interoperability. With the help of the Context Dissemination Processor, the Pure Overlay Space will serve as a middleware and multicast base for the development and maintenance of an application-layer context-aware dissemination protocol, the *CSON-D protocol*. The overlay infrastructure supports spatial decoupling, semantic clustering, personalization, and agent-based optimizations for virtual customized interaction between context entities. We design and implement a context-specific overlay architecture made up of multi-level overlay networks. The CSON-D protocol's multi-level overlay scheme exhibits adaptive behaviour with minimized casting functionality to disseminate contexts. We also propose an efficient context monitoring scheme with adaptive routing and learning engine to aid the dissemination process. Alongside the conceptual discussion, this thesis is complemented with implementation details with respect to ContextWare and result evaluation.

Acknowledgements

I would like to thank my academic supervisor, Prof. Amiya Nayak at the University of Ottawa, for his ideas, feedbacks, continued support, and inspiration. I have learned a lot from him. I am very grateful to graduate as his student.

I would like to thank my supervisors at Cistel Inc and Vayyoo Inc for providing me the opportunity to utilize their on-site systems and infrastructure for my project.

Finally, I would like to thank my friends and family for their understanding, unconditional love and support.

Table of Contents

List of Tables

List of Figures

List of Symbols, Abbreviations, and Nomenclature

ACS	Ambient Control Space
AN	Ambient Network
ANI	Ambient Network Interface
AOS	Agent Object Subscriptions
ASI	Ambient Service Interface
CAS	Context Aware System
CB	Context Benefit
CDP	Context Dissemination Processor
CDP-L1	First Context Dissemination Processor Level
CDP-L2	Second Context Dissemination Processor Level
CIB	Context Information Base
CID	Context Identifier
C_R	Context Requestors or Context Consumers
C_S	Context Sources
CS	Context Score
CSC	Context Sensitive Communications
CSN	Context Specific Node
CSON	Context Specific Overlay Network
CSON-D	Context Specific Overlay Network for Dissemination
DHT	Distributed Hash Table
DT	Dissemination Tree
ER	Entity-Relationships
GPS	Global Positioning System
ID	Identifier
IP	Internet Protocol
KB	Knowledge Base
KML	Keyhole Markup Language
LE	Learning Engine

LMU	Local Multiple Unicasts
MA	Mobile Agent
MP3	MPEG Audio Encoding Layer III
MSC	Message Sequence Charts
ON	Overlay Node
ON_L	Leader Overlay Node
OS	Overlay Space
OWL	Web Ontology Language
PA	Proxy Agent
PAN	Personal Area Network
POSpace	Pure Overlay Space
RSN	Request Specific Node
RSON	Request Specific Overlay Network
RT	Routing Table
SCF	Service Creation Framework
SN	Service Node
UI	User Interface
UCI	Universal Context Identifier
URI	Universal Resource Identifier
XML	Extensible Markup Language

CHAPTER 1
Introduction

1.1 Overview

The mobile device market is growing rapidly and mobile devices have evolved from an expensive luxury to an indispensable requirement with increased processing power and memory capacity. Now-a-days, significant improvements and contributions are being made towards mobile and ubiquitous computing each and every day [1], thanks to the rapid progress in wireless communication hardware and networking standards. This means that interactive and sophisticated applications and systems need to be aware of their dynamic surrounding environments and adapt accordingly, in an instant. As a consequence, the service providers have to create and deliver attractive multimedia services (all-IP based) by evolving their current networks to an architecture that can deliver such services in a highly adaptable and cost-effective way. Though user location and sensor information were the main forms of context for several years, currently, due to the convergence of different networks due to all-IP solutions, the context is not just about location anymore [2]. As the term context is not precisely defined and is not limited in scope, the first step is to understand what is context. In general, context information is any information that can be used to characterize the situation and/or operational state of an entity under different circumstances [3]; wherein, an entity can be a person, place, physical object, or computational object that is considered relevant to the interaction between a user and an

application, including the users and applications themselves. Hence, context information could denote devices, users, services, persistent data, etc. Context awareness is becoming an 'obvious' characteristic in systems and services as it offers an opportunity to increase system productivity in a substantial way, since human interaction is the ultimate bottleneck in networking and computing. New trends and emerging technologies focus on solving user tasks with minimum user intervention. Though the range of context types and context-aware features is never ending, most applications and systems have mostly focussed on identity and location from a user's point of view. This is because of the fact that a context information is difficult to use and manage because of its properties. Recently, novel solutions are often introduced by various researchers, including ourselves, to tackle this problem and develop context-aware applications.

Context-awareness [4]--[6] is the ability to use contextual information in support of services, operations, and decision making. Context awareness is also an important and integral part of heterogeneous environments research in next generation networks [7]--[8]. Context-awareness plays a vital role in such networks and systems, and is an efficient approach for personalizing network services [4], [9]--[10]. The basic order of relevance is to use a base or mediator to first collect the appropriate data (i.e., context sensing and monitoring), then process and model it as per the requirements (i.e., context management), and finally disseminate the refined information which could be used by a context aware system or service for its operation or adaptation. Information dissemination in a mobile environment has evolved for more than a decade [11], and today context dissemination is widely discussed as a part of context awareness. The dissemination of context information to enable context-awareness in various applications (for e.g., resource update and management) and systems is being under vigorous research. Though there is no fixed definition for context dissemination, it can be generalized as the process of disseminating or propagating the acquired fast evolving context information from information sources (e.g., context sensors) to various interested sinks (e.g., context sensitive clients). The dissemination of context information is essential for smooth operation of various demanding and self-organizing systems (for example, the systems that anticipate daily context updates and take necessary action without conscious mediation). As we access widespread contexts

by making the network itself context-aware (such as, network connectivity status), the problem of moving contexts from source to sink is not trivial. At the same time, due to context-aware networking, significant improvements and contributions are being made in heterogeneous environments each and every day for network operations and decision making.

A common framework [5] for context awareness across all functions in the control space is required in order to adapt service availability and service propagation in heterogeneous networks and dynamic environments. Autonomous systems in heterogeneous and dynamic environments, such as Ambient Networks (AN) [7], [12]-[13] are complex entities which offer an open environment for dynamic resource integration. Though the concept of disseminating necessary contextual information is studied under various circumstances and networks, its use from an Ambient Network point of view is little known. ANs comprise several different kinds of networks, from PAN to satellite networks, whose main aim is to provide a domain-structured, edge-to-edge view for the network control i.e., they dynamically embrace the heterogeneity arising from different network control technologies such that it appears homogeneous to the potential users of the network services. They enable co-operation between heterogeneous networks, on demand, in order to provide ubiquitous connectivity to end-users. The process of dynamic context dissemination is vital for ANs to operate effectively. We tend our work towards an overlay network design because of its flexible, distributed, and self-organization characteristics. Since ANs are highly dynamic, the overlay network should be adaptable to fit in the new environment. Overlay networks act as the key to assemble and interconnect context nodes and hide the heterogeneity and variability of the underlying networks to clients. We focus on the part of disseminating context information using overlays to enable context awareness in various users, devices, and applications, and hence enabling context-based applications and systems in ANs. As an extension to the proof of concept of the dissemination mechanism, a real-time application (*mobile asset tracking*) which involves context monitoring too, is developed. The role of mobile asset tracking and related applications have become critical in many medium to large scale organizations as it is cumbersome to track travelling assets (including smart phones and PDAs) in real time and precisely manage their routing. A vast amount of current

research and development is trying to take advantage of this functionality, and as a result, by utilizing its advantages, many new applications are coming forward.

1.2 Motivation

Most of the currently published research in the field of context dissemination are based on frameworks, such as simple Publish/Subscribe [14], wherein assumptions such as: context sources/ requestors/ brokers are always present, easily found, knowledge-oriented, fault-tolerant, etc are made. The commonly existing context-aware systems [5], [15]-[16] (refer section 2.6) (i) do not satisfy the demand for advanced and complex dissemination mechanisms for context-aware services, (ii) do not support non-specific structures, and (iii) are not efficient enough for autonomous systems. These constraints are mainly because of the fact that most of the related systems are (i) centralized, (ii) immobile in terms of involved entities, (iii) lack system state adaptation and customization, and (iv) importantly not context-aware. In reality, it is vital to have personalized environmental knowledge, i.e., utilize contexts. The major problem in most of the currently existing context dissemination systems is that they take a partial view of the overall dissemination problem and lack dynamic adaptation and interoperability i.e., not flexible enough to suit various applications at the same time. Though many studies have been done towards context-aware computing, the existing technologies are not sufficiently accomplished towards an adaptive context information dissemination scheme in next generation networks [16]. Any service provision or adaptation based on outdated context information could cause various problems. This in turn demands up-to-date dissemination of context information as the nature of the managed information is diverse and distributed. As per current technological standards, efficient context monitoring and dissemination software is lacking when compared with hardware solutions.

With the dynamics and heterogeneity envisioned in ANs, context-awareness is a key feature. Based on recent related work (refer Chapter 2), the demand for seamless provisioning of context information to AN-entities based on their dynamically changing requirements in a scalable (defined with respect to load balance, cost, and delay) manner, is

increasing. The existing dissemination systems do not satisfy the demand for ANs to enable context-based applications and lack efficient implemented solutions. Very little in-progress research work focus on network-centric context awareness, that is, context-aware computing from a network point of view. In line with the recent use of overlay networks for content delivery, we tend our work towards an overlay design. Various frameworks support various overlay topologies and schemes for dissemination, but the current focus is towards the non-functional attributes of such systems. These overlay schemes might create bottlenecks in the underlying network, overload network resources, and thus reduce network performance. Though some existing traditional overlay network based approaches [17]--[18] are reliable and provide efficient content delivery, they lack context-awareness or need rich resources for overlay multicasting. Despite its advantages, incorporating contexts in such infrastructures in a scalable and flexible fashion is a challenging task as it poses overhead and complexity due to non automated service convergence. While multicasting solves scalability problem to an extent, it introduces several non-trivial problems such as caching, consistency, and scheduling.

The real-time *mobile asset tracking* process can be costly and inefficient over wireless data transmission systems where cost is based on the rate of data being sent. We thus studied various asset tracking techniques and surveyed a wide range of wireless data transmission technologies to efficiently transmit the tracked data (e.g., GPS coordinates), text messages, or digital images from a mobile device to backend servers [19]--[20]. The various sources of location information have their own characteristics. We also studied specific constraints with regard to employing an always active GPS receiver in resource deprived mobile devices. We found that commercially available systems are generally optimized for a single application, such as sending a GPS coordinate, instead of efficient hybrid transmission with switching capability [21]. Also, most of the current service providers cannot correctly identify and track the traversed routes in multiple route based scenarios. We researched to understand the behaviour of typical traffic generated by a mobile device for reporting GPS data in various demographics and terrains [22]. This research helped us to understand the expected behaviour pattern and to develop a benchmark for the real-time application.

1.3 Challenges

The main challenges that the current computing world faces in a context-aware system (to perform context dissemination in ANs) are listed in this section as follows. We handle these challenges in our context dissemination protocol.

1) *Formality:* Context-aware networking, though enables new types of applications and services in ambient environments, lacks a standard framework and support from the network for the provision of contextual information. This hinders the widespread use context-aware applications and the rapid development of the field of context-aware computing. It is crucial that all entities within the network share the same interpretation of the information exchanged. This can achieved by a formal (i.e., precise and unambiguous) description of contexts. Support for a common framework [5] [16] for context awareness across all functions in the control space is required in order to adapt service availability and service propagation.

2) *Support for efficient reasoning and inference mechanisms:* Efficient reasoning mechanisms are needed for: deriving higher-level context information from lower-level pieces of information, evaluating context queries, model checking, and reasoning about relationships between different models.

3) *Support for interoperability*: Algorithms and protocols should be developed to enable interoperability between the dissemination protocols employed in different networks. These interoperable solutions should be precisely managed.

4) *Support for imperfect contextual information:* Context information, by its nature can be uncertain, incomplete, or unavailable at times. This aspect should be covered, for instance by interpolation of incomplete data on the instance level. The generic context-based challenges including semantic context profile creation and relationship matching between context profiles should also be considered.

5) *Support for fast access to contextual information*: As context information evolves fast, it should be rapidly disseminated in order to be valid in the dynamic environment offered by ANs [23].

6) *Support for adaptation (i.e., dynamic creation, modification, and extension):* As ANs are formed by dynamically collaborating heterogeneous networks, to guarantee continuous service delivery to context entities, the context monitoring and dissemination schemes should support adaptation as the networks evolve [24].

7) *Support for context knowledge validation:* It is essential to validate contexts on structure and instance levels, so as to be fault-tolerant.

8) *Applicability to existing service environments:* From the implementation point of view, the dissemination scheme should be applicable within the existing infrastructures envisaged for ambient computing.

9) *Support for ubiquitous/ pervasive computing:* Automatically cooperate and run applications that are sensitive to user's preferences and requirements by anticipating their actions and creating a user-friendly data model. The main technical issues in such networks include ubiquitous-API, power management, personalization, mobility, flexibility, scalability, and dynamicity.

10) *Delivery option and periodicity*: Customary dissemination requirements include the selection of appropriate delivery and periodicity options, such as push- or pull-based and periodic or a-periodic, respectively. The dissemination scheme should consider the receiving client's type, preference, status, location, etc.

1.4 Objectives

Based on the above discussed motivation and challenges, to enable context-based applications in ANs, a novel dissemination strategy is essential. These constraints motivated us to develop an application protocol that focuses on solving user tasks with minimum user effort and intervention. We thus focus on the part of disseminating context information using overlays to enable context awareness in various users, devices, and applications, and hence enabling context-based applications and systems in ANs. Thus, the aim of our overall research is thus to build a high-level (i) Interoperable, (ii) Balanced, (iii) Adaptable, and (iv) Realistic context dissemination mechanism. The solutions for these four main functional objectives are the main contributions of this thesis as sequentially listed in section 1.6.

AN's general architecture, ContextWare, provides context-aware functionally, but it lacks efficient implemented solutions [7]--[8], [25]. The dynamics and heterogeneity envisaged in ANs accentuate the need, resulting in dissemination of dynamic and distributed context at different layers [24], [26]. ANs aim at expanding the existing network capabilities in the areas of network composition, mobility, and heterogeneity [27]. In order to support these ambient networking characteristics and features, the above listed challenges must be imposed on the dissemination approaches. The complexity of these challenges increase along with the current growth of ANs. To tackle the above discussed objectives, the overlay-based context aware system (CAS) should be capable of answering the following questions regarding context dissemination: (i) *Why* dissemination is necessary? (ii) *What* exactly should be disseminated? (iii) *When* and *where* the dissemination process should take place? and (iv) *How* can this dissemination process be performed precisely? The questions should have unambiguous solutions in context dissemination approaches. By making the network itself context-aware, significant improvements and contributions are being made towards ubiquitous / pervasive computing each and every day for network operations and decision making. In order to enable dynamic behaviour, the context awareness' decision making capabilities must be implemented in support of each function of the system control space. For example, the multimedia routing entity by being aware of the devices available to a user, should adapt its behaviour and direct streams to an appropriate device. This routing is achieved via context dissemination, wherein flexibility and scalability factors are the common concerns. Another functionality we try to optimize in this research work is *multicasting*. In order to minimize multicasting problems, we reduce the casting functionality to a minimum by employing a multi-level overlay scheme and avoid the use of complex multicast and broadcast routing protocols.

As the context information has to be provided explicitly unlike human interaction and as AN-based applications (such as, smart classrooms) need a time-constraint constant flow of information, the aim of our overall research is to build a balanced, adaptable, and interoperable overlay-based context dissemination mechanism in ANs. Herein, balance concerns are with respect to a node's load and benefits, adaptability relates to dynamic

overlay formation, and interoperability denotes context handling in heterogeneous autonomous environments. Developing such a CAS and disseminating necessary context from a myriad of today's contexts is complicated and tedious. To summarize, the main challenges and requirements are the seamless and spontaneous provisioning of contextual information to AN-entities by adapting to the context entity's dynamic requirements. With regard to real-time application development, though much work is taking place on mobile asset tracking, not many are working on route learning and route adaptation. Based on the study (see section 1.2), we aim to design efficient solutions to improve GPS-utilized asset tracking solutions and consume valuable mobile resources to create an accurate and cost effective route tracking and learning mechanism for the mobile assets by utilizing our context dissemination infrastructure. Hence, our overall objective of the real-time application is to *adaptively* adjust the transmission interval of the geographical tracking data (geo-data) so that it is reported only when needed.

1.5 Context Dissemination Approach

CSON-D, short for *Context Specific Overlay Network for Dissemination*, is a novel overlay-based context dissemination scheme which aids in enabling network users, devices, and applications to be context-aware. It supports all types of context information (includes networks, systems, protocols, applications, etc) in order to exhibit balanced, adaptable, and interoperable context dissemination. The type of context disseminated has no limit other than semantic restrictions. Due to the above discussed challenges and also due to technical deployment barriers in the IP layer, we combine the generality of IP solutions with deployment benefits of overlay solutions for virtual interaction between context entities (i.e., source-contexts, C_S and request-contexts, C_R). We also evaluate the existing dissemination approaches with respect to the challenges (see section 1.3) and believe that our adopted dissemination scheme satisfies most of the objectives. We deal with the challenges and propose a context dissemination system by designing and implementing a hierarchical semantic-based context-specific overlay architecture made up of multi-level overlays. Our basic approach is to first divide the contexts, then form a mesh structure via bisected context key-spaces, which later alters into a top-layer overlay for unicasting.

Casting refers to the virtual coordination of the context entities in the interaction layer. The CSON-D protocol includes (learning-based) intelligence characteristic which concerns adaptability and interoperability to network dynamics, (customization-based) personalization, (virtually redundant contextual nodes based) fault tolerance optimization framework, and an efficient platform defined with respect to load balancing, cost, and latency. An outline of the mainly proposed components of our approach which are discussed in this section is illustrated in Figure 1.1.

In this thesis, we propose a context-specific *Pure Overlay Space* construction and maintenance architecture on top of overlays to disseminate context information in Ambient Networks. The Pure Overlay Space architecture enables context incorporation in ANs by coping with the dynamic nature of today's applications and systems. For this purpose, we tend towards an overlay infrastructure with spatial decoupling and semantic clustering. The POSpace design acts as an interaction medium with regard to space and representation and adopts a completely logical overlay capable of achieving dynamic balancing between context entities. It guides the CSON-D protocol towards ambient networking by handling: (i) local transformations by peer (i.e., agent objects) grouping, (ii) flexibility optimizations i.e., personalized requirements by personalizing overlays, and (iii) hybrid customizations. The POSpace is proposed with respect to *ContextWare*. ContextWare is a model for context processing which acquires and distributes context data among interested context entities (i.e., context sources and context requestors). We propose a *Context Dissemination Processor* for processing the Pure Overlay Space's contextual nodes by handling heterogeneous contexts and their respective profiles as agents to attain network interoperability. The CDP, therefore, is responsible for pre-processing the contextual nodes as overlay nodes by communicating with context entities (in CDP-L1) and the overlay space (in CDP-L2). We employ agent objects for virtual customized interaction between the overlay nodes i.e., context entities. The CDP guides the CSON-D protocol towards realization by proposing and evaluating optimizations such as *proxy overlay mechanism*, *hybrid agent structure*, and *aggregated multiple objects*. The POSpace and CDP 'mirror' with the ContextWare model, and thus serve as its implementation scheme.

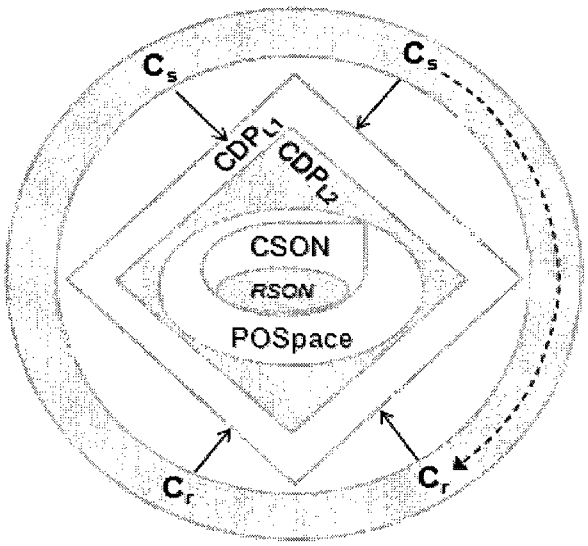


Figure 1.1 The mainly contributed components.

With the help of our CDP, the POSpace will serve as a base for the development of an application-layer context-aware overlay multicast protocol i.e., *multi-level overlay infrastructure*. This overlay scheme acts as a middleware and multicast-base for virtual interaction between context entities to suit futuristic systems, networks, and environments. By autonomic management of target-specific context information at network-level we reduce the amount of human attention required. The multi-level overlay scheme is mainly made up of multi-level CSON (Context-Specific Overlay Network) and RSON (Request-Specific Overlay Network) clusters to disseminate contexts via unicast interactions. We design and implement this context-specific overlay architecture which exhibits adaptive behaviour with minimized casting functionality. Overlay multicasting is preferable to be performed through multiple unicasts, as Internet routers do not commonly support advanced IP multicast protocols. In order to minimize multicasting problems, we reduce the casting functionality to a minimum and avoid the use of complex multicast and broadcast routing protocols. The overlay layering concept's significance with respect to casting requirements is also analyzed in conjunction. In our research work we focus more on context-specific topology formation and optimization against C_R requirements/ subscriptions, and hence do not explicitly address the issue of efficient routing and casting of messages from top-level overlays. The multi-level overlay scheme satisfies our goals with regard to creating an adaptive data model for context monitoring and dissemination, thus exhibiting dynamic and

fair context dissemination. While we mainly focus on the part of disseminating context information, we also propose an efficient *context monitoring* scheme to aid the CDP. Context monitoring is performed to monitor mobile entities (e.g., PDA) and their properties in an efficient fashion [23], [28]. Since the communications channels used for such transmissions are considered to be relatively expensive resources, transmission of the monitored context information at a fixed transmission rate is cost ineffective. We infer manageable *policy profiles* to logically and dynamically adapt the daily monitoring operations of a mobile context [29]. The monitoring scheme handles redundant context information and improves valuable resource utilization in mobile entities, and thus aid the CSON-D protocol.

1.6 Thesis Contribution

The main contributions of the thesis are as follows:

1) A *Context Dissemination Processor* functional area with complete agent-based context node handling which satisfies the *interoperable* functionality objective. The CDP is responsible for processing of overlay nodes by handling heterogeneous customizable contexts such as networks, users, and applications, and their respective profiles. The basic approach is published in [30].

2) A *Pure Overlay Space* architecture with various overlay-based optimizations which satisfies the *balanced* functionality objective. The POSpace copes with the dynamic changing nature of today's networks and systems and ensures the availability of up-to-date contextual information. The basic approach is published in [30] and the optimization schemes are published in [31]--[32].

3) A *Multi-level* overlay structure with intelligence, personalization, and fault tolerance features which satisfies the *adaptable* functionality objective. It exhibits adaptive behaviour with minimized casting functionality. The basic approach is published in [33] and the optimization schemes are published in [34].

4) A *Context Monitoring* scheme to track contextual mobile resources and to manage their dynamic location. It acts as an optimization scheme for the dissemination protocol along with an adaptive routing scheme and a personalized learning engine. Together, the context dissemination infrastructure is utilized to develop a real-time application to track mobile assets using geographical data, which satisfies the *realistic* functionality objective. The basic approach and optimizations are published in [35]--[36].

5) A ContextWare mirroring scheme to satisfy a context requestor's requirements via an implementation architecture.

1.7 Thesis Outline

The thesis has so far introduced the context dissemination scheme by: stating the motives for developing a context dissemination protocol, stating the scheme's objectives, stating the challenges faced for developing such a scheme, and listing its requirements. The first chapter overviewed the approach undertaken for designing and implementing the context dissemination protocol, and stated the principal contributions. The rest of the thesis is organised as follows.

The second chapter presents an overview of background concepts that help better understand the thesis. It discusses the thesis' main theories, including *Context Awareness*, *Ambient Networks*, *Overlay Networks*, *Agent Technology*, and *Policy Profiles*. In addition, the chapter surveys and reviews recent related work of the above stated theories with respect to context dissemination. The chapter finally compares selected related work with each other.

The third chapter first defines the main terminologies used. It then overviews the context dissemination concept and discusses its features with regard to ambient networking. The chapter describes the high level architecture of the context dissemination protocol and ContextWare mirroring, which are sub-divided into four design model chapters. The chapter

also briefs an example usage scenario of the context dissemination protocol which will be used and discussed in the following chapters as a running example.

The fourth chapter discusses the functional area which processes the context entities as overlay nodes. The chapter elaborates the CDP's main components one-by-one with related algorithms. The basic functionalities of these CDP components are enhanced by multi agent object dissemination. By comparing the CDP component's usage with respect to the example scenario, the chapter further illustrates CDP's significant aspects. The chapter finally states the employed implementation architecture and infrastructure to evaluate the results of the CDP.

The fifth chapter discusses the design model of the Pure Overlay Space. It states how it copes with the dynamic changing nature of today's networks and systems by discussing the employed spatial decoupling and semantic clustering techniques. The POSpace's basic functionalities are then enhanced by proposing optimized techniques to infer personalization and customization. It also discusses the hybrid agent mechanism, and context scoring and context fairness schemes that are employed to maintain dynamicity and fairness in the POSpace. With the use of the running example from previous chapters, we illustrate POSpace's significant aspects. The chapter finally evaluates the performance results of the POSpace.

The sixth chapter elaborates the multi-level overlay networking scheme with its pros and cons. It illustrates the CSON-D protocol by discussing the formation and maintenance of Context Specific Overlay Networks and Request Specific Overlay Networks along with the various technical challenges faced. The employed casting scheme and its delivery semantics are discussed. The chapter also discusses the adaptation and fault tolerance modules to optimize the multi-level overlay scheme. The chapter finally evaluates the performance results of the CSON-D protocol with respect to multi-level overlay with the help of an example scenario.

The seventh chapter provides an overview of the role of context monitoring and context dissemination in a real-time application. It serves as an extension to the proof of concept of the CSON-D protocol discussed in the previous chapters i.e., the use of context dissemination mechanism along with context monitoring in a real application. As the application is based on mobile asset tracking, we first discuss its related word, and elaborate the geo-tracking scheme which has its own algorithms and strategies to track and mange mobile assets (i.e., contexts). The requirements, specifies, and the approach carried out to monitor these contexts are discussed. This context monitoring scheme is the basis of the real-time application and is a part of the context dissemination infrastructure. It then proposes approaches, including a personalized learning engine, to aid the context monitoring scheme. The chapter finally evaluates the performance of the context dissemination infrastructure with regard to the mobile asset geo-tracking application.

Finally, the eighth chapter summarizes the thesis contributions and concludes the thesis. It also discusses possible future research directions which would extend the proposed CSON-D protocol in an ambient network infrastructure.

CHAPTER 2
Background and Related Work

This chapter presents an overview of background concepts that help better understand the thesis. It discusses the thesis' main theories, including *Context Awareness*, *Ambient Networks*, *Overlay Networks*, *Agent Technology*, and *Policies*. In addition, the chapter surveys and reviews recent related work of the above stated theories with respect to context dissemination. The chapter finally compares traditional and recent specific dissemination approaches with each other.

2.1 Context Awareness

The rapid growth and progress of wireless communication hardware and networking standards brings us closer to the point where context-awareness is a popular commodity. The ability to use contextual information in support of operations and decision making can lead to adaptive systems that interact with the surrounding environment and function according to emerging conditions [37]. Context-awareness establishes a base for the adaptation. Such systems are intelligent from a user's point of view as they are customized according to user's contexts, similar to ubiquitous computing. The terms context-awareness, pervasive computing, and ubiquitous computing are in one way or other related to one another. A key drawback of context aware systems that has to be overcome is that different context-aware systems with the same context information could behave differently based on

how they perceive the contexts. Another drawback of context awareness is the static evaluation of context against predefined rules. Thus, it is essential that the use of context information must be based on *policies* and *semantics* in order to achieve adaptability of a more dynamic nature. Semantic context, a high level contextual term, is nothing but a knowledge representation wherein its meanings are defined within the representation. Context-awareness plays a vital role in supporting autonomous applications, providing by the same token an efficient approach for personalizing network services. This suits the requirements of today's autonomous systems (for e.g., Ambient Networks), wherein context information evolves fast. Initial context awareness research focused on *ambient intelligence* [38]--[39], whose research started as early as 1998. Context information gathered from various sources is the basis of ambient intelligence that assists sinks' daily activities in an autonomous fashion. By making the network itself context-aware, significant improvements and contributions are being made towards ambient computing. Network based contexts could include network infrastructures, underlying nodes, or also network policy agreements [40]. Other contexts types that we mainly deal with include but are not limited to devices, users, and services. As context is regarded as a function over time, it can take various different types and sub-types, which together form a context hierarchy. As we access widespread contexts, the problem of moving contexts from source to sink is not trivial. In a real-time intelligent environment, systems require sensors that capture and provide contexts. Various sensors are being used innovatively to provide contexts, but their discussion is out of scope of this thesis.

Context management involves several complicated tasks such as collecting the raw contextual data from various sources (context sensing using Context Sensitive Communications[1] (CSC) [41]), processing and modeling[2] of this data, and finally disseminating the refined information which could be used by a context aware system or service, for its operation or adaptation [37], [42]. As yet, network context acquisition,

[1] In Context Sensitive Communications, the communication channel is established between a pair of devices (for e.g., network nodes or mobile devices) based on some specific contexts (for e.g., time or bandwidth).

[2] The concept of context modeling is out-of-scope of this thesis.

illustration, dissemination, and transfer negotiation and management are not effectual with existing well-accepted networking technology. Although these tasks were studied under various circumstances and for different types of networks, their execution from an ambient network point of view is worth examining. One of the main facts for the lack of such applications in mobile, wireless, and dynamically composed networks is that there is no flexible and scalable way to support context management.

- ***Context acquisition***: Context acquisition is the process of collecting heterogeneous context information from various (distributed) context sources such as widgets, networking components, etc [42]. Since the usual focus is on acquiring context information from networks, particularly mobile and dynamic networks, and not just from the application's immediate environment, it is necessary to consider all possible information available in the network. Also, the client's present situation needs to be sensed. Considering the vast amount of heterogeneous information to be acquired, several methods including sensors [42], the VIEW concept [43], push-pull strategy [44], UDDI [45], etc have been discussed for acquiring information, which is useful for the application under consideration. As context information evolves fast, the acquired context is expected to be spontaneously processed and disseminated to effectively perform CSCs.

- ***Context processing and modeling***: The attributes that are collected from various context sources are then processed to form meaningful and concrete context information. Network-centric context awareness could result in vast amount of context information being available for processing (for e.g., filtering, merging, extrapolation, and adaptation) [46]. Context modeling deals with modeling the meaningful context so as to satisfy the requirements of specific applications, services, or users. Several context models are suggested so as to aid applications, services, or users to find the context information they are seeking i.e., context modeling techniques help to completely and reliably represent the situation. By doing so, the gap between applications and service deployments is bridged. This is a multi-step process i.e., in each and every step, 'refinement' techniques are applied to the acquired data or already derived context

information to derive the requisite [16]. In order to support context-awareness, there is a need for developing a formal context model to facilitate context representation, sharing, and semantic interoperability of heterogeneous systems.

- ***Context Dissemination:*** On one hand, a well-designed context information model is needed to efficiently utilize and share the wealth of information available. On the other hand, a suitable context dissemination strategy is critical to ensure the availability of up-to-date contextual information, in a scalable manner. Dissemination can be generalized as the process of propagating the acquired context information from information sources to sinks. Achieving context information dissemination in a rapidly changing time-constrained heterogeneous domain, transparently to the potential users is a challenge. Hence, active and seamless dissemination of contextual information is an important component of context awareness design which could either be event-based or flooding-based. Further generic and technical details about context dissemination are discussed in section 3.1.

2.2 Ambient Networks

Ambient Networks comprise several different kinds of networks, from PANs to satellite networks, whose main aim is to provide a domain-structured, edge-to-edge view for the network control. ANs [12]--[13], dynamically embrace the heterogeneity arising from different network control technologies such that it appears homogeneous to the potential users of the network services. It provides an internetworking solution for the integration of heterogeneous networks based on the dynamic composition of those networks, by supporting distinctive proprieties [27]. Network composition is the key Ambient Networking concept that implies a new set of requirements for creation and management of control relationships between various heterogeneous networks. The TurfNet architecture distinguishes network compositions between two different types: *vertical composition* and *horizontal composition* [47]--[48]. When ANs compose vertically, one of the several composing networks assumes a (context) provider role, while the others assume a (context) consumer role. Such network composition hides administrative, control, and routing

functionalities, and also reduces the complexity of global interaction as new ANs can join locally without any global interaction. While in horizontal network composition the composing networks establish a direct peering agreement. When two personal area networks compose, such network composition takes place. Therefore, during horizontal composition, there is no need for any AN to assume the fixed role of either a provider or a consumer. The ANs that take part in horizontal network composition only are called *pure* ANs, while the vertically composed ones are called *specialized* ANs. Realistically, AN hierarchy should contain both vertical and horizontal compositions. The network integration is translated in the guarantee of service continuity for the end-user, enabling him/her to use any device to access services seamlessly across any type of network. This is achieved through a set of universal control functions used as an overlay control plan to seamlessly integrate any network, the Ambient Control Space (ACS) [23]. The ACS encompasses all control functions in a certain network domain; together with a connectivity network it is called an AN. The concept of ANs is based on all-IP mobile networks, which can be characterized by a clear separation between transport and control (ACS in AN) related tasks. The main characteristics of an all-IP based AN thus includes: (i) heterogeneity, (ii) comparability, (iii) autonomic, (iv) self-manageability, (v) dynamicity (on-demand and real-time), and (vi) mobility.

Context awareness is an important property of ANs, which should be suitably incorporated within traditional networks in order to create an ambient network [26]. In such a network, the need for context dissemination can be easily perceived [24]. The dynamics and heterogeneity envisaged in ANs accentuate the need, resulting in dissemination of dynamic and distributed contexts at different layers [8]. The basic requirement in any AN regarding contexts is support for a common framework for context awareness across all functions in the ACS. This is very important to maintain scalability and flexibility as all the involved components in various networks should understand the vocabulary using which context information is disseminated. In order to enable such dynamic behaviour, decision making capabilities must be implemented in support of each function of the ACS. This approach aims at making the network itself context-aware by collecting and making use of the surrounding contextual information for network operations and decision making. In this

regard, context aware customizable overlays are expected to play a vital role in providing an efficient approach for personalizing AN services. For example, the multimedia routing entity by being aware of the devices available to a user, could adapt its behaviour and direct streams to an appropriate device. This contextual information is not only used by the network entities (i.e., network services) but can also be provided to applications or end-user services. Figure 2.1 illustrates this network-centric approach. The Ambient Service Interface (ASI) is an interface towards user/application plane, while Ambient Network Interface (ANI) is a network level interface used for network composition [12]--[13]. Additionally, the ACS operates on an abstraction of the physical infrastructure, wherein only the resources that are controlled by the ACS are visible. The resources provide access to the ACS via a common reference point called the Ambient Resource Interface (ARI).

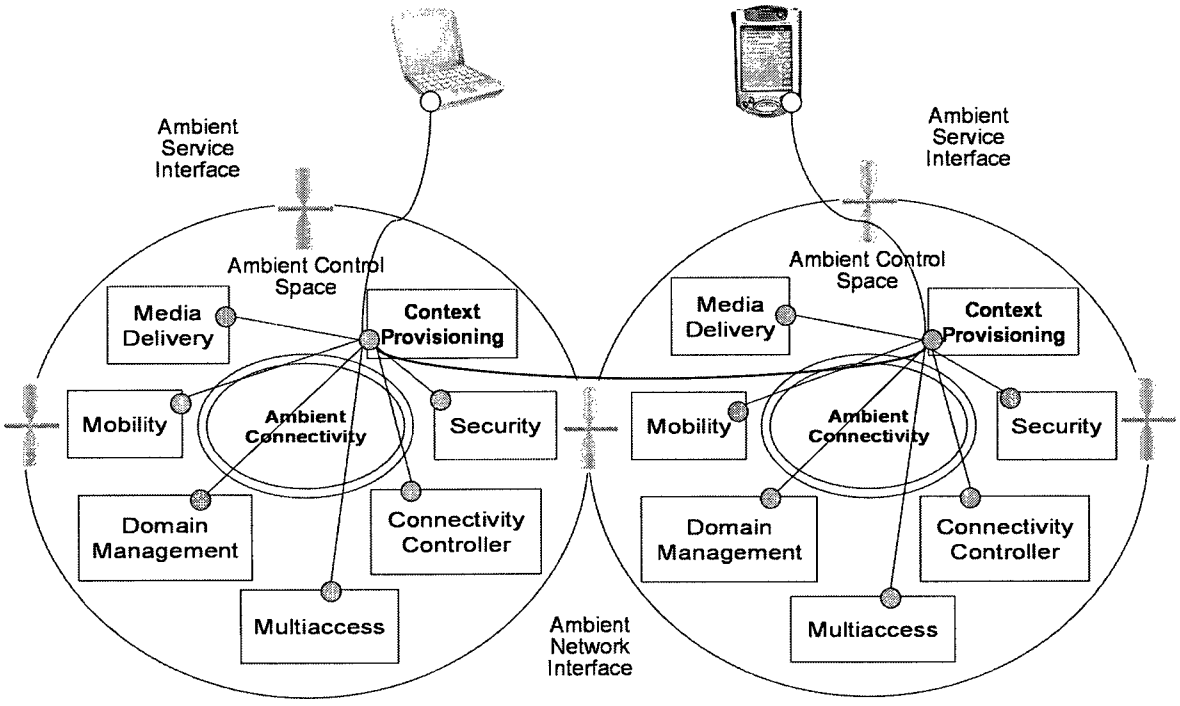


Figure 2.1 The network-centric context management approach used in ANs.

As context acquisition and context management are normally considered in conjunct with context dissemination, an information base or a mediating function could normally be employed to encompass the physical and logical source and sink for context data. The

common framework includes support for collection and management of context information and a distributed repository for extraction of up-to-date context information i.e., a Context Information Base (CIB) [8]. The mediator function, to collect appropriate data, is improvised by a CIB in an AN, which represents a distributed repository from which up-to-date context information can be extracted, whose ultimate goal is context-aware service provisioning. The CIB is the generic component that is used for interaction with application, device, and service specific components in an AN. A CIB, which in general terms could be viewed as a central web server, is used to improvise this fact in an AN. The CIB repository contains all relevant context data and computation that belong to the entities that are subjected to context awareness. The task of collecting, processing, and distributing context information to and from the AN entities poses the requirement for a generic component, rather than application- or device- specific components in the AN.

Context information sources at different layers (for example, sensors detecting user location, QoS monitors, etc.) must have interactions with a common control plane, such as the Context Control Plane (CCP) in ANs, in order to allow notification of changes as soon as they happen. The CCP is the context-aware control and management functionality that delivers information to context-aware clients. The CIB stores the context information received from different sources residing at different layers so that effective dissemination is feasible. For example, consider two ambient networks AN#1 and AN#2, where an agreement exists between both the networks to cooperate in order to provide an end-to-end service to a specific user. The CIBs of both the ANs are the context sources and sinks for the exchange of context data, normally via a control interface. The interface could enable inter-domain context negotiation across domains. Context data collection and dissemination by utilization of the CIBs of the involved ANs ensure continuity of the service the user has subscribed to. From a service provider's point of view, the CIB or the context interpreter (provides high-level contexts by interpreting low-level contexts; thereby operates as a context provider) is the central source of end user context profiles [49]. In the case of mobile end users, an archive of the users' previous movements can be cached in the CIB.

2.3 Overlay Networks

In order to successfully route data from one point in a wide area network to another or to another network altogether, well established collaboration is essential between the involved routing entities operating in various domains [17]. This data routing presents a limitation for the deployment of new services, especially in the Internet, as there exists several legacy networks whose restrictions can prevent the deployment of the services into the network. As it is resource consuming and time consuming to make all network components aware of this new service and as new network functionalities need new control protocols, additional capabilities are usually offered as overlays over the underlying network infrastructure with overlay nodes providing the additional functions. This viable solution uses network overlays as the initial deployment mechanism, which would serve as an add-on that can be incrementally deployed over an existing/underlying (legacy) network to demonstrate the benefits of proposed new services. The underlying network is usually formed by a set of end systems, routers, and paths between the two. In general, an overlay network is a logical abstraction of the underlying physical network, wherein the end-to-end paths and the end nodes form an overlay network. Unlike the underlying network, overlay networks do not require any modification on the physical network infrastructure or router support.

Overlays are structured virtual topologies on top of the physical level based on node-content attributes by maintaining and converging clients. They are also defined as a collection of data tunnels connecting network edge routers or access routers, supporting the same processing functions. Overlay networks have a network semantics layer above the transport protocol level that organizes the network topology as per the node's content. Overlays are used for content delivery as they provide effective and reliable services. Traditionally, content distribution networks are the most successful network systems that used notions and techniques similar to overlay networks. Overlays are also evolving into a critical component for self-organizing systems and network topologies [50]. They facilitate guaranteed context delivery and enable automated coordination and management. Depending upon the application under consideration, various overlay topologies could be chosen to guarantee high performance with minimum delivery latency and link stress. For

example, the star topology is preferred for systems which are delay sensitive. In some cases, such as in very large systems, each layer / cluster with an overlay could have its own topology according to its (contextual) requirements. Several types of overlays have been proposed each with specific goals in mind. For example, MBone (built on top of the existing Internet to provide multicasting capabilities), 6Bone (built on top of the Internet enabling IPv6 demonstration and application deployment), and ABone (for supporting active networks research). Overlay networks act as the key to assemble and interconnect context nodes and hide the heterogeneity and variability of the underlying networks to clients [18], [51]. They thus suit the autonomous network characteristics and system infrastructure. The overlay integration capitalizes the wide-range of overlay advantages in autonomous systems, unlike many context distribution systems. The basic constraint in all overlay based applications is the construction of overlay topologies. The way in which a (overlay) topology is constructed has a direct impact on system performance. Thus, it is essential for applications to carefully select paths between overlay node pairs. In dynamic and heterogeneous environments, both paths and nodes tend to change frequently, and hence the constructed overlay topology has to be adapted accordingly. Since ANs are highly dynamic, the overlay network should be adaptable to fit in the new environment. The topology or properties of the underlying networks or users might change, which might necessitate new overlay nodes or re-configuration. By introducing the concept of adaptive overlay networks, the overlays can adapt and respond to those changes. Adaptation should occur without any user involvement in a transparent fashion between the participating active context entities. Context-aware customizable overlays are expected to play a vital role in providing an efficient approach for personalizing ambient network services. For example, a multimedia routing entity by being aware of the devices available to a user, could adapt its behaviour and direct streams to an appropriate device.

There are three significant overlay based ideas which are worth mentioning [52]: (i) Resilient overlay network [53] is a generic application level overlay where the resilient overlay nodes monitor the quality and availability of paths between themselves in order to improve the resiliency and availability of paths over the existing Internet, (ii) Semantic overlay network [54]--[55] is a peer-to-peer based overlay network that utilizes the

semantics of the participating nodes (which are clustered together based on semantic matching) to determine how to achieve intelligent query routing, and (iii) Network virtualization [56] moves the overlay into the network itself allowing the simultaneous coexistence of multiple overlay networks, each with potentially varying technologies, on top of a single physical substrate. We tend our work towards an overlay design because of its flexible, distributed, fault tolerance, controlled lookup-times, load balance, and self-organization characteristics. On the other hand, overlay schemes might create bottlenecks in the underlying network and overload network resources, thus reducing network performance. We utilize the attractive overlay qualities and avoid its negative aspects to overcome issues such as routing inefficiency, path inflation, unbounded lookup times, disorganized clients, unbalanced load, and topology restrictions.

2.4 Agent Technology

An agent is an entity that represents a person, an organization, or a software application, which independently or by interacting with other agents executes a task or set of tasks. Agents usually have a given itinerary which they follow in order to complete tasks and report results. In the case of intelligent agents, the itinerary may change dynamically depending upon agents' status at a particular terminal or node in a network. Agent systems could result in performance bottlenecks due to lack of resources or overloading [57]. Common solutions to the performance bottleneck problem include delegating agents' tasks to other agents, making agents autonomous, and migrating agents to foreign hosts. Agents can move from one location to another within the same platform, or can move from one platform to another i.e., mobile agents [58]. Mobile agents are autonomous and intelligent programs that move through a network, searching for and interacting with services on one or more user's behalf. The agents move when they choose to do so and they should be executable on every machine in a network without having to install the code on every machine it moves to. The agent mobility feature significantly improves agents' performance and it makes the agent-based framework application-independent [57]. By using this feature, the network traffic could significantly be reduced. Mobile agents approximately need four times lesser bandwidth to complete tasks involving intense remote communication, when

compared with the client-server approach. Mobile agents complete their tasks in less time and reduce latency when compared with non-mobile agents. Also, the mobile agent based approach doesn't need full-time active network connection to perform tasks i.e., after the agent code has been migrated, operations can be performed in the disconnected mode. An alternate approach to agent mobility is agent cloning [57]. The agent cloning approach solves insufficient resources and agent overloading problems. According to the agent cloning approach, agents may clone, pass tasks to other agents, die, or merge. Therefore, the agent cloning approach features both the agent mobility and transfer of tasks. For example, an agent clone could be created on a foreign host and tasks could be transferred from an overloaded agent in the local host to the cloned agent. This cloned agent could finally die after performing the required tasks. If additional resources are available at the local host then agents could first be cloned locally and later be migrated to a foreign host. Information monitoring and filtering agents keep track of only the important information, and discard the rest.

Agents could communicate with other agents within the same platform or with agents in remote platforms. An agent platform is the execution environment wherein agents can be dynamically created, moved, cloned, or destroyed [59]. Agent platforms should recognize foreign agents and messages received from foreign agents. Agent technology is sometimes known as (and compared to) push technology when information is autonomously fetched and delivered by agents [60]. Agents can be viewed as a component of a software application in certain cases [61]. Voice recognition, mobility, location sensitivity, and personalized intelligent filters can be enabled in software applications using agents. Agent-based computing can be considered as a natural extension to object-oriented programming due to similarities in organising and handling complex structures. The distributed nature of agents makes them extensible and simple. Agents are well suited for mobile device applications due to their small size and properties such as adaptability, scalability, mobility, etc. Applications for mobile devices using agent technology need limited bandwidth as the mobile device's wireless interface should be active only when sending and receiving data. Even though agents are heterogeneous, there are some general properties that most agents do possess or recommended to possess. They are: Autonomous, Communicative/

Cooperative, Reactive, and Intelligent [62]. Agents are classified based on their properties and agents possessing all the above mentioned properties will effectively carry out assigned tasks. Some agents could be proactive and possess goal directed behaviour. In addition to various application-specific usages (for e.g., user agents), agents can also be used in various fields including medical services, personal shopping, tourism, government services, etc [63].

