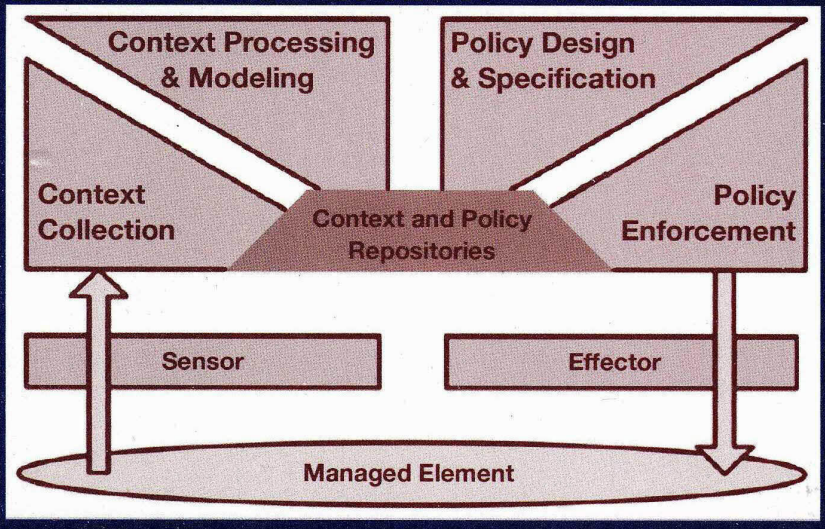
Chapman & Hall/CRC
Studies in Informatics Series

CONTEXT-AWARE COMPUTING AND SELF-MANAGING SYSTEMS



Edited by
Waltenegus Dargie

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

CONTEXT-AWARE COMPUTING AND SELF-MANAGING SYSTEMS

Dargie

CRC

CONTEXT-AWARE COMPUTING AND SELF-MANAGING SYSTEMS

Chapman & Hall/CRC
Studies in Informatics Series

SERIES EDITOR

G. Q. Zhang
Case Western Reserve University
Department of EECS
Cleveland, Ohio, U.S.A

PUBLISHED TITLES

Stochastic Relations: Foundations for Markov Transition Systems
Ernst-Erich Doberkat

Conceptual Structures in Practice
Pascal Hitzler and Henrik Schärfe

Context-Aware Computing and Self-Managing Systems
Waltenegus Dargie

Chapman & Hall/CRC
Studies in Informatics Series

CONTEXT-AWARE COMPUTING AND SELF-MANAGING SYSTEMS

Edited by
Waltenegus Dargie

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

First issued in paperback 2019

ISBN-13: 978-1-4200-7771-1 (hbk)
ISBN-13: 978-0-367-38584-2 (pbk)

Library of Congress Cataloging-in-Publication Data

Context-aware computing and self-managing systems / [edited by] Waltenegus Dargie. -- 1st ed.
p. cm. -- (Context-aware computing and self-managing systems)
Includes bibliographical references and index.
ISBN 978-1-4200-7771-1 (alk. paper)
1. Autonomic computing. I. Dargie, Waltenegus.

QA76.9.A97C65 2009
004--dc22 2008038056

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

To Pheben, with love. Welcome to the world.

Preface

This book brings two research issues together: context-aware computing and the self-managing aspect of autonomous computing. Context-aware computing is an extensively researched area, while self-managing systems are emerging. The goal of this book is to investigate the various roles context-aware computing can play to develop self-managing systems, where a self-management system can be a device, a middleware, an application, or a network.

The first chapter of the book identifies aspects that are common to both context-aware computing and autonomous computing. It offers a basic definition of context-awareness and provides several examples — more focus is given to the acquisition, presentation and management of context information. It presents as well basic aspects of self-managing systems and offers a few examples of self-managing systems.

The remaining part of the book is divided into context-awareness and self-management. The context-awareness subpart demonstrates how a context can be employed to make systems smart; how a context can be captured and represented; and how dynamic binding of context sources can be possible. The self-management subpart of the book demonstrates the need for "implicit-knowledge" to develop fault-tolerant and self-protective systems. It also presents a higher-level vision of future large-scale networks.

Several researchers have participated in editing this book. I would like to acknowledge the contributions of Prof. Noriaki Kuwahara (Kyoto Institute of Technology), Prof. Ren Ohmura (Keio University), Prof. Markus Endler (Catholic University of Rio de Janeiro), Prof. Antonio Alfredo F. Loureiro (The Federal University of Minas Gerais), Prof. Mieso Denko (University of Guelph), and Dr. Daniel Schuster (Technical University of Dresden) for reviewing some of the chapters. Of course, there were also a plethora of reviewers whose names I have not mentioned here, but who have reviewed each chapter of the book and provided critical feedbacks.

I would like to acknowledge the contribution of my former post graduate student, Rami Mochaourab, who worked tirelessly with LaTeX to provide the book the shape it now has. He was always available, always willing to try new ideas, and always on time. Without his support, the book would never be finished on time.

Dr. Waltenegus Dargie
Technical University of Dresden
Germany

Contributors

Abe, Akinori
ATR Knowledge Science
Laboratories
Kyoto, Japan

Beltran, Victoria
Wireless Networks Group
Department of Telematics
Technical University of Catalonia
Barcelona, Spain

Breitman, Karin
Departamento de Informática
Pontifícia Universidade Católica
(PUC-RJ)
Rio de Janeiro, Brazil

Briot, Jean-Pierre
Departamento de Informática
Pontifícia Universidade Católica
(PUC-RJ)
Rio de Janeiro, Brazil

Bouvry, Pascal
University of Luxemburg
Luxemburg

Casademont, Jordi
Wireless Networks Group
Department of Telematics
Technical University of Catalonia
Barcelona, Spain

Catalan, Marisa
Wireless Networks Group
Department of Telematics
Technical University of Catalonia
Barcelona, Spain

Charif, Yasmine
Laboratoire d'Informatique de Paris
Université Paris
Paris, France

Dargie, Waltenegus
Chair of Computer Networks
Faculty of Computer Science
Technical University of Dresden
Dresden, Germany

Davoli, Franco
Department of Communications,
Computer, and Systems Science
University of Genoa
Genoa Italy

Ding, Jianguo
University of Luxembourg
Luxembourg

El Fallah Seghrouchni, Amal
Laboratoire d'Informatique de Paris
Université Paris
Paris, France

Endler, Markus
Departamento de Informática
Pontifícia Universidade Católica
(PUC-RJ)
Rio de Janeiro, Brazil

Guan, Haibing
School of Information Security
Engineering
Shanghai Jiao Tong University
Shanghai, P. R. China

Hadjiantonis, Antonis M
Centre for Communication Systems
Research
Department of Electronic
Engineering,
University of Surrey
Guildford, UK

Hartel, Pieter
University of Twente
Enschede, The Netherlands

Klauser, Bruno
Cisco Europe
Glattzentrum, Switzerland

Kogure, Kiyoshi
ATR Knowledge Science
Laboratories
Kyoto, Japan

Krämer, Bernd J.
Fern Universität in Hagen
Hagen, Germany

Kuwahara, Noriaki
Kyoto Institute of Technology
Kyoto, Japan

Liang, Alei
Software School
Shanghai Jiao Tong University
Shanghai, P. R. China

Liu, Lei
Sun Microsystems, Inc.
Menlo Park, CA, USA

Luis Ferrer, Jose
Wireless Networks Group
Department of Telematics
Technical University of Catalonia
Barcelona, Spain

Mazuel, Laurent
Laboratoire d'Informatique de Paris
Université Paris
Paris, France

Naya, Futoshi
ATR Knowledge Science
Laboratories
Kyoto, Japan

Ohboshi, Naoki
Kinki University
Osaka, Japan

Ozaku, Hiromi Itoh
ATR Knowledge Science
Laboratories
Kyoto, Japan

Paradells, Josep
Wireless Networks Group
Department of Telematics
Technical University of Catalonia
Barcelona, Spain

Pavlou, George
Centre for Communication Systems
Research
Department of Electronic
Engineering
University of Surrey
Guildford, UK

Sabouret, Nicolas
Laboratoire d'Informatique de Paris,
Université Paris
Paris, France

Sanchez-Loro, Xavier
Wireless Networks Group
Department of Telematics
Technical University of Catalonia
Barcelona, Spain

Scholten, Hans
University of Twente
Enschede, The Netherlands

Sundramoorthy, Vasughi
Lancaster University
Lancaster, UK

Viterbo, José
Departamento de Informática
Pontifícia Universidade Católica
(PUC-RJ)
Rio de Janeiro, Brazil

Wolter, Ralf
Cisco Systems
Duesseldorf, Germany

Contents

List of Tables

List of Figures

Chapter 1

Context and Self-Management

Waltenegus Dargie

Technical University of Dresden, 01062, Dresden, Germany

Abstract

This chapter provides an introduction to Context-Aware Computing and Self-Managing Systems. It begins by explaining why self-management is desirable in complex systems and by describing self-management aspects (self-configuration, self-optimization, self-healing and self-protection). For all these features, a self-managing system's needs to have a perpetual awareness of what is taking place both within itself and without. It is this duly awareness of one's state and surrounding that leads to self-adaptation. As a result, the chapter tries to demonstrate the scope and usefulness of context-aware computing in developing self-managing systems.

1.1 Introduction

Computing systems are becoming very complex, highly heterogeneous and distributed. At the same time, the users of these systems are usually mobile and demand greater flexibility and efficiency in terms of response time, resource utilization, robustness, etc., to achieve critical business goals. The implication is that operating and maintaining computing systems is becoming an increasingly expensive business. In fact, Fox and Patterson claim that annual outlays for maintenance, repair and operations far exceed total hardware and software costs, for both individuals and corporations [1].

This high cost of ownership of computing systems has resulted in a number

of industry initiatives to reduce the burden of operations and management by making computing systems - at least gradually - self-managing. A few examples are IBM's Autonomic Computing, HP's Adaptive Infrastructure and Microsoft's Dynamic System Initiatives [2].

Self-management derives its basic principles from the autonomous nervous system, which governs our heart rate and body temperature, *thus freeing our conscious brain from the burden of dealing with these and many other low-level, yet vital, functions* [3]. This essential principle, if transferred well, enables computing systems, whether acting individually or collectively, to receive higher-level objectives from their operators (users) but manage to maintain and adjust their operation in the face of changing components, workloads, demands and external conditions as well as imminent hardware and software failures.

According to Kephart and Chess [3] and Tesauro et al. [4], a self-managing system contains an autonomic manager software and a (hardware or software) managed element. The managed element is what is being made self-managing and provides a sensing and actuating interface. Through the sensing interface, an array of sensors measure vital internal as well as external (environmental) phenomena which may potentially influence the system's short and long term performance. The actuating interface provides a way for the autonomic manager to modify the behavior of the managed element. The autonomic manager itself contains components for monitoring and analyzing sensor data and for planning and executing management policies. Common to all of these components is knowledge of the computing environment and service-level agreement as well as other related facts.

The monitoring component inside the autonomic manager is responsible for reducing the amount of raw sensor data by applying filtering and correlation operations on the data. The analysis component gets refined data from the monitoring component in order to identify emerging or foreseeable problems or potential causes of adaptation. The planning component accommodates workflows that specify a partial order of actions which should be carried out in accordance with the results of the analysis component. And finally, the execution component controls the execution of such workflows and provides coordination if there are multiple concurrent workflows.

1.2 Aspects of Self-Management

A system is said to be self-managing if it exhibits one or more of the following characteristics: self-configuration, self-optimization, self-healing, and self-protecting.

Self-configuration refers to the capability of a system to dynamically ad-

just one or more parameters to accommodate expected or unexpected change within itself or in the operating environment. The change may be due to departure, arrival or failure of a component; a change in the business policy of the user; or environmental, social or political constraints. A self-configuration capability of a system enables it to keep on functioning in the presence of continually changing and unforeseen obstacles.

Self-optimization refers to the ability of a system to tune its parameters so that it can function most efficiently. Efficiency can be measured in terms of cost, quality of service, throughput, etc. A self-optimizing system improves its performance by finding, verifying and applying the latest software updates.

Self-healing refers to the ability of a system to detect, localize, diagnose, and repair problems resulting from bugs or failures in software and hardware.

Finally, self-protection refers to the ability of a system to defend itself as a whole against large-scale, correlated problems arising from malicious attacks or cascading failures that remain uncorrected by self-healing measures. This also includes the anticipation of potential dangers and the carrying out of predictive measures to avoid or mitigate premeditative attacks.

1.3 Examples of Self-Managing Systems

So far, the motivation for self-managing systems was discussed conceptually. In the following two subsections, we will present examples of self-managing systems. In the first subsection, we will present a self-optimizing system which autonomously regulates the transmission power and data rate of distributed and independent wireless local area network access points in densely deployed metropolitan environments. In the second subsection, we will present a self-recovering satellite-receiver which localizes problems and dynamically reinitializes only those components which are the direct causes of the problem instead of considering component-specific problems as global phenomena.

1.3.1 Self-Managing Chaotic Networks

Akella et al. [5] propose self-managing algorithms to manage what they call *spontaneous* or *chaotic* wireless networks. As opposed to carefully planned and deployed wireless networks, chaotic networks are typically deployed spontaneously by individuals or independent organizations that set up one or a small number of APs. This type of unplanned, uncoordinated and unmanaged deployment results in highly variable densities of wireless nodes and APs and causes considerable interference and inefficient utilization of valuable resources such as spectrum and energy. Akella et al. report that in some metropolitan cities in the US as much as 8000 APs are deployed randomly in close proxim-

ity, each AP having more than 80 interfering APs. Moreover, users of these networks use default (factory-set) configurations for key parameters such as the transmission channel and transmission power.

The researchers propose two algorithms to autonomously manage transmission power levels and data rates. In combination with a careful channel allocation technique, the power control system attempted to minimize the interference between neighboring APs by reducing transmission power on individual APs. The power management algorithm reduces transmission power as long as the link between an AP and its client could maintain the maximum possible speed. An experiment result shows that the power control system improves throughput from 0.15 Mbps to 3.5 Mbps.

1.3.2 Recovery-Oriented Computing

Fox and Patterson report that operator error was a leading cause of problems of Internet systems [1]. They remarked that traditional efforts to boost the dependability of software and hardware have for the most part overlooked the possibility of human mistakes. Motivated by these observations, they propose four recommendations for developing self-recovering computing systems: Accordingly, developers should assume operator errors as inevitable problems and should therefore design systems that recover quickly. Second, operators should be provided with tools to localize the sources of faults in multicomponent systems. Third, the systems should provide support of an *undo* function so that operators can correct their mistakes. Forth, the systems should accommodate the injection of test errors to evaluate and predict system behavior.

To demonstrate the usefulness of this guideline, the researchers implemented a number of self-recovering systems. As an example, they built a satellite receiver in a *traditional* fashion, by employing inexpensive ground receivers assembled from commonly available PCs, low-cost ham radios and home developed software to capture incoming satellite data. Not surprisingly, whenever the system experienced failures, the operators had to restart it either preemptively (because the system was behaving strangely), or reactively (because it had crashed or seized up). The researchers reported that without a human operator to reactivate the equipment manually, the satellite signal could be lost, and with it, all the data for that orbit. Later, the researchers acquired domain knowledge about the most frequent causes of failure and modified each receiving-station software module so that only a subpart of the system's components should be reinitialized in the event of imminent failure. They made them succeed to automate the recovery process for a range of recurring problems. Furthermore, they improve the average restoration time from 10 minutes to 2 minutes.

1.4 Context-Aware Computing

An essential aspect of a self-managing system is the employment of sensors to unobtrusively measure and report relevant system properties. As far as dealing with sensor data is concerned, there are four main challenges [6]:

- The data can be incomplete, representing only a partial view of the state of the system or the operating environment;
- The data can be imprecise due to the limitation of the employed sensing elements. Different sensors have different resolutions, accuracies and sensing ranges. Besides, the performance of the sensors can be influenced by external factors such as surrounding noise or temperature;
- The data may not directly represent the desired aspect to be observed or measured. For example, end-to-end response time during a business transaction over the Internet is difficult and expensive to measure. Thus, surrogate metrics such as CPU queue length can be used to infer it [2]; and,
- There can be multiple measurement sources that produce both interval and event data, and the intervals may not be synchronized, for example, 10 seconds vs. 1 minute vs. 1 hour [2].

A significant body of work exists on sensor data fusion, filtering, interpolation and correlation. In this book, we will be investigating the role of context-aware computing in dealing with sensed data, in particular, and developing self-managing systems, in general. Some of the aspects of context-awareness we will be investigating more closely include declarative specification of sensors; dynamic binding of data sources; modeling and representing sensed data; and data analysis and reasoning.

1.4.1 Context-Awareness

The initial motivation for context-aware computing was the reduction of the explicit information a user needs during an interaction with a computer. The premises for this are the mobility and activity of users in ubiquitous computing environment. For example, a mobile user can interact with a computer while driving, talking to other people, holding a lecture or attending to a child [7].

Earlier approaches in the HCI community attempted to address this issue by (1) presenting to a user multiple modalities of interaction (gesture, voice, graphic, tactile, etc.) to make interaction intuitive; and (2) increasing the vocabularies of each modality to make interaction rich. In both cases, however, the interaction model requires explicit input from the user because the

computer is entirely unaware of and quite unable to utilize background information which can be vital to the understanding of the user's intentions or wishes.

The complementary approach is the use of implicit information which can be vital to the understanding of explicit inputs from a user [8]. The information may related to the user directly or to the physical surrounding wherein the interaction is unfolding. This information can be obtained from a variety of sources, including sensors and software services monitoring the status of a device, an application, a computing platform, a network or a part of the physical world.

The idea of using implicit information is taken from the way human beings communicate with each other, and how they exploit implicitly available information to increase their communication bandwidth. For example, when people attend a meeting, their eyes communicate to convey agreements or disagreements to what is said or unsaid; voices are whispered to exchange impromptu opinions; facial expressions reveal to the other participants fatigue, boredom or disinterest. More importantly, speeches may not be grammatically correct or complete. Previous as well as unfolding incidents enable the participants to capture what cannot be expressed verbally. Speakers shift from one language to another and use words with multiple meanings, and still the other participants can follow.

Flexibility and adaptation is possible because the social and conceptual setting (i.e., the context) encompassing human-human interaction is effortlessly recognized by all participants. As a result, within the perceived context, many activities unfold, some of which are unpremeditated, yet consistent with the context, while other activities express the freedom associated with the recognition of the context - for example, using incomplete or incorrect statements, or using words with multiple meanings. Still other activities reflect the participants' adjustment of behavior in compliance with the context of the setting - for example, participants whispering to exchange impromptu ideas.

Dey calls systems that use implicit information to provide useful services in a proactive manner context-aware [9] and categorizes context-awareness in one of the following aspects:

1. The presentation of information and services to a user;
2. The automatic execution of a service; and
3. The tagging of context to information for later retrieval.

In the first category, a context-aware system employs implicitly available information (for example, the location of a mobile user) to provide a service that is associated with the user's current activity. For example, a user whose present activity is printing a document will be presented with the list of nearby available printers; and a tourist will be provided with a description of a point of interest that matches his preference and location.

In the second category, a service is dynamically executed in accordance with his present context (activity or location). A typical example is the dynamic adjustment of the physical and ambient setting of a car (seat and mirror position; cabin temperature; and preferred radio station, etc.) according to a user's identity. In this case the context of interest is the identity of the user and it can be represented by an RFID.

In the third category, specific information is associated with activity, identity, location and time contexts, among others. The information will be displayed to the user when his context matches the ones with which the information is associated. For example, a user can request an email application to list all the emails he sent while attending a particular meeting. For this to happen, the email application must associate outgoing emails with the current setting (activity) of the user.

In all of the three categories the acquisition of a context of interest remains the same. Physical or software sensors are used to capture certain phenomenon. The sensor data are processed to extract meaningful information, and this information will be used as an implicit input for the system to carry out a task with minimum user's involvement. This aspect fits well with the vision of autonomous computing or self-managing systems.

In the following subsection, we provide some examples of context-aware systems to demonstrate how a context of interest is captured by employing physical sensors.

1.4.2 Surrounding Context

Eronen et al.[10] classify auditory scenes into predefined classes by employing two classification mechanisms: 1-NN classifier and Mel-frequency cepstral coefficients with Gaussian mixture models. The aim is to recognize a physical environment by using audio information only. The audio scene comprises several everyday outside and inside environments, such as streets, restaurants, offices, homes, cars, etc.

The features to be extracted for the purpose of classification are: zero-crossing rate (ZCR) and short-time average energy in time domain; band-energy ratio, spectral centroid, bandwidth, spectral roll-off and spectral flux in frequency domain; and linear prediction and cepstral features such as linear prediction coefficients (LPC), cepstral coefficient and Mel-frequency cepstral coefficients (MFCC). The classification systems classify 17 out of 26 indoor and outdoor scenes with an accuracy of 68.4% with analysis duration of 30 seconds. Each classified scene has at least five samples from different recording sessions before a classification process started. The classification performance is evaluated using leave-one-out cross-validation, where a classifier is trained with all instances except the one that is left out for the classification.

Korpip et al. [11] propose a multi-layered context-processing framework to carry out a similar work. The bottom layer is occupied by an array of sensors enclosed in a small sensor. The sensor board is attached to a shoulder strap of

a backpack containing a laptop. When collecting scenario data, a user carries the backpack. A cordless mouse controls the measurement system to mark the scenario phase. Nine channels are used to obtain data pertaining to a physical environment: three of the channels for a three-axis accelerometer, two for light intensity and one for temperature, humidity, skin conductivity and audio.

The other layers in the context processing hierarchy include a feature extraction layer incorporating a variety of audio signal processing algorithms. A naive Bayesian classifier reasons about a higher-level context by classifying the features extracted by the DSP algorithms. A total of 47 quantized audio features, including harmonicity ratio, spectral centroid, spectral spread, spectral flatness and fundamental frequency are used to describe seven audio related contexts: speech, rock music, classical music, car, elevator, running tap water or other sounds.

The framework provides support for quantifying the uncertainty associated with a recognition process. An additional merit of the framework includes training the model with data to recognize new contextual states.

1.4.3 Activity on a Street

Moenne-Loccoz et al. [12] propose architecture for modeling the temporal evolution of visual features characterizing a human behavior, and to infer their occurrences. The architecture consists of a vision module, an interpretation module, and a knowledge base. The vision module performs segmentation and classification on a video stream input. It tracks individuals or groups of individuals. The interpretation module recognizes a set of behaviours such as the fighting of individuals or vandalism. To ease the interpretation task, three entities are introduced to the knowledge base:

- State: it refers to the property of a mobile object. Examples are: seating/standing, still/walking/running.
- Event: it characterizes a change of state. Examples are: to sit down/to stand up, to stop/to begin running - to sit down, for instance, is the change of the state standing into the state seating.
- Scenario: it is a combination of states, events, and/or sub-scenarios. Examples are: running towards a train, following someone.

Additionally, the knowledge base defines a detailed description of the scene environment. This knowledge is used by the interpretation module. Knowledge of a scene environment includes the nature and position of still environment such as walls, benches and doors. The expert knowledge defines a complex scene in terms of simple scenes. For example, running towards a train is described by a combination of the chain of events: running, train present and trajectory is towards the train.

The prior knowledge along with the representation of the scene presented by the vision module is supplied to a Bayesian network inside the interpretation module to recognize hierarchically all the occurrences of states, events, and scenarios, which signify human activities.

Over 600 frames were used to train the network to recognized violent behaviors such as people fighting or show some pronounced agitation. 80% of the frames contain anticipated behaviour (violence) while the remaining 20% were spurious (no violence).

The architecture provides support for the dynamic definition of higher-level contexts; the input contexts are, unfortunately, limited to video features. Since a Bayesian network is employed for a recognition purpose, the uncertainty associated with a recognition task can be quantified. Belief revision is not treated in the architecture.

1.4.4 User's Attention in a Meeting

Wu [13] extends the functionalities of Dey's Context Toolkit [14] to support context fusion. Even though the Aggregator proposed by Dey gathers all relevant contexts of a particular entity, it does not actually process these contexts to achieve a meaningful understanding of the situation of the entity. This assignment is left to the applications themselves. Dey's argument for this is that an aggregation task is specific to each application. While this holds true, due to physical limitations of sensing elements and other external factors, propositions made by context sources (sensors) may lack the appropriate precision or abstraction.

Wu applies Dempster-Schafer's theory of evidence to deal with uncertainty associated with context sensing. In his implementation, an Aggregator receives video and audio features from a camera and a set of microphone widgets to determine the likelihood of a participant's focus of attention in a meeting.

The application scenario comprises a small round table in a small meeting room, where a few people sitting around the table participate in a discussion. An omni-directional camera at the center of the table captures the activities of the participants, while a microphone in front of each participant measures relative sound strength. A skin-color based face detector recognizes the face location, from which a participant's head pose is estimated using neural network algorithms. A Gaussian model is assumed to describe the head pan angle distribution. The head pose estimated from this process is the basis for estimating a participant's focus of attention.

Meanwhile, the relative sound strength from each microphone is used to determine the speaker at any given time. Hence, the audio widget takes signal strength from all microphones as input to determine who is speaking at a given moment and who has been speaking a short while before. An essential assumption to infer a participant's focus of attention is that non-speakers focus their attention on the present speaker.

Accordingly, a participant's focus of attention is estimated independently by two different sensing modalities. The Dempster-Schafer theory of evidence is used to combine the beliefs of the two sources in order to arrive at a reliable proposition.

The strength of this approach lies in its ability to improve the quality of a context obtained from various sources. It also quantifies the uncertainty associated with context aggregation.

1.4.5 Activity Context from Multiple Sensors

Mntyjärvi et al. [15] proposes a four-layered framework for recognizing a user's activity. At the lowest level there are context information sources. These sources deliver sampled raw measurements which map to physical properties. The middle layers are occupied by the context measurement and context atoms extraction unites - the raw sensor data are sampled and pre-processed in these two layers. In the case of sensor measurements, signal values are calibrated and rescaled. Pre-processed signals are used as inputs to various feature extraction algorithms in time and frequency domains, producing features to describe context information. For example, the root mean square (RMS) value of an audio signal describes the loudness of a surrounding. The first task in context extraction is to abstract raw sensor signals and compress information by using different signal processing and feature extraction algorithms. The features to be extracted are chosen according to how well they describe some parts of the real world context. Extracted features are called context atoms since they contain the smallest amount of context information. The upper layer is occupied by the context information fusion unit, which manipulates the context atoms to produce higher-level contexts that represents a real-world event.

An implementation of the context information unit employs k-means clustering and minimum-variance segmentation algorithms. Sensor data are logged from a self-contained device that encloses an array of sensors comprising three accelerometer sensors, illumination sensors, humidity sensors, thermometers, skin conductivity sensors and a microphone.

The higher-level contexts recognized include various user's activities such as running, walking and climbing a flight of stairs; contexts related to a mobile device includes whether it is being held in a hand or being placed on a table; and so on.

1.4.6 IBadge

The iBadge [16] wearable system monitors the social and individual activities of children in a kindergarten. It incorporates sensing, processing, communication and actuating units. The sensing unit includes a magnetic sensor, a dual-axis accelerometer, a temperature sensor, a humidity sensor, a pressure sensor and a light sensor. It also includes a ultrasound transceiver

and a RF transceiver for position and distance estimations. The processing unit includes speech and sensor data processing. A server side application assists a teacher by receiving and processing location, orientation, ambient and audio contexts from the iBadge to determine the social and learning status of a child. The location and orientation contexts are used to determine whether a child is isolated or associates with other children while the audio context is used to determine whether a child is sociable or aggressive.

1.4.7 Mediacup

Mediacup [17] is an ordinary coffee mug in which a programmable hardware for sensing, processing and communicating context is embedded. The hardware is a circular board designed to fit into the base of the cup and incorporates a processor subsystem, a sensing (accelerometer and temperature sensors) subsystem and a wireless transceiver. The mug continuously monitors its state by aggregating data from the two sensors, producing a higher-level context and communicating the result to a remote application via a wireless link. Heuristic-based rules are employed to reason about movement related contexts. The various propositions include whether a cup is stationary or not; whether someone is drinking from it or playing with it; or whether it is being carried around. The higher-level contexts related to temperature include whether a cup is freshly filled with coffee or whether a coffee is cooling off.

1.5 Context-Aware, Self-Managing Systems

So far, context-aware computing and self-managing systems are emerging independently. The application domains for which they are studied are different as well. The two approaches have several features in common. For example, both approaches aim at reducing human involvement: while context-aware computing aims at reducing the amount of explicit input a user should provide to computing systems, autonomous computing (self-management) aims at reducing the operational and maintenance cost of a system.

If one takes the conceptual framework of Kephart and Chess [3] as a reference framework of self-managing system, the sensing, actuating and analysis component are typical components of context-aware computing. Much work has been done by the research community of context-aware computing to support context acquisition, context modeling, context representation, context reasoning and context management. Researchers of autonomous computing can benefit a great deal by considering the usefulness of this work to develop self-managing systems.

1.6 Organization of the Book

The aim of this book is twofold:

1. To enable researchers of context-aware computing to identify potential applications in the area of autonomous computing; and

2. To support researchers of autonomous computing in defining, modeling and capturing dynamic aspects of self-managing systems.

The Merriam-Webster's English Dictionary defines a system as *a regularly interacting or interdependent group of items forming a unified whole.* We adopt this definition to define a *system* in the context of this book. Therefore, we use the word system to refer to a composition of software and hardware components which work together as a whole to accomplish a specific task on behalf of a user. In this regard a system can be a device, an application, a middleware, or a network.

The book is organized as follows: Chapter 2 provides a context-aware applications that assists nurses in hospitals to efficiently and safely administer medicaments. Chapter 3 provides a detail account of service discovery approaches and their place in self-managing systems. Chapter 4 discusses how heterogeneous context sources and their content can be managed in a distributed manner. It discusses also the usefulness of ontology as a context representation and exchanging tool. Chapter 5 discusses content negotiation in web environments.

The remaining 4 chapters focus on self-managing networks. Chapter 6 presents a higher-level vision for future self-managing networks. Chapter 7 presents in detailed the use of context-aware computing and policy-based approach to build self-managing networks. The remaining 2 chapters, namely, chapter 8 and 9, focus on two aspects of self-managing networks, i.e., self-protection and self-healing.

References

[1] A. Fox and D. Patterson. Self-repairing computers. *Scientific America*, 228(6), 2003.

[2] Y. Diao. Self-managing systems: A control theory foundation. In *The 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, 2005.

[3] J.O. Kephart and D.M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[4] G. Tesauro. Reinforcement learning in autonomic computing. *IEEE Internet Computing*, 11(1), 2007.

[5] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployment. In *The 11th Annual International Conference on Mobile Computing and Networking*, 2005.

[6] W. Dargie and T. Springer. Integrating facts and beliefs to model and reason about context. In *In Proceedings of the 7th IFIP International Conference on Distributed Applications and Interoperable Systems*, 2007.

[7] W. Dargie and T. Hamann. A distributed architecture for reasoning about a higher-level context. In *In Proceedings of the 2nd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2006)*, 2006.

[8] W. Dargie. *Architecture for computing context in mobile devices.* PhD thesis, Technical University of Dresden, 2006.

[9] A.K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1), 2001.

[10] A.J. Eronen, V.T. Peltonen, J.T. Tuomi, A.P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi. Audio-based context recognition. *IEEE Transaction on Audio, Speech, and Language Processing*, 14(1), 2006.

[11] P. Korpipää, M. Koskinen, J. Peltola, S.-M. Mäkelä, and T. Seppänen. Bayesian approach to sensor-based context awareness. *Personal Ubiquitous Computing*, 7(2):113–124, 2003.

[12] N. Moenne-Loccoz, F. Bremond, and M. Thonnat. Recurrent bayesian network for the recognition of human behaviours from video. In *In Proceedings of the 3rd International Conference on Computer Vision Systems*, 2003.

[13] H. Wu. Sensor data fusion for context-aware computing using Dempster-Schafer theory, 2003.

[14] A. Dey. *Providing architectural support for building context-aware applications.* PhD thesis, Georgia Institute of Technology, 2000.

[15] J. Mäntyjärvi, J. Himberg, and P. Huuskonen. Collaborative context recognition for handheld devices. In *In Proceedings of the 1st IEEE international Conference on Pervasive Computing and Communications*, 2003.

[16] A. Chen, R. Muntz, S. Yuen, I. Locher, S. Park, and M. Srivastava. A support infrastructure for the smart kindergarten. *IEEE Pervasive Computing*, 1(2):49–57, 2002.

[17] H.-W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mob. Netw. Appl*, 7(5):341–351, 2002.

Chapter 4

Managing Distributed and Heterogeneous Context for Ambient Intelligence

José Viterbo, Markus Endler, Karin Breitman

Departamento de Informática, Pontifícia Universidade Católica (PUC-RJ), 22453-900 Rio de Janeiro, Brazil

Laurent Mazuel, Yasmine Charif, Nicolas Sabouret, Amal El Fallah Seghrouchni, and Jean-Pierre Briot

Laboratoire d'Informatique de Paris 6 (LIP6), Université Paris 6 - CNRS, 75015 Paris, France

Abstract

Practical realization of applications for Ambient Intelligence (AmI) poses several challenges to software developers, many of them related to heterogeneity, dynamism (i.e., mobility) and decentralization. In this chapter, we focus on approaches for decentralized context reasoning and for semantic mediation, since we understand that these are some of the main challenges for enabling interactions among heterogeneous and context-aware entities in open and dynamic AmI environments.

4.1 Introduction

The *anywhere/any time* paradigm is becoming the new challenge to the conception, design and release of the next generation of information systems. New technologies, like Wi-Fi networks and 3rd generation mobile phones,

are offering the infrastructure to conceive information systems as ubiquitous, that is, systems that are accessible from anywhere, at any time, and with (almost) any electronic device. However, the use of such ubiquitous access to information systems requires new conceptualizations, models, methodologies and support technologies to fully explore its potential.

In this context, mobility introduces new accessibility scenarios and increases complexity. New issues, such as how to enable users to retain their ability to cooperate while located in different workplaces, the role of context and location in determining cooperation, the support for *ad hoc* cooperation in situations where the fixed network infrastructure is absent or cannot be used, are beginning to arise. The approaches and technologies for supporting these new ways of working are still under investigation. Nevertheless, a particularly interesting trend explores the Ambient Intelligence paradigm, a multidisciplinary approach that aims at the integration of innovative technologies that support user activities through specific services of the environment, which are provisioned with minimal user intervention. Essentially, an Ambient Intelligence system should be aware of the presence of a person, perceive the needs of this person and be able to adapt to the needs of the users in a relaxed and unobtrusive manner [20].

Ambient Intelligence (in the following, abbreviated as AmI) requires new environments for software development and deployment, where large quantities of different devices and sensors need to be integrated, building a programmable and auto-configurable infrastructure. Several projects, e.g., Gaia, CoBrA, CHIL, etc., have developed prototypes of such environments, but usually with focus only at specific use cases, user tasks or application domains. Hence, most researchers have come up with pragmatic, problem-specific solutions, which are difficult to generalize and port to other applications. However, we believe that in a few years, nearly every public and private space will be equipped with sensors and smart appliances that are able to automatically adapt to the preferences and demands of the local user(s) and as such provide special context-specific services to them. Such systems will be open, i.e., these spaces will potentially serve any user with a communication device (e.g., a smartphone with powerful computing and multimedia capabilities), which will be the unique digital interface of the user with the ambient services and with the devices of other users. Thus, openness entails that both software agents responsible for user devices (e.g., agents for assisting the user), and agents responsible for smart spaces (e.g., agents that control the devices of a room according to its current use), must be prepared to interact with an *a priori* unknown set of other software entities.

In this chapter, we present existing technologies and current proposals toward the integration of heterogeneous entities within an Ambient Intelligence system. Practical realization of applications for AmI poses several challenges to software developers, many of them related to heterogeneity, dynamism (i.e., mobility) and decentralization. We make no claim of a complete or exclusive treatment of the subject. In fact, there are also several other related chal-

lenges [55], for example, identification of user intent, knowledge acquisition, negotiation, etc. But since these issues are a complex subject on their own right, in this work we will discuss only challenges related to context reasoning, distribution, interoperability and heterogeneity issues.