2.5 Policy Profiles

Policies are rules or conditions set by the user in order to govern the behaviour of entities within a specific domain i.e., a set of factual and behavioural specifications that are included on every computing element and resource within a domain. Policy-based network management schemes can relieve a network administrator from the burden of configuring each and every entity / device manually. It is more flexible since network entities can be reconfigured by creating or updating policies [64]. Each policy generally consists of a triggering event, a set of conditions, a set of actions, and a life time. Policies are generally applied for security (for restricting access), management (to assign rules for participating entities), and conversations (to structure and carry out conversations between entities) [23], [65]. Scalability and flexibility are the main benefits from employing a policy-based system in autonomous systems. While scalability is achieved by applying policies to a large number of users, devices, and applications, flexibility is achieved by separating the policy from system implementation. Policies can be used to set a client's attributes, constraints, relationships, credentials, network types, data types, content types, resources, and spatial and temporal parameters. There are several policy languages that aim at formalizing the specification of policies so that they can be represented and interpreted by machines. Policies are yet to be standardized on a global scale and this will be essential in the near future considering its wide usage. Policies can also be considered as a tool for managing agents as a particular matching policy acts as a guide for decision-making [29]. As an example, policies could direct agents to either cache the necessary information or forward the information without caching. Thus, the behaviour of an agent system can be modified without influencing system architecture.

In a highly dynamic environment such as ANs with varying clients, a policy-based framework would be essential to enable interoperation and to make sure that desired system behaviour is exhibited without conflicts [64]. Policies are well suited in context-aware systems as context is usually complex, changing, and layered [40]. Context information could be expressed as policy conditions in the form of Boolean expressions (for example, *<context variable> MATCHES <context value>*). The modeling of context information thus starts from context variable and context value. Policies are ideal for context modeling as they easily facilitate the implementation of CAS in the underlying networks where policy-based network management is very useful. In such networks, a profile is created to represent the threshold values of a context entity. Each profile indicates the characteristics of a context entity and consists of a basic set of variables that are initially required for unique identification of that entity. Profiles can be divided into user profiles, device profiles, and network profiles, to name a few, and have to be dynamically selected during a context entity association in the overlays space. To effectively handle the wide context knowledge base (KB), manageable context profiles are made to be a part of the KB and would be adapted using policies to suit the current conditions [66] i.e., the existing profiles are updated through the use of differential profiles. The policy conditions and attributes allowed with in a profile are usually selected and managed by a policy/profile manager. This manager would act as a middleware between the operating system layer and the application layer and should be independent of both. Applications could learn from their current systems' behaviour and create policies at run-time based on the current conditions. Thus adaptation is made using either existing policy profiles or instantly created policy profiles.

2.6 Related Work

The following sub-sections discuss related work of context dissemination (and monitoring) under various requirements using various techniques and infrastructure.

2.6.1 Context Management

HiCon [67] by K. Cho, et al. describes a software framework for both monitoring and composing contextual information. Given that a large collection of sensors producing sensor data in real time can overwhelm a system with a large volume of redundant and low-level data, the authors propose a framework for hierarchical context monitoring and composition that scales well across individual, regional, and global contexts levels while allowing dynamic composition of diverse contexts. These three levels have their own components (with processing optimizations), called (i) *PocketMon* (a processing- and energy-efficient personal context monitoring platform running atop resource-limited mobile devices), (ii) *HiperMon* (a high-performance massive context monitoring system responsible for collecting numerous personal contexts, composing, and monitoring the collected contexts in real time, thereby providing the contexts of interest to services effectively), and (iii) *EGI* (a scalable delivery infrastructure to facilitate global context composition which composes an overlay multicast network of EGI nodes widely deployed on the Internet), respectively. HiCon's approach combines context from each of these levels to create a richer and comprehensive set of context-based applications. It facilitates efficient context monitoring and addresses the performance issues to achieve a high level of scalability. Thus, HiCon's major challenges include: (i) support for dynamic composition of diverse contexts covering various scales, (ii) context composition and monitoring be in both horizontal and vertical manners, (iii) reflect the hierarchical nature of networks mapping to the ubiquitous space with different coverage, and (iv) handle the substantial scale of computation for context composition and monitoring.

ACoMS [68] by H. Peizhao, et al. describes a model-based autonomic context management system that can dynamically configure and reconfigure its context information gathering and pre-processing functionality in order to provide fault-tolerant provisioning of context information without relying on redundancy. The functions for context monitoring, processing, and reconfiguring are provided entirely by the context management system and not by the application itself. Furthermore, it is a context model-based approach in which the context models inform the selection of suitable context sources and the dynamic

composition of pre-processing elements. Crucially, this means that the solution retains the considerable benefits of current model-based approaches including support for high-level reasoning and sophisticated programming techniques. The approach uses 'standards' based descriptions of context information sources to increase openness, interoperability, and scalability of context-aware systems.

In line with the ambient intelligence notion [38], reliable, efficient, and adaptive dissemination services for embedded devices [69] and VANETs [21] are also increasingly becoming available. Existing context-aware service and network adaptation schemes employ SIP proxy [70], SLA based CLA [70]--[71], or reinforcement learning [72], to name a few. Interestingly, P. Nurmi, et al. use a social identity theory to find (geo)-locations that are significant to an end-user [73], which is in line with context-based location management theories discussed in [74]--[75]. They deal with complex numerical coordinates by using the notion of *place* (i.e., locations that carry some meaning to user, for example home, so that policies can be applied). They focus on fully autonomous and precise extraction of places from discontinuous (i.e., only when the GSM cell identifier changes) GPS data using a statistical approach (based on the non-parametric Bayesian framework) for identifying places. A multi-dimensional scheme for classifying traffic flows into logical flows based on contexts, which is comparable to our proposed profile-based route learning scheme, is proposed in [74] by R. Ocampo, et al. Not only this location information is practically relevant for end-users, but also consumes fewer resources as unnecessary information is filtered. Location related information has been the most widely studied source of contextual information. With regard to context aware route tracking, H.K. Kim, et al. [75] propose a method based on magnetic sensors which utilize context-awareness to process and track the location of an asset (magnetic line tracer). They utilize the data concerning the context awareness to avoid collisions and analyze to demonstrate location accuracy, which is a problem in current location tracking schemes. The method is 'self-adjusting' to its contextual environment and uses a proximity based approach with the help of agent technology to track and report a line tracer in real-time despite its low processing power. Other related work specific to our real-time application is further discussed in section 7.1.

2.6.2 Middleware

With the increasing demand of context-aware mobile applications, middleware can be used to efficiently develop mobile applications for highly heterogeneous mobile environments. For context dissemination using a middleware, an intelligent context-aware system architecture is composed of a middleware, context server, and client to deal with context-aware application services in pervasive computing environment, as discussed in [76] by J. Chang, et al. The middleware component can define context objects and track a context's current location. It recognizes a moving node by executing an appropriate module according to the context acquired from a context server. Once a connection is established, the moving node determines whether or not the connection between them should hold, by periodic communication. The context server stores context objects and their values depending on a variety of contexts and manages users' current locations by using spatial indexing. The client provides users with context-adaptive application services [37].

The dissemination mechanism discussed in [77] by Y. Luo, et al. employs a Spatio-Temporal approach to selectively disseminate data, but in mobile peer to peer networks. In this approach, the involved nodes are: (i) prioritized based on highest novelty probability, (ii) decentralized, and (iii) stored and transmitted instantly in a proactive fashion. They perform data dissemination by maintaining a *reports database* in every moving node. Pair-wise communication between moving nodes can occur, in the form of reports exchange, when the distance between them is smaller than a node's transmission range. The number of reports exchanged depends on factors such as the moving node's bandwidth, memory allocation, and period of time the transmission range is within limits. The authors define *novelty probability* as the probability that a report at a moving node will be new to other nodes encountered by the moving node after a point in time. Their Spatio-Temporal approach is based on the prediction that the novelty probability of a report depends on some inherent attributes of the report (such as *Distance* and *Age*), which are called the *novelty factors*. Based on the novelty factors, their distributed algorithm runs on each moving node and goes through three phases: (i) Initialization, (ii) Receiving, and (iii) Transmitting, to achieve selective data dissemination. Similarly in [78], a context middleware for adaptive

services in wireless networks extends the SIMPLE [79] service network as the delivery infrastructure. The main reason for middleware employment for context dissemination is to maintain and guarantee data dissemination reliability in critical applications, in addition to reducing a system's overall burden. A context dissemination middleware flexibly routes the modeled contexts to appropriate heterogeneous receivers based on the content, channel, or subject. However, this approach is inadequate for ambient environments due to its high time consumption and mediocre dynamic characteristics. As it widely differs from the process of service delivery, the employed middleware isn't a typical proxy server or clustered backup of servers. The Information Flow Graph (IFG) model [80] by G. Banavar, et al. uses this approach.

2.6.3 ENS, Graph, and Modular -based Approaches

Context dissemination using an Event Notification Service (ENS) normally follows the publish/subscribe mechanism (for e.g., Siena [81] by A. Carzaniga, et al.). These content-based systems offer more flexibility than the traditional publish/subscribe systems, as they allow clients to compose context requests based on the content of published events [82]. Event generators publish notifications to the infrastructure which sends them to event subscribers. Herein, the infrastructure's main functions include notification selection and dissemination. Notification dissemination can be performed using event-servers, shortest-path matching and forwarding algorithms, and network-level multicast communication techniques. In the case of distributed ENSs, servers are interconnected, each serving a subset of the clients [81].

Graph-based abstractions are used to perform graph-based context dissemination in popular projects such as the IFG [80] and Solar [82] by G. Chen, et al. A common approach to solve the basic network connection sharing problem in current context-aware systems and applications is to construct a common context service [83], such as a mediator or base. In such systems, a mediator receives contexts from different sources and disseminates the current context information and changes to appropriate applications. If either broadcast or multicast is used, scalability can be increased by the use of an operator graph. An operator is

an object which accepts subscriptions (event streams) as input, processes them, and finally disseminates a new event stream. An operator graph is a directed acyclic graph of operators formed from the combination of applications' subscription trees. While this approach offers advantages in terms of flexibility and scalability, it also poses the challenge of performing context-sensitive dissemination from a publisher to multiple subscribers.

An open, modular, and scalable system that provides context information to services is discussed in [2] by R. Cohen, et al. Due to the complexity of the dissemination process for next generation services in converged dynamic networks, it's irrational for service providers to handle this task. Therefore, a producer-consumer approach is employed wherein the system entities are either context providers or context-aware services. Producers create complex contexts from raw contexts and thus avoid the need for context processing. The distribution service binds the consumers and producers and distributes the necessary context using a well-defined API, and hence itself manages the quality of context. Herein, the employed API which acts as a context ontology is both powerful and simple enough to provide the necessary context and to be implemented, respectively.

2.6.4 Ambient Networks

In [25], K. Bałos, et al. discuss a context dissemination and aggregation mechanism based on the Jini prototype. Basically, the authors consider implementation of the ContextWare [7]--[8] architecture. Jini is used to implement the ContextWare prototype which offers functionalities including: (i) lookup service (Jini service registry with distributed lookup mechanism found via lookup discovery service), (ii) leasing service (assures that only up-to-date services exist in lookup service), (iii) communication proxies (uses service specific protocol), (iv) dynamicity, and (v) remote event support. The mapping of ContextWare architecture services to customized Jini services is specified and validated. To tackle the mapping concerns and thus the interoperability requirements, the context information is semantically specified using ontologies with context ontology database. The paper mainly focuses on ontological descriptions of contexts by employing *Web Ontology Language* (OWL) instead of the actual ContextWare implementation. As they do not engage

distributed hash tables (DHT) unlike most other related approaches, their Jini based approach offers a much more open and flexible solution which reduces maintenance cost and avoids localization bottleneck as per their evaluations.

A bio-inspired pheromone-based context gathering and dissemination protocol is developed by loosely coupling contexts in [84] by C. Jacob, et al. They focus on heterogeneous and dynamic environments which arise by the convergence of mobile computing systems and smart environments. The protocol has three types of functional components: (i) applications, (ii) context sources, and (iii) context relays. While *applications* query for context information to realize context based functionality on behalf of the user, *context sources* provide context information on demand and *context relays* are intelligent and supportive components. The protocol describes a request-driven model for an intelligent interaction of context providers and consumers in a distributed manner based on the data really requested and needed. To achieve these request-driven loosely coupled interactions [85], it utilizes *semantic web technology* i.e., a semantic data space which acts as a suitable interaction medium that supports spatial and temporal coupling of the three functional components. These components are also utilized to achieve advanced provisioning of context information, context representation, and administration. But it is restricted to a Publish/Subscribe scheme and OWL usage, and it assumes access to a perfect knowledge base. With regard to bio-inspired pheromone-based dissemination, the basic idea is to mark the context information when applications request it. This monitored information is stored in the *request ontology*. Every time a similar request is made, the pheromone density belonging to this request is raised. So the context relay can provide highly important context information as and when requested.

An overlay-based approach for adaptable service delivery that is designed specifically for ambient networks is proposed by B. Mathieu, et al. in [51]. The overlays are termed Service-aware Adaptive Transport Overlays (SATO) and are set up according to service requirements, constraints, and needed service components [50]. The *Ambient Service*

Interface[3] (ASI) [86] is employed to create an overlay network upon a service provider's request and it allows the services to manage "its" SATO. This SATO based mechanism supports dissemination of all services without any restrictions, unlike current delivery networks which are limited to specific services, and can be customized to suit end-user's dynamic context needs. SATO is not restricted to end users and is applicable to the network nodes. The intermediate nodes in the network host SATOPorts, which are the components that process the data. The intermediate nodes can host multiple SATOPorts at a given time, which could be of different types. Also, one or more end users fulfill the role of clients, SATOClient, and one or more take the role of a server, SATOServer. Independent of their role, each node is called a SATO-Node, which hosts specific components, useful for the management of the overlay networks. SATO's dynamic and secure deployment mechanism improves the delivery of adapted services by using *Host Identity Protocol* (using cryptographic identifiers between the application and the IP layer), *Host Identity Tag* (stored in a trusted database), and *Open Services Gateway Initiative* framework (to implement the execution environment). While in [87], a generic design infrastructure supporting the ambient network composition for context acquisition is discussed by T. Szydlo, et al. It employs the default ambient network interfaces and functional entities in two main scenarios - context acquisition from one AN (AN1) and acquisition of context from two composed ANs (AN1 and AN2). As ASI is a standard way to access context information in ANs, it is made available at the same node where the context client resides and it is a point where context data is being used by the client's application. Thus, the client can use a single local interface and communicate with many ContextWare entities located on various overlay nodes. The design is thus restricted to ambient networks only and lacks support for generic context dissemination cases and for futuristic systems, environments, and networks.

[3] The ASI, defined by AN architecture, is used connect external services to existing ANs by providing uniform access to AN functionalities from higher layers.

2.6.5 Overlay Networks and Content-based Routing

Semantic casting using overlays eliminates the need for content filtering brokers, semantically splits data based on their content, and propagates them across independent dissemination trees *i.e., semantic multicast channels*. SemCast by O. Papaemmanouil, et al. [55] based on *content based publish/subscribe model* [88]--[89] is intended for high volume and highly distributed data dissemination. In order to design a content-based pub/sub infrastructure on top of the standard communication and programming model provided by common structured overlay networks, they address two main issues – (i) the gap between the rich language that describes an subscription event and the single key identifier used to route a message in overlays, and (ii) the inefficiencies due to implementing 1 to N communications on top of the overlay unicast connections. A distributed content-based pub/sub system comprises a set of nodes, each of which can act as a producer / consumer of information, playing the role of publisher / subscriber. In SemCast, a cost model is used to decide how many channels to use as well as the contents and destinations of each channel. The model uses (i) stream contents and rates, (ii) profile characteristics, and (iii) network locations of the message destinations. Each channel is implemented as a multicast tree, and clients subscribe to the channels by joining the corresponding trees. Semcast uses statistical and syntactic information to updates its channels. While in Overcast [18] by J. Jannotti, et al., single source multicasting is achieved using data distribution trees that adapt to changing network conditions. Using this distribution tree, Overcast provides large-scale, reliable multicast groups, especially suited for on-demand and live data delivery. To support dynamic and fast adaptation, it tracks the global status of a changing distribution tree.

Similar to SemCast, NICE [90] by S. Banerjee, et al., is specifically intended for data stream applications with large receiver set and low bandwidth. This comparable strategy is taken up in NICE wherein end-to-end latency is used as the distance metric between hosts during grouping. The NICE protocol arranges end-hosts as a hierarchy, which implicitly defines the data paths. The member hierarchy is crucial for scalability, since most members are in the bottom of the hierarchy and only maintain state about a constant number of other members. Logically, each member maintains state about other members that are near in the

hierarchy, and only has limited knowledge about other members in the group. The hierarchical structure is also important for localizing the effect of member failures. The simulation results show that the NICE protocol has lower link stress (by about 25%) for groups of size 32 or more, improved or similar end-to-end latencies, and similar failure recovery properties when compared with Narada application-layer multicast protocol over Internet-like topologies [91]. Another related overlay adaptation based technique, termed oEvolve, is discussed by Y. Zhu, et al. in [92]. It is a distributed overlay algorithm that maximizes bandwidth by constructing an overlay data dissemination topology that progressively evolves and adapts. It is based on 'on-the-fly' metrics and takes into consideration the topological variability and link correlation properties of overlays. It sends data on the current topology, while continually updating the topology by adding successive trees spanning only those receivers with sufficient bandwidth. The topology is thus evolving based on inferred knowledge of the network, by continuous observation of performance in the current topology.

To facilitate content routing over mesh-based overlay networks to heterogeneous clients, application-level XML routers are used to forward data streams in the form of XML packets, such as in [93] by A.C. Snoeren, et al. These routers disseminate individual packets either to clients or to other routers using their own novel protocol called the Diversity Control Protocol (DCP). DCP works by reassembling a received stream of packets from one or more senders using the first copy of a packet to arrive from any sender, thus increasing reliability when compared with single-sender approaches. Their approach is to assign a "monotonically increasing 32-bit application serial number" to every DCP packet when it is created at a root router. Every router that forwards these packets maintains the last packet serial number sent on each output link. This information is included in the next packet along with the next packet's current serial number. By doing so, a receiver is permitted to reassemble the stream of packets from a sender in the presence of missing serial numbers. This approach is more reliable and dynamic than tree-based networks as the distributed network of XML routers stream real-time contexts. Hence, this approach aids to achieve context dissemination in varying heterogeneous domains. In the case of XTreeNet [94] by W. Fenner, et al., scalable overlays for XML based content dissemination is proposed by

unifying the publish/subscribe and query/response models. This is achieved by using a common XML aware overlay network made up of XML routers that connect content producers and consumers. XtreeNet employs network-based identification, exploits overlay multicast, and introduces the key concept of content descriptors (e.g., ontology based hierarchy for data matching) to ease the problem of matching data generated by producers with consumer's interests. The content descriptors work by permitting each data entity generated by a producer and each consumer query filter to be independently mapped to sets of content descriptors, which are represented as fine grained core-based distribution trees. A data entity is said to match a query filter only if their respective sets of content descriptors have at least one content descriptor in common i.e., an overlap between the sets of content descriptors means that a data entity matches the query filter. Therefore, for each content descriptor, XtreeNet constructs two multicast trees, wherein one leads to all of the producers and the other leads to all of the consumers. Relatively, C. Ma, et al. propose a battery-aware routing protocol for data streaming in [22], which is *Wireless Sensor Network* specific and supports large volumes of data. The routing protocol is sensitive to the battery status of routing nodes and avoids energy loss. Their main idea is to select the *well recovered* sensors as routers and the *fatigue* sensors for recovery. By dynamically scheduling routing paths to efficiently recover the node battery capacity, they minimize the discharging loss on sensor nodes and maximize the lifetime and data throughput between a source-destination pair. This results in significantly prolonging the lifetime of batteries in comparison with existing routing protocols for data dissemination.

2.6.6 Fault Tolerance

Existing wireless service composition architectures did not take into account the issues of fault tolerance and context awareness, which are especially important while dealing with mobile devices [95]. The simple and common approaches for handling system failures are very inefficient in contextual environments due to (i) unrealiability between context entities, (ii) increased traffic overhead for message checking, (iii) complexity, and (iv) operability only during process-based service description. In UbiSrvInt [96] by F.Y. Chen, et al., service execution is delegated to low-fault correlation peers. This reduces the percentage of

failures and improves the system efficiency. By eliminating the use of a central node, single point of failures are avoided. The proposed fault-tolerance module provides capabilities for preventing failures in advance and recovering failures once any component fails. Among the common redundancy based approaches to tolerate faults and dependability, physical (i.e., service components) redundancy is believed to be the most appropriate due to the dynamic nature of the context information. UbiSrvInt infrastructure is mainly composed of the following components: (i) Reasoning Agent (responsible for receiving and interpreting service requests for its user), (ii) Sub-Tasking Agent (decides how to segment the task received from the reasoning agent), (iii) Discovery Agent (responsible for discovering all services who functionalities are listed in a subtasks list), (iv) Execution Agent (returns the execution result of a subtask as a service is sent to the one of dispatcher peers), (v) Composition Agent (responsible for managing the order of integration of the executed services), and (vi) Fault-Tolerance Module (the main component inspired by Introspective Failure Analysis [97]). The fundamental assumptions of this infrastructure include: (i) a stable and reliable wireless ad-hoc networking technology is deployed, (ii) heterogeneous devices can effortlessly connect to each other through the communication gateway, and (iii) mobile services can be unfolded on the heterogeneous devices through transformation between different forms.

As surveyed in [98], user transparency is vital while detecting and tolerating faults. The effects of system faults stretch beyond immediate consequences, as one fault leads to another, resulting in excessive utilization of resources in resource-constrained environments and sensing contexts inaccurately. The commonly existing fault-tolerant approaches that are applicable in systems considered in this research work include: (i) surrogate application usage, (ii) alternate fault notification mechanisms, (iii) handling context sensing errors, and (iv) N-version approach.

2.6.7 Policy Profiles

End-systems enable application-based customization and can improve the overall performance by informed virtual collaboration with each other. For example, in profile-

driven XPORT (eXtensible Profile-driven Overlay Routing Trees) by O. Papaemmanouil, et al. [99], a generic tree-based data dissemination system that supports an extensible set of data types and profiles and an optimization framework that facilitates easy specification of a wide range of useful goals are discussed. The implemented tree-based overlay network can be customized as per the target application by using a small number of methods that encapsulate application-specific data-profile matching, profile aggregation, and cost optimization logic. Data-profile matching eliminates the flooding problem, while the aggregation model specifies the system cost. With regard to extensibility, XPORT modifies the overlay tree using local transformations to adapt to changing network and workload conditions affecting the system's performance. XPORT uniformly and easily supports disparate dissemination-based applications, such as content-based data dissemination and multicast-based content distribution. Their simulation results show that, while XPORT's initial performance is lower than the star topology, after a small number of transformations, the routing tree converges to an optimal tree.

In [14], by G. Gehlen, et al., a mobile web service framework and a rule description language are employed to construct a context dissemination middleware optimized for reduced network load and latencies. The proposed middleware is based on the concept of the Service Oriented Architecture and is implemented using XML Web Services technologies for mobile devices. It eases the development of context aware applications by providing an abstraction layer hiding complex protocols. The policy-based rule description language has been designed, developed, and structured to provide monitoring service of arbitrary context information. Their system architecture does not suit ambient network characteristics (particularly heterogeneous composition) as they employ a context broker using web services eventing. These service components are Subscribe, Unsubscribe, GetStatus, and Renew. At the client side, proxies of these services exist and the client can receive the notifications through a notification service.

A multi-dimensional scheme for classifying traffic flows into logical flows based on contexts, which is based on profile-based learning, is proposed in [74] by R. Ocampo, et al. They consider flow as an entity of interest and consider its context as any information that

may characterize the situation in which the flow was generated, transmitted, processed, or received. Location related information has been the most widely studied source of contextual information. Not only this location information is practically relevant for end-users, but also consumes fewer resources as unnecessary information is filtered. In order to classify flows based on context into real and practical implementations, they demonstrated the use of ontologies to formally model flow context for software design purposes and as a vocabulary for runtime context exchange and processing.

2.6.8 Casting

As a globally deployed multicast service does not exist and as multitudes of problems such as manageability, lack of a robust inter-domain multicast routing protocol, scalability, and heterogeneity exist, an efficient casting process is essential. In Scattercast by Y. Chawathe, et al. [100], a set of ScatterCast Proxies (SCXs) organize themselves into a data delivery tree. A strategically placed network agent that collaboratively provides the multicast service for a session for nearby located clients is called a SCX. Unicast interconnections are established between and within SCX and clients. The problem with Scattercast is that it does not organize the tree based on the relative importance of the SCXs. Instead, clients have to explicitly register with SCXs to receive multicast data. It assumes the existence of a cluster management platform that provides the function of creating SCXs when required. Both SCX organization and grouping of clients with SCXs takes place using a topology management protocol called the Gossamer protocol, which is proposed by the authors themselves [100]. This simplistic protocol has the danger of resulting in excessive network load near the source due to duplicate packets. Thus, the goal of the Gossamer protocol is to build a degree-restricted spanning tree across SCXs while at the same time keeping the average delay between the source and all destinations at a minimum. SCXs also run a variant of a distance-vector routing protocol to build reverse-shortest-path trees. Their simulation results show that the latencies incurred by transmitting data over the Scattercast connections are typically within 1.6 to 1.9 times those associated with directly unicasting or multicasting the data from the source to the various destinations. But the structure generated

by the Gossamer protocol substantially limits the bandwidth usage of the physical Internet links.

In the case of the Narada protocol by Y.H. Chu, et al. [91] the authors explore *End System Multicast* instead of traditional multicasting. In end system multicasts, end systems implement all multicast related functionality including membership management and packet replication. The Narada protocol, for instance, focuses on end system based multicasting for conference-type applications, wherein a standard multicast routing protocol runs on top a mesh structure, but lacks in cost and load properties. The end systems organize into an overlay structure using a fully distributed protocol and they try to optimize the efficiency of the overlay by adapting to network dynamics as a two-step process. First, it constructs a mesh and tries to ensure that the mesh has desirable performance properties. In the second step, Narada constructs spanning trees of the mesh wherein each tree is rooted at the corresponding source using common routing algorithms. The key concern is the performance results (with respect to end-to-end delays) of such a model due to the introduction of duplicate packets on physical links. Their Internet experiments are conducted on a wide-area test-bed of twenty hosts and the results indicate that end system multicast can achieve bandwidth performance comparable to IP Multicast, while at the same time achieve mean receiver latencies that are about 1.3 to 1.5 times latencies seen with IP Multicast. Their simulation results conclude the same. Narada nodes maintain global knowledge about all group participants, which is not feasible in an ambient network scenario.

2.6.9 Comparison of Related Approaches

In this section we first compare the basic traditional context dissemination approaches and discuss their advantages and constraints. The evaluation is done with respect to the challenges and objectives discussed in section 1.3. The results are summarized in Table 2.1. With respect to the evaluation we conclude that the demand for context dissemination mechanism for complex context-aware services is immense, as none of the surveyed approach's applicability is entirely satisfactory. As achieving context dissemination in a

balanced, adaptable, and interoperable manner in a heterogeneous infrastructure is difficult using existing dissemination solutions, we propose a contextual overlay based approach to disseminate contexts between active context entities.

Table 2.1 Evaluation of basic context dissemination approaches.

Approach / *Properties*	Middleware-based	Content-based using ENS	Graph-oriented	Content-based using XML
Adaptability	++	+	++	-
Interoperability	-	-	-	-
Inference	-	-	-	+
Content-based	+	++	+	++
Delivery-option	hybrid	push	push	Push
Periodicity	hybrid	aperiodic	aperiodic	Aperiodic
Spontaneity	-	+	-	+
Applicability	+	+	+	+

Using the same set of challenges and objectives, we then compare selected specific recent dissemination approaches (in Table 2.2) in order to compare each other and demonstrate their advantages and constraints with respect to our requirements.

Table 2.2 Evaluation of specific context dissemination approaches.

Approach / *Properties*	Jini-based	Bio-inspired	SemCast	SATO	XPORT	Scattercast
Adaptability	+	++	++	++	+	++
Interoperability	++	-	-	+	+	+
Inference	++	++	+	+	++	-
Context-based	++	++	+	++	+	-
Delivery-option	push	pull	push	Hybrid	push	Hybrid
Periodicity	aperiodic	aperiodic	periodic	Hybrid	periodic	Periodic
Spontaneity	-	-	+	+	-	-
Applicability	++	++	+	++	++	-

The relevant alternate schemes and theories that we considered, as a part of our research, for the purpose of context dissemination include:

- *OWL Ontology*: In general, ontology is defined as a formal representation of a set of concepts within a domain and the relationships between those concepts. Contexts and ontologies play a crucial role in knowledge representation and reasoning [25], [84], [71]. Therefore, with respect to context awareness, ontology captures the context of use in which a user is interacting with a particular computing platform in a given physical environment in order to achieve an interactive task. The (context) ontology is formally specified in the OWL.

- *Context Level Agreements (CLA)*: CLAs are used to negotiate context specifications according to specific ontology-based models [71]. They provide a mechanism for C_R to state their requirements and for C_S to offer dynamic customized context information. CLAs can also be used across various domains to provide seamless contexts in varying environments.

- *Quality of Context (QoC)*: Quality of Context is a type of context attribute that allows common interpretation of the context, independent of the semantic interpretation of the context information. QoC is essential to reduce the gap between existing systems and real-time applications. Ontologies with related models and metrics are required to specify the QoC data such as aspects of degradation, consistency, etc.

- *Active Event Channels and Templates*: Consumers and sources could engage in one-to-one event exchange dialogs over active event channels that designate named instances of particular Quality of Service templates, which correspond to system-level plug-ins that execute in the overlay nodes of the event dissemination overlay network. This event channel, one hop at a time, could multicast registry items and queries away from a source or unicast replies towards a destination. This concept can be extended to achieve restricted access controlled context dissemination.

- *Ambient Virtual Pipe (AVP)*: Ambient virtual pipe platform [101] provides a flexible approach towards the creation and management of autonomic, secure, self-adaptable,

service-aware overlays in Ambient Networks. The platform uses an action object base that enables flexible management of potentially conflicting adaptation policies and provides a secure and Quality of Service assured environment for network composition in ANs. This concept can be extended to use a cache overlay mechanism with dissemination brokering.

2.7 Summary

This chapter presented an overview of background concepts that are related to our context dissemination approach. The chapter mainly focussed on introducing the thesis' main theories, including *Context Awareness*, *Ambient Networks*, *Overlay Networks*, *Agent Technology*, and *Policy Profiles*. In addition, it surveyed and reviewed recent related work of the above stated theories with respect to context dissemination, and compared selected related work with each other. The next chapter introduces the context dissemination approach and discusses the context dissemination protocol's high-level architecture.

CHAPTER 3
Context Dissemination Infrastructure – Overview and Architecture

This chapter first defines the main terminologies used. It then overviews the context dissemination concept and discusses its features with regard to ambient networking. The chapter describes the high level architecture of the context dissemination protocol and ContextWare mirroring, which are sub-divided in to four design model chapters. The chapter also briefs an example usage scenario of the context dissemination protocol which will be used and discussed in the subsequent chapters.

3.1 Terminologies

This section briefs selected terms that are used frequently in this thesis as follows:

- ***Context Entities*** denote the various context sources (i.e., producers) and context requestors (i.e., consumers).
- ***ContextWare:*** Ambient network's middleware model for context processing.
- ***Context Dissemination Processor*** is a functional area (within ACS after AN integration) which spontaneously processes modeled context information into

overlay nodes in order to be disseminated. It controls the Context Specific Overlay Networks (CSON).

- ***CSN and RSN:*** Context- and Request- Specific Nodes first represent a context profile (illustrates context's properties) and are later converted into agent tags and associated in the overlay space as overlay nodes.

- ***Agent tags*** are entities that represents a person, an organization, or a software application, which independently or by interacting with other agents executes a task or set of tasks. Agents can be created, moved, cloned, and destroyed.

- ***Agent Object Subscriptions:*** The agent tags are represented as *Agent Object Subscriptions* within the agent manager. Each of these subscriptions act as a context router when associated in the overlay space.

- ***Mobile Agents*** are the agents that can dynamically move from one location i.e., agent environment to another, under their own control to perform tasks.

- ***Proxy (Agent)*** is a system or application that is authorized to act on behalf of another [102].

- ***Policies*** are rules and configurations set by users or administrators. A particular matching policy acts as a guide for decision-making.

- ***Overlay Space:*** The overlay network wherein the context entities are associated before being further processed for Unicasting.

- ***CSON:*** The Context Specific Overlay Network, clustered semantically and changing dynamically based on underlying network / user requirements, categorizes context information which is dynamically provided to appropriate target users or applications.

- ***RSON:*** The Request Specific Overlay Network deploys a different overlay network for every unique context dissemination process. These overlays are destroyed as soon the dissemination service is performed.

- ***Leader Overlay Nodes:*** The varying cluster heads of the Context Specific Overlay Networks.

- ***Casting:*** The process of virtual coordination of the context entities in the interaction layer.

- ***Learning Engine:*** The dissemination protocol's feature set to achieve learning-based intelligence and adaptation.

- ***Hierarchical arrangement/ levelling:*** The context node types, sub-types, and special types are levelled hierarchically in such a way that look-up between the types is feasible, thus enabling internal localized communication.

- ***Knowledge Base:*** A context-aware system's data repository.

- ***Routing Table:*** It is a local request forwarding table maintained by the CDP lookup service which caches overlay operation.

3.2 Dissemination Overview and Features

Information dissemination and the issues involved to carry this out in a mobile environment is not a new topic of discussion. In fact, Bill N. Schilit and Marvin M. Theimer discussed about *Disseminating Active Map Information to Mobile Hosts* [11] a decade ago. They propose dissemination using a basic unicast design, a single multicast channel, and multiple multicast channels. Many basic dissemination based applications including videoconferencing, event notification, e-newsletters, traffic information systems, and update services frequently disseminate information to multiple interested entities [44]. Today, (context) dissemination is widely researched as a part of context awareness, as a whole, in

the background of mobile and dynamically created networks i.e., from a network point of view. The process of context dissemination, at times, could be misunderstood with service delivery. Unlike context dissemination, service delivery denotes just the rapid creation and deployment of new services. Context dissemination, in many cases is considered in conjunction with other concepts such as context acquisition and context modeling i.e., refinement of context information, which are discussed earlier in section 2.1. A considerable number of research projects are considering context dissemination as a part of context-sensitivity and situation-awareness. The key concepts that are studied for achieving context dissemination, for general information dissemination, includes but not limited to:

- Broadcasting, Multicasting, & Unicasting
- Context agreement negotiations and Conflict resolution techniques
- Middleware-based approach
- Event-notification using pub/ sub mechanism
- Information-flow and Operator graphs
- Content-based routing
- Dissemination in Ambient Networks

Their related work was discussed earlier in section 2.6. Context dissemination in general can be defined as to how context information is distributed among interested context-clients (i.e., context-requestors, C_R) from context providers (i.e., context-sources, C_S). As a context client itself can dually act as a context provider, several uncertainties exist. As a solution, we require a user-friendly and adaptable communications system built in a user-centric manner. Herein, user or device specific information, such as capabilities and preferences, could be easily communicated across networks in order to achieve the necessary goals with no active user involvement. The way context dissemination takes place depends upon several parameters, such as: the used device type, the current connectivity, bandwidth available, service priority, etc [2]. Therefore, a controlled dissemination of user contexts is essential. The dissemination process's performance generally depends on factors such as context availability, probability of correctness, trust-worthiness, resolution, and validity. The context dissemination process is an important component to enable context awareness and could be event-based (i.e., on-demand) or flooding-based (i.e., push broadcast). Spontaneous dissemination of modeled

contextual information from C_S to C_R is essential to continuously adapt the transformations and be in line with the target application's dynamically changing requirements [37]. Examples of efficient mechanisms for disseminating context information are the push mechanism, pull mechanism, tuple shared space, polling, and persistence mechanisms. Context-sensitive inter-object communication is essential for achieving the processes mentioned previously in the order of relevance in a user-centric computing and communications world. The process's facilitation in ANs and context-aware ambient applications is significant to maintain scalability and dynamicity. The complexity of uncertainties and the level of queries in the process of context dissemination increase along with the current network growth.

While context information can be categorized into various classes based on physical and collaborative context factors, each requires a specific handling technique. For instance, private contexts cannot be disseminated to everyone. As context dissemination deals with seamless provisioning of information to entities based on their situation, all questions with respect to why, what, when, where, and how to disseminate, are significant. This is very true from an ambient network point of view. Also, any policy based system must have some type of ontology that defines what can be specified by a policy and how it can be specified. The main cause to perform context dissemination is to provide and update information that is actually useful to the users, hosts, domains, and applications. For example, user's awareness of the surroundings could be improved and applications could automatically adapt based on the information disseminated i.e., be more responsive. This answers the question of *why* dissemination is necessary. *What* actually should be disseminated depends upon the entities' requirements based on the current situation. For instance, the information and meta-information of contexts (for example, time, location, etc), network-related information such as network characteristics, and the rules or policies governing the inter- and intra-domain interaction operations could be disseminated. But, how can a system/service know when and where to disseminate? *When* appropriate conditions and timings are met, the (context-sensitive) dissemination process would be triggered. For example, depending upon the point of trigger, the actual dissemination process could either be a reactive or proactive action. *Where* should the information be disseminated again depends upon the entities that actually

benefit from the process of dissemination than without it i.e., data is disseminated to individually rational entities. The entities, for example, could be end or intermediate terminals such as context clients, mediators, etc. For example, in Xerox Parc's distributed architecture [103], the decision is made based on the user's i.e., publisher's policies. Finally, *how* to disseminate depends upon the combination of all four questions discussed above i.e., it takes why, what, when, and where to disseminate as inputs and requires state-of-the-art algorithms and protocols for output. For example, content-based routing and content-matching algorithms could be employed, but need further development. How can this dissemination process be performed precisely from a myriad of today's contexts is complicated and incoherent. Finding the solution for the question *how can the dissemination process be performed precisely* is a significant research challenge as the novel algorithms should consider both 'context-driven' or 'context-sensitive' networks[4] and provide solutions for scalability, wide coverage, adaptation, correctness, and fault tolerance [95], to name a few. Before performing the actual dissemination, the user's current location and preferences may need to be considered depending on whether the network is 'context-driven' or 'context-sensitive'.

Any CAS performing information dissemination should be normally capable of achieving data propagation in either of the following mechanisms [44]. Also, it is suggested that the system should support more ways of data propagation so as to be optimized for various criteria. This is briefly illustrated in Figure 3.1. The different data delivery options are discussed as follows:

1) *Push mechanism:* The data is disseminated to the clients in advance of any specific request in the *push* mechanism. Since data transfer is initiated by a server or base push, it is termed as push-based dissemination.

---

4 While context-driven networks depend on up-to-date information which could change in a rapid and unpredictable fashion, a context-sensitive network depends on changes in the context information i.e., disseminates the required information updates based on the changes.

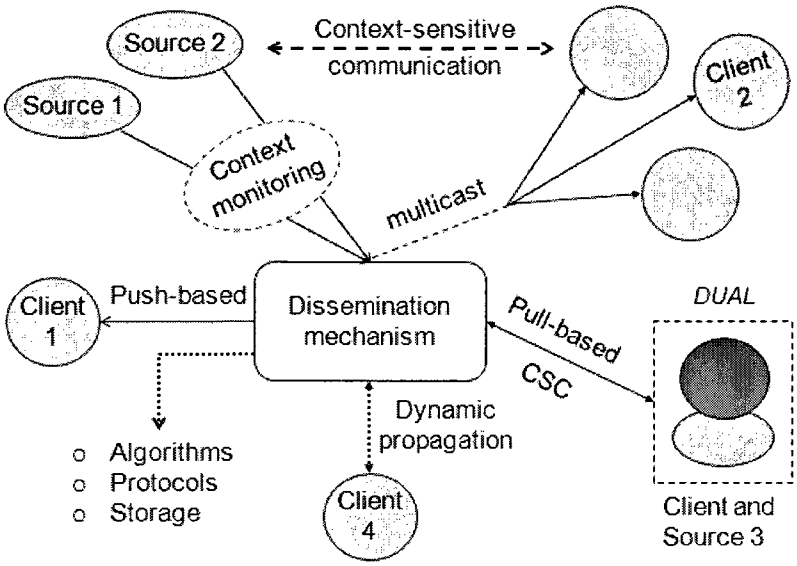


Figure 3.1 Illustration of possible context dissemination approaches.

2) *Pull mechanism:* The data is disseminated to the clients only after receiving the requests from them and locating the requested information. Here, the dissemination is initiated by a client *pull*.

3) The push- and pull- based dissemination mechanisms are the two main efficient mechanisms for collecting and disseminating context information. By combining both push and pull mechanisms, a multicast protocol could adjust to dynamic components with varying speeds. Pull-based mechanisms have several issues such as frequent server interruption, delivery scheduling flexibility limitation, etc, and most of the issues are avoided in push-based mechanisms. The main challenge in push-based dissemination is the problem of deciding which set of data should be disseminated to clients without any pre-known requests, and its sub-problems.

4) *Periodic vs. A-periodic:* Both the above discussed mechanisms can be performed in either periodic or a-periodic mode. Regular repeating dissemination schedule is termed periodic, while vice-versa symbolizes the a-periodic approach. Thus, in a-periodic push the server would itself push meaningful data to intended clients without any fixed schedule. This method of data dissemination is widely employed

and effective in ubiquitous computing. For example, event notification based pub/sub systems employ this type of dissemination.

5) P-to-P vs. 1-to-N: The final characteristic depends upon the client's interests. In P-to-P, information is delivered from one data source to another data client, while in 1-to-N, either broadcast or multicast is performed. Using the latter mode, scalability can be increased, provided such a situation exists. If employed in vice-versa situation, scalability would in fact be decreased. Considering the characteristics of ANs, multicasting would be more typical than broadcasting, and hence increased reliability can be achieved.

The main factors and characteristics of context dissemination in ANs are discussed as follows. Considering the characteristics of ANs, dissemination in ANs is more than just forwarding or routing. The basic requirement in any AN regarding contexts is consideration of the characteristics of the involved network and support for a common framework for context awareness across all functions in the ACS. As the dissemination protocols should provide solutions for scalability, correctness, etc in both context-driven and context-sensitive networks, the framework used should be as a whole be precise and consistent. When ANs compose with each other, a trust is created between the composed parties. The security and trust framework within the AN architecture should support multi-domain requirement, wherein multiple ANs should negotiate contexts and form agreements. By employing the context profiles along conflict resolution techniques, the framework can be enhanced by building relationships between the involved context sources and context clients i.e., context-sensitive dissemination. In order to achieve network robustness and fault tolerance while disseminating contexts, single point of failures should be avoided [95]. The dissemination protocol's design should also allow ANs to properly perform its tasks even when certain AN components have failed or when it is isolated from the rest of the network. The dissemination protocol should support a simple plug and play connection between ANs. It should be backward compatible as ANs, by default, do not provide the full functionality in legacy network infrastructures. By featuring an adaptation layer, we can provide new functionalities and services to legacy networks. This interoperability is essential to support and provide context information that is actually demanded by the users. An outline of the

basic dissemination process along with other context management processes in an AN is shown in Figure 3.2. In the illustration, we consider three ANs wherein AN #2 gets raw contexts from its context sources and from context sources in AN #1. Once these contexts are processed using context-aware techniques (with aperiodic updates), the AN #2's Context Coordinator and Manager component performs the actual context-based dissemination process to propagate selected context information to its context clients and to the context clients in AN #3. As an additional option, the clients could send content-reduction policies along with their profiles, as in PACK [104]. They basically require a constant flow of information in order to adapt to their changing context [83]. In general, as computing in ANs can be viewed as extended Ubicomp, push-based distributed dissemination (using distributed CIBs) needs to be widely employed unlike context-aware middleware architectures. As the information flow in an AN is 'emergent' and as the client's attributes are itself derived from the information content, the dissemination process is policy-specific and content-based in these networks [12]--[13]. The employed protocols should be capable of achieving context information dissemination in a rapidly changing time-constrained heterogeneous domain.

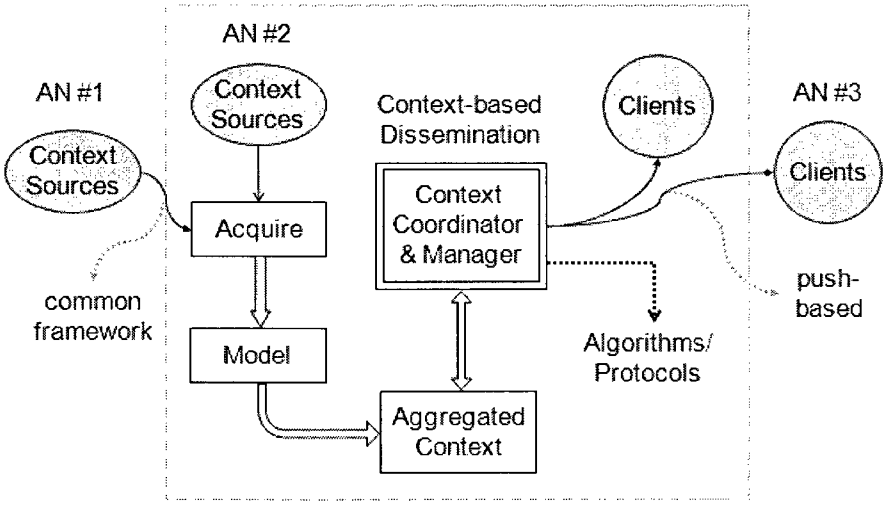


Figure 3.2 A simple dissemination process setup in an ambient network.

3.3 High-level Architecture

A high-level view of the context dissemination architecture is schematically illustrated in Figure 3.3. The engaged external entities including the data repository (via database listener) are also illustrated in the figure. First off, the various context sources and context requestors could be from several physical network entities, applications, etc. As we do not base our processing based on the type of context or the source of context, our architecture do not initially prioritize or distinguish between C_S and C_R while they are being processed and associated in the overlay. This is in-line with ambient network's heterogeneity characteristic. Instead, any and all types of context information from various context entities are processed. The Context Dissemination Processor is responsible for processing the context entities by handling heterogeneous contexts and their respective profiles to attain network interoperability (i.e., CDP-L1). The processing functionalities include context-based aggregation, inference, and translation to attain network interoperability. The CDP views all context entities that are involved in the dissemination process as overlay nodes in the form of agents, resulting in context agents. The agents are managed by the *agent manager* with a look-up service. The CDP controls the Context Specific Overlay Networks (CSON) and manages their creation and controls the adaptation of existing CSONs. As this overlay network creation is a logical concept, its composition has virtually no limitations and can be changed easily by the CDP. As the CDP itself is split into multiple processing and optimization levels, some processes and optimizations might fail but not the entire CDP. To strengthen CDP's protection and avoid single point of failures, hybrid (both vertical and horizontal) composition strategy is adopted, and due to pure AN involvement, every involved AN must be capable enough to run its own CDP.

Our overlay adaptation principles are based on a straightforward solution i.e., (i) initialize the overlay with an initial topology, (ii) dynamically collect context entities, (iii) re-organize the initial topology based on the properties of these context entities, and (iv) adapt the overlay with the new topology. The dynamically collected context information is also monitored by periodically collecting related context information to improve overlay topologies. The CDP also controls the addressing and routing on the overlay network via its

efficient lookup service. The Pure Overlay Space design acts as an interaction medium with regard to space and representation and adopts a completely logical overlay capable of dynamic adaptation without any physical relationships. The efficiency of an overlay topology usually depends on the structure of the underlying physical topology, but we try to avoid this fact in our Pure Overlay Space. As context's properties are viewed as an agent's attributes, their spatial and semantic association is a key point (as just using object-oriented programming languages is not sufficient as they lack expressiveness). The overlay nodes acting as contextual agents initiate spatial decoupling and semantic clustering. Herein, decoupling denotes the uniform dynamic distribution of the overlay space and clustering denotes the semantic overlay network formation. The context agents are uniformly distributed in the spatially decoupled overlay space. The agent manager represents an interface between the resulting agent object subscriptions and overlay associations. Based on the properties associated with the agents that are associated in the overlays, those overlay nodes self-organize based on their properties to form overlay topologies. These overlay nodes are specialized AN nodes that poses specific characteristics to be a part of CSONs. Even though we consider both vertical and horizontal AN composition, the context dissemination specific components are more suitable during vertical network composition. A single CSN can provide the necessary contexts for several RSNs at the same time, and a single RSN can be served by several CSNs at the same time. The POSpace dynamically (re-) configures itself with the aid of the CDP to adapt the path of context dissemination i.e., the CSONs change their structure based on the changes in the underlying networks, users, and policy profiles (to accommodate mobility, composition of networks, or congestion control). This will change the topology or properties of the overlay network, which might now include new overlay nodes. The efficient design of agents represents effective overlay maintenance with CSONs and RSONs. As a part of the POSpace, the CDP (i.e., CDL-L2) also performs overlay-based optimization schemes such as *dynamic context scoring* (keeps track of the state of contexts in the overlay space) and *context benefit fairness* (maintains node fairness, instead of maximizing its own benefit) based on context request's current state of affairs. As a part of the architecture, the figure also shows multi-level overlays i.e., CSONs and RSONs. We tend towards a context-specific overlay infrastructure for virtual interaction between context sources and requestors as it is necessary for the POSpace to be

efficient for effectual dissemination of context information, and hence context-based applications in ambient networks. More specifically, the CSON-D protocol achieves up-to-date context dissemination (i.e., spontaneous dissemination of latest contexts) by constructing and maintaining a two-level overlay infrastructure using minimal casting functionality. It is made up of decomposed hierarchy of sub-nodes in the first level, followed by fully mapped sub-nodes in the second level (i.e., RSON). A single overlay node can be a part of multiple CSONs at the same point in time. The re-configuration or adaptation and fault-tolerance of the CSON/RSON imply optimization of the network according to the target user's or service's demands. This is represented as 'Adaptation and FT modules' in the figure between the multi-level overlays and CDP-L2. The figure also shows our contribution with regard to utilizing the context dissemination infrastructure to develop a real-time application by means a context monitoring scheme with adaptive routing and a personalized learning engine.

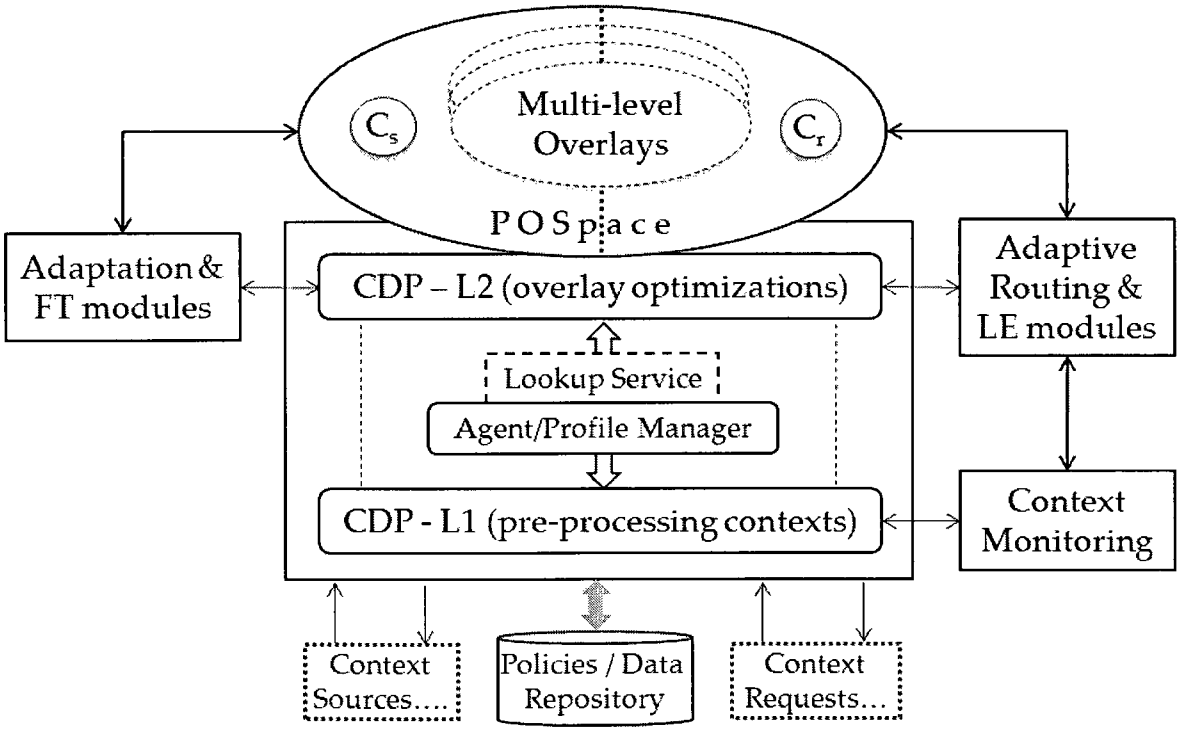


Figure 3.3 Context dissemination protocol - High-level architecture.

To summarize, the overall functionalities of the CSON-D protocol is divided into (i) CDP, (ii) POSpace, (iii) Multi-level Overlays, and (iv) Context Monitoring with Learning-based Adaptive Routing. These functionalities are elaborated in Chapters 4, 5, 6, and 7 respectively, and their real-time usage is elaborated in Chapter 7 as well.

3.4 ContextWare Mirroring

Ambient networking aims at providing an internetworking solution for the integration of heterogeneous networks, based on the dynamic composition of those networks. Its ContextWare architecture, a model for context processing, deals with sources of context localization and identification, acquires context data from various sources, distributes contexts to interested context entities, and caches and stores contexts in databases. As (i) ContextWare is a middleware model for context processing that does not rely on any particular implementation technology, (ii) context information has to be provided explicitly unlike human interaction, and (iii) applications need a time-constraint flow of context information: a context dissemination mechanism with above discussed requirements is essential. Hence, a novel overlay space (i.e., the POSpace) that supports heterogeneous composition and precludes service brokers is developed on top of traditional overlays. The selected specifics of ContextWare include:

- C_S should be localizable and independent i.e., distinct technology can be employed for implementation and communication with C_R.
- Direct communication between C_S and C_R should be provided.
- C_R should be dynamically equipped with a suitable functionality to communicate with the selected C_S with suitable protocol and communication pattern.
- Context sources could be grouped and combined into one complex context data. The complex information should not be any different than a standard C_S.
- The involved context entities should be aware of the dynamic nature of the context information provided or requested by other context clients.
- The architecture's main functional area, ConCoord, should be scalable, distributed, fault-tolerant, and secure.

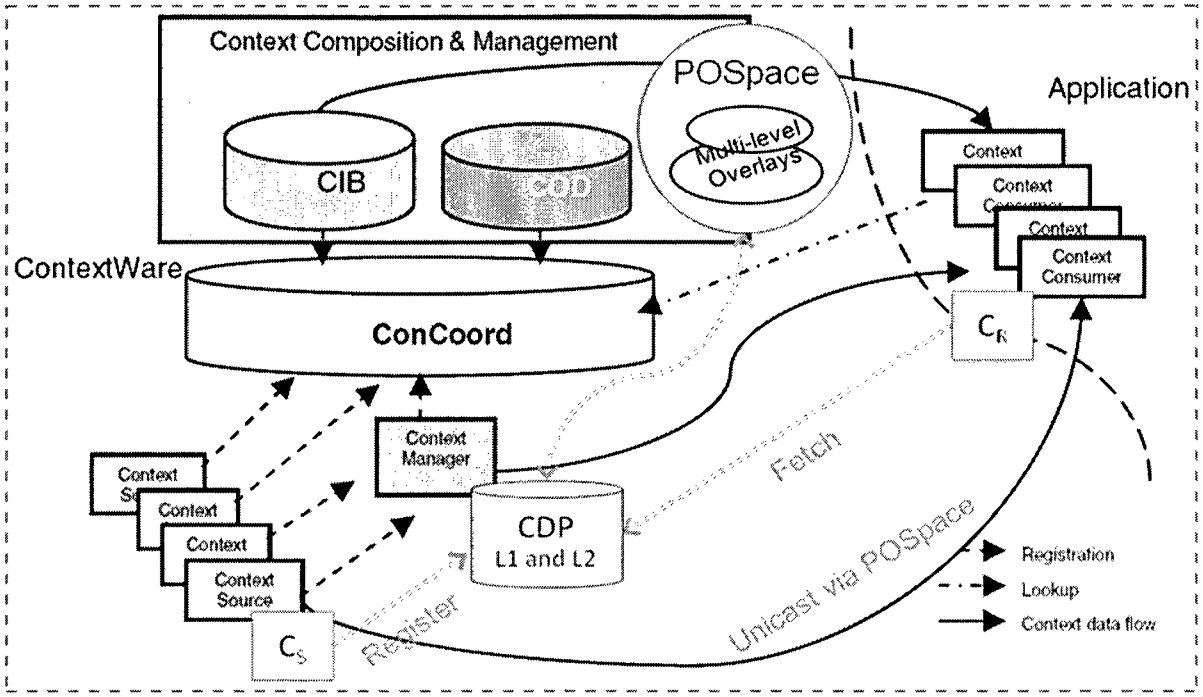


Figure 3.4 The ContextWare architecture (mirrored with POSpace-based dissemination mechanism).

With regard to mirroring (refer Figure 3.4): (i) the context sources and consumers in ContextWare are represented as C_S and C_R in our approach, (ii) the Context Manager (manages the contexts of the context information base and their associations) and ConCoord (provides functionality to map necessary Universal Context IDs to information on demand) in ContextWare are represented as multiple CDP levels which essentially manage the entire POSpace and its contents, and (iii) the essential context composition and management process (equipped with context information base and ontologies) is mirrored and achieved by the POSpace via context-specific overlay groups. We initiate an implementation of ContextWare by proposing an architecture to construct and maintain a novel Pure Overlay Space. The registration, fetching, and casting processes, which denote the actual dissemination process based on virtual contexts, are discussed as a part of the implementation architecture.

3.5 Example Usage Scenario

As our dissemination approach is typically based on overlay network construction and adaptation, and as many other approaches are based on the same, it is important to state what type of applications one would expect from the underlying network. Thus, we consider the following usage scenario as a running example throughout this thesis to illustrate the practical operation of the dissemination mechanism. The illustrated example scenario (see Figure 3.5) covers both horizontal and vertical context dissemination modes and represents the 'campus' and 'transit' life of students (i.e., context entities) who need additional information on real-world objects via their electronic equipments. At any time, the students might be interested in the time, location, or any object-based context. Considering the amount of students and events in any campus, the information exchange process in such an environment tends to become complex without the aid of a context-aware system. In the case of a context-unaware system, the clients have to rely on possibly outdated and limited public information to satisfy their needs.

A group of university students have scheduled a meeting to discuss their group's upcoming year activities. Assume, while en route from home to campus, Emily (a student with a Smartphone and laptop, comprising a PAN) communicates with public ambient networks via CDPs (for instance, in transit trains which she rides to reach campus) to request specific MP3 files via her Smartphone. While transiting Emily uses her Smartphone for entertainment, communication, and also work-related activities; while in campus she uses her laptop for both entertainment and work-related activities. Once within the trusted campus environment (private CDP), the PAN transfers any open sessions (i.e., the audio file) to her Wi-Fi connected laptop. The PAN also establishes a VPN with her home network to access her secure files which must be printed. The active sessions might be moved to and between campus Wi-Fi hot-spots, while Emily is moving within the campus environment. Emily shares context knowledge about the scheduled meeting (and any updates) with other students. Also assume, while en route from home to campus in her car, Sara (another student with a PAN) communicates with the car's ambient network to transfer: (i) MP3 files from her Smartphone to play in the car's stereo and (ii) addresses

from her appointment/chores schedule in the Smartphone to get directions in the car's GPS navigation system. While in campus, Sara requests the meeting location. During the meeting with Sara, Emily establishes audio conference with other students who like to participate in the meeting. Before leaving to home, any unfinished work-related activities are left open in a secure connection. On the way back home, Emily's PAN as well as Sara's PAN repeat the same 'en route' process, for example, plays a paused MP3 file.

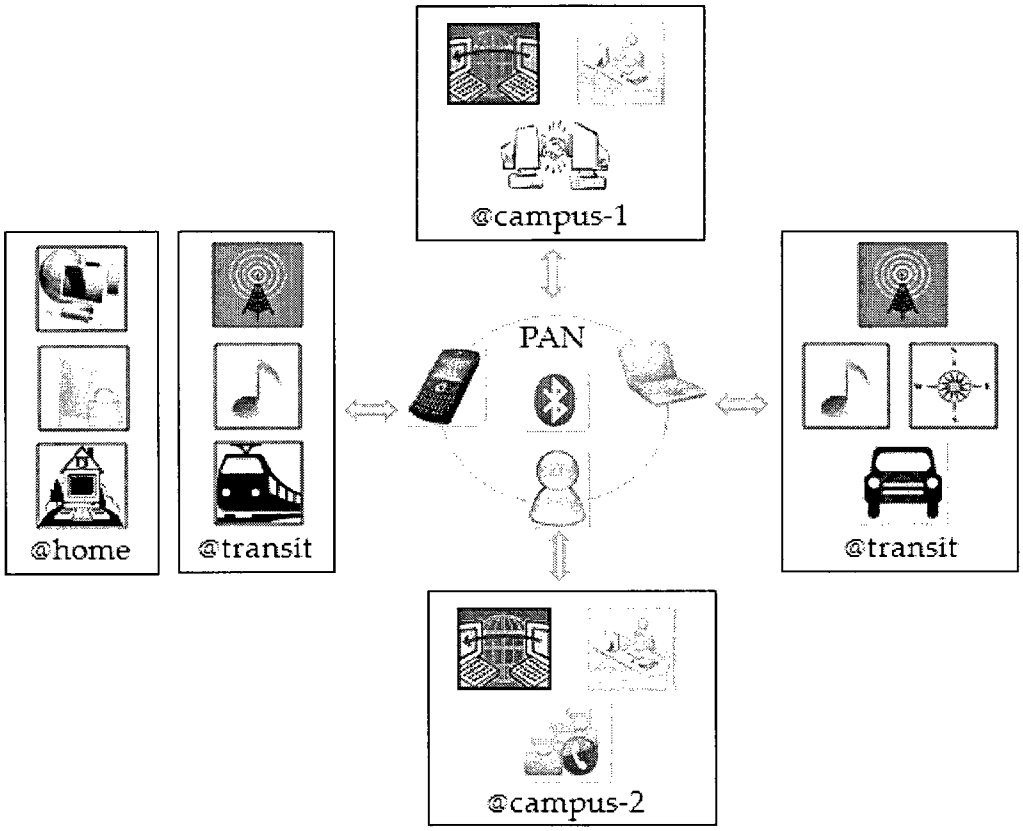


Figure 3.5 Example Usage Scenario.

3.6 Summary

The *Context Dissemination Overview and Architecture* chapter first defined the main terminologies used. It then overviewed context dissemination concepts and discussed context dissemination features with regard to ambient networking. The chapter mainly described the high level architecture of the context dissemination protocol, and

ContextWare mirroring, which are sub-divided in to four design model chapters (refer Chapters 4, 5, 6, and 7). The chapter also briefs an example usage scenario of the context dissemination protocol which will be used and discussed in the subsequent chapters. The next chapter discusses the first part of the proposed high-level architecture i.e., the Context Dissemination Processor.

CHAPTER 4
Context Dissemination Processor

This chapter discusses the functional area which processes the context entities as overlay nodes. The chapter elaborates the CDP's main components one-by-one. The basic functionalities of these CDP components are then enhanced by aggregated multiple objects. By comparing the CDP component's usage with respect to the example scenario, the chapter further illustrates CDP's significant aspects. The chapter finally states the employed implementation architecture and infrastructure to evaluate the results of the CDP.

4.1 Main Components

In this section, the CDP functionality's main components are summarized. The CDP employs a hybrid model, wherein it handles both request-driven and overlay middleware knowledge-based context provisioning. This is essential since we do not distinguish context entity handling based on their type. The first CDP level (CDP-L1 in Figure 4.1) is the central functional area responsible for handling context entities, monitoring context entities, and also reformatting the handled entities. Its main processes include: (i) Context identifier naming, (ii) Semantic profile decoder, (iii) Classifier and Agent tags, and (iv) Agent Manager (made up of agent tags with look-up service). Its extension template include: (i) Service framework, (ii) Performance indicator, and (iii) Trust handler, which are discussed as a part of the context monitoring optimization scheme (see section 7.3). The CDP-L1

processing structure made up of the above discussed processes is illustrated in Figure 4.1. With regard to the ambient network infrastructure, the CDP is viewed as a part of the Context Manager. While in the second CDP level (i.e., CDP-L2 functional area), the processing and management of the context-specific overlay space is optimized and adapted with the aid of a routing table (RT), as discussed in sections 5.3 and 5.4. In general, the CDP functionality satisfies special dissemination requirements to provide specific services without having to deploy new equipments or modify existing protocols, thus reducing cost, minimizing delay, and maintaining scalability (refer performance evaluation results).

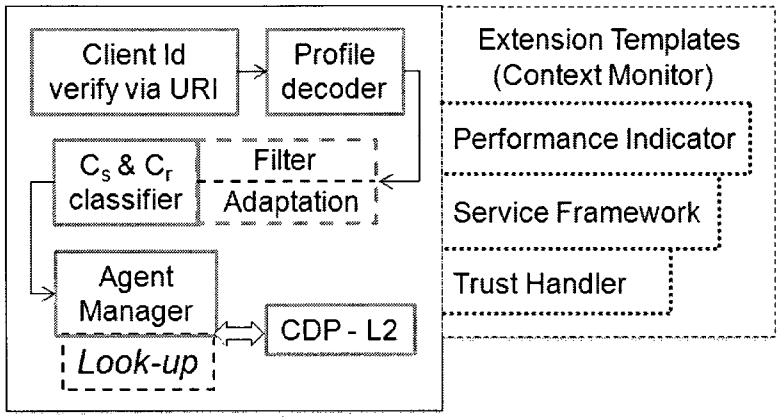


Figure 4.1 CDP's main components.

4.1.1 Context Identifier Naming

In a dynamic and heterogeneous environment the different terminologies used for context description creates a problem for communication between context entities. Context-aware networking lacks a standard framework for contextual information identification. Within our dissemination scheme, the context identifier (C_{ID}) naming scheme identifies the contexts before associating them as profiles. This is achieved by following the URI (Universal Resource Identifier) scheme [105] for identification represented in (E.1). It considers and includes the context entity's network domain information, location information, and private and public options. A type of URI for contexts, UCI (Universal Context Identifier), takes the form shown in (E.2). UCIs uniquely identify a given context object, but not its location within the network. Though there are many existing ontologies

for context dissemination, we stick with UCIs in this work as ontologies are out-of-scope of this thesis. Every piece of context information provided in an AN is identified by a UCI. For example, for a printing service, the context identifier would take the form shown in (E.3).

$$URI://NW_{ID}.L_{ID}.C_{ID} \quad (E.1)$$

$$URI\ scheme://Domain\_name/Location\_path/Options \quad (E.2)$$

$$ctx://pan/devices/printer/hp\_model/options \quad (E.3)$$

As a part of the context verification process, the entities for which the UCI scheme cannot be fully formed are not immediately filtered. In this naming scheme, for the purpose of context verification, we consider the elements found in equations (i), (ii), and (iii) from right-to-left. This is done to logically compute sub-C_{IDs} for specific services when full C_{ID} fails to verify. By following this naming scheme, the subsequent CDP steps can efficiently avoid unnecessary information with the help of the trust handler extension template. In this naming scheme, for the purpose of context decoding too, we consider the elements found in the equations from right-to-left. Alternately, for the purpose of context classification and subscription, we consider the equations from left-to-right.

4.1.2 Semantic Profile Decoder

The verified context entities are considered as context profiles, whose generic structure which follows XML format is represented in Table 4.1 (a). The decoding of these unstructured context profiles is necessary due to the lack of common ambient network context inference mechanism. We perform semantic decoding using our own java-based XML decoder (refer sections 4.4 and 7.7.3) by utilizing the context entity's source performance level (with respect to context requirements), known trust levels, priorities, and already existing (i.e., learned from previous decoding) policies for such context entities. These policies are accessed from the CDP knowledge base (CDP_KB), which maintains

them within the data repository. With the help of C_{ID} we obtain profile relationships with priority to reduce path inflation due to redundant contexts and also discard unnecessary details. As we do not split the contexts into smaller chunks, the missing chunks problem is avoided. A sample of a decoded C_S profile is shown in Table 4.1 (b).

Table 4.1 Context Profiles.

(a) AN generic C-profile	*(b) Decoded C_S-topology*
AN_C_profile	AN_C_s_profile
+ C_profile_ID	+ C_S_0122
- URI scheme	- URI://NW_{ID}.L_{ID}.C_{ID}
: indexing & locality (displayStyle)	- XML structure
- <data type> elementSets	- elementSets
: value (name, type, group)	: values [0122, device]
: properties [memory, component, n/w, etc]	: properties [device-desc, options-desc]
: characteristics [fixed, etc]	: specific characteristics
- profile constraints	- profile constraints
: time value, validity, etc	: #access-time, #date, #options-specific
- optional C_{ID} relationships	- URI://NW_{ID}.L_{ID}.C_{ID1}.C_{ID2}

4.1.3 C_S and C_R Classifier and Agent Tags

Once decoded, the semantic classifier filters (to sort contexts and eliminate unnecessary profiles), adapts, and classifies the structured context profiles. Based on the decoding results we mainly classify contexts related to network contexts as: the networks themselves, devices, users, services, persistent data, and unknown. This is the current inclusive list but is not a limited one, and hence can be modified. Each of these context types has appropriate context sub-types. In order to deduce these sub-types, the classifier (for both C_S and C_R – see Figure 4.2) matches the decoded results with a pre-defined set of customizable context categories. The context categories are also represented as policies (in CDP_KB) which are to be followed for specific context entity's properties. In general, we employ these categories for context classification based on the target application's or user's preferences. It

- ***Identify***
  - decoded AN_C_S_profiles/ AN_C_R_profiles
  - Policy_IDs, CDP_KB hash table, RT
- ***Verify* and *Filter***
  - decoded profile's constraints are validated
  - filter validated profile's tags based on CDP_KB
  - <data type> checked for context category relevancy
- ***Classify***
  - check for redundancy
  - Policy_ID usage for matching and property extraction
  - relationship association
    - compare with existing CSN/RSN agent tags
    - establish entity-relationships and states
    - form agent association set by comparing with *leader* (refer section 5.1) agent tags
  - agent tag path based on indexing
- ***Display Style* and *Others***
  - CSN or RSN agent tag in an XML-based structure
  - *update*: dissemination topology, hierarchical CDP_KB, RT
  - *forward*: to agent manager

Figure 4.2 C_S and C_R Classifier.

forms a more complex and complete context profile structure when compared with the basic context profile discussed before. The classified context profiles (with types and multiple sub-types) are outputted as agent tags and updated in the CDP_KB as a RT. These tags are dynamic XML structures that maintain context attributes using agent technology's communication toolkits (refer section 4.4) [106]. Context Specific Nodes (CSN) or Request Specific Nodes (RSN) are either of the two resulting agent tags which represents C_S classification or C_R classification, respectively. These agent tags are compared with already formed agent tags to establish relationships between them, if any. The relationships form the base of the dissemination topology structure. The output of the context classification

process indirectly defines the dynamic topology used for dissemination, and hence has an impact on the efficiency of the overall dissemination.

4.1.4 Agent Manager and Look-up Service

```
<xml version = "1.0">
<document>
<context-element>
    <context type value = "device-profile" />
    <device>
        <device type value = "printer" />
        <device name = "hp-model-xxx" />
    </device>
    <properties>
        <options>
            <printer-type value = "laser" />
        </options>
        <syntax>
            <var num = "?"/> <var type = "?"/> <var value = "?"/>
        </syntax>
    </properties>
    <context owner value = "PAN" />
</context-element>
<context-association>
    <association-set value = tag paths />
    <relationship-set value = entities and states />
</context-association >
</document>
```

Figure 4.3 A sample agent tag.

A sample agent tag is illustrated in Figure 4.3. The CSN and RSN agent tags and their inter-links are managed by the *agent manager* whose attributes define the context's properties. The agent tags are represented as *Agent Object Subscriptions* (AOS) within the agent manager. Each of these AOS act as a context router (either inter- or intra- router based on the topology) when associated in the overlay space (which is the next step in our dissemination mechanism - see Chapter 5), and they are all distinct with priority rankings

(i.e., based on decoding) and node status. Thus at any time, any one agent tag cannot act as both C_S and C_R. These AOSs are hierarchically arranged (in FIFO format) with partially-structured topology and with relationships between each other so as to match with the constraints of the target application or user. Partial structuring denotes the relationship between the agent tags from the decoding and classification processes, which in turn represent the star/mesh topology employment in the overlay space. By having such topology options, different context entities can chose different topology paths which guarantee high performance for their specific metrics. Due to this type of structuring, bandwidth and latency problems tend to show up as the agent tags representing context information need complete relationship properties of each other.

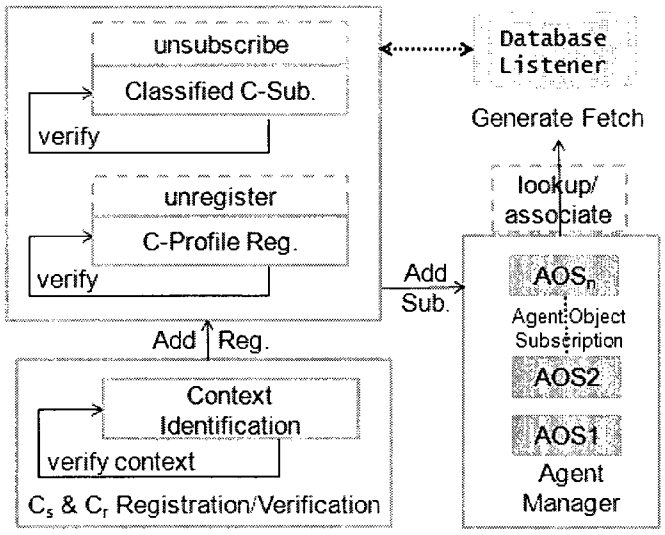


Figure 4.4 CDP-L1 functionalities up to agent management.

As we only maintain context pointers within the CDP_KB to track the tags, a lookup service is essential for proper maintenance of the CDP_KB when the agent tags move on to the next level. This lookup service will help to track the tags in the overlay space and hence assists the dissemination mechanism. When a node initiates communication, if local lookup is not feasible, the lookup request propagates to the vertically composed ANs. In addition, the ANs could forward the lookup request to horizontally composed peers as we assume a hybrid (both horizontal and vertical) composition strategy in ANs. The agent's lookup

service also follows *sequential priority assignment* and includes a *renewal* service. The sequential priority assignment denotes agent priority and status based on first come first serve basis within the same context types. Hence, for various context types and sub-types, a separate priority assignment will be followed. The renewal service tracks the updated agent tags within the RT and hence aids in finding the latest contexts. The so-far discussed CDP-L1 processes and agent management functionality are illustrated in Figure 4.4.

4.1.5 Aggregated Multi-Objects

The virtual control and management of context entities is achieved without any brokers by filtering unnecessary content, minimizing overlapping information, tackling reconciliation problems during multi-C_S involvement, and reducing routing state. The overlay structure guarantees perfect routing i.e., the disseminated contexts are received by all and only those C_R that have subscribed for it. The CDP-L1 extension templates discussed in section 7.3 aids in achieving this. With the help of AOS (refer Figure 4.4), the coexisting entities are dynamically evolved. For efficient overlay clustering in the next level, this subscription aggregation feature is essential for both C_S and C_R. With the existence of multiple loosely coupled [85] (i.e., hierarchically organized with partial/unstructured topology and relationships between each other) agent object subscriptions aggregated together, we initiate multi-object dissemination. This multi-object dissemination feature is reflected in the design of the overlay node join and leave procedures discussed in section 6.2. Such parallel operation signifies simultaneous context handling for context information dissemination and also for discarding service instances after dissemination.

4.2 Assumptions and Constraints related to CDP

Assumptions:

1) The independent entities with no prior topology relationships or constraints are aware of their contexts, or else inconsistencies could occur between their formulation and context parameter.

2) The context data, before being processed by the CDP, is already modeled by some external context modeling scheme.
3) The modeled contexts include their context attributes and application type.
4) The context data is acquired and stored by some external context acquisition and storing schemes.
5) Whenever a client wants to obtain specific context information, it knows the UCI of the entity representing this information. Similarly, a context source is assumed to know the UCIs of the context entities it wants to publish.
6) There exist pre-defined context types (and sub-types) in a public domain.
7) Various context entities have knowledge about various CDPs via their ContextWare API.
8) We assume a decentralized infrastructure with: (i) no Internet topology, (ii) no definite location awareness, (iii) no neighbour searching, (iv) no network latency, and (v) no minimum link bandwidth between hetero-entities.

Constraints:

1) Inherit intelligence by interpreting human behavior, reasoning about it, and acting upon it, to automatically cooperate and run applications that are sensitive to user's preferences/ requirements [39].
2) A distributed CIB-like component is essential to maintain the network's mobility and flexibility properties unless excessive inter-communication is performed.
3) The C_S / C_R classifications are based on 'pre-defined' policy tags and hence are not dynamic, but developer-customizable.
4) The real-time performance of the look-up service is under concern as its real-time applicability is not proven.
5) Use of a centralized routing table and mobile data repository for local request forwarding and caching.

4.3 Visualization of the Example Scenario

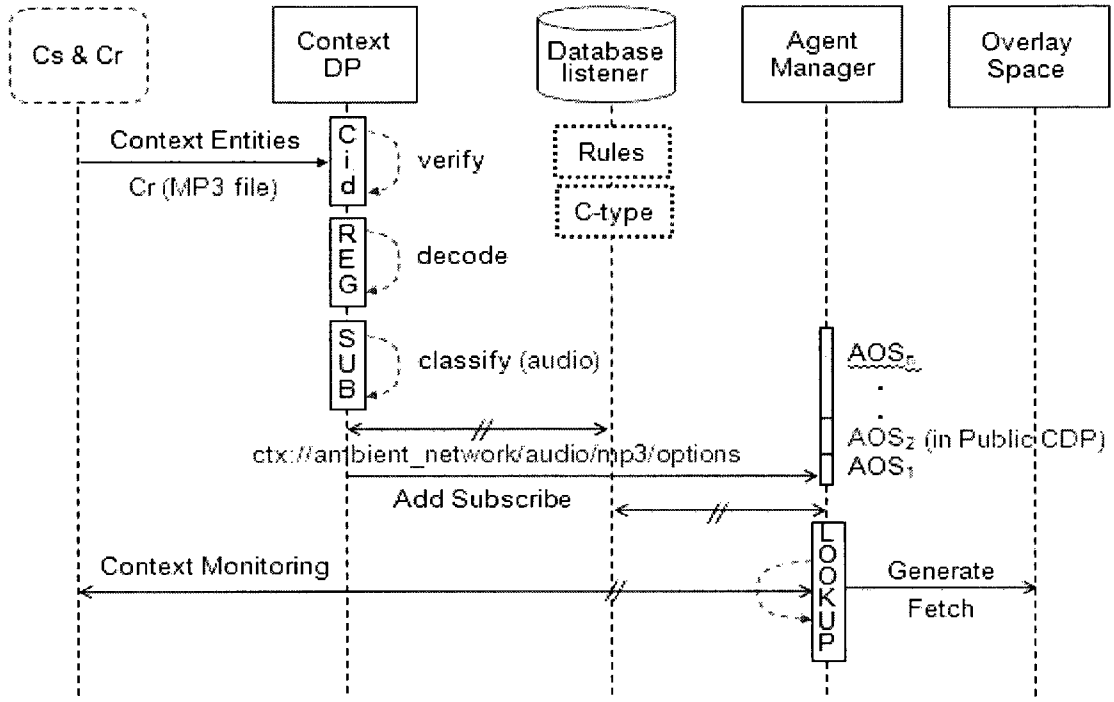


Figure 4.5 Example scenario with respect to CDP-L1 functionalities.

In our running example scenario (refer Section 3.5), Emily (a student with a Smartphone and laptop, comprising a PAN) communicates with publicly available ambient networks via CDPs (for instance, in transit trains which she rides to reach campus) to request specific MP3 files via her Smartphone. The audio context request takes the form ctx://ambient_network/audio/mp3/options by following the URI prototype. This process is illustrated with respect to the CDP-L1 steps in Figure 4.5 (in 'red'), using Message Sequence Charts (MSC). Note that, Emily, while acting as a C_R, can also act as a C_S at the same time, thereby triggering multi-object aggregation. For example, while requesting a MP3 file, Emily could share another MP3 file as well.

4.4 Performance Evaluation

4.4.1 Implementation Architecture

The context dissemination processor implementation details are discussed as a sequence of events by following ambient network context protocol primitives [101], [105]. As a common step, all events are 'updated' in the database listener which persuades data-centric storage. To begin with, the source-context REGISTERS itself with the core functional area, the CDP. The CDP is the first point of contact for the context entities. While in the case of request-context, registration is followed with 'resolve'. The registration is then validated by 'verifying' the C_{ID} (i.e., both C_S and C_R following context identifier naming scheme). The verified contexts follow context profile decoding and semantic classification in sequence in the data processing layer. Each step is verified to confirm context validity. For example, if decoding fails or if the context information is filtered, the C_{ID} UNREGISTERS. This is followed by the SUBSCRIPTION of classified context profiles as AOS. The local subscriptions are hierarchical arranged (FIFO for C_R) and processed by the agent manager functionality. By means of the lookup service, the fetching process is initialized, which is discussed in section 5.8.1.

Although we try to avoid any centralized component, we still need to use a routing table and mobile data repository. The routing table (refer Table 4.2 – also indicates the main database columns) acts as the local request forwarding table which caches overlay operation. This table is maintained by the CDP lookup service within the data repository (CDP_KB) to identify and preserve a list of context clients (with their properties) that are involved in overlay node creation, association, and maintenance (in the next level of the dissemination mechanism). The table includes details about the context type (C-type), context client's identifier, received packet's sequence number, time when last packet was received, overlay relationship links (O-links), and context score, to name a few. We can identify the set of request-contexts whose requirements are disseminated and those that are yet to be disseminated. Each leader node (ON_L – refer section 5.1) and subsequent overlay nodes maintain their state information which is accessible to neighboring nodes.

Table 4.2 Sample Routing Table (Database).

C-type	*C-subtype*	C_{ID}	*Property*	*Classify*
<str>	{non-persistent}	<UCI>	Priority, TTL, etc	XML structure
Status	*Interaction*	*O-links*	*Benefit*	*Score*
{*CSN, RSN, or* ON_L }	<boolean>	{ }	<int>	<int>

4.4.2 Infrastructure

To affirm our context dissemination mechanism, we simulated the CSON-D protocol based CDP functionalities and optimizations, as an implementation prototype of ContextWare [7]--[8], with focus on context composition and management part. We don't split the contexts into smaller chunks as in WSNs, and only the monitored non-redundant contexts are disseminated and received by all and only those C_R that have subscribed for it. The simulation study is performed under realistic practical assumptions (refer section 4.2) to analyze the approach's specific performance areas. We also assume that the underlying system's transport layer supports differentiated services. We make maximum use of the available resources to enable autonomous flexibility with low persistent storage space. In all instances, the 'average percentage' results are obtained by either starting off from zero or allocating optimum energy levels. With respect to load, latency, activity, etc we measure the consumed energy. As the PCs used for simulation have high processing power, the context protocol primitive based implementation architecture processes consume less time to execute compared to the time consumed on data transmission. The system architecture is built upon the Microsoft Windows XP platform with Bluetooth stack. We use a customized BlackBerry JDE dependent simulator to represent GPS-enabled mobile devices which support cross platform dynamic data overlapping [107]. We employ Java ME (J2ME) CLDC's package, javax.microedition.location, which contains the basic classes needed to request/get location information on a BlackBerry [108]. The communication services incorporated within the scenario use Robust Audio Tool (RAT) [109]. The database is setup using SQL Server 2005 for representing an optimized knowledge base. J2SE SDK is chosen for programming this model since it is system independent and ensures adaptability, and XML is used as the encoding language for context entities since it provides portability and

flexibility [110]--[112]. The prototype's components are java-based XML structures. The CDP implementation basics are discussed alongside CDP functionalities. The CDP KB based semantic decoding is achieved by employing a URI profile processor [105], java-based XML parser [113] and a customized JSTL based parser, which results in agent tag formation. These agents are represented in Agent Communication Language (ACL) form and are developed and executed using the FIPA-OS agent platform [59], [106], [114]. The lightweight MicroFIPA-OS agent platform is used for simulating scenarios involving resource constrained mobile devices [114]. FIPA compliant agent platforms follow necessary FIPA specifications. Agent platforms are generally made up of platform-specific agents and transport services. By default, the *http* transport protocol is used for both inter- and intra- communications as it is the only transport protocol that is supported by the MicroFIPA-OS agent platform. The prototype's topology is made up of 1 to 25 cluster-heads, 1 to 500 C_S, and 1 to 100 C_R, though it varies between experiments. The agent manager simultaneously handles up to 15 hierarchically arranged agent tags for association.

4.4.3 Results

In this section, we only study the delay to create and associate agents to compare CSNs and RSNs. Figure 4.6 shows the results of the time delays to create and associate C_S and C_R agent tags (from Agent Object Subscriptions) in the overlay space. It can be seen that the delay increases as more simultaneous entries take place, which is in line with the characteristics of loose-coupling that we follow in our design architecture. The figure also illustrates that the delay with respect to RSN is more linear – this is due to semantic classification and subscription aggregation. It should be noted that the agent creation process also takes up considerable amount of time due to delays in the agent platform that we cannot control.

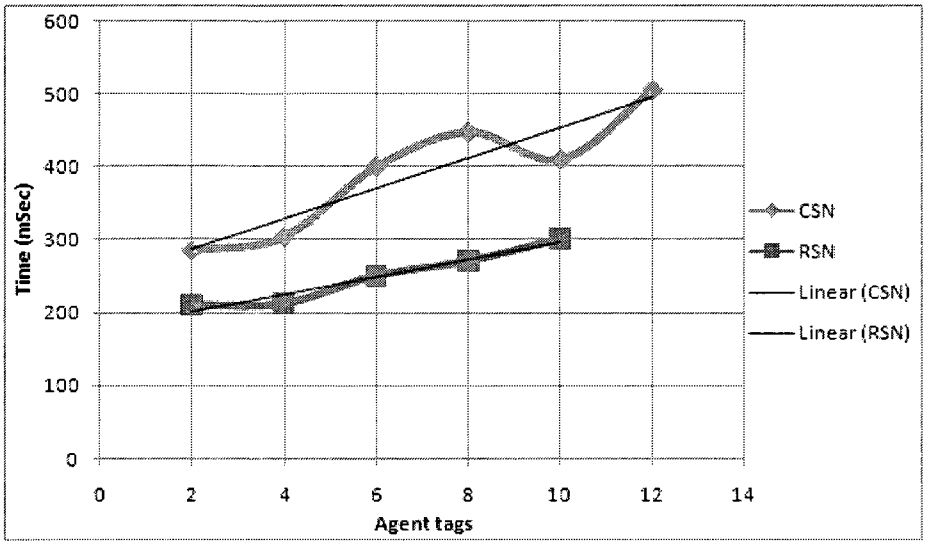


Figure 4.6 Delay during agent tag creation and association in overlay space.

As the CDP contributes to the operation of the POSpace, multi-level overlay model, and the context monitoring scheme, other results related to the CDP are discussed in subsequent chapters. By evaluating the effect of multi-object dissemination and proposed optimizations in Chapters 4, 5, and 6, we study *Success rate vs. Multiple overlay agent nodes* in Figure 6.23. The overall time delay of the CSON-D protocol which includes the CDP functionalities is studied in Figure 6.24.

4.5 Summary

This chapter discussed the CDP functional area which processed the context entities as overlay nodes. The chapter elaborated the CDP's main components one-by-one. The basic functionalities of these CDP components were then enhanced by the multi-object scheme. By comparing the CDP component's usage with respect to the example scenario, the chapter further illustrated CDP's significant aspects. The chapter finally stated the employed implementation architecture and infrastructure and evaluated the results of the CDP. The Pure Overlay Space scheme is discussed next.

CHAPTER 5
Pure Overlay Space

This chapter discusses the design model of the Pure Overlay Space. It states how it copes with the dynamic changing nature of today's networks and systems and ensures latest contexts by discussing the employed spatial decoupling and semantic clustering techniques, along with related optimizations. It discusses the context scoring and context fairness schemes that are employed to maintain dynamicity and fairness in the POSpace. It discusses the hybrid agent overlay mechanism. The POSpace's basic functionalities are then enhanced by proposing techniques to infer personalization and customization. With the use of the running example from previous chapters, we illustrate POSpace's significant aspects. The chapter finally evaluates the performance results of the POSpace.

5.1 Operation Steps

Due to the requirements of ContextWare, a novel overlay space that supports heterogeneous composition of overlay nodes and that precludes service brokers is developed on top of traditional overlays, and is illustrated in Figure 5.1. Once contexts entities are classified as agent tags, the POSpace is created and initialized, and acts as an interaction medium with regard to space and representation. We define POSpace as a completely logical overlay layer capable of application-based adaptation based on the target application's (or end-user's) needs, and satisfying all major overlay properties. This

abstraction layer above the current networking technologies copes with heterogeneous underlying network infrastructure. We view the mechanism as an application-layer overlay multicast, whose basic approach is to first divide the contexts, then form a mesh structure via bisected context key-spaces, which later alters into a top-layer overlay for unicasting. The POSpace adopts on-demand dissemination instead of flooding, wherein it spontaneously propagates the processed contexts from C_S to C_R. Spontaneity denotes dissemination of latest contexts as AN's context information is short-lived. While there is no direct link between the functionalities and structure of the components within the overlay space to the underlying layer, the dynamic adaptation of an overlay would ultimately take place in response to changes in the underlying network. This pure logical characteristic is highly aided by utilizing the agent's functionalities, more specifically the Proxy Agent (PA) overlay mechanism. The target application's requirements fulfillment is assisted by Mobile Agent (MA) functionality. These agent node components are strategically placed in the overlay to automatically arrange themselves as overlay nodes without any user interaction. More specifically, all actively involved (i.e., connected to some node in the network) heterogeneous context entities act as spatially and semantically composed agents, which take a hierarchical context-specific hybrid mesh structure for dissemination. A hierarchy is necessary to support high-level corporation without a fully balanced structure. Each of these composed sections is a disjointed data space containing information concerning a particular context. These hierarchical overlay structures dynamically reconfigure themselves with the aid of the CDP-L1 (previously discussed in Chapter 4) to adapt and optimize the path of context dissemination due to context updates. The overlay space would eventually be closed if there are no further context interactions (for e.g., if idle time exceeds). This is achieved by tracking the context data changes in a routing table, which is maintained in a data repository. Hence, the dissemination scheme not only supports the data representation components, but also the data presentation components. This scheme is further optimized to infer customization and personalization. The POSpace operation steps are initialized in Figure 5.2, and subsequently discussed with other POSpace based functionalities as and when they appear in this chapter.

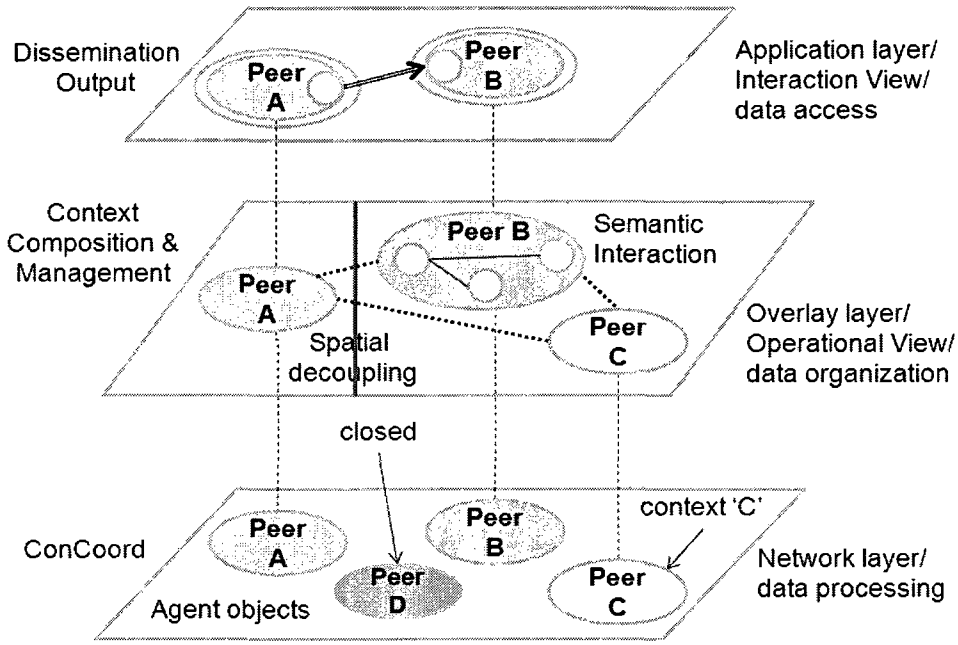


Figure 5.1 Pure overlay scheme with component architecture and ContextWare labels.

DECLARE

C_{type} *and* $C_{sub\text{-}type}$; // context type and sub-type

OS and ON; // overlay space and overlay network

L_{max}, N_{max}, D_{max}, L_{prop}, N_{prop}, D_{prop}; /* max optimal location, neighbourhood, and content load properties*/

ON_{ID}; // overlay nodes

ON_{LID}; // leader overlay nodes

ON_{CID}; // overlay cluster identifier

C_{ID}, CR_{ID}; // context- and request- specific tag identifiers

INITIALIZE

initialize OS, L_{max}, N_{max}, D_{max};

OS identification, size, and addressing scheme;

spatial independence and semantic addressing;

adjust overlay properties based on POSpace properties;

Figure 5.2 POSpace Operation Steps: Initial.