Context reasoning for AmI is very complex due to the dynamic, imprecise and ambiguous nature of context data, the need to process large volumes of data and the fact that reasoning needs to be performed in a decentralized, cooperative way among several entities of the system, e.g., entities representing spaces, devices or users. Decentralization takes the form of physical distribution of computing and sensor devices, of context providers and consumers, of entities responsible for reasoning and brokering, of applications and of users who may potentially engage into a spontaneous collaboration.

Besides, AmI spaces are intrinsically heterogeneous at several levels: At the infrastructure level, they include a wide range of appliances and gadgets with very specific data and control access protocols, and which are typically interconnected through different kinds of (wireless) networks with specific protocols and QoS parameters. Also the providers and the types of context information are usually very specific for each space and device, as well as their representations and models, making it difficult to achieve a common representation for different entities of the ambient context. A similar problem of heterogeneity can be identified at the level of services due to the very different kinds of ambient control functions provided, combined with the lack of standardized interfaces for service access. Finally, heterogeneity problems are found also at the level of knowledge representation and modeling, where systems may employ very different kinds of knowledge bases, descriptions and reasoning techniques. Hence, even if two elements are conscious of the same concrete fact, there is the problem of alignment of their knowledge representations.

In this chapter, we will focus mainly on decentralized context reasoning and on semantic mediation, since we understand that these are some of the main challenges for enabling interactions among heterogeneous and context-aware entities in open and dynamic environments of Ambient Intelligence. In the following subsection we describe a simple scenario, which highlights several problems related to AmI, such as location-specific context-awareness, ontology based distributed reasoning, heterogeneous knowledge basis and semantic mediation.

4.1.1 Scenario

Silva is a Brazilian professor and researcher who works at PUC-Rio. He is visiting LIP6 with several other researchers. Their purpose is to have joint workshops related to a collaboration project. Silva carries with him his smartphone and his notebook, both executing the Campus middleware services dedicated to collecting and interpreting context information and for collective reasoning with other ambient services and applications. The devices also host some context-aware applications that support the platform's

self-configuration to adapt to different situations, according to user's preferences and environment conditions.

When Silva arrives at LIP6, his Wi-Fi and GPS enabled smartphone (SMP-1) connects to the network, and using the current GPS data, queries a location service to find out that its user (Silva) is at LIP6. It then determines that this university is a partner institution of PUC-Rio; obtains the IP address of the Ambient management service at LIP6 and registers with it, indicating the user's identity and preferences.

The Ambient management service registers SMP-1 and determines that it belongs to Silva, a visiting professor from PUC-Rio. The system verifies that Silva is involved with the collaboration project and sets a workspace for him, communicating with a service running on Silva's notebook (NTB-1) to configure it to grant access to the proper network directories and services. This system also informs other project members at LIP6 about Silva's arrival.

A personal agenda application running on SMP-1 contacts the context infrastructure to be notified about the beginning of each event involving the whole project team, based on the project schedule and the location. Another application on SMP-1, the Configuration manager, requests to be notified whenever Silva is in a room in which an *activity* has started, so that it may set the smartphone to vibe-mode, and as soon as the activity ends, switch it back to the ring mode.

Notice that when this application interacts with the Ambient's local context provider, there could be a semantic mismatch between the terms "activity," used in the device's ontology, and the terms "meeting" or "class," used in the Ambient ontology. Due to this semantic mismatch, Silva's application would not get the expected response from the Ambient Service and would issue a request for semantic mapping from a mediation service, which would try to identify equivalence or subsumption among the concepts and adjust the local ontology to reflect this new classification. Hence, we identify that the main requirements of AmI are context-specific reasoning capabilities (i.e., to enable the spaces and the interacting computing entities to "understand what is going on") and the ability to adapt services/behaviors to the current situation and user preferences. As the AmI environment is an open system, reasoning is inherently distributed.

Due to the intrinsic characteristics of Ambient Intelligence systems, ontological and distributed context-reasoning using multi-agent systems seems to be the most suitable development paradigm (i.e., each agent interacts with other agents to reinforce and complement its own knowledge about the context). However, the main problem with distributed reasoning is that heterogeneous knowledge bases and models have to be mapped (i.e., aligned, mediated), which leads to the problem of identifying and resolving semantic mismatch of knowledge representations.

4.1.2 Outline

The rest of this chapter is organized as follows: next section presents several fundamental concepts for AmI which are dealt with in this chapter. Section 4.3 discusses the related work on context awareness for AmI. In Section 4.4 we review the main approaches to deal with the ontology alignment problem. Section 4.5 presents the Campus approach for dealing with context and semantic heterogeneity in AmI. Section 4.6 concludes the chapter.

4.2 Fundamental Concepts

4.2.1 Ambient Intelligence

Ambient Intelligence (AmI), i.e., "intelligent" pervasive computing, builds on three recent key technologies [2]: Ubiquitous Computing, Ubiquitous Communication and Intelligent User Interfaces. Ubiquitous Computing is the integration of microprocessors into everyday objects like furniture, clothing, white goods, toys, even paint. Ubiquitous Communication enables these objects to communicate with each other and the user by means of *ad hoc* wireless networking. Intelligent User Interfaces enable the inhabitants of an AmI environment to control and interact with the environment in a natural (voice, gesture) and personalized way (preferences, context).

AmI aims at making use of those entities in order to provide users with an environment, which offers services when and if needed. One great challenge of such environments is how to adequately address the heterogeneity and dynamic nature of users, services and devices. Key issues of the development of AmI are context-awareness and reasoning and how to identify and activate the appropriate service within a continuously changing multitude of services [39]. The ultimate goal is to make the ambient services more *intelligent* and adaptive to the specific needs of their users.

4.2.2 Context Awareness

Context awareness is the ability of a system to sense the current environment and autonomously perform appropriate adaptations in regard to its optimal operation, general behavior and user interaction. When a user enters a new context, it is desirable that the applications on his devices be able to adapt to the new situation, and the environment be able to adapt its services to the presence of the new user.

There exist several definitions for context and context-awareness, but one of the most referenced one can be found in [19]: "*Any information which can be used to characterize the situation of an entity. An entity is a person,*

a place or an object which is considered relevant for the interaction between a user and an application, including the user and the application." In an attempt to classify context, Chen and Kotz [16] identified four basic types of context: computational context (i.e., state of resources at the device and of the network), user context (i.e., persons, places and objects), physical context (e.g., luminosity, noise, temperature) and temporal context (e.g., hour, day, period of the year). Abowd et al. [1] proposed the notions of *primary context* (localization, identity, activity and time) and of *secondary context*, where the latter one can be deduced from the former one and may be used for making adaptation decisions at a higher level of abstraction.

Conceptually, context provisioning can be organized in three layers [33]: data acquisition and distribution, interpretation and utilization. Before raw context data acquired from sensors and devices can be utilized, it must be interpreted and evaluated with respect to its accuracy, stability and reliability. The interpretation layer may also combine context data from different sources to enhance its reliability or completeness. For applications to be able to understand, describe and manage context-aware adaptations, it is necessary to have a context model, which can be defined at the application or the middleware layer. Strang and Linnhoff-Popien [62] identified and compared six types of context models: attribute-value pairs, schema-based models, graphic models, logic-based models, object-oriented models and ontology-based models. The author's main conclusion is that the object-oriented and the ontology-based models are the most complete and expressive ones, and hence are the most suited for modeling context for ubiquitous computing.

4.2.3 Ontology

Ontology has not only the advantage of enabling the reuse and sharing of common knowledge among several applications [58], but also of allowing the use of logic reasoning mechanisms to deduce high-level contextual information [68]. Therefore it has been widely adopted over other conceptual models, such as taxonomy, relational database schema and OO software models, for representing context information in ubiquitous systems.

A taxonomy is a set of terms arranged in a generalization-specialization (parent-child) hierarchy because they are much more expressive [34]. A controlled vocabulary simply lists a set of terms and definitions. A taxonomy may or may not define attributes of these terms. A relational database schema defines a set of terms through classes, attributes and a limited set of relationships among those classes. An OO software model defines a set of concepts and terms through a hierarchy of classes and attributes and a broad set of binary relationships among classes. Constraints and other behavioral issues may be specified through methods on the classes (or objects).

An ontology can express all of the preceding relationships, models and diagrams as well as n-ary relations, a rich set of constraints, rules relevant to usage or related processes and other differentiators including negation and

disjunction [25]. Table 4.1 summarizes the benefits of the adoption of ontology.

Table 4.1: Benefits of adopting formal ontology to model ambient knowledge in Campus.

- Ontologies are semantically richer, i.e., have greater expression power than taxonomies, entity relationships or OO models;
- Conceptual knowledge is maintained through complex and accurate representations above and beyond hierarchical approaches;
- Ontologies are formal — OWL DL ontologies map directly to Description Logic (a dialect of first order logics);
- Formal ontologies in the OWL DL standard can be verified/classified with the aid of Inference Mechanisms, e.g., RACER and FaCT:
  - consistency checks;
  - classification;
  - new information discovery;
- OWL ontologies use a XML/RDF syntax that allows them to be automatically manipulated and understood by most resources on the Internet;
- Ontologies capture and represent finely granulated knowledge;
- Ontologies can be used to reduce ambiguity so as to provide a model over which information can be freely shared and acted upon by autonomic managers;
- Ontologies are modular, reusable and code independent — ontology driven applications are specified separately from the ontology itself. Changes to the ontology should not impact the code or vice versa;
- Ontologies can be combined with emerging rule languages, such as SWRL.

4.2.4 Context Reasoning

Reasoning is necessary in context aware systems to deal with the intrinsic imperfection and uncertainty of context data, and also to infer secondary context data. Henricksen and Indulska [28] have characterized four kinds of context imperfectness: unknown, ambiguous, imprecise and erroneous. The main tasks of reasoning are to detect possible errors, make estimates about missing values, determine the quality and validity of the context data, trans-

form context data into meaningful information and infer new, implicit context information that may be relevant for the applications. Reasoning is also fundamental for any kind of context-oriented decision-making, e.g., system adaptations according to user-provided or learned decision rules.

According to [46] reasoning for context-aware systems can be approached from four main perspectives: the low-level perspective, which includes basic tasks such as data pre-processing, data fusion and context inference, usually performed by the sensors or the middleware, the application-oriented perspective, where the application can use a wide variety of reasoning methods to process the context data, the context monitoring perspective, where the main concern is a correct and efficient update of the knowledge base as the context changes and, finally, model monitoring perspective, where the main task is to continuously evaluate and update learned context classifiers/interpreters and their models, also taking into account user feedback. Although Nurmi and Floren give an interesting perspective on context reasoning, we understand that instead of four perspectives, these are in fact complementary tasks, which should be present in every approach for reasoning in context-aware systems.

For context reasoning, several approaches have been adopted: ontological reasoning, rule-based reasoning, distributed reasoning and probabilistic reasoning [7]. Instead of presenting and comparing the general reasoning approaches, which are very well surveyed in Bikakis et al. [7], in this paper, we will focus only on ontological and distributed reasoning approaches. On the one hand, ontologies offer high expressiveness and the possibility to develop a formal context model that can be shared, reused, extended to specific domains, and on the other hand, distributed reasoning is a direct requirement that arises from the open, dynamic and heterogeneous nature of AmI.

In the next section, we review the main approaches to deal with the contextual reasoning and ontological representations for AmI. Section 4.4 will focus on ontology alignment and semantic mapping between concepts, to deal with semantic heterogeneity in AmI. Section 4.5 will present our proposition to tackle both issues within the Campus framework.

4.3 Ontological Representation and Reasoning about Context

In this section we survey several research works that deal with ontological representation and reasoning about context for Ambient Intelligence. We first present the main criteria used for comparison and a proposed taxonomy; then we present each work with respect to each criteria; and finally, we classify the systems according to our taxonomy and discuss their suitability for implementing Ambient Intelligent environments.

4.3.1 Evaluation Criteria and Taxonomy

There is much research work on middleware systems that support context modeling and ontological reasoning about context. However, as expected, each one is based on a different notion of context, uses ontologies in a different way, has specific goals and approaches for context-specific reasoning and handling heterogeneity, and is targeted at specific applications or use scenarios. In this section, we will compare the works in regard to the following criteria:

4.3.1.1 Types of Context.

Which types of context information are collected, processed and distributed by the system (e.g., system context, location, physical context, user role, preferences, etc.). This information will give an idea of the framework's usefulness, scalability and practical feasibility.

4.3.1.2 Ontologies.

Which ontologies are used, and for which purpose? What sorts of concepts and relationships are represented? Is the ontology extensible? How are context instances updated and persisted, etc.? This criterion assesses the system's expressiveness and flexibility.

4.3.1.3 Inference/Reasoning Techniques.

What kind of reasoning is supported? What sorts of higher-level context is inferred? Does the work consider uncertainty of the inferred context? This aspect determines the expressive power, reliability, completeness and preciseness of the systems reasoning, as well as its practical applicability.

4.3.1.4 Knowledge Management.

Is the knowledge base static, or do most of the facts in the knowledge require continuous updates? Does the system handle decentralized or heterogeneous knowledge bases, and if so, do they handle evolving knowledge models (ontologies)? If heterogeneity is supported, what is the basic mediation or semantic alignment technique employed and how powerful is it? The evaluation with regard to this aspect will give insight on how well the system is suited to deal with the inherently dynamic, decentralized and unpredictable nature of Ambient Intelligence.

4.3.1.5 Architecture.

Is the system based on a centralized, fully decentralized or hybrid architecture, with respect to the knowledge bases, the reasoning process and the mediation/brokerage support? By discussing this aspect, we have an idea on the system's scalability, reliability and of the implicit execution overhead related to the distributed interactions.

Although there are many possible means of classifying the context systems, we believe that the following aspects are the most relevant for assessing their suitability for developing open and heterogeneous Ambient Intelligence environments. Hence, we will use them as the basis for our taxonomy.

1. Centralized *versus* decentralized knowledge base;
2. Static *versus* dynamic (or extensible) set of context providers;
3. Main goal of context reasoning: enhance reliability of context information, derive higher-level context facts, or both;
4. Means of handling heterogeneous knowledge bases, if any.

In the following subsections we summarize and analyze the most representative middleware systems with regard to the presented criteria, and in Subsection 4.3.10, classify each system according to the proposed taxonomy.

4.3.2 Gaia

Gaia provides a generic computational environment that integrates physical spaces and their ubiquitous computing devices into a programmable computing and communication system [52]. It is similar to traditional operating systems in that it manages the tasks common to all applications built for physical spaces [50]. Each space is self-contained, but may interact with other spaces. Gaia provides core services, including events, entity presence (devices, users and services), discovery and naming. By specifying well-defined interfaces to services, applications may be built in a generic way so that they are able to run in arbitrary active spaces. Gaia uses CORBA to enable distributed computing. Gaia is a mature project. The first prototypes were implemented in 2002 and several applications for active-classrooms have already been developed.

4.3.2.1 Types of Context.

The Gaia Context Infrastructure allows applications to obtain a variety of contextual information. Various components, called Context Providers, obtain context from either sensors or other data sources. These include sensors that track people's locations, room conditions (for example, temperature and sound) and weather conditions. Context Providers allow applications to query them for context information. Some Context Providers also have an event channel to asynchronously send context events. Thus, applications can either query a Provider or listen on the event channel to get context information.

4.3.2.2 Ontologies.

Gaia's context model is based on first-order predicates. The name of the predicate indicates the type of context that is being described (e.g., location,

temperature or time), and its typed arguments describe the properties of the context. For example, if the predicate is "location," the first argument has to be a person or object, the second argument has to be a preposition or a verb like "entering," "leaving" or "in" and the third argument must be a locationID. The structures of different context predicates are specified in an ontology. Each context type corresponds to a class in the ontology, which also defines the corresponding arguments of the predicate. Moreover, Gaia uses ontologies to describe various concepts of an Ubiquitous Computing Environment, such as kinds of applications, services, devices, users, data sources and other entities. They also define all terms used in the environment and the relationships between different terms. These ontologies are written in DAML+OIL.

4.3.2.3 Inference/Reasoning Techniques.

Context Synthesizers are Gaia components that get sensed context data from various Context Providers, derive higher level or abstract context from these lower-level context data and provide these inferred contexts to applications. Whenever a Synthesizer deduces a change in the inferred context, it publishes the new information. Gaia adopts two basic inference approaches. Rule-based Synthesizers use pre-defined rules written in first order logic to infer different contexts. Each of the rules also has an associated priority, which is used to choose one rule when multiple rules are valid at the same time. However, if all the valid rules have the same priority, one of them is picked at random. Alternatively, some Synthesizers may use machine learning techniques, such as Bayesian learning and reinforcement learning, to infer high-level contexts. Past context information is used to train the learner.

4.3.2.4 Knowledge Management.

All the ontologies in Gaia are maintained by an Ontology Server. Entities contact the Ontology Server to get descriptions of other entities in the environment, information about context or definitions of various terms used in Gaia. The server also supports semantic queries to get, for instance, the classification of individuals or subsumption of concepts. The Ontology Server also provides an interface for adding new concepts to existing ontologies. This allows new types of contexts to be introduced and used in the environment at any time. The Ontology Server ensures that any new definition is logically consistent with existing definitions. Since the ontologies clearly define the structure of contextual information, different agents can exchange different types of context information easily. For example, Context Providers and Context Synthesizers can get the structure of contexts that they provide, while Context Consumers query the Ontology Server for the structure of the requested context, and then frame appropriate queries to Context Providers to get the context information they need.

4.3.2.5 Architecture.

The Gaia kernel consists of a Component Management Core that dynamically loads, unloads, transfers, creates, and removes any Gaia component or application. Each active space is self-contained but may interact with other spaces. For each space, Gaia manages its resources and services; provides location, context and event services; and stores information about it. Gaia provides a set of basic services to be used by all applications. Among them, the Space Repository stores information about all software and hardware entities in the space and lets applications browse and retrieve an entity on the basis of specific attributes. The Space Repository learns about entities entering and leaving the active space through the Presence Service, which detects and maintains soft state information about applications, services, devices and people in a active space. When the Presence Service detects that an entity is no longer available in an active space, it notifies the rest of the space that the entity left. In the context infrastructure, the Context Provider Lookup Service allows searches for different context providers. Providers advertise the set of contexts they provide in the form of a first order expression that describes the context provided. Applications can query the Lookup Service for a context provider that provides contextual information it needs.

4.3.3 CoBrA

Context Broker Architecture (CoBrA) is an infrastructure that supports agents, services and devices that interact in order to explore context information in active spaces [17, 18]. Its main component is an intelligent agent called *context broker*, which is responsible for providing a common model to represent context information, mediating the information exchanged between context providers and resource constrained context consumers, and inferring higher-level context information not directly available from sensors [17]. In addition, the context broker is capable of detecting and correcting inconsistent context data, and supports the enforcement of privacy policies defined by the users to control the sharing of their contextual information among other users. The proposed architecture is based on a central entity that was implemented as a FIPA-compliant agent using Jade.

4.3.3.1 Types of Context.

CoBrA has a context-acquisition module, which is a set of library procedures for acquiring contextual information from sensors, agents and the Web. This library includes procedures for collecting information from Smart Tag sensors (location) and environment sensors (temperature, sound, luminosity, etc.), but any other information can be added.

4.3.3.2 Ontologies.

The base ontologies used for representing context information are the CoBrA Ontology (COBRA-ONT) and SOUPA. COBRA-ONT is a set of ontologies for agents to describe contextual information and to share context knowledge. It defines concepts for representing actions, agents, devices, meetings, time and space. The SOUPA ontology, on the other hand, is a standard ontology for supporting pervasive and ubiquitous computing applications. It consists of vocabularies for expressing common concepts that are associated with person, agent, belief-desire-intention (BDI), action, policy, time, space and event, and also a set of vocabularies for supporting specialized domains of pervasive computing, such as smart spaces and peer-to-peer data management. The developer of a new system must design its specific ontology reusing some others that may be adequate.

4.3.3.3 Inference/Reasoning Techniques.

CoBrA's context reasoning is backed by the Jena rule engine, the Java Expert System Shell (JESS) and the Theorist system. The reasoning for interpreting context information uses two different rule-based systems. Jena rule-based reasoners are used for OWL ontology inferences and the JESS rule engine is used for interpreting context using domain specific rules. CoBrA supports also reasoning for maintaining a consistent context model by detecting and resolving inconsistent information, and the Theorist system is used for supporting the necessary logical inferences in that case. When a new context data is asserted into the knowledge base, the context broker first selects the type of context it attempts to infer (such as a person's location or a meeting's state). If such information is unknown, the broker decides whether it can infer this type of context using only ontology reasoning (Jena Rules). If logic inference is required, the context broker attempts to find all essential supporting facts by querying the ontology model and asserts them into the Jess engine. Before asserting the new inferred information into the knowledge base, ontology reasoners are used to infer whether the context described by the instant data is consistent with the model defined by the ontology. If not, a Theorist assumption-based reasoning is used for resolving inconsistent information.

4.3.3.4 Knowledge Management.

The system provides a centralized (and homogeneous) model of context that all devices, services, and agents in the space must share. The knowledge of the context broker is represented as RDF statements and is stored in a persistent knowledge base. To acquire contextual information, all agents must send query messages to the context broker.

4.3.3.5 Architecture.

CoBrA has a centralized architecture, where a single context broker agent should be deployed and all computing entities must be aware of this broker from the beginning. Usually, a single context broker is sufficient to support a small-scale smart space. However, being the main service provider in the space, the context broker may become the bottleneck of the system and a single point of failure. A team of context brokers can be deployed to overcome this problem, as well as to improve system robustness through redundancy.

4.3.4 Semantic Space

Semantic Space [67, 68] is a context infrastructure developed to address three key issues. First, it aims to provide an explicit representation of the raw context data that is obtained from various sources in different formats. Furthermore, it provides means for the applications to selectively access a subset of context data through expressive context queries. Finally, it provides reasoning capabilities for inferring higher-level contexts. A prototype of the context infrastructure has been developed, and a prototype context-aware application was also implemented. The application, called SituAwarePhone, adapts mobile phones to changing situations while minimizing user distraction.

4.3.4.1 Types of Context.

In Semantic Space, *context wrappers* obtain raw context information from various sources such as hardware sensors and software programs and transform them into context data. Some context wrappers work close to the hardware sensors deployed in the prototypical smart space, gathering information such as user's location, environmental temperature, noise, and light, status of doors (open or closed) of rooms, etc. Software-based context includes the activity of the user, based on the schedule information from Outlook Web Access; the status of different networked devices (such as voice over IP or mobile phones), the status (idle, busy, closed) of applications such as JBuilder, Microsoft Word, and RealPlayer from their CPU usage; and weather information obtained by periodically querying a weather web service.

4.3.4.2 Ontologies.

Semantic Space uses the CONtext ONtology (CONON) for modeling context in pervasive computing environments [68]. Rather than completely modeling all sorts of context in different kinds of smart spaces, this ontology aims to be an extensible upper-level context ontology providing a set of basic concepts that are common to different environments. To characterize smart spaces, there are three classes of real-world objects (user, location and computing entity) and one class of conceptual objects (activity), which together

form the skeleton of a "contextual-rich environment." Consensus domain ontologies such as friend-of-a-friend (FOAF), RCAL Calendar and FIPA Device Ontology were also integrated into CONON to model users, activities and device contexts, respectively.

4.3.4.3 Inference/Reasoning Techniques.

Two context reasoners are available, a description logic based reasoner and a first-order logic based situation reasoner, both implemented using Jena Semantic Web Toolkit to perform forward reasoning over the knowledge base. The description logic based reasoner was built to carry out ontology reasoning. The more flexible first-order logic based situation reasoner deduces a wide range of higher-level, conceptual context from relevant low-level context, such as user's activity. Semantic Space requires developers to write rules describing higher-level context information for each particular application based on its needs.

4.3.4.4 Knowledge Management.

In each smart space resides a *Context Knowledge Base*, which provides persistent context knowledge storage. It stores the extended context ontology for a particular space and the context data provided by users or gathered from context wrappers. The *Context Aggregator* is responsible for discovering context wrappers, gathering context data from them, and then asserting the gathered data into the context knowledge base. It updates the knowledge base whenever a context event occurs. The scope of contexts that the knowledge base manages may change depending on the availability of wrappers. When a context wrapper joins the smart space, the context aggregator adds the provided contexts to the knowledge base, and when the wrapper leaves, the aggregator deletes the contexts it supplied to avoid stale information.

4.3.4.5 Architecture.

The architecture is centralized around a *Context Aggregator* and a *Context Knowledge Base*. Developers can add new wrappers to expand the scope of contexts in a smart space or remove existing wrappers when the contexts it provides are no longer needed.

4.3.5 CHIL

The middleware infrastructure developed in the CHIL (Computers in the Human Interaction Loop) Project [60] provides mechanisms for service access, context modeling, control of sensors and actuators, directory services for infrastructure elements and services, as well as fault tolerance mechanisms. In general, this middleware infrastructure allows developers to focus on the service logic, rather than on the details of context processing and utility services,

also providing a framework with several components that can be reused across different ubiquitous computing services. Mechanisms for modeling composite contextual information and describing networks of situation states are also available. The middleware has been implemented as a distributed multi-agent system where the agents are augmented with fault tolerance capabilities using the agent's capacity to migrate between hosts.

4.3.5.1 Types of Context.

The infrastructure can exploit numerous sensors for context acquisition, and new sensors can be plugged into the framework to provide information that may be used to compound derived contextual information or define situations that will trigger system's responses. Context information is obtained from sensors by software agents and made accessible to other agents of the system through the *Knowledge Base Agent*. Monitoring and control of sensors is performed through special *Proxy agents* that represent the sensors in the world of agents. Each proxy agent exposes a *universal virtualized interface* to the agent framework. A sensor specific driver is required to adapt the universal interface commands to the low-level capabilities of each particular sensor. This low-level driver is based on the control API offered by the sensor. Actually, three concrete proxy agents were implemented: one generic, one for microphones and one for cameras.

4.3.5.2 Ontologies.

The CHIL ontology aims to establish a general-purpose core vocabulary for the various concepts comprising a multi-sensor smart space and the context-aware applications associated [47]. It was modularized to allow different parts to be used in different contexts and applications. Separated namespaces are used so that developers may safely introduce new concepts locally in their module's name-space without interfering with other modules. Assuming that other modules use similar concepts that should be merged, the core module may provide a merged version of the concept. To globally put together all the modules, the ontology consists of a main OWL file, which imports all modules. Developers interested only in a subset of modules can define a main OWL file of their own that imports only the modules of interest. The main component is the core module *chil-core*, which introduces concepts of perceivable entities such as, for example, *Person*, *MeetingRoom*, *Table* or *Whiteboard*, as well as perceivable roles of such entities, such as the *Location* of a *Person* or the *ActivityLevel* of a *MeetingRoom*.

4.3.5.3 Inference/Reasoning Techniques.

The approach adopted by CHIL to infer high-level contexts is based on the notion of *networks of situation states*. According to this approach a situation is considered as a state description of the environment expressed in terms of

entities and their properties. Changes in individual or relative properties of specified entities correspond to events that signal a change in the situation. The concept of *role* serves as a variable for the entities to which the relations are applied, thus allowing an equivalent set of situations to have the same representation. A role is played by an entity that can pass an acceptance test for the role, in which case, it is said that the entity can play or adopt the role for that situation. For example, in the scope of a meeting involving short presentations, at any instant, one person plays the role of "presenter," while the other persons play the role of "attendees." Dynamically assigning the role of "presenter" to a person makes it possible to select sensors to acquire images and sound of the current speaker. Detecting a change in some role allows the system to reconfigure the video and audio acquisition systems.

4.3.5.4 Knowledge Management.

The knowledge base was developed as a server accessible both locally and remotely through a unique interface. The server remote interface is programming language independent, so that client components may be written in a variety of programming languages. The knowledge base server API is tailored to OWL.

4.3.5.5 Architecture.

This architecture is centralized around some core agents, which are independent of the service and smart room installation. They provide the communication mechanism for the distributed entities of the system, control of the sensing infrastructure, and allow service providers to register their service logic into the framework. Besides, some agents that provide basic services, such as the ability to track composite situations, the control of sensors, access to the knowledge base, are tightly coupled with the installed infrastructure of each smart room.

4.3.6 SAMOA

SAMOA framework [8] supports the creation of semantic context-aware social networks, which consist of logical abstractions that represent groups of mobile users who are in physical proximity and share common affinities, attitudes and social interests. In particular, SAMOA lets mobile users create *roaming social networks* that, following user movements, at each instant reflect all nearby encounters of interest. Mobile users interested in creating social networks are called *managers*. They are responsible for defining the scope (i.e., radius) of discovery of their social network and the selection criteria. Other users located within the discovery boundaries are those *eligible* to become members of the *manager*'s social network. But only the users that are selected by the *manager* become *affiliated* with that social network.

4.3.6.1 Types of Context.

To support the creation of social networks in ubiquitous environments, SAMOA relies on geographical context information, e.g., a user's location and reciprocal proximity, user attributes and social preferences, and place descriptions. Users' location and proximity are determined either by the network cell (or the WiFi access point) the user is currently attached to, or by the number of network hops between users in an *ad hoc* network. The middleware provides graphic tools for specifying profiles of users and places.

4.3.6.2 Ontologies.

SAMOA models and represents context data in terms of semantic metadata. Places and users are the entities in the system. They are associated with profiles describing their characteristics. A *place profile* has an identification and an activity parts. The former includes a unique identifier, a name and a description of the physical place, and the latter includes all of the social activities that characterize the place, and which sorts of information members located in that place are expected to share. The *user profile* consists of an identification and a preference part. The identification part provides user naming information and describes user properties, such as age, gender and education, and the preference part defines the activities the user is interested in and, for each of these activities, the user's specific preferences. Besides *place* and *user profiles*, managers also have a *discovery profile* associated with each place, defining which preferences *user profiles* must match to join the manager's social network at that place. Preferences in *discovery profile* include desired client attributes for each activity. While activities and preferences in the *place profile* and in the manager's *discovery profile* are represented as classes, activities and preferences in a *user profile* are defined as instances.

4.3.6.3 Inference/Reasoning Techniques.

SAMOA exploits two semantic matching algorithms for analyzing profiles and inferring potential semantic compatibility among users. The first algorithm operates on *user* and *place profiles* to identify a first set of eligible members located within an area of interest around a place. Only those users whose profiles have activities that are semantically related to that of the *place profile* activities become *eligible members*. The second matching algorithm selects among the previously selected *eligible members* only those users whose attributes semantically match the preferences included in the manager's *discovery profile* for that particular place. Moreover, the matching algorithms perform also ontology reasoning to identify if the activity or preference in the *user profile* is an instance of a more generic activity or preference class, or an instance of a more specialized activity or preference class in the manager's *place* or *discovery profile*. SAMOA relies on the Pellet DL reasoner [59] for implementing both matching algorithms.

4.3.6.4 Knowledge Management.

No centralized database is kept in SAMOA. Place and discovery profiles are maintained and analyzed separately. The user's mobile devices keep their own user profiles. Some users that may become managers of a social-network keep on their devices discovery profiles associated with each place. Stationary devices may keep place profiles for each place. The manager communicates only the place profile to co-located users, preserving the privacy of its discovery profile. Similarly, users return their user profiles only to managers that provided places with activities of interest. In addition, keeping place and discovery profiles separate lets SAMOA distribute the overhead of the social-network extraction among all users, since the semantic analysis of the *place profile* is performed on user's devices, and semantic matching between *discovery* and *user profiles* is performed on manager devices.

4.3.6.5 Architecture.

The SAMOA middleware has totally distributed architecture organized in two logical layers: *the basic service layer* and the *social-network management layer*. The basic service layer provides facilities for naming, detection of co-located users and device communication. In this layer, the location/proximity manager (L/PM) lets SAMOA entities advertise their online availability by periodical broadcasts of advertisement messages. L/PM senses incoming advertisements and builds a table of "discovered" co-located users. The social-network management layer includes facilities for semantic-based social network extraction and management. In this layer, the *place-dependent social-network manager* (PSNM) creates and maintains a table that includes all members of the manager's social-network that are currently co-located with the manager. The *global social-network manager* (GSNM) keeps a record (in a dedicated table) of all place-dependent social networks previously formed at the visited places, i.e., the manager's global social network. In addition, the table stores the *place profile* and the *discovery profile* of the manager, which guided the selection of each member.

4.3.7 CAMUS

Context-Aware Middleware for URC (Ubiquitous Robotic Companion) System (CAMUS) is a context-aware infrastructure for the development and execution of a network-based intelligent robot system [36]. It was designed to overcome limitations of the ubiquity, context-awareness and intelligence that existing mobile service robots have. CAMUS gathers context information from different sensors and delivers appropriate context information to different applications. Moreover, CAMUS provides context-aware autonomous service agents that are capable of adapting themselves to different situations.

4.3.7.1 Types of Context.

In CAMUS, a *sensor framework* processes input data from various sources such as physical sensors, applications and user commands and transfers them to the *Context Manager* through an *Event System*. The Context Manager manages context information collected from the Sensor Framework. When context information in the environment is changed, the Context Manager transfers events to the Event System. The context represented includes the user context, environment context and computing device context. User context includes user profile, user's task information, user preference, etc. Environment context includes hierarchical location information, time, etc. Computing device context includes information about available sensors and actuators.

4.3.7.2 Ontologies.

The context model in CAMUS is represented as a four-layered space, where each layer has a different abstraction level. In the *common ontology layer* are modeled the ontology concepts that are commonly used in various applications. The common ontology provides the high-level knowledge description to context-aware applications. Generally, highly abstracted knowledge can be easily reused by various applications. The *domain ontology layer* comes below the *common ontology layer*. It provides the domain specific knowledge to context-aware applications. This layer is composed of the *infrastructure domain ontology* and a set of *specific domain ontologies* for the application. The *infrastructure domain ontology* is the schema of the context model that is represented and managed in the context-aware system. The *specific domain ontology* is about specific services, for example, a presentation service. The *domain ontology layer* provides the schema to the layer below, the *instance layer*, where instances of the ontology concepts are represented. Above the *common ontology layer* there is the *shared vocabulary layer*, where is defined a set of shared vocabulary (and their semantics) used in the *common ontology layer*.

4.3.7.3 Inference/Reasoning Techniques.

CAMUS context reasoning engine includes many different reasoners, which handle the facts present in the repository and produce higher-level contexts [26]. The reasoning service is used by some context mapping services and context aggregators. They invoke the reasoners through a fixed API, providing the reasoners with context data. All new inferred facts will be inserted into that context data for later queries. The use of a fixed interface for all kinds of reasoning engines makes it possible to add and handle different reasoners. Multiple reasoning mechanisms are available. Reasoners can infer high-level contexts using rules written in different types of logic like first order logic, temporal logic, description logic (DL), higher order logic, fuzzy logic, etc. In-

stead they can also use various machine learning techniques, such as Bayesian learning, neural networks, reinforcement learning, etc. The middleware defines wrappers for each reasoner type. Besides, a Racer server [27] provides ontology reasoning to infer subsumption relationships, instance relationships and consistency of context knowledge base.

4.3.7.4 Knowledge Management.

The application context model is stored in the CAMUS context storage through a Knowledge Base adaptor. Applications can refer and change application context through Jena APIs. Moreover, application context models can be updated and changed through the Jena rule engine and OWL reasoner depending on application-specific inference rules and subsumption reasoning.

4.3.7.5 Architecture.

CAMUS has a centralized architecture composed of three parts: Main Server, Service Agent Manager and Service Agents. The Main Server manages context information delivered from Service Agent Managers. It generates and disseminates appropriate events to applications according to the context changes. The Service Agent Manager provides the container where Service Agents are executed. A Service Agent is a software module that acts as a proxy to connect various external sensors and smart devices to CAMUS. It delivers information of sensors in environment to the Main Server, receives control commands from the Main Server, controls devices in the environment and conducts applications. The entities in the system communicate using PLANET, a lightweight and fault-tolerant communication mechanism which also supports the disconnected operations and asynchronous operations.

4.3.8 OWL-SF

The distributed semantic service framework, OWL-SF [44], supports the design of ubiquitous context-aware systems considering both the distributed nature of context information and the heterogeneity of devices that provide services and deliver context. It uses OWL to represent high-level context information in a semantically well-founded form. Devices, sensors and other environmental entities are encapsulated and connected to the upper context ontology using OMG's Super Distributed Objects technology [54] and communicate using the Representational State Transfer protocol [23]. Integrated reasoning facilities perform the automatic verification of the consistency of the provided service specifications and the represented context information, so that the system can detect and rule out faulty service descriptions and can provide reliable situation interpretation. A prototype of the system has been implemented and tested.