5.2 Spatial Decoupling and Semantic Clustering

The context entities as agent tags are uniformly distributed in the overlay space to achieve feasible context dissemination in a load-balanced and degree-constrained manner. This uniform distribution is carried out by spatially decoupling the POSpace based on the node's context types or categories as shown in Figure 5.3. For example, if there are a number of context nodes with three unique context types initially, the entire coordinate space will be partitioned among these three context types. The decoupled space itself is dynamically altered based on subsequent other context type overlay associations. For example, when a node with a new context type wants to join, an existing node in a particular partition splits its allocated space.

The connections between these decoupled overlay nodes representing end-hosts are influenced by contexts, instead of end-to-end topology. This context-based connection establishment is achieved by performing semantic clustering. While the pure overlay structure is based on several source-context's and request-context's characteristics, its main characteristic is that it is semantically structured. Within each decoupled overlay space, the self-contained agent tags are evenly distributed by semantically clustering them in a logical and distributed fashion based on their attributes. Herein, there are two types of agent tags - the one that decoupled the overlay (called the leader node) and the one that joined the leader node i.e., the sub-node. Thus several levels of semantic clusters are feasible within a decoupled overlay space. Clusters are an efficient (defined with respect to local balancing, cost, and latency) way of providing robustness and availability. The multiple overlay clusters can overlap and its overlay layers are leveled hierarchically (as shown in Figure 5.3). This semantic clustering is performed by following a logical tree-based overlay structure. The logical tree-based overlay structure doesn't need any topology assumptions for its generation, as it is constructed reactively based on up-to-date contexts, and is compatible in an ambient network environment. Each hierarchically organized cluster thus takes the form of a loosely coupled and partially-balanced mesh that tackles the context entity's mobility and resilience (via agents – refer section 4.1.4). By loose coupling, a C_R cannot expect a direct response from a C_S, and also cannot mandate a response within a

certain time. With the same note, the context time-to-live property could still be valid. It is partially-balanced because of the different mesh balancing strategies adopted in the hierarchical multiple overlay levels, which is the next part of our dissemination mechanism. This overlay structure favors easy location of the required (i.e., interested) contexts, as there will be less number of context entities within each cluster, while multiple clusters exist. The unused contexts are clustered together to minimize the processing of unnecessary information. As the target application's requirements keep changing in a heterogeneous environment, extending the successfully created overlay structure is almost always necessary. With low number of nodes for intra- communication we explore all possible paths so that the extended overlays will match the latest needs of the target application. This context node exploration within a cluster is performed by first defining rules for every leader overlay node, and then performing an intra-guided search at various instances. This in turn results in context requirements matching with minimum overlay node selection and with no overlay network flooding. The POSpace operation steps for spatial decoupling and semantic clustering are outlined in Figure 5.4. This semantic clustering idea is further developed as discussed in the subsequent sections.

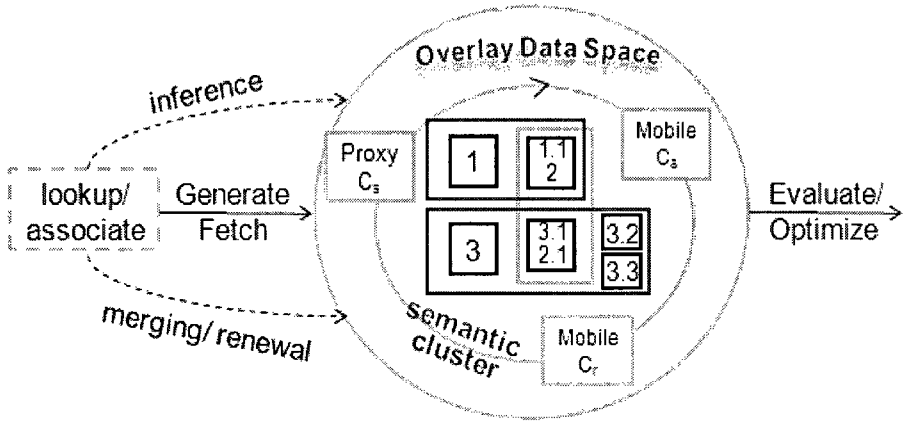


Figure 5.3 Semantically clustered dissemination mechanism.

UNIFORM AND LOGICAL HOST DISTRIBUTION

/* each ON has location, neighborhood, and (optional) content properties */

for every ($L_{prop} < L_{max}$ and $N_{prop} < N_{max}$) do

if OS = <null>

Perform randomized distribution of ON_{ID} in OS

else if (belongs to ON_{CID})

Perform leader- or sub-agent- tag association

else

Perform uniform random distribution of ON_{ID} in remaining OS

end for

INTER- AND INTRA- HIERARCHICAL OVERLAY SEMANTICS

ON_{LID}: Establish inter-relationship with other ON_{LID}

ON_{ID}: Establish intra-relationship with other ON_{ID} and inter-relationship with other ON_{LID} via current ON_{LID}

TAG ASSOCIATION EXTENSION/UPDATES

// executed if initial semantic clustering is successful

find nearest ON_{LID} based on locality sub-procedure, if any

if (no ON_{LID}) exit

else

for (($ON_{ID\text{-}prop}$-$C_{type} \in$ (selected) $ON_{LID\text{-}prop}$) ||

($ON_{ID\text{-}prop}$-$C_{sub\text{-}type} \in$ (selected) $ON_{LID\text{-}prop}$))

determine the neighbourhood score // Figure 5.5

determine the neighbourhood fairness // Figure 5.6

if the scores and resource states are under limit

optimize overlay sub-network by splitting and merging

perform updated ON clustering

end for

UPDATES

update lookup, CDP_KB, RT

Figure 5.4 POSpace Operation Steps: Main.

5.3 Context Scoring Scheme

```
// ON_agent-list denotes the cluster's active PA and MAs
initialize context score (CS) in RT
initialize scoring update timer
set CS_min and CS_max (via CDP look-up)
repeat
    if (C_ID or CR_ID associated as ON_ID to ON_L)
    else if (C_ID updates as ON_L)
    else if (external C_ID or CR_ID associated to ON_L)
    else if (semantic overlay clustering)
      check timer validity
      CS++ // for appropriate C_ID or CR_ID
    end if
    // repeat for context score decrement
    check CS utilization by comparing with CS_min and CS_max
until ON_agent-list is empty
```

Figure 5.5 POSpace Operation Steps: A Simplified Scoring Scheme.

The scoring scheme keeps track of the state of contexts in the overlay space and manages this status to affect the overall state of the dissemination mechanism. This CDP-L2 component works in conjunction with the agent look-up functionality. With reference to Figure 5.8 callouts, every time: (1) an overlay leader node decouples the overlay space, (2) a C_S agent tag associates as a sub-node with a leader node, (3) a C_R agent tag associates as a sub-node with a leader node, (4) star/mesh structures are formed/ merged, or (5) an overlay node restructures in an inter- or intra- fashion, the Context Score (CS) of the involved overlay nodes are updated. This is outlined in Figure 5.5 and the results are noted as a part of the routing table (see Table 4.2 for a sample). This scoring scheme also updates the score of context types whenever a C_S or C_R of a particular context type is observed or modified. For example, when more number of C_R demand a certain C_S or when the request frequency is high, the particular context type's score is increased. Alternatively, the context score is

decreased when a context type is idle for a certain period of time, and hence prompts changes in the overlay structure and node priority. The need for employing such a scheme is to enable easy access to frequently engaged context types by efficient re-structuring of the overlay based on the scoring results. By doing so, most of the overlay nodes are unaffected most of the time, thus reducing the dissemination mechanism's cost and latency. The resulting scores will be valid for a certain period of time before being reset.

5.4 Context Fairness Scheme

Similar to the context scoring scheme, we optimize the POSpace by applying the context fairness scheme which is the benefit-based functionality of the overlay nodes. For every 'received' benefit, an overlay node has to 'contribute' appropriate benefit to balance load i.e., maintain fairness, instead of maximizing its own benefit. This is achieved by quantifying the benefits within local transformations (see section 5.6.2) and then altering or prioritizing the local links based on the results. The computed 'selfless' benefit values are included in each context's header and in the routing table, and thus serve as a source for cluster updates. Even though load balancing is naturally achieved during the initial semantic clustering via routing table entries and leader node assignment, this scheme is necessary as it is not trivial to maintain the initial structure and properties during re-structuring. We measure benefit-value when a leader node decouples the overlay space, when a sub-node is associated with one of more leader nodes, or when a sub-node search is performed for overlay node re-structuring in an inter- or intra- fashion. The Context Benefit (CB) function is formulated as given below:

$$CB = CS / \sum_{i=1 \text{ to } N} (P * S * L) \qquad \text{(E.4)}$$

Herein, N is number of instances, and P, S, and L denote parallel (inter-communication), semantic (inter-transformation), and logic (context dependent factors) respectively. As outlined in Figure 5.6, the fairness implementation is based on the benefit timer value, the minimum benefit threshold, and the overlay node's context score. For every C_R, the chosen C_S would have the highest fairness ratio.

```
// ON_agent-list denotes the cluster's entire PA and MAs
initialize context benefit (CB) in RT
initialize benefit update timer
set CB_min (via CDP look-up)
repeat
    while timer within limit
     if (ON_L decouples as ON_ID or leaves cluster)
     else if (C_ID or CR_ID associated as ON_ID to ON_L)
     else if (external C_ID or CR_ID associated to ON_L)
     else if (semantic overlay clustering)
       compute CB using equation (E.4)
     end if
    update CB based on CS increment or decrement
    check CB utilization by comparing with CB_min
    end while
until ON_agent-list is empty
```

Figure 5.6 POSpace Operation Steps: A Simplified Fairness Scheme.

5.5 Visualization of the Example Scenario

The scenario-based case study's specific instances are visualized in Figure 5.7. The first layer of consideration is the physical layer consisting of C_S and C_R in no particular order or format. In our scenario, several students would have registered with the campus's private CDP for various purposes. These context entities are converted into network peers (i.e., agent tags). As these agents are processed in the overlay space based on their context properties, some are selected while others are modified or filtered, resulting in context specific overlays. The students with active tags would be a part of the POSpace. Based on further context composition and management, the context requests are disseminated in the interaction view. For example, in Figure 5.7, consider the audio conference between Emily and Sara.

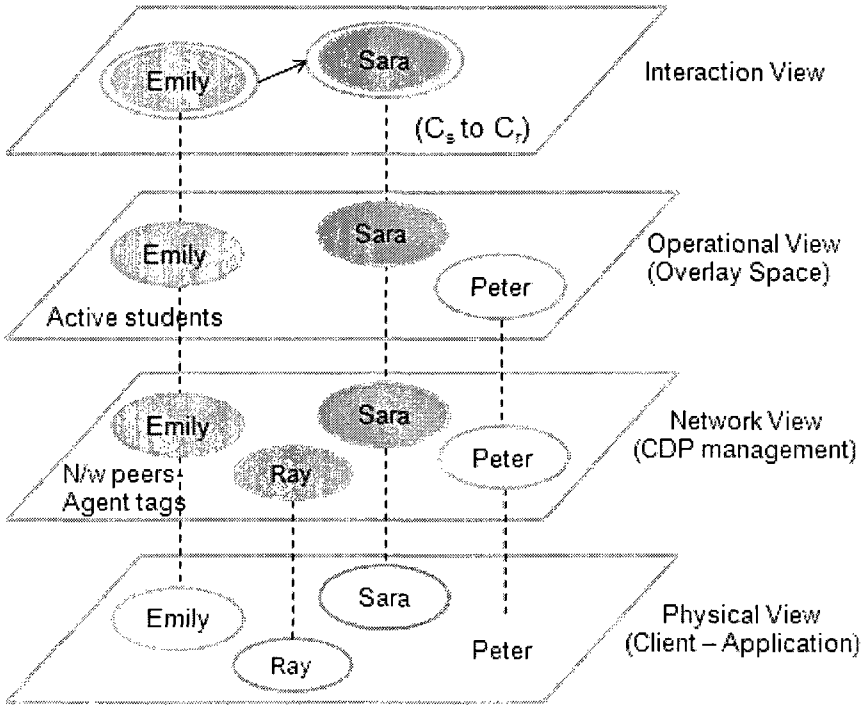


Figure 5.7 Visualization of protocol abstractions.

5.6 Optimizations

5.6.1 Hybrid Overlay Agent Mechanism and Structure - Proxy Agents and Mobile Agents

The proposed agent mechanism is a novel proxy service based overlay mechanism which acts as a middleware between the actual overlay space and the existing underlying network i.e., as a virtual backup node functionality. A CSN or RSN (i.e., the result of the CDP-L1) represents data end points (context attributes) which could either be a sender, a receiver, or both. These loosely coupled agent tags take the form of either PAs or MAs based on their context types and sub-types. A PA overlay node is usually the initial node (i.e., CSN or RSN) to be associated of every new context type i.e., the leader node functions as a PA. The PA triggers spatial decoupling and optimizes the overlay by splitting/ merging the subnet. It avoids full decentralization as it manages its associated member sub-node's

contexts (for e.g., location, replicas, peer access frequency, etc). By association we mean the semantic clustering of agents. These members are nothing but all other agent tags that follow a matching context type URI i.e., the member sub-nodes are the semantically matching agents that match with the constraints of the target application wherein a CSON-D instance could be employed. These member sub-nodes act as MAs, and the MA functionality reduces communication costs by moving the processor to data rather than bringing data to a central processor. The MAs react to dynamic network changes by moving from one cluster to another based on overlay node pre-processing results and lookup function requirements. Part of the lookup service can be made distributed and mobile by including it in the MA header. The MAs clone during time-to-live (TTL) for data gathering. The PA overlay nodes act as a middleware between the MA overlay nodes that are connected to it and the CDP's lookup-service. PAs alter the source of dissemination and maintain the resource state of its members periodically. The PA itself can be adapted by assigning its functions to a MA based on its dissemination history or network dynamics.

The POSpace characteristics are highly optimized by utilizing the hybrid overlay agent mechanism and structure. Hybrid denotes concurrent intra- and inter- corporation of overlay nodes with varying structure and balance. By doing this, we virtually structure the overlay to maintain only its logical characteristics and ease internal communication by reducing the number of internal node associations independent of underlying network structures. It aids to support both request-driven and overlay-middleware knowledge-driven context dissemination. Without the overlay middleware knowledge, a RSN has to wait for an appropriate CSN subscription, and without on-demand option valuable resources might be wasted. This allows us to achieve a de-centralized, parallel, and fast re-structuring of overlays without unwanted network partitioning and communication breakdown i.e., with balanced cost and latency to tackle context mobility and resilience. We adopt this adaptive hybrid agent structure for context dissemination without the cost of underlying structure maintenance. The agent's performance bottleneck problem is handled by using this proxy agent overlay mechanism. This hybrid agent mechanism works around virtualization constraints and significantly reduces the number of overlay rebuilding instances.

The basic functionalities of the overlay agents are illustrated in Figure 5.8 (P, S, and R denote ON_L, CSN, and RSN respectively) and the algorithm is outlined in Figure 5.9. The figure's bullets represent: (1) PA association to partition overlays, (2) MA association as C_S or C_R during overlay updates, (3) PA matching with associated MAs to form agent-based overlay clusters, (4) formation of star/mesh structures if the resource state of the neighbourhood is under limit, (5) either PA or MA or both restructure in an intra- fashion to maintain/update the resource state of the overlay cluster's members, and (6) MAs move from one cluster to another and clone for data gathering to maintain the resource state of the inter- overlay cluster members.

LEADER- AND SUB- NODE OVERLAY AGENT TAG ASSOCIATION

// based on agent tag, locality, and neighbourhood properties

for each $ON_{ID} \in C_{type}$ *do*

if $(D_{prop} < D_{max})$

  if ON_{ID} *is the first* ON_{ID} *in a particular* ON_{CID}

    register itself as a neighbourhood leader (PA)

    index ON_{ID} *to allocate leader as* ON_{LID}

    ON_{LID} *association in OS* $\leftarrow$ *true*

  else ON_{LID} *association* $\leftarrow$ *false*

for each $ON_{ID} \in C_{sub\text{-}type} \in C_{type}$ *do*

if $(D_{prop} < D_{max})$

  if ON_{ID} *is not the first* ON_{ID} *in a particular* ON_{CID}

    register itself as a neighbourhood node

    index ON_{ID} *to allocate node as* $[C_{ID}$ *or* $CR_{ID}]$

    $[C_{ID}$ *or* $CR_{ID}]$ *association in OS* $\leftarrow$ *true*

    perform 'locality sub-procedure' based on indexing and D_{prop}

    perform association: ON_{ID} *with* ON_{LID}

  else ON_{ID} *association* $\leftarrow$ *false*

Figure 5.8 Simplified proxy- and mobile- agent functionalities.

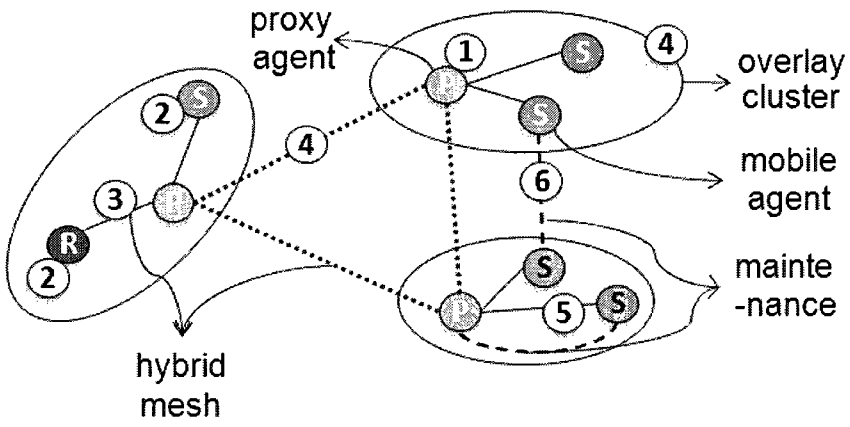


Figure 5.9 Overlay agent functionality and context optimization schemes.

5.6.2 Personalization and Local Transformation

The adaptation and interoperability features achieved by means of the functionalities discussed in the previous sections are further complemented by personalizing the POSpace. To meet the varying requirements of various applications and systems, the dissemination mechanism should be personalized to meet their exact requests. While the functionalities discussed in the previous sections support our 'no' user interaction claim, user interaction is necessary to personalize and customize overlays. The application developers or to-be-composed ANs can personalize the context data types utilized in our dissemination scheme based on their needs (see Figure 5.10 – based on leader and sub-node association). As the multiple context-specific groups are constructed based on the semantic properties of the agent tags, altering the tag formation process will result in varying the overlays i.e., end-system based dissemination. This guarantees overlay node location with minimum sub-node search even in composed ANs, as all active overlay's properties are observed. For an existing semantic cluster, inter- and intra- overlay node connections are personalized based on the latest context attributes by either dynamically extending or narrowing it. Personalization is not restricted to context tag processing alone, but extends to overlay maintenance. While performing this end-system based dissemination, we maintain the context client's localization and independence characteristics. This maintenance is important as any composed AN could decompose any time. For example, in our scenario

(see Figure 3.5), assume that the agent profile representing Emily is looking to print a coloured document. When multiple Wi-Fi networks are found by Emily's laptop within a campus building, the PAN connects to a Wi-Fi network that has a *colour* network printer connected to it, instead of a Wi-Fi that has a *blank and white* printer. Herein, both network and service personalization takes place (to result in adaptation).

for each $ON_{ID} \in C_{type}$ *do*
if $(D_{prop} < D_{max})$
 if ON_{ID} *is the first* ON_{ID} *in a particular* ON_{CID}
  index ON_{ID} *to allocate leader as* ON_{LID}
  update association based on personalized ER states
  ON_{LID} *customized association in OS* $\leftarrow$ *true*
 else ON_{LID} *association* $\leftarrow$ *false*
for each $ON_{ID} \in C_{sub\text{-}type} \in C_{type}$ *do*
if $(D_{prop} < D_{max})$
 if ON_{ID} *is not the first* ON_{ID} *in a particular* ON_{CID}
  index ON_{ID} *to allocate node as* $[C_{ID}$ *or* $CR_{ID}]$
  update association based on personalized ER states
  perform local transforms based on indexing & D_{prop}
  $[C_{ID}$ *or* $CR_{ID}]$ *customized association in OS,* ON_{ID} $\leftarrow$ *true*
 else $[C_{ID}$ *or* $CR_{ID}]$ *association* $\leftarrow$ *false*

Figure 5.10 Personalized and localized CSON-D scheme with respect to node association.

Herein, dissemination is based on local request forwarding tables by employing a personalized cache overlay mechanism for active nodes in the POSpace, resulting in balanced contribution and high system throughput. We choose a local transformation based personalization scheme within the overlay space to achieve continuous adaptive optimization of the dissemination graph (i.e., star/mesh structure) to eventually achieve spontaneous propagation of personalized contextual information. Local transformation denotes data and structural changes that influence local nodes only to protect global nodes.

By grouping the overlay nodes as peer groups via semantic clustering, these transformations are performed within an overlay group and without user interaction. As each group operates independently, any limitation is usually limited to that particular group. Thus the cost to perform dynamic updates is minimized (refer performance evaluation results). As the POSpace is made up of multiple overlay groups instead of nodes directly, the overlay space will be altered (i.e., customized) with minimum node adaptation. Also, by locally altering the routing table specific to a particular overlay group in a light-weight fashion, the overlay-based routing is made to be highly dynamic and scalable.

5.6.3 Profile and Performance -based Customization

The overlay groups are adapted based on two types of customizations as a part of the previously discussed personalization scheme. The first type is profile-based, wherein the pre-defined policies, defining context types, are customized. These changes trigger the way in which the context clients are decoded, classified, and overlay associated (i.e., the basic CDP and POSpace functionalities are altered). Hence the overlay groups are formed in a dynamic fashion so that the scheme is applicable in real-time applications, wherein each group will be capable of querying the entire network. By doing this, (i) we find and disseminate a more accurate overlay node that matches the target application's demands, (ii) enable application-based personalization, and (iii) improve the overall performance of the dissemination mechanism by informed 'virtual' collaboration of the context clients. (outlined in Figure 5.11). The second type of overlay customization is termed performance related customization. Customization of overlays based on performance metrics is more complex than profile-based customization. Herein, the application developers or the composed ANs can specify their own resultant criteria such as cost and latency metrics, for example. The metrics denote the intended characteristics of the interaction layer of the CDON-D protocol. We achieve effective requirements matching via minimum adaptation by defining rules within PAs and performing local transformations at various instances. By incrementally matching the local transformations in the overlay operation layer, we eventually achieve global performance results in the interaction layer, transparent to the

end-user. By doing this, we optimize the POSpace and achieve performance related overlay customization as relatively outlined in Figure 5.11.

An example for profile and performance related customization in our running example scenario could be the selection of secure files from Emily's home and the selection of a Wi-Fi network based on signal strength, respectively. Herein, the external network's CIB, after incurring the user's profile and requirements, establishes the necessary connection as and when possible and performs CSC in order to acquire the required data. After appropriate processing i.e., for example, after altering the file format, the data dissemination process takes place.

for every verified context element (C_S and C_R)
  decode context based on customized knowledge base
  check for category based on customized policy tags
for every classified element (ON_{LID},C_{ID},CR_{ID}) in ON_{CID}
  initialize overlay cluster identifier
  (or) check for cluster conditions
  associate based on personalized ER states
while receiving node updates in ON_{CID}
  validate node inter- and intra- associations in-
  -customized overlays based on performance criteria
  update ON_{CID}
check location, neighbourhood, and load properties
if customization fails
  start over association procedure

Figure 5.11 Customizing the CSON-D scheme.

5.7 Assumptions and Constraints related to POSpace

Assumptions:

1) We do not focus on how the context data is managed as overlays. Management of overlays refers to administration, configuration, and deployment of underlying network resources.
2) The context entities are said to be in the overlay space as long as no error (or update) report is issued by the dissemination processor about the overlay associated entity.
3) The POSpace (and CDPs) of various ambient networks are already part of a trust model.
4) An ambient network's secure public database is accessed and used for context registration.
5) There exist device, node, and other sensors for Quality of Context (QoC) maintenance.

Constraints:

1) If different components are responsible for different actions such as sensing, refining, and dissemination, then the up-to-date requirements of the clients may not be simultaneously updated in all the components.
2) Excessive inter-communication is performed, which increases the overall cost and overhead.
3) Lack of context-awareness privacy in CSON-D due to the employment of a database listener. We could not limit the use of private context information to authorized entities only and we could not hinder identification of users, thus introducing security and privacy concerns.
4) Prevent and recover from failures when one or more components fail to seamlessly provide contextual information to ambient network entities..

5.8 Performance Evaluation

5.8.1 Implementation Architecture

The implementation details of the Pure Overlay Space for context dissemination are discussed as a sequence of events by following ambient network context protocol primitives [101], [105]. As a common step, all events are 'updated' in the database listener which persuades data-centric storage. Following the CDP related events discussed in section 4.4.1, the fetching process is initialized, thus initializing the POSpace. The agent tags are associated one by one in the POSpace in the data organization layer by following the spatial decoupling and semantic clustering processes. The resulting clusters, which could overlap, are further extended and optimized via the previously discussed context dissemination optimization techniques to FETCH the appropriate contexts. By supporting context access and organization, the context entities collaborate and disseminate in the interaction layer, whose implementation details are discussed in section 6.7. By locally altering the routing table (maintained in the CDP_KB) specific to a particular overlay in a light-weight fashion, the overlay-based routing is made to be highly dynamic and scalable.

5.8.2 Infrastructure

To affirm our context dissemination mechanism, we simulated the CSON-D protocol based on the POSpace and proposed optimizations, as an implementation prototype of ContextWare [7]--[8]. The simulation study is performed under realistic practical assumptions to analyze the approach's specific performance areas. We make maximum use of the available resources to enable autonomous flexibility with low persistent storage space. The protocol requires low persistent storage space and the deployed software/ hardware configurations change regularly to maintain cost-effectiveness. Overlays are simulated with increasing number of random overlay nodes while degree constraints are employed. For every overlay and in all instances, the 'average percentage' results are obtained by either starting off from zero or allocating optimum energy levels. With respect

to load, latency, activity, etc, we measure the consumed energy. While with respect to weight, degree, and stretch, we measure the remaining energy levels by consistently decreasing them until they reach a steady state. Java is chosen for programming this model since it is system independent and ensures adaptability. XML is used as the encoding language for context entities since it provides portability and flexibility. For implementing local database, SQL Server 2005 is used along with JDBC for storing and retrieving data. The prototype's components are java-based XML structures including the proxy overlay. The agent tags (representing context entities with unique C_{ID}) can become members of the various overlay groups, wherein namespace for these groups are chosen carefully based on their semantics. The overlays are managed using a java-based overlay control interface. If the pre-defined context types and sub-types fail to match during overlay association or if the overlay nodes are not found in the decoupled overlay space, the best effort scenario is applied. Herein, the undefined and unknown contexts are assigned in the pure overlay's remaining cluster space. The prototype's topology is made up of 1 to 25 ON_L, 1 to 500 C_S, and 1 to 10 C_R, though it varies between experiments.

5.8.3 Results

Scalability:

The POSpace cluster's functional constraint, scalability, is studied with respect to cost-ratio and latency-ratio as shown in Figure 5.12. The POSpace copes with the dynamic changing nature of today's networks and systems and ensures the availability of up-to-date contextual information. Once the POSpace is initialized and the overlay nodes are decoupled and grouped, the effect of context scoring scheme guided scalability is evaluated. It illustrates the average remaining energy levels of several clusters (represented as points in the graph) at different time intervals, in comparison to the theoretical optimal energy level. It can be seen that the steady state is reached within few seconds. As we have not tested the dissemination architecture in a real-time AN based scenario, the scalability for such cases is under question.

Load Balance:

Figure 5.13 illustrates the load distribution (rounded values) in the context specific overlay clusters. To evaluate load balance, we trigger node increment/ decrement at a random time. The five sample cluster's load balance at various time stages is achieved with the help of the context fairness scheme and managed by the POSpace. The $CSON_5$ represents 'unknown' context type, for example. To achieve this, availability of accurate, reliable, and up-to-date contexts, along with contextual information tailored to the target requirements in the POSpace is essential.

Bandwidth:

Figure 5.14 illustrates several clusters' mean in-bound and out-bound bandwidth for the same clusters and confirms that out-bound bandwidth is within limits. This is achieved by employing the benefit fairness scheme. This result along with the previous two contribute to the efficiency goals of our overlay based architecture.

Customization scheme's efficiency:

The customized adaptability schemes for dynamic dissemination with load balancing, cost effectiveness, and degree constraints consume negligible time. This creates an opportunity for composing context-aware services on the fly. We handle ubiquitous issues [115] by combining contextualization, customization, and personalization functionalities along with overlay networks. Based on these optimizations, the overlay routing and resource efficiency for sample overlay groups during low to high loads is illustrated in Figure 5.15. It exemplifies that the proposed schemes have better impact during high loads.

Benefits of overlay optimizations and average latency:

Next, we study the average time to successfully realize a context request with and without CDP-L2 optimization techniques that contribute to the efficiency goals. This is illustrated in Figure 5.16 (the points represent C_R) along with random failures. The average time denotes the end-to-end delay, while the number of overlay nodes denotes the combined

total of C_S and C_R within the CSON wherein the C_R is associated. The results show that most of the time consumption is when no CDP-L2 optimizations are performed and for larger number of simultaneous overlay nodes. The optimization based evaluations are made possible as we design and implement the optimization features as individual layers which can be added or removed as when needed. The results indicate that the effect of random failures without any optimization to tolerate faults significantly increases the time required to disseminate contexts. Aside from CDP-L2 benefits, the figure also exemplifies the impact of parallel operation by executing various number of overlay nodes (includes both C_S and C_R, except the C_R under focus). The results indicate that the time based impact is minimal. In general, though latencies occur at every stage of the dissemination process, we mainly focus on 'CDP to overlay leader' and 'overlay leader to MA' latencies.

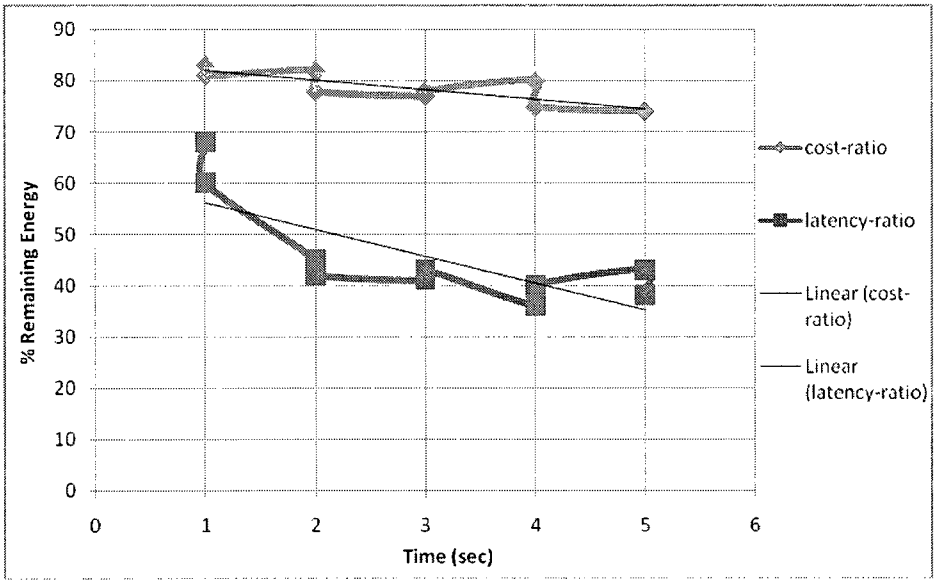


Figure 5.12 POSpace's decoupled cluster properties with respect to incurred cost and latency.

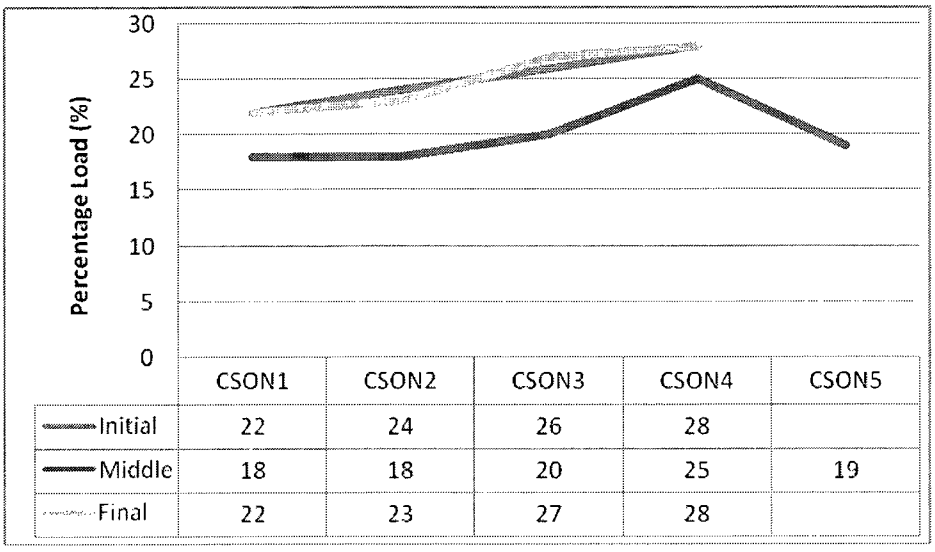


Figure 5.13 POSpace's load balance with respect to sample overlay clusters.

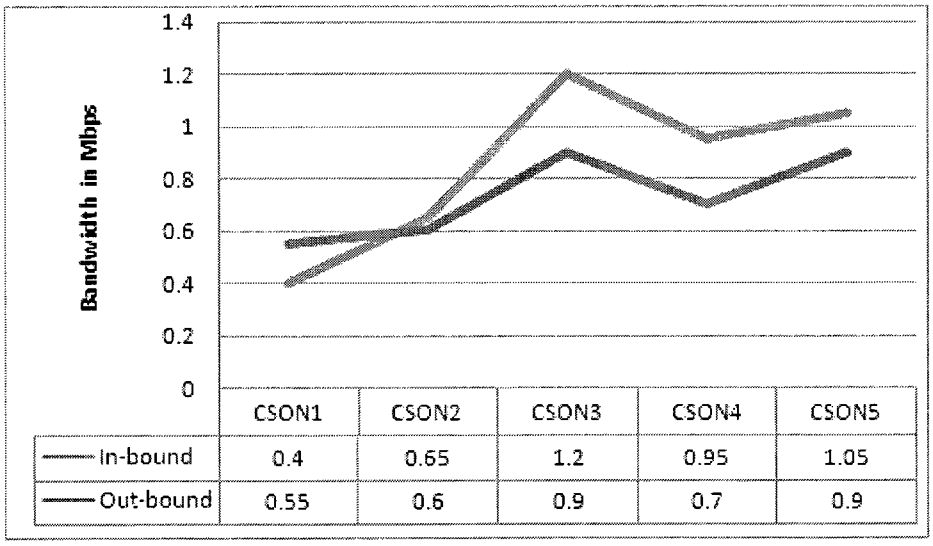
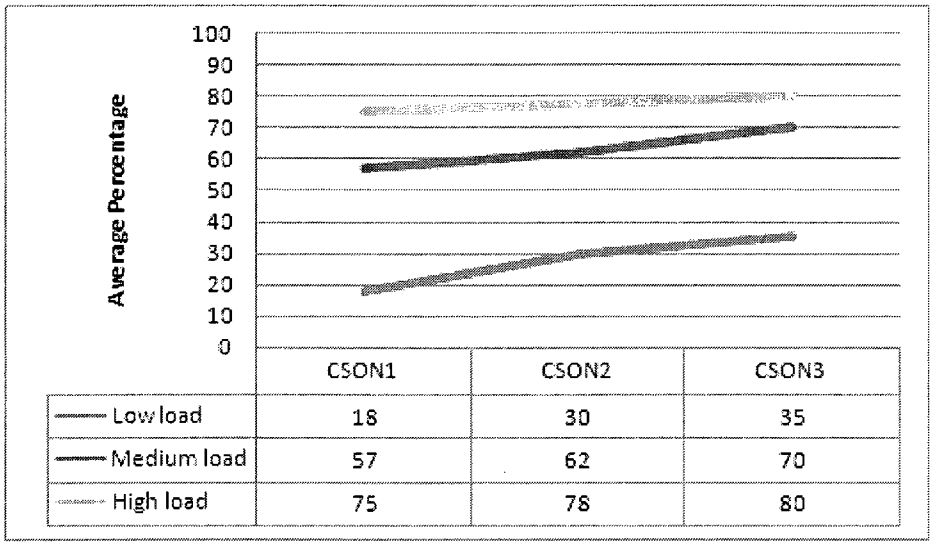


Figure 5.14 Mean bandwidth of several sample overlay clusters.



	CSON1	CSON2	CSON3
Low load	18	30	35
Medium load	57	62	70
High load	75	78	80

Figure 5.15 Sample overlay routing and resource efficiency by adopting custom functionalities.

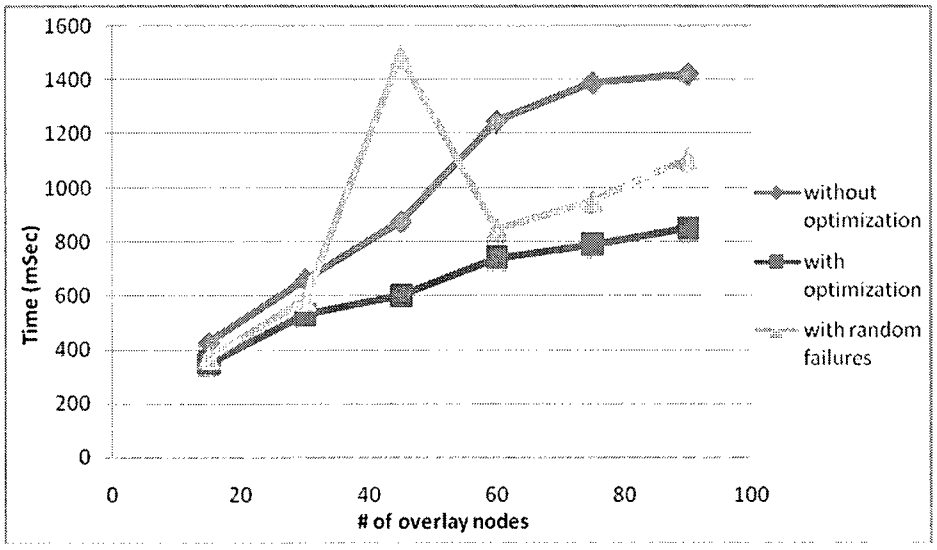


Figure 5.16 Average time to realize C_R requirements with and without (CDP-L2) optimizations in a perfect case, and with random failures.

5.9 Summary

This chapter discussed the design model of the Pure Overlay Space scheme. It employed spatial decoupling and semantic clustering techniques, along with related optimizations for this purpose. It discussed the context scoring and context fairness schemes that were employed to maintain dynamicity and fairness in the POSpace. It also discussed the hybrid agent overlay mechanism. The POSpace's basic functionalities were then enhanced by proposing techniques to infer personalization and customization. With the use of the running example from previous chapters, we illustrated POSpace's significant aspects. The chapter finally evaluated the performance results of the POSpace. After the POSpace discussion, the next step is the multi-level overlay model discussion.

CHAPTER 6
Multi-level Overlay Model

This chapter first elaborates the multi-level overlay networking scheme with its pros and cons. The chapter illustrates the CSON-D protocol by discussing the formation and maintenance of Context Specific Overlay Networks and Request Specific Overlay Networks along with the various technical challenges faced. The employed casting scheme and its delivery semantics are also discussed. The chapter then discusses the adaptation and fault tolerance modules to optimize the multi-level overlay scheme. The chapter finally evaluates the performance results of the CSON-D protocol with respect to multi-level overlay with the help of an example scenario.

6.1 Overview

Instead of using the overlay model like a simple service providing middleware [14] between C_S and C_R, we view it as a software infrastructure that supports easy building of context-aware applications in heterogeneous environments. As discussed in the following sub-sections, it provides context monitoring, (re-)configuration, and provisioning by answering the substantial questions, *what*, *when*, *where*, and *how*, with respect to context dissemination, and by taking the load off the resource-constrained contexts. The multi-level overlay model acts as a multi-level filter. More specifically, it is divided into two main levels which might be sub-divided into multiple sub-layers (see Figure 6.1), wherein:

- The bottom level is for deploying a different overlay network for every context type. It is made up of decomposed hierarchy of nodes (i.e., semantically characterized agent based overlay nodes and sub-nodes representing contexts) that are dynamically clustered together to form CSONs (Context-Specific Overlay Networks). Dynamic clustering (i.e., non-flooding based context grouping) is based on the involved overlay node's context attributes and adaptation (i.e., intelligence, personalization, and fault tolerance) optimizations (see section 6.2.4.1). Each of these nodes of the overlay network act as a context router (either inter- or intra- router based on the node status), and they are all distinct with priority rankings and status (i.e., leader nodes or mobile sub-nodes). Thus at any time, any one overlay node cannot contain both C_S and C_R. The bottom level is illustrated in the bottom part of Figure 6.1 wherein 'P' represents the leader node, while 'S' and 'R' represent C_S and C_R cluster sub-nodes respectively.

- The top level is for deploying a different overlay network for every context dissemination service or a group of aggregated services. It is made up of fully mapped and balanced sub-nodes, forming RSONs (Request-Specific Overlay Networks). Only the contexts that pass through the bottom level will proceed to the top level for dissemination. Overlay multicasting is performed through multiple unicasts, using forwarding edge nodes. Casting refers to the virtual coordination of the context entities in the interaction layer. The top level is illustrated in the top part of Figure 6.1 wherein 'S' and 'R' represent the selected C_S and C_R cluster sub-nodes respectively.

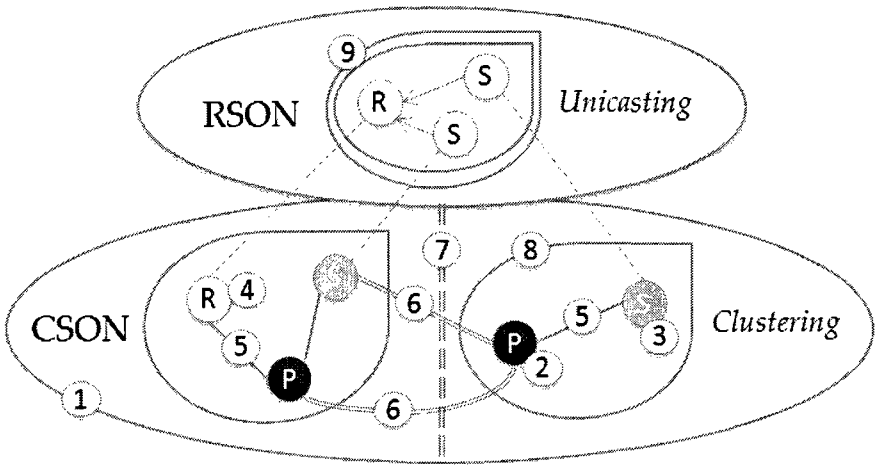


Figure 6.1 Multi-level overlay scheme overview.

We focus more on flexibility concerns via CSON and less on scalability via RSON. The 'dynamic' nature of the CSON-D protocol representing robust interoperability is vital to tackle the heterogeneity in contexts, and is necessary to maintain flexibility. To stress 'fairness' functionality (i.e., scalable balancing), we employ hybrid schemes wherein (i) both *on-demand* and *knowledge-push-based* provisioning are handled, (ii) both *piggyback* and *critical- or chance- based* optimizations are performed [99], and (iii) both *mesh* and *star* overlay topologies are employed. The multi-level overlay model achieves full matching, eliminates redundancy, and minimizes casting requirements, all with maintaining flexibility and scalability. This multi-level overlay architecture allows adaptation of the overlay nodes and networks to satisfy dynamic context requests, thus avoiding issues such as: path inflation, unbounded lookup times, disorganized clients, unbalanced load, and topology restrictions. With reference to Figure 6.1, the basic functionalities of the CSON-D protocol with regard to the multi-level overlay model includes (in sequence with respect to Figure 6.1 callouts): (1) overlay initialization, (2) leader agent node association, (3) C_S association as a mobile cluster sub-node, (4) C_R association as a mobile cluster sub-node, (5) intra-hierarchical clustering by semantic colligation, (6) inter-hierarchical association of overlay agents and sub-node search, (7) uniformly distributed decoupling, (8) adaptive overlay network formation, and (9) top level overlay network formation. These terminologies and functionalities are discussed in the following sub-sections as and when they appear next.

6.2 CSON

CSON, the core of our mechanism, serves as a base for contexts from producers and consumers to be independently mapped (as distinct context groups) based on their contextual attributes. These overlay based network entities are specific management channels which enable the dynamic transformation and provision of context information. As it is unrealistic to expect either C_S or C_R to manage the context transformations, these network entities are required for this purpose. The multiple CSONs are hybrid mesh of agent based overlay structures capable of disseminating context information in parallel. 'Hybrid' denotes concurrent intra- and inter- corporation of overlay nodes with varying structure and balance. 'Parallel' denotes simultaneous context handling. With reference to subsequent figures and algorithms, the CSON's construction and maintenance strategies are illustrated step by step as follows.

6.2.1 Initial Cluster Node Assignment

The agent-based modeled overlay nodes based on the POSpace design aspects form a virtual hybrid mesh of self grouped context entities. This CSON formation process is triggered as soon as the first agent tag (representing either C_S or C_R) is ready for overlay association. We start off with the initial node formation followed by context identification and address allocation (via the *UCI scheme*) based on the agent tag's properties (refer Figure 6.3). The location for allocation is determined by dynamic overlay spatial decomposition. The first associated C_S or C_R agent node of a pre-defined C-type is the cluster-head or leader node (for example, C_1 in Figure 6.2) because of the fact that all forthcoming node associations to its overlay space will inter- and intra- communicate with others via this node. These forthcoming nodes will act as mobile agent based cluster-nodes or sub nodes (for example, $C_{1.1}$ in Figure 6.2). Subsequent leader nodes follow proportional spatial allocation.