4.3.8.1 Types of Context.

OWL-SF uses Super Distributed Objects (SDOs) [54] to encapsulate context providers, which may be sensors, devices, user's interfaces (GUIs) or services.

4.3.8.2 Ontologies.

Each SDO that encapsulates context providers and service-providing devices is an OWL-SDO. This OWL extension adds new methods to a standard SDO which allow accessing the current state of an object as an OWL description. Each functional entity implemented as OWL-SDO has to be described using its own ontology containing terminological knowledge that enables the automatic classification of the object into appropriate service categories. The state of an object stores context values and is represented by an instance of a class in the ontology.

4.3.8.3 Inference/Reasoning Techniques.

Deduction servers (DSs) are specific OWL-SDO with an RDF inference mechanism and an OWL-DL reasoner. The rule-based reasoning process is provided by the RDF inference component and the deduced facts are used to trigger events to other SDOs and to process service calls. A subscription notification mechanism is used to monitor the SDO parameters to generate notifications whenever an observed parameter changes, triggering the deduction process to update the global ontology model accordingly. The RDF inference component is connected to the OWL-DL reasoner, which is responsible for classification and answering OWL-DL queries. The Racer system [27] is used as an OWL-DL reasoner.

4.3.8.4 Knowledge Management.

Besides providing deductive support, DSs are responsible for collecting the status of SDOs, published in the OWL format, and building an integrated OWL description accessible to the reasoning process. The semantic representation of each SDO is added to the internal database of the DS. This semantic representation consists of a set of instances augmented with rules. Facts deduced from rules are only used to change parameters and to call services but never modify the knowledge base.

4.3.8.5 Architecture.

OWL-SF is a distributed system and its functional architecture integrates two basic building blocks: OWL-SDOs and DSs. A system may be composed of multiple components of both types which can be added and removed dynamically at runtime. DSs use the SDO discovery and announcement implementation to become aware of new SDOs in the environment. Whenever

a new SDO is discovered, its semantic representation is added to the internal database.

4.3.9 DRAGO

Distributed Reasoning Architecture for a Galaxy of Ontologies (DRAGO) is a distributed reasoning system, implemented as a peer-to-peer architecture in which every peer registers a set of ontologies and mappings, and the reasoning is implemented using local reasoning in the registered ontologies and by coordinating with other peers when local ontologies are semantically connected with the ontologies registered in other peers [57]. DRAGO is implemented to operate over HTTP and access ontologies and mappings published on the web.

4.3.9.1 Types of Context.

DRAGO does not implement a context layer, i.e., it does not have any service for context collection, storing or distribution.

4.3.9.2 Ontologies.

DRAGO considers a web of ontologies distributed among a peer-to-peer network. Each peer may contain a set of different ontologies describing specific domains of interest (for example, ontologies describing different activities of users in a university). These ontologies may differ from a subjective perspective and level of granularity. In each peer there are also semantic mappings defining semantic relations between entities belonging to two different ontologies. These semantic mappings are described using C-OWL [9]. To register an ontology at a peer the users specify a logical identifier for it, i.e., a URI, and inform a physical location of the ontology in the web. Besides that, it is possible to assign semantic mappings to the ontology, providing, in the same manner, the location of the mappings on the web. New peers may be added dynamically to the system, providing new ontologies and semantic mappings.

4.3.9.3 Inference/Reasoning Techniques.

The reasoning process may compare concepts in different ontologies to check concept satisfiability, determining if a concept subsumes the other (i.e., the latter is less general than the former), based on the semantic mappings relating both ontologies. In a set of ontologies interconnected with semantic mappings, the inference of concept subsumption in one ontology (or between ontologies) may depend also on other ontologies related to the previous ones through those mappings. Every peer registers a set of ontologies and mappings, and provides reasoning services for ontologies with registered mappings. Each peer may also request reasoning services from other peers when their local ontologies are semantically connected (through a mapping) with the

ontologies registered at the other peer. The reasoning with multiple ontologies is performed by a combination of local reasoning operations, internally executed in each peer for each distinct ontology. A distributed tableau algorithm is adopted for checking concept satisfiability in a set of interconnected ontologies by combining local (standard) tableaux procedures that check satisfiability inside the single ontology. Due to the limitations of the distributed tableau algorithm, for a semantic mapping DRAGO supports three types of rules connecting atomic concepts in two different ontologies: *is equivalent*, *is subsumed* and *subsumes*. A Distributed Reasoner was implemented as an extension to the open source OWL reasoner Pellet [59].

4.3.9.4 Knowledge Management.

As each peer registers sets of heterogeneous ontologies and mappings, the knowledge base is totally distributed. When users or applications want to perform reasoning with a registered ontology they refer to the corresponding peer and invoke its reasoning services giving the URI to which the ontology was bound.

4.3.9.5 Architecture.

DRAGO aggregates a web of ontologies distributed amongst a peer-to-peer network in which each participant is called a *DRAGO Reasoning Peer* (DRP). A DRP is the basic element of the system and is responsible for providing reasoning services for ontologies using the semantic mappings registered. As these mappings establish a correlation between the local ontology and ontologies assigned to other DRPs, a DRP may also request reasoning services of other DRPs as part of a distributed reasoning task. A DRP has two interfaces that can be invoked by users or applications. A *Registration Service Interface* is available for creating/modifying/deleting registrations of ontologies and mappings assigned to them. A *Reasoning Service Interface* enables requests of reasoning services for registered ontologies. Among the reasoning services DRAGO allows to check for ontology consistency, build classifications, verify concepts satisfiability and check entailment.

4.3.10 Conclusion

In this section, we classify the surveyed systems according to our taxonomy (cf. Table 4.2), and discuss their suitability for implementing context-oriented ontological reasoning for Ambient Intelligence. The eight systems we presented not only have different features, but some of them have been developed with different purposes. Gaia, CoBrA, Semantic Spaces and CHIL offer middleware infrastructure for Smart Spaces; SAMOA is designed specifically to support applications that deal with social networks in ubiquitous environments; CAMUS provides an infrastructure for the development and execution of a network-based intelligent robot system; OWL-SF supports the

design of generic distributed context-aware systems; finally, DRAGO provides reasoning about heterogeneous ontologies.

Table 4.2: Classification of middleware systems for context-oriented ontological reasoning.

	Set of context providers	Ontology update	Knowledge base	Main goal of reasoning	Handling of heterogeneous knowledge bases
Gaia	Dynamic	Dynamic	Distributed	Derive higher-level facts	No
CoBrA	Dynamic	Static	Centralized	Both	No
Semantic Spaces	Dynamic	Static	Centralized	Derive higher-level facts	No
CHIL	Static	Static	Centralized	Derive higher-level facts	No
SAMOA	Static	Dynamic	Distributed	Derive higher-level facts	No
CAMUS	Dynamic	Dynamic	Centralized	Derive higher-level facts	Shared vocabulary layer
OWL-SF	Dynamic	Static	Distributed	Both	No
DRAGO	—	Dynamic	Distributed	Classification	Semantic Mapping

Comparing the four frameworks for Smart Spaces, it may be said that Gaia is the only one that supports distributed knowledge bases and is the one that best deals with dynamic scenarios, allowing context providers to be added or removed dynamically and ontologies to be dynamically modified with regard to types of context and their properties. Despite not being tailored specifically for smart spaces, OWL-SF may be used for implementing such

systems, as its singular characteristic is its support for distributed inference. Similar to Gaia, OWL-SF considers a distributed knowledge base. Hence, in each space, the aggregated context information will depend on the available providers, avoiding communication bottlenecks and allowing more efficient information processing and dissemination. The disadvantage of this approach is that context consumers cannot know beforehand which context information will be available at each space, and that it may happen that the necessary information may not be available.

While most systems have no mechanism to deal with heterogeneity of context representation through different spaces, CAMUS and DRAGO pay attention to this subject. CAMUS has its ontology structured in layers to provide a shared vocabulary as an approach to tackle the problem, while DRAGO is the only one that supports the inclusion of generic mappings between the ontologies. However, DRAGO does not provide a context infrastructure, i.e., it does not have any service for context acquisition and distribution. In fact, it is solely dedicated to support reasoning with heterogeneous ontologies.

AmI applications are composed of independent entities that act autonomously in an open-ended environment, driven by their own goals. In order to fulfill their tasks, collaboration with peers is often required. Different entities are very likely to employ different knowledge representations; therefore the ability to align such representations into a single one that can be shared by different applications is paramount to ensure communication. DRAGO architecture, presented in this section, relies on pre-defined mappings to align different ontologies. Nevertheless, in practical implementations of AmI it is not feasible to build in advance mappings of all possible pairs of different ontologies that may be needed. There are other techniques to overcome the barrier of heterogeneous representations in such conditions. The next section is dedicated to a survey of approaches that try to solve exactly this problem.

4.4 Approaches for Ontology Alignment

Entities acting autonomously in an open-ended environment will often require collaboration with peers to fulfill their goals. Because different entities are likely to provide separate ontologies, the ability to integrate the ontologies into a single representation is paramount to ensure overall communication. This need, often referred to the *ontology alignment* problem [35], consists in finding a set of equivalence between a set of nodes in ontology A and a set of nodes in ontology B (see Figure 4.1). More formally, the problem of ontology alignment can be compared to that of database schema matching. Given two schemas, A and B, one wants to find a mapping m from the concepts in A into the concepts of B in such a way that, for all $(a, b) \in A \times B$, if $a = \mu(b)$,

then b and a have the same meaning. Several approaches have been proposed to perform such alignments. They can be organized into three categories [35]: structural methods (which rely only on the structure of the ontology and the nodes labels), instance-based methods (which compare the instances of each concept in the ontologies) and methods based on a reference ontology which acts as a mediator. This field is wide and complex,[1] but its application to the interaction of entities in ubiquitous environments leads to the specification of a sub-category of problems:

- The alignment process must be performed *on the fly* and in a *limited amount of time*. Indeed, in open systems, it is not possible to know in advance the nature of the entities that interact, which makes impossible to compute in advance the alignment of their ontologies.
- The entities that interact share common goals or common capacities. Thus, one can consider in most applications that the intersection of ontologies will not be empty. As a consequence, there always exists an acceptable alignment between two ontologies. However, one cannot take for sure that concepts will appear at the same level of specialization. For instance, one ontology can have a single class for the concept of *research paper*, while the other directly works with the sub-concepts *journal*, *conference_proceedings*, etc.
- The ontology alignment must be performed automatically (whereas a lot of work in this domain relies on semi-automatic approaches). As a consequence, entities must decide on alignments without the validation of a human expert. Thus, they must be able to evaluate the trust they have in the resulting alignment, e.g., by valuating the equivalence links depending on their ambiguity.

The next subsection presents the lexical alignment (a.k.a *anchoring*) that is used as a basis by all ontology alignment approaches. We then present the three main approaches for ontology alignment (structural, instance-based and mediation-based). We illustrate the advantages and drawbacks of each technique and a brief overview of the most significant work in each category. Subsection 4.4.5 then presents a brief overview of semantic similarity measures and how they can offer a new solution for ontology alignment.

4.4.1 Lexical Alignment

Lexical anchoring is, generally, the first processing step of ontology alignment tools. It is possible to differentiate several kinds of approaches, with advantages and drawbacks. First are classical Natural Language Processing

[1] See http://www.ontologymatching.org for a complete description.

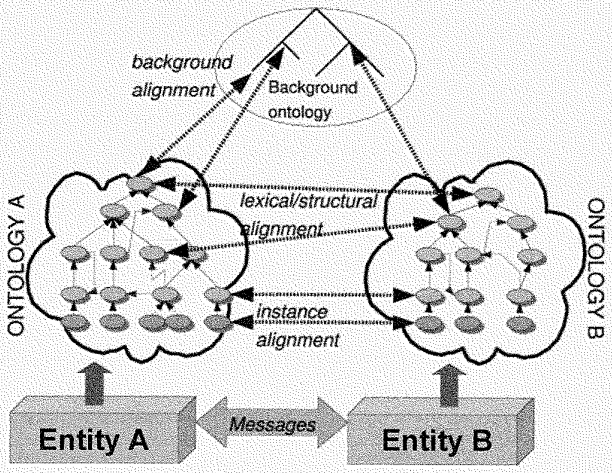


FIGURE 4.1: Ontology alignment is a set of equivalences between nodes of both ontologies. This schema presents the three classical solutions: alignment based on the structural properties, alignment based on the concepts instances and alignment based on a "background ontology."

tools, such as lemmatization (which constructs singular or infinitive forms of words, for instance, determining that *kits* is the plural of *kit*, *bought* is a derived form of *buy*), tokenization (which considers each word of a compound concept, like *long_brain_tumor subClassOf long_tumor* [3]) or suffix/prefix approach (which searches in a sub-part of the words. For instance, like *net* is an abbreviation of *network*, *ID* can stand for *PID*). However, these approaches have some limitations: the lemmatization can be ambiguous (out of the sentence context, *left* can be lemmatized either into *left:adjective* or *leave:verb*); the tokenization requires choosing the correct sub-concepts inference (is *brain_tumor subClassOf brain* a valid association?); and the prefix/suffix alignment is strongly dependent on the language (for instance, *hotel* should not match *hot*, nor can *word* be seen as an abbreviation of *sword*). For these reasons, the lexical anchoring has to be used with great care and to be completed and/or confirmed with other techniques.

A complementary approach of all these methods is the lexical distance measure, so called "edit distance" between two strings (Hamming distance or Levenhstein distance). For example, the Levenhstein distance is given by the minimum number of operations needed to transform one string into the other, where an operation may be an insertion, deletion or substitution of a single character. It is widely used for spell checking. The main advantages of edit distance are that it reproduces NLP approaches when words do not have too much complexity. For instance, the translation from plural to the singular form has a cost of 1 in most words (removing the trailing "s"). However,

some drawbacks still remain, like *sword* is equivalent to *word*, which has a cost of 1 and could be wrongly accepted.

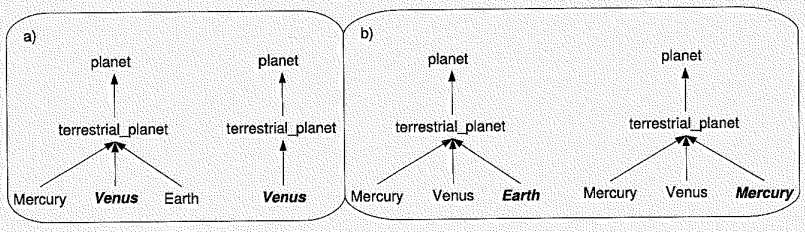


FIGURE 4.2: Structural alignment error example based on hierarchy analysis.

4.4.2 Structural Approaches

Structural approaches are based on the structural comparison of the two concepts graphs (in the meaning of graph theory). It relies on lexical anchoring as a first step for associating lexically-close labels from both ontologies. The complementary alignment pairs are obtained by an extended hierarchy comparison around these anchored concepts (e.g., in CATO [14], the authors make use of a specific algorithm for tree comparison (so-called TreeDiff) to find the largest common substructure between trees. The CATO system will be presented in more details in Subsection 4.5.4. More generally, such structural methods will match terms like PC and Personal Computers when sub-classes and properties describe the same concept (like ID, model, etc.). However, structural alignment may fail if the information is not classified using the same criterion or if the ontologies do not cover the same fields or instances. As Figure 4.2a shows, the concept "Venus" from the ontology to the right will be correctly aligned with the concept "Venus" from the ontology to the left, because they share lexically-close concepts in their whole hierarchical structures. But in Figure 4.2b, the concept "Mercury" from one ontology will be wrongly aligned with the concept "Earth" from the other ontology, because, although they do not have the same meaning, they also share lexically-close concepts in their whole hierarchical structures.

4.4.3 Instances-Based Approaches

The objective of these methods is to determine an alignment using common instances between the two ontologies. When the common instances are identified, the main idea is to suppose that the hierarchy declares these instances under the same concepts (maybe structurally or lexically different).

For example, in [30], the authors tried to align the category's hierarchy of Google and Yahoo. An instance is identified using the URL of websites. Regarding [63], the positive and/or negative matches of instances between two concepts allows them to compute subsumption alignment, in addition to equivalence alignment. For example, in Figure 4.3, if instances of the concept "a" of ontology A are classified as instances of the concept "c" of ontology B and the opposite is not true, then it is possible to deduce that the concept "c" is a super-class of the concept "a".

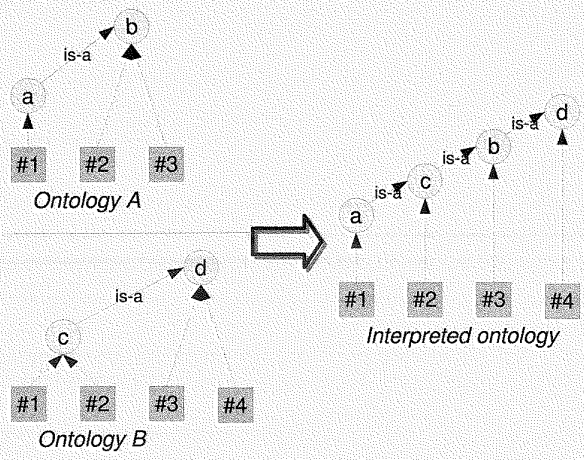


FIGURE 4.3: Instance-based alignment allows construction of subsumption alignment in addition to equivalence alignment.

However, the main drawback of this approach is the instances detection. For example, the work by Ichise et al. comparing Yahoo and Google hierarchy only generates 10% of common instances. Moreover, in van Diggelen work, it is difficult to conclude if instances intersection is not complete (i.e., if one class does not contain all instances of another class), even if it is just a problem of misidentification of concepts in one of the two ontologies.

4.4.4 Mediated Approaches

Mediated approaches are based on the use of a third ontology to mediate the alignment process (see for instance [4, 10]). The main advantage of these methods is to be more robust in case of ontologies that differ greatly either lexically or structurally, or when no instances are provided. For example, in [3], the authors align two ontologies with very different formalisms, which

could not be done using a structural approach. Thus, one of the major prerequisites (which is also the major limit of the approach) is to have access to a mediator-ontology with enough information to anchor concepts from the two initial ontologies on it. The anchoring stage is generally a lexical anchoring as presented in the previous section. After the anchoring stage, the two ontologies are represented by two set of concepts from the mediator ontology. For example, in Figure 4.4, following the subsumption relation allows the authors to find an alignment between "jeep" (ontology A) and "car" (ontology B), even if the two ontologies do not share any label. The main difficulty in mediated approaches is to define a strategy to construct semantic paths between these two sets, using the structure of the mediator ontology.

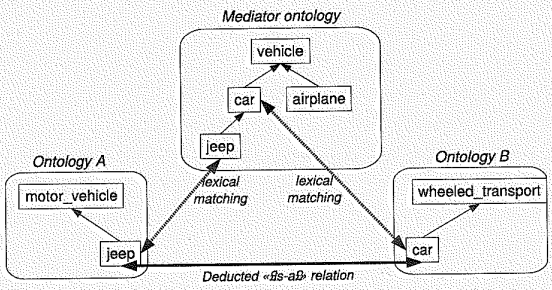


FIGURE 4.4: Mediated alignment approach.

4.4.5 Alignment Based on Semantic Similarity

Finding paths between concepts in an ontology is at the core of mediated approaches but it can also be used to complete lexical and structural alignments. For instance, a given ontology concept may not be directly attached to an application-defined concept, as required for context interpretation. This case may happen, for instance, if the alignment was difficult, or if the ontology is large. Thus, we can use the *semantic similarity measure* on the entity ontology (as it is done on the background ontology in mediated approaches) to compute correct semantic paths and to valuate the strength of this path.

In this section, we first recall the general principle of semantic similarity and we then propose to use it within a mediated approach for aligning two entity ontologies.

4.4.5.1 Semantic Similarity

From a theoretical point of view, semantic similarity is a formula that allows users to evaluate the amount of common attributes shared by two concepts. Different kinds of approaches were proposed, based on a concepts hierarchy (e.g., [15]), on glosses from a dictionary (e.g., [6, 43, 48]), etc. In this paper, we will focus in similarity for ontology, since it is suitable with our initial problem of interaction between entities.

Work on semantic similarity with ontologies can be split in two major methods: the *edge-based* approach and the *node-based* approach. The edge-based approach [49] makes use of the shortest taxonomic path to define the semantic similarity between two concepts. The main idea behind it is intuitive: in a semantic taxonomy, the longer the path, the less semantically similar are the two concepts. Recent work in this area has focused on the issue of weighting the edges, which allows refining the value of a semantic link (e.g., [29, 31, 70, 71]).

The node-based approach [51] is the most used nowadays and is considered to be the most efficient for the semantic similarity ([15] provides a good survey on the subject). The weight of a node represents the information content of the concept. In other words, the more general a concept is (i.e., near from the root), the less information it contains. There exist different kinds of formulae that combine the information content of the two target nodes and their closest common parent. The *closest common parent* is the node that is the most specific in the set of common ancestors nodes (e.g., [32, 37, 38]).

4.4.5.2 Semantic-similarity Based Alignment

In Aleksovski's work (Section 4.4.4), the use of mediator ontologies is limited to a reduced set of patterns of paths, which are considered to be "semantically correct." Thus, two concepts can only be related with a binary relation (the alignment exists or it does not exist). In the framework of interacting entities, we suggest that it is necessary to have a solution to valuate the strength of an alignment. This weight will be useful to solve ambiguity and to propose more complex dialogue strategies, as proposed in [41].

The mediator ontology should be either WordNet [22] (especially for human/agent communication) or the ontology of a third mediator agent if this ontology contains some common concepts of the two other ontologies. A first lexical anchoring of terms within the mediator ontology is performed, using the edit distance of Levenshtein. Then, the system computes the set of all possible semantic scores between each concept from the set of anchored concept of the first ontology to the set of anchored concept of the second ontology.

The key problem in this approach is that a real ontology inherently contains a lot of different relation types. To tackle this problem, we have proposed a measure of semantic relatedness [42], which is more general than the similarity measure [51], to take into consideration the entire graph and not only the hierarchy. The preliminary evaluation of our measure, applied to human-machine communication, emphasized that our correlation factor with human

judgment is approximately 20% better than other measures.

4.4.6 Conclusion

Ontology alignment is a key issue in open systems that require some form of distributed reasoning, such as in open Ambient Intelligence. While most middleware systems currently use a single ontology, openness and heterogeneity require distributed entities to be able to interpret information derived from other peers, based on an a priori unknown ontology. The three main approaches for ontology alignment we presented (structural, instance-based and mediation-based) all contain some limitation. While the structural methods are the most efficient ones, they require that the ontologies be very similar (e.g., two ontologies derived from a single initial specification). The instance-based approaches offer the consistency of Description Logic inference rules, but they work only if each concept is associated with a complete set of instances (e.g., document URIs). Mediation allows aligning very heterogeneous ontologies (even the knowledge representation formalisms can differ), but the background ontology is generally very large (i.e., larger than each one of the mediated ontologies).

4.5 The Campus Approach

In this section, we discuss Campus, a framework for the development of Ambient Intelligence applications [56]. Based on multi-agent systems technology, Campus provides an infrastructure to develop innovative context aware applications that accommodate mobile devices and environment sensor devices. The Campus architecture is intended as a configurable framework in which users can decide what services they want to enable in their environments, rather then a monolithic application. It is composed of three levels: the context-provisioning layer, the communication and coordination layer and the ambient services' layer, as illustrated by Figure 4.5. In a nutshell, the bottom level is responsible for offering basic middleware services and functionalities, such as providing context information and device discovery. The communication and coordination layer offers support for semantic interoperability, providing discovery, exchange and collaboration among hybrid entities, regardless of proprietary representations of information. Finally, the topmost layer provides application specific and ambient services and acts as a hotspot, i.e., allows users to extend the framework by plugging in specific services required by a particular user, environment, type of collaboration, of interest to their environment.

As shown in Figure 4.6, agents distributed through the two bottom layers

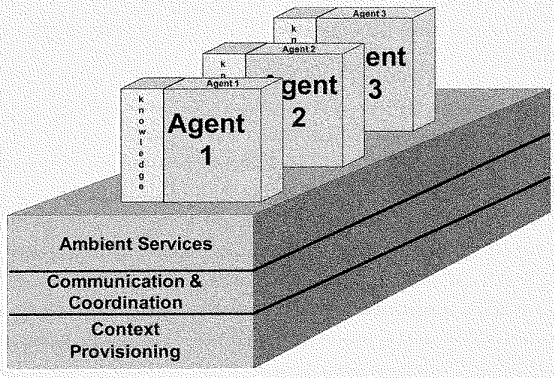


FIGURE 4.5: Abstract view of Campus architecture.

implement the main functionalities of Campus. In the context-provisioning layer, *Context Monitor Agents* (CMA) collect raw context data from various sources such as devices, sensors and applications, and make it available for interested entities. *Distributed Reasoning Agents* (DRA) infer and disseminate higher-level context information. In each smart space a *Local Knowledge Agent* provides persistent knowledge storage. It aggregates the context information obtained from context providers (i.e., CMAs and DRAs) available in that area, and builds a partial ontological view. The LKA may be queried by entities interested in finding context providers in that area. In the communication and coordination layer, a *Knowledge Interoperability Agent* (KIA) is responsible for semantic alignment of ontologies. It will provide this information to LKA whenever needed. In a further part of this section, the main features of Campus are discussed separately.

4.5.1 Context Types

In the Campus framework, context data comprises not only information about mobile devices, users' preferences and roles, description of institutional physical spaces, but also data collected from personal and smart spaces applications (e.g., appointments in a personal agenda, list of activities in an organizational scheduler, etc.). *Context Monitor Agents* (CMA) are responsible for collecting raw context data from various sources such as sensors, devices and user applications; interpreting it as context information according to a predefined ontology; and making this information available for interested entities.

Most information about mobile devices is acquired with the aid of the MoCA middleware [53], a component that supports the development of context-aware applications and services for mobile computing. MoCA provides efficient services to collect context information associated with mobile devices (e.g., CPU

usage, available memory, battery level, etc.). This information comprises not only raw data related to the device's resources and the wireless links (currently, only IEEE 802.11), but also the symbolic location of each device, which is inferred from the RSSI values measured at the device with respect to all the Wi-Fi Access Points in the device's vicinity. On the other hand, much context data is obtained from applications and files that store personal and organizational information. For example, a CMA running on the notebook of professor Silva could make available information about Silva's agenda and preferences, and also about Silva's notebook. A CMA agent running on a fixed computer at LIP6 could collect data about the schedule of activities, worker profiles, etc.

4.5.2 Ontologies

The Campus upper ontology serves as a knowledge base for the framework implementation, i.e., provides the necessary semantics to allow high-level exchanges, including brokering, negotiation and coordination amongst software entities. It contains precise definitions for every relevant concept in the framework, e.g., it defines that context providers and services are described by a tuple containing its name, a parameter list, a capability list, the communication port number and protocol. Of course, the concepts of name, parameter list, capabilities list, port and protocol are also defined in the ontology. This ontology serves as a static model of our domain and will be used as a basis upon which mediation services will try to reason and understand the information provided by entities in the environment.

4.5.3 Reasoning

In Campus, we propose a distributed reasoning mechanism to infer and disseminate higher-level context information, i.e., context information that may be deduced using data obtained from other context providers. In our approach, each reasoning element is called a *Distributed Reasoning Agent* (DRA). A DRA is able to deduce new knowledge reasoning about description logic rules. In such rules, atoms that depend on some context data compose the antecedent of the rule, while the consequent of the rule defines a new piece of context information. CMAs collect context information from several sensors or obtain it from applications and database files that contain user's preferences, device's descriptions or specific data, such as the list of activities scheduled for a set of rooms, and make it available for DRAs. Facts are deduced in runtime, described according to the respective ontology, and updated in the knowledge base. DRA implements also an event-based communication interface to where other entities subscribe their interest about a high-level context (defined by a rule). These entities are notified whenever the state context satisfies a given rule [64].

Since we consider a fully distributed environment, some context informa-

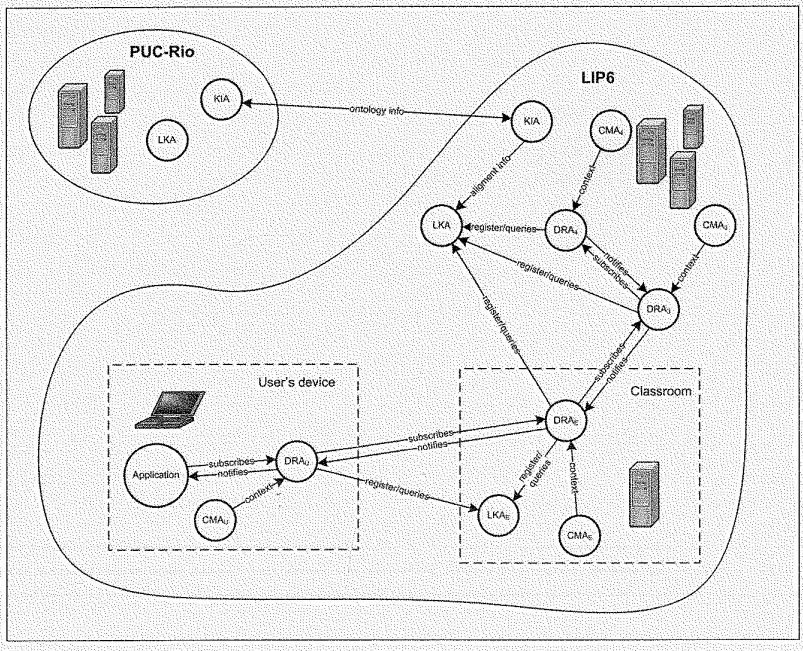


FIGURE 4.6: Campus multi-agent architecture.

tion necessary for processing the logical inference of a given rule may not be directly available from CMAs for a DRA. In such a case, the DRA will subscribe at other DRAs, which are capable of providing (by inference) the required piece of context information. In this sense, each DRA acts simultaneously as a provider, a reasoner and a consumer of context knowledge. This distributed approach brings several advantages. It allows the distribution of the high computational cost of the inferences process. As each DRA may have a different "context view," depending on the device on which it is running and its location, not all items need to be kept at a single database, avoiding a communication bottleneck. Besides, distributed inference may hide private data still revealing context information that may be inferred from it.

Figure 4.6 shows an example of this distributed interaction in the scenario where Mr. Silva enters a classroom to attend a meeting with the Campus team. In this case, Silva's smartphone executes DRA_U (user's DRA) that has access to the data obtained from the sensors at the smartphone (sound, luminosity, movement), its own location data and some administrative data available for Silva. As he enters the room, i.e., changes its location, DRA_U initiates a discovery process, and as a result, it detects the presence of LKA_E and then another reasoner, DRA_E, which is responsible for accessing the room's sensors, storing their context data and doing ambient-specific context

reasoning. We assume that an application at the smartphone responsible for managing the ring tone has already subscribed at DRA_U to get control notifications for the ring tone adjustment according the following rule:

$$\textbf{Device(?d)} \wedge \textbf{isLocatedIn(?d,?e)} \wedge \textbf{ClassroomInUse(?e)} \Rightarrow \textbf{InSilenceMode(?d)}$$

While a CMA that wraps a location service delivers the smartphone's binary property "isLocatedIn" to DRA_U, the "ClassroomInUse" unary property is only made available by the DRA_E responsible for the classroom. This property is inferred from the following rule:

$$\textbf{Environment(?e)} \wedge \textbf{hasScheduledActivity (?e, ?a)} \wedge \textbf{ClassActivity(?a)} \wedge \textbf{ActivityOncourse (?a)} \wedge \textbf{LecturerPresent(?e)} \Rightarrow \textbf{ClassroomInUse(?e)}$$

Moreover, supposing that DRA_E has no direct access to the activities' time schedule and time reference, nor to the location data of every person in the institution, it has to rely on context knowledge inferred by another two agents: DRA_3 to monitor the "ActivityOnCourse" unary property and DRA_4 to obtain notifications of the "LecturerPresent" unary property. Then, DRA_3, running on an "activity manager" (i.e., fixed device running a CMA dedicated to monitor the time and the schedule of activities), will process the following rule:

$$\textbf{Activity(?a)} \wedge \textbf{startTime(?a, ?t1)} \wedge \textbf{finishTime(?a, ?t2)} \wedge \textbf{presentTime(?t3)} \wedge \textbf{isLessThan(?t1, ?t3)} \wedge \textbf{isBiggerThan (?t2, ?t3)} \Rightarrow \textbf{ActivityOncourse (?a)}$$

At the same time, DRA_4, running on a "location manager" which has access to location information from all mobile devices detected in the building, will process the following rule:

$$\textbf{Device(?d)} \wedge \textbf{isLocatedIn(?d,?e)} \wedge \textbf{isCarriedBy(?d,?p)} \wedge \textbf{playsRole(?p, ?r)} \wedge \textbf{Lecturer(?r)} \Rightarrow \textbf{LecturerPresent(?e)}$$

In fact the reasoning outcome of DRA_U will be the result of the cascading reasoning, with new context data being inferred initially by DRA_4.

It is straightforward to notice that for interactions among DRAs referencing different ontologies, it is necessary to provide an intermediating agent that has the ability to resolve – or at least, try to resolve – the semantic mismatch among nodes in the different ontologies. For example, when the DRA_U, which is a foreign entity, wants to obtain the "ClassroomInUse" context data, and this same information is represented as "RoomBusy" in DRA_E, then the intermediate agent will have to support the interaction between the entities identifying the identity of "ClassroomInUse" and "RoomBusy" in the scope of this particular application.

4.5.4 Ontology Alignment

In this section, we will consider the problem of ontology alignment (as defined in Section 4.4) in the Campus architecture. To enable the automation of the process, we coded the resource representations using W3C's OWL-DL ontology standard, for the reasons previously discussed in Subsection 4.2.3. Ontologies are expressive, formal, machine processable representations that fulfill the knowledge requirements of AmI applications.

Our past experience with semantic interoperability enabled us to provide CATO [11, 13, 14, 21], a solution that combines well known algorithmic solutions, e.g., natural language processing, the use of similarity measurements, and tree comparison, to the ontology alignment problem. We propose to incorporate CATO to the kernel of the Campus framework. The philosophy underlying CATO's strategy mixes syntactical and semantic analysis of ontological components. Its current implementation combines the lexical and structural approaches discussed in Subsections 4.4.1 and 4.4.2 respectively.

During the alignment, lexical and structural comparisons are performed in order to determine if concepts in different ontologies should be considered semantically compatible. A refinement approach is used that alternates between lexical and structural comparison between ontological concepts. The process begins when concepts from both ontologies go through a lexical normalization process, in which they are transformed to a canonical format that eliminates the use of plurals and gender flexions. The concepts are then compared, with the aid of a dictionary. The goal is to identify pairs of lexically equivalent concepts.

We assume that lexically equivalent concepts imply the same semantics, if the ontologies in question are in the same domain of discourse. For pairs of ontologies in different domains, lexical equivalence does not guarantee that concepts share the same meaning [45, 61]. To solve this problem, we adopted a structural comparison strategy. Concepts that were once identified as lexically equivalent are now structurally investigated. Making use of the intrinsic structure of ontologies, a hierarchy of concepts connected by subsumption relationships, we now isolate and compare concept sub-trees. Investigation on the ancestors (super-concepts) and descendants (sub-concepts) will provide the necessary additional information needed to verify whether the pair of lexically equivalent concepts can actually be assumed to be semantically compatible.

Lexical comparison is done during the first and second steps of the strategy. Structural analysis is done in the second and third steps of the strategy. The final result is a OWL document containing *equivalent class* statements (<owl:equivalentClass>) that relate the equivalent concepts from the two input ontologies. This is equivalent to a mapping between conceptual schemas. The proposed strategy is depicted in Figure 4.7.

4.5.4.1 First Step: Lexical Comparison

The goal of this step is to identify lexically equivalent concepts between two different representations, as presented in Subsection 4.4.1. We begin by assuming that lexically equivalent concepts are also semantically equivalent in the domain of discourse under consideration, an assumption that is not always warranted.

Each concept label in the first ontology is compared to every concept label present in the second one, using lexical similarity as the criteria. Filters are used to normalize the labels to a canonical format: (i) If the concept is a noun, the canonical format is the singular masculine declination; (ii) if the concept they represent is a verb, the canonical format is its infinitive. Besides using the label itself, synonyms are also used. The use of synonyms enriches the comparison process because it provides more refined information. For example, in the scenario proposed in Subsection 4.1.1 of this chapter, the "activity," "class" and "meeting" concepts were identified as synonyms in our database.