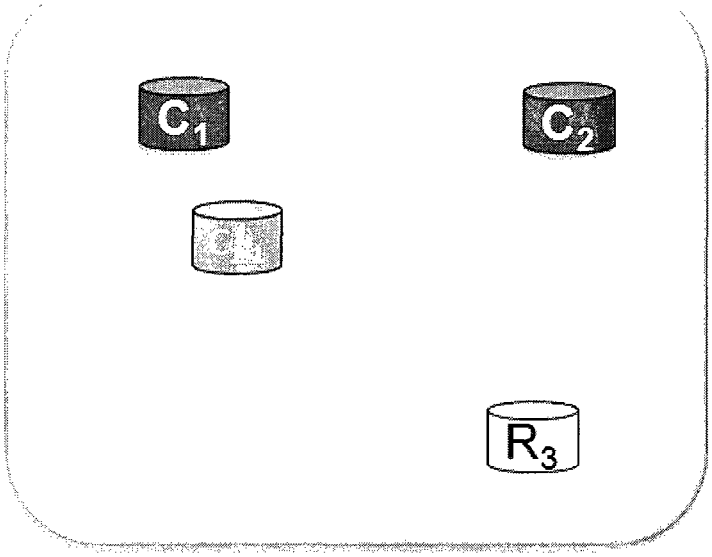


Figure 6.2 CSON functionality - Initial node formation and association of various types of contexts.

```
//VAR
  OS, OS_top; // multi-level overlay space
  L_prop, N_prop, D_prop; /* location, neighborhood, and content load properties */
  ON_ID, ON_LID; // mobile and leader overlay nodes
  ON_CID; // overlay cluster identifier with join/leave
  ON_CID-top, C_ID-top, CR_ID-top; // top-level overlay cluster identifiers
  C_x, C_x.y, R_x; // context-agent-tags
//Assume that the identified and verified ONs were already associated with initialized OS
//Initialization of overlay clusters when ON_LID joins
for every element (C_x, C_x.y, R_x) in ON_CID-list
  Check for overlay cluster 'null' condition
  On receiving C_x or R_x on ON_CID from agent manager
    Initialize overlay cluster identifier
    Associate C_x or R_x as leader overlay node
  Check load (L_prop, N_prop, D_prop)
```

Figure 6.3 CSON scheme - Initial cluster node assignment.

6.2.2 Semantic Cluster Formation/ Extension

Every overlay leader node of a new C-type forms a new cluster. Every sub node is allocated to appropriate leader node based on its semantics and thus forms extended semantic clusters. Hence, not only the node's location information is considered, but also its semantics. This semantic (intra-) grouping is represented in Figure 6.5, wherein the leader nodes take the form C_X and mobile sub-nodes take the form $C_{X.Y}$ (refer Figure 6.4). The main advantage of self-organized grouping is that no user interaction is required. Each of these overlay nodes can thus act as: (i) a leader node, (ii) a sub-node representing C_S, or (iii) a sub-node representing C_R, but not all three at the same time. For efficient clustering, subscription (both C_S and C_R) aggregation is essential. With the help of the agent manager (refer Figure 4.1), subscription aggregation is achieved by dynamically evolving the coexisting overlay networks. With the existence of multiple clusters at any given time, parallel dissemination is always likely.

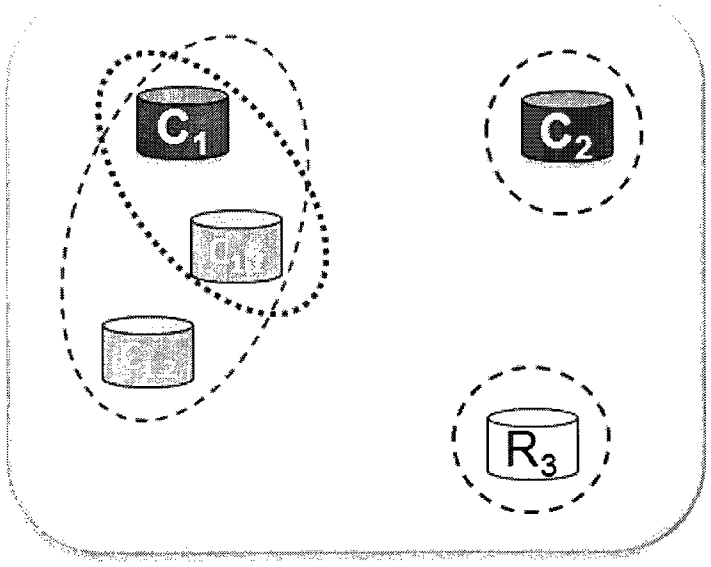


Figure 6.4 CSON functionality – Cluster formation and extension.

```
//Intra- and Inter- semantic hierarchical grouping
//CSON formation
Identify the properties of selected leader and mobile nodes
for each mobile that is a subset of the selected leader
    Merge mobiles with similar property constraints
    Filter mobiles that doesn't satisfy the constraints
Update, validate, & check load of merged/filtered mobiles
Forward mobile properties from one leader node to others
Other leader nodes disseminates the remote mobile's properties to their neighborhood
Repeat grouping process for inter-hierarchy
//Extension of overlay clusters when ON_ID joins
for every element (C_x, C_x.y, R_x) in ON_CID-list
  Check for overlay cluster 'non-null' condition
  On receiving C_x.y or R_x on ON_CID
    Check if cluster has only the leader overlay node or both leader and mobile nodes
      Associate C_x.y or R_x as mobile overlay node
      Spatial dependence
      Establish intra-relationships with other mobiles
      Associate the mobile node with leader overlay node
  Check load (L_prop, N_prop, D_prop)
```

Figure 6.5 CSON scheme - Cluster formation and extension.

6.2.3 Star-Mesh Strategy and Routing

The leader nodes act as in a star topology for internal communication (i.e., for local communication), while as a mesh topology for external communication (i.e., for global communication), as illustrated in Figure 6.6. By employing this strategy and limiting node information to be known only to neighbours, CSONs are more stable and scalable. A CSON reduces the problem of semantic-based searching to overlay routing. With low number of nodes for intra- communication we achieve more accurate and realistic measurements for

overlay updates with minimum overlay node selection and association. A network interconnected by only a pair of highest-degree nodes disseminates context more efficiently than a randomly interconnected network. The routing of a new node into the overlay space only affects a small number of existing nodes in a small locality of the coordinate space due to the CSON's localized construction scheme. Also, no routing protocol is needed as any end-system casting technique on top of the mesh is sufficient (see section 6.4).

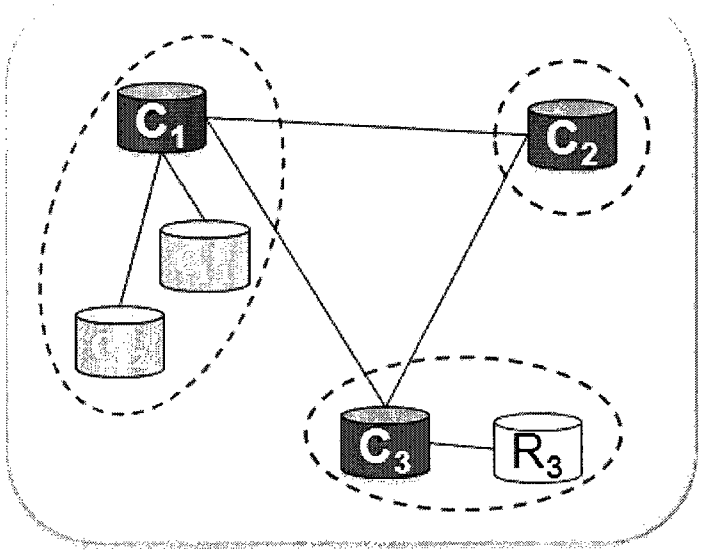


Figure 6.6 CSON functionality - Star-mesh strategy.

6.2.4 Periodic Cluster Maintenance

Fast re-structuring of overlays is almost always necessary in a heterogeneous environment. For an already existing semantic overlay cluster, the grouping structure is adapted by either dynamically extending or narrowing it. The interlinked CSON clusters are dynamically maintained based on node updates with the help of contextual mobile agents. The updates are inferred based on changing target application requirements. The maintenance scheme is outline in Figure 6.8. Herein, not only the load-balance is maintained but also the benefit-balance of the cluster nodes i.e., maintains fairness. For example, in Figure 6.7, the leader node C_3 moves to a neighbouring cluster as a sub-node $C_{2.1}$ and groups with another leader node C_2.

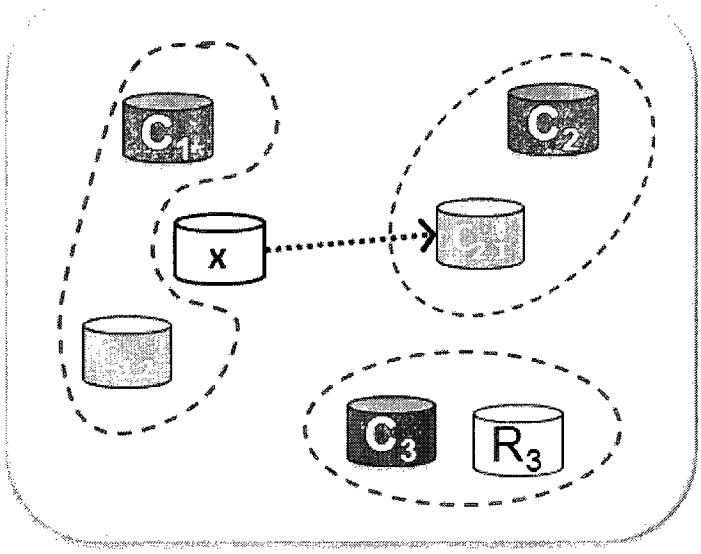


Figure 6.7 CSON functionality - Periodic Maintenance.

```
//Adapt to ONLID and ONID updates
for node updates in ONCID
   Identify and Validate mobile node association with -
         leader node on ONCID and inter-associations
   Update ONCID
for systematic node leaves from OS
   Update inter- and intra- relationships for leave event
   Calculate the neighbourhood node's score/ priorities
      Determine new leader node from ONCID-list
   Update inter- and intra- relationships for join event
for node failures in OS
   Go to: Failure Recovery //Figure 6.11
Check load (Lprop, Nprop, Dprop)
if adaptation fails, start over association procedure
```

Figure 6.8 CSON scheme - Cluster Maintenance.

The CSON formation scheme takes care of 'overlapping information' and 'reconciliation problems'. As we limit the CSON node degree and as the leader node's properties are based on finite pre-defined context-types, there might not be a CSON for agent-tag association at all times. During these instances, we use a *share capacity group* to temporarily associate context tags. Or, if provided with sufficient resources, multiple CSONs together form an overlay network. The coexisting overlays in such a scenario are dynamically evolved to result in parallel dissemination. These CSON clusters will remain alive as long as there is at least one context entity to serve, while idle C_S and C_R in the POSpace with no errors will not be saved. The periodic cluster maintenance is optimized via an adaptation module (discussed in the following sub-section), wherein we consider adaptation via intelligent learning-based personalization to disseminate 'exact' matching contexts.

6.2.4.1 Optimization – Adaptation Module

The adaptation module requirements and components are discussed with respect to the multi-level overlay based dissemination protocol (see Figure 6.1). The adaptation module copes with the dynamic changing nature of today's networks and systems and ensures the availability of personalized up-to-date contextual information in the POSpace. The basic principle of adaptability demands context updates to suit the requirements of a context service when the environment changes. In general, context services become personalized when they are tailored to the context and adapted to changing situation. The intelligence in contexts can be achieved by interpreting human behaviour, reasoning about it and acting upon it. The adaptation features achieved in our basic version using pre-defined supervised policies are further complemented by the novel personalized adaptations. With respect to the actual context propagation, the basic version lacked learning, which increased the failure rate due to incorrect or inaccurate contexts. The concept of saving C_S and C_R in the POSpace as long as no error report is issued by the CDP is not feasible anymore. The multi-level overlay model has the following unique features to aid personalized adaptation: combines partial matching to form full matching, eliminates unnecessary evaluations in the early stages of the scheme, and hybrid composition (both horizontal and vertical). By

personalizing a context service with respect to dissemination we try to understand and restrict the amount of system action possibilities for a specific context service action. The action, for example, could be re-creation of the context profiles, re-configuration of an existing profile, or inter- and intra- communication priority changes. By learning how a system might respond for a particular context, we not only perform context-specific adaptation, but also intelligent adaptation. Within a context node itself, an event could trigger different adaptations in various circumstances. This is valid, for example, when a user moves from his/her home domain to a foreign domain.

In our overlay-based dissemination architecture, the connections between overlay nodes representing end-hosts are influenced by context profiles, instead of end-to-end topology. Context profiles are XML documents containing information about a C_S or C_R such as its identity and any other required information to determine the view it will receive. Each individual context has its own local and foreign profiles. In ANs, the large collection of real time data could overload the overlay space with redundant and low-level data. By sequentially clustering them together based on either 'profiled' or 'derived' personal and infrastructure contexts, we end up with high-level global contexts that allow dynamic composition of diverse contexts. The adaptation module (the main components of which are shown in Figure 6.9) enables self-adaptation of context (and context service) changes by utilizing a set of semantic-based context agent tags managed by the *agent manager* (already a part of our dissemination scheme – see section 4.1.4). The agent tags are assigned new roles which include self-organized: (i) negotiation within CSONs, (ii) learning between CSONs, (iii) advertisement / feedback within and between CSONs, and (iv) aggregated dissemination within RSONs (i.e., context representation based on adaptation results). The home agent manager, for example, caches a context's data while it is visiting a temporary foreign network so as to instantly continue an open session upon its return. In order to achieve this profile-based personalized adaptation, each context profile and sub-profile is given a 'weight' based on its contents' (i) criticality, (ii) history, (iii) availability, (iv) local/foreign state, and (v) reliability, and is managed by the *weight handler*. This dynamically changing priority assignment and management is a key step to manage personalized decision making. It is achieved by making use of the routing table and

database listener (part of CDP), along with sequential priority assignment. For example, if there exists more than one matching C_S for a C_R, the C_S with high personalization factor is preferred.

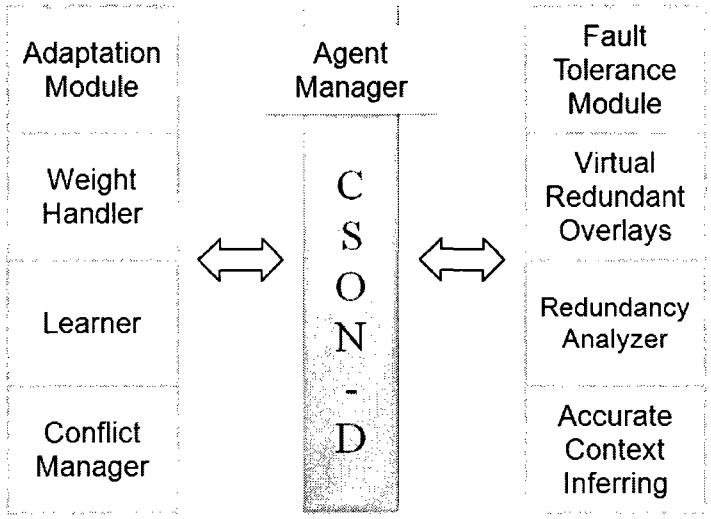


Figure 6.9 Main components of the adaptation and fault tolerance modules.

In our scheme, each agent tag makes the adaptation decision based on its own state and rules, instead of using a centralized manager, unless the node itself fails. Thus, instead of giving full control to a single entity, the adaptive context services are centered on itself. With such a decentralized design strategy, conflicts could occur between various agent tag decisions. To avoid unintended conflicts that may arise by uncontrolled adaptations we use a partially-supervised *conflict manager*. The conflict manager controls context aware adaptations by bi-directional context flows between the involved context entities (utilizes routing tables). This is done only within RSONs in order to maintain scalability. We utilize the learning (behavioural changes) patterns of an agent tag to discover hidden patterns exhibited by the learning mechanism and map them together in order achieve an intelligent learning scheme (i.e., the *learner* component). This is achieved by sequentially comparing an agent's original context profile with recently adapted (new) profiles, without depending on centralized pre-defined rule-based policies. This also voids the common constraints in static environment mappings. Therefore, when the agent manager detects a change in a context's movement pattern, it redirects corresponding context services to the moved

context's new location learned by semantic matching, or if feasible, it updates the context profile to suit the location's characteristics by intra-negotiation and re-configuration. This learning scheme is further discussed in section 7.5.

6.2.5 Failure Recovery

We mainly tolerate faults by employing virtual backup node functionality via the proxy overlay mechanism. As the agent's lookup service follows sequential priority assignment, the nodes that appear earlier in the semantic clusters are more important than those appearing later. When a leader node fails, an appropriate mobile sub-node becomes the leader node based on priority, dissemination history, and network dynamics (instead of a random sub-node) to fail proofing overlays. For example in Figure 6.10, when the leader node C_1 fails, the sub-node $C_{1.1}$ becomes the leader node based on priority, instead of the sub-node $C_{1.2}$. For the case of mobile sub-node failure, the regular cluster maintenance procedure is followed. Other employed fault tolerance schemes are discussed via the proposed fault tolerance optimization module in the next sub-section.

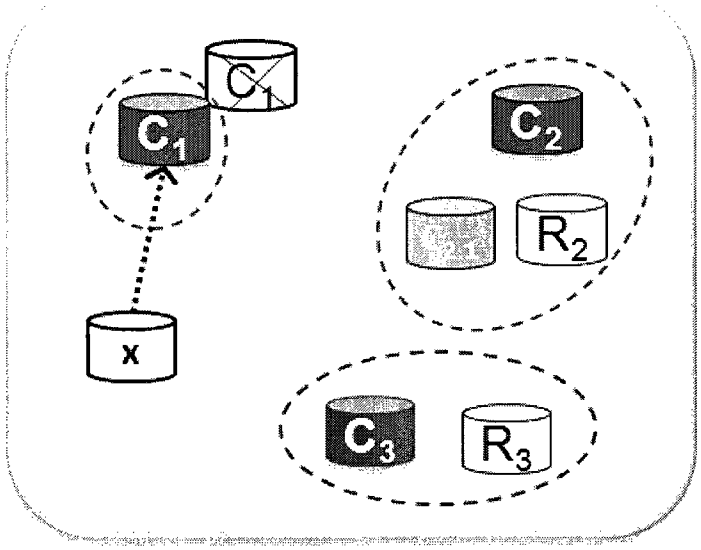


Figure 6.10 CSON functionality - Failure Recovery.

6.2.5.1 Optimization – Fault Tolerance Module

The adaptation and fault tolerance features, which complement each other, are proposed as an extension and elaboration of our basic multi-level overlay model. The main components of our fault tolerance module are shown in Figure 6.9. When considering fault tolerance in context-aware systems it is generally sufficient to limit the problem to fault-tolerant context dissemination [95], [98]. For the dissemination of context information to be reliable, accurate contexts must be available whenever it is required. One of the important characteristics of the dissemination protocol is the use of a fault tolerance module that provides capabilities to prevent faults in advance and recover from failures when one or more components/ context overlay nodes fail. Context overlay node priority plays a vital role to handle failures by alternating a node's role assignment based on its weight. We also handle basic faults in CDP during context pre-processing. Because of this fault tolerance based design, we do not separately design communication protocols to handle failures. Instead, we employ the following three steps within our CSON-D protocol to tolerate (software or hardware) faults:

1) *Fault detection*: A fault is detected when a rule that is supposed to be triggered for certain contexts fail or when context processing reaches a deadlock due to infinite looping. For example, a context node repeatedly searches for a non-existing context profile within a CSON, thus utilizing excess resources which could lead to failures.

2) *Fault classification*: In addition to the interoperability problems in context-aware systems, faults in general can arise due to device, application, network, or service failures.

3) *Fault correction*: Because of the fact that internet servers do not maintain an active connection throughout a C_R and significant message overload already exists in environments where dynamic contexts are involved, the commonly used *time redundancy* and *checkpoint* based approaches for fault correction are not feasible. This applies to other common fault tolerance solutions too including *rollback* and *restart*. For example, a single C_S may be a part of multiple context services, and

hence rollback or restart cannot be initiated. The proposed solution for fault correction is discussed as follows.

Identify and Validate redundant (C_{ID}, R_{ID}) association with ON_{LID}
Identify and Validate ON_{LID} inter-associations
On ON_{LID} or ON_{ID} fault detection
  Retrieve failed node's history, if available
  FOR known/repetitive failures
    Update inter- and intra- relationships
  IF more than one node redundant node is available
    Compare the weights of matching nodes
    Associate the selected node with ON_{LID}
  ELSE IF no matching redundant node found due to dynamic matching failure
    Perform inter-communication and repeat the previous step
  ELSE IF no matching redundant node found due to concurrent failures (or)
      the failed node is the only node in the overlay
    Associate contexts procedure is re-started
  ELSE Associate the node with ON_{LID}
  Determine new ON_{LID} based on pre-processed priorities, if ON_{LID} fails
Adapt associations in and within CSONs
  Update RT, agent manager, and all altered node's identifier properties
Verify ON_{LID} validity after adaptation
Verify location, neighbourhood, and load-balance properties
IF adaptation fails → fault correction fails
Associate contexts procedure is re-started

Figure 6.11 CSON scheme - Failure Recovery.

As we employ an overlay-based middleware approach for context dissemination, fail proofing the overlay structure is essential. Also, inferring contexts inaccurately is a key problem. In our fault-tolerant solution, we take necessary steps to handle these faults, yet maintain user transparency. Due to the nature of contextual nodes in heterogeneous

environments, a single point of failure is usually avoided by default, but delegating context services to low-fault nodes is not feasible. We thus employ a redundancy based approach in a novel way. We mandatorily compose (i.e., subscription aggregation) and utilize virtually redundant contextual nodes within a CSON / RSON. This also tackles resource management problem in redundancy based approaches. If the virtually redundant C_S or C_R can satisfactorily continue execution of the affected context service, it replaces the failed context node by re-configuring the overlay cluster, otherwise, inter-linked CSONs are considered. If more than one redundant context node is available, the *redundancy analyzer* utilizes the involved node's weights. Also, by making use of the intelligent learning scheme (see section 6.2.4.1), faults can be exploited even before the actual failure occurs, and corrected. Most importantly, the inaccurate context inferring problem is avoided due to the novel structure of the CSON / RSON itself (with conflict manager) i.e., high-level 'global' contexts are derived from individual 'local' contexts. If all fault-tolerant solutions fail, the affected context service has to be restarted and user transparency might not be maintained. The fault tolerance module's fault correction algorithm to recover from failures is outlined in Figure 6.11, as a part of routine CSON maintenance.

6.3 RSON

Our basic approach is to first divide the contexts, then form a mesh structure via bisected context key-spaces, which alters into a top level overlay (i.e., Request Specific Overlay Network, RSON – see Figure 6.12) for unicasting in the interaction layer to notify appropriate C_S and C_R. The top level overlay networks are fine-tuned to allow the configuration of high-level dissemination paths that meet the exact context requests based on hybrid optimizations, and the POSpace's and CDP's dynamicity and fairness functionalities. As soon as a C_R is associated with a CSON by semantic grouping, the top level RSON mesh structure formation is trigged. These are formed and altered as a part of *periodic cluster maintenance*, and are also based on the results of the dynamicity and fairness functionalities. For every C_R, a set of minimal overlay agents organize themselves into a top level overlay mesh of unicast connections and build data distribution trees on top of this overlay that exhibit: low weight, low node degree, low edge count, and redundancy.

To achieve this, a *DF search-based technique* (as in agent communications) is employed to search the sub-nodes of all interoperable CSONs under consideration, except the leader node to avoid redundant searching.

As the multiple context- and request- specific groups are constructed based on the semantic properties of the contextual agents, altering the agent tag formation process in the CDP will result in varying the overlay network i.e., end-system based personalized dissemination. This was discussed in detail in Chapters 4 and 5. By separating the RSON transport overlay, made up of balanced (partial structuring denotes a hybrid balanced mesh) and loosely-coupled (denotes no direct dissemination in this level) data forwarding end points, from the CSON, we use network resources in a per-request fashion. This enables us to form fine-grained overlay meshes on an as-needed basis (to save cost and latency), wherein every RSON is customized and discarded accordingly (piggyback optimization) i.e., the pre-defined rule-based policy profiles, defining context types, are customized. These changes trigger the way in which the contexts are decoded, classified, and overlay associated (i.e., the basic functionalities of the CSON-D protocol are altered). Hence the tags and overlay groups are formed in a dynamic fashion so that the scheme is applicable in real-time. The customization metrics denote the intended characteristics of the interaction layer of the multi-level overlay structure. We achieve effective requirements matching via minimum adaptation by defining rules within leader overlay nodes and performing intra-guided search (via local transformations) at various instances. By customizing these tags based on specific application needs, we enable application-based personalization and improve the overall performance of the dissemination mechanism by informed 'virtual' collaboration of the context entities. This guarantees overlay node location with minimum sub-node search even in composed ambient networks, as all active overlay node's properties are observed.

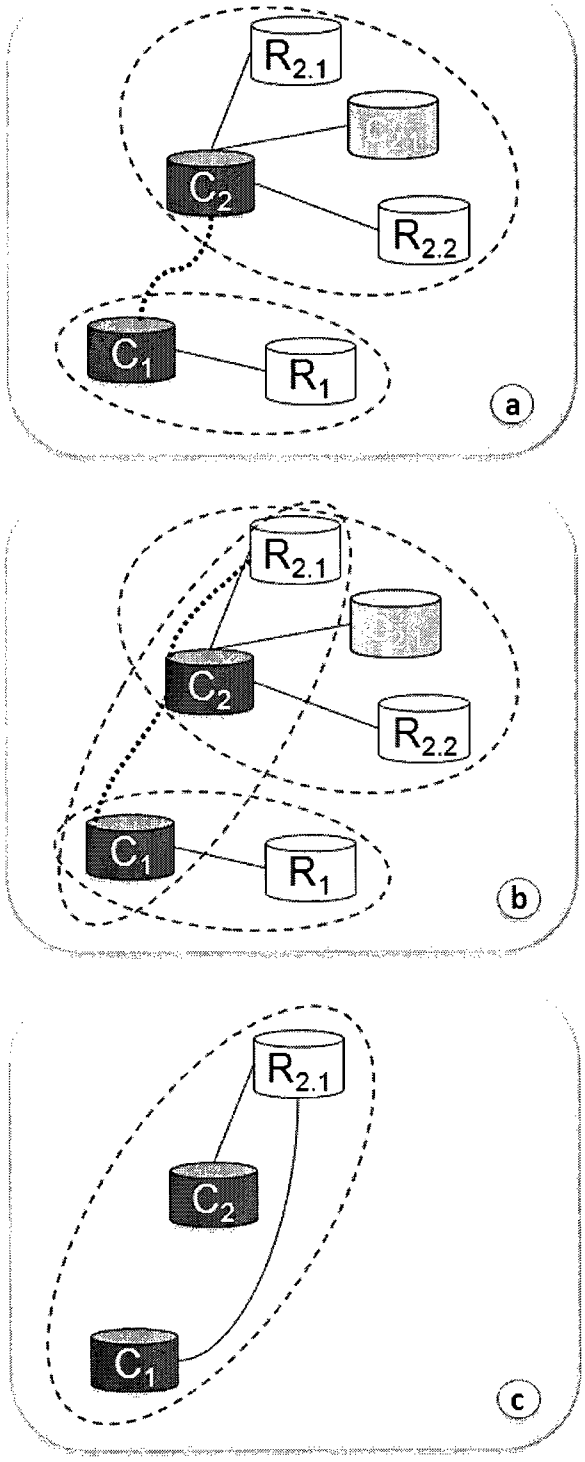


Figure 6.12 Step-by-step RSON construction strategy: (a) CSONs with inter- and intra- hierarchical links, (b) an RSON instance on top of two CSONs after leader- and sub- node search, (c) top-level RSON formation with unicast links.

In our RSON scheme, each node makes the adaptation decision based on its own state and rules, instead of using a centralized manager, unless the node itself fails. With such a decentralized design strategy, conflicts could occur between various contextual agent decisions. We avoid unintended conflicts that may arise by uncontrolled adaptations by controlling context aware adaptations by bi-directional context flows between the involved context entities (utilizes routing table). This is done only within RSONs (to scale up to ambient systems without having to deploy new equipment or modify existing protocols), as it is not cost-effective and scalable to do this in inter-CSONs. In addition, a *match-first* strategy is employed as illustrated in Figure 6.13 instead of flooding or distributed broker employment. Though this strategy is not scalable for CSON characteristics, it suits the 'minimal-node' RSON characteristics. This scheme also supports knowledge-push-based provisioning i.e., a hybrid approach. Knowledge push denotes middleware-based share-capacity group utilization via mobile agent characteristics. In RSONs, the data delivery path follows the balanced hierarchy structure with no additional route computations necessary. These points emphasize the RSON scheme's scalability along with the fact that all context nodes maintain the same amount of state information. Within the RSON, C_S acts as a virtual publisher(s), while C_R acts as a virtual subscriber(s) i.e., *a proxy-based publish/subscribe mechanism*. Either multiple C_R will subscribe to a single C_S, or vice-versa. By adopting such a structure, we disseminate only to a small number of nodes as matching decisions are made beforehand, and route hops and resource usage are limited. Interlinked RSONs (i.e., *mesh merging*) is a likelihood, wherein one node would be in more than one RSON. Finally, the actual casting of contexts to C_R entities is not associated with any overlay network interface as further discussed in the next section. This routine to achieve robust interoperability and optimized balancing is illustrated in a step-by-step fashion in Figure 6.12, wherein, for example, a CSON is made up of more than one RSN and a RSON is made up of more than one CSN.

for each R_x that is a subset of every ON_{CID}
  Identify the properties of selected C_x and $C_{x,y}$,
  for each R_x, that is a subset of $C_{x,y}$, which is a subset of the selected C_x
    //Overlay Matching
    //Leader and sub-node search and match
    for each neighbour node set ($ON_{ID\text{-}list}$)
    Check if R_x properties are matching any neighbour node set properties
      Extract C_x & other (request-) mobile node targets
      Add targets to match_set
    Validate, Filter, and Merge match_set
    Perform inter-hierarchical overlay grouping
    Go to: Minimized Casting Scheme //Figure 6.14
  Check load (L_{prop}, N_{prop}, D_{prop})
Update lookup, DP_kb, RT

Figure 6.13 RSON formation.

6.4 Casting and Delivery Semantics

As a globally deployed multicast service does not exist and as multitudes of problems such as manageability, lack of a robust inter-domain multicast routing protocol, scalability, and heterogeneity exist, an efficient casting process is essential. By exploiting various overlay multicast protocols for different purposes we conclude that 'multicasting requirements' and 'approach performance' are indirectly proportional to each other [18], [90]. Our objective is to keep the casting requirements to a minimum, for example, to achieve 'simple' direct and local unicasting between selected entities i.e., *critical* optimization. The constraints faced while designing and implementing such a casting-based approach include: (i) edge-node forwarding, (ii) distributed loosely-coupled mobile nodes, (iii) large number of nodes, and (iv) local transformations. The context transformations, as discussed before, are performed locally to continuously adapt data or structure locally so that global nodes are protected; thus achieving incremental global transformation of the

dissemination structure to perform Local Multiple Unicasting (LMU). During CSON / RSON grouping, if local transformation fails, a transformation based on only the target application's properties is employed. This is termed *probabilistic transformation* which follows *chance*-based optimization. Either many C_R could subscribe to one C_S, or one C_R could subscribe to many C_S, wherein C_S, intermediate forwarding nodes, and C_R form a dissemination tree (DT) for subscription. A discriminative treatment of CSON meshes is employed to achieve minimized casting. For example, even though $C_{X.3}$ is suitable for R_X, the sequential priority assignment focuses on C_X, $C_{X.1}$, and so on, and hence if the required context is found in high priority nodes we stop further searching. This is done to maximize the overlay performance and scalability. Herein, not only the load-balance is maintained but also the benefit-balance of the cluster-nodes i.e., maintains fairness. For example in Figure 6.10, a cluster-node moves to neighbouring node to become cluster-head. The RSONs utilize minimum resources as they can be discarded as soon as the casting succeeds. We trade off delay and out-degree in the balanced RSON to achieve maximum throughput.

As we focus on context-specific mesh formation and optimization against C_R requirements/ subscriptions in our research work, we do not explicitly address the issue of efficient routing and casting of messages from top-level overlays. Just before notifying the subscriber, the overlay network's final interaction and delivery semantics become involved. Casting within each top-level overlay mesh is performed by treating the mesh nodes as in a time-dependent DT. Since data distribution trees have degree constraints, we impose similar constraints while constructing the RSON mesh. Based on message ordering and prioritization, the edge nodes perform the casting process one-by-one. An example is illustrated in Figure 6.14, and the casting scheme is outlined in Figure 6.15. To sum up, a richer balanced mesh structure made up of unicast connections is essential. Once this is achieved, any end-system casting technique on top of the mesh will do the job without any routing protocol requirement.

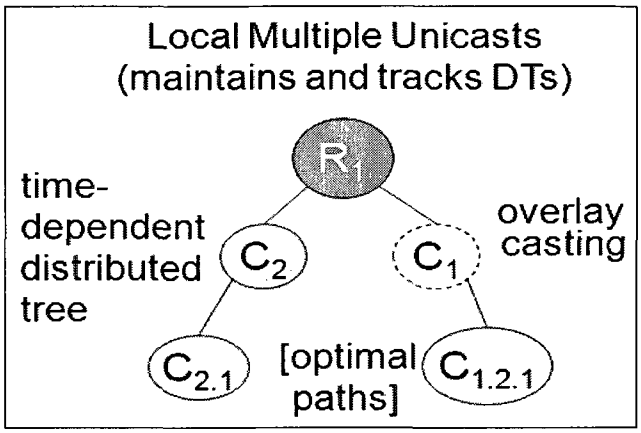


Figure 6.14 Casting scheme.

while OS periodic maintenance
Initialize top-level overlay space, OS_{top}
Initialize top-level clusters, $ON_{CID\text{-}top}$*, when* $CR_{ID\text{-}top}$ *joins*
Update merged/ filtered match_set as $C_{ID\text{-}top}$ *in* $ON_{CID\text{-}top}$
  Extract message 'msg' from match_set
Check if $ON_{CID\text{-}top}$ *is valid*
  Perform unique association
  Perform intra-hierarchical overlay association
  Follow prioritization order
  Route msg() //sub-proc
Update/Aggregate $ON_{CID\text{-}top}$ *with new contexts if invalid*

Figure 6.15 Minimized casting scheme.

6.5 Assumptions and Advantages related to CSON and RSON

Assumptions:

1) The requirements of the context entities are up-to-date in all the involved components of the dissemination protocol.
2) The centralized routing table and mobile data repository for local request forwarding and caching are up-to-date.

3) The context entities that are actively part of the multi-level overlay space are all valid and there exist no dummies.
4) Though the ambient network entities may fail, the main components of the multi-level overlay scheme do not fail.
5) There exist external mechanisms that handle the security, privacy, and trust concerns of context information.
6) There exist external mechanisms that take care of different actions such as context sensing and refining.
7) There exist external mechanisms that handle the security, privacy, and trust concerns of the knowledge base.

Advantages:

1) In line with the generalized context-aware middleware architecture [116], our overall architecture, abstracted into a hierarchy of layers, deals with heterogeneous context sensing/identification, context processing/reasoning for adaptation, context grouping for representation, and contextual interaction between various context-aware applications.
2) With respect to CSON-D functionalities, the virtual control and management of context clients is achieved without brokers by filtering unnecessary content, minimizing overlapping information, and tackling reconciliation problem (during multi-source involvement).
3) The to-be-disseminated context is not known in advance, in line with ANs.
4) We do not distinguish between C_S and C_R while they are being processed and associated in the overlay in order to tackle AN's heterogeneity.
5) The CSON's node- join and leave procedures maintain the overlay self-organizing property.
6) During CSON updates, if incremental local transformation fails, a probabilistic transformation based on the target application is performed to achieve self-healing.
7) Overlay re-structuring is achieved by adapting the (mesh) structure in a load-balanced fashion.

8) By adopting a mesh, we disseminate only to a small number of nodes as matching decisions are made beforehand and hence limit route hops and resource usage.
9) The CSON-D protocol avoids flooding.
10) The CSON-D protocol reduces the number of overlay networks by fusing similar contexts.
11) The CSON-D protocol reduces the routing state i.e., minimizes casting through dynamic and fair treatment of CSONs. It utilizes a global view of the system to build and maintain efficient overlay casting.
12) Along with these advantages, the dissemination protocol is also standardized and interoperable, so as to be utilized in networking environments other than ANs i.e., an internetworking solution.

6.6 Example Visualization

Based on the scenario, that is quite common in next generation network research, we evaluate our framework as well as visualize our dissemination protocol components usage in specific instances by constructing a MSC and summarizing the operations as illustrated in Figure 6.16. The MSC takes into account the assumptions discussed in previous sections, and assumes configuration, authentication, security, and policies. It also assumes that all ambient network clients have knowledge about and access to the CDP and there exists mobile databases in ambient networks. A user-centric approach is taken, wherein a device is moving between access points with different network access technologies, and sessions are moving between devices. As shown in Figure 6.16, the context entities are converted into context agent tags. These agent tags once processed in the POSpace, result in context specific overlays. Using the multi-level overlay scheme, a specific C_R is disseminated in the interaction view via unicasting. In our example, Sara's meeting location request (C_R) is disseminated from Emily (C_S). The utilized CSON-D protocol, transparent to the user, semantically processes the C_R and associates it with matching contexts in the overlay space, wherein the meeting information is disseminated via unicasting after RSON processing i.e., the meeting location request made by Sara is disseminated instantly via the multi-level overlay process, resulting in a customized RSON formation. As our approach is based on

loosely coupled nodes, and as we have pointers to context entities at all times, any update regarding the meeting context profile will be disseminated to Sara, wherein the history between the entities is updated.

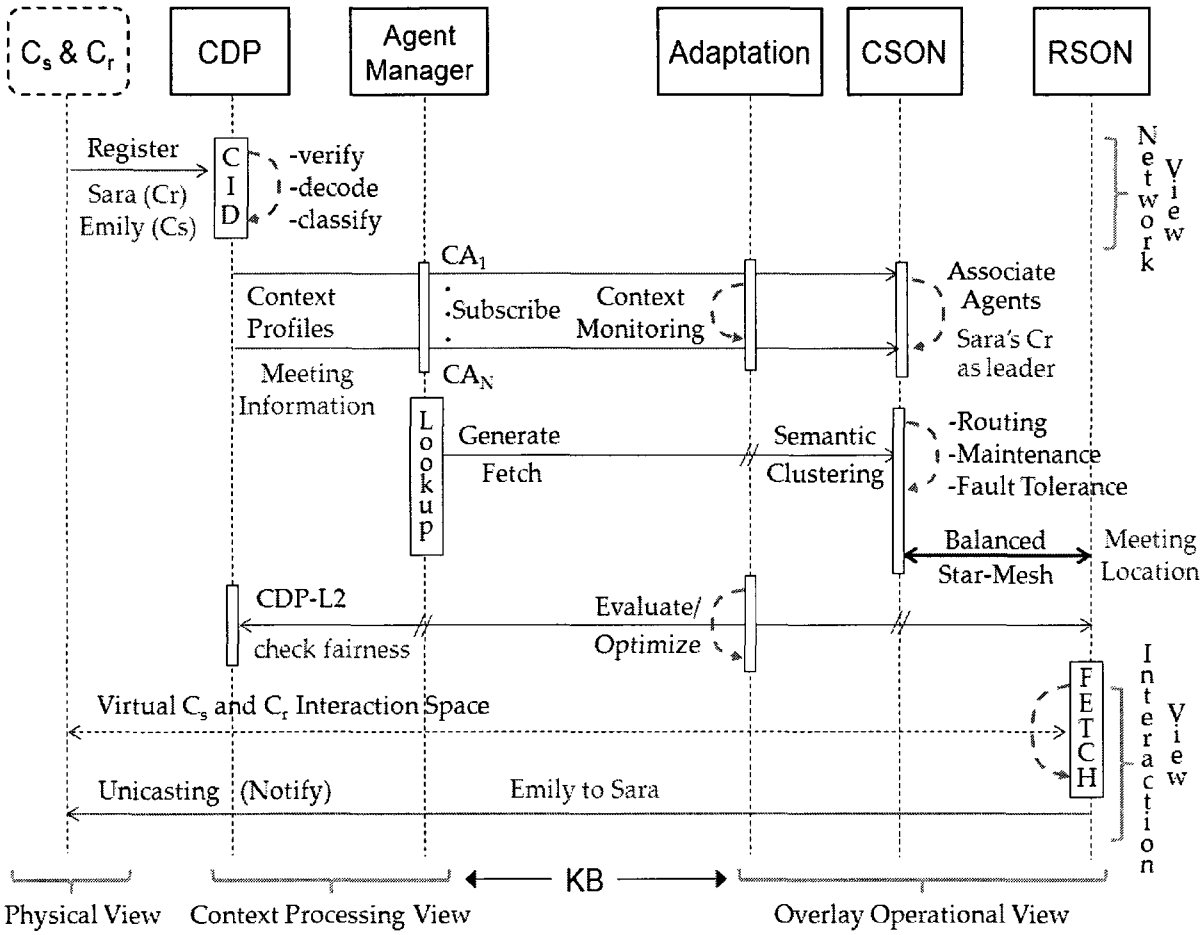


Figure 6.16 Visualization of protocol abstractions with respect to the example scenario using MSCs.

6.7 The CSON-D Protocol – Performance Evaluation

The implementation details of the multi-level overlay model for context dissemination are discussed as a sequence of events by following ambient network context protocol primitives [101], [105]. The implementation architecture and infrastructure are discussed in sections 4.4.1 and 5.8.1. To summarize, the implementation architecture's main events (see

Figure 6.17) include (i) Register, (ii) Resolve, (iii) Verify, (iv) Subscribe, (v) Lookup, (vi) (Optimized) Fetch, and (vii) Notify, along with database listener updates. Following the CDP related events and the POSpace related events, the context entities (i.e., virtual C_S and C_R) collaborate in the operational layer and disseminate via unicasting in the interaction layer. The results are NOTIFIED to the appropriate service providers and requestors, which is repeated for every matching context update. To affirm our context dissemination mechanism, we simulated the multi-level overlay architecture with proposed optimizations. The multi-level overlay-based dissemination scheme could also be used in P2P networking as it is based on P2P network management and dissemination middleware concepts.

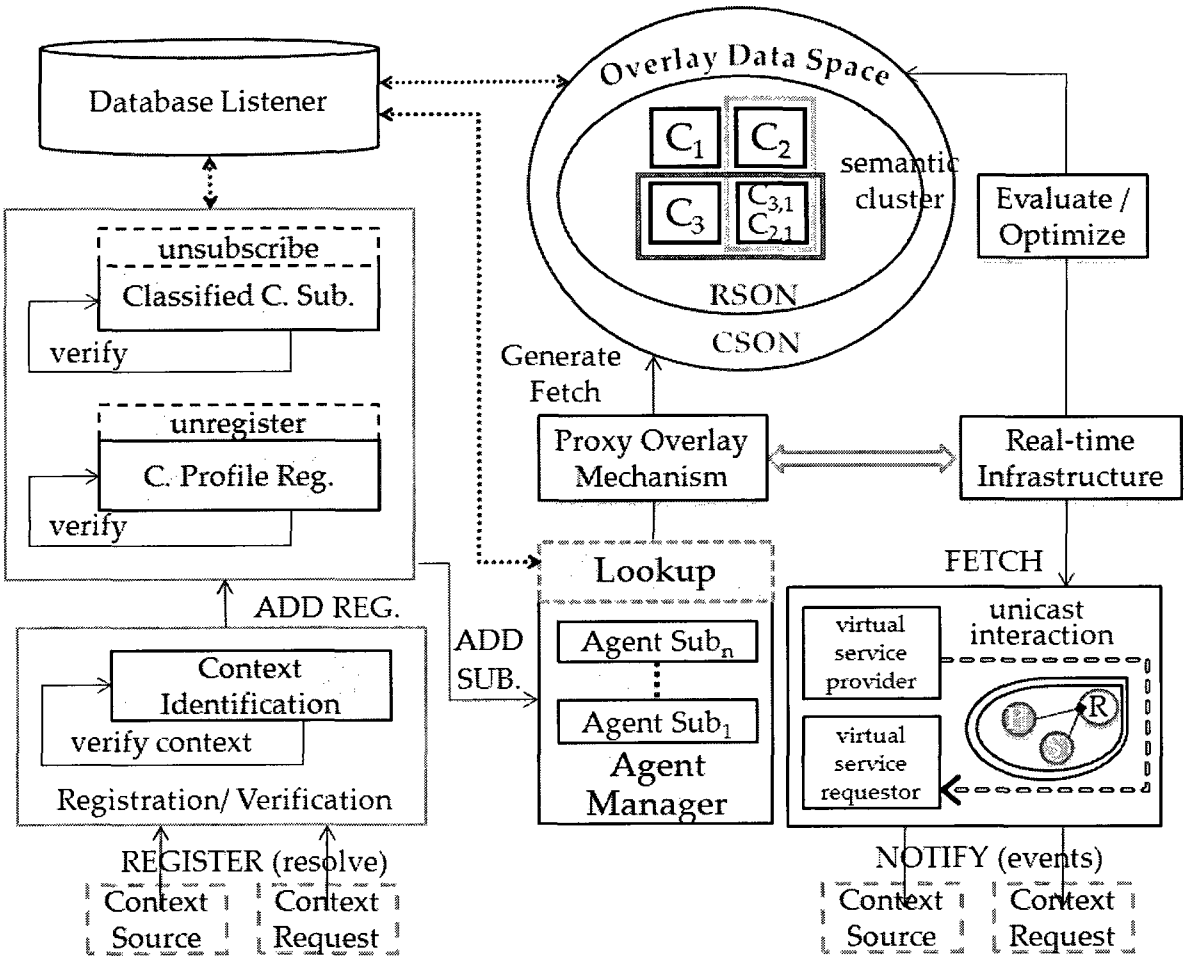


Figure 6.17 Context dissemination implementation architecture.

6.7.1 Results

Time delay vs. Overlay network events:

The additional features designed as optimization schemes to enhance the basic CDP and POSpace operation are implemented as customizable individual layers. Hence, even though the schemes' relate and cooperate with each other, each one is a distinctive component. By setting up triggering points, some of the performance metrics are evaluated. For example, a CSON cluster update could be triggered during a new node join, node exit, or while searching a node. Once the overlay networks are grouped, the time delays for various overlay events such as (i) overlay node entry, (ii) node exit, (iii) node update, and (iv) node search are measured and illustrated in Figure 6.18. It can be seen that the delay increases as more simultaneous entries take place. The delay is controlled by proportionally increasing the number of source-context nodes as request-context nodes increase.

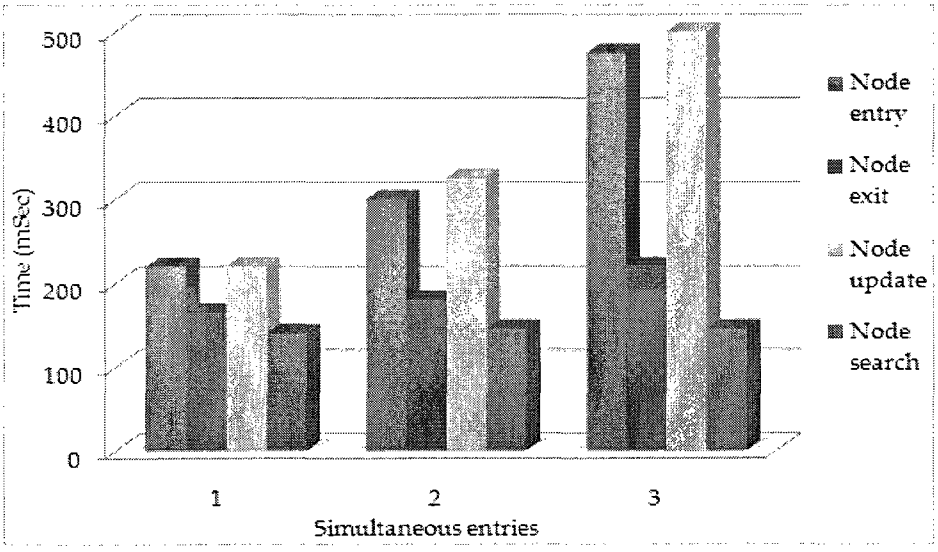


Figure 6.18 Practical time-delay of the CSON's and RSON's various internal events with respect to simultaneous entries.