Lexical similarity alone is not enough to assume that concepts are semantically compatible. We also investigate whether their ancestors share lexical similarity. It is important to note that the alignment strategy in this step is restricted to concepts and properties of the ontology. As a result of the first stage of the proposed strategy, the original ontologies are enriched with synonyms and links that relate concepts that are known to be lexically equivalent.

4.5.4.2 Second Step: Structural Comparison Using *TreeDiff*

Comparison at this stage is based on the subsumption relationship that holds among ontology concepts, similarly to what was discussed in Subsection 4.4.1, not taking into consideration ontology properties and restrictions. Our approach is thus more restricted than the one proposed by Noy and Musen [45], that analyzes the ontologies as graphs, taking into consideration both taxonomic and nontaxonomic relationships among concepts.

Because we only consider lexical and structural relationships in our analysis, we are able to make use of well-known tree comparison algorithms. We are currently using the *TreeDiff* [65]. Our choice was based on its ability to identify structural similarities between trees in reasonable time.

The goal of the *TreeDiff* algorithm is to identify the largest common substructure between trees, described using the DOM (*Document Object Model*) model [5]. This algorithm was first proposed to help detect the steps, including renaming, removing and addition of tree nodes, necessary to migrate from one tree to another (both trees are the inputs to the algorithm).

The result of the Tree Diff algorithm is the detection of concept equivalence groups. They are represented as subtrees of the enriched ontologies. Concepts that belong to such groups are compared in order to identify if lexically equivalent pairs can also be identified among the ancestors and descendants of the

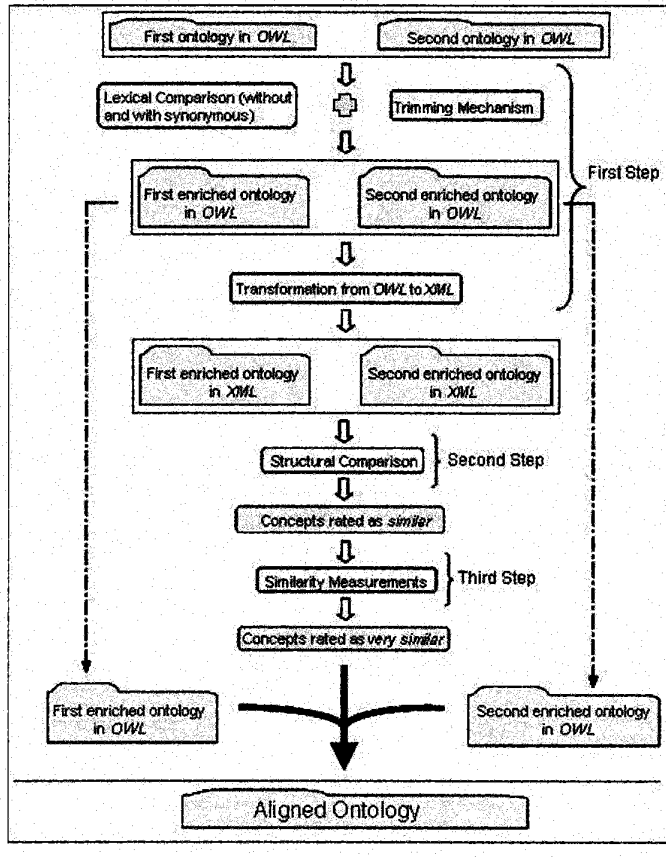


FIGURE 4.7: CATO ontology alignment strategy.

original pair. Differently from the first step, where we based our analysis and compared concepts that were directly related to one another, we are now considering the structural vicinity of concepts. Every concept in the equivalence group is investigated in order to determine lexically equivalent pairs, number of matching sons, number of synonymous concepts in the subtrees, available from the previous step, and ancestor equivalence.

4.5.4.3 Third Step: Fine Adjustments Based on Similarity Measurements

The third and last step is based on semantic similarity measurements, as discussed in Section 4.4.5.1. In CATO, concepts are rated as very similar or little similar based on pre-defined similarity thresholds. We only align

concepts that were both classified as lexically equivalent in the second step, and thus rated very similar. Thus the similarity measurement is the deciding factor responsible for fine-tuning our strategy. We adapted the semantic similarity measurement strategies proposed in [40]. The similarity threshold is fixed by the users, and can be adjusted to enforce a firmer similarity policy.

During this step the "Activity" and "Class" concepts, from the visiting professor scenario, are aligned with the "Talk" and "Lecture" concepts belonging to the Campus upper ontology. Those concepts were rated equivalent during the second step. Their similarity level is calculated in the third step.

The final ontology, containing mappings between concepts imported from the two input ontologies, will provide a common understanding of the semantics represented by both input ontologies. This representation can now be shared by entities searching for information, seeking to discover or to compose with other AmI applications. Table 4.3 depicts a part of the output ontology for this example.

In Campus, the ontology alignment is implemented by the *Knowledge Interoperability Agent* (KIA), which is responsible for applying the CATO strategy for determining the equivalent classes between different ontologies, whenever foreign entities come to interact together. If an entity queries the context infrastructure looking for some context information represented using a different ontology, it will resort to KIA to obtain the equivalent class in the prevalent ontology.

Table 4.3: A short example of the code for ontology alignment.

```
<owl:Class rdf:ID="Activity">
   <rdfs:subClassOf rdf:resource="#CATO_Thing"/>
   <owl:disjointWith rdf:resource="#Event"/>
   <owl:equivalentClass>
      <owl:Class rdf:about ="Talk">
   </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="Lecture">
   <rdfs:subClassOf rdf:resource="Educational_Activity"/>
   <owl:equivalentClass>
      <owl:Class rdf:about ="Class">
   </owl:equivalentClass>
</owl:Class>
```

The main contribution of CATO's strategy is to combine well-known algorithmic solutions, such as natural language processing and tree comparison, to the ontology integration problem. CATO is fully implemented in Java and relies on the use of the JENA API. The use of the API helped us to focus on the alignment process, for it made ontology manipulation transparent. JENA

reads and filters information from the tags of files written in an ontology language and transforms it to an abstract data model in which ontological concepts can be manipulated as objects.

4.6 Conclusion and Open Problems

Design and operation of open Ambient Intelligence Environments pose several huge challenges to the research community, which are caused mainly by the inherently heterogeneous, distributed and dynamic nature of these systems. In this chapter we first surveyed, analyzed and classified several middleware systems that propose partial solutions to the corresponding complex problem of distributed context reasoning. It turned out that only some of the systems support distributed context reasoning, and in fact only two tackle the problem of managing heterogeneous context knowledge. Then, we discussed the main general approaches for semantic alignment of ontologies, as this is a basic requirement for coping with heterogeneous knowledge representations. Finally, we presented our approach for distributed reasoning and semantic alignment in the scope of our ongoing effort to develop a multi-agent based framework Campus for development of Ambient Intelligence.

Within the Campus framework we focused on the provision of distributed context reasoning – using Distributed Reasoner Agents (DRAs) – and the construction of a software component responsible for the automatic alignment of ontologies – the Knowledge Interoperability Agent (KIA). Our strategy is based on the application of well-known software engineering strategies, such as lexical analysis, tree comparison and the use of similarity measurements, to the problem of ontology alignment. Motivated by the requirements of AmI applications, we proposed an ontology alignment strategy and tool that produces an ontological representation that makes it possible for such applications to share common understanding over information available on environment [69].

4.6.1 Discussion and Future Work

Building complex Ambient Intelligence environments requires the integration of several different context providers, which may be dedicated sensors, user's applications, databases monitors, etc. The inclusion of new types of context providers will require the implementation of new Context Monitor Agents with specific interfaces and functionalities.

The distributed inference of high-level context information brings the advantage of sharing the complexity of the reasoning process among several devices, allowing quicker response times. But on the other hand it requires efficient context dissemination to work efficiently. Aiming for efficient perfor-

mances, the balanced distribution of DRAs among the devices that compose Ambient Intelligence typical scenarios is an issue to be investigated.

Any automated ontology alignment solution presents some degree of risk, in the sense that it cannot fully guarantee that the most adequate equivalence between concepts will be always identified. Limitations of the algorithms used, time to perform the computations and possible lack of information coded in the original ontologies may, in some of the cases, prevent the automated solution to identify answers that would be otherwise manually found. The success of the CATO approach depends on the volume and quality of the information coded in the input ontologies. The richer and more complete the information, the better the results. Conversely, if the input ontologies are poorly defined, incomplete or lacking, the ontology integration engine has little data to work upon, and thus is not likely to deliver adequate results.

To tackle such situations, an alternative solution may be the use of an instance-based approach, as presented in Subsection 4.4.3. Each implementation of a device, such as Dr. Silva's smartphone (SMP-1) or notebook (NTB-1) can be thus represented by an ontology, containing a set of classes, restrictions, properties (data schema), that corresponds to the internal knowledge representation of each device. The goal of the instance-based approach is the same, i.e., find matching classes across different ontologies.

The instance based approach uses a query probing technique that consists of exhaustively sending keyword queries to original ontologies [66]. Further analysis of the results using learning algorithms and statistical analysis provides indication of good matches. This approach can be generalized to any domain that provides a reliable substitute for an unique instance identifier. In the Geographic Information systems domain, for example, there are various georeferencing schemes that associate a geographic object with a description of its location on the Earth's surface. This location acts as a universal identifier for the object, or at least an approximation thereof. We have successfully applied this approach to build mediators for Geographic Data Catalogs [11, 12, 24]. We are currently adapting the approach to be part of the Campus Framework kernel and help improve CATO results.

The CATO semantic adjustment makes use of the Maedche architecture [40], to confirm that a "very similar" rated alignment is semantically correct (and not only lexically and structurally). However, as stated by Maedche himself, the semantic measure used has two limits: 1) it only uses the taxonomic information from the ontology; 2) it does not consider that two different given edges in a taxonomy do not carry the same information content (as demonstrated in [51], see Subsection 4.4.5.1). We have proposed in [42] a new measure of semantic relatedness, which considers different weight for edges and different edges types. The preliminary evaluation of our measure shows that it increases approximately by 20% the correlation factor with human judgment. We currently try to integrate this measure in a refinement of the Maedche algorithm, in order to enhance the semantic adjustment in the CAMPUS framework.

References

[1] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles. Towards a Better Understanding of Context and Context-Awareness. In *Proc.of the 1st international symposium on Handheld and Ubiquitous Computing (HUC 99)*, pages 304–307, London, UK, 1999. Springer-Verlag.

[2] J. Ahola. Ambient Intelligence. Final report, IST Advisory Group, February 2001.

[3] Z. Aleksovski, M. Klein, W. ten Kate, and F. van Harmelen. Matching Unstructured Vocabularies using a Background Ontology. In *Proc. of Knowledge Engineering and Knowledge Management (EKAW)*, pages 182–197, 2006.

[4] Z. Aleksovski, W. ten Kate, and F. van Harmelen. Exploiting the structure of background knowledge used in ontology matching. In P. Shvaiko, J. Euzenat, N. Noy, H. Stuckenschmidt, R. Benjamins, and M. Uschold, editors, *Proc. of First International Workshop on Ontology Matching (OM-2006), co-located with the 5th International Semantic Web Conference (ISWC-2006), Athens, Georgia, USA*, CEUR Proceedings, November 2006.

[5] V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A.L. Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood. Document Object Model (DOM) Level 1 Specification. Recommendation, 1998.

[6] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 805–810, 2003.

[7] A. Bikakis, T. Patkos, G. Antoniou, and D. Plexousakis. A Survey of Semantics-based Approaches for Context Reasoning in Ambient Intelligence. In R. Bergmann, K.-D. Althoff, U. Furbach, and K. Schmid, editors, *Proceedings of the Workshop "Artificial Intelligence Methods for Ambient Intelligence" at the European Conference on Ambient Intelligence (AmI'07)*, pages 15–24, November 2007.

[8] D. Bottazzi, R. Montanari, and A. Toninelli. Context-Aware Middleware for Anytime, Anywhere Social Networks. *IEEE Intelligent Systems*, 22(5):22–32, 2007.

[9] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing Ontologies. In *Proc. of the Second International Semantic Web Conference (ISWC-2003)*, volume 2870 of *Lecture Notes in Computer Science*, pages 164–179. Springer, 2003.

[10] P. Bouquet, L. Serafini, and S. Zanobini. Semantic Coordination: A New Approach and an Application. In Dieter Fensel, Katia P. Sycara, and John Mylopoulos, editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2003.

[11] D. Brauner, M.A. Casanova, and R. Milidiú. Mediation as Recommendation: An Approach to Design Mediators for Object Catalogs. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops (1)*, volume 4277 of *Lecture Notes in Computer Science*, pages 46–47. Springer, 2006.

[12] D. Brauner, C. Intrator, J.C. Freitas, and M.A. Casanova. An Instance-based Approach for Matching Export Schemas of Geographical Database Web Services. In L. Vinhas and Antonio C. R. Costa, editors, *Proc. of the IX Brazilian Symposium on GeoInformatics. Porto Alegre : Sociedade Brasileira de Computao (GeoInfo)*, pages 109–120. INPE, 2007.

[13] K. Breitman, D. Brauner, M.A. Casanova, R. Milidiú, and A.G. Perazolo. Instance-Based Ontology Mapping. In *Proc. of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems EASe 2007*, pages 117–126. IEEE Computer Society Press, 2007.

[14] K. Breitman, C. Felicíssimo, and M.A. Casanova. CATO - A Lightweight Ontology Alignment Tool. In Orlando Belo, Johann Eder, Joo Falco e Cunha, and Oscar Pastor, editors, *CAiSE Short Paper Proceedings*, volume 161 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[15] A. Budanitsky and G. Hirst. Evaluating WordNet-based Measures of Semantic Distance. *Computational Linguistics*, 32(1):13–47, 2006.

[16] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College, 2000.

[17] H. Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems.* PhD thesis, University of Maryland, Baltimore County, December 2004.

[18] H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 2003.

[19] A. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[20] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelma. Scenarios for Ambient Intelligence in 2010. Final report, IST Advisory Group, February 2001.

[21] C. Felicíssimo and K. Breitman. Taxonomic ontology alignment - an implementation. In *Proceedings of Workshop em Engenharia de Requisitos (WER 2004)*, pages 152–163, 2004.

[22] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[23] R.T. Fielding and R.N. Taylor. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, 2002.

[24] A. Gazola, D. Brauner, and M.A. Casanova. A Mediator for Heterogeneous Gazetteers. In *Proc. of the XXII Brazilian Symposium on Databases (SBBD), Poster Session*, volume 1, pages 11–14. Sociedade Brasileira de Computao, 2007.

[25] A. Gómez-Pérez, M. Fernadéz-Peréz, and O. Corcho. *Ontological Engineering.* Springer-Verlag, London, 2004.

[26] D. Guan, W. Yuan, S.J. Cho, A. Gavrilov, Y.-K. Lee, and S. Lee. Devising a Context Selection-Based Reasoning Engine for Context-Aware Ubiquitous Computing Middleware. In J. Indulska, J. Ma, L. T. Yang, T. Ungerer, and J. Cao, editors, *UIC*, volume 4611 of *Lecture Notes in Computer Science*, pages 849–857. Springer, 2007.

[27] V. Haarslev and R. Möller. RACER System Description. In *Proc. of the International Joint Conference on Automated Reasoning (IJCAR'01)*, volume 2083 of *Lecture Notes in Computer Science*, 2001.

[28] K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Proc. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops.* IEEE Computer Society, 2004.

[29] G. Hirst and D. St-Onge. Lexical chains as representation of context for the detection and correction malapropisms, chapter 13, in WordNet: An Electronic Lexical Database,. MIT Press, 1998.

[30] R. Ichise, H. Takeda, and S. Honiden. Integrating multiple internet directories by instance-based learning. In G. Gottlob and T. Walsh, editors, *IJCAI*, pages 22–30. Morgan Kaufmann, 2003.

[31] M. Jarmasz and S. Szpakowicz. Roget's thesaurus and semantic similarity. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *RANLP*, volume 260 of *Current Issues in Linguistic Theory (CILT)*, pages 111–120. John Benjamins, Amsterdam/Philadelphia, 2003.

[32] J.J. Jiang and D.W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *Proc. on International Conference on Research in Computational Linguistics, Taiwan*, pages 19–30, 1997.

[33] G. Jones. Challenges and opportunities of context-aware information access. In *UDM '05: Proceedings of the International Workshop on Ubiquitous Data Management*, pages 53–62, Washington, DC, USA, 2005. IEEE Computer Society.

[34] M. A. Casanova K. Breitman and W. Truszkowski. *Semantic Web: Concepts, Technologies and Applications.* Springer-Verlag, 2007.

[35] Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: The State of the Art. In Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, and M. Uschold, editors, *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005.

[36] H. Kim, Y. Cho, and S. Oh. CAMUS: A Middleware Supporting Context-aware Service for Networkbased Robots. In *Proc. of the IEEE Workshop on Advanced Robotics and Social Impacts*, 2005.

[37] C. Leacock and M. Chodorow. Combining local context and WordNet similarity for word sense identication, chapter 11, in WordNet: An Electronic Lexical Database, pages 265–283. MIT Press, 1998.

[38] D. Lin. An Information-Theoretic Definition of Similarity. In Jude W. Shavlik, editor, *Proc. of the 15th International Conference on Machine Learning (ICML 1998), Madison, Wisconson, USA, July 24-27, 1998*, pages 296–304. Morgan Kaufmann, 1998.

[39] J. Lindenberg, W. Pasman, K. Kranenborg, J.Stegeman, and M.A. Neerincx. Improving service matching and selection in ubiquitous computing environments: a user study. *Personal Ubiquitous Computing*, 11(1):59–68, 2006.

[40] A. Maedche and S. Staab. Comparing Ontologies: Similarity Measures and a Comparison Study. Internal report, Institute AIFB, University of Karlsruhe,, 2001.

[41] L. Mazuel and N. Sabouret. Generic Command Interpretation Algorithms for Conversational Agents. In *IAT*, pages 146–153. IEEE Computer Society, 2006.

[42] L. Mazuel and N. Sabouret. Degré de relation sémantique dans une ontologie pour la commande en langue naturelle. In *Plate-forme AFIA, Ingnierie des Connaissances 2007 (IC 2007)*, pages 73–83, 2007.

[43] S. Mohammad and G. Hirst. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings, 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, Sydney, Australia, July 2006.

[44] B. Mrohs, M. Luther, R. Vaidya, M. Wagner, S. Steglich, W. Kellerer, and S. Arbanowski. OWL-SF - A Distributed Semantic Service Framework. In *Proc. of Workshop on Context Awareness for Proactive Systems (CAPS), Helsinki, Finland*, pages 67–78, 2005.

[45] N. Noy and M. Musen. The PROMPT suite: Interactive tools for ontology merging and mapping. *Int. J. Hum.-Comput. Stud.*, 59(6):983–1024, December 2003.

[46] P. Nurmi and P. Floreen. Reasoning in Context-Aware Systems, 2004.

[47] I. Pandis, J. Soldatos, A. Paar, J. Reuter, M Carras, and L. Polymenakos. An ontology-based framework for dynamic resource management in ubiquitous computing environments. In *Proceeding of the 2nd International Conference on Embedded Software and Systems (ICESS 2005)*, pages 1–8. Northwestern Polytechnical University of Xi'an, PR China, December 2005.

[48] S. Patwardhan and T. Pedersen. Using WordNet-based context vectors to estimate the semantic relatedness of concepts. In *Proc. of the EACL 2006 workshop, making sense of sense: Bringing computational linguistics and psycholinguistics together. Trento, Italy*, pages 1–8, 2006.

[49] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 19, pages 17–30, 1989.

[50] A. Ranganathan and R. Campbell. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. *Lecture Notes in Computer Science*, 2672:143–161, January 2003.

[51] P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proc. of IJCAI*, pages 448–453, 1995.

[52] M. Román, C.K. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt. A Middleware Infrastructure for Active Spaces. *IEEE Pervasive Computing*, 1(4):74–83, October-December 2002.

[53] V. Sacramento, M. Endler, H.K. Rubinsztejn, L.S. Lima, K. Gonalves, F.N. Nascimento, and G.A. Bueno. MoCA: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10), 2004.

[54] S. Sameshima, J. Suzuk, S. Steglich, and T. Suda. Platform Independent Model (PIM) and Platform Specific Model (PSM) for Super Distributed Objects. Final adopted specification OMG document number dtc/03-09-01, Object Management Group, September 2003.

[55] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

[56] A.F. Seghrouchni, K. Breitman, N. Sabouret, M. Endler, Y. Charif, and J.-P. Briot. Ambient intelligence applications: Introducing the campus framework. In *Proc. 13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008), Dublin.* IEEE Computer Society Press, 2008.

[57] L. Serafini and A. Tamilin. DRAGO: Distributed Reasoning Architecture for the Semantic Web. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 361–376. Springer, 2005.

[58] A. Shehzad, H.Q. Ngo, K.A. Pham, and S.Y. Lee. Formal modeling in context aware systems. In *Proceedings of the First International Workshop on Modeling and Retrieval of Context*, September 2004.

[59] E. Sirin and B. Parsia. Pellet: An OWL DL Reasoner. In Volker Haarslev and Ralf Mller, editors, *Description Logics*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, June 2004.

[60] J. Soldatos, N. Dimakis, K. Stamatis, and L.Polymenakos. A Breadboard Architecture for Pervasive Context-Aware Services in Smart Spaces: Middleware Components and Prototype Applications. *Personal and Ubiquitous Computing Journal*, 2007.

[61] G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In *Proceedings of International Semantic Web Conference (ISWC 2005)*, pages 624–637, 2005.

[62] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *First International Workshop on Advanced Context Modelling, Reasoning and Management*, Nottingham, England, September 2004.

[63] J. van Diggelen, R. Beun, F. Dignum, R. van Eijk, and J. Meyer. Combining normal communication with ontology alignment. In *Proceedings of the International Workshop on Agent Communication (AC'05)*, volume 3859 of *LNCS*, 2005.

[64] J. Viterbo, M. Endler, and J.-P. Briot. Ubiquitous service regulation based on dynamic rules. In *Proc. 13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008), Dublin.* IEEE Computer Society Press, 2008.

[65] J. Wang. An Algorithm for Finding the Largest Approximately Common Substructures of Two Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):889–895, 1998.

[66] J. Wang, J.-R. Wen, F. H. Lochovsky, and W.-Y. Ma. Instance-based Schema Matching for Web Databases by Domain-specific Query Probing. In M. A. Nascimento, M. T. Ozsu, D. Kossmann, R.J. Miller, J. A.

Blakeley, and K. B.Schiefer, editors, *VLDB*, pages 408–419. Morgan Kaufmann, 2004.

[67] X.H. Wang, J.S. Dong, C.Y. Chin, S.R. Hettiarachchi, and D.Q. Zhang. Semantic Space: An Infrastructure for Smart Spaces. *Pervasive Computing*, 3(3):32–39, July-September 2004.

[68] X.H. Wang, D.Q. Zhang, T. Gu, and H.K. Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of 2nd IEEE Conf. Pervasive Computing and Communications (PerCom 2004), Workshop on Context Modeling and Reasoning*, pages 18–22, Orlando, Florida, March 2004. IEEE Computer Society Press.

[69] A. Williams, A. Padmanabhan, and M. Brian Blake. Local consensus ontologies for B2B-oriented service composition. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 647–654. ACM, 2003.

[70] Z. Wu and M.S. Palmer. Verb Semantics and Lexical Selection. In *Proc. of the 32nd. Annual Meeting of the Association for Computational Linguistics (ACL 1994)*, pages 133–138, 1994.

[71] J. Zhong, H. Zhu, J. Li, and Y. Yu. Conceptual Graph Matching for Semantic Search. In *Proceedings of the 10th International Conference on Conceptual Structures ICCS 2002*, pages 92–106, London, UK, 2002. Springer-Verlag.

Chapter 7

Policy-Based Self-Management in Wireless Networks

Antonis M. Hadjiantonis and George Pavlou

Center for Communication Systems Research, Dept. of Electronic Engineering, University of Surrey

Abstract Self-management is expected to motivate significant research efforts, both in academia and in industry, because of the apparent benefits it can offer. Future networks and systems will transparently integrate self-management capabilities, relieving users and managers from painstaking tasks. As research progresses, the current separation of self-* properties in configuration, healing, optimization and protection will diminish, gracefully amalgamating all in a Self-maintaining operation. We envision a policy-based system as a future-proof solution, where business objectives and user preferences will be encapsulated in policies. Context-awareness will provide secure and accurate feedback to the system, assisting in fully customized and personalized user experience. Eventually policies and context will vanish inside systems, allowing users to enjoy truly ubiquitous networking.

7.1 Introduction, Background and State-of-the-Art

7.1.1 Self-Management Concepts and Challenges

Self-management refers to the ability of independently achieving seamless operation and maintenance by being aware of the surrounding environment. We propose this definition to assist in the exploration of the newly emerging field of Self-Managing Networks and Systems. This ability is widely embedded in the natural world, allowing living organisms to effortlessly adapt to diverse habitats. Take, for example, our ability to regulate our body temperature. Without planning or thinking about it, our body's functions work in the background to maintain a constant temperature. By attempting to imitate this ability for Systems, we need to provide the logic and directions for their operation and in addition the means to sense their operating environment. Sensing the environment is crucial in order to achieve awareness of surrounding conditions, threats and resources. This collective awareness of a system's operating environment is referred to as 'context-awareness'. Through context-awareness, a System can combine context with its provided logic and directions in order to adapt to changing conditions. Beyond context-awareness, in order to achieve true Self-management, a flexible and reliable way to provide necessary logic and directions is needed. For this purpose, we rely on policies and policy-based management (PBM). Once a System is informed of the policies governing its behavior and achieves context-awareness, it can independently operate and maintain itself, thus becoming a Self-managed System. The work presented in this chapter attempts to investigate Self-management through the interaction of context-awareness and policy-based management, focusing on a management framework for wireless ad hoc networks.

Among pioneering research efforts, IBM had introduced the concept of Autonomic Computing in 2001, which encapsulates the aspects of self-management in an architectural blueprint [2]. The concept was inspired by the ability of the human nervous system to autonomously adapt its operation without our intervention and has appealed to researchers worldwide. IBM's vision [3] has fueled intense research efforts both in industry and academia. In essence, autonomic computing and self-management are considered synonymous. According to IBM, autonomic computing is 'a computing environment with the ability to manage itself and dynamically adapt to change in accordance with business policies and objectives.' This fundamental definition continues to identify the quintessential four properties of a self-management system, frequently referred as self-* or self-CHOP properties: 'Self-managing environments can perform such activities based on situations they observe or sense in the IT environment rather than requiring IT professionals to initiate the task. These environments are self-Configuring, self-Healing, self-Optimizing,

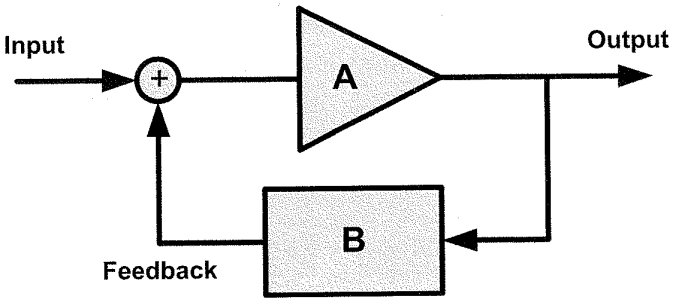


FIGURE 7.1: Closed-loop controller

and self-Protecting.'

Beyond theory, actual realization of self-management properties in computing systems poses significant challenges and remains an open and active research topic. Self-management is closely related with control systems [1] and particularly to closed-loop controllers (Fig. 7.1). By using a system's output as feedback, a feedback loop allows the system to become more stable and adapt its actions to achieve desired output. While such control loops are widely used in electronics (e.g., operational amplifiers), these concepts are increasingly used in computing after the introduction of the autonomic manager (AM) component (Fig. 7.2), as proposed in [2]. This architectural component has become the reference model for autonomic and self-managing systems and will serve as our reference point for the rest of this chapter. The autonomic manager is a component that manages other software or hardware components using a control loop. The closed control loop is a repetitive sequence of tasks including monitoring, analyzing, planning and executing functions. The orchestration of these functions is enabled by accessing a shared Knowledge base. The reference model is frequently referred to as K-MAPE or simply MAPE, from the initials of the critical functions it performs. By analyzing the definition for Self-Management, we identify policies as the basis of such systems, encapsulating high-level business objectives. Policy-Based Management (PBM) is the first building block of the Self-Management framework presented in this chapter and policies are its cornerstone. Equally important and complementary is the system's ability to sense and observe its surrounding environment. To enable these, context-awareness is employed as the second building block of the framework. As a result a policy-based context-aware framework is designed as the foundation for the implementation of self-management properties. A policy-based framework can serve as the Plan and Execute components of a self-management system, as presented in Fig. 7.2. Policy design and specification constitute the Planning phase of autonomic management while policy enforcement constitutes the Execute

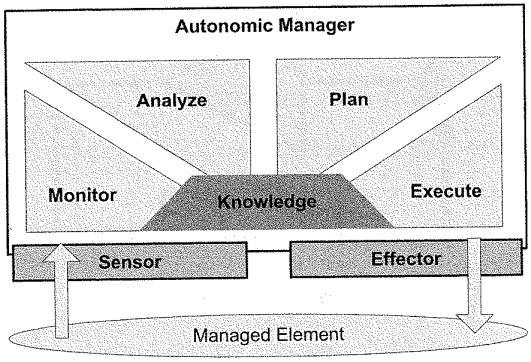


FIGURE 7.2: Functional diagram of IBM's autonomic manager (K-MAPE)

phase. On the other hand, a context-aware framework is assigned the Monitor and Analyze functionality, thus closing the necessary feedback loop. Context sensing and collection constitute the Monitoring phase while context aggregation and inference rules constitute the Analyze phase. The specification of policies and context together with their interaction form the essential Knowledge element. Policy and context repositories are the Knowledge centerpiece of both frameworks gracefully integrating the presented self-management solution. Figure 7.3 illustrates these concepts, in parallel with IBM's autonomic manager [2]. In this chapter a policy-based context-aware framework is presented, aiming to offer a platform for the self-management of wireless ad hoc networks. Wireless networks pose significantly different requirements in their management. As a result, existing solutions for fixed networks are often inapplicable, causing severe traffic overhead and performance degradation. In addition, the emergence of pervasive computing and the proliferation of wireless devices accelerate the spontaneous formation of ad hoc networks without any central administration. The investigation of these issues motivates the presented research efforts of this chapter, aiming to offer a customized solution for wireless ad hoc networks.

7.1.1.1 Self-Management in Autonomic, Pervasive, Ubiquitous Computing

Having introduced the basic concepts of self-management and autonomic computing, we further elaborate and delve into pervasive and ubiquitous computing realms. Often these concepts are used interchangeably, although slight differences exist. Pervasive and ubiquitous computing is mostly targeting user-created networks by transparently integrating appropriate hardware and software within relevant devices and infrastructure. Autonomic computing on

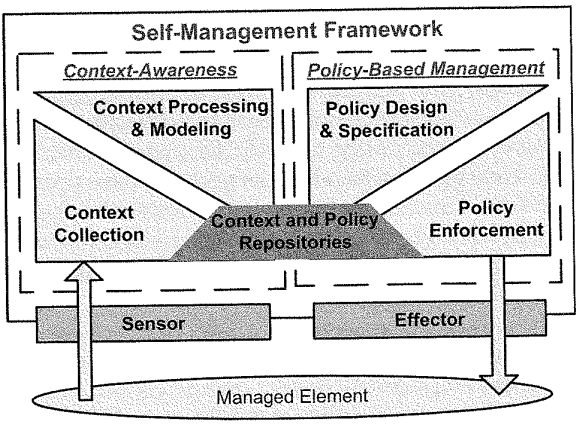


FIGURE 7.3: Mapping of proposed high-level framework to autonomic manager component

the other hand traditionally targets large-scale enterprise networks, aiming to relieve network administrators from painstaking management tasks. The realms of autonomic, pervasive and ubiquitous computing embrace the concept of Self-Management and therefore may be used interchangeably as long as their difference in focus is considered. Pervasive management is receiving intense interest from academia and industry, aiming to simplify and automate ubiquitous network operations. Pervasive management aims to vanish inside devices, relieving users from tedious configuration and troubleshooting procedures. Ideally, autonomic elements exhibit self-configuration, self-optimization, self-protection and self-healing capabilities. When combined, these capabilities can lead to adaptive and ultimately autonomic systems. In reality, the deployment of ubiquitous networks is withheld from several obstacles that need to be overcome in order to realize such a vision. These issues provide motivation for researchers, aiming to realize a system with self-management capabilities and fueling efforts for gradual transition to autonomous self-management.

7.1.2 Open Issues and Motivation

Ubiquitous networking has received both academic and commercial interest. In [4] a detailed description of the challenges for ubiquitous computing is presented from different perspectives. With the proliferation of wireless networks and increasingly networked environments, different approaches have been adopted. In [5], ubiquitous computing is proposed for home networks and in [6, 7] spontaneous approaches to networking are presented, focusing

on user's interaction and services. Today, there exists an increasing interest towards wireless and particularly ad hoc networking, as the enabling technologies of ubiquitous environments. In wireless ad hoc networks, users own mobile nodes and can move randomly and unpredictably. Their devices need to organize themselves arbitrarily; thus the wireless network's topology may change rapidly. Conventional wireless networks require some form of fixed network infrastructure and centralized administration for their operation. In contrast, since wireless ad hoc networks are self-creating, individual nodes are responsible for dynamically discovering other nodes they can communicate with. This way of dynamically creating a network often requires an equally dynamic ability to manage the network and supported services, according to higher-level management goals (i.e., policies) and taking into account the surrounding conditions (i.e., context).

As introduced earlier, policies and context are critical building blocks for the self-management of wireless ad hoc networks; therefore we present a thorough review of the applicability of Policy-Based Management (PBM) and Context-Aware Systems (CAS). Research efforts have shown that such highly dynamic environments can benefit from a PBM approach and the emerging context-driven autonomic communications trend. One of the major advantages of adopting a policy-based approach is the relevant 'controlled programmability' that can offer an efficient and balanced solution between strict hard-wired management logic and unrestricted mobile code migration and deployment.

The diverse nature of wireless ad hoc networks calls for differentiation from traditional organizational models. All views tend to adopt a distributed model and several variations exist. In this chapter we present a distributed and hierarchical (hybrid) organizational model, recognizing the emerging trend of peer-to-peer (P2P) computing for network management. The high degree of decentralization and robustness to node disconnection are useful features to be considered. While P2P systems generally scale well, there is a limiting obstacle of provisioning and synchronizing all nodes.

An additional burden in the management of wireless networks is the increasing heterogeneity of participating devices. Therefore interoperability of devices and networks is a critical issue to be addressed. Through literature, we observe that neither policy-based nor context-aware paradigms have been fully standardized, leading to increased fragmentation of research and market value. As the maturity of both paradigms is reached, there is a crucial need for interoperability. Standardization efforts are necessary to promote the usability and penetration of every new protocol or paradigm, hence we base our solution on existing IETF standards.

At the same time, the wide adoption of ad hoc networking in various aspects of our communication needs has diversified the role of managing and managed entities. An individual user of an ad hoc network demands control over his/her device and is not willing to grant permission for reconfiguration of personal settings. Such a reconfiguration might be implied from the enforcement of policies in a node that just joined the network. In [38] the author states

'personal freedoms come into play' and 'no absolute control will be accepted' from future wireless network users. The transient nature of ad hoc networking should be considered and the voluntary entrance or departure of a user from it differentiates the traditional policy enforcement paradigm. Obviously, user control is an important issue which adds special requirements on the design of management frameworks. Policies are affected by this fact, while non-uniform policy enforcement may need to be considered. Additionally context-aware systems are affected; in order to cater for the above requirements, user preferences and input need to be collected and processed.

Context is inevitably connected with the personal information of every user, especially in the case of wireless personal networks. Hence, privacy issues are raised and need to be addressed. When it comes to managing a network where the networked devices belong to individuals rather than organizations, issues like privacy and data protection should be considered. In the European Union for example, strict legislation by the European Data Protection Supervisor (EDPS, Directive 95/46/EC, `http://www.edps.europa.eu`) mandates the processing and acquisition of personal data and national authorities have been established to monitor their enforcement. Different regulations apply in the US, where a territorial approach is adopted. It is evident that the management of a network consisting of individual's devices should or is legally obliged to respect the directives regarding the collection and processing of personal data. The advancement of wireless devices and peripherals can accurately provide context which can help network management, e.g., a GPS receiver providing location data. However sensitive data like user location are private data and the user should be able to explicitly permit or deny access to them. In general, context-aware management could exploit context information available on a user's device, but a user's permission must be granted.