Leader nodes:

The proposed adaptive scheme for context dissemination with respect to overlay load-balance during context node maintenance and updates, mean-delay for average hop count, and overhead effectiveness suggests that the energy consumption is within limits and is gradual with respect to lowest and highest cluster-head measurements. The overlay leaders balance their properties by optimizing the overlay structure. The leader or cluster-head size is based on the number of overlay nodes attached, wherein the node-degree does not determine node-capacity. This novel scheme has enhanced its related feature's results. This is illustrated in Figure 6.19 by measuring the number of cluster-heads against the evaluated feature's average percentage consumption in sample overlays at various time stages.

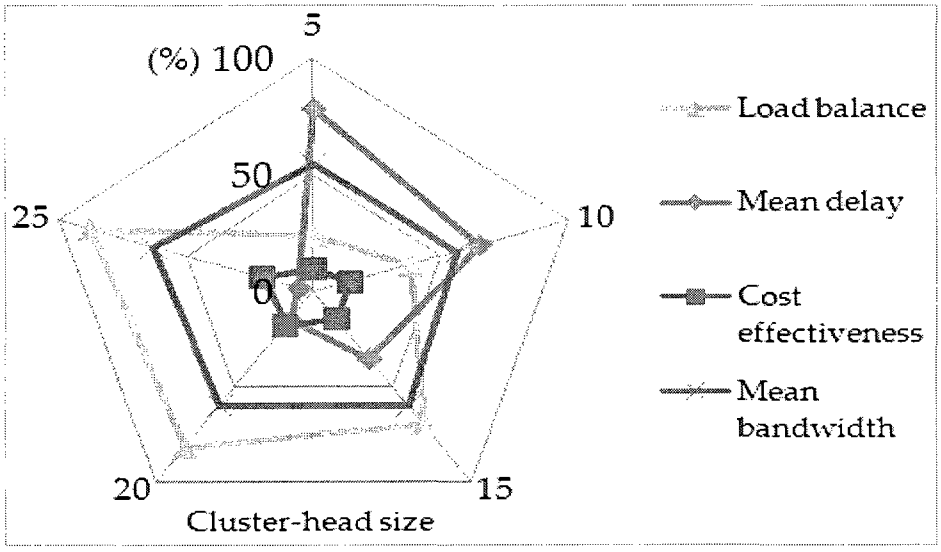


Figure 6.19 Performance of leader nodes of various sizes (i.e., cluster-head size) with respect to load-balance, mean-delay, and overhead (with respect to cost and bandwidth), due to the proposed adaptive module.

Time delay vs. Faults:

In Figure 6.20, the average time to successfully realize a context request in a perfect case and with random failures is studied. The average time overhead to tolerate random faults successfully is illustrated against the number of overlay nodes. The considered failure cases (FC) include: (i) cluster-head failure, (ii) cluster-node failure, and (iii) inferring incorrect context. The time consumed denotes the end-to-end delay for fault tolerance overhead only, while the number of overlay nodes denotes the combined total of C_S and C_R within the CSON wherein the C_R is associated. By employing a large number of contextual agents at one time, the figure also illustrates the minimal impact of parallel operation.

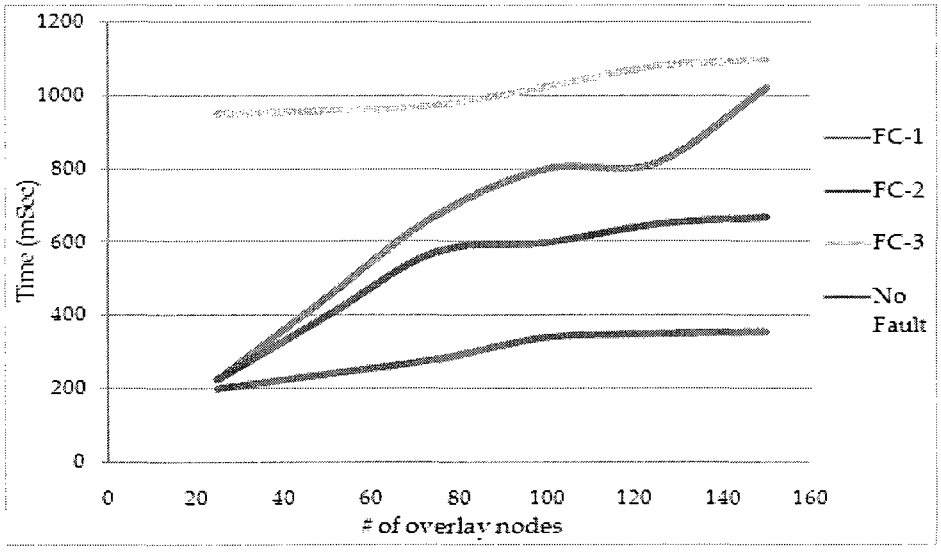


Figure 6.20 Average time delay to realize C_R requirements due to fault-tolerant solutions with respect to random failures in select FCs and without random failures.

CSON vs. RSON:

To evaluate the multi-level overlay scheme, we measure the overlay's success rate, time delay, and average characteristics. We achieve 100% success rate while semantically grouping agent objects as either CSON or RSON. It represents the correctness and performance of the overlay grouping process. We evaluate the overlay's properties by measuring the average weight, degree, and stretch for various matching C_R in CSON and RSON. Herein, with respect to a context dissemination request, the metrics denote (i) cluster-head priority, (ii) active cluster node joins, and (iii) cluster topology efficiency, respectively. The results illustrated in Figure 6.21 correspond with our multi-level overlay scheme discussed in section 6.1. Though an existing overlay network formation approach such as the chord protocol [117] could be improvised, it is not suitable enough as it mandates a specific network structure or assumes total location control. Thus, the performance hit when compared with native router-supported multicast mechanism is minimal.

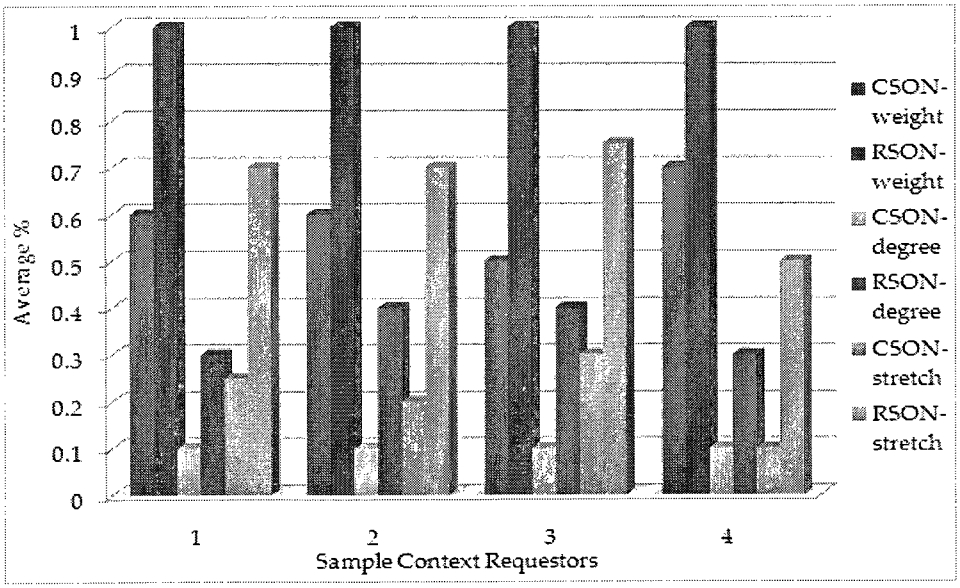


Figure 6.21 Comparison of initial- and final- level overlay network properties for matching context requestors.

Multi-level overlay scheme optimizations:

The adaptation and fault tolerance modules help to maintain load-balance (with low latency) by preventing overloading in context nodes by maximizing intra-context re-grouping instead of inter-context grouping. The optimization's practical results demonstrate better performance towards adaptive context dissemination. Figure 6.22 illustrates load distribution in sample CSONs at various time periods (TP), assuming that all the involved CSONs will remain active during those time periods.

In our scenario, consider the case wherein Sara communicates with her car's ambient network. When Sara's PAN is composed with the car AN, a new customized AN results based on their context profiles. Herein, various faults might occur while disseminating context information from one AN to another. For example, the address information disseminated to the car's GPS navigation system might be incorrectly inferred. Hence virtual redundancies are setup with a CSON wherein both network and service adaptation is required. In case multiple addresses are disseminated, the *learning history* as well as *weight handling* characteristics are utilized to select the right address order.

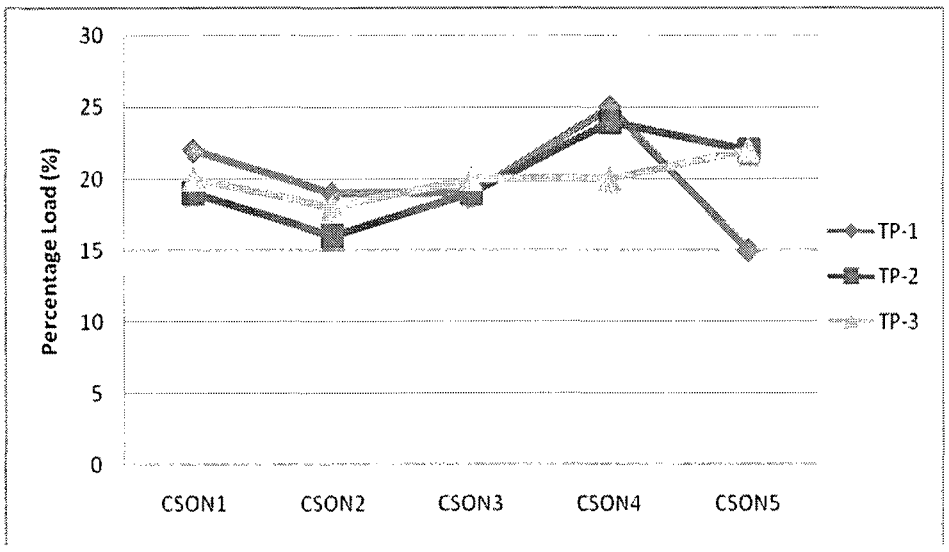


Figure 6.22 Overall load-balance and overhead in sample overlay clusters due to the adaptation and fault tolerance optimization modules.

Success rate vs. Multiple overlay agent nodes:

By evaluating the effect of multi-object dissemination and proposed optimizations in Chapters 4, 5, and 6, we conclude that as the number of (context) *requests* increase, corresponding increase in (context) *sources* is necessary i.e., they are relative to each other. This increase is necessary to take advantage of the optimization techniques. This is illustrated in Figure 6.23, wherein we compare the success (indicates the creation of a RSON with at least one fully matching context) rates of different number of parallel C_R (herein, 1, 10, and 100) with respect to number of C_S (herein, 0 to 300 with 50 C_S increments). The 'BO' and 'AO' in Figure 6.23 indicate *before optimization* and *after optimization* cases. Herein, the optimization schemes are developed as individual layers, which include: the monitoring scheme, the proxy and mobile agent functionalities, customizable overlay characteristics, and hybrid mesh scheme. The possible cases wherein the overall success rate is more than 95% are also illustrated.

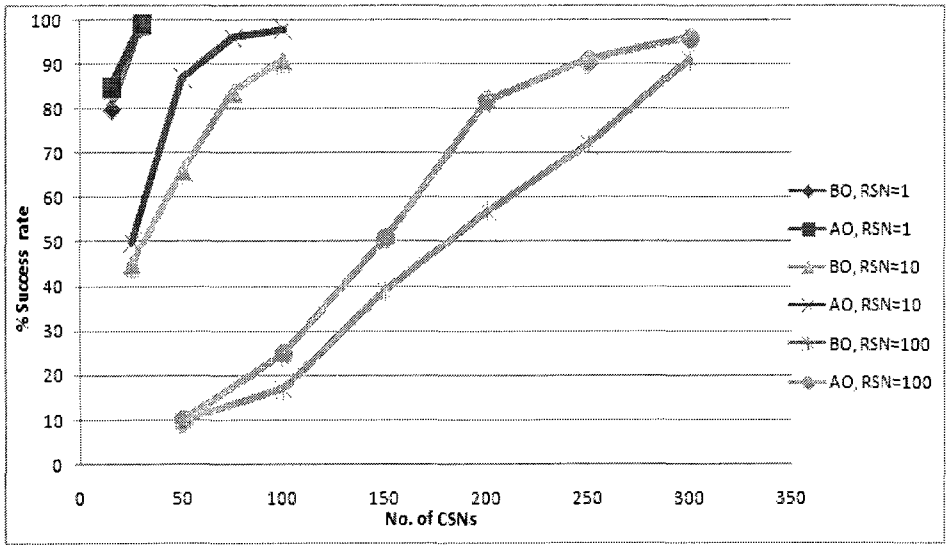


Figure 6.23 Before and after optimized success rates for parallel C_R = 1, 10, and 100.

Overall time delay of the CSON-D protocol:

The percentage practical time delays of the CSON-D protocol's various events are illustrated in Figure 6.24. Herein, the protocol events mainly considered include: (i) the pre-processing CDP functionalities, (ii) POSpace management processes, (iii) context agent formation and related functions, (iv) context monitoring scheme, (v) multi-level overlay scheme, (vi) adaptive optimization frameworks, and (vii) casting and database updates. As illustrated, the proposed scheme's and optimization's minimal time consumption is traded off to achieve successful and accurate dissemination. As the PCs used for simulation have high processing power, the implementation architecture consumes less time to execute compared to the time consumed for data transmission.

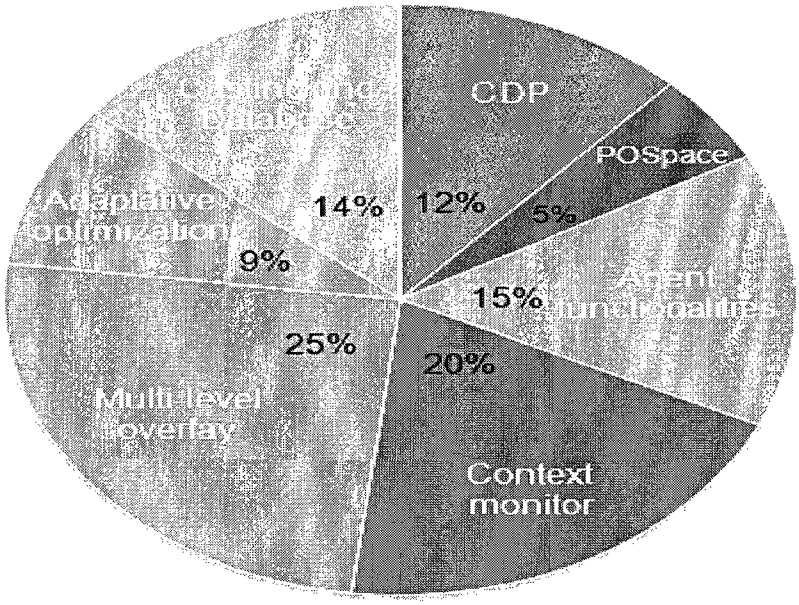


Figure 6.24 Practical time-delay (in %) of the dissemination protocol's various main events.

6.8 Summary

This chapter first elaborated the multi-level overlay networking scheme with its pros and cons. The chapter illustrated the CSON-D protocol by discussing the formation and maintenance of Context Specific Overlay Networks and Request Specific Overlay Networks along with the various technical challenges faced. The employed casting scheme and its delivery semantics were also discussed. The chapter then discussed the adaptation and fault tolerance modules to optimize the multi-level overlay scheme. The chapter finally evaluated the performance results of the CSON-D protocol with respect to multi-level overlay with the help of an example scenario.

This concludes the discussion about the design and implementation of the core CSON-D protocol in Chapters 4, 5, and 6. As an extension to the proof of concept of the CSON-D protocol, the use of context dissemination mechanism along with the proposed context monitoring scheme in a real-time application is discussed in the next chapter.

CHAPTER 7
Towards a Real-Time Context Dissemination Infrastructure

This chapter first provides an overview of the role of context monitoring and context dissemination in a real-time application. As the real-time application is based on mobile asset tracking concepts, their related work is discussed. The chapter then elaborates the real-time application itself i.e., a geo-tracking scheme which has its own algorithms and strategies, to track and mange mobile assets (i.e., contexts). The requirements, specifies, and the approach carried out to monitor these contexts are discussed next. This context monitoring scheme is the basis of the real-time application and is a part of the context dissemination infrastructure. To achieve efficient context dissemination in this real-time application, we then discuss approaches that were proposed in previous chapters and propose new approaches to aid the context monitoring scheme. The approaches mainly include *dynamic adaptation*, *fault tolerance*, and *a personalized learning engine*. The chapter finally evaluates the performance of the context dissemination infrastructure with regard to the mobile asset geo-tracking application. We also implemented a server-side user interface for managing the disseminated data.

7.1 Overview and Background

Presently, smart phones with built-in Global Positioning System (GPS) and Bluetooth capabilities are more affordable, resulting in numerous novel GPS based applications. Thus, the real-time application that we considered is one such application which is essential for many organizations to keep track of their valuable contextual assets (including smart phones and PDAs) that are on the move. In such *mobile asset tracking* applications, as the tracked GPS data's volume and frequency is directly proportional to the mobile asset's resource usage, the common fixed data flow based technique for forwarding location information at any particular instant may utilize more or less resources than it actually requires as it is static in nature i.e., most of the data transmitted over the air for tracking information (such as position and velocity) is on a periodic basis and is not based on real-time conditions. Since the communication channels used for such transmissions are considered to be relatively expensive resources, transmission of the GPS data at a fixed transmission rate is cost ineffective [19]. While vehicle powered or laptop powered GPS products are more reactive to receive/disseminate location information, a resource constrained mobile asset is not so favourable. Randomly reducing the message reporting frequency to reduce cost and resource usage could be serious for some mission critical applications. GPS-less location algorithms using only RF signals in Wireless Sensor Networks (WSN) for the estimation of a node position is a possibility, but they might result in severe distance errors. Hence a balancing technique is essential to reconcile this resource usage with a mobile asset's resource availability (and also to balance transmission cost and route accuracy). Therefore, novel context monitoring and dissemination are applied to transform a sequence of location measurements (with inherent errors) into smooth and reliable estimates of the dynamic state of the asset.

Based on the studies and motivation discussed in section 1.2, we designed and implemented an adaptive learning based scheme, which utilizes context monitoring and dissemination schemes, to make an optimized judgment of data transmission to: (i) Reduce superfluous retransmissions, and (ii) Increase tracking and routing precision by making sure that the highly relevant data are transmitted. The tracking software (i.e., the GeoTracker

software) provides a common operating picture through an automated localization system that provides a control center with frequent messages that include the mobile asset's data. This introduced a need for an environmental condition aware (i.e., context-aware) infrastructure. We infer context-aware and policy-based data management and dissemination schemes at the *tracking server* to logically and dynamically adapt the daily tracking operations of the client. We aim to make the adaptive tracking algorithm intelligent with reduced overhead and increased user-transparency with geo-referenced predefined routes and route characteristics (such as area density), and by using mobile agents. We study and apply Artificial Intelligence techniques for customized routing and route tracking based on the requirements of the target application i.e., the network sends only the important and understandable information and discards the rest.

As the real-time application is for mobile asset tracking based on geographical tracking, we study its related work. Most of the asset tracking systems are based on RFID, Wi-Fi, cellular, WSN, or GPS [19]--[20]. In the case of RFID, the RollCall system [118] by G.D. Bhanage, et al. is a good example of a low-cost, long-lived, and uninterrupted asset tracking system. They attach active RFID tags to assets which periodically broadcast their presence (via IDs) so that missing assets can be quickly identified. The benefits include low-power consumption from short transmissions, hardware cost, and small tag size. However many practical RFID and active-tag systems can only identify (mostly immobile) assets, but are far from tracking it in real-time. For example, a system for an educational institution to automate students' attendance is proposed by F. Silva, et al. in [119], but it doesn't track their movement. Active tags are also employed in [120] by R. Beuran, et al. to develop a pedestrian localization system. The data communication and processing features of active tags are used in real-time so as to provide to a centralized engine the information needed to dynamically calculate the trajectory and the current position of the user carrying the active tag. The main problems encountered include battery capacity reduction and realistic experimentation constraints. They overcame these problems using a wireless network emulator so that realistic wireless communication conditions between active tags were recreated. In [121], J.H. Youn, et al. discuss a WLAN-based real-time asset tracking system that employs Wi-Fi tags to smartly track mobile assets in busy and cumbersome

environments such as hospitals, transport terminals, etc. Due to the light-weight nature of WLAN infrastructures, WLAN based tracking systems provide a cost-effective solution to the lack of asset visibility problem when compared with cellular or GPS based systems, but often depend on learning techniques (e.g., RADAR [122] from Microsoft research) to utilize existing WLAN infrastructures.

While the most successful tracking systems use GPS for outdoor mobile assets, inadequate coverage in closed and complex environments prompts GPS in conjunction with WSNs. It is ideal to track mobile assets in hospital-like environments via WSNs. K. Kim, et al. talk about a medical asset tracking application using WSNs in [123]. In [124], S.J. Kim, et al. utilize a detachable ZigBee-based sensor module to track the location and status of the assets, wherein communication between sensors is required to achieve accurate and low-cost localization. They proved that the Zigbee-based low power asset tracking system is extremely effective in monitoring mobile assets. In the case of [125], N. Rajendran, et al. use a cost-effective asset tracking system based on sensor networks for asset theft detection. They divide the environment to be protected as clusters. Each cluster would contain a certain number of assets to be tracked and a base station that controls the sensors attached to the asset. The system reads the alert conditions to track a theft. An optional choice for indoor environments is the use of wireless assisted GPS (A-GPS), which provides an average accuracy of 5–50m (for e.g., gpsOne [126]). A-GPS receivers use network elements such as assistance server or other data from a network, to deal with standalone GPS receiver constraints. Several corporations are developing or have developed commercial asset tracking solutions using the above discussed techniques. Examples include: (i) Mobile Asset Tracker from CMO Global Services [127], (ii) MobileTrak 4.1 from RF Code [128], and (iii) TrackServ from Maptuit [129]. For outdoor environments, an asset tracker using off-the-shelf cellular technologies and GPS solutions is developed for tracking moving vehicles in [130] by G.H. Truelove, et al. A java program running on a GPS equipped Motorola iDEN handset is used to monitor the geo-data. The program selectively filters position updates and periodically forwards position information to the database server via the cellular service provider's data network, similar to our geo-tracking scheme. Their user

interface (UI) uses web-based and client-server map display solutions for displaying an asset's position and status.

With regard to the mobile asset tracking application, other relevant schemes that we studied, as a part of our research, include:

- Neural Networks
- Fuzzy Computation
- Prediction Techniques
- Evolutionary Computation
- Genetic Algorithms
- Self-organizing Maps
- Ambient Intelligence

7.2 Real-Time Application – Mobile Asset Tracking

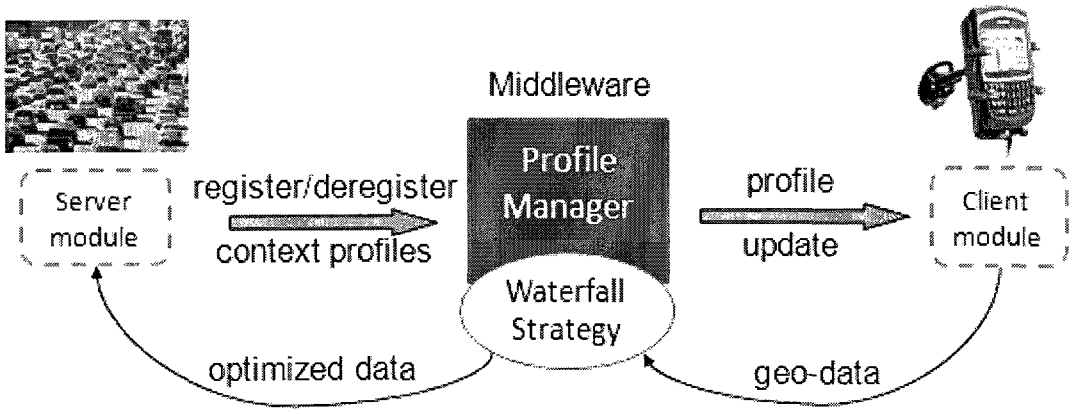


Figure 7.1 An outline of the mobile asset geo-tracking application.

An overview of the architecture employed to track routes based on geographical coordinates and report the tracked routes to the database server is illustrated in Figure 7.1. As seen in the figure, the architecture is divided into three basic parts: (i) client-side processing, (ii) middleware, and (iii) server-side processing. The client-side processing is

the representation of the mobile asset's actions including GPS signal reception, while the server-side processing represents data parsing, database update, and GeoTracker UI. The middleware represents client-server integration, which acts as the base for geo-data to be processed / monitored and context profiles to be processed / disseminated.

We simulate a GPS equipped Smartphone (currently, BlackBerry) mounted on the asset as an asset tracker and create an accurate and cost-effective route tracking mechanism for the asset. The mobile asset is loaded with our tracking software, the client-side GeoTracker, executing to record, optimize, and analyze the asset's geo-data. There are many variables that effect the GPS information, including *Latitude, Longitude, Altitude, Date, Time, Speed,* and *Direction.* The asset's raw data is packaged in a specific format understandable by the server-side GeoTracker software in the base station (i.e., the application server with the proposed adaptive routing strategy). As a GPS receiver cannot send the received data to the server by itself, cellular network connectivity is usually needed. The objective of the application is to improve the selective reporting procedure to achieve:

1) Optimization of data bandwidth used to reduce communication costs,
2) Selective reporting to generate better route traces fitting the geographical maps, and
3) Save battery life of the mobile devices.

The GeoTracker with adaptive routing is tested over simulated and real-time conditions using the BlackBerry API to generate a mix of data transmit patterns. The initial phase started with designing and implementing the basic geo-tracking algorithms on mobile assets to improve data transmission efficiency, wherein the GeoTracker is loaded in the mobile asset to report location data on a fixed interval. After the tracking algorithms were set up, the analysis of the reported geo-data is done to understand the behaviour of typical traffic generated in various demographics. These algorithms are further enhanced to add more features which allow the user to optimize (i.e., adjust) the value of reporting interval and other variables. There are four algorithms using which our GeoTracker application sends geo-data to the database server. The algorithms, based on the above discussed variables, techniques, and paths are described next as follows.

7.2.1. Fixed Algorithm

The fixed interval algorithm is derived from a GPS functionality that is commonly used. As the name indicates, the fixed algorithm sends specific data to the database server after every fixed interval of time i.e. it uses one variable to keep track of the time change in the mobile asset and relates it with frequency (see Figure 7.2). It knows at what time the last point was reported and when the next point will be sent. If the calculated time interval is larger than the predefined *Minimum_Update_Interval* (generally set as 1 minute and up), the device sends the data to the server (which is later analyzed), otherwise it discards it. This simple reporting strategy falls short in terms of cost vs. benefit as the customer usually pays a penalty (to the service provider) for transmitting excessive data.

```
SET Minimum_Update_Interval = xxx (in sec);
IF   Getting Data_New == true   THEN
       Retrieve previous reported data, Data_Reported , from memory;
       Interval = Time(Data_Reported) - Time(Data_New);
       IF   Interval > Minimum_Update_Interval   THEN
              Report Data_New to server;
              Store Data_New to memory;
       ELSE
              Discard Data_New;
       END IF
ELSE
       Keep listening location data from GPS receiver;
END IF
```

Figure 7.2 Pseudo code of the fixed time interval algorithm.

7.2.2. Radius Algorithm

The radius algorithm is mainly related with the distance between the tracked points i.e., it deals with points that are within a specified radius from the device. This algorithm keeps track of the last location data reported to the database server and the distance between the last and the current location. Only if the value of the calculated distance is more than a certain pre-defined distance (*Minimum_Radius* in meters), the device sends the current point to the server. Along with the *radius* variable it also involves the *fixed time interval*, using which it checks the frequency to send the data. The radius algorithm is similar to the fixed interval algorithm except for the distance calculation part. The outline of the radius algorithm is illustrated as a pseudo code in Figure 7.3.

```
SET Minimum_Radius = xxx (in meters);
IF Data_New == true THEN
   Retrieve previous reported data, Data_Reported, from
   memory;
   Radius = Calculate_Distance (Data_Reported, Data_New);
   IF Radius > Minimum_Radius THEN
      Report Data_New to server;
      Store Data_New to memory;
   ELSE
      Discard Data_New;
   END IF
ELSE
   Keep listening to location data from GPS receiver;
END IF
```

Figure 7.3 Pseudo code of the distance checking algorithm.

7.2.3. Direction Algorithm

The direction algorithm keeps track of the change in direction (i.e., angle) of the mobile asset so that a set of location data with similar angle will not be reported i.e., it keeps track of the last sent point and the current point and the angle between the new points. Along with the *degree* variable it also involves the *fixed time interval*, using which it checks the frequency to send the data to the database server. The direction algorithm is similar to the fixed interval algorithm except for the degree change instead of the time change part and the calculation of *Azimuth*. To calculate the azimuth, we use the method: *javax.microedition.location.Coordinates.azimuthTo()*. The azimuth is relative to true north. The current azimuth is calculated based on two continuous location data i.e., the new point and the previous reported point. The mobile asset retrieves the previously stored azimuth and then calculates the difference in angle between the two azimuths. If the absolute value of the moving angle is larger than the predefined *Minimum_Angle* (in degrees), the device sends the data to the server, instead of just calculating the difference between $Data_{Reported}$ and $Data_{New}$ as in the previous two algorithms (see Figure 7.4).

7.2.4. Speed Algorithm

The speed algorithm handles speed changes in the mobile asset, wherein any change in speed will result in change in frequency of the data sent to the database server. The following variables are involved in the speed algorithm.

1) *Speed*: This variable keeps track of the moving speed of the mobile asset.
2) *Current Interval*: This variable keeps track of the frequency to check new points and forwards it to the other components of the algorithm.
3) *Low Interval*: It is the lowest frequency to check and send new points to the server.
4) *High Interval*: It is the highest frequency to check and send new points to the server.

The current speed is compared with the predefined speed threshold and if the current speed is higher than the threshold, the high speed counter is increased and the low speed counter is reset, and vice versa. If any counter reaches the counter threshold, we

```
SET Minimum_Angle = xxx (in degrees);
IF Data_New == true THEN
   Retrieve previous Data_Reported from memory;
   Retrieve previous Azimuth_Reported from memory;
   Azimuth_New = Calculate_Azimuth (Data_Reported,
   Data_New);
   Angle = Absolute_Value (Azimuth_New –
   Azimuth_Reported);
   IF Angle > Minimum_Angle THEN
      Report Data_New to server;
      Store Data_New to memory;
   ELSE
      Discard Data_New;
   END IF
ELSE
   Keep listening to location data from GPS receiver;
END IF
```

Figure 7.4 Pseudo code of the direction checking algorithm.

dynamically adjust the data update interval (i.e., *HighSpeed_UpdateInterval* or *LowSpeed_UpdateInterval*). For example, assume that an asset is moving at the speed of 60km/h and sending data every 60sec. When the speed changes to 80km/h, the algorithm waits for the next five points before it changes the frequency to 30sec. On the other hand if the speed changes to 40km/h, the frequency changes to 90sec immediately. Thus slower speeds corresponded to less data transmission and higher speeds require more data to be transmitted to maintain the same tracking precision. After the adjustment, the mobile asset follows this new update interval to update its location as further illustrated below. The outline of the radius algorithm is illustrated as a pseudo code in Figure 7.5.

```
SET Threshold_Speed = xxx and Threshold_Counter = xxx (in km/h);
SET HighSpeed_UpdateInterval = xxx and LowSpeed_UpdateInterval = xxx (in sec);
SET Counter_High = 0 and Counter_Low = 0;
IF  Getting Data_New == true  THEN
    Retrieve previous reported data, Data_Reported , from memory;
    Retrieve previous reported speed, Speed_Reported , from memory;
    Speed_New = Getting_Speed (Data_New);
    IF  Speed_New >= Threshold_Speed  THEN
        Counter_High ++;        RESET Counter_Low;
        IF  Counter_High >= Threshold_Counter  THEN
            Report Data_New to server and Store Data_New to memory;
            EXECUTE Data_Update (HighSpeed_UpdateInterval);
            RESET Counter_High;
        END IF
    ELSE IF  Speed_New < Threshold_Speed  THEN
        Counter_Low ++;         RESET Counter_High;
        IF  Counter_Low >= Threshold_Counter  THEN
            Report Data_New to server and Store Data_New to memory;
            EXECUTE Data_Updatie (LowSpeed_UpdateInterval);
            RESET Counter_Low;
        END IF
    END IF
END IF
PROCEDURE Data_Update (Interval)
    SET Timer = 0;
    WHILE  Timer < Interval
      Timer ++;  Stop listening to location data from GPS receiver;
    END WHILE
    Listening to location data from GPS receiver;
END PROCEDURE
```

Figure 7.5 Pseudo code of the speed checking algorithm.

7.2.5. Waterfall Strategy

While the four previously discussed algorithms can be manually selected by the user of the asset, this process can be costly and inefficient over wireless data transmission systems where cost is based on the rate of data being sent. We further refined these geo-tracking algorithms to be dynamically selected based on a Waterfall strategy (see Figure 7.6). According to this strategy, the algorithms are selected sequentially based on the outcome of the preceding algorithm. The distance or radius algorithm is selected first and depending upon its true or false conditions the subsequent algorithms (the direction algorithm followed by the speed algorithm) are selected i.e., if the preceding algorithm returns false. Hence data transmission is based on a particular algorithm being true, if not it continues until it is true, otherwise the data is rejected. The reporting will be temporarily cancelled if the mobile asset stops moving. Many scenarios which might have redundant data were taken into consideration as a part of a Knowledge Base, such as traffic lights, stop signs, heavy traffic, highway driving, etc. The location data is sent on an as needed basis combined with the mapping pattern. Based on this pattern, the mobile asset could pre-decide the routing and can aggregate several location data together. Thus, the data transmission rate is considerably reduced by optimizing the tracking algorithms by employing the Waterfall strategy.

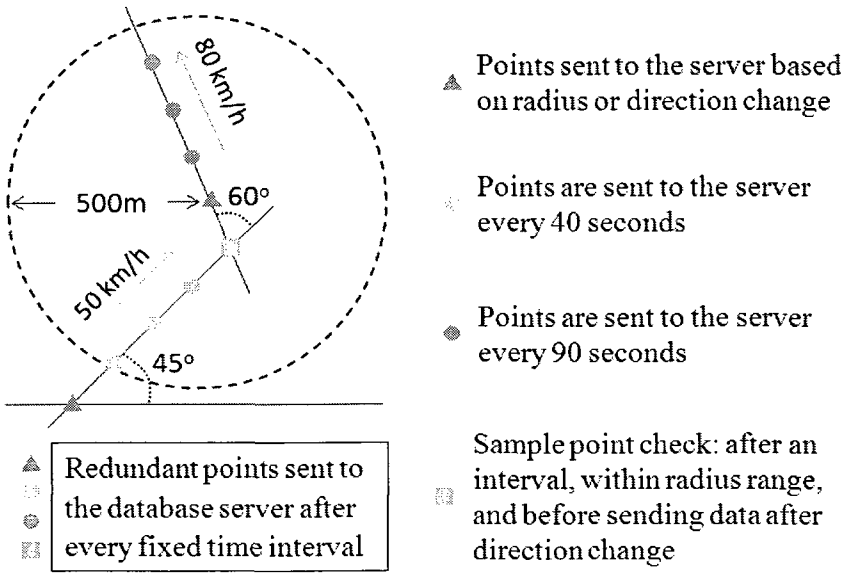


Figure 7.6 Summary of the algorithm involved in the Waterfall Strategy.

7.3 Context Monitoring Scheme

7.3.1. Overview

Context monitoring is performed to track (mobile) context entities and manage their dynamic context information in an efficient fashion [28]. It is an important step that aids the CSON-D protocol, as the specifics of the context disseminated depend upon the availability of the transformed context information. As the monitored context information's volume and frequency are directly proportional to the mobile entity's (especially low power mobile devices) resource usage, the common fixed data flow based technique for forwarding context information is not efficient. This is because, at any particular instant, the 'fixed interval' technique may utilize more or less resources than it actually requires as it is static in nature. Randomly reducing the monitoring frequency to reduce cost and resource usage could be serious for some mission critical applications. We studied specific constraints with regard to employing an always active context monitor in resource deprived mobile entities [131]. We found that commercially available systems are generally optimized for a single application, such as either sending text based location information or digital images, instead of efficient hybrid transmission with switching capability. Also, current content monitoring schemes cannot precisely identify and monitor the travelling contexts in scenarios where in multiple instances (for e.g., routing paths) are feasible. Therefore, a novel balanced technique is required to transform a sequence of monitored context information (with inherent errors) into smooth and reliable estimates of the dynamic state of the context, to reconcile resource usage with a mobile entity's resource availability.

7.3.2. Specifics

With reference to the CDP extension templates (refer section 4.1), the service framework extension template creates an instance of the dissemination service for every RSN to take care of CSN associations. The performance indicator is responsible to keep track of the properties of the identified context entities via this monitoring scheme. The trust

handler is used to customize the monitored context entities (both CSN and RSN) so as the fitness level of the contexts and the (decoded) agreements between contexts and the CDP are taken into consideration. By including context monitoring as a part of the CDP, we handle large volume of redundant context information (wherein, various sources have their own characteristics) to improve valuable resource utilization in mobile contexts. We designed and implemented an adaptive learning based context monitoring scheme that makes an optimized judgment of context dissemination to reduce superfluous retransmissions and increase monitoring precision by making sure that the highly relevant context information are considered first. We allow the target user or application to optimize the value of reporting interval and other variables (via the decoder). This dynamic adjusting scheme, with regard to the mobile asset tracking application, alters the reporting mechanism and the transmitted size of the context information to generate adaptable data transmit patterns based on the current environmental conditions i.e., contextual properties.

7.3.3. Approach

Our context monitoring scheme is illustrated in Figure 7.7. We reduce overhead and increase user-transparency with geographically referenced predefined context characteristics based on the requirements of the target user i.e., monitor and send only the important and understandable context information and discard the rest. To reduce the monitored context's signalling overhead in the monitoring scheme due to frequent updates, by optimizing the data bandwidth used and the communication cost, we use *mobile agents* to represent CSN and RSN. The context coordinates are tracked via the agent lookup service and the data transport takes place via existing communication protocols, or else scalability constraints would occur. It infers / recognizes matching patterns to adaptively monitor and manage the reported context information by analyzing the information using various techniques (see sections 7.4 and 7.5 for context monitoring scheme optimizations and extensions). The context monitor maintains non-redundant data in the CDP_KB. To effectively handle the wide context KB, manageable rule-based policy profiles [14], [132] are made to be a part of the CDP_KB and would be adapted to suit the current conditions. Each policy profile represents the threshold values of a mobile context entity and its unique contextual

properties. By customizing the rule-based policy profiles based on the actual mobile entity's learned contexts (by studying the history of the stored context points and paths in the CDP KB), we generate new policy profiles (in the receive-module). By cooperating with the CDP, we enforce or terminate the adapted policy profiles (in the send-module) with optional actions such as: *actionStartTime*, *actionEndTime*, *actionExpectedDuration*, etc. The adapted results of the context monitoring scheme will not only be utilized by the CDP-L1 components, but also in the overlay space. The results will be sequentially stored as (new or updated) policy profiles in the CDP_KB. Also, the send- and receive- modules for context monitoring coexist to provide caching functionality and synchronize locally discovered contexts.

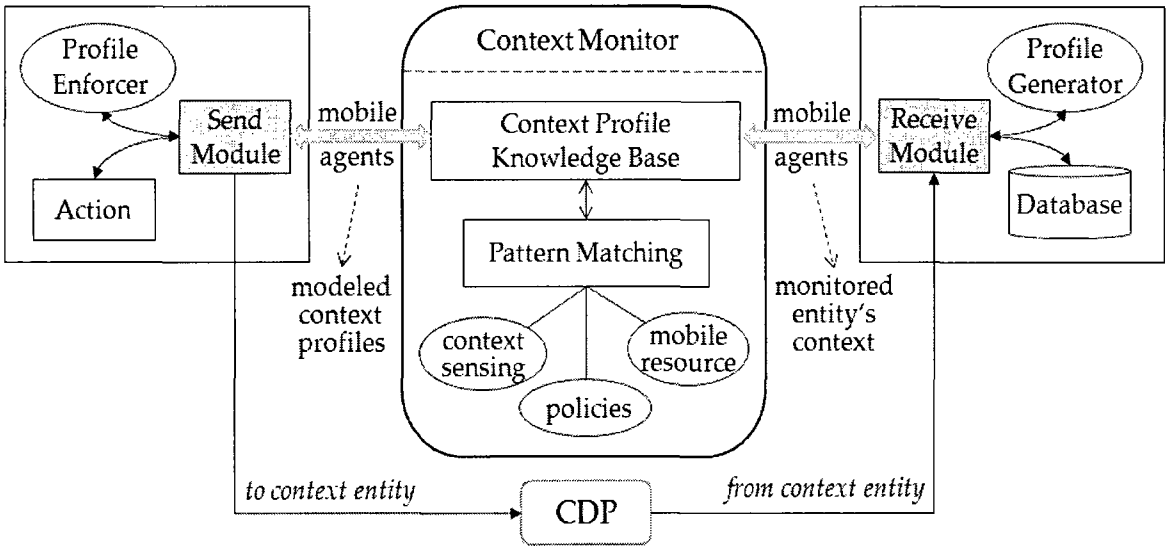


Figure 7.7 Context Monitoring Architecture.

7.4 Dynamic and Fault-Tolerant Adaptation

The collected data is reported in its raw form (with inherent errors) to the tracking server which acts as a middleware/ gateway between the mobile client and the application server. The tracking server initially utilizes terminal mobility management from a network perspective and direction-sense based knowledge to adaptively track and manage the reported GPS data. Terminal mobility support in a wireless network requires protocols for

location management and handoff management, and herein, both location and handoff management are utilized. The tracking algorithms are further enhanced to add more features which allow the user to optimize (i.e., adapt) the value of reporting interval and other variables. After the tracking algorithms were set up, the analysis of the reported geo-data is done to understand the behaviour of typical traffic generated in various demographics and tolerate faults. The reported data is researched using various techniques, such as:

1) Buffering points,
2) Sending points in bursts,
3) Collecting points while out of coverage, and
4) Sending and collecting contexts in various path scenarios (curve fitting techniques of data reporting to real representation of routes).

There are different types of driving path scenarios considered, including:

1) *Busy Paths*: Paths where the traffic is very busy and moving slowly. They test the application's data transmission rates.
2) *Easy Paths*: Paths which are light and move fast. They test the application scenarios wherein the device is moving in a predefined way.
3) *Long Paths*: Paths which are more than 200 km in length, and can be either busy or easy. Test scenarios include no frequent turns and devices moving at high speeds.
4) *Multiple Turn Paths*: Paths with frequent changes in both latitude and longitude due to frequent direction change scenarios.

A dynamic adjusting algorithm, to alter the reporting mechanism and the transmitted size of the location data, to generate adaptable data transmit patterns is thus formulated. This dynamic mechanism generates a route fitting track instead of straight lines connecting location data instances by using the Waterfall strategy and the current conditions (i.e., environmental awareness). We tend towards environmental aware contexts to gain adaptability and prevent highly relevant data from not being transmitted, as the context information is used to characterize the situation or operational state of the asset [74]. This could include contexts such as users, devices, networks, etc, and herein, the context-based data of the mobile asset could be the latitude of its current position, the IP address of its

active network connection, its (or user's) preferences, or its available resources. Similar to the policy profiles usage, to handle the wide context KB, manageable context profiles are made to be a part of the KB and would be adapted to suit the current conditions. A context profile represents the algorithm threshold values of an asset and the unique contextual properties of a route. A profile manager is designed for the overall management of the location context based profiles. The profile manager is aided by policies to customize location contexts based on rule-based policy tags, the actual mobile asset's resources, and context sensors, in order to adapt the operations based on actual routes and enhance the learned routes. The profile manager has access to the policy repository as well as the location database server. By applying the above discussed features to the Waterfall strategy, we dynamically adjust the Waterfall strategy parameters to adapt routing and overcome faults. This adaptive routing process is discussed as an algorithm in Figure 7.8 (and illustrated along with routing learning in Figure 7.9), wherein the adapted changes, if any, are instantly emailed back to the mobile device, which in turn updates its persistent memory.

7.5 Personalized Learning Engine

The application of novel computational intelligence methods is a potential solution for intelligent route learning, but one requires an expert that can collect data in cities, villages, and suburbs at different shapes of routes such as curves, u-turns, zigzag, etc. Hence, as a solution to the problem of adaptive learning with respect to routing a mobile asset, we restrict our research to the use of environmental contexts, and learn and utilize:

1) predefined routes within the target route,
2) the sequences of locations with desired speed,
3) the impact of the road shape, and
4) the surrounding area's density.

Knowledge Base

- Database history
- (Sample) Profile schemes
  - Action (Increase/Decrease a single or a set of monitoring conditions)
  - Action with actionStartTime, actionEndTime, or actionExpectedDuration conditions
  - Action with a combination of conditions (also denotes geo-data intervals)
  - Contextual constraints: Device-specific, Time-specific, and Location-specific

Identify, Verify,* and *Filter

- CDP-L1 functionalities (with monitoring conditions)
- Check for relevancy with access to CDP_KB

Study Monitored Contexts

- Analyze the geo-data to overcome faults
- Maintain FIFO sequence; Compare with (multiple) existing contexts
  - Check for redundancy; Establish Entity-Relationships (ER) between contexts
- If no (sufficient) previous data, skip studying

Profile Scheme Matching

- Analyze ER with a set of policies
  - Policies: Multiple *rules* amid variable *if* conditions with *negotiation* parameters
- Select a semantically matching profile scheme based on the result of the analysis
- Select combinations within the profile scheme based on the analysis, if applicable

Create New Profile* and *Other Operations

- New monitoring conditions, if any
- Verify relationship associations in the new profile with access to CDP_KB
- Forward the output profile to the appropriate (mobile) context entity to achieve dynamic adaptive routing by adjusting the Waterfall strategy parameters

Figure 7.8 Adaptive routing to aid mobile asset tracking.