The issue of privacy is tightly coupled with security. The assumption of secure and trusted environments is made for the majority of presented literature, as well as for the proposed solution. However, once the assumption is lifted, major concerns are raised and need to be addressed. Self-managing systems are vulnerable to intrusion and compromise. Once again, the nature of wireless ad hoc communication adds to the problem's complexity and requires significant effort to ensure secure networking. Security is a continuous issue for every network system. In ad hoc networking, the security issues are more difficult because the wireless interface is used and access to it can not be controlled [46, 64]. Security features should exist in the management system without making it resource demanding and hard to implement. Also the lack of a centralized coordinator makes this task harder, since we cannot rely on a certificate authority for example. Furthermore novel security techniques like the detection of misbehaving wireless nodes (malicious or selfish) could be devised in order to find and block nodes that may compromise system's reliability and safety [33].

7.2 Policies and Context for Self-Management

7.2.1 Policy-Based Management (PBM) Principles

Policy-Based Management (PBM) simplifies the complex management tasks of large scale systems, since high-level policies monitor the network and automatically enforce appropriate actions [8, 9, 10, 11, 12, 13, 14, 15, 16]. In general, policies are defined as Event-Condition-Action (ECA) clauses, where on event(s) E, if condition(s) C is true, then action(s) A is executed. PBM approaches for wireless networks have been proposed in [10, 17, 18] and industry envisions autonomic computing as dynamically managed by business rules and policies [3]. In this section we present the basics of Policy-Based Management and provide a thorough literature review of related research efforts and open issues.

The main advantage which makes a policy-based system attractive is the functionality to add controlled programmability in the management system without compromising its overall security and integrity. Real time adaptability of the system can be mostly automated and simplified by the introduction of the PBM paradigm. Policies can be viewed as the means to extend the functionality of a system dynamically and in real time in combination with its preexisting hard-wired management logic [14, 19]. Policies offer to the management system the unique functionality of being re-programmable and adaptable, based on the supported general policy types. Policies are introduced to the system and parameterized in real time, based on management goals and contextual information. Policy decisions generate appropriate actions on the fly to realize and enforce those goals.

7.2.1.1 PBM Basics

The components of a PBM system are shown in Fig. 7.4 in block fashion and also in simplified UML notation. This framework had been originally proposed by the IETF and has been widely used and accepted in research and industry [12, 20, 21, 22, 23, 24, 25]. The Policy Repository (PR) is an integral part of every policy-based system because it encapsulates the management logic to be enforced on all networked entities. It is the central point where policies are stored by managers using a Policy Management Tool (PMT) and can be subsequently retrieved either by Policy Decision Points (PDP) or by one or more PMT. Once relevant policies have been retrieved by a PDP, they are interpreted and the PDP in turn provisions any decisions or actions to the controlled Policy Enforcement Points (PEP). Although a PR is a centralized concept, various techniques exist to physically distribute its contents. The reasons for distribution are obviously resilience and load balancing [11, 26, 27]. Typical implementations of a PR are based on Lightweight Directory Access

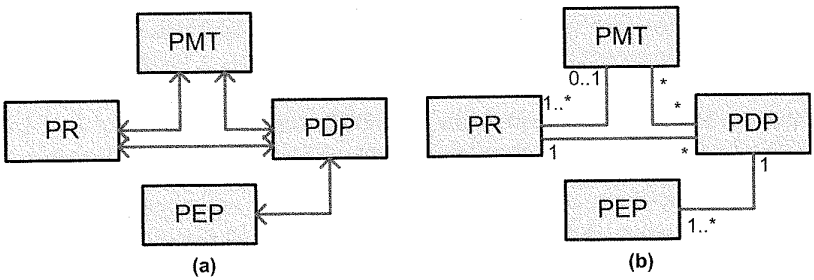


FIGURE 7.4: IETF's framework for PBM (a) block diagram, (b) generic UML notation

Protocol Servers (LDAP v3, RFC 4511 [28]), also known as Directory Servers (DS). We will refer to a DS with its directory content (i.e., policies) as a directory. A single point of failure would make policy-based systems vulnerable; therefore replication features of DS are often exploited. When designing a Policy Repository for the policy-based management of wireless networks, there exist additional requirements that need to be taken into account, e.g., tolerance against connection intermittence and multi-hop communications. These issues are examined in the proposed framework and motivate the design of a Distributed Policy Repository (DPR). In brief, standardization efforts within IETF Policy WG have specified an LDAP schema to represent policies that follow IETF specifications. Originally this representation was targeted towards the representation of QoS policies for IntServ and DiffServ architectures. However the appealing benefits of policy-based management have led different bodies from industry and academia to extend the specification and independently develop new ones, both in terms of application domains and representation. A PBM approach needs to be examined in contrast with the popular mobile code techniques as well as other traditional management schemes [29]. The benefit of policy-based management which makes it applicable to wireless ad hoc networking is the ability to control the re-programmability of the system by allowing the manager to install and remove software modules with the desired functionality on the fly. On the contrary, mobile-code techniques allow full re-programmability of the system but are quite vulnerable to malicious code execution. These security concerns have been the main obstacle in the wider adoption of mobile-code paradigms, although at first sight they appear attractive. PBM overcomes this obstacle, since the programmability of the system depends on the supported generic policy types. Although this may seem restrictive, it provides the assurance that the installed modules have been pre-approved during the system compilation and can be instantiated safely. As detailed later, there are still unresolved issues which may cause system instability and this happens when policy con-

flicts occur. Research in policy analysis and conflict detection and resolution is intense and may provide solutions in the near future.

7.2.1.2 Policy Representation and Definition

The representation of policies in a system and the policy language used is an important issue. Both depend on the selection of an appropriate information model which will provide the common ground for identifying managed objects and defining policies. Standardization efforts have focused on the development of an Information Model rather than a formal language for policy definition. These efforts are driven by the combined work between IETF [24] and DMTF [25]. The defined Information Models are conceptual models for representing and managing policies across a spectrum of technical domains. Their purpose is to provide a consistent definition and structure of data (including policies), using object-oriented techniques. These models define policy classes and associations sufficiently generic to allow them to represent different policies [48]. The Policy WG of IETF [12] has concluded during 2004 and the output was a series of RFCs defining the Policy Core Information Model (PCIM) [21, 22] and extensions for QoS (RFC 3644, RFC 3670), as well as mapping guidelines for the LDAP [36] model representation [60, 61]. However work under DMTF has continued, producing newer versions of the information model, referred to as CIM (Common Information Model) Policy Model (v2.9 Jan 2005) [60] which slightly differs from IETF's PCIM (which was based on CIM Schema v2.2). We must outline that IETF/DMTF do not define a policy language but implicitly provide a generic definition of policy rules. This definition is in the form of: if <condition >then <action >, where a policy is defined as a set of rules to administer, manage and control access to network resources [21]. IETF's policies can have some additional functionality like policy roles, grouping and prioritization, which are defined in the PCIMe information model RFC [22]. A missing element from IETF's PCIM solution is an explicit triggering mechanism which would make the system event-driven. This is important in a policy-based system, since the generic policy rule event-condition-action is widely accepted [42, 46]. Recent work on the DMTF CIM Policy Model suggested a triggering mechanism and a special query language (CIM Query Language (CQL) [47].

In academia policies have also received intense research interest. Significant work was done at Imperial College which defined a formal policy language named Ponder [42]. Ponder is a declarative, object-oriented language for specifying security and management policies for distributed object systems. Ponder does not rely on an information model for policy definition. A formal grammar is introduced instead and policies must comply with it [42, 49]. Ponder has four basic policy types: authorizations, obligations, refrains and delegations and three composite policy types: roles, relationships and management structures that are used to compose policies [49]. Other

efforts for a policy language specification for security and management are presented in [46].

Beyond the selection of a language to define policies, another issue to address is what policies are to be defined for the purpose of managing a network. The issue of policies definition is mostly independent from the policy language and representation. It is more related to what are the management goals and objectives rather than what is to be managed. A standardized information model, e.g., CIM schema, can be used to implement object-oriented design aspects of the network, using managed objects (MO). In the case of wireless ad-hoc networks, no information model describes efficiently their diverse features. However, extension of the CIM model, or any other proper information model, is possible. Literature efforts [31] have proposed an extension to SNMP MIB, called anmpMIB. An interesting effort is presented in [50] and [47], where the concepts of CIM as an extensible information model are used in combination with the Ponder policy language. Therefore the use of both Ponder as the policy language definition and an expanded version of CIM as the managed objects definition are possible and could also apply in ad hoc network environments. Additionally some specific management goals have to be defined in conjunction with a proper case study, in order to extract and refine the low level policies to be implemented. Refinement is the process of deriving a more concrete specification from a higher-level objective [46]. The task of refinement is a complex issue regarding policies, since a fully automated process is not possible.

7.2.1.3 Policy Provisioning and Storage

In a ubiquitous environment like wireless ad hoc networks, a special distribution technique of policies is vital for their effective and reliable dissemination. Since the centralized architectural model is not applicable, a central entity to disseminate policies across all nodes would become a single point of failure. Distributed and collaborative ways are needed to fulfill the special requirements of wireless ad hoc network. In order to share the overhead in the network and avoid bottlenecks, special distribution protocols need to be designed. Those protocols should cater for the varying needs of ad hoc networks and provide fast, low in overhead, reliable, resource-conscious and secure policy distribution. We should note that the policy distribution issue is tightly related to policy storage. Previous efforts on implementing policy distribution protocols are limited and here the most notable are examined. Regarding policy distribution and provisioning, we first review IETF's standardized Common Open Policy Service (COPS) and then we examine the approach of the Ponder toolkit. Finally some promising XML based solutions are reviewed.

Efforts from IETF's Resource Allocation Protocol Working Group (RAP WG) [43] have produced COPS (Common Open Policy Service) Protocol [44]

and COPS-PR for Policy Provisioning [45]. COPS is a simple query and response protocol that can be used to exchange policy information between a policy server (Policy Decision Point or PDP) and its clients (Policy Enforcement Points or PEPs). The basic model of interaction between a policy server and its clients is compatible with the framework document for policy based admission control [20]. The focus of IETF's efforts has been mainly to provide a protocol to carry out the task of policy distribution mostly related to QoS parameters and setup. Beyond the initial deployment efforts in industry during 2000 [66], COPS has not found general acceptance and interest. In academia the effort of K.Phanse described in [37, 112] utilize COPS-PR solely for the purpose of QoS configuration. The intermittent nature of ad hoc communications, though, would require that the PEP in IETF's architecture be less dependent on PDP. Therefore the usage of COPS is not a strong candidate for policy provisioning. Furthermore, different architectures introduce a dual node functionality [19], where each managed device acts both as a PDP and as a PEP, thus making the usage of COPS unnecessary. More deficiencies of COPS and COPS-PR are outlined in [65], while researchers are looking into emerging technologies to substitute COPS completely [69]. Looking into policy distribution and provisioning techniques of Ponder toolkit [42] such a protocol does not exist. Instead, remote procedure techniques are used to propagate policy decisions towards the enforcement points, using Java RMI. Researchers are looking into alternative policy provisioning techniques, mainly using XML-based architectures [63, 64] to exchange XML fragments wrapped in an HTTP message. The use of web based protocols (e.g., SOAP over HTTP) for the dissemination of policies and the usage of Web-Services has also been considered [51]. Authors of [69] have investigated substitution of COPS with NETCONF or SOAP and their evaluation results look promising.

Concurrently with policy distribution, issues of policy storage need to be considered. The existence of a policy repository (PR) in most architectures requires an efficient policy storage implementation. The prevailing solution for policy storage is the Lightweight Directory Access Protocol (LDAP) but not the only one. As mentioned before, XML based solutions exist and should be considered as an alternative. Independently of the storage technology used, other equally important issues arise and their solution is vital for the survivability of ad hoc networks. These issues include the distributed storage of the repository through replication and/or fragmentation. Similarly, issues like how to compose a distributed repository for policy updates and policy lookup, or how to check if all copies exist and are updated need to be addressed. These special issues are important in a distributed environment, but in a wireless ad hoc network they should be considered as mandatory in order to realize a robust and efficient management system.

The LDAP protocol [36, 53] is designed to provide access to the X.500 Directory while not incurring the resource requirements of the Directory Access Protocol (DAP) [52]. This protocol is specifically targeted at simple management applications and browser applications that provide simple read/write

interactive access to Directories. The reasons for the dominance of LDAP as a policy repository are some of the useful features it has to offer. The object-oriented design and implementation of a Directory using LDAP makes storage of policy objects very convenient and easy to access. The operations/services it offers like search, modify, add, etc. as well as filtering and authentication capabilities can be used in a natural way for policy retrievals, modifications and look-ups. Furthermore, the capabilities to distribute and/or replicate the directory among network nodes make it very attractive to wireless ad hoc networks management. The LDAP directory can be distributed on several physical nodes by utilizing the inherent LDAP's replication capabilities. Finally, LDAP's built-in security mechanisms can provide various levels of access control over the contents and the access to the policy repository. On the other hand, we should note that LDAP technology is more optimized towards frequent search and look-up operation rather than updates and modifications. These limitations should be considered in combination with the frequency of policy modifications in ad hoc networks [53, 54, 55, 56, 57, 58].

On another perspective, for the purpose of storing policies, XML based solutions are considered as an alternative to LDAP. The reasons are the significant penetration of XML in several devices and systems and the wide support it receives as a uniform and interoperable technology for sharing and representing data [59, 62, 63, 64]. Previous attempts in policy-based management systems for ad hoc networks have adopted different solutions for their policy repositories. In [32] a relational database is used as a Policy Repository. Specifically, MySQL database server stores policies in a proprietary way not described. This database, named CMDB, is also used for the storage of configuration and monitoring data on every node. The approach in [37, 112] is based on the COPS-PR protocol and considers a Policy Information Base (PIB) as the policy repository.

7.2.1.4 Policy Enforcement and Conflicts

Some further policy-related issues need to be addressed in order to achieve a complete and efficient PBM system. These issues relate to the decision making process and to the enforcement of policies in the network. Furthermore one has to consider whether the enforcement of policies will be uniform or choice will be given to nodes. Policy decisions are made at a PDP according to the IETF's architecture. However different architectural approaches require readdressing the decision making process and solutions are expected to be highly distributed. Both intelligence and management logic could be shared between nodes according to capabilities and roles. Local, remote or delegated decisions tactics should be considered according to the examined scenario. Traditionally, policy enforcement is expected to be uniform, i.e., all nodes conforming to same policies. However in a user-created wireless ad hoc network this is not necessary, since the purpose and formation of such

networks is different from fixed ones. An important issue emerges, regarding whether the policies should apply to all users and how their preferences are respected.

A revolutionary realization of policy enforcement would be to allow network nodes to partly conform to a global policy set. These concepts are introduced in [38] and motivate solutions that consider an enforcement mechanism which would respect user preferences and special requirements [10]. In [10, 92] cases are examined where no absolute control from an authority is accepted, discussing whether all policies should apply to all users and how their preferences should be respected. In [106] a 'promise theory' attempts to provide 'political autonomy' to entities and decentralize policy management. Such requirements significantly increase system's complexity, but yet need to be addressed in combination with the user's need to control their devices and respect their privacy.

Moreover, in an environment where a number of policies need to coexist, there is always the likelihood that several policies will be in conflict, either because of a specification error or because of application-specific constraints. It is therefore important to provide the means of detecting conflicts in the policy specification [102, 103]. Considering different conflict types, it is possible to define rules that can be used to recognize conflicting situations in the policy specification. These rules usually come in the form of logic predicates and encapsulate application-specific data and/or policy information as constraints. Examples on how these rules can be used as part of a detection process can be found in [104, 105].

7.2.2 Context and Context-Awareness

Having discussed the properties of self-management systems and their first pillar, i.e., policy-based management (PBM), we turn our attention to the second one: context-awareness. A context-aware framework is assigned the Monitor and Analyze functionality of a Self-Management system. Context sensing and collection constitute the Monitoring phase while context aggregation and inference rules constitute the Analysis phase of management. In this section we present an in-depth analysis and literature review of context and context-awareness for self-management.

In the literature, context is defined by Dey [74] as any information that can be used to characterize the situation of an entity. An entity is defined as the person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. Another definition from Malatras [72] defines the context of a system as the set of information of every nature that describes the system, influences system aspects and that is being affected by the system's operation, the ownership of which is not necessarily solely held by the system. According to [72], context awareness refers to the ability of a system to adapt dynamically and continuously its status and operation according to context information. In

essence, context is synonymous to information and it is this information that needs to be collected, modeled and processed to become useful Knowledge. Self-management systems can exploit Knowledge, combined with policies.

Regarding self-management of networked devices, context refers to their computational and physical environment and is strongly coupled with the employed management framework. Therefore we first present a taxonomy of context, to assist the reader in understanding the diversity and different forms it can take. Using the taxonomy, a system designer can characterize available context and decide on what context is needed and when. We then present an overview of context models in the literature. Context modeling is necessary to achieve true context-awareness since various context sources produce different data that have to be structured and organized under a unified representation scheme. In other words, a context model acts as a communication protocol among context aware entities, allowing interoperable and efficient processing. Finally, context storage mechanisms are discussed, addressing the need for efficient and reliable storage and retrieval of Knowledge.

7.2.2.1 Taxonomy of Context Information

A classification is presented in [71], where context is distinguished by its persistence, medium and nature. We further extend and elaborate based on Fig. 7.5 that represents a sample context information taxonomy:

- By its persistence:
  - Persistent: No updating is needed as context does not evolve or change in time (e.g., name, ID card)
  - Temporary: Updating is needed for context information that doesn't remain constant over time (e.g. position, health, interface load). Distinguished by its temporal situation:
    - Past: This category is for that context which took place in the past. The implied context history contains all previous user contexts.
    - Present: This category is for the current context, valid at the invocation moment, e.g., where am I at this moment, etc.
    - Future: Context that can be scheduled and stored a priori for future actions, i.e., the venue where a meeting will be held tomorrow morning. Prediction of future context would be very useful.
- By its nature:
  - Physical: Measurable context information that is tangible, e.g., geographical position, network resources, temperature.

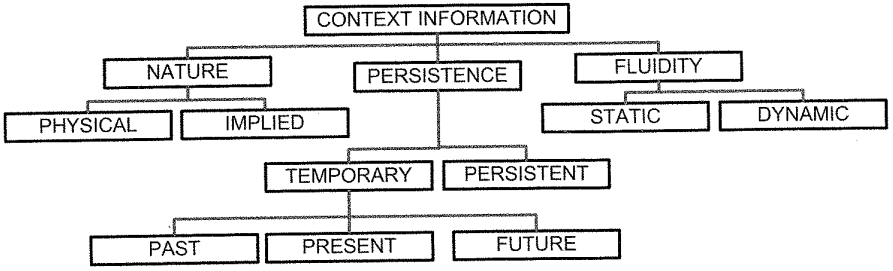


FIGURE 7.5: Taxonomy of context information

  * Necessary: Context information that must be retrieved for a specific task to run properly.
  * Optional: Additional context information which could be useful for better performance or completeness.
- Implied: Non-measurable by means of physical magnitudes, e.g., name, hobbies (it is likely that this kind of information will be introduced by the users themselves).

- By its fluidity:
  - Static: Context that does not change very quickly, e.g., the temperature along the day.
  - Dynamic: Context that changes quickly, i.e., the position of a person who is driving.

7.2.2.2 Context Modeling

Context can be exploited if it can be represented in a notion comprehensible by the entities that want to use it for decision making or monitoring. Hence a context model is required. Important requirements are simplicity, ease of deployment, performance, scalability, applicability and usefulness. From literature, two prominent context models are presented, namely based on entity-relationship model and based on Unified Modeling Language (UML). Other approaches also exist and are mentioned below.

7.2.2.2.1 Context model based on Entity-Relationship model

This context model is based on the concepts of entity and relationship and is derived from previous definition of context as well as the one given in [75]. Based on object oriented modeling, an entity can be understood as anything

that could have any kind of influence or relevance at any time in the performance of the activity of an application or service addressed to a user or a managed system [71]. An entity is composed by a set of intrinsic characteristics or attributes that define the entity itself, plus a set of relationships with other entities. The relationships belong to a specific type, among a set of available types of relationships. The concept of local context of an entity can be understood as the information that characterizes the status of the entity. This status is composed by its attributes and its relationships. Moreover, the relations that can exist between the different entities inside the model, as well as entities, can represent many different types of influence, dependence, association and so on, depending mainly on the type of entities that these relationships connect. Using this model, a network of entities and relationships can be constructed to represent the world that surrounds the activity of a context-aware service and that can influence its development. According to [71], the steps to realize a context model based on entity-relationship concept are:

1. Classification of contextual information
2. Mapping context information into an entity-relationship model
   (a) definition of generic entities
   (b) definition of generic relationships
3. Representation and implementation tools

7.2.2.2.2 Context model based on UML principles

Based on work presented in [76, 18, 96, 77, 78] another context model is presented that is better suited to resource-constrained devices. This model exploits design principles of Unified Modeling Language (UML). The popularity and usability of UML makes it appropriate for representing context. In brief, using this model, complex information is derived from a collection and combination of simpler pieces of information. The context of a device is constituted of higher level contexts that have been deduced from simpler ones. Each context can be split into atomic attributes that fully describe the initial context and are not composed of any simpler attributes. Hence, context is composed of self-explanatory attributes and perhaps other contexts, leading to more complex context structures. In addition, semantic-based relationships infer high-level context and can span from simple inference rules such as mathematical functions to semantic or user-defined operations. Semantic information is stored as context metadata, that describe its functionality, operation or meaning accordingly. The potential use of ontologies to identify semantic proximity and pattern matching is a promising research direction. UML has also been used in [79, 80] to model context information, though these solution incur a significant level of complexity on basic UML options.

7.2.2.2.3 Other approaches on Context Models

Depending on the data structures used to exchange contextual information in the respective system [81], relevant approaches are mentioned here:

- Key-Value Models: Schilit et al. [82]
- Mark-up Models: Comprehensive Structured Context Profiles (CSCP) [83]
- Graphical Models: Henricksen et al. [79], Object-Role Modelling (ORM) [80]
- Object-Oriented Models: TEA project [84], Active Object Model [85]
- Logic-Based Models: McCarthy [86]
- Ontology-Based Models: Otzturk and Aamodt [87]and CoBrA [88]

7.2.2.3 Context Storage Mechanisms

Beyond modeling of context, the actual representation of these data is critical, in order for the system to be able to process and handle them. Extensible Markup Language or XML has been widely used due to its inherent advantages:

- it is a architecture independent mark-up language suitable for structured information representation
- it can be validated and checked for errors using mature tools like XML Schemas or DTDs (Document Type Definition)
- due to its architecture independence, it is extremely interoperable and can be used as a mechanism to exchange and store data
- the context represented is searchable and can be manipulated using XQuery, a powerful XML search engine

However XML has some drawbacks:

- it is a hierarchical language that restricts database oriented architecture
- due to its verbosity, it increases the amount of information to be stored

To overcome these drawbacks and comply with the major requirement of minimizing the amount of context information transferred throughout the network, common characteristics of context can be used to reduce overheads and to avoid traffic bottlenecks and bandwidth consumption:

- Context aggregation: Context information is periodically aggregated and average values sampled over time are actually transmitted

- Normalization of context values: Context values are normalized in certain ranges, allowing for less data to be transmitted.

- Threshold criteria: Criteria associated with specific contexts may result in context transmission only when certain thresholds have been exceeded.

7.2.3 Management of Wireless Ad Hoc Networks and Self-Management Capabilities

Among various wireless technologies, Mobile Ad hoc Networks (MANET) [30] have received intense interest, especially from the research community. This interest however has not led to significant industrial exploitation or widespread adoption. According to [89], the major reason for the negligible market impact of the 'pure general-purpose MANET' paradigm is the lack of realism in the research approach. As a result, MANET are normally deployed in labs or by a few experienced users. To avoid such pitfalls, frameworks should be based on realistic assumptions and tightly coupled design with implementation and deployment on wireless testbeds. The notion of 'hybrid mobile ad hoc networks' [89] was introduced by relaxing the main constraints of pure general-purpose MANET, i.e., to consider the deployment of a network that consists of user devices with limited infrastructure support and connectivity. This assumption allows the MANET paradigm and its research results to be applied to several interesting paradigms and cases studies, e.g., mesh networks [99]. We refer to this paradigm as wireless ad hoc for the rest of this chapter, generalizing the deployment of MANET.

The deployment of wireless ad hoc networks suffers from limitations in wireless link connectivity and capacity, due to the design of Physical (PHY)/ Data Link (MAC) layers and the wide use of TCP/IP which is optimized for fixed networks. The capacity and throughput are limited and severely degrade as the user population and number of hops grow [90]. Intermittence and interference amplify the problem, since enabling wireless technologies need to share the same spectrum and ISM (industrial, scientific and medical) frequency bands are by definition subject to interference. In spite of these drawbacks, the percentage of ad hoc networks in cities worldwide accounts for an average 10% of total WLAN deployments [91], reaching 13% in Paris. In addition, the results of field measurements during CeBIT in 2006 (trade show for Telco and IT), counted 291 wireless connections of which 42% were in ad hoc mode [91]. These facts confirm that there is an increased demand for self-management of wireless ad hoc networks. By facilitating easy and efficient deployment of ad hoc networks, one can take advantage of MANET routing protocols and mesh principles to deploy wireless ad hoc networks, on top of which services can be provided. Mobile Ad Hoc Networks (MANET) offer fast and cheap deployment without the need of an existing infrastructure while emerging Mesh technologies attempt to combine the benefits of MANET

with the support of wired access points. Managing MANET and wireless ad hoc networks in general is an extremely challenging task. If we depart from cases of special-purpose deployments such as emergency scenarios and military operations, these networks typically consist of heterogeneous devices deployed by their users spontaneously in order to serve a relatively short-term purpose, e.g., file-sharing, online gaming or Internet connection sharing. These devices cannot be fully controlled from a network manager and this fact provides a fruitful ground for self-management solutions. Traditionally, network managers have authority over managed devices (routers, switches), but in ubiquitous wireless networks, users own the managed devices (laptops, PDAs). The increased heterogeneity of devices enlarges relevant problems.

Another issue that needs to be addressed in wireless ad hoc networks is the assured forwarding of packets among the participating nodes. This is one of the basic requirements for any networked application to be deployed over multi-hop ad hoc networks. The duties of fixed routers are carried out by participating wireless nodes and network operation relies on their good intentions to forward the received traffic. This not always the case and often selfish or malicious nodes refuse to forward packets, leading to congestion or, even worse, bringing the network down. Incentives mechanisms have received a lot of research interest; their deployment, though, is limited. Detection mechanisms are also investigated, aiming to determine which nodes are misbehaving and taking appropriate measures against them.

One of the crucial problems of wireless ad hoc networks is the establishment of connectivity without central administration. The basic connectivity settings for devices joining existing WLANs, e.g., public hotspots or home networks, are automatically provisioned by the controlling wireless access point (WAP). Lower levels (PHY/MAC) are automatically configured by the wireless hardware drivers, based on the WAP control packets (beacons). For ad hoc networks, the apparent obstacle is how to establish communication in the absence of a WAP. In general, one of the ad hoc devices assumes the role of a master, acting as a WAP for the rest of the devices. In most cases, initial MAC/PHY configuration is arbitrarily set at the master device by adopting default software driver and/or hardware dependent parameters. Because of the variety of software drivers and hardware-specific implementations, many wireless configuration problems arise during the initial MAC/PHY setup. The use of 'default' settings may work for isolated networks, but in cases of simultaneous network deployments can lead to interference and performance degradation. Imagine a conference venue, where different groups attempt to form ad hoc networks for file exchange, using the default settings. Most likely they will use the same channel (frequency), causing severe interference to each other and throughput reduction.

Beyond framework design, realistic implementations are needed to verify and benchmark frameworks. Simulations have been widely used in academia, e.g., using the ns2 simulator [68], mainly to overcome the obstacle mentioned above and to enable a large scale deployment of networks. Unreservedly,

simulations can provide insightful results and indications of problems and bottlenecks. They need however to be used with caution and with careful parameter setup. Research [108] has shown how careless simulation can lead to major errors and false results, as well as how the simulated results differ from reality. They have even demonstrated how different simulator tools provide different results for exactly the same simulation setup. All of these lessons have identified the need for realistic network deployments of frameworks and protocols. In real life, in order to deploy wireless network testbeds, the family of IEEE 802.11 standards [93] is usually considered, since it is the most widely deployed technology. Devices based on 802.11(a,b,g,n) are operating in ISM radio bands and can arbitrarily use any of the defined channels for deployment. The design of appropriate MAC layer algorithms makes these technologies fairly tolerant against interference and noise, but this comes at a price. Speed and performance are sacrificed in order to allow multiple stations to share the same wireless medium, i.e., the available spectrum. CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) protocols attempt to reduce the collision probability by sensing the wireless channel and backing off if it is sensed busy. The classic problem of hidden terminal is quite common. An additional measure to prevent collisions is used, the RTS/CTS handshake (Request To Send / Clear To Send), but this introduced the exposed terminal problem [94]. The use of Spread Spectrum modulation techniques can cause increased collisions due to interference between different channels (co-channel interference). This happens because channel spacing is overlapping for maximum frequency reuse. Depending on the enabling technology and modulation, different channels are likely to interfere with each other and interference increases the nearer the channels are. For example, 802.11bg technology defines 14 channels in the 2.4GHz ISM band, with center frequency separation of only 5 MHz and overall channel frequency occupation of 22 MHz. Recommended deployments in the Federal Communications Commission (FCC) region use three non-overlapping channels (1,6,11) [95]. All of the above problems provide a challenging application domain for the design of self-management capabilities. In Section 7.4 we provide a thorough solution to these issues.

In order to address the specific problems of wireless ad hoc networks, we need to review existing solutions. We present here the work on the management of mobile ad hoc networks (MANET), which are a special case of wireless ad hoc networks. Related literature in mobile ad hoc networks management is limited and proposed solutions attempt to only partly solve relevant issues. These solutions, however, take into account the requirements of ad hoc communications and are an excellent starting point towards realizing a framework for the self-management of wireless ad hoc networks in general. Existing approaches vary regarding the adopted organizational model. Recently there has been a shift towards PBM systems through a hierarchical approach. Also the proposed deployment of mobile agents amplifies their inherent security implications. The policy-based paradigm [109, 110] offers a promising so-

Table 7.1: Taxonomy of related work on MANET management

	Tiers	Hierarchical	Distributed	Policy-based	Agent-based	Modules	Storage	Managers
A.Hadjiantonis et al. [18]	2	+	+	+	-	2	LDAP	≥1
R.Chadha et al. [17]	2	+	-	+	+	1	mySQL	1
C.Shen et al. [34]	2	+	+	-	+	4	MIB	1
K.Phanse et al. [37]	2	+	-	+	-	2	PIB	1
R.Badonnel et al. [70]	2	+	+	-	-	1	anmpMIB	1
W.Chen et al. [111]	2	+	-	-	-	1	anmpMIB	1

lution since it allows dynamic alteration and controlled programmability of management logic based on the supported policy types.

The first efforts to tackle MANET management were presented in [111]. The suggested Ad hoc network management protocol (ANMP) was based on hierarchical clustering of nodes in a three level architecture. The two proposed clustering algorithms limit severely its applicability due to their centralization. The 'Guerilla' architecture [34] adopts an agent-based two-tier distributed approach where at the higher level 'nomadic managers' make decisions and launch active probes to fulfill management objectives. Mobile agents exploit a utility function to decide their migration and probe deployment. In [17] a PBNM system using intelligent agents is proposed. Policy agents are deployed and manage the network through a two tier hierarchical architecture. Policy definitions follow the principles of IETF but the use of several proprietary protocols (YAP, AMPS, DRCP/DCDP) restricts its wider adoption. Another PBM approach is presented in [112] in order to provide QoS in MANET. The proposed k-hop clustering scheme and extensions to COPS for policy provisioning (COPS-PR) protocol add policy server delegation and redirection capabilities. Although in RFC status, COPS and COPS-PR have found little acceptance and their relatively heavyweight nature may limit their applicability to MANET. Recently, a modified version of ANMP [70] has proposed the probabilistic management of MANET, where a percentage of nodes is guaranteed to be efficiently managed, depending on their connectivity properties. Finally, a context-aware solution for MANET management was introduced in [18], integrating many interesting features. This work introduces a novel organizational model specifically targeted to the needs of MANET by incorporating context awareness to dynamically adapt to the continuously changing conditions. Context information can be used to trigger cross-layer changes (network and application configurations) according to policies, leading to a degree of autonomic decision-making. Table 7.1 presents a taxonomy and summarizes related work in the area.

The context-aware PBM framework introduced in [18] was the first to consider exploiting context information in conjunction with policies for self-management of MANET, hence it has been selected for detailed presentation in this chapter. Further extensions will be introduced later, implementing realistic self-management capabilities for wireless ad hoc networks

7.3 A Framework for the Self-Management of Wireless Networks

Wireless networks pose major research challenges because of their diverse nature and their ubiquity. Their significant differentiation from fixed networks, in terms of requirements and applicability, makes traditional management approaches infeasible and expensive, thus motivating the research efforts presented here.

As networks become more and more complex, it is evident that frameworks with self-management capabilities can significantly expedite and simplify management tasks. Towards this direction we design a self-management framework for wireless networks based on policies and context to realize an adaptive closed feedback loop. The combination of these two concepts, namely policy-based management (PBM) and context-awareness, has made possible the implementation of realistic case studies on wireless testbeds. The proposed architecture combines policy design, specification and distribution with context gathering, processing and dissemination. As it will be explained later, policies and context interact by exchanging information to proactively achieve management tasks. Policies express high-level objectives, guiding the self-management of the wireless networks and providing guidelines as to what action should be executed when certain conditions are met. At the same time, context monitoring achieves a real-time understanding of the network conditions and of the surrounding environment and is used for policy conditions evaluation.

In order to achieve self-management according to higher-level objectives specified as policies, the described process is repetitive, leading to an adaptive closed loop of control. The adaptation loop is initiated with the deployment of uniform high-level policies, which are dynamically translated into management logic and distributed to capable wireless nodes. Policies can drive context gathering, i.e., the monitored context depends on the types of policies deployed, and in turn the gathered context drives policy activation and execution, leading thus to autonomic decision making.

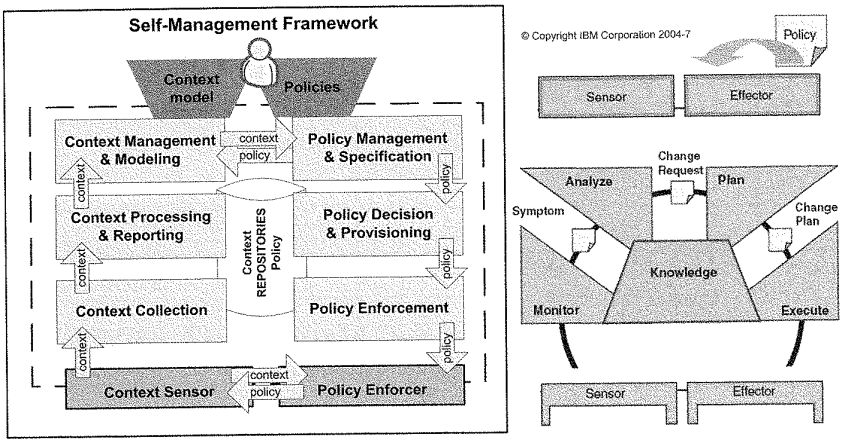


FIGURE 7.6: General diagram of closed-loop management with context and policies

7.3.1 High Level Framework Overview and Design

A policy-based framework can serve as the Plan and Execute components of a self-management system, as described by IBM's K-MAPE blueprint of an autonomic manager [2]. Policy design and specification constitutes the Planning phase of autonomic management while policy enforcement constitute the Execute phase. On the other hand, a context-aware framework is assigned the Monitor and Analyze functionality. Context sensing and collection constitute the Monitoring phase while context aggregation and inference rules constitute the Analyze phase of management. The specification of policies and context together with their interaction form the essential Knowledge element. Policy and context repositories are the Knowledge centerpiece of both frameworks gracefully integrating the presented self-management solution. Figure 7.6 illustrates these concepts, in parallel with IBM's autonomic manager. The large-scale deployment of wireless networks suggests that centralization is not an option and motivates alternative paradigms. The organizational aspects of such networks have been investigated, aiming to provide scalable and robust management. The presence of intermittence, limited capacity and other characteristics inherent to wireless ad hoc networks need to be taken into account. Furthermore, the employment of a policy-based context-aware design affects this design, adding requirements like reduced traffic overhead or anticipation of frequent disconnection and data loss.