We use pattern recognition and adaptable policies to make the tracking algorithms more intelligent with these geo-referenced routes and route characteristics in order to turn regularly tracked routes into "optimally" tracked routes on the fly. Herein, optimal is defined with respect to the above listed environmental contexts. The policy tags, managed by the policy manager, adapt to changing communication conditions with respect to the mobile asset's geo-data (i.e., the actual routes instead of learned routes), and it generally consists of a triggering event, a set of conditions, a set of actions, and a life time. In addition, to reduce the location update signalling overhead by optimizing the data bandwidth used and the communication cost, and to build a user-transparent handover mechanism between the mobile device and server software, we use intelligent mobile agents. We try to provide perfect and complete data about a system and introduce their own system dynamics and distortions as well.

The feature set for route learning and adaptation is termed as the Learning Engine (LE), and is divided into server-side LE module and client-side LE module (see Figure 7.9). The mobile device software (i.e., the client-side) with an intelligent LE would provide data processing and optimize the reporting interval for various individual personalized services to dynamically adjust the data transmit patterns and preserve resources. The server-side software manages network communication to improve battery life and maintain non-redundant database. For example, synchronizes field employees' work pattern with the back end. The intelligent mobile agent model learns how to adapt its behaviour according to behaviours of other agents it is interacting with (i.e., the asset's/user's preferences are important) by using explicit communication and sends a client agent itself to the server, which executes its task in the server and brings back the results. The agent model (denoted as the service creation framework, SCF, in Figure 7.9) views every active routing and route tracking as a service (denoted as service nodes, SN). With the help of the policy tags, the 'similar' service nodes are grouped together as hierarchical clusters in order to customize the routing process. The server LE module co-operates with the profile generator sub-component of the profile manager, wherein it studies the history of the stored points and paths, and adapts the route taken by the corresponding mobile asset in the future. The profile generator makes appropriate decisions to create, adapt, and terminate personalized

context profiles. While the client LE module co-operates with the profile enforcer sub-component, which controls client-side resources and enforces configuration changes based on the dissemination of the generated context profiles. The results of the LE will be sequentially stored as profiles in the KB, resulting in service node activation in the SCF. The logic-based KB accesses the MDC [133] infrastructure and resources and aids in service provisioning. The client and server LE modules coexist with the (context-based) profile manager and act as 'ports' that provide caching functionality and mediate (or alternately, synchronize) locally discovered information. A newly adapted profile based on the LE will thus provide an appropriate customized route tracking path that meets the requirements of a service (i.e., the mobile asset's day-to-day activities), in order to make realize realistic route tracking.

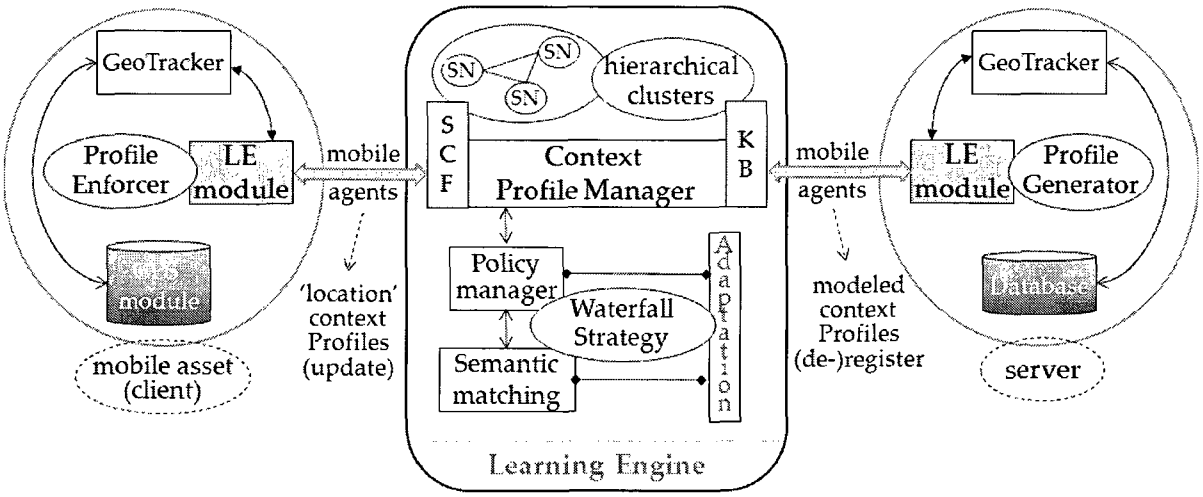


Figure 7.9 Personalized learning engine to aid the context dissemination infrastructure in mobile asset tracking.

7.6 CSON-D Protocol and Real-time Application: Flow Structure and Comparison

With reference to Figure 7.10 callouts, the schematic summary of the main algorithmic steps of the entire CSON-D protocol (discussed in Chapters 4, 5, and 6) are listed in sequence as follows: (1) overlay space initialization, (2) underlying context registration,

identification, and verification, (3) context client's profile decoding, (4) C_S semantic classification and association as either C_X (represents the leader nodes i.e., the main context type that decouples the overlay) or $C_{X,Y}$ (represents the sub-nodes), (5) C_R classification and association, (6) uniformly distributed decoupling, (7) bottom-level intra-hierarchical CSON formation, (8) inter-hierarchical association and sub-node search, and (9) top-level inter-hierarchical RSON formation with minimized unicasting. The figure shows the context flow structure. As a common step, all events are updated in the database listener which persuades data-centric storage.

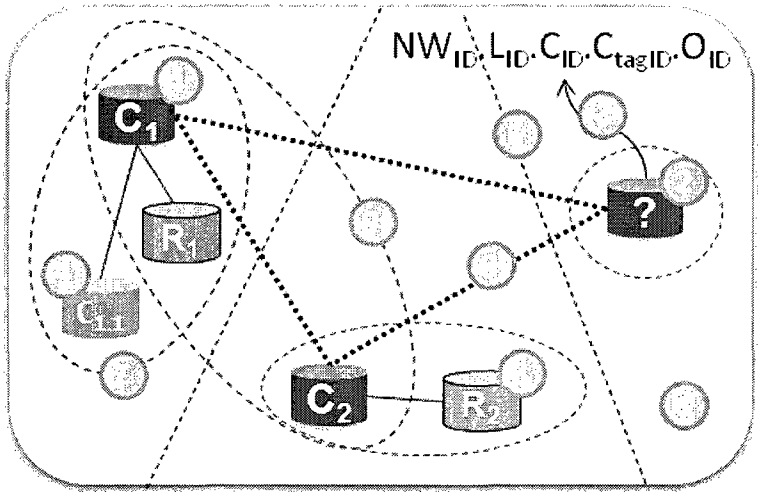


Figure 7.10 Schematic summary of the context dissemination protocol and context flow structure.

The mobile asset tracking application's flow structure is schematically summarized in Figure 7.11. The figure shows the MSC view of the 'practical' usage of the client and server modules within the GeoTracker application prototype. Herein, the prototype events that are mainly considered include: (i) mobile device and database activities, (ii) waterfall strategy with geo-tracking algorithms (i.e., location updates), (iii) adaptation optimization with policy manager, (iv) client LE module, (v) server LE module, (vi) KB, (vii) SCF, and (viii) profile manager. The time delays of these events are illustrated later in Figure 7.14. The various events of every component's internal operations could be optional based on the scenario involved. In our case, we assume a user is moving from one point to another via

multiple via points as a part of his/her daily routine. The 'via' points might change but the starting and ending points are always the same.

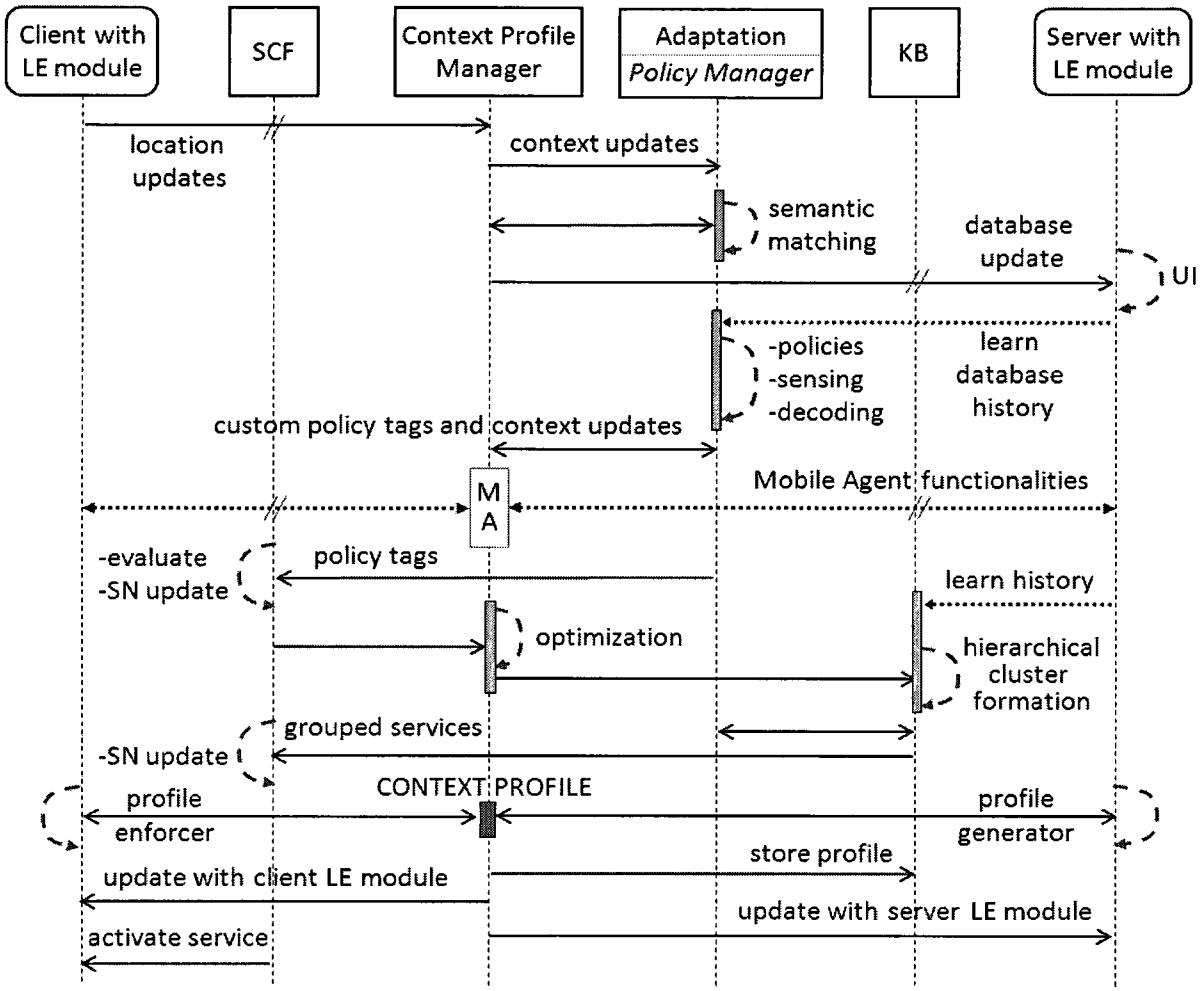


Figure 7.11 The context flow structure of the mobile asset tracking application prototype representing various events

Even though a direct comparison of the CSON-D protocol and the real-time application is not possible due to the lack of an AN infrastructure in the real-time application, we perform a brief prototype comparison of the components and steps involved in the two as follows:

- Context profile manager is based on CDP – L1
- The context monitoring scheme, adaptive routing, LE modules, and the policy manager are based on and suit CDP – L2
- SCF is based on overlay space
- SN represented as MAs representing users/devices denote C_S and C_R
- Semantic matching and hierarchical clusters are based on the multi-level overlay scheme
- KB is based on data repository and the routing table
- Profile generator is based on the lookup service using the agent manager
- Profile enforcer is based on RSON creation and the casting scheme

7.7 Performance Evaluation

7.7.1 Implementation Architecture

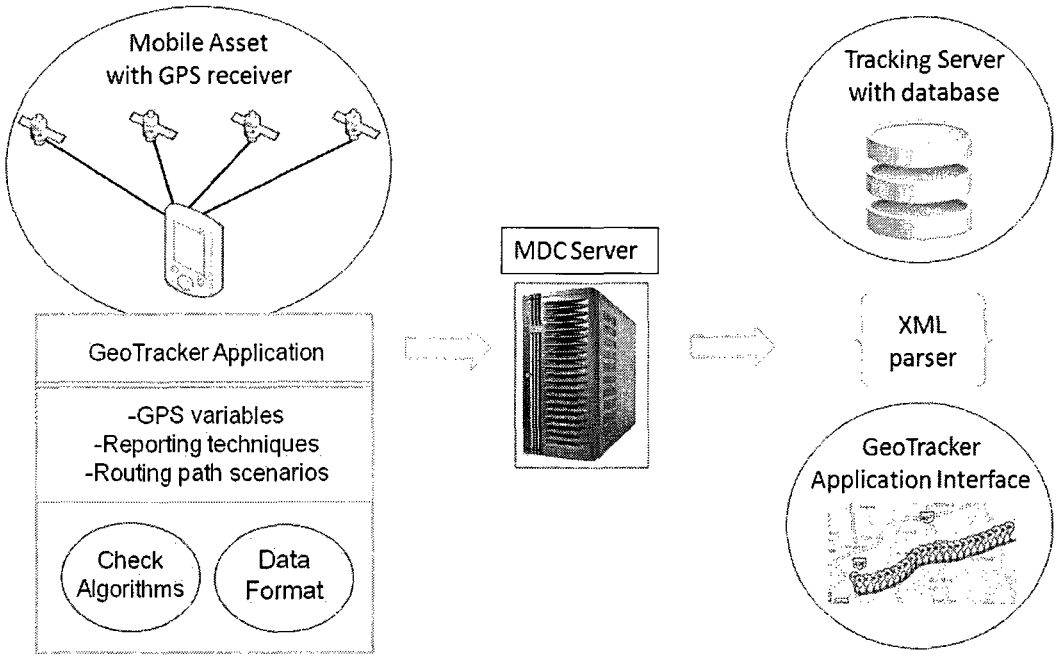


Figure 7.12 An overview of the implementation architecture.

Once the design document of the context-based Learning Engine was created to run on the target mobile asset, an application protocol to monitor and disseminate contexts was implemented. Along with this experimental learning mechanism implementation, a server-side intuitive user interface (UI) provides an option to customize and view specific asset's detailed location and path information. The implementation architecture of the context dissemination scheme based mobile asset tracking application is outlined below in Figure 7.12. The middleware represented by Vayyoo's Mobile Data Convergence (MDC) platform [133] is for client-server integration (see Appendix A).

7.7.2 Infrastructure

Our target mobile asset is a Blackberry mobile device, and we use a BlackBerry JDE dependent simulator to represent GPS-enabled mobile devices which support cross platform dynamic data overlapping. The simulator is customized to integrate with the tracking algorithms, the predefined routes, and the geo-location transmissions to produce the tracking parameters and the measures of performance. This is performed so that the geo-data updated in the database will be 'complete' points with not only the coordinate information but also the deployed algorithm information. The simulator however has its own discrepancies, such as (i) Automatic changing of inputted points, (ii) Automatic addition of extra points, and (iii) Unit measurements conflicts. To avoid these discrepancies, we developed another independent simulator tool, the input of which is a text file with routing information in a specific format containing the latitude and longitude coordinate values. The independent simulator can be deployed as an individual Java object without the BlackBerry device simulator i.e., as a light-weight simulator for data collection and processing with ease. The collected data is then analyzed to observe the reporting behavior pattern based on vital characteristic changes. Both the simulators: (i) Input speed randomly (the minimum and maximum speeds can be customized depending upon the tracking algorithm being tested), (ii) Input date and time from the mobile asset's properties, and (iii) Input all other variables such as esnID and device-type either as constants or from the mobile asset.

The database (see Table 7.1) wherein the reported data is stored is setup using SQL Server 2005 for representing an optimized KB. We establish JDBC type4 communication between the database and server-side UI. XML tags are used in a location document to display location information (both real-time and simulated) in a map- and location- client application. We employ Java ME to request/get location information on a BlackBerry. Some of the package features like coordinates provisioning, timestamp, accuracy details, etc are mandatory to avoid fragmentation and implementation conflicts. By using the javax.microedition.location package, a BlackBerry Java application can display the current position of a GPS enabled Blackberry. With regard to GPS modes, we prefer the assisted mode as it provides location information faster than the autonomous mode and is more accurate than the cell site mode. The GoogleMaps JSP TagLibrary [134] is utilized as a part of the context monitoring scheme.

Table 7.1 Database columns.

esnID	longitude	latitude	device_type
device_date	device_time	heading (direction)	Speed
Algorithm	Interval	low_interval	high_interval
Radius	Angle	s_speed	uniID

7.7.3 Results

The graph illustrated in Figure 7.13 compares the context monitoring results of the default fixed interval scheme with the proposed monitoring scheme. The number of context information points denotes the amount of overhead in sending and receiving context information to the server in both these schemes. The common fixed interval scheme handles a large number of contexts when compared with our monitoring scheme. Each test number denotes a separate test path with sub paths to test various path scenarios. The four paths that are tested are denoted as *R1* for a local path, *R2* for a city path with several activities, *R3* for a dense highway path, and *R4* for long paths. As illustrated, between the tested schemes, the difference is minimal in short local paths, while maximal in long highway paths.

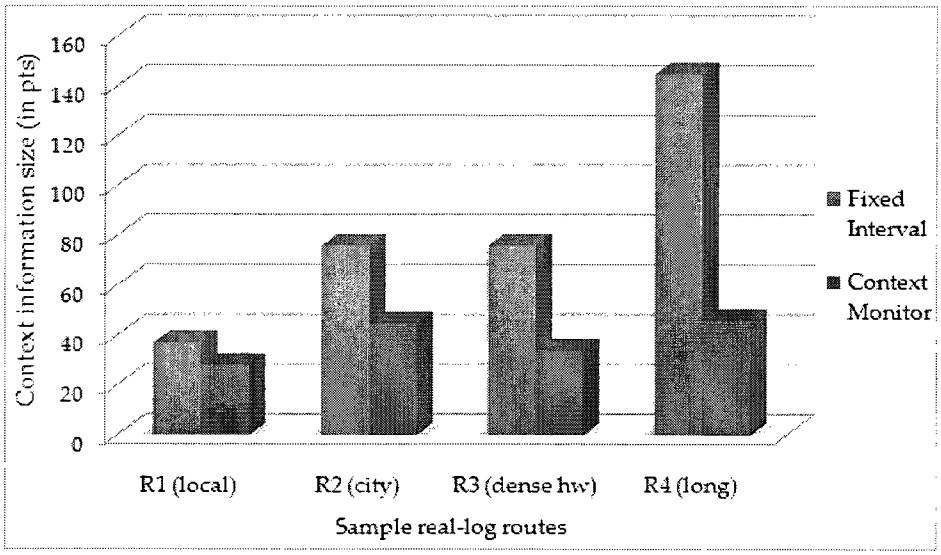


Figure 7.13 Comparison of the default and the proposed context monitoring schemes in various scenarios.

As illustrated in Figure 7.14, the proposed optimization's minimal time consumption is traded off to achieve successful tracking and accurate routing. As the mobile device and the PCs used for simulation have high processing power, the implementation architecture consumes less time to execute compared to the time consumed for data transmission. The prototype events that are mainly considered include: (i) Mobile device and database activities, (ii) Waterfall strategy with geo-tracking algorithms, (iii) Adaptation optimization, (iv) Client LE module, (v) Server LE module, (vi) KB and SCF, and (vii) Profile/ Policy managers.

Once the GeoTracker application is launched in the mobile asset, it displays the information shown with real GPS data. The application software's settings can be configured in real-time and the asset will keep updating its location information until stopped. The current location, saved into persistent storage, is compared with the previous location information. If changes are deemed necessary, the location data is sent to the server in a specific XML format (see Figure 7.15 for an example). The XML as HTTP POST is first LISTENED, then PARSED, and finally UPDATED in the database via the previously discussed database structure.

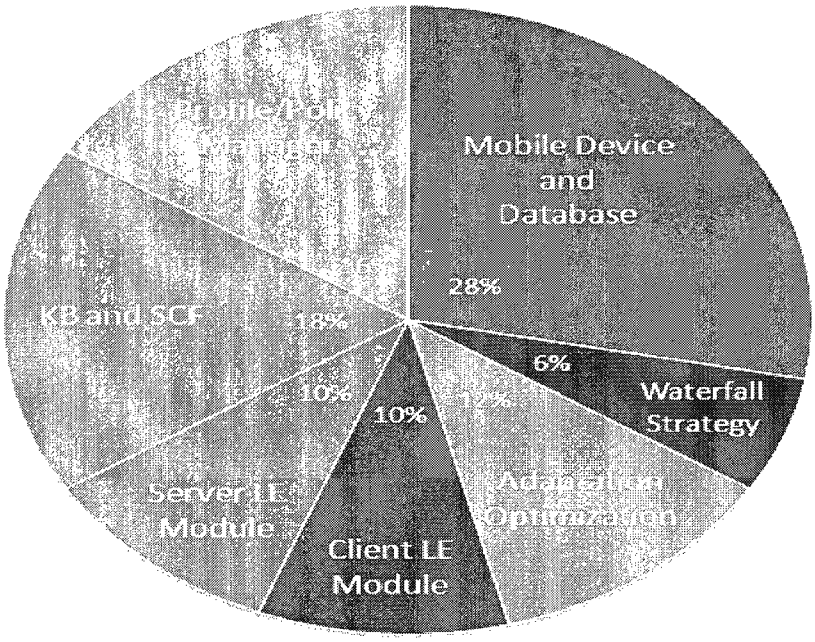


Figure 7.14 Practical time delay of the GeoTracker's main events.

```
<?xml version=\"1.0\" encoding=\"UTF-16\"?>
<positions version="1.0" time="2007-05-29 17:56:59" messageID="{73226407}">
<position esn="Anu" positionTime="2008-05-29 17:56:59" deviceType="BlackBerry">
<lat>45.352580555555555</lat> <long>-75.94021666666666</long>
<speed units="km/s">53</speed> <heading>West</heading>
<algorithm>Fixed</algorithm>
<interval units="mSec">30000</ interval>
<low_interval units="mSec">20000</low_interval>
<high_interval units="mSec">40000</high_interval>
<radius units="m">500</radius> <angle units="deg">265</angle>
<s_speed units="km/s">60</s_speed> <altitude units="m">1355</altitude>
</position></positions>
```

Figure 7.15 Sample XML file containing the geo-data and the adapted parameters.

The receiver that parses and updates the geo-data in the database could return an error, for example, if the received *esnID* is not already present in the database. Based on whether it succeeds or fails, the following text/xml response is sent back to the POST source. A sample response is shown in Figure 7.16.

```
<serviceResponse version="1.0" status="OK"/>
```

```
<serviceResponse version="1.0" status="Error">
        <exception>BAD REQUEST – NOT ACCEPTABLE</exception>
</serviceResponse>
```

Figure 7.16 Sample response from the XML parser.

A UI is designed to visualize the routes, the transmission locations and times, as well as the metrics and parameters involved with the tracking algorithms. In the UI's main screen the admin has options to *Register User* for application management, *Register Asset* to track, or *Login* (using java Authenticator) to the GeoTracker. In the user registration screen each UI user is assigned with one active *company*, while in the asset registration screen each asset is assigned to a device user and a company. After logging in, the UI user can select the mobile asset (i.e., the associated user in a company) to be tracked and the appropriate *date* parameters from a dynamically populated list. In our UI, every mobile asset can be tracked in 3 different ways (see Figure 7.17 for a simplified view without authentication status).

1) The last reported position.
2) A sequence (ordered by date and time) of previously reported positions separated by pre-defined time or distance. While it is currently set as 50 positions, it has no limit.
3) Time-based asset tracking with the use of *from* and *to* time parameters; positions are separated based on pre-defined time or distance.

A *GO* button and a *GET KML* button are provided for each of these tracking procedures. Once the GO button is clicked, the reported positions are displayed as markers in customizable Google Maps API. If multiple positions are reported, the path taken by the mobile asset is shown by connecting the tracked points using a polyline, as illustrated in

Figure 7.18. The points are centered and the width is based on the page width. The points that are in motion are illustrated with a green icon and the ending point with a red icon.

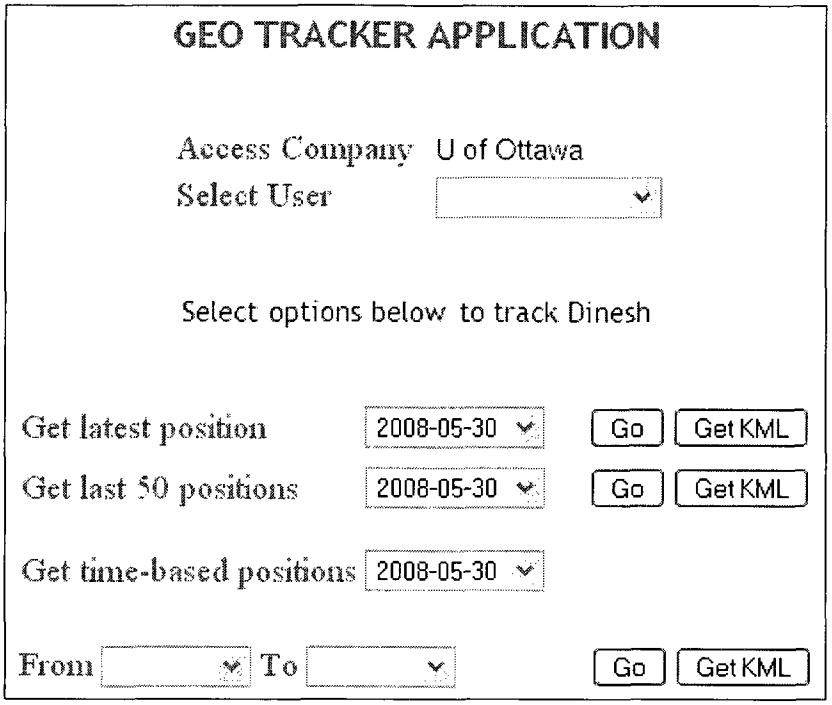


Figure 7.17 The GeoTracker UI's simplified view options screen for a user named 'Dinesh' from 'University of Ottawa'.

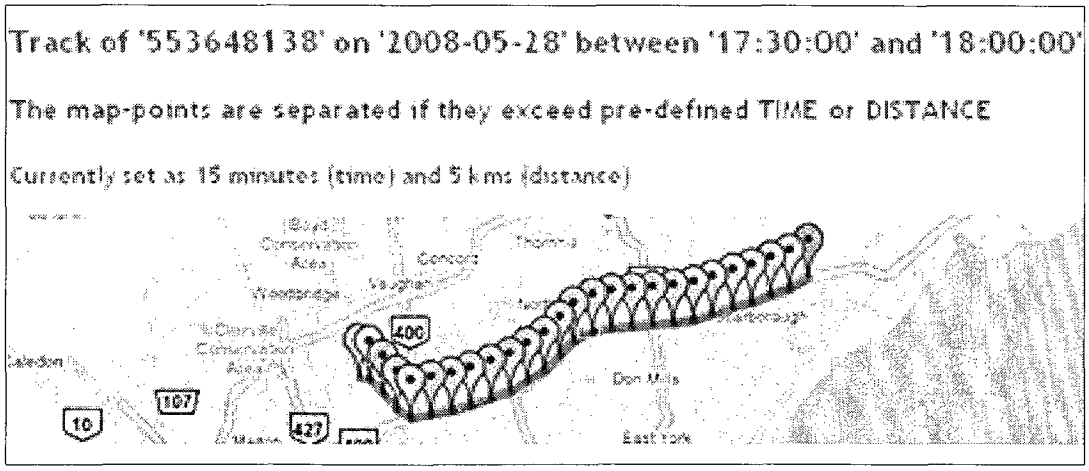


Figure 7.18 The result of time-based asset tracking with pre-defined constraints represented as points with a path.

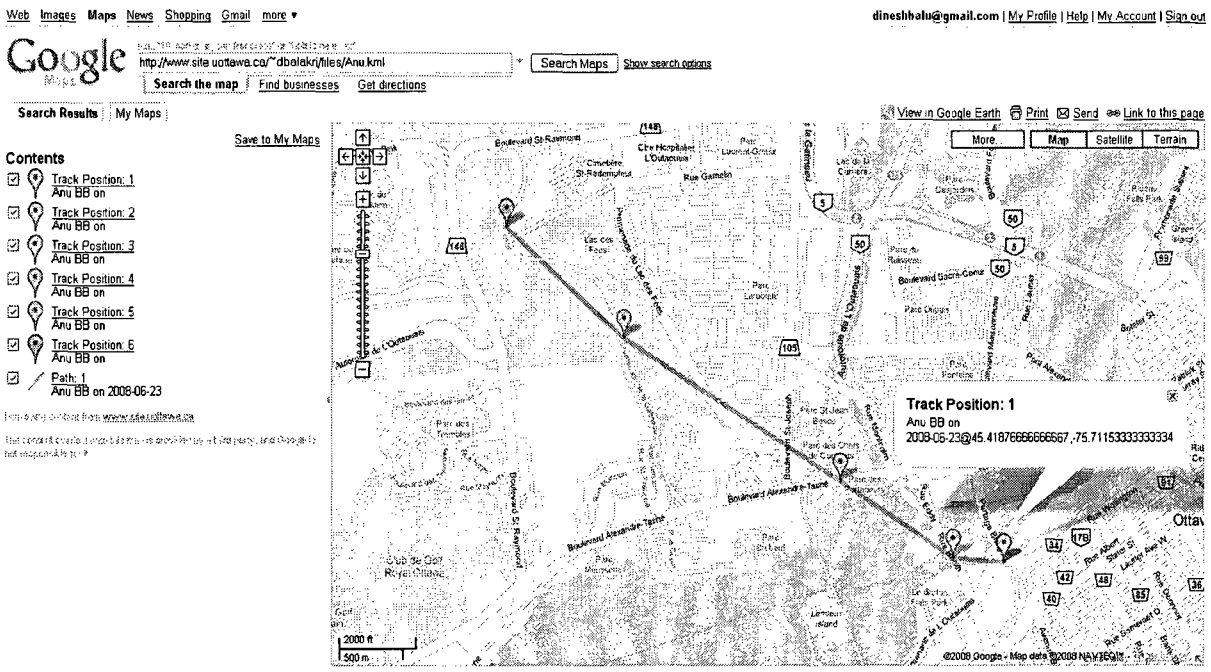


Figure 7.19 The tracked mobile asset's location and travelled path represented as a KML file in Google Maps.

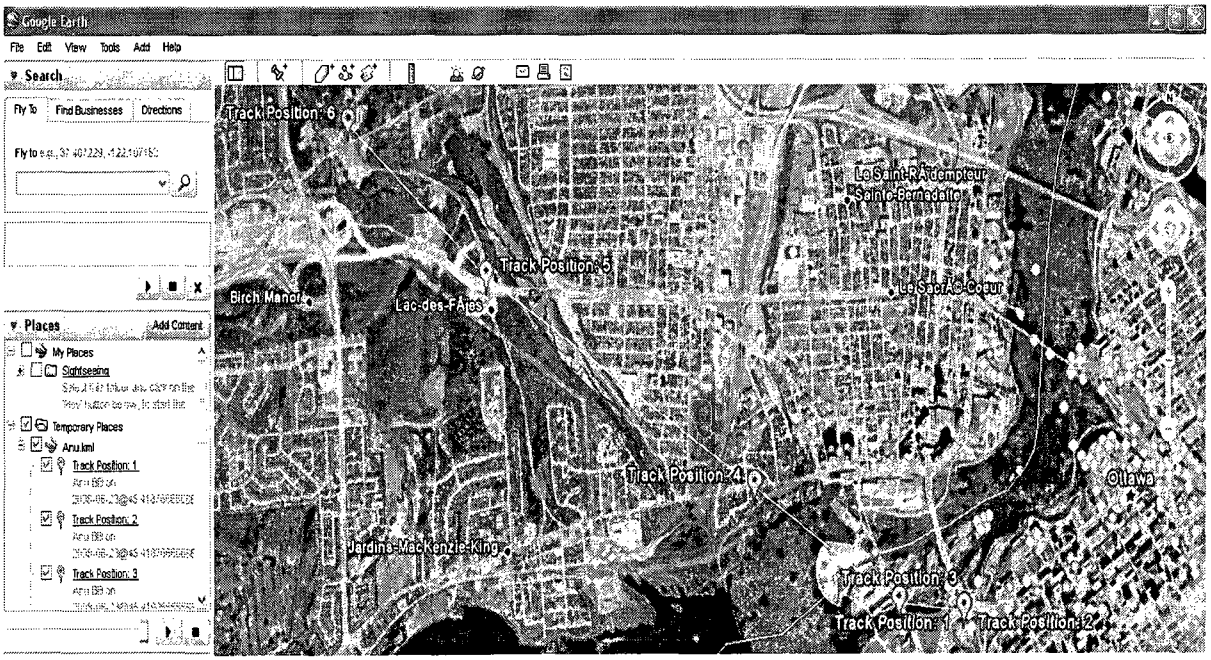


Figure 7.20 The tracked mobile asset's location and travelled path represented as a KML file in Google Earth.

Clicking on the GET KML button will generate a KML file [135] based on the selected company/asset/date/time parameters. The UI user can either open the file using the Google Earth [136] program or store it to be later opened using Google Maps in a web browser. The saved files need to be hosted and their web addresses are used in the *Search Maps* field to display the result. In KML, each point and polyline, presenting a *placemark*, can be customized and the icons are coloured appropriately. Figures 7.19 and 7.20 show the tracked route results (in KML format) in Google Maps and Google Earth, respectively.

As discussed in section 7.4, the adaptive algorithm, based on its profile-based processing, if changes are deemed necessary, generates new parameters, such as speed, and instantly sends them back to the mobile asset. Each of these algorithm parameters is associated with changing time intervals for data reporting. The algorithm is capable of handling more than one mobile asset at a time. The initial phase started with manually adjusting the threshold values of the adaptive algorithm. Later, depending upon the mobile asset's recent (for e.g., last 5 to 10) reported geo-data the threshold values are automatically generated based on profile updates, due to volatile road and traffic contexts. A tracked mobile asset's location and path information *before* and *after* route adaption optimizations are shown in Figures 7.21 (a) and 7.21 (b), respectively. The results are illustrated as a KML file in Google Earth, wherein the ending point is denoted with a red coloured icon. We use the same path with the same starting and ending points. We also illustrate the test results of the fixed interval, direction, radius, and speed –based tracking algorithms along with the Waterfall strategy in Table 7.2. The abbreviations in the table are Pts, R, D, FI, CI, LI, and HI and they denote Points, Radius, Direction, Fixed Interval, Current Interval, Low Interval, and High Interval, respectively. In the table, each *test number* denotes a separate test route with sub-test numbers denoting route variations to test various routing scenarios. The values in *total points* and *points sent* columns compare the data reported results of the fixed algorithm with the Waterfall strategy. The employed tracking algorithm indicated in the *algorithm* column could be radius, direction, or a mix of radius and direction, each with or without speed variations within the specified speed limit. Each of these algorithm *parameters* is associated with changing time intervals for data reporting. The test results

suggest that there has been approximately 50% average reduction of data transmitted over the air when compared with the fixed time interval algorithm.

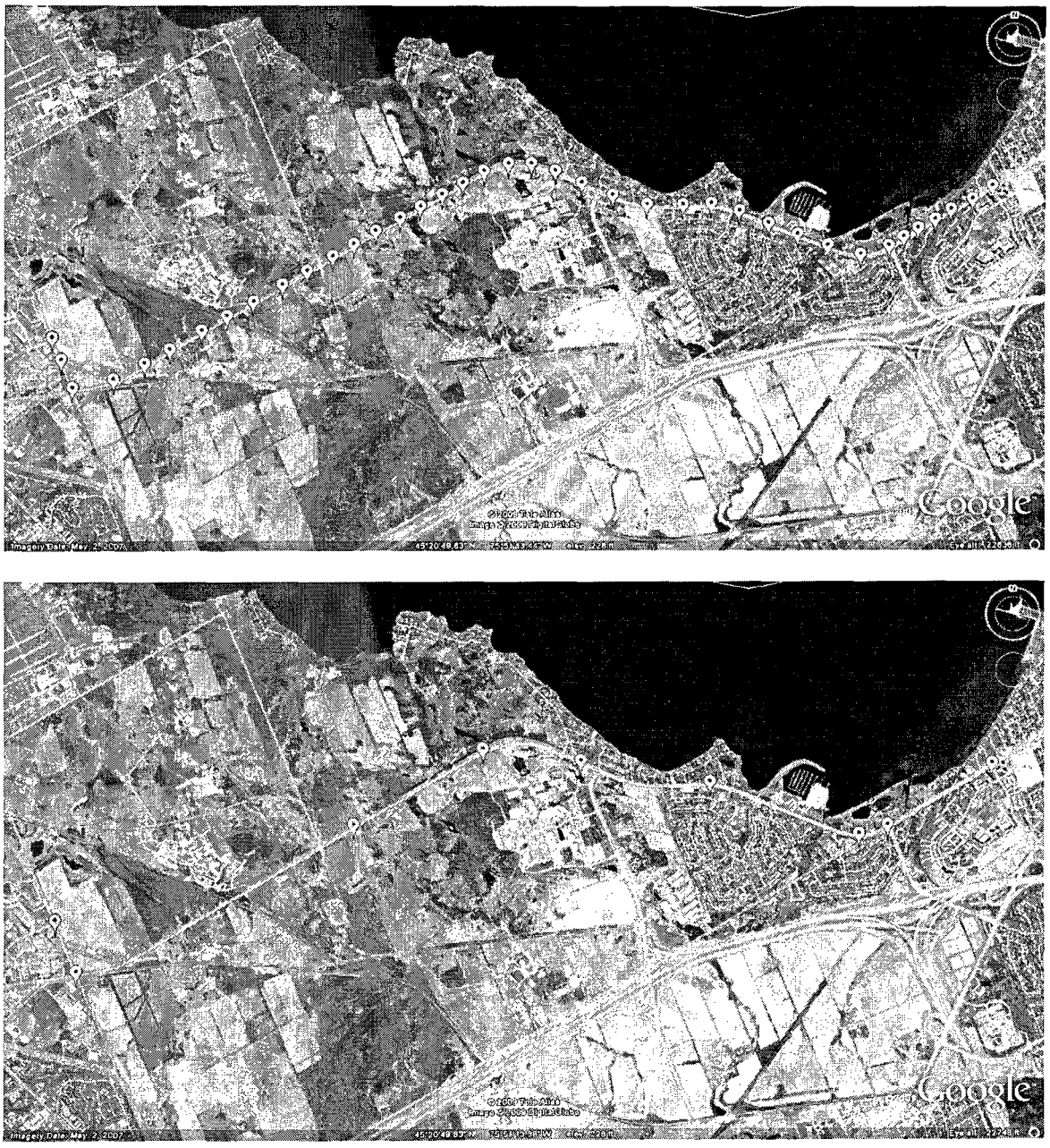


Figure 7.21 The tracked mobile asset's location and travelled path represented as a KML file in Google Earth (a) (top) 'before' optimizations, and (b) (bottom) 'after' route adaption optimizations.

Table 7.2 Test results of the adaptive tracking algorithms and the Waterfall strategy.

Test No.	Total Pts/Pts Sent	Algorithm	Parameters									Speed limit
			Radius		Direction		Speed				Fixed	
			Radius	FI	Angle	FI	CI	Speed	LI	HI	FI	
1.1	66/35	Radius	500	30	340	30	30	50	30	60	60	50-70
1.2	66/22	Mix of R and D	1000	60	320	60	30	70	30	60	60	50-70
1.3	66/18	Mix of R and D	1250	60	320	60	30	70	30	60	60	50-70
2.1	37/13	Direction	2000	60	317	60	30	70	30	60	60	70-90
2.2	37/8	Mix of R and D	2000	60	330	30	30	90	30	60	60	70-90
2.3	37/13	Mix of R and D	1250	30	330	30	30	90	30	60	60	70-90
3.1	201/147	Mix of R and D	2000	60	350	60	30	100	30	60	60	90-100
3.2	201/68	Mix of R and D	5000	60	250	60	30	100	30	60	60	90-100

The test results of the LE-based routing technique are illustrated in Figure 7.22. Based on the LE, enhanced route tests are done to analyze the Waterfall strategy too. The illustrated graph compares the total geo-location points (i.e., geo-data) sent to the server by a mobile asset while using fixed interval algorithm, Waterfall strategy, and LE for various routes. The three routes that are tested are denoted as *R1* for a busy route, *R2* for a dense route, and *R3* for a highway. As illustrated, the basic fixed interval scheme sends a large number of geo-location points when compared with the Waterfall strategy and LE-based routing technique. Between the Waterfall strategy and LE, the difference in the number of geo-location points sent is minimal in dense routes, while maximal in highways. The test results prove that the proposed system successfully and satisfactorily performs dynamic filtering of raw geo-data by intelligent route learning and adaptive route tracing.

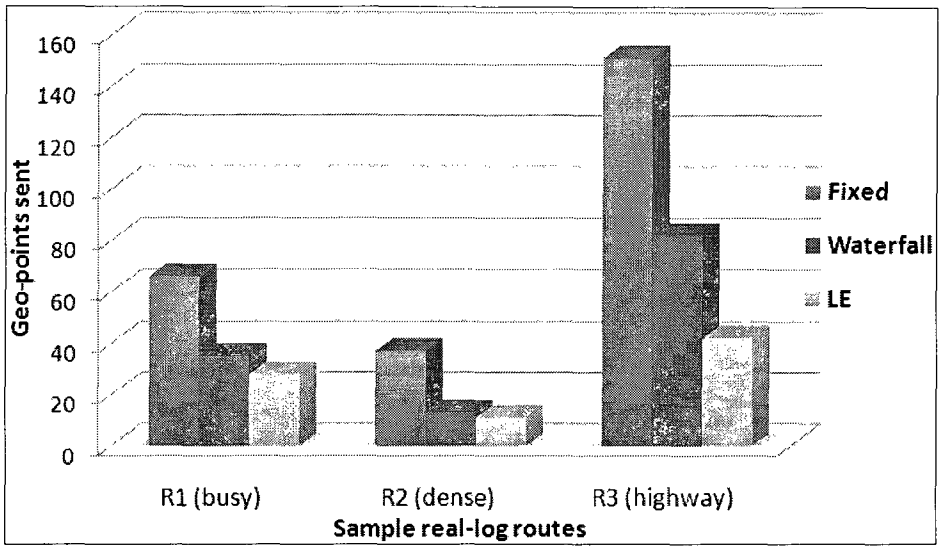


Figure 7.22 Test results of the LE-based routing technique for mobile assets.

7.8 Summary

This chapter first overviewed the role of context monitoring and context dissemination in a real-time application. It elaborated the geo-tracking based real-time application, which has its own algorithms and strategies, to track and mange mobile assets (i.e., contexts). The requirements, specifies, and the approach carried out to monitor these contexts were discussed. We then discuss approaches that were proposed in previous chapters and propose new approaches to aid the context monitoring scheme a part of the context dissemination infrastructure. The approaches mainly included techniques for dynamic adaptation, fault tolerance, and personalized learning. The chapter then evaluated the performance of the context dissemination infrastructure in detail with regard to the mobile asset geo-tracking application. The thesis is concluded in the next chapter by summarizing the main contributions of this thesis. Our suggested future research plans are also discussed.

CHAPTER 8
Conclusion

8.1 Contributions Summary

User-centric computing and communications (i.e., ubiquitousness) is becoming increasing popular and reliable, thanks to the rapid progress in wireless communication hardware and network standards. At the same time, one has to face several challenges to realize such ubiquitous applications in mobile, wireless, and dynamically composed networks which include, but not limited to, the realization of mobility, flexibility, scalability, adaptability, and personalization properties. The major problem in most of the current dissemination systems in ambient network infrastructures is the lack of environmental knowledge and adaptability to environmental changes. By discussing the dissemination scheme's constraints with respect to generic autonomous systems and related works, we determined that achieving context dissemination in a balanced and interoperable manner with the existing dissemination protocols in a heterogeneous environment is not practical. The networking environment executing the dissemination protocol demands ubiquitous connectivity and on-demand cooperation of heterogeneous networks. As discussed in the recently published book on ambient networks [137] and as discussed in this thesis, context-awareness and effective context information dissemination are essential features for user-centric computing and context-sensitive communications in ambient

networks. Context dissemination, an important non-functional requirement component to enable context awareness in heterogeneous environments, aids in ambient network service composition and decomposition, in both specialized and pure ambient networks. We derived requirements imposed by ambient networking on the dissemination approaches to be used, and evaluated the existing approaches with respect to these requirements. Various frameworks support various overlay topologies and schemes for dissemination, but most of these frameworks do not satisfy the demand for advanced and complex dissemination mechanisms for context-aware services in ambient networks.

The thesis presents a context dissemination architecture based on an overlay environment for autonomous systems in heterogeneous environments, especially ambient network's ContextWare. Our overlay based architecture can satisfy special context requirements to solve specific conflicts and be in-line with the dynamic evolution of the Internet, all without having to deploy new equipment or modify existing protocols, thus exhibiting flexibility and scalability. The dissemination scheme's optimization techniques make best use of the advantages of agent technology, overlay networks, and policy profiles, while solving associated problems to design an efficient context dissemination system as required by next generation services and applications in modern converged networks. These optimizations in turn helped us to demonstrate better performance towards a realistic multi-level pure overlay based context dissemination scheme that improves the overall performance by informed virtual client collaboration with each other. The outcome of these techniques corresponds with the features of ambient networking such as heterogeneity and self-management. The Context Dissemination Processor and Pure Overlay Space based CSON-D protocol achieves spontaneous dissemination of up-to-date contexts by constructing and maintaining a multi-level overlay infrastructure with context monitoring and minimized casting.

Our initial Pure Overlay Space approach was published in [30] and the optimization schemes were published in [31]--[32]. Our initial Context Dissemination Processor was published in [30] and the proposed optimization scheme was used in a related real-time application in [35]--[36]. The multi-level overlay structure with minimized casting

functionality was published in [33] and the optimization schemes to exhibit adaptive behaviour were published in [34]. The journal versions of these papers are currently under review.

8.1.1 Context Dissemination Processor

The Context Dissemination Processor controls the identification of all types of contexts, interprets the valid contexts, classifies them as hierarchical agent objects, and associates them in the overlay space. As our CDP itself identifies and monitors the producers and consumers, it aids in achieving resource scalability. The context identification, interpretation, and classification as hierarchical agent objects are controlled by the context dissemination processor. The agent functionality works around virtualization constraints and significantly reduces the number of overlay rebuilding instances. This involved proxy and mobile agent functionalities.