Looking at the two extreme cases of organizational models, we have on one hand strictly hierarchical ones and on the other fully distributed ones. Each is better suited to different networks, but for the needs of wireless ad hoc net-

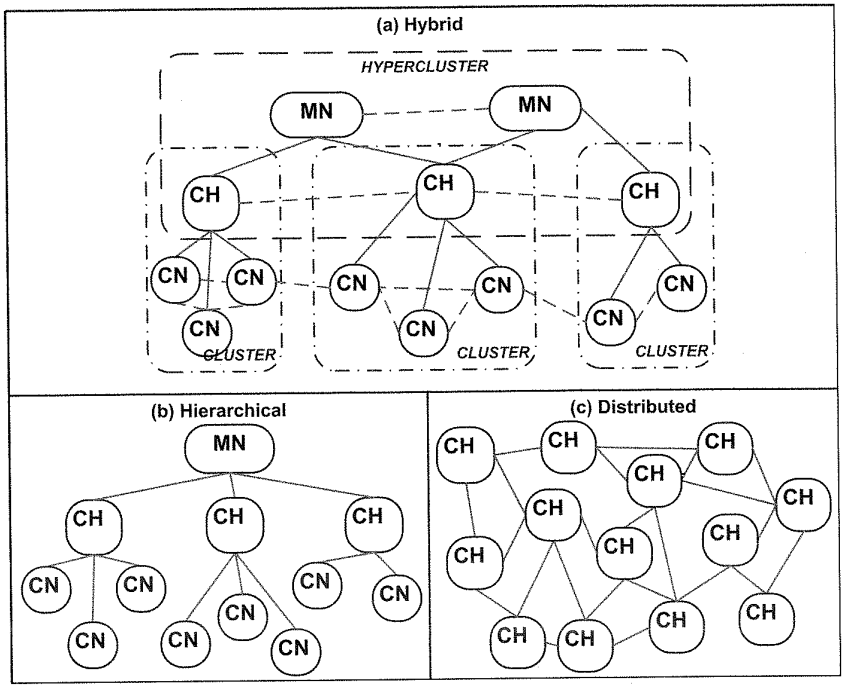


FIGURE 7.7: Organizational models: (a) hybrid, (b) hierarchical, (c) distributed

works, we adopt a hybrid approach. We aim to offer a balance between the strictness of hierarchical models and the fully-fledged freedom of distributed ones. As it will be explained, beyond hybrid deployment, the proposed model embraces both paradigms and can also be deployed as either distributed or hierarchical (Fig. 7.7). The hybrid design is based on a loose two-tier hierarchy by employing node clustering to achieve locality and restrict dissemination of traffic overhead. On top of clusters, a distributed management federation forms the 'hypercluster,' including one or more privileged nodes (managers) with extended capabilities. The multi-manager paradigm and the hyper-cluster formation are two of the distinctive elements of the introduced organizational model.

7.3.2 Policy-Based and Context-Aware Organizational Model

The interaction of policies with context benefits highly from a hybrid organizational model. Traditionally, policy-based management is used on hierarchical models for fixed networks. In such networks, over-provisioning of

bandwidth and physical resources eliminates any single points of failure and traffic bottlenecks. Obviously, this solution can not be applied to wireless networks because resources are quite limited. Battery power and bandwidth can be optimized by employing the proposed model. Specifically, wireless networks have an amplified element of locality, which is evident in a wide range of applicability scenarios. For example, ad hoc networks can be formed for a corporate meeting or can be formed from an emergency response unit, reporting to a confined disaster area. Bearing in mind the characteristics of wireless links, unacceptable delays and traffic flooding can be restricted if decision making is performed locally. To achieve that, we need a local control loop, capable of provisioning the network with fast and reliable responses. Ideally, self-management capabilities should accelerate the decision making process. By enabling clustering for management purposes, the element of locality is preserved and the requirements mentioned above are achieved. Hence, a role-based policy design is integrated to the model, e.g., cluster heads are employed in a policy-based hierarchy as local self-managing elements. A cluster head (CH) is responsible for aggregating context from its cluster nodes (CN) and provisioning them with appropriate policy enforcement decisions. Such local clusters can operate autonomously but remain fully aware of network-wide conditions and management decisions. Network-wide awareness is achieved through the collaboration of cluster heads and managers within the hyper-cluster. Based on policies, every CH reports only critical events (context) to other CHs, drastically reducing context dissemination overhead. The aggregated cluster-wide collected context can provide a collaborative network wide view of network conditions. The presence of privileged nodes as manager nodes (MN) allows the review of overall management objectives and the specification of appropriate policies. Policies in turn benefit from the clustered organization model, as they can be selectively applied, e.g., only to cluster heads or only to a specific cluster when needed, further reducing management overhead. Presented case studies will confirm these claims by providing tangible benefits. The introduction of roles and a policy hierarchy motivate the definition of different 'policy enforcement scopes.' Paired with layered context aggregation and dissemination, both architectural elements fully exploit the benefits of a hybrid organizational model. Subsequent sections will further elaborate on policy enforcement scope and layered context aggregation (Section 7.3.6).

Before delving into the policy-based and context-aware details, we first introduce some concepts on role-based management and cluster formation. Clustering is widely used in ad hoc networks for the reasons already explained. Roles are naturally introduced, to cope among others with the complexities of cluster creation and maintenance. For the purpose of our design, we rely on three roles, namely Manager Nodes (MN), Cluster Heads (CH) and Cluster Nodes (CN). These roles have been traditionally used in network layer clustering schemes for MANET routing. However, in the presented work, clustering is used at the application layer for management purposes and each role is asso-

ciated and guided by special policies. In addition, the introduced hypercluster is formed to distribute and load-balance management tasks among resource-constraint wireless nodes. The hypercluster consists of CHs and MNs, emerging as an overlay above clusters. It can execute distributed algorithms for its own maintenance (e.g., reformation, reaction to node disconnection, etc) and eliminates the single point of failure of a strict hierarchy. A wide range of algorithms can be used for cluster formation and maintenance [126, 127], depending on the applicability scenario and the network composition. For example, ad hoc deployment for tactical operations has quite different requirements than user-initiated social networks. The flexibility of a policy-based design allows the integration of different clustering schemes and in addition the real-time parameterization of their operation. Based on previous work we refer the reader to [18, 96], for a complete solution based on the Dominating Set algorithm. In brief, each node executes a distributed algorithm to assign a role to each device and to select the most capable ones to form the hypercluster. These nodes create the dominating set (DS) of the graph of capable nodes, thus ensuring one-hop accessibility for the remaining nodes. The idea is borrowed from backbone overlay networks used for routing in MANETs where the use of DS is prominent [116, 117, 118]. The novelty proposed is the exploitation of a context-aware capability function as an optimization heuristic. In [18, 96] full details and implementation specifics of such a context-aware clustering scheme are provided.

Returning to node roles, we look into the internal components necessary for each role. A highly modular design architecture is used that takes into account the heterogeneity of wireless networks and their wide applicability range. Each role has an increasingly more complex structure and added functionalities as shown in Fig. 7.8. Their components are separated in policy-based and context aware ones. The typical PBM components are used, i.e., the Policy Management Tool (PMT), the Policy Decision Point (PDP), the Policy Enforcement Point (PEP) as well as a special version of the Policy Repository (PR), the Distributed PR (DPR). Aiming to form a closed feedback loop, we complement the above components with their context aware ones. Hence we introduce respectively the Context Management Tool (CMT), the Context Decision Point (CDP), the Context Collection Point (CCP) and the Context Repository (CR). The major design difference is that the flow of information is reversed to the one in PBM systems, where a top-down approach is adopted. Here, context is collected and processed at the lower layers of the architecture and is passed to the higher layers for management decisions to be taken. At each level, respective components interact, aiming to achieve self-management and autonomic decision making. How this is achieved is detailed in subsequent sections and case studies.

Looking at a bottom up network composition, we present roles from the simplest one to the most complex. It should be noted that simpler nodes employ a component subset of the more complex ones, aiming for increased modularity and functionality reuse. This decision is mainly motivated by the

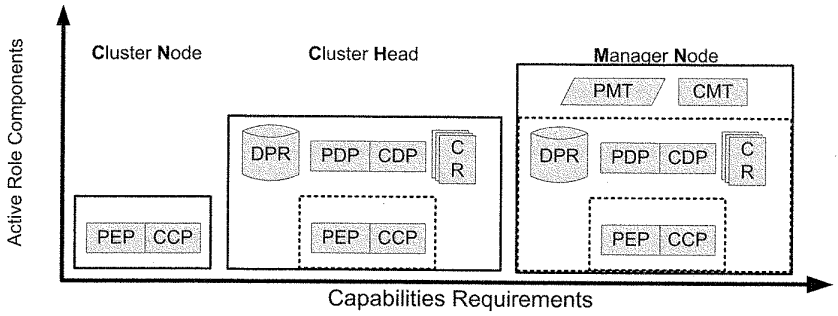


FIGURE 7.8: Block diagram of each role and internal components

need for simplified management interfaces and uniform software design. A device in Cluster Node (CN) role only uses a PEP and a CCP. Being at the lower tier of the hierarchy, such nodes participate in a single cluster and are responsible for reporting to their CH. In addition to its own local PEP and CCP, a device in Cluster Head (CH) role additionally employs a CDP and a PDP. The latter components are responsible for managing the devices belonging to their cluster, on one hand by collecting relevant context information and on the other by provisioning them with appropriate enforcement decisions. To assist decision making, a CH may locally use DPR and CR components. However, the activation of these components is also policy-based to preserve resources. Effectively a CH controls a limited number of CN that form its cluster. Finally, to enable uniform management, a role with fully-fledged policy-based and context-aware capabilities is required, hence the need for a Manager Node (MN). The extra components employed are the PMT and CMT. They offer a management interface to the human network manager, where high-level policies can be defined and altered to achieve business objectives. Depending on the applicability scenario, the assignment and responsibilities of MN can vary significantly. Figure 7.9 presents a high-level view of an example deployment using the proposed hybrid organizational model. The interaction of roles and internal components can be seen as well as the complementary policies and context information flows. Traditionally, one logical MN node is employed for network management, strictly specifying through policies the behavior of managed devices, e.g., routers, firewalls, etc. But in the examined case of wireless networks, we allow for more than one logical manager to coexist (Fig. 7.9) to cater for multi-manager scenarios presented later. Having more than one manager gives the flexibility to form networks between distinct trusted administrative authorities. This is performed without any of these being forced to forfeit its management privileges. Instead managers cooperatively introduce policies which guide the overall network's behavior. For example, an ad hoc network can be setup for a corporate meeting be-

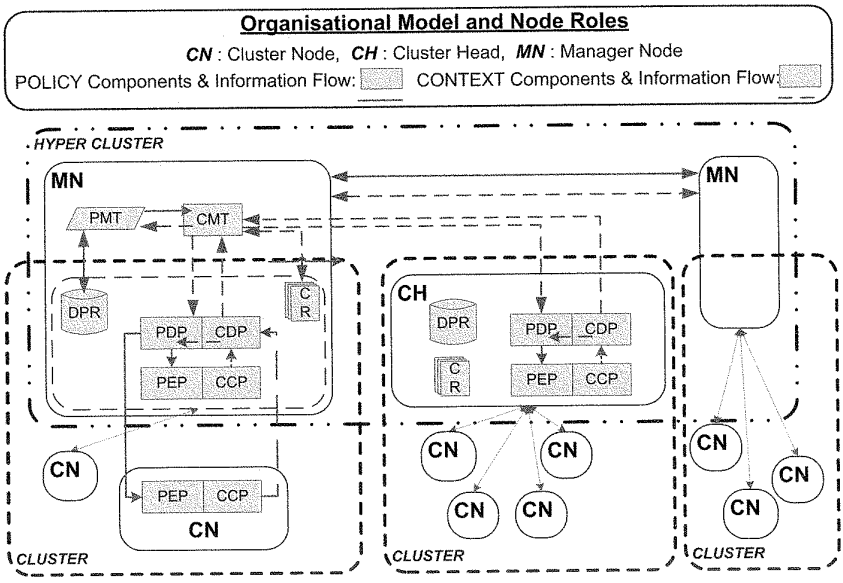


FIGURE 7.9: Hybrid organizational model with internal components and information flow

tween two companies' representatives. The multi-manager paradigm treats the companies' managers as equals and allows both to affect network behavior by introducing policies. In addition, from a functional point of view, in large scale ad hoc networks scalability issues demand more than one manager in order to control and administer effectively the numerous cluster heads and cluster nodes. The managed devices in such scenarios are user-owned devices, like PDAs, media players, etc. These devices are not strictly managed and can benefit from a multi-manager paradigm, e.g., for service provisioning.

Before elaborating on the specifics of the Policy-based and context-aware aspects of the introduced organizational model, we present a detailed schematic representation of internal components in Fig. 7.10. Components are presented regardless of mentioned roles, to better illustrate their interactions.

7.3.3 Policy-Based Design for Autonomic Decision Making

Policies can serve as the Plan and Execute functionality of a self-management system, as described by IBM's well-known MAPE blueprint of an autonomic manager [2]. Policy design and specification constitute the Planning phase of autonomic management while policy enforcement constitutes the Execute phase.

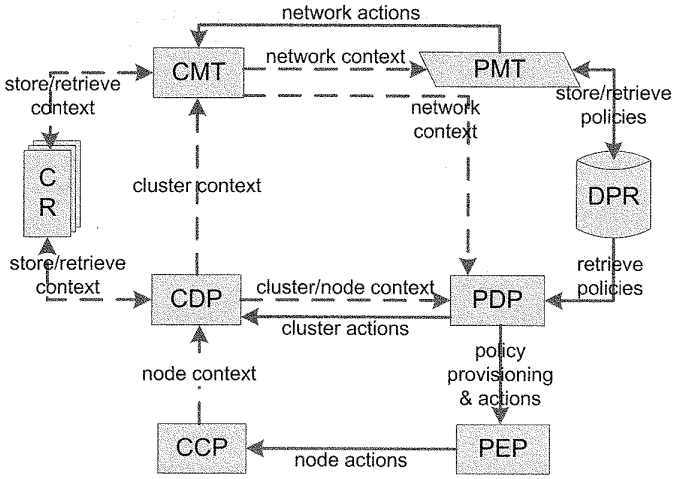


FIGURE 7.10: Policy-based and context-aware components and interactions

Wireless networks and particularly ad hoc ones are difficult to accurately plan, mainly because the majority of the participating devices are mobile users. The unpredictable nature of human users hinders the efforts to provision the network and provide reliable services to them. This is an important point that the presented design attempts to exploit for the benefit of management performance. The main concept is to Plan using policies to dynamically adapt the management capacity of the network according to the users' population and contextual information. Management capacity refers to the distributed use of processing power and resources from user devices for collaboratively serving the wireless network. For example, consider the case of planning the deployment of a temporary wireless network for the needs of a large conference. Instead of over-provisioning the area with several wireless access points, a dynamic number of user devices can be employed as cluster heads and use multi-hop routing protocols to achieve connectivity. In practice, the over-provisioning solution is preferred because of its predictable configuration and administrative simplicity, in spite of its higher cost. In order to entrust the management of a network to user devices, there is a need to infuse predictability and simplicity. To this purpose, policy-based management is employed to provide the means for controlled programmability and self-management capabilities.

The designed policy-based framework provides a highly distributed management environment that can cater for the self-management of wireless ad hoc networks. Management logic is encapsulated in policies that are transparently enforced to devices. For example, Network Operators and Service

Providers use the multi-manager PBM system to introduce the appropriate policies aiming to set guidelines for the management of numerous user devices. Contrary to traditional management systems, the designed system may not require the mandatory enforcement of policies and tight control of managed devices. Instead, the system physically and logically distributes the policies among devices, making them available to vast numbers of users that choose to enforce the relevant policies eventually relieving them from manual configuration. Policy enforcement is the Execute engine of a self-managing system.

In subsequent sections we introduce a sophisticated context-aware model to Monitor and selectively Analyze critical contextual information, complementing the PBM design and providing a real-time understanding of network conditions without adding significant overhead (Section 7.3.4). Context and policies are the essential Knowledge, necessary to every self-management system. Knowledge management depends on accurate context modeling and a Distributed Policy Repository (DPR). The DPR is designed as an extension to the traditional Policy Repository and responsible for the distribution of policies among the network (Section 7.3.5).

Policy Design, Specification and Enforcement

In order to apply the PBM paradigm to our system we adopt the standardized by IETF/DMTF information model for policy representation. Defined policies are represented according to PCIM/PCIMe [121, 122]. To overcome the lack of an event notation in PCIM, we extend the abstract Policy element as PolicyEvent, without loss of interoperability. Based on this representation, mapping policies to the standardized LDAP data model [123, 124, 125] is straightforward. The focus here is placed on the definition of the necessary policies for wireless network management, rather than the formal definition of a policy language. We believe this representation is both effective and lightweight so as to cater for the policy needs in the resource-poor wireless ad hoc environment:

```
{Roles} [Event] if {Conditions} then {Actions}
```

The Event-Condition-Action notion (ECA) is widely used in the literature due to its simplicity and effectiveness. It allows complex policy structures to be formed (e.g., policy groups), as well as increased reuse of both policy Conditions and Actions. Roles element defines which devices will need to apply the specific policy. It also helps grouping policies and easily retrieving them. Event element triggers the evaluation of policy conditions. It can be a periodic, time-based or scheduled event, as well as dynamic real-time event or event correlation. Depending on system's capabilities and complexity, a sophisticated event bus and correlation engine can be implemented. For the purpose of the presented design, a context-aware event service is used. The Conditions element is a Boolean expression containing one or more conditions to be evaluated. If the condition is true, that would trigger the execution of specified actions. The Actions element contains one or more actions needed

to be enforced once a specific event has occurred and policy conditions are true.

As already mentioned, to cater for the needs of policy-based design, the four components proposed by IETF are employed and modified. We further elaborate on the internal structure and implementation guidelines of these components, aiming to provide system engineers with an insightful presentation of how they can be employed for the management of wireless networks. In the following section, policy examples will further contribute to this goal. Taking a bottom-up approach we begin with the PEP, followed by PDP and finally PMT. We recommend consulting Fig. 7.10 (pp. 230) for better understanding of each component. The DPR is presented later in a special section (Section 7.3.5):

7.3.3.1 Policy Enforcement Point (PEP)

PEP is the simplest component of a PBM system, responsible for the enforcement of policy decisions on the Managed Objects (MO) it carries, as these decisions are provisioned by the controlling PDP. Depending on the management interface, a PEP may need to be tightly integrated with its host device as it may directly communicate with hardware elements. Contrary to fixed network managed devices, where an MIB agent is normally available, managed devices in wireless networks rarely integrate one; hence a device-dependent middleware layer may be needed to set required configuration parameters and monitor available objects. There is an apparent need for accurate description and interfacing with available MO and, to this end, various information models can be used and/or extended. The presented design extends the PCIM/PCIMe information model to define the actions PEPs need to enforce. Providing PEPs with instructions to what actions need to be enforced is referred to as policy provisioning and it is the responsibility of PDPs.

7.3.3.2 Policy Decision Point (PDP)

PDP is the component responsible for ensuring that policies are applied to all the PEPs it has under its control. It is responsible for evaluating the condition policies and provisioning the required actions to PEPs (policy provisioning). IETF has defined COPS/COPS-PR protocols for this purpose, although in practice they are rarely used. More lightweight approaches are adopted in wireless networks, like XML-RPC, RMI, etc. PDPs also communicate with a Policy Repository to retrieve current policies and instantiate them as Policy Objects (PO). The communication protocol between PDP and PR is primarily LDAP, although recently approaches based on Web Services are explored.

7.3.3.3 Policy Management Tool (PMT)

PMT is the interface between the PBM system and a human manager. Using the PMT a manager can specify high-level directives and management objectives, usually in a simplified graphical user interface (GUI). PMT translates the high-level directives to the internal policy language specification to validate their consistency and feasibility. In addition, it may perform policy analysis, aiming to identify possible conflicts. Conflict detection and resolution (CDR) is a critical task, receiving intense research interest. We elaborate on such issues in the next paragraph. If conflicts exist, a resolution process is initiated that can be either automated or may require human intervention. Once policies are checked, PMT communicates with the Policy Repository to store the new policies, normally over LDAP. A similar procedure is followed on policy modification.

Policy-based management simplifies the complex management tasks of large scale networks, by using policies to automatically enforce appropriate actions in the system. But in an environment where a number of policies need to coexist, there is always the likelihood that several policies will be in conflict, either because of a specification error or because of application-specific constraints. It is therefore important to provide the means of detecting conflicts in the policy specification. Considering the different conflict types, it is possible to define rules that can be used to recognize conflicting situations in the policy specification. These rules usually come in the form of logic predicates and encapsulate application-specific data and/or policy information as constraints. Examples on how these rules can be used as part of a detection process can be found in [104, 105]. By adopting a multi-manager scheme we allow more entities to offer different services to the users, without violating their privacy concerns and preferences. This creates an increased need for an automated conflict detection and resolution mechanism that will prevent policy inconsistencies among different managers and allow for truly self-managed systems to be realized.

7.3.4 Context-Aware Platform for Information Collection and Modeling

Realizing a truly self-managing system requires an integrated Monitor and Analyze functionality to complement Planning and Execute. It has been shown that policy design and specification constitute the Planning phase of autonomic management while policy enforcement constitute the Execute phase. In this section, we introduce a context-aware framework assigned to the Monitor and Analyze functionality of the designed autonomic management system. Context sensing and collection constitutes the Monitoring phase while context aggregation and inference rules constitute the Analyze phase of management.

Based on the above, the IETF policy framework is extended by comple-

menting the policy-related components with a novel group of entities related to context collection and processing. These are necessary for a system to become capable of sensing, communicating with its surrounding environment and adapting to changing conditions. Incorporating context awareness into the policy-based management framework makes it flexible and dynamic in response to the inherently unstable domain wireless ad hoc networks, allowing a degree of autonomy to be reached.

Context information collected from all the nodes forming the MANET refers to their computational and physical environment and is tightly coupled with the policy-based management system since it is this information being monitored that may trigger a certain policy. Every node collects its own context information based on its available sensors. The term sensor is generic since it can refer to a battery monitor, a GPS receiver, etc. To increase performance and scalability, a context model is needed to represent the collected information efficiently and accurately. Based on these requirements we employ a context model based on Unified Modeling Language (UML) design principles, full details of which can be found in [96]. For clarity, we briefly introduce its main features. The model incorporates the notion of semantics to describe context information, sensors and their relationships. The general context of a node consists of higher level contexts that have been deduced from simpler ones, i.e., mobility prediction of nodes deduced from device capabilities, GPS readings, personal diaries, etc. For this purpose, relationships are defined in terms of simple inference rules or mathematical functions. The UML model is inherently associated with the data representation of the collected context. It allows for expressiveness and can be easily mapped to an XML document for interoperable storage. In addition it allows for integrity validation using well-established XML techniques (e.g., XML Schema).

The presented system involves four novel components that deal with context awareness: the Context Collection Point (CCP), the Context Decision Point (CDP), the Context Management Tool (CMT) and the Context Repository (CR). There is an obvious matching of these components to the four elemental components of a PBM system. The major design difference is that the flow of information is reverse of the one in PBM systems, where a top-down approach is adopted. Here, context is collected and processed at the lower layers of the architecture and is passed to the higher layers for management decisions to be taken. For further details on the used context model, the reader is referred to [96]. A detailed case study is also described in [96], where the presented components are customized for the management of MANET. In the following paragraphs, a more generic approach is followed to allow the use of these components and the framework in various wireless scenarios and case studies. As in the presentation of policy-based components, we recommend consulting Fig. 7.10 (page 230) for better understanding (CR is presented later in a special section, Section 7.3.5):

7.3.4.1 Context Collection Point (CCP)

The CCP is installed on every node and its responsibilities include monitoring the environment through the available sensors and collecting the relevant information. Sensors may include a power monitor, a GPS receiver, a Bluetooth monitor, etc. CCP's operation is guided by the enforced action of its collocated PEP. These actions include the definition of context objects to report and monitor configuration of context collection/reporting intervals and various parameters related to the local management of context within a node. A local context manager coordinates the operation of CCP by interacting with the context aggregator/optimizer. After some basic pre-processing, context information is stored in context objects using the available context model. Only the requested information is forwarded from every CCP to their respective Cluster Head's CDP.

7.3.4.2 Context Decision Point (CDP)

The CDP is installed on devices in CH or MN roles, i.e., on devices currently consisting of the hypercluster. Its main responsibility is to extract context information from the controlled cluster: (1) for feedback to the collocated PDP and (2) for forwarding aggregated cluster context to other hypercluster nodes. Like with the PEP/CCP interface, the local PDP configures the collocated CDP, which in turn reports the requested context. The Cluster Context Aggregator/Optimizer communicates with all CCPs belonging to the same cluster through the Cluster PEP/CCP Communication Adaptor to extract the aggregated cluster-wide context. It can use specified inference rules of the context model to infer and combine simple node contexts to more complex cluster-wide ones. Context objects maintain an updated view of current cluster conditions. The local PDP evaluates conditions of cluster-wide Policy Objects with received context, to check whether the actions of a certain policy are triggered for that cluster. Specific aggregated context information from each cluster is also forwarded to their respective MNs, to allow them to acquire a network-wide view of context and conditions.

7.3.4.3 Context Management Tool (CMT)

The CMT is available only at nodes in the MN role and interacts with a collocated PMT to provide a graphical interface for a human manager or administrator. Similarly to CDP, it adds another level in the context hierarchy, the network-wide level. Each CMT collects and aggregates cluster-wide context information from its controlled CHs. This context is processed and exchanged among the other CMTs of MNs. This ensures a network-wide common knowledge regarding context information. It is this context information

that is returned from the CMT to the hypercluster PDPs and may trigger appropriate network-wide or hypercluster-wide policy actions. The above ensure network-wide concurrent triggering of policies and thus network-wide adaptation when required.

7.3.5 Distributed Policy and Context Repositories — The Importance of Knowledge Management

Policy and context repositories are the Knowledge centerpiece of the presented framework, gracefully integrating the self-management solution. Knowledge is important at every phase of a self-management system and needs to accurately depict managed resources, management objectives, network conditions, contextual information and devices' status. All of these elements need to be acquired through the network in a consistent, efficient and scalable manner. The specification of policies and context, together with their interaction, form the essential Knowledge element. Policies encapsulate high level directives as well as low level actions to achieve management objectives. Context modeling on the other hand provides a layered view of network conditions by collecting and combining simpler context to complex ones. Both policies and context need to be available among wireless nodes when needed, hence the critical need for Policy and Context Repositories.

The policy repository (PR) is a critical component in every PBM system and we cannot rely on a single node to store it. The idea of storage replication is not new and is widely used in fixed networks as a backup in case of failures. In wireless networks however due to the intermittent nature of links, it is expected that nodes will become disconnected frequently. Thus access to a central repository cannot be guaranteed depending on the networks' volatility and mobility. In order to tackle this deficiency the DPR (Distributed Policy Repository) has been proposed. The key point is to ensure uniform network management. This can be achieved by the presence of a synchronized Distributed Policy Repository among network nodes, which in turn guarantees that all PDPs behave in the same way and enforce the same actions in a network-wide fashion. Because of the unified manner aggregated context is presented to the MNs, the conditions evaluation is the same, ensuring robustness and consistency. The DPR concept anticipates the need for provisioning large-scale wireless ad hoc networks, without the need for over-provisioning management resources, e.g access points, bandwidth or human effort. Because the deployment of such networks varies significantly in terms of spatial and temporal parameters, accurate planning and pre-provisioning is extremely difficult. Hence the proposed distribution of management tasks among Policy Decision Points (PDP) hosted on user devices, based on the policy guidelines stored in the DPR.

The innovation in designing a novel DPR lies in the adoption and customization of features from existing fixed network repositories for use in a wireless environment. When designing a Policy Repository for the policy-based man-

agement of wireless networks, there are additional requirements to be taken into account:

- PDPs may be intermittently connected to the ad hoc network and occasionally may not have a route to a Policy Repository instance.
- The nearest PR instance may be several hops away from PDPs, thus introducing significant traffic and latency overhead to the propagation of new or updated policies.
- Multi-hop networks suffer from severe bandwidth degradation as the number of hops increases.
- Additional PDPs may need to be dynamically assigned to anticipate fluctuation in PEPs population. Special conditions may lead to spatiotemporal increase of PEPs density. Current repositories do not consider such conditions.
- Wireless networks increasingly consist of heterogeneous end-user devices that cannot be fully controlled by a network manager.

These requirements prevent the unmodified adoption and deployment of a Policy Repository (PR) using the various techniques targeting fixed networks and have motivated research efforts for an enhanced PR, the Distributed Policy Repository (DPR).

The presented DPR idea was introduced in [18] where different replication states were enforced depending on network's mobility. Details of this proposal are mentioned as a policy example in Section 7.3.6. According to [18], the DPR can have different degrees of replication (e.g., number of replicas) according to how volatile the network is. In [18], the DPR concept is extended to combine a priori knowledge of localized events (e.g., scheduled sport event) with dynamic real-time context information (e.g., processing load or free memory of each PDP). A series of algorithms can be included in the implementation of policy actions, resulting in a highly customizable deployment of the DPR overlay. In effect policies and context guide the DPR behavior and replicas' distribution, ensuring on one hand maximum repository availability (distributed copies) and on the other hand a single logical view of the stored policies (replicated content). Thus, efficient management of clusters can be achieved even when temporarily disconnected from the network manager. The immense importance of Knowledge management has further motivated the integration of a Context Repository (CR) to the presented framework. It can gracefully integrate with the Distributed Policy Repository to effectively create a complete Knowledge management solution for autonomic management of wireless ad hoc networks.

There are a number of significant differences in the design of a Context Repository compared to the design of a Policy Repository:

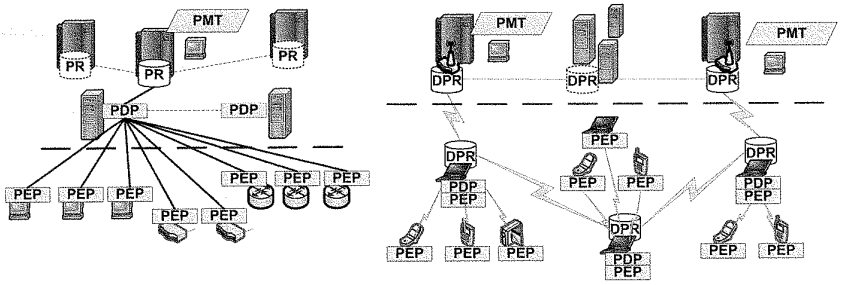


FIGURE 7.11: Traditional (left) and proposed (right) policy repository deployment

- Context has a localized and temporal importance and as such it is not necessary to widely distribute and rigorously require consistency as in the case of policy storage and distribution.
- Context is archived and outdated far more quickly than policies; hence its freshness and accuracy pose different requirements.
- Context at different hierarchy levels can be stored locally and be used independently by each node or cluster of nodes. Only context needed for the inference of network-wide context is required to be distributed.
- Network-wide context is important and needs to be consistently stored and distributed among hypercluster nodes. This is the main motivation for a Context Repository, beyond the local context storage.

A Context Repository instance can be deployed on the nodes constituting the hypercluster. This allows them to maintain a wider view of network conditions and specifically be notified of context information needed for the evaluation of network-wide and hypercluster-wide policy conditions. Contrary to the standardized IETF/DMTF use of LDAP for policy storage, the implementation of context storage is open. Storage of context in the CR is tightly dependent on the employed Context model. For example, use of the aforementioned UML-based context model would imply the use of XML format for context representation and storage.

7.3.6 Context and Policies Interaction for Closed-Loop Autonomic Management

The designed framework is further enhanced by integrating a closed control loop, as described in this Section. Having in mind the component's structure presented earlier (Fig. 7.10, page 230), we introduce the concept of 'enforcement scope.'

To achieve layered closed-loop autonomic control, we define different decision layers to limit context dissemination and control traffic overheads. We define the 'enforcement scope' of a policy as the set of nodes where actions need to be enforced, when the policy is triggered by the context collected in this set. Figure 7.12 illustrates the realization of layered closed-loop autonomic control through component interaction. Based on the above definition, three enforcement scopes are realized for the needs of our design:

7.3.6.1 Cluster-wide Enforcement Scope

Policies can be triggered at a hypercluster node by the context aggregated within its cluster. Decisions are enforced only at the cluster nodes belonging to the cluster where the policy was triggered. These policies are identified by their assignment to the CN and CH role.

7.3.6.2 Hypercluster-wide Enforcement Scope

Policies can be triggered at all hypercluster nodes by the context aggregated within the hypercluster. Decisions are enforced only at the hypercluster nodes. These policies are identified by their assignment to MN and CH node roles only.

7.3.6.3 Network-wide Enforcement Scope

Policies can be triggered at the MNs by the context collected and aggregated from all network nodes. The PDPs of MNs decide to enforce the actions network-wide and delegate those actions to CHs to be enforced to all PEPs of wireless nodes. These policies are identified by their assignment to all three roles (MN, CH, CN).

Regarding the actual policy design, in the following section we present realistic examples of policy types in order to illustrate these concepts. These policies provide a first step towards a flexible and adaptable management framework specifically designed for the needs of a wireless network.

7.3.7 Overview of Applicability and Policy Examples

To demonstrate the effectiveness and combined applicability of policies and context, in this subsection we present illustrative examples, based on a case study for the management of Mobile Ad Hoc Networks (MANET). MANET are a representative example of wireless ad hoc networks and we have already presented the motivation for their efficient management. Through these examples we aim to demonstrate the effectiveness of simple policy rules and the applicability of the defined policy enforcement scope. The chosen policies are not overly complex for clarity and serve as an introduction to policy design. In spite of their simplicity, they provide powerful tools for MANET self-management since they can proactively take measures to prevent the depletion

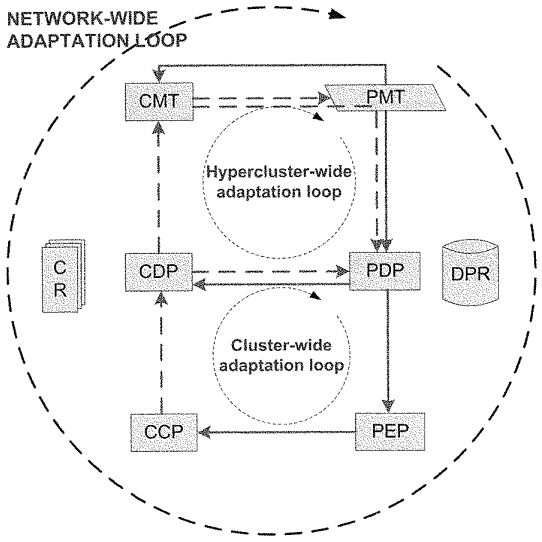


FIGURE 7.12: Diagram of closed-loop adaptation at different levels

of resources (energy conservation example) and degradation of performance (repository replication and routing adaptation examples). In Section 7.4, we present more complicated policy design in the context of various case studies of wireless networks.

7.3.7.1 Energy Conservation Policy with Cluster-wide Enforcement Scope

A major issue in MANET is the conservation of device resources. We tackle this by introducing a policy type that adaptively configures energy consumption according to their current state and environment as well as the overall management objectives:

```
{CN}[T] if {BP=(n..m)} then {TransPow:=k}
```

This policy type is used to manage effectively the device resources by influencing relevant configuration parameters. The Battery Power (BP) context is used here to affect the node's transmission power (TransPow). In implementation k =1,2, where 1=Normal Power and 2=Low Power, so two policies are implemented:

```
{CN}[bp_event] if {BP=(00..33]} then {TransPow:= 2:Low}
{CN}[bp_event] if {BP=(33..99]} then {TransPow:= 1:Normal}
```

The idea is to use a threshold average battery level in order to reduce transmission power and conserve remaining battery power. Policies of this type only need cluster-wide context knowledge since their enforcement is independent among clusters. The PDP of every CH receives context information for the registered variables and enforces the actions to the PEP of the cluster CN. Periodic receipt of individual BP context subsequently generates periodic bp-event, causing the evaluation of the two conditions and triggering of respective actions. In these cases, context information is withheld within the cluster, thus reducing overall traffic load and processing resources.

The effect of this policy is battery power conservation, since one of the main energy consumers of mobile devices is actually their wireless transceiver. This policy is better suited for dense network deployments to avoid node disconnection with their CH. The reduction of transmission power causes a reduction of transmission range that may result in one way link breaks from CN to CH as well as two-way link breaks between CN. To anticipate potential disconnection, additional conditions may be added to the policy, depending on network deployment parameters.

7.3.7.2 Repository Replication Policy with Hypercluster-wide Enforcement Scope

The need for repository replication has already been explained in Section 7.3.5. For this purpose we model a policy type to guide the replication degree of the Distributed Policy Repository. A manager node has the ability to dynamically define the behavior and the replication degree of the DPR by introducing related policies on the fly and without disrupting system's operation or the DPR component:

```
{MN,CH}[T] if {FM=(n..m)} then {ReplDegState:=k}
```

The above policy type is used to guide the replication degree (ReplDegState) of the Distributed PR (DPR) component. The fluidity metric (FM) is a hypercluster-wide aggregated context that represents how volatile the network is. Three states of replication are implemented, namely k=1:Single, k=2:Selective and k=3 Full. These states reflect the need for PR replicas within the hypercluster nodes and adapt according to the volatility of the MANET (Fig. 7.13). As mentioned earlier, the idea is to increase the DPR replication degree when network fluidity increases, hence the three policies below:

```
{MN,CH}[fm_event] if {FM=[00..25)} then {ReplDegState:=1:Single}
{MN,CH}[fm_event] if {FM=[25..70)} then {ReplDegState:=2:Select}
{MN,CH}[fm_event] if {FM=[70..99)} then {ReplDegState:=3:Full}
```

Based on the collected hypercluster-wide information (in this case the FM), the CDP of each CH informs the collocated PDP and policies of this type

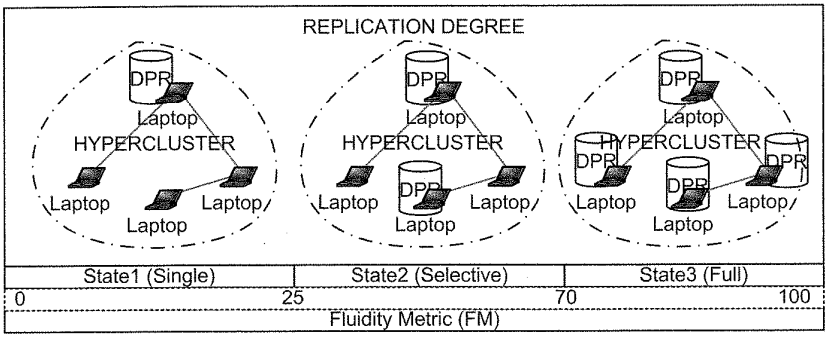


FIGURE 7.13: Replication degrees depending on network fluidity

may be triggered for hypercluster-wide enforcement. Once triggered, their respective actions are enforced only to PEP of hypercluster nodes. The formed adaptation loop ensures the correct replication state is enforced depending on perceived network fluidity among hypercluster nodes. The importance of a reliable and robust DPR has motivated our decision for further analysis of its Self-configuration. Based on the described concepts, an implementation case study is presented in Section 7.4.2, dealing with actual policy provisioning and enforcement. By exploiting LDAP synchronization features, the DPR concept is extended to combine a priori knowledge of localized events (e.g., scheduled sport event) with dynamic real-time context information (e.g., processing load or free memory of each PDP). Thus, a highly customizable deployment of a DPR overlay can be formed.

Based on the above concepts, an implementation case study is presented in Section 7.4. By exploiting LDAP synchronization features, the DPR concept is extended to combine a priori knowledge of localized events (e.g., scheduled sport event) with dynamic real-time context information (e.g., processing load or free memory of each PDP) and a highly customizable deployment of the DPR overlay can be formed.

7.3.7.3 Routing Adaptation Policy with Network-wide Enforcement Scope

A plethora of protocols has been proposed to solve the multihop routing problem in MANETs. A generic classification can distinguish them into proactive and reactive regarding the strategy used to establish routes between nodes. Based on the above, we model a policy type which would enable dynamic on the fly adaptation of the routing protocol. Network conditions/context on one hand and manager defined goals on the other, can both

be expressed by this type of policy which effectively alters routing strategy and increases network performance:

```
{MN,CH,TN}[T] if {RM=(n..m)} then {RoutProt:=k}
```

The above policy type is used to adapt network behavior by switching the routing protocol (RoutProt) according to the network's relative mobility (RM). RM is aggregated context information extracted from the network-wide knowledge of node movements, e.g., GPS positioning data, mobility ratio or other context. The simple condition monitors if RM value lies within the range (n..m) in order to enforce the associated action that activates the appropriate routing protocol. For implementation, the idea is to use a proactive routing protocol (OLSR, k=1) when relative mobility is low and a reactive (AODV, k=2) when high. Depending on management goals and on network-wide aggregated context, compound conditions and actions can be introduced in all the proposed policy types, in order to take more parameters into account. Two policies can enforce the described management goals:

```
{MN,CH,TN}[rm_event] if {RM=[00..35)} then {RoutProt:=1:OLSR }
{MN,CH,TN}[rm_event] if {RM=[35..99]} then {RoutProt:=2:AODV}
```

The network-wide enforcement scope of this policy implies that the condition variables used (e.g., RM) should have an aggregated network-wide value. The RM value for example is extracted from the gradual aggregation and processing of simpler node context (e.g., speed) to cluster context and eventually network context. Cluster context is collected at the Context Management Tool (CMT) components of the Manager Nodes and this allows them to compose the network-wide context variables. This higher level context information will drive the triggering of actions that should be enforced globally. Each CMT forwards this value to the local PDP and to the PDPs of all CHs they control. Each PDP enforces the triggered action to all cluster nodes, including itself, and reports successful execution to their MNs. These actions ensure the smooth and controlled execution of network-wide adaptation, in a self-managing manner.

7.4 Implementation and Evaluation of Self-Management Capabilities

Research on autonomic systems has been intense during the past years, aiming to embed highly desirable self-managing properties to existing and future networks. In previous Sections we have thoroughly examined the necessary components to build a scalable framework for managing wireless ad hoc networks, integrating the notion of IBM's autonomic manager to form a

Knowledge-based loop to Monitor, Analyze, Plan and Execute (K-MAPE). In this Section, we elaborate how these components can be used to realize and implement self-* properties:

- Self-Configuration
- Self-Healing
- Self-Optimization
- Self-Protection

A complete self-management solution is not available to the best of our knowledge. Instead, researchers have attempted to partially tackle self-management by implementing some of the desired properties. In the following sections we present attempts to integrate self-* capabilities to the aforementioned management framework. The solutions demonstrate how the framework with the interaction of policies and context forms the basis for implementation and evaluation of self-management solutions. Detailed policy specifications are presented and their deployment on real wireless ad hoc testbeds demonstrates important quantifiable results and experiences.

7.4.1 Self-Configuration and Self-Optimization in Wireless Ad Hoc Networks

The presented case study deals with the dynamic configuration of communication channels in a wireless ad hoc network based on IEEE 802.11. The solution first addresses the Self-Configuration of ad hoc network deployment by initiating communications using the best available wireless channel. The second issue addressed is the Self-Optimization of ad hoc wireless communications by evaluating wireless channel conditions and dynamically switching to a new optimal channel.

Currently, in dense deployments of WLANs (e.g., conferences, stadiums) users manually initiate ad hoc networks without relying on any infrastructure support. This results in poor performance and interference problems among WLANs, even regulatory violations in some cases. The deployment of ad hoc networks and their coexistence with managed WLANs has not received enough research interest, since in most cases the assumption is that an interference free area is available and all ad hoc stations communicate using the same channel. This assumption allowed research to focus on inter-station interference and MAC layer performance, yielding useful conclusions. On the other hand, industrial interest has been limited, mainly due to the lack of a compelling business model.

The described case study of wireless ad hoc networks is suitable to fully exploit the benefits of the aforementioned policy-based context-aware framework. We design the policies and algorithms necessary for the deployment of such networks and evaluate their performance and applicability through

testbed implementation. By making appropriate policies available in the Distributed Policy Repository, user devices are assisted by receiving guidelines that transparently configure the ad hoc network, choosing the best available wireless channel to avoid interference and dynamically switching channels if performance degrades. This is an important first step towards the implementation of fully self-managing systems, since the presented solution effectively addresses the Self-Configuration and Self-Optimization need of channel assignment in wireless ad hoc networks.

We argue that by facilitating a predictable and controlled ad hoc network deployment, the performance of both managed WLAN and ad hoc networks can be significantly improved. The solution can be deployed on top of existing and future access networks using a technology-independent policy-based and context-aware management layer. The approach spans among different architecture layers of the protocol stack, exploiting context and cross layer principles but at the same time preserving the layers modularity. This paradigm was deemed necessary, since the applicability domain of ad hoc networks is based on a majority of off-the-shelf end-user devices and only a few special purpose devices, e.g., mesh routers or programmable access points. In addition, standards conformance is important for any solution to be applicable.

Inter-layer communication is used between MAC and Application layers, aiming to make the PBM system aware of the wireless channel conditions. This specialized context collection method provides a feedback mechanism for policies. Based on specified application events (e.g., reduced goodput), the triggered policies can initiate relevant procedures that with the inspection of MAC layer headers provide feedback to the system and possibly trigger further policies to correct the problem or report unresolved issues to the user or the network manager. As already explained in Section 7.3.6, a closed control loop is formed that adds a degree of self-management to the network. There are two important advantages with the adoption of this approach:

1. By using a policy-based design, the system is highly extensible and easily configurable. Policies can change dynamically and independently of the underlying technology.

2. By implementing decision logic at the Application layer, based on policies and inter-layer context extracted from lower layers, modularity is preserved without modifying the MAC protocol.

As mentioned already, today the deployment of ad hoc networks is becoming a popular and convenient solution for quick network setup and spontaneous or opportunistic networking. Unfortunately, user experiences have been disappointing, mostly because of difficulties in setup and poor performance. We have identified two potential obstacles that need to be overcome in order to make the deployment of ad hoc networks easy, efficient and safe:

1. interference between newly created ad hoc networks and existing WLAN

2. regulatory conformance of ad hoc networks deployment.

End-users have no need to be aware of channels and regulations, as long as they are connecting to infrastructure-based WLAN, regardless of their geographic area. In managed WLAN, devices connect to infrastructure-based wireless Access Points and automatically adjust to the correct channel, thus reducing the probability of misconfiguration. The problems described are bound to ad hoc networks, since it is up to the initiating device to select a channel for deployment. In addition, it would be useful to ensure that roaming users are conforming to regional regulations with minimal inconvenience. We attempt to propose solutions to the above problems based on the designed policies for a policy-based management system (Table 7.2).

1. Interference between ad hoc and WLAN networks
Interference between deployed ad hoc networks and existing infrastructure-based WLAN, as well as interference with already deployed ad hoc networks in the same area is the main reason for the disappointing performance of ad hoc networks and it can lead to severe problems in the throughput and coverage of collocated infrastructure-based WLAN. As already mentioned, devices operating in ISM bands can arbitrarily use any of the defined channels and should be able to cope with interference from devices competing to access the same unlicensed bands. The MAC layer can be fairly tolerant against interference and noise at the cost of speed and performance. Choosing a random channel is likely to have a detriment effect for the ad hoc network performance. The above problem has been verified by testbed measurements. To tackle this problem, we design policies P1 to P8 (Table 7.2) that exploit context extracted from MAC layer information, first for the initial configuration and secondly for the dynamic adaptation of the deployed wireless channel

2. Regulatory conformance of ad hoc networks deployment
Although this issue is rarely addressed, it is indirectly affecting the popularity and usability of ad hoc networks. Users attempting to deploy ad hoc networks may be breaking the law, especially if their devices have been configured with the default settings of a different geographic area than their current. For example, the regulatory domain of Japan allows the use of all 14 defined channels of the 802.11bg standards for the deployment of WLAN. For most devices used in this region, the default channel for ad hoc deployment is channel 14. However, the rest of the regulatory domains, e.g., Europe or Americas, explicitly forbid the use of channel 14 by WLAN. In the Americas, channels 12 and 13 are also forbidden, adding to the confusion of ad hoc network users. To prevent such problems, additional policies (Table 7.2:P9,10) are introduced by the regional network managers, which in turn influence the criteria for the policy-based channel selection described above (Table 7.2:P2,3,4,8). For example, for P9 list1 = 1..11 and for P10 list2 = 1..13

To illustrate our proposed solution we investigated wireless networks based on IEEE 802.11 standards, since it is the most widely deployed technology for WLAN and offers support for ad hoc networks. Let us assume that a user

Table 7.2: Wireless ad hoc networks self-management policies

P#	*Event*	if $\{Conditions\}$ then $\{Actions\}$
1	Init_new_adhoc	if $\{ready\}$ then $\{scanChannels()\}$, $\{generateScanComplete(results)\}$
2	ScanComplete(results)	if $\{otherWLANdetected = true\}$ $\hat{}\{FC := freeChannels(results), FC = true\}$ $\hat{}\{PC := preffered(FC, ch\_list), PC = true\}$ then $\{optimizeChannel(PC, algorithm_1(criteria_1))\}$
3	ScanComplete(results)	if $\{otherWLANdetected = true\}$ $\hat{}\{FC := freeChannels(results), FC = true\}$ $\hat{}\{PC := preffered(FC, ch_list), PC = false\}$ then $\{optimizeChannel(FC, algorithm_2(criteria_2))\}$
4	ScanComplete(results)	if $\{otherWLANdetected = true\}$ $\hat{}\{FC := freeChannels(results), FC = false\}$ then $\{optimizeChannel(all, algorithm_3(criteria_3))\}$
5	NewWLANdetected	if $\{dyn\_adapt = true\}$ then $\{generateStartAdapt(newWLANinfo)\}$
6	LinkQualityCheck	if $\{LinkQuality < thr_a\}$ $\hat{}\{dyn\_adapt = true\}$ then $\{generateStartAdapt(cachedWLANinfo)\}$
7	StartAdapt(WLANinfo)	if $\{channel\_distance(WLANinfo, current) < dist\}$ $\hat{}\{app\_specific\_metric < thr_b\}$ then $\{scanChannels()\}$, $\{generateAdaptChannel(results)\}$
8	AdaptChannel(results)	if $\{results\_evaluation() = true\}$ then $\{channel\_switch(all, algorithm_4(criteria_4))\}$, $\{verify\_switch()\}$
9	SystemBoot	if $\{region = FCC\}$ then $\{set\_criteria(approvedChannels[list_1])\}$
10	SystemBoot	if $\{region = EU\}$ then $\{set\_criteria(approvedChannels[list_2])\}$

initiates an ad hoc network using a device supporting 802.11bg. The device is set in IBSS mode (Independent Basic Service Set or ad-hoc/peer-to-peer mode) and device-dependent software and hardware configure the transmission parameters. The device assumes the role of the wireless Access Point and its wireless interface begins to emit beacon messages advertising the existence of an ad hoc network on the statically defined channel. Other parameters are also advertised, like the beaconing interval and any encryption methods used, thus enabling nearby devices to join the ad hoc network in a peer-to-peer manner. If we realistically assume deployment in a populated area and not in an anechoic chamber, such deployment would imply the coexistence of various WLAN (either ad hoc or infrastructure-based) and inevitably their interference. Choosing the default channel or even a random channel is likely to have a detrimental effect for the ad hoc network performance. Unwanted side effects will also be noticed in the operation of nearby infrastructure WLAN or ad hoc networks. The problems arise from the access to the wireless medium and three cases can be identified during the deployment of an ad hoc network on a channel: a) the channel is already in use by other WLAN

b) adjacent or nearby channels are in use by other WLAN
c) no nearby channels in use by other WLAN.
In practice, cases b) and c) are difficult to be separated since co-channel interference depends on unpredictable environmental factors and is also technology dependent.

The above cases were examined on an experimental testbed and measurements were taken. We have deployed a policy-based solution that aims to dynamically assign the best available channel and autonomously adapt to changes in the wireless environment. To prevent the detrimental effects of interference, context information extracted from the headers of Layer 2 frames was used. This can be achieved by two methods and either can be used depending on the scenario and hardware support:

1. The device is using the wireless interface to passively monitor all packets it can hear (also know as sniffing or rf-monitor) and forwards them to the monitoring policies for processing of the 802.11 MAC headers as well as the 802.3 Link Layer headers. Therefore the device can extract useful information about the MAC layer performance for its one hop neighbors and by processing this information can trigger appropriate adaptation policies. The advantage of this method is that it fully exploits management frames and headers of 802.11 without associating to any AP or network. If the device has more than one wireless interface it can also assess its own performance. The drawback of this method is that the monitoring interface cannot be used for communication.

2. The device is using the wireless interface in promiscuous mode and associates to a wireless network as normal. The traffic packets received by the device are examined and information can be extracted from them. In this case not all packets transmitted on the channel are captured, since the device cannot overhear the channel while transmitting. This may be a drawback since the device cannot have a complete view of the neighborhood and may continue to cause interference to other devices without being able to detect that. However, the apparent advantage is that the device can still use the interface for communication, which is important in the case of devices with a single wireless interface.

In order to assess the performance of our policy-based approach we used a wireless testbed to evaluate the implementation's performance. In addition, we used the testbed to measure the effects of interference between devices using the same channel or devices with varying channel distance. Experiments were performed in a confined indoor space, matching the typical conditions of the described case studies.

Our experimental testbed consists of 10 nodes: 2 laptops, 4 PDAs and 4 Internet Tablets. All devices are equipped with internal 802.11b wireless interfaces, while the two laptops have an additional PCMCIA external wireless card. Table 7.3 includes more information on the used equipment. For the

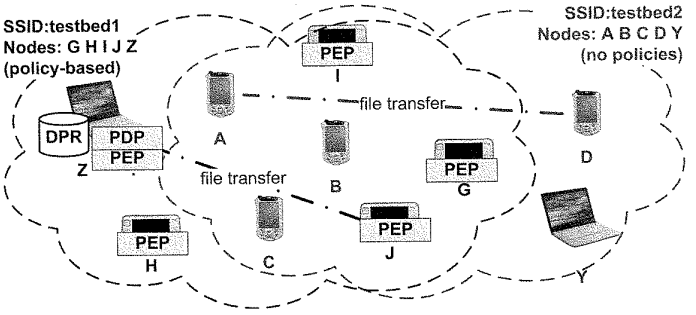


FIGURE 7.14: Wireless ad hoc network testbed deployment

configuration of the wireless interfaces, Linux scripts were used with wireless-tools v28. For monitoring the wireless channel we have modified the source code of airodump-ng, a popular open source 802.11 packet sniffer, part of the aircrack-ng suite [73]. The modifications allowed us to view and dynamically use the captured information within the policy-based interface. Communication between nodes was done either by SSH or by HTTP.

For the purpose of our experiments, the devices were organized in two independent clusters of five nodes as seen in Fig. 7.14. The clusters were set up using different SSID (Service Set Identifiers) in IBSS (ad hoc) mode. The manufacturer's default channel for ad hoc networks creation was found to be Channel 1 (2412Mhz). The network speed (rate) was set to 11Mbps, to allow comparable results among nodes. One of the clusters (testbed1) integrated context-aware policy-based management (PBM) support and the cluster head deployed a PDP for the needs of its cluster. After the PDP had retrieved policies 1 - 8 (Table 7.2) from the nearest DPR, it had accordingly instantiated policy objects (PO) for the monitoring and enforcement of decisions among cluster nodes. For evaluation purposes the PBM support was selectively used to measure its effect on the network performance.

7.4.1.1 Self-configuration for Initial Channel Assignment

In the beginning, we performed static measurements of the channel performance in the presence of multiple ad hoc networks with varying channel distance. According to this scenario, the two clusters would simultaneously attempt to initiate file transfer among peers of the same cluster, as shown in Fig. 7.14. First, the two ad hoc networks were formed on the same default channel (Channel 1). This was possible by using different network names (SSIDs), namely 'testbed1' and 'testbed2.' Afterwards, the same networks were deployed in different channels and file transfers were performed. While

'testbed2' was always deployed on the default Channel 1, 'testbed1' was deployed on Channels 1,2,4 and 6 to vary channel distances and evaluate its effect.

Table 7.3: Wireless testbed specifications

	Operat.System (Linux Kernel)	Processor (MHz -family)	Ram (MB)	Wifi support
Sony Vaio Z1XMP	Debian R4.0 (2.6.18)	1500 - Intel	512	802.11bg
HP iPAQ H5550	Familiar v0.8.4 (2.4.19)	400 - ARM	128	802.11b
Nokia N800	IT OS2007 (2.6.18)	330 - ARM	128	802.11bg

The Cluster Node J of cluster 'testbed1' downloaded a media file from Cluster Head Z and measured the received data download throughput (goodput) and download completion times. The results of the average goodput for each channel combination (T1,T2) are shown in Table 7.4, where T1 the deployment channel of 'testbed1' and T2 that of 'testbed2.' What is worth noticing is that the goodput performance of ad hoc deployment in consecutive channels is even worse than deployment on the same channel by approximately 13%. This can be explained by considering the MAC layer functionality, where on the same channel, all devices hear Request To Sent (RTS) frames and backoff from using the channel and thus can avoid collisions and excessive MAC frames retransmissions. On the contrary, when nearby channels are used, frames from different channels are perceived as interference and increased channel noise, causing the MAC layer to retransmit lost frames and possibly reduce transmission rate to avoid excessive BER. As recorded by our measurements this effect is reduced the furthest apart the channels are, although is still noticeable even when 'non-overlapping' channels are used (e.g., 6,1). This can be explained because of the proximity of most devices which results in the near-far effect.

Additional measurements of missed and sent frames further confirm the detrimental effects of randomly assigning channels to deployed ad hoc networks. All measurements displayed in Fig.7.15 were taken from the cluster head (node Z) of 'testbed1' using its second wireless interface in rf-monitor mode. The purpose was to verify how the device perceives the wireless channel while transmitting using its first interface.

Two sets of measurements are shown, for deployment and monitoring on channel 1 for same channel deployment (T1,T2)=(1,1) and on channel 2 for consecutive channel deployment (T1,T2)=(2,1). Frame measurements provide a good indication of channel utilization and the level of occurred collisions (missed frames). In brief, two points worth noticing are:
(1) Missed frames are in both cases more than sent frames, but in case (2,1) are increased by approx. 15% compared to case (1,1);

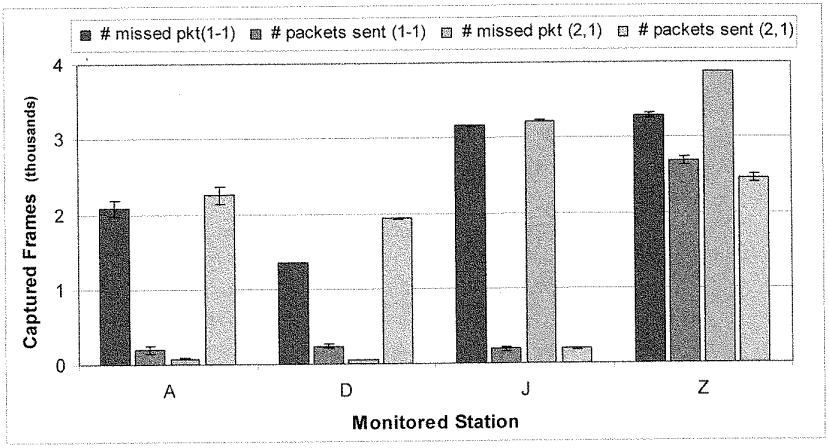


FIGURE 7.15: Packet measurements at node Z for same channel deployment (1,1) and for consecutive channel deployment (2,1)

(2) Node Z can hear a significantly increased number of frames sent from nodes (A,D) of a competing ad hoc network.

By enabling the PBM support for testbed1, the cluster head (node Z) ensures that policies 1-4 are applied during the initial phase of ad hoc deployment. After P1 scanned channels, P2 detects the presence of testbed2 on channel 1 and the scan results indicate channels 2-10 as free (FC=true,PC=true). Since channel 6 of the preferred (non-overlapping) channels list was free, method assignChannel initiates the ad hoc network on the selected channel and the rest of the cluster nodes join using SSID testbed1 on the same frequency. The policy-based initial channel configuration results in the optimum configuration (T1,T2)=(6,1), as confirmed by further measurements. Effectively, the cluster is self-configuring the initial ad hoc deployment and this results in a 20.4% increase of average goodput when compared to using default channels (1,1) and up to 33.3% increase for random channel assignment (2,1). File download completion time is accordingly improved.

7.4.1.2 Self-Optimization for Dynamic Channel Switch

The second implemented scenario investigates the dynamic adaptation of hybrid ad hoc networks to anticipate interference and throughput degradation. Based on the topology of Fig. 7.14, we assume the coexistence of two separate ad hoc networks on the same channel (testbed1 and testbed2 on channel 1). Initially, no traffic transfers are performed between nodes. The

Table 7.4: Initial channel assignment measurements

testbed1,2 (channel)	Goodput testbed1(Mbps)	Goodput decrease (%)	Downl.Time increase (%)
1,1	3.48	-20.38	+20.00
2,1	2.92	-33.27	+46.67
4,1	4.26	-2.68	0.00
6,1	4.38	–	–

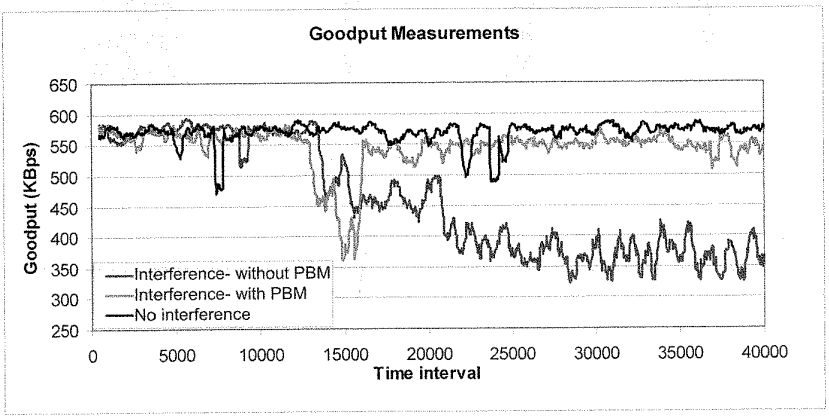


FIGURE 7.16: Policy-based channel assignment measurements

scenario execution has two phases:
Phase 1: ad hoc network testbed1 initiates a file transfer between nodes, with cluster node J downloading a 46MB file from cluster head Z.
Phase 2: ad hoc network testbed2 initiates another file transfer between nodes A and D.

To evaluate the implemented solution, two sets of the described scenario experiments were executed, one set with the PBM solution enabled and enforcing policies 5-8 and another set without any PBM functionality. We have tried to maintain the same execution conditions during all experiments to allow comparison of taken measurements. A representative extract of our measurements is presented (Fig. 7.16) and discussed below.

The measured results demonstrate a significant improvement in network performance when the proposed PBM solution is used (Fig. 7.17). The ad hoc cluster testbed1 is self-optimizing by monitoring events and conditions, resulting in reconfiguration of the transmission channel to avoid interfering WLAN. When the competing ad hoc network (testbed2) initiates a file trans-

fer (phase 2), this results in increased collisions and missed frames for both clusters, which is reflected in reduced Link Quality reported by the wireless interface at node Z. Policy P6, triggered by LinkQualityCheck event, evaluates the moving average of LinkQuality as less than 50% (thr_a) and executes action generateStartAdapt to initiate the adaptation process for channel optimization. In turn, policy P7 is triggered and monitors the specified application metric, in this case the moving average of goodput measurement for the file download between nodes Z and J (app-specific-metric). The measurements of this metric are shown as bold lines in Fig. 7.17 (top), while thin lines show instantaneous goodput measurements in Fig. 7.17 (bottom). Comparing the two graphs of Fig. 7.17, we verify that the use of a moving average smooths goodput fluctuations and prevents false triggering of adaptation policies. Once policy P7 detects the reduction of goodput below 3.67Mbps (thr_b), it acts by scanning the wireless channel, triggering policy P8 and passing scan results (event AdaptChannel). Policy P8 acts by executing channel-switch method using the weighted average algorithm ($algoritm_3$) with specified weights ($criteria_3$). The method indicates that a better channel is available and initiates dynamic switch of ad hoc network testbed1 to channel 6. A channel switch period takes place, causing temporary disconnection of nodes from their cluster head Z. The measurements show that L2 disconnection and connectivity loss occur; however the effect on the ongoing file transfer between J and Z was temporary goodput reduction with a quick recovery to significantly higher goodput. In fact, when compared to the execution without PBM support, the described self-optimization resulted in a 33.5% peak increase of goodput with an average increase of 20.3%. Also, average download time for a 46MB file dropped from 116sec to 50sec.

7.4.2 Self-Configuration of a Distributed Policy Repository

DPR is an enhanced version of the Policy Repository [120] and consists of repository replicas distributed among hypercluster's nodes. Instead of simply replicating the PR among the nodes, we incorporate a sophisticated policy-based replication scheme. By utilizing real-time context information and a priori knowledge, the introduced policies automatically enforce the appropriate replication state among hypercluster nodes.

The Distributed Policy Repository is a set of distributed and/or replicated instances (replicas) of Directory Servers (directories) based on LDAP. Each replica can be either tightly integrated to a master repository instance or loosely coupled to a master or another replica. The design is based on the advanced replication and distribution features of modern LDAP servers. The DPR component is available to nodes consisting of the hypercluster. In order to balance resource consumption and policy accessibility, a selection of the hypercluster nodes activate their DPR component and carry a replica of network policies. The DPR state of each node is imposed by the network policies which define the overall policy replication state. Management objectives reflected

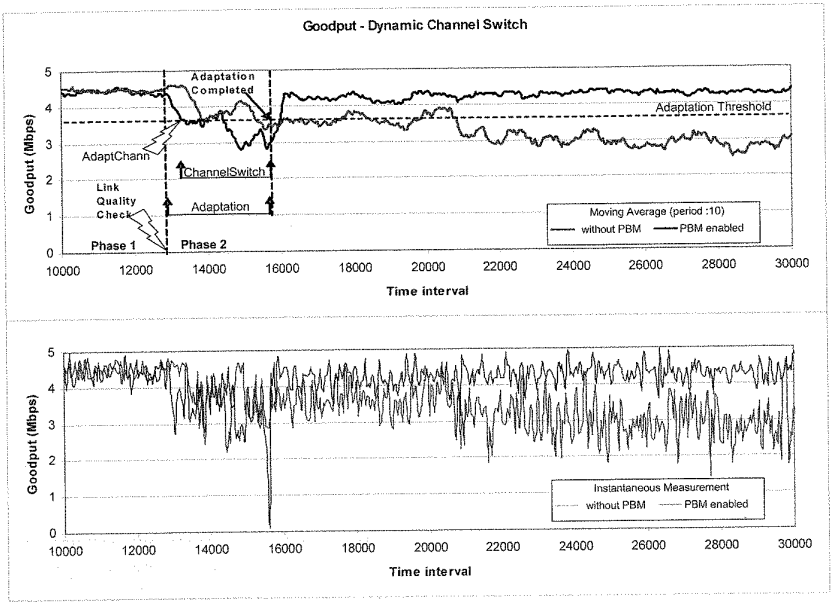


FIGURE 7.17: Testbed measurements of goodput using dynamic channel switch. Top: Moving average, Bottom: Instantaneous

in policies and network conditions influence the DPR replication degree to conserve resources and ensure maximum repository availability.

Specifically, when network mobility is high and links are exceedingly intermittent, reliable access to a remote Policy Repository may be impossible. In this case, policy objects (PO) monitoring network mobility detect the high volatility and proactively increase the replication degree of DPR. Effectively the network will respond with increased decentralization of the policy repository, pushing the storage points (DPRs) closer to the decision points (PDPs). Each manager node (MN) or cluster head (CH) with an active DPR, accommodate a replica of the repository that can serve as an access point for repository requests within their cluster, balancing this way processing load and traffic in the network. A CH with a dormant DPR can access policies from a list of neighboring CHs or MNs with an active DPR. The state of each node is imposed by network policies (Table 7.5).

One of the innovative features of the proposed DPR design is the ability to deploy and maintain special purpose partial replicas of the Policy Repository. These replicas provide a partial view of network policies and can relate to a specific service or location. Accordingly, attached PDPs are responsible only for the enforcement of a policy subset and can be dynamically deployed

Table 7.5: DPR management policies

P	Event	if	{*Conditions*} then {*Actions*}
a	chkDPR	if	$\{t = t_{Weekday}\}\hat{}\{countPDPs(area_1)/countDPRs(area_1) > thr_1\}$
		then	$\{locatePDPs(area_1)\}$,
			$\{selectDPRhost(algorithm_1, context_1)\}$
			$\{deployDPR(all)\}$
b	chkDPR	if	$\{t = t_{Weekend}\}\hat{}\{countPDPs(area_1)/countDPRs(area_1) > thr_2\}$
		then	$\{locatePDPs(area_1)\}$,
			$\{selectDPRhost(algorithm_1, context_1)\}$,
			$\{deployDPR(all)\}$
c	chkDPR	if	$\{t = t_{Kickoff-2h}\}$
			$\hat{}\{countPDPs(stadium_1)/countDPRs(stadium_1) > thr_3\}$
			$\hat{}\{countUsers(stadium_1)/countPDPs(stadium_1) > thr_4\}$
		then	$\{locatePDPs(stadium_1)\}$,
			$\{selectDPRhost(algorithm_2, context_1)\}$,
			$\{deployDPR(service_1, service_2)\}$

to provision time-based events or localized conditions. This feature can be employed when there is a need for localized control in areas with dense user population, such as a conference site or a stadium. In such cases, while node population (i.e., users) increases, the management system can deploy special-purpose DPR replicas and accordingly more Policy Decision Points (PDPs) that will be responsible for the distributed enforcement of specific management tasks. Special policies guide the deployment of partial DPR copies, based on a priori knowledge of localized events (e.g., scheduled sport event) with dynamic real-time context information (e.g., processing load on each CH) (Table 7.5).

For the implementation of the Distributed Policy Repository (DPR) we have used OpenLDAP. This selection was made because it is a free and open source implementation of a very fast and reliable LDAP v3 Directory Server for Linux. In addition, the minimum specifications required for running this server allow an extensive range of devices, including low-spec laptops to efficiently host a directory replica. We will refer to a Directory Server with its directory content (i.e., policies) as a directory. The DPR consists of one or more Master read-write directories and several read-only directory replicas (shadow copies). Master directories are hosted and controlled by the managing network entities, i.e., Network Operator and/or Service Providers. These entities are responsible for providing the overall management objectives and guidelines to the wireless network by specifying appropriate policies. The policies are configured and introduced to the systems using the Policy Management Tool and their LDAP representation is stored in a master directory. Based on replication policies (Table 7.5), selected user devices that serve as Policy Decision Points (PDPs) are also chosen to host a directory replica, i.e., a part of the Distributed Policy Repository. To achieve the above, we exploit OpenLDAP's replication engine to enable the policy-based distribution of replicated read-only directories (shadow copies) among the user devices, as well as partial copies for specific purposes (e.g., policies for multimedia ser-

vices). OpenLDAP implements a Sync replication engine (syncrepl), based on the Content Synchronization Operation (RFC 4533). Syncrepl engine offers client-side (consumer) initiation for replication of all policies or for a custom policy selection, relieving the providing directory (provider) from tracking and updating replicas. This functionality is very useful since the operation of a directory provider is not disrupted by the presence of consumers and can operate even when they are temporarily disconnected because of wireless link intermittence. Upon reconnection, the directory consumers compare their current content with their provider's and retrieve any updates. For the defined policies in Table 7.5, the generic method selectDPRhost(($algorithm_1$,$criteria$) can use different algorithms for the best possible placement of replicated directories. The optimal placement solution is a computationally intensive task, hindered by the distributed nature of wireless systems and is out of the scope of this paper. In [18, 96] we have described and evaluated a distributed algorithm based on context-aware heuristics to form a dominating set of nodes that share management responsibilities. The same approach is adopted for the implementation of algorithm1 for policies in Table 7.5. The criteria parameter affects the heuristics used, by modifying the weights of metrics used in the algorithm. Method deployDPR() is used to set up and initiate a replicated directory, part of the Distributed Policy Repository. First the directory configuration file (slapd.conf) is modified and once the replica is initiated, it connects and retrieves policies from its defined master directory. Method parameters (all, service1, service2) define policy groups that will be replicated.