8.1.2 Pure Overlay Space

We demonstrated the advantages of the proposed Pure Overlay Space based architecture which acts as a base for the implementation of context information dissemination. The spatial decoupling and semantic clustering were based on novel proxy overlay mechanism. The optimizations adapt the spatial and semantic overlay network formation process in a localized fashion to directly satisfy the context requestor's requirements. This thesis presented an ambient network aware context dissemination architecture based on a personalized and customized overlay environment towards ambient networking. We support both profile-rated and performance-related overlay customization and personalization, as application developers can specify their own resultant criteria which may include latency, cost, and other-universal metrics and thresholds. The mandatory human attention that is necessary to personalize the overlays in our context dissemination mechanism can be seen as a bottleneck in network computing. By autonomic management of target-specific context information at network-level we can reduce the amount of human attention required.

8.1.3 Multi-level Overlay Model

The CSON-D protocol's multi-level overlay structure, with the help of learning-based intelligence, personalization, and fault tolerance features, exhibits adaptive behaviour. The overlay mechanism acts as a multi-level filter for context dissemination. A single CSON/ RSON demonstrates the advantages of the dissemination architecture in heterogeneous environments by answering the substantial questions: *whom to tell* and *whom to ask*, thus avoiding flooding. The overlay clustering functionality features low delay, low cost, and low resource utilization including bandwidth. As a result, we get the benefits of a structured approach without having to maintain one. In addition, the clusters are extensible, thus making the dissemination mechanism scalable to wide range of systems and applications. The scenario wherein each overlay node can be a part of more than one CSON is also considered. Herein, such nodes could be responsible for more than one context dissemination service at the same time. When combined with RSON as a hybrid loosely coupled multi-level overlay structure that adopts fairness and dynamicity, it exhibits dynamic context dissemination in a load balanced and cost effective fashion based on context requestors' current state of affairs. Another main objective in designing the application-layer dissemination protocol is to keep the casting requirements to a minimum. The top-level overlays are made up of multiple unicasts to perform casting. We also follow sequential priority assignment to tolerate node failures.

8.1.4 Real-Time Context Dissemination Infrastructure

The geo-tracking service is essential for organizations that recognize the value of mobile asset tracking and routing for increasing productivity, improving asset utilization, and increasing visibility. Though many adaptation approaches are made available in recent years, only few of them are suitable for real time adaptation, to result in an efficient tool for geo-data retrieval and context knowledge discovery. The aim of our application is to discuss furthering of the geo-tracking mechanism, making it adaptive and intelligent, with efficient and cost-effective route tracking by utilizing context monitoring and dissemination

schemes. Context monitoring is performed to track mobile resources and manage their dynamic location.

The fundamentals of four types of algorithms (fixed interval, radius, direction, and speed) along with the combined Waterfall strategy have been described. We developed such a mechanism to optimize reporting and generate traces which are more fitting with actual routes to give realistic and cost effective route tracking. Our proposed GeoTracker made this goal feasible by efficiently analyzing the raw GPS data based on current mapping conditions using context-based learning-capable profiles with agent utilization to make dynamic reporting and routing more economical and logical. Though the use of AI has been historically discarded for such applications due to the constraints on data processing and power consumption, the GeoTracker's adaptive tracking algorithm uses an intelligent LE-based route learning technique to decide if and when the tracking information has to be sent to the database server. In densely populated areas with multiple routes, lack of this mechanism will be a critical issue.

8.2 Future Work

Suggested future research directions to extend the proposed CSON-D protocol for the betterment of network composition in ANs [137] and to help the wireless and mobile industry to reduce the cost of data transmission and consumers to save money, include:

- Autonomic management of target-specific context information at network-level to reduce the amount of human attention required.
- Find a concrete solution to the storage issue i.e., ownership of CSONs along with mobile database support for dynamic network composition
- Automatically cooperate and run applications by anticipating user's actions to realize *Ambient Intelligence* in the resulting customized RSON.
- Consider the use of context inter-domain agreements and contracts to minimize faults and improve the *Quality of Context* disseminated. The employed

dissemination strategy should be capable of making dynamic decisions based on real-time negotiation of entity's contracts.

- Alongside the improvement of SCF and KB performance in the real-time application, support for *Security* and *Privacy* to honour privacy concerns of GeoTracker users in their everyday life.
- Identify potential adaptive learning based solutions through the application of novel *Computational Intelligence* methods.
- Perform real-time evaluations in an ambient network infrastructure in future to test the flexibility and scalability of the protocol in large scale scenarios by integrating with Ambient Network Interface.

List of Publications

- Dineshbalu Balakrishnan and Amiya Nayak, "Adaptive Context Dissemination in Heterogeneous Environments", *IEEE Transactions on Mobile Computing (TMC Journal)*, under review.

- Dineshbalu Balakrishnan and Amiya Nayak, "A Context-specific Pure Overlay Space for Disseminating Contexts in Ambient Networks", *IEEE Transactions on Services Computing (TSC Journal)*, submitted.

- Dineshbalu Balakrishnan, Amiya Nayak, and Pulak Dhar, " Adaptive and Intelligent Route Learning for Mobile Assets using Geo-Tracking and Context Profiles ", *7th IEEE/IFIP Intl. Conference on Embedded and Ubiquitous Computing (EUC '09)*, Vancouver, Canada, Aug 2009.

- Dineshbalu Balakrishnan and Amiya Nayak, "A Context Dissemination Scheme for Adaptive and Fault-Tolerant Services in Heterogeneous Environments", *IEEE Global Information Infrastructure Symposium (GIIS '09)*, Hammamet, Tunisia, Jun 2009.

- Dineshbalu Balakrishnan, Amiya Nayak, Pulak Dhar, and Shailesh Kaul, "Efficient Geo-Tracking and Adaptive Routing of Mobile Assets", *11th IEEE Intl. Conference on High Performance Computing and Communications (HPCC '09)*, Seoul, Korea, Jun 2009.

- Dineshbalu Balakrishnan, Amiya Nayak, and Pulak Dhar, "CSON-D: An Ambient Network Aware Context Dissemination Scheme towards Ubiquitous Networking", *4th IEEE/IFIP International Symposium on Network Centric Ubiquitous Systems (NCUS '08) held in conjunction with 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC '08)*, Shanghai, China, Dec 2008.

- Dineshbalu Balakrishnan, Amiya Nayak, and Pulak Dhar, "Towards a Realistic Context Dissemination Protocol using Pure Multi-level Overlay Networks", *16th IEEE International Conference on Networks (ICON '08)*, New Delhi, India, Dec 2008.

- Dineshbalu Balakrishnan and Amiya Nayak, "CSON-D: Context Dissemination in Autonomous Systems using Overlays", *4th IET International Conference on Intelligent Environments (IE '08)*, Seattle, USA, Jul 2008.

- Dineshbalu Balakrishnan and Amiya Nayak, "An Efficient Context-specific Pure Overlay Space for Context Dissemination in Ambient Networks", *IEEE 11th Intl. Conference on Computational Science and Engineering (CSE '08)*, Sao Paulo, Brazil, Jul 2008.

- Dineshbalu Balakrishnan, May El Barachi, Ahmed Karmouch, and Roch Glitho, "Challenges in Modeling and Disseminating Context Information in Ambient Networks", *2nd International Workshop on Mobility Aware Technologies and Applications (MATA '05)*, Montreal, Canada, Oct 2005.

References

[1] N. Davies and H. Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems", *IEEE Pervasive Computing*, vol.1 no.1, pp. 26-35, Jan 2002.

[2] R. Cohen and D. Raz, "An Open and Modular Approach for a Context Distribution System", *2004 IEEE/IFIP Network Operations & Management Symposium (NOMS2004)*, Korea, pp. 365-379, Apr 2004.

[3] A.K. Dey, "Understanding and using context", *Journal of Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4-7, Feb 2001.

[4] M.K. Pinheiro, M. Villanova-Oliver, J. Gensel, Y. Berbers, and H. Martin, "Personalizing Web-Based Information Systems through Context-Aware User Profiles", *2nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM '08)*, Spain, pp. 231-238, Sep-Oct 2008.

[5] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research," *tech. report TR2000-381*, Department of Computer Science, Dartmouth College, Nov. 2000.

[6] A.K. Dey and G.D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", *In the Workshop on the What, Who, Where, When, and How of Context-Awareness (CHI 2000)*, Netherlands, Apr 2000.

[7] R. Szymacha, T. Szydlo, K. Zielinski, K. Jean, and A. Galis, "ContextWare Architecture for Ambient Networks", *Second International Conference on Communications and Networking (CHINACOM '07)*, China, pp. 734-738, Aug 2007.

[8] A. Karmouch, A. Galis, R. Giaffreda, T. Kanter, A. Jonsson, A.M. Karlsson, R. Glitho, M. Smirnov, M. Kleis, C. Reichert, A. Tan, M. Khedr, N. Samaan, L. Heimo, M.E. Barachi, and J. Dang, "Contextware Research Challenges in Ambient Networks", *1st International Workshop on Mobility Aware Technologies and Applications*, Brazil, pp. 62-77, Oct 2004.

[9] M. Michou, A. Bikakis, T. Patkos, G. Antoniou, and D. Plexousakis, "A Semantics-based User Model for the Support of Personalized, Context-Aware Navigational Services", *2008 First International Workshop on Ontologies in Interactive Systems (ONTORACT '08)*, U.K., pp.41-50, Sep 2008.

[10] A. Moon, Y.I. Choi, and B.S. Lee, "Context-Aware User Model for Personalized Services", *3rd International Conference on Digital Information Management (ICDIM '08)*, U.K. pp. 858-863, Nov 2008.

[11] B.N. Schilit and M.M. Theimer, "Disseminating active map information to mobile hosts", *IEEE Network*, vol. 8 no. 5, pp. 22--32, 1994.

[12] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, and H. Karl, "Ambient Networks - An Architecture for Communication Networks Beyond 3G", *IEEE Wireless Communications*, vol. 11, no. 2, pp. 14–22, 2004.

[13] N. Niebert, M. Prytz, A. Schieder, L. Eggert, N. Papadoglou, F. Pittmann, and C. Prehofer, "Ambient Networks: A Framework for Future Wireless Internetworking", *IEEE 61st Semiannual Vehicular Technology Conference (VTC 2005)*, Sweden, pp. 2969-2973, May-Jun 2005.

[14] G. Gehlen, F. Aijaz, S. Muhammad, and B. Walke, "A Rule Based Publish/Subscribe Context Dissemination Middleware", *In Proceedings of Wireless Communications and Network Conference (WCNC '07)*, Hong Kong, pp. 2541-2546, Mar 2007.

[15] D. Balakrishnan, M.E. Barachi, A. Karmouch, and R. Glitho, "Challenges in Modeling and Disseminating Context Information in Ambient Networks", *2nd International Workshop on Mobility Aware Technologies and Applications (MATA '05)*, Canada, pp. 32-42, Oct 2005.

[16] C. Anagnostopoulos, A. Tsounis, and S. Hadjiefthymiades, "Context awareness in mobile computing environments", *Wireless Personal Communications*, vol. 42, no. 3, pp. 445-464, 2007.

[17] T. Gu, E. Tan, H.K. Pung, and D. Zhang, "ContextPeers: Scalable Peer-to-Peer Search for Context Information", *In Proceedings of International Workshop on Innovations in Web Infrastructure (IWI '05)*, Japan, May 2005.

[18] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, J.W. O'Toole Jr., "Overcast: Reliable Multicasting with an Overlay Network", *In Proceedings of the 4th Symposium on Operating System Design and Implementation*, U.S.A., pp. 197-212, Oct 2000.

[19] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems", *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 37, no. 6, pp. 1067-1080, Nov 2007.

[20] B. Ding, L. Chen, D. Chen, and H. Yuan, "Application of RTLS in Warehouse Management Based on RFID and Wi-Fi", *In Proceedings of the 4th IEEE International Conference on Wireless Communications, Networking, and Mobile Computing*, Dalian, China, Oct 2008.

[21] S. Eichler, C. Schroth, T. Kosch, and M. Strassberger, "Strategies for Context-Adaptive Message Dissemination in Vehicular Ad Hoc Networks", *In Proceedings of the Second International Workshop on Vehicle-to-Vehicle Communications (V2VCOM '06)*, U.S.A., pp. 1-9, Jul 2006.

[22] C. Ma and Y. Yang, "Battery-aware routing for streaming data transmissions in wireless sensor networks", *ACM/Kluwer Mobile Networks and Applications*, vol. 11, no. 5, pp. 757-767, Oct 2006.

[23] M. Johnsson, J. Sachs, T. Rinta-aho, and T. Jokikyyny, "Ambient Networks – A Framework for Multi-Access Control in Heterogeneous Networks", *2006 IEEE 64th Vehicular Technology Conference (VTC Fall 2006)*, Montreal, Canada, pp. 1-5, Sep 2006.

[24] J. Rey, D. Lozano, S. Herborn, K. Ahmed, S. Schmid, F. Hartung, M. Kampmann, and M. Kleis, "SMART: A Media Service Delivery Architecture for Ambient Networks", *14th IST Mobile and Wireless Communications Summit*, Germany, Jun 2005.

[25] K. Balos, T. Szydło, R. Szymacha, and K. Zieliński, "Context Dissemination and Aggregation for Ambient Networks: Jini Based Prototype," *1st European Conference on Smart Sensing and Context*, Netherlands, pp. 54-66, Oct 2006.

[26] A. Karmouch, R. Giaffreda, A. Jonsson, A. Galis, M. Smirnov, R Glitho, and A. Karlsson, "Context Management Architecture for Ambient Networks", *Proceedings of WWRF 12 – Wireless World Research Forum Meeting 12*, Canada, Nov 2004.

[27] F. Belqasmi, R. Glitho, and R. Dssouli, "Ambient Network Composition", *IEEE Network Magazine*, vol. 22, no. 4, pp. 6-12, Jul-Aug 2008.

[28] R. Ocampo, L. Cheng, K. Jean, A. Galis, and A.G. Prieto, "Towards a Context Monitoring System for Ambient Networks", *First International Conference on Communications and Networking (CHINACOM '06)*, China, pp. 1-3, Oct 2006.

[29] R. Dantas, J. Fidalgo, D. Sadok, C. Kamienski, and B. Ohlmarr, "Policies for the Management of Ambient Networks: From Theory to Practice", *IEEE Workshop on*

Policies for Distributed Systems and Networks (POLICY '08), U.S.A., pp. 37-45, Jun 2008.

[30] D. Balakrishnan and A. Nayak, "An Efficient Context-specific Pure Overlay Space for Context Dissemination in Ambient Networks", *IEEE 11th International Conference on Computational Science and Engineering (CSE '08)*, Brazil, pp. 23-30, Jul 2008.

[31] D. Balakrishnan, A. Nayak, and P. Dhar, "CSON-D: An Ambient Network Aware Context Dissemination Scheme towards Ubiquitous Networking", *4th IEEE/IFIP International Symposium on Network Centric Ubiquitous Systems (NCUS '08) and 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC '08)*, China, pp. 543-548, Dec 2008.

[32] D. Balakrishnan, A. Nayak, and P. Dhar, "Towards a Realistic Context Dissemination Protocol using Pure Multi-level Overlay Networks", *16th IEEE International Conference on Networks (ICON '08)*, India, Dec 2008.

[33] D. Balakrishnan and A. Nayak, "CSON-D: Context Dissemination in Autonomous Systems using Overlays", *4th IET International Conference on Intelligent Environments (IE '08)*, U.S.A., Jul 2008.

[34] D. Balakrishnan and A. Nayak, "A Context Dissemination Scheme for Adaptive and Fault-Tolerant Services in Heterogeneous Environments", *IEEE Global Information Infrastructure Symposium (GIIS '09)*, Hammamet, Tunisia, Jun 2009.

[35] D. Balakrishnan, A. Nayak, P. Dhar, and S. Kaul, "Efficient Geo-Tracking and Adaptive Routing of Mobile Assets", *11th IEEE International Conference on High Performance Computing and Communications (HPCC '09)*, Seoul, Korea, Jun 2009.

[36] D. Balakrishnan and A. Nayak, "Adaptive and Intelligent Route Learning for Mobile Assets using Geo-Tracking and Context Profiles", *7th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC '09)*, Vancouver, Canada, Aug 2009.

[37] R. Kodikara, C. Åhlund, and A. Zaslavsky, "Towards Context Aware Adaptation in Wireless Networks", *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM '08)*, Valencia, Spain, pp. 245-250, Sep-Oct 2008.

[38] N. Shadbolt, "Ambient Intelligence", *IEEE Intelligent Systems*, vol. 18, no. 4, 2003.

[39] W. Fontijn and P. Boncz, "AmbientDB: P2P Data Management Middleware for Ambient Intelligence", *Middleware Support for Pervasive Computing Workshop at the 2nd Conference on Pervasive Computing*, U.S.A., pp. 203-207, Mar 2004.

[40] H. Moungla, "A Context-Aware, Policy-based Framework for Ambient Network", *IEEE Workshop on Policies for Distributed Systems and Networks (POLICY 2008)*, Palisades, NY, pp. 203-206, Jun 2008.

[41] S.S. Yau and F. Karim, "An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments", *Real-Time Systems*, vol. 26, no. 1, pp. 29-61, Jan 2004.

[42] B. Goncalves, P. Costa, and L.M. Botelho, "Context-Awareness", Chapter 5 in *CASCOM: Intelligent Service Coordination in the Semantic Web*, M. Schumacher, H. Helin, and H. Schuldt, Book Series: Whitestein Series in Software Agent Technologies and Autonomic Computing, ISBN: 978-3-7643-8575-0, Springer, 2008.

[43] C. Julien and G.C. Roman, "Egocentric Context-Aware Programming in Ad Hoc Mobile Environments", *Proceedings of the tenth ACM SIGSOFT symposium on Foundations of software engineering*, U.S.A., pp. 21-30, Nov 2002.

[44] M. Franklin and S. Zdonik, "Dissemination-Based Information Systems", *IEEE Data Engineering Bulletin*, vol. 19, no. 3, Sep 1996.

[45] P.D. Costa, J. Gonçalves, P. Filho, and M.V. Sinderen, "Architectural Requirements for Building Context-Aware Services Platforms", *9th European Summer School and IFIP Workshop on Next Generation Networks (EUNICE 2003)*, Hungary, pp. 62-69, Sep 2003.

[46] D. Petrelli, E. Not, M. Zancanaro, C. Strapparava, and O. Stock, "Modeling and Adapting to Context", *Personal and Ubiquitous Computing*, vol. 5, U.K., pp. 20-24, 2001.

[47] C. Kappler, P. Mendes, C. Prehofer, P. Pöyhönen, and D. Zhou, "A Framework for Self-Organized Network Composition", *Proceedings of the 1st IFIP International Workshop on Autonomic Communication (WAC 2004)*, Berlin, Germany, pp. 139-151, Oct 2004.

[48] S. Schmid, L. Eggert, M. Brunner, and J. Quittek, "Towards Autonomous Network Domains", *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, Miami, USA, pp. 2847-2852, Mar 2005.

[49] T. Gu, H.K. Pung, and D.Q. Zhang, "A Middleware for Building Context-Aware

Mobile Services", *In Proceedings of IEEE Vehicular Technology Conference (VTC 2004)*, Italy, pp. 2656-2660, May 2004.

[50] B. Mathieu, M. Song, A. Galis, L. Cheng, K. Jean, R. Ocampo, M. Brunner, M. Stiemerling, and M. Cassini, "Self-Management of Context-Aware Overlay Ambient Networks", *10th IFIP/IEEE International Symposium on Integrated Network Management (IM '07)*, Munich, Germany, pp. 749-752, May 2007.

[51] B. Mathieu, A. Galis, K. Jean, R. Ocampo, Z. Lai, M. Stiemerling, M. Cano, M. Kampmann, M. Tariq, K. Balos, K. Ahmed, B. Busropan, and M. Prins, "Dynamic Adaptable Overlay Networks for Personalised Service Delivery", *1st Ambient Networks Workshop on Mobility, Multiaccess, and Network Management (M2NM '07)*, Australia, Oct 2007.

[52] S.B. Kodeswaran, "Content and Context Aware Networking using Semantic Tags", *Doctoral Dissertation*, Department of Computer Science and Electrical Engineering, University of Maryland, 2008.

[53] D.G. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient Overlay Networks", *Proceedings of the eighteenth ACM symposium on Operating systems principles*, Banff, Canada, pp. 131-145, Oct 2001.

[54] A. Crespo and H. Garcia-Molina, "Semantic overlay networks for p2p systems", *Lecture Notes in Computer Science - Agents and Peer-to-Peer Computing*, vol. 3601, pp. 1-13, 2005.

[55] O. Papaemmanouil and U. Cetintemel, "SemCast: Semantic Multicast for Content-based Data Dissemination", *In Proceedings of the 21st International Conference on Data Engineering (ICDE '05)*, Japan, pp. 242-253, Apr 2005.

[56] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization", *IEEE Computer Magazine*, vol. 38, no. 4, pp. 34-41, Apr 2005.

[57] O. Shehory, K. Sycara, P. Chalasani, and S. Jha, "Agent Cloning: An Approach to Agent Mobility and Resource Allocation", *IEEE Communications Magazine*, vol. 36, no. 7, pp. 58-67, Jul 1998.

[58] H. Helin, H. Laamanen, and K. Raatikainen, "Mobile Agent Communication in Wireless Networks", *In Proceedings of the European Wireless '99*, Germany, pp. 211-216, Oct 1999.

[59] S.J. Poslad, S.J. Buckle, and R. Hadingham, "The FIPA-OS agent platform: Open Source for Open Standards", *Proceedings of the Fifth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000)*, U.K., pp. 355-368, Apr 2000.

[60] T. Kapyla, I. Niemi, and A. Lehtola, "Towards an Accessible Web by Applying PUSH Technology", *4th ERCIM Workshop on User Interfaces for All*, Sweden, pp. 133-147, Oct 1998.

[61] H.S. Nwana, "Software Agents: An Overview", *Knowledge Engineering Review*, vol. 11, no. 3, pp.1-40, Sep 1996.

[62] P. Maes, "Agents that Reduce Work and Information Overload", *Communications of the ACM (CACM)*, vol. 37, no. 7, pp. 31-40, Jul 1994.

[63] D. Balakrishnan and A. Karmouch, "A Personal Assistant for Mobile Device Users using Agents", *22nd Biennial Symposium on Communications*, Canada, pp. 405-407, May-Jun 2004.

[64] C. Kamienski, J. Fidalgo, R. Dantas, D. Sadok, and B. Ohlman, "Design and Implementation of a Policy-based Management Framework for Ambient Networks: Choices and Lessons Learned", *IEEE Network Operations and Management Symposium (NOMS 2008)*, Salvador, Bahia, pp. 775-778, Apr 2008.

[65] L. Kagal, T. Finin, and A. Joshi, "A Policy Language for A Pervasive Computing Environment", *IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, Italy, pp. 63-76, Jun 2003.

[66] K. Jean, K. Yang, and A. Galis, "A Policy Based Context-aware Service for Next Generation Networks", *8th London Communication Symposium*, U.K., Sep 2003.

[67] K. Cho, I. Hwang, S. Kang, B. Kim, J. Lee, S. Lee, S. Park, J. Song, and Y. Rhee, "HiCon: A Hierarchical Context Monitoring and Composition Framework for Next-Generation Context-Aware Services", *IEEE Network (Special Issue: Composable Context Aware Services)*, vol. 22, no. 4, pp. 34-42, 2008.

[68] H. Peizhao, J. Indulska, and R. Robinson, "An Autonomic Context Management System for Pervasive Computing", *The Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, Hong Kong, pp. 213-223, Mar 2008.

[69] V. Naik, A. Arora, P. Sinha, and H. Zhang, "Sprinkler: A Reliable and Energy Efficient

Data Dissemination Service for Extreme Scale Wireless Networks of Embedded Devices", *IEEE Transactions on Mobile Computing*, vol. 6, no. 7, pp. 762-776, Jul 2007.

[70] T. Tang, M. Zhengkun, and P. Rongqun, "Adaptive Service Provisioning through Context-aware SIP Proxy", *Fourth International Conference on Networking and Services (ICNS '08)*, Guadeloupe, pp. 277-281, Mar 2008.

[71] Y. Al Ridhawi and A. Karmouch, "Ontology-based negotiation protocol and context-level agreements", *4th IET International Conference on Intelligent Environments*, USA, Jul 2008.

[72] A. Moon, Y.I. Choi, and B.S. Lee, "Context-aware user model for personalized services", *Third International Conference on Digital Information Management*, UK, Nov 2008, pp. 858-863.

[73] P. Nurmi and S. Bhattacharya, "Identifying Meaningful Places - The Nonparametric Way", *Proceedings of the 6th International Conference on Pervasive Computing*, Australia, pp. 111-127, May 2008.

[74] R. Ocampo, A. Galis, C. Todd, and H. De Meer, "Towards Context-Based Flow Classification", *Proceedings of the 2006 International Conference on Autonomic and Autonomous Systems (ICAS '06)*, U.S.A., Jul 2006.

[75] H.K. Kim and J.J. Moon, "Route Tracking of Moving Magnetic Sensor Objects and Data Processing Module in a Wireless Sensor Network", *International Conference on Smart Manufacturing Application*, Korea, pp. 343-348, Apr 2008.

[76] J. Chang, S. Na, and M. Yoon, "Intelligent Context-Aware System Architecture in Pervasive Computing Environment", *The 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA-08)*, Sydney, Australia, pp. 745-750, Dec 2008.

[77] Y. Luo, O. Wolfson, and B. Xu, "A Spatio-Temporal Approach to Selective Data Dissemination in Mobile P2P Networks", *In Proceedings of the 3rd International Conference on Wireless and Mobile Communications (ICWMC '07)*, French Caribbean, 2007.

[78] C.G. Jansson, M. Jonsson, T. Kanter, F. Kilander, G. Maguire, and L. Wei, "Context Middleware for Adaptive Services in Heterogeneous Wireless Networks", *IEEE 61st Vehicular Technology Conference*, Sweden, pp. 2954-2958, June 2005.

[79] IETF charter: *SIP for Instant Messaging and Presence Leveraging Extensions (simple)*, May 2004, available at http://www.ietf.org/html.charters/simple-charter.html.

[80] G. Banavar, M. Kaplan, K. Shaw, R.E. Strom, D.C. Sturman, and W. Tao, "Information Flow Based Event Distribution Middleware", *In Proc. of the 19th IEEE International Conference on Distributed Computing Systems (ICDCS)*, U.S.A., pp. 114-121, May-Jun 1999.

[81] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, "Design and evaluation of a wide-area event notification service", *ACM Transactions on Computer Systems (TOCS)*, vol. 19, no. 3, pp. 332–383, Aug 2001.

[82] G. Chen and D. Kotz, "Solar: A pervasive-computing infrastructure for context-aware mobile applications", *tech. report TR2002-421*, Department of Computer Science, Dartmouth College, Feb 2002.

[83] G. Chen and D. Kotz, "Context Aggregation and Dissemination in Ubiquitous Computing Systems", *In Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, U.S.A., Jun 2002.

[84] C. Jacob, D. Linner, S. Steglich, and I. Radusch, "Autonomous Context Data Dissemination in Heterogeneous and Dynamic Environments", *In Proceedings of the 4th Consumer Communications and Networking Conf. (CCNC '07)*, USA, pp. 425-429, Jan 2007.

[85] D. Linner, I. Radusch, S. Steglich, and C. Jacob, "Loosely Coupled Service Provisioning in Dynamic Computing Environments", *In Proceedings of the 1st International Conference on Communication and Networking in China (CHINACOM '06)*, Beijing, China, Oct 2006.

[86] M. Kampmann, S. Bleiholder, A. Tariq, K. Jean, L. Zhaohong, M.C. Soveri, K. Balos, and M. Stiemerling, "ASI - the Ambient Network Service Interface", *Second International Conference on Communications and Networking (CHINACOM '07)*, China, pp. 747-753, Aug 2007.

[87] T. Szydlo, R. Szymacha, K. Balos, and K. Zielinski, "Context Dissemination for Ambient Networks Composition and Decomposition", *1st Ambient Networks Workshop on Mobility, Multiaccess, and Network Management (M2NM '07)*, Sydney, Australia, Oct 2007.

[88] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg, "Content-Based Publish/Subscribe over Structured Overlay Networks", *In Proceedings of 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, U.S.A., pp. 437–446, Jun 2005.

[89] Y. Choi and D. Park, "Mirinae: A peer-to-peer overlay network for content-based publish/subscribe systems", *IEICE transactions on communications*, vol. 89, no. 6, pp. 1755-1765, 2006.

[90] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast", *In Proceedings of ACM Special Interest Group on Data Communications (SIGCOMM 2002)*, U.S.A., pp. 205-217, Aug 2002.

[91] Y.H. Chu, S.G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast", *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456-1471, Oct 2002.

[92] Y. Zhu, J. Guo, and B. Li, "oEvolve: Towards Evolutionary Overlay Topologies for High Bandwidth Data Dissemination", *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1237-1251, Sep 2004.

[93] A.C. Snoeren, K. Conley, and D.K. Gifford, "Mesh-Based Content Routing using XML", *In Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Canada, pp. 160-173, Oct 2001.

[94] W. Fenner, M. Rabinovich, K.K. Ramakrishnan, D. Srivastava, and Y. Zhang, "XTreeNet: Scalable Overlay Networks for XML Content Dissemination and Querying", *In Proceedings of the 10th International Workshop on Web Content Caching and Distribution (WCW '05)*, France, pp. 41-46, Sep 2005.

[95] S. Chetan, A. Ranganathan, and R. Campbell, "Towards fault tolerant pervasive computing," *Pervasive 2004 Workshop on Sustainable Pervasive Computing*, Linz/Vienna, Austria, pp. 38-44, Apr 2004.

[96] F.Y. Chen and S.T. Yuan, "A Contextualized Fault-tolerant Infrastructure for P2P Mobile Service Composition", *Proceedings of the 2004 IEEE International Conference on Services Computing*, China, pp. 217–224, Sep 2004.

[97] H. Weatherspoon, T. Moscovitz, and J. Kubiatowicz, "Introspective Failure Analysis: Avoiding Correlated Failures in Peer-to-Peer Systems", *In Proceedings of the International Workshop on Reliable Peer-to-Peer Distributed Systems (SRDS'02)*, Japan,

pp. 362-367, Oct 2002.

[98] S. Chetan, A. Ranganathan, and R. Campbell, "Towards Fault Tolerance Pervasive Computing", *IEEE Technology and Society Magazine*, vol. 24, no. 1, pp. 38-44, Spring 2005.

[99] O. Papaemmanouil, Y. Ahmad, U. Cetintemel, and J. Jannotti, "Application-Aware Overlay Networks for Data Dissemination", *22nd International Conference on Data Engineering Workshops (ICDEW '06)*, U.S.A., Apr 2006.

[100] Y. Chawathe, "Scattercast: an adaptable broadcast distribution framework", *Multimedia Systems*, vol. 9, no. 1, pp. 104-118, Jul 2003.

[101] K. Jean, L. Cheng, R. Ocampo, and A. Galis, "Contextualisation of Management Overlays in Ambient Networks", *International Multi-Conference on Computing in the Global Information Technology (ICCGI '06)*, Bucharest, Romania, Aug 2006.

[102] T. Finin, A. Potluri, C. Thirunavukkarasu, D. Mckay, and R. Mcentire., "On Agent Domains, Agent Names and Proxy Agents", *In Proceedings of the ACM CIKM Intelligent Information Agents Workshop (CIKM '95)*, U.S.A., Dec 1995.

[103] M. Spreitzer and M. Theimer, "Providing location information in a ubiquitous computing environment", *In Proceedings of the fourteenth ACM symposium on Operating systems principles (SOSP '93)*, U.S.A., pp. 270-283, Dec 1993.

[104] G. Chen and D. Kotz, "Application-Controlled Loss-Tolerant Data Dissemination", *tech. report TR2004-488*, Department of Computer Science, Dartmouth College, Feb 2004.

[105] C. Reichert, M. Kleis, and R. Giaffreda, "Towards Distributed Context Management in Ambient Networks", *14th IST Mobile and Wireless Communication Summit*, Germany, Jun 2005.

[106] S. Willmott, F.O.F. Pena, C. Merida-Campos, I. Constantinescu, J. Dale, and D. Cabanillas, "Adapting Agent Communication Languages for Semantic Web Service Inter-Communication", *In Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, France, pp. 405-408, Sep 2005.

[107] BlackBerry Device Simulator, available at http://na.blackberry.com/eng/developers/resources/simulators.jsp.

[108] Java ME CLDC, available at http://java.sun.com/products/cldc.

[109] Robust Audio Tool (RAT), UCL Network and Multimedia Research Group, UK, at

http://www-mice.cs.ucl.ac.uk/multimedia/software/rat.

[110] Java Platform, Standard Edition (Java SE 6), available at http://java.sun.com/javase.

[111] Extensible Markup Language (XML) 1.0, available at http://www.w3.org/XML

[112] J. Bosak, "XML, Java, and the future of the Web", *World Wide Web Journal (Special issue on XML: principles, tools, and techniques)*, vol. 2, no. 4, pp. 219-227, 1997.

[113] Xerces Java Parser, v. 1.2.1, The Apache XML Project, available at http://xerces.apache.org/xerces-j.

[114] FIPA-OS and MicroFIPA-OS agent platforms, at http://sourceforge.net/projects/fipa-os.

[115] H. Tokuda, "Challenges in Creating New Context-aware Ubiquitous Services", *9th IEEE International Conference on E-Commerce Technology and 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, Japan, Jul 2007.

[116] R. Schmohl and U. Baumgarten, "A Generalized Context-aware Architecture in Heterogeneous Mobile Computing Environments", *The 4th International Conference on Wireless and Mobile Communications (ICWMC '08)*, Greece, pp. 118-124, Jul-Aug 2008.

[117] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications", *In Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM 2001)*, U.S.A., pp. 149-160, Aug 2001.

[118] G.D. Bhanage, Y. Zhang, Y. Zhang, W. Trappe, and R.E. Howard, "RollCall : The Design For A Low-Cost And Power Efficient Active RFID Asset Tracking System", *The International Conference on Computer as a Tool*, Poland, pp. 2521-2528, Sep 2007.

[119] F. Silva, V. Filipe, and A. Pereira, "Automatic Control of Students' Attendance in Classrooms Using RFID", *3rd International Conference on Systems and Networks Communications*, pp. 384-389, Malta, Oct 2008.

[120] R. Beuran, et al., "Active Tag Emulation for Pedestrian Localization Applications", *5th International Conference on Networked Sensing Systems (INSS 2008)*, Kanazawa, Japan, pp. 55-58, Jun 2008.

[121] J.H. Youn, H. Alia, H. Sharif, J. Deogun, J. Uher, and S.H. Hinrichs, "WLAN-based Real-time Asset Tracking System in Healthcare Environments", *3rd IEEE International*

Conference on Wireless and Mobile Computing, Networking, and Communications, USA, Oct 2007.

[122] P. Bahl and V.N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System", *In Proceedings of IEEE Infocom 2000*, Israel, pp. 775 - 784, Mar 2000.

[123] K. Kim, J. Jun, S. Kim, and B.Y. Sung, "Medical Asset Tracking Application with Wireless Sensor Networks", *Second International Conference on Sensor Technologies and Applications*, France, pp. 531-536, Aug 2008.

[124] S.J. Kim, J.H. Seo, J. Krishna, and Sun-Joong Kim, "Wireless sensor network based asset tracking service", *Portland International Conference for Management of Engineering and Technology*, South Africa, pp. 2643-2647, Jul 2008.

[125] N. Rajendran, P. Kamal, D. Nayak, and S.A. Rabara, "WATS-SN: a wireless asset tracking system using sensor networks", *The 2005 IEEE International Conference on Personal Wireless Communications*, India, pp. 237- 243, Jan 2005.

[126] QCTConnect, at http://www.qctconnect.com.

[127] Mobile Asset Tracker from CMO Global Services, at http://www.cmoglobal.com/ebusiness_solutions/wireless_mobile/mobile_asset_tracker_pda_web_tracking_software.html.

[128] MobileTrak 4.1 from RF Code, available at http://www.rfcode.com/index.php?option=com_content&view=article&id=137&Itemid=147.

[129] TrackServ from Maptuit, at http://corporate.maptuit.com/maptrack/trackserv.php.

[130] G.H. Truelove, M.A. Foster, V.K. Kohli, and T.G. Raslear, "Real-time asset tracking and monitoring using low-cost cellular networks", *In Proceedings of the 2006 IEEE/ASME Joint Rail Conference*, USA, pp. 315-318, Apr 2006.

[131] W. Ballantyne, G. Turetzky, G. Slimak, and J. Shewfelt, "PowerDown: Achieving Low Energy-Per-Fix in Cell Phones", *GPS World*, vol. 17, no. 7, Jul 2006, available at http://www.gpsworld.com/gpsworld/article/articleDetail.jsp? id=352807.

[132] G. Chen and D. Kotz, "Policy-Driven Data Dissemination for Context-Aware Applications", *3rd IEEE International Conference on Pervasive Computing and Communications*, USA, pp. 283-289, Mar 2005.

[133] Vayyoo Inc., at http://www.vayyoo.com.

[134] Google Maps JSP Tag Library, available at heep://www.lamatek.com/GoogleMaps.

[135] Keyhole Markup Language, available at http://code.google.com/apis/kml.

[136] Google Maps API, available at http://code.google.com/apis/maps; Google Earth, available at http://earth. google.com.

[137] N. Niebert, A. Schieder, J. Zander, and R. Hancock, Eds., *Ambient Networks: Co-operative Mobile Networking for the Wireless World*, Chichester: Wiley, 2007.

Appendices

Appendix A – GeoTracker Application

Once the GeoTracker application is launched in the mobile asset (in our case, BlackBerry), it displays the information shown with real GPS data in a user interface as seen in Figure A.1. As illustrated, the application software's algorithm *settings* can be configured in real-time. The asset will keep updating its location information until the *Stop* button is pressed. The GeoTracker with simulated location information from a streamed route file is shown in Figure A.2, wherein the *Map me* button is used to display the current location in BlackBerry Maps (see Figure A.3).

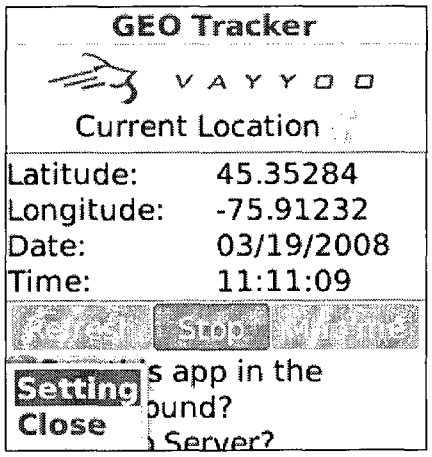


Figure A.1 Client device/module view: A snapshot of the GeoTracker application in a GPS-enabled BlackBerry.

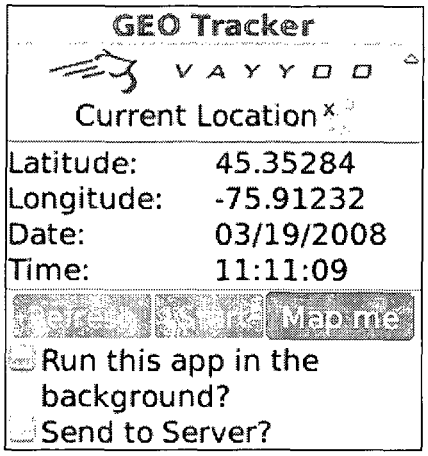


Figure A.2 Client device/module view: A snapshot of GeoTracker with simulated coordinates and *Map me* feature.

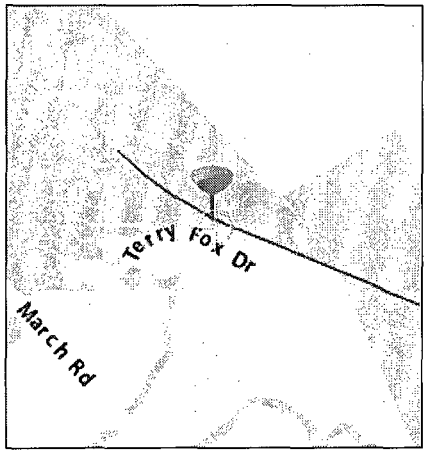
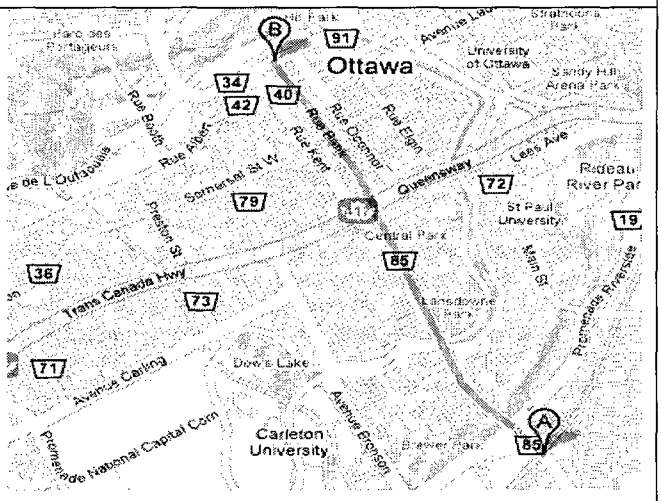
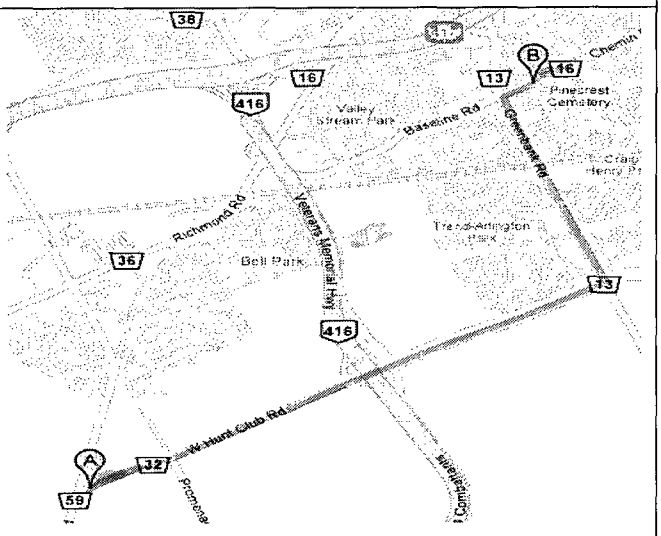


Figure A.3 Client device/module view: A snapshot of GeoTracker showing the result of *Map me* in BlackBerry maps.

The middleware for client-server integration is represented by Vayyoo's Mobile Data Convergence (MDC) platform [133]. The MDC platform supports cross platform dynamic data overlapping and allows for optimized communication between a mobile device and backend server or database. If changes are deemed necessary based on location information, the location data sent to the server in a specific XML format as HTTP POST is *listened*, *parsed*, and *updated* in the database.

Our independent simulator tool's input is a text file with routing information in a specific format containing the latitude and longitude coordinate values. They make sample routes which are geo-referenced predefined routes with route characteristics (for example, busy routes). The sample characteristics of different driving path scenarios are illustrated in Table A.1 and a sample route file is illustrated in Table A.2.

Table A.1 Sample driving path scenarios.

Route / Path	Start	End	Time	Map
1.1 / Busy	Bank Street + Riverside Drive	Bank Street + Wellington Street	Morning and Evening	
2.1 / Easy	Pinecrest Road + Baseline Road	W. Hunt Club Road + Richmond Road	Noon	
continued				

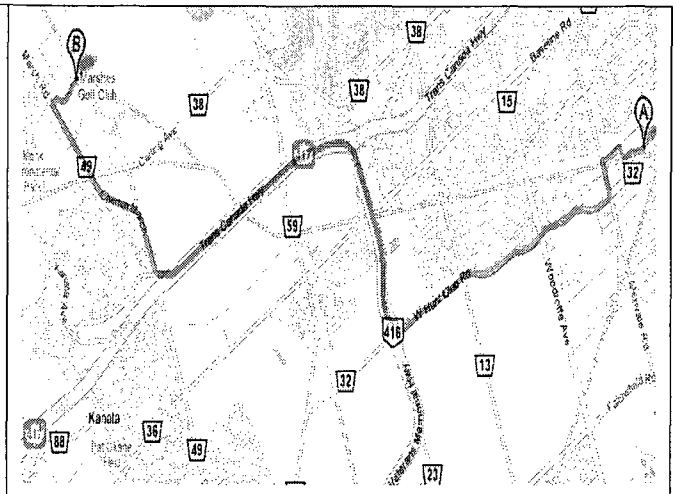
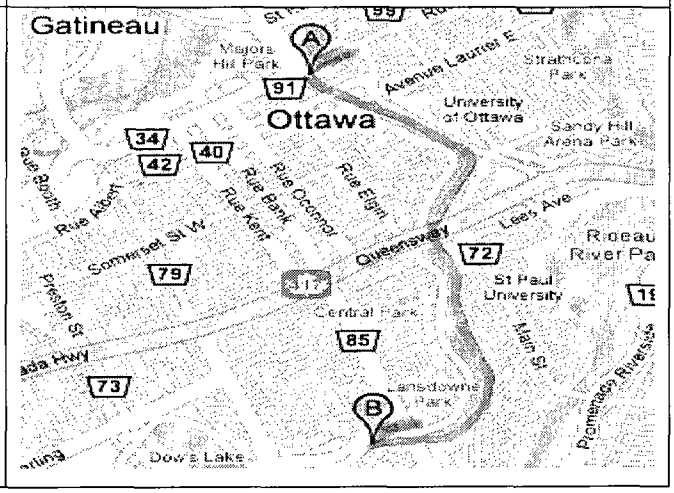
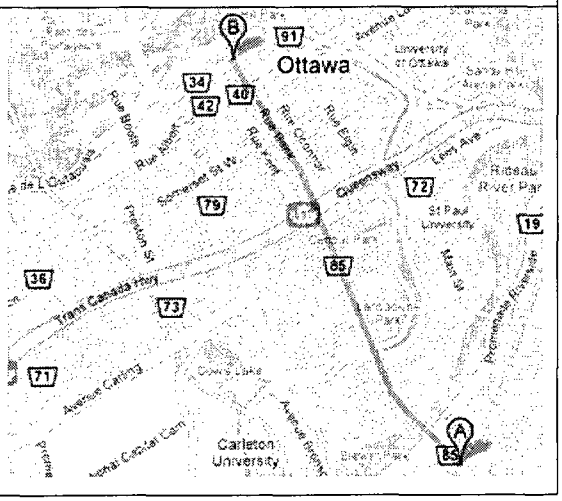
3.1 / Long	210 Colonnade Road	350 Terry Fox Drive	Busy – Evening Easy – Noon	
4.1 / Multi-turn	Colonel By Drive	Colonel By Drive	Any Time	

Table A.2 A sample route file.

Route	S#	Latitude	Longitude	Altitude	Map
1.1	1	45.421626	-75.701659	272.309711	
	2	45.421264	-75.70138	272.309711	
	3	45.420737	-75.700929	288.713910	
	4	45.420059	-75.70035	275.590551	
	5	45.419419	-75.699803	295.275590	
	6	45.418636	-75.699127	265.748031	
	7	45.417356	-75.698032	246.062992	