7.4.3 Self-Protection of User Privacy and Preferences

Another interesting extension of the aforementioned framework touches on the sensitive issue of privacy and preference respect of users. The presented case study introduces a self-protection mechanism for networked devices based on the user's input and current regulations [10].

The multi-manager policy-based framework is employed to establish the high-level management objectives of Network Operators and Service Providers. These objectives are encapsulated in policies and distributed among the wireless network. In order to provide a rich and customizable experience to users, these policies often need to collect user information. The collected context can be anything from their location, movement patterns or frequency of service access. The particular type of context differs significantly from network collected context (e.g., throughput, lost packets, etc.) because it is tightly related to the privacy of the individual user. This privacy needs to be protected. To facilitate a protection mechanism for user's privacy, this case study introduced a twofold scheme to modify the presented framework. As explained here briefly, policies and context are both affected by this scheme. For more details, we refer the reader to [10].

Self-management is a compelling functionality and a highly desired attribute of large-scale complex networks. But as already pointed out, wireless

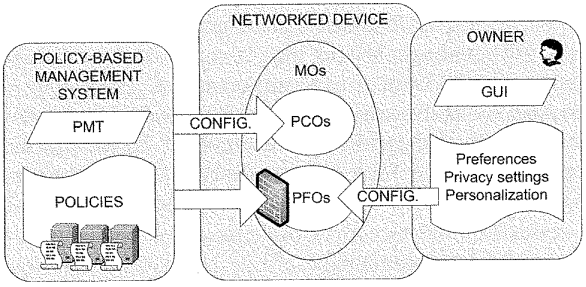


FIGURE 7.18: Policy free and policy conforming objects

networks formed by user devices are not directly managed by an administrative authority and, in addition, often involve processing and collection of user generated personal context. As a first step, the described case study offers the capability to a user to define which high-level managed object (MO) he/she wants to control on his/her own and which he/she entrusts to the PBM system. In addition, users can explicitly specify their own access control rules as preferences and limit the exploitation of specific contextual information. Effectively, this leads to the separation of policy/context objects into Policy-Free Objects (PFO) and Policy-Conforming Objects (PCO). The concept is graphically presented in Fig. 7.18. For example, different policy/context objects can be managed objects controlling access to location data, battery consumption profile, sharing resources, etc. While Services may need to alter those objects, e.g., a Location-based Service (LBS) needs to enable a GPS receiver and access location data, a user may not be willing to allow access to personal context. Hence, using a GUI can set the desired preferences and privacy restrictions to the system.

The second protection mechanism introduced relies on a policy-based regulation scheme. In addition to the explicit user defined preferences, the PBM system has the ability to control unfair exploitation of user data by deploying a regulation scheme with appropriate policies. In a multi-manager case study, we consider a data protection agency (Regulator) that has the control of one Manager Node. Using the PMT interface, the Regulator has the ability to manage the lifecycle of policies and introduce appropriate policies to the managed system according to current regulations. In addition, it can review, edit or disable existing policies so as to ensure users' personal data are not collected or exploited by other entities, e.g., by the Network Operator or a Service Provider.

For example, users who are willing to reveal their location data should be protected from services that can continually track their position. Tracking is possible by frequently polling the user location and comparing consecutive measurements, depending on the accuracy of the available positioning method

and users' speed. With the increased penetration in the consumer market of high accuracy GPS-enabled devices and improvement of indoor positioning methods, this issue is becoming quite important. Further than configuration policies, a regulatory body can use the policy-based system to monitor the collection of user data and gather information for offline processing. Simple policies can periodically log information about the services that retrieve user data. The logged details can be reviewed and analyzed statistically to extract information about how Service Providers use the location data of users and investigate their unfair exploitation.

The twofold protection scheme introduced in [10] is an important step to control the exploitation of context information and offer to users a degree of control. By introducing appropriate regulatory policies and user-oriented access control, the PBM system is armed with self-protection capabilities to prevent the unfair exploitation of participating devices. The privacy concerns raised are important requirements in the design of a context-aware solution. Privacy and control of context dissemination are tightly connected with self-protection and self-healing. As discussed in the following Section, although self-protection and self-healing are critical capabilities of autonomous systems there are limited case studies implementing them.

7.5 Conclusions and the Future of Self-management

7.5.1 Summary and Concluding Remarks

Self-management is gradually becoming a reality and it is expected that in the future more research efforts will be looking towards this direction. We have rigorously presented the history and evolution of self-management through extensive literature review. We have seen that a computing environment with the ability to manage itself and dynamically adapt to change in accordance with business policies and objectives defines autonomic computing and encompasses the fundamental Self-Management definition. In future self-management systems, the characteristic four self-CHOP or self-* properties, namely self-configuration, self-heal, self-optimization and self-protection, will be gracefully integrated and become invisible. But in order to reach this point of integration, these properties are currently separated and studied individually. For the realization of Self-Management capabilities, we have adopted closed-loop control as a repetitive sequence of tasks including monitor, analyze, plan and execute functions. A Knowledge base caters for the orchestration of these functions. This reference model by IBM [2] is frequently referred to as K-MAPE or simply MAPE and has been the basis for the presented system analysis, design and evaluation.

The focus of the work presented in this chapter has been in wireless ad hoc

networks. This is due to the increasing popularity and penetration of such networks worldwide. In order to apply Self-Management to wireless ad hoc networks, we have analyzed the definition for Self-Management and identified policies as the basis of such systems, encapsulating high-level business objectives. Policy-Based Management (PBM) is the first building block of the presented Self-Management framework and policies are its cornerstone. Equally important and complementary is the system's ability to sense and observe its surrounding environments. To enable these, context-awareness is employed as the second building block of the framework. As a result a policy-based context-aware framework is designed, as the foundation for the implementation of self-management properties. The motivation and applicability of the presented framework is evident considering the closed-loop controller (Fig. 7.1) or IBM's autonomic manager (Fig. 7.2).

The policy-based part of the presented framework is analogous to the initial input and processing unit of a closed-loop control system, i.e., input and unit A in Fig. 7.1. At the same time, the context-aware part is analogous to the returned feedback that is processed and closes the adaptation loop. Similarly, the designed framework can be mapped to the widespread K-MAPE reference model of IBM [2]. Policy-based management can serve as the Plan and Execute components of a self-management system, as presented in Fig. 7.2. Policy design and specification constitute the Planning phase of autonomic management while policy enforcement constitute the Execute phase. On the other hand, a context-aware framework is assigned the Monitor and Analyze functionality. Context sensing and collection constitute the Monitoring phase while context aggregation and inference rules constitute the Analyze phase of management. The specification of policies and context, together with their interaction, form the essential Knowledge element. Policy and context repositories are the Knowledge centerpiece of both frameworks, gracefully integrating the presented self-management solution. Figure 7.6 (page 224) illustrates these concepts, in parallel with IBM's autonomic manager.

We believe that the highly dynamic environment of wireless ad hoc networks can benefit from a Policy Based Management (PBM) and context-aware approach. One of the major advantages of adopting a policy-based approach is the relevant 'controlled programmability' that can offer an efficient and balanced solution between strict hard-wired management logic and unrestricted mobile code migration and deployment. While there has been previous research on deploying PBM solutions for wireless ad hoc networks, the work presented here introduced a novel organizational model specifically targeted to the needs of such networks by incorporating context awareness to dynamically adapt to the continuously changing conditions. Context information can be used to trigger cross-layer changes (network and application configurations) according to policies, leading to a degree of autonomic decision-making and self-management. Based on the introduced distributed and hierarchical organizational model, a Distributed Policy Repository (DPR) is deployed to efficiently cater for the policy distribution and provisioning needs of the

network.

A fully operational self-management solution is not currently available to the best of our knowledge. However, researchers have attempted to partially tackle self-management by implementing some of the desired properties. We have presented some notable efforts to integrate self-* capabilities based on the aforementioned policy-based and context-aware management framework. The presented solutions have demonstrated how the framework with the interaction of policies and context forms the basis for implementation and evaluation of self-management solutions. An important consideration regarding the presented research effort is the realistic deployment of ideas on real wireless ad hoc testbeds. Detailed policy specifications and context modeling were implemented and have demonstrated quantifiable performance improvement, as well as lessons and experiences learned.

7.5.2 Future Trends and Challenges

It should be noted that current research on Self-management has mainly focused on addressing the self-configuration and self-optimization properties, while self-healing and self-protection remain in infancy. This is to be expected since the road to Self-management is gradual and complex. Small steps are made each time leading to integration of all properties. For example, presented solutions in Section 7.4 deal with self-configuration and self-optimization of wireless ad hoc networks and only touch on issues of self-protection. According to IBM's roadmap to autonomic computing and self-management [107], five transition levels for gradual adoption are suggested. Today most systems are in either 'Basic' or 'Managed' level, meaning there is a significant amount of human effort in monitoring and controlling operations, assisted by limited management systems. Current research efforts have elevated management to 'Predictive' and 'Adaptive' levels, by integrating automated context correlation and applying high level objectives through manually refined low-level policies.

The ultimate step to 'Autonomic' level has been the focus of self-management research. Major research challenges need to be addressed and resolved before this transition is made. Notably, crucial issues related to policy-based management expected to receive research interest are mentioned here:

- Automation of policy refinement: Today, policy specification languages require a highly technical person to analyze the system and specify the required management objectives as policies. This results in significant effort for setting up the system and writing detailed low-level policies. In addition, policy updates and notifications require same skills and time. It is envisioned that in the future, an automated policy refinement process will take up this time-consuming and difficult task and will automate the creation of low level policies from business objectives. A manager will be able to directly influence and control the system

behavior and always ensure compliance with high level goals.

- Automation of policy analysis: Another crucial issue that needs to be resolved is the analysis of policy specification. This includes the detection of conflicts in policy specification and their resolution. Currently, both issues are manually handled by experienced technical persons, but active research aims to automate these processes. Conflicts can occur either during specification of new or changed policies (static conflicts) or during system run-time because of dynamically changing conditions (dynamic conflicts). Both types can compromise the integrity of the system and disrupt its operation; hence reliable detection and resolution mechanism should be in place, ideally functioning unsupervised.

- Efficient and distributed policy provisioning: As already described, the large scale of managed systems and their highly distributed nature further complicate policy provisioning, i.e., timely providing appropriate enforcement action as required by active policies. Especially in wireless networks, the task is even harder due to mobility and frequent disconnection. Being able to access all or at least the majority of managed devices is important for offering value-added services and generating revenue. While probabilistic management solutions are investigated, p2p computing also offers promising solutions. A highly distributed policy storage facility can ease provisioning, provided it can remain synchronized and up to date. It is expected that this issue will receive intense research efforts in the future.

Paired with policy-based management, Context Awareness will remain an invaluable component of Self-managing systems. More crucial challenges need to be addressed and resolved to achieve truly autonomic computing. Some important open research issues are mentioned here:

- Efficient and secure context modeling: As emphasized earlier, context modeling is important for any context-aware system. It affects the way context is collected and processed and can severely affect the operation of a network. More expressive and efficient models are required to cater for the resource constrained nature of wireless systems, since bandwidth is limited. Future models need to integrate security features to protect and respect the privacy of users.

- Automated extraction of context from heterogeneous devices: With the increasing heterogeneity of devices and networks in general, context collection becomes complicated and fragmented. Overcoming these problems in an automated manner is a challenging research topic. Device and equipment standardization is important for uniform context collection.

- Interoperable context collection and exchange: The current absence of standards further inhibits context extraction and hinders desired interoperability of systems. Software engineering methods and practices

can be utilized to create necessary interfaces between context-aware elements, while ontologies and semantics are important tools to be exploited. In addition, protocols for secure and efficient exchange of contextual information should be engineered, departing from the proprietary solutions used today.

Self-management is expected to motivate significant research efforts, both in academia and in industry, because of the apparent benefits it can offer. Future networks and systems will transparently integrate self-management capabilities, relieving users and managers from painstaking tasks. As research progresses, the current separation of self-* properties in configuration, healing, optimization and protection will diminish, gracefully amalgamating all in a Self-maintaining operation. We envision a policy-based system as a future-proof solution, where business objectives and user preferences will be encapsulated in policies. Context-awareness will provide secure and accurate feedback to the system, assisting in fully customized and personalized user experience. Eventually policies and context will vanish inside systems, allowing users to enjoy truly ubiquitous networking.

7.6 Acknowledgments

Research work in this chapter was partly supported by the EU EMANICS Network of Excellence on the Management of Next Generation Networks (IST-026854). The authors also wish to thank Dr. Apostolos Malatras for his contribution to the context-aware aspects of the framework.

7.7 Abbreviations

Acronym	Explanation
AM	Autonomic Manager
CCP	Context Collection Point
CDP	Context Decision Point
CMT	Context Management Tool
COPS	Common Open Policy Service
CR	Context Repository
DMTF	Distributed Management Task Force
DPR	Distributed Policy Repository
DS	Directory Server
IETF	Internet Engineering Task Force

LDAP	Lightweight Directory Access Protocol
PBM	Policy-Based Management
PCIM	Policy Core Information Model
PCIMe	Policy Core Information Model Extensions
PDP	Policy Decision Points
PEP	Policy Enforcement Point
PMAC	Policy Management for Autonomic Computing
PMT	Policy Management Tool
PR	Policy Repository
UML	Unified Modeling Language
XML	eXtensible Markup Language

References

[1] G.F. Franklin, J.D. Powell, A. Emami-Naeini, Feedback Control of Dynamic Systems, Prentice Hall, ISBN-13:978-0131499300

[2] IBM Inc., An architectural blueprint for autonomic computing, accessed online Dec.2007 http://www-03.ibm.com/autonomic/pdfs/ACBP2_2004-10-04.pdf

[3] J.O. Kephart , D.M. Chess, The vision of autonomic computing, IEEE Computer, Vol.36/Iss.1, Jan. 2003. pp.41-50

[4] D. Chalmers et al., Ubiquitous Computing: Experience, Design and Science, Ver.4, http://www-dse.doc.ic.ac.uk/Projects/UbiNet/GC/Manifesto/manifesto.pdf, Jun.2004

[5] H. Schulzrinne et al., Ubiquitous computing in home networks, IEEE Communications Magazine, Vol.41/Iss.11, Nov.2003

[6] L.M. Feeney, B. Ahlgren, A. Westerlund, Spontaneous networking: an application oriented approach to ad hoc networking, IEEE Communications Magazine, Vol.39/Iss. 6, Jun.2001

[7] J. Latvakoski, D. Pakkala, P. Paakkonen, A communication architecture for spontaneous systems, IEEE Wireless Communications, Vol.11/Iss.3,

[8] M. Sloman, E. Lupu, Policy Specification for Programmable Networks , Proceedings of First International Working Conference on Active Networks (IWAN'99), Berlin, June 1999

[9] D.C. Verma, Simplifying network administration using policy-based management, IEEE Network, Vol.16Iss.2, Mar-Apr.2002

[10] A.M. Hadjiantonis, M. Charalambides, G. Pavlou, A policy-based approach for managing ubiquitous networks in urban spaces, IEEE Int. Conf. on Communications 2007, Glasgow (ICC2007)

[11] R. Boutaba, S. Omari, A. Virk, SELFCON: An Architecture for Self-Configuration of Networks, Journal of Communications and Networks, Vol.3/No.4, pp.317-323, 2001

[12] IETF's Policy Framework Working Group (POLICY WG), concluded in 2004, IETF Website http://www.ietf.org/html.charters/OLD/policy-charter.html Accessed September 2005

[13] D. Verma, Policy-Based Networking, Architecture and Algorithms, Pearson Education, ISBN:1578702267, 2000

[14] P. Flegkas, P. Trimintzios, G. Pavlou, A policy-based quality of service management system for IP DiffServ networks, IEEE Network, Vol.16/Iss.2, March-April 2002, pp.50-56

[15] M. Sloman, Policy driven management for distributed systems, Journal of Network and Systems Management, Vol.2/No.4, Dec. 1994, pp. 333-360, accessed Sep.2005, `http://www-dse.doc.ic.ac.uk/dse-papers/management/pdman.ps.Z`

[16] J. Strassner, Policy-Based Network Management, Solutions for the Next Generation, Morgan Kaufmann, ISBN:1558608591, 2003

[17] R. Chadha et al., Policy Based Mobile Ad hoc Network Management, 5th IEEE Int. Work. on Policies for Distributed Systems and Networks (Policy 2004)

[18] A.M. Hadjiantonis, A. Malatras, G. Pavlou, A context-aware, policy-based framework for the management of MANETs, 7th IEEE Int. Work. on Policies for Distributed Systems and Networks (Policy 2006)

[19] P. Flegkas, P. Trimintzios, G. Pavlou, A. Liotta, Design and implementation of a policy-based resource management architecture, IFIP/IEEE 8th International Symposium on Integrated Network Management, March 2003, pp.215-229

[20] R. Yavatkar et al., A Framework for Policy-based Admission Control, RFC 2753, Informational, Jan.2000

[21] B. Moore et al., Policy Core Information Model-Version 1 Specification, RFC 3060, Standards Track, Feb.2001

[22] B. Moore, Policy Core Information Model (PCIM) Extensions, RFC 3460, Standards Track, Jan.2003

[23] D.C. Verma, Simplifying network administration using policy-based management, IEEE Network, Vol.16/Iss.2, March-April 2002, pp.20-26

[24] Internet Engineering Task Force (IETF), `http://www.ietf.org`

[25] Distributed Management Task Force (DMTF), `http://www.dmtf.org`

[26] IBM Corp., Policy Management for Autonomic Computing, V1.2, accessed on Sep.2007, `http://www.alphaworks.ibm.com/tech/pmac`

[27] D.Chadwick et al., Coordination between distributed PDPs, 7th IEEE Int. Work. on Policies for Distributed Systems and Networks (Policy 2006)

[28] J. Sermersheim, Lightweight Directory Access Protocol (LDAP): The Protocol, RFC4511, Standards Track, Jun.2006

[29] R. Montanari, E. Lupu, C. Stefanelli, Policy-based dynamic reconfiguration of mobile-code applications, IEEE Computer, Vol.37/Iss.7, Jul.2004, pp.73-80

[30] IETF's Mobile Ad-hoc Networks Working Group (MANET WG), IETF Website, accessed September 2005. `http://www.ietf.org/html.charters/manet-charter.html`

[31] W. Chen, N. Jain, S. Singh, ANMP Ad hoc network management protocol, IEEE Journal on Selected Areas in Communications, Vol.17/Iss.8, Aug. 1999, pp.1506-1531

[32] R. Chadha, C. Yuu-Heng, J. Chiang, G. Levin, L. Shih-Wei, A. Poylisher, Policy-based mobile ad hoc network management for drama, IEEE Military Communications Conf. (MILCOM 2004), Vol.3, pp.1317-1323

[33] R. Badonnel, R. State, O. Festor, Management of mobile ad-hoc networks: evaluating the network behaviour, 9th IFIP/IEEE International Symposium on Integrated Network Management (IM2005), May 2005 pp.17-30

[34] C. Shen, C. Srisathapornphat, C. Jaikaeo, An adaptive management architecture for ad hoc networks, IEEE Communication Magazine, Vol.41/ Iss.2, Feb.2003, pp.108-115

[35] A. Westerinen et al., Terminology for Policy-Based Management, RFC 3198, Informational, Nov.2001

[36] M. Wahl et al., Lightweight Directory Access Protocol (v3), RFC 2251, Standards Track, Dec.1997

[37] K. Phanse, Policy-Based Quality of Service Management in Wireless Ad-hoc Networks. PhD thesis, Faculty of the Virginia Polytechnic Institute and State University, August 2003.

[38] M. Burgess, G. Canright, Scalability of peer configuration management in logically ad hoc networks, eTransactions on Network and Service Management, Vol.1 No.1 Second Quarter 2004

[39] C.S. Murthy , B.S. Manoj, Ad Hoc Wireless Networks, Architectures and protocols, Prentice Hall PTR, ISBN:013147023X, 2004

[40] L.M. Feeney, B. Ahlgren, A. Westerlund, Spontaneous networking: an application oriented approach to ad hoc networking, IEEE Communications Magazine, Vol.39/Iss.6, June 2001, pp.176-181

[41] M. Sloman, E. Lupu, Policy Specification for Programmable Networks, Proceedings of First International Working Conference on Active Networks (IWAN'99), Berlin, June 1999

[42] N. Damianou, N. Dulay, E. Lupu, M. Sloman, The Ponder Specification Language, Workshop on Policies for Distributed Systems and Networks (Policy 2001), Bristol, Jan 2001

[43] IETF's Resource Allocation Protocol Charter (RAP WG, concluded), IETF website, accessed Sep.2005,`http://www.ietf.org/html.charters/OLD/rap-charter.html`

[44] D. Durham et al., The COPS (Common Open Policy Service) Protocol, RFC 2748, Standards Track, Jan.2000

[45] K. Chan et al., COPS Usage for Policy Provisioning (COPS-PR), RFC 3084, Standards Track, Mar.2001

[46] M. Sloman, E. Lupu, Security and management policy specification, IEEE Network, Special Issue on Policy-Based Networking, Vol.16/Iss.2, pp.10-19

[47] A. Westerinen, J.Schott, Implementation of the CIM Policy Model using PONDER, 5th IEEE Int. Work. on Policies for Distributed Systems and Networks, POLICY 2004, 7-9 June 2004 pp.207-210

[48] DMTF website, CIM Policy Model White Paper for CIM v2.7.0 , accessed September 2005 `http://www.dmtf.org/standards/documents/CIM/DSP0108.pdf`

[49] N. Dulay, E. Lupu, M. Sloman, N. Damianou, A policy deployment model for the Ponder language, IEEE/IFIP Int. Symposium on Integrated Network Management, May 2001 pp.529-543

[50] L. Lymberopoulos, E. Lupu , M. Sloman, Using CIM to realize policy validation within Ponder Framework, DMTF Website, accessed September 2005 `http://www.dmtf.org/education/academicalliance/lymberopoulos.pdf`

[51] G. Pavlou, P. Flegkas, S. Gouveris, A. Liotta, On management technologies and the potential of Web services, IEEE Communications Magazine, Vol.42/Iss.7, July 2004, pp.58-66

[52] The Directory: Overview of Concepts, Models and Services, Recommendation X500, ISO/IEC 9594-1 , ITU-T

[53] V. Koutsonikola, A. Vakali, LDAP: framework, practices, and trends, IEEE Internet Computing, Vol.8/Iss.5, 2004, pp.66-72

[54] Sun website, A Technical Overview of the Sun ONE Directory Server 5.2, accessed Feb. 2005 `http://www.sun.com/software/products/directory_srvr_ee/wp_directorysrvr52_techoverview.pdf`

[55] E.J. Thornton, D. Mundy, D.W. Chadwick, A comparative performance analysis of seven LDAP Directories, TERENA website accessed Septem-

ber 2005 www.terena.nl/conferences/tnc2003/programme/papers/p1d1.pdf

[56] T.A. Howes, M.C. Smith, S.G. Gordon, Understanding and deploying LDAP directory services, 2nd edition, Addison-Wesley Professional ISBN: 0672323168, 2003

[57] W. Dixon, T. Kiehl , B. Smith, M. Callahan, An Analysis of LDAP Performance Characteristics, General Electric Global Research, Technical Information Series, tech. report TR-2002GRC154, Jun. 2002

[58] X. Wang, H. Schulzrinne , D. Kandlur , D. Verma, Measurement and Analysis of LDAP Performance, Int. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS'2000), Santa Clara, CA, Jun. 2000, pp. 156-165

[59] Organization for the Advancement of Structured Information Standards, Directory Services Markup Language v2.0, OASIS Standard 2002, accessed Sep. 2005 www.oasis-open.org/committees/dsml/docs/DSMLv2.doc

[60] J. Strassner et al., Policy Core Lightweight Directory Access Protocol (LDAP) Schema, RFC3703, Standards Track, Feb.2004

[61] M. Pana et al., Policy Core Extension Lightweight Directory Access Protocol Schema (PCELS), RFC 4104, Standards Track, Jun.2005

[62] A. Vakali, B. Catania, A. Maddalena, XML Data Stores: Emerging Practices, IEEE Internet Computing, Vol.9/Iss.2, 2005, pp.62-69

[63] A. Matheus, How to Declare Access Control Policies for XML Structured Information Objects using OASIS' eXtensible Access Control Markup Language (XACML), 38th Annual Hawaii Int. Conf. on System Sciences 2005, HICSS'05, Jan.2005

[64] M. Lorch, D. Kafura, S. Shah, An XACML-based policy management and authorization service for globus resources, 4th Int. Work. on Grid Computing, Nov. 2003, pp.208 - 210

[65] J. Schoenwalder, A. Pras, J.-P. Martin-Flatin, On the future of Internet management technologies, IEEE Communications Magazine, Vol.41/Iss.10, 2003, pp.90-97

[66] R. Sahita, COPS Protocol Provides New Way of Delivering Services on the Network, Intel website,accessed Sep.2005 http://www.intel.com/technology/magazine/communications/nc05021.pdf

[67] The openLDAP Foundation, http://www.openldap.org

[68] Network Simulator v2, accessed Sep.2005 http://www.isi.edu/nsnam/ns/ns-build.html

[69] T. F. Franco et al., Substituting COPS-PR: an evaluation of NETCONF and SOAP for policy provisioning, 7th IEEE Int. Workshop on Policies for Distributed Systems and Networks (Policy 2006)

[70] R. Badonnel, R. State, O. Festor, Probabilistic Management of Ad-Hoc Networks, IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)

[71] G. Pavlou, A. Hadjiantonis, A.Malatras (editors), EMANICS NoE Deliverable D9.1. Frameworks and Approaches for Autonomic Management of Fixed QoS-enabled and Ad Hoc Networks, online, Dec. 2006, `http://emanics.org/component/option,com_remository/Itemid,97/func,fileinfo/id,47/`

[72] A. Malatras, Context-Awareness for the Self-Management of Mobile Ad Hoc Networks, PhD Thesis, Univ. of Surrey, 2007

[73] Aircrack-ng WLAN Tools, `http://www.aircrack-ng.org`

[74] A. K. Dey, , Understanding and using context, Journal of Personal and Ubiquitous Computing, Vol.5/Iss.1, pp.4-7, 2001

[75] A. K. Dey, G. D. Abowd, Towards a better understanding of context and context awareness, ACM Conference on Human Factors in Computer Systems (CHI 2000), April 2000

[76] A. Malatras, G. Pavlou, S. Gouveris, S. Sivavakeesar, V. Karakoidas, Self-Configuring and Optimizing Mobile Ad Hoc Networks, IEEE Int. Conf. on Autonomic Computing (ICAC'05), 2005

[77] A. Malatras, G. Pavlou, Context-driven Self-Configuration of Mobile Ad hoc Networks, IFIP Int. Work. on Autonomic Communications (WAC 2005), October 2005

[78] A. Malatras, S. Gouveris, S. Sivavakeesar, G. Pavlou, Programmable Context-Aware Middleware for the Dynamic Deployment of Services and Protocols in Ad Hoc Networks, 12th Annual HP-OVUA Workshop, 2005

[79] K. Henricksen, J. Indulska, A. Rakotonirainy, Generating Context Management Infrastructure from High-Level Context Models, 4th Int. Conf. Mobile Data Management (MDM2003) Jan. 2003, pp.1-6

[80] K. Henricksen, J. Indulska, A.Rakotomirainy, Modeling context information in pervasive computing systems. LNCS 2414, 1st Int. Conf. on Pervasive Computing (Switzerland), 2002, pp.167-180

[81] J.M. Serrano, J. Serrat, Context Modeling and Handling In Context-Aware Multimedia Applications, IEEE Wireless Communications Magazine 2006, Special Issue on Multimedia in Wireless/Mobile Ad-hoc Networks, October 2006

[82] B. N. Schilit, N. L. Adams, R. Want, Context-aware computing applications. IEEE Work. on Mobile Computing Systems and Applications, 1994

[83] A. Held, S. Buchholz, A. Schill, Modeling of context information for pervasive computing applications, SCI 2002/ISAS 2002

[84] ESPIRIT PROJECT 26900: Technology for enabled awareness (tea), 1998.

[85] GUIDE Project: Understanding Daily Life via Auto-Identification and Statistics http://seattleweb.intel-research.net/projects/guide/

[86] J. McCarthy, Notes on formalizing contexts, 13th Int. Joint Conf. on Artificial Intelligence, pp.555-560

[87] P. Otzurk, A. Aamodt, Towards a model of context for case-based diagnostic problem solving, Interdisciplinary conf. on modeling and using context(Rio de Janeiro, February 1997), pp. 198-208

[88] H. Chen, T. Finin, A. Joshi, Using OWL in a Pervasive Computing Broker, Work. on Ontologies in Open Agent Systems (AAMAS 2003)

[89] M. Conti, S. Giordano, Multihop Ad Hoc Networking: The Reality, IEEE Communications Magazine, Vol.45/Iss.4,2007, pp.88-95

[90] P. Gupta, P. R. Kumar, The capacity of wireless networks, IEEE Transactions on Information Theory, Vol.46,2000, pp. 388-404

[91] A. Gostev, R. Schouwenberg, War-driving in Germany - CeBIT2006 accessed Oct.2007, http://www.viruslist.com/analysis?pubid=182068392

[92] M. Burgess, G. Canright, Scalability of peer configuration management in logically ad hoc networks, eTransactions on Network and Service Management, Vol.1/No.1, 2nd Q. 2004

[93] IEEE 802.11 WLAN Working Group, accessed Sep.2007 http://grouper.ieee.org/groups/802/11/

[94] A. Jayasuriya et al., Hidden vs. Exposed Terminal Problem in Ad hoc Networks, Proc. of Australian Telec. Net. and App. Conf., Dec 2004

[95] Cisco Systems, Channel Deployment Issues for 2.4-GHz 802.11 WLANs, accessed on Sep.2007, http://www.cisco.com

[96] A. Malatras, A.M. Hadjiantonis, G. Pavlou, Exploiting Context-awareness for the Autonomic Management of Mobile Ad Hoc Networks, Journal of Network and System Management, Special Issue on Autonomic Pervasive and Context-aware Systems, Vol.15/Iss.1, Mar.2007

[97] OpenLDAP Foundation, OpenLDAP 2.3 Directory Services, `http://www.openldap.org`

[98] K. Zeilenga, J. H. Choi, The Lightweight Directory Access Protocol Content Synchronization Operation, RFC4533, Experimental, Jun.2006

[99] R. Bruno,M. Conti, E. Gregori, Mesh networks:commodity multihop ad-hoc networks, IEEE Communications Magazine, Vol.43/Iss.3, Mar.2005

[100] M. Sloman, E. Lupu, Policy Specification for Programmable Networks, Proceedings of First International Working Conference on Active Networks (IWAN'99), Berlin, June 1999

[101] D. C. Verma, Simplifying network administration using policy-based management, IEEE Network, Vol.16,Iss.2, Mar-Apr.2002

[102] E. Lupu, M. Sloman, Conflicts in policy-based distributed systems management, IEEE Transactions on Software Engineering, v.25, 1999

[103] J. D. Moffett, M. Sloman, Policy conflict analysis in distributed system management, Journal of Organizational Computing, v.4, 1994.

[104] M. Charalambides et al., Policy conflict analysis for quality of service management, 6th IEEE Workshop on Policies for Networks and Distributed Systems (Policy 2005)

[105] M. Charalambides et al., Dynamic policy analysis and conflict resolution for DiffServ quality of service management, IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)

[106] M. Burgess, An approach to understanding policy-based on autonomy and voluntary cooperation, 16th IFIP/IEEE Distributed Systems: Operations and Management Workshop (DSOM2005), LNCS 3775

[107] IBM Inc., Autonomic Computing Adoption Model, Accessed online Dec 2007 `http://www.ibm.com/autonomic/about_get_model.html`

[108] T. R. Andel, A. Yasinsac, On the credibility of manet simulations, IEEE Computer, Vol.39/Iss.7, July 2006

[109] M. Sloman,Policy Driven Management For Distributed Systems, Journal of Network and Systems Management, Vol 2(4), Dec. 1994

[110] J. Strassner, Policy-Based Network Management, Solutions for the Next Generation, Morgan Kaufmann, ISBN: 1558608591, 2003

[111] W. Chen, N. Jain, S. Singh, ANMP Ad hoc network management protocol, IEEE Journal on Selected Areas in Communications, Vol 17/Iss.8, Aug.1999

[112] K. S. Phanse, L. A. DaSilva, Protocol support for policy-based management of mobile ad hoc networks, IEEE/IFIP Network Operations and Management Symposium (NOMS), 2004

[113] A. Malatras, G. Pavlou, Context-driven Self-Configuration of Mobile Ad hoc Networks, IFIP International Workshop on Autonomic Communications (WAC 2005), Oct.2005

[114] P. Bellavista, A. Corradi, R. Montanari, C. Stefanelli, Context-aware middleware for resource management in the wireless Internet, IEEE Transactions on Software Engineering, Vol.29/Iss.12, Dec.2003

[115] S. S. Yau, F. Karim, Y. Wang, B. Wang, S. K. S. Gupta, Reconfigurable Context-Sensitive Middleware for Pervasive Computing, IEEE Pervasive Computing, Jul.-Sept. 2002

[116] P.-J. Wan, K. M. Alzoubi, O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks, IEEE Infocom 2002

[117] R. Friedman, M. Gradinariu, G. Simon, Locating cache proxies in MANETs, ACM MobiHoc 2004

[118] U. Kozat, L. Tassiulas, Network layer support for service discovery in mobile ad hoc networks, IEEE Infocom 2003

[119] S. Gouveris, S. Sivavakeesar, G. Pavlou, A. Malatras, Programmable Middleware for the Dynamic Deployment of Services and Protocols in Ad Hoc Networks, IEEE/IFIP Integrated Management Symposium (IM 2005), May 2005

[120] A. Westerinen et al., Terminology for Policy-Based Management, RFC 3198, Informational, Nov.2001

[121] B. Moore, E. Elleson, J. Strassner, A. Westerinen, Policy Core Information Model-Version 1 Specification, RFC 3060, Feb.2001

[122] B. Moore, Policy Core Information Model (PCIM) Extensions, RFC 3460, Jan.2003

[123] J. Strassner, B. Moore, R. Moats, E. Elleson, Policy Core Lightweight Directory Access Protocol (LDAP) Schema, RFC 3703, Feb.2004

[124] M. Pana, A. Reyes, A. Barba, D. Moron, M. Brunner, Policy Core Extension Lightweight Directory Access Protocol Schema (PCELS), RFC 4104, Jun.2005

[125] M. Wahl, T. Howes, S. Kille, Lightweight Directory Access Protocol (v3), RFC 2251, Dec. 1997

[126] C. Zhao , G. Wang, Fuzzy-control-based clustering strategy in MANET, Fifth World Congress on Intelligent Control and Automation, WCICA 2004, Vol.2, pp.1456-1460

[127] J. I. Kim, J-Y. Song, Y-C. Hwang, Location-Based Routing Algorithm Using Clustering in the MANET, Future Generation Communication and Networking, Vol.2, Dec. 2007, pp.527-531

Chapter 9

Probabilistic Fault Management

Jianguo Ding, Pascal Bouvry
Faculty of Science, Technology and Communication (FSTC)
University of Luxembourg, L-1359 Luxembourg, Luxembourg

Bernd J. Krämer
Department of Mathematics and Computer Science
FernUniversität in Hagen, D-58084 Hagen, Germany

Haibing Guan
School of Information Security Engineering
Shanghai Jiao Tong University, Shanghai 200030, P. R. China

Alei Liang
Software School
Shanghai Jiao Tong University, Shanghai 200030, P. R. China

Franco Davoli
Department of Communications, Computer, and Systems Science (DIST)
University of Genoa, Genoa 16145, Italy

9.1 Introduction

With the growth in size, heterogeneity, pervasiveness and complexity of applications and network services, the effective management of networks has become more important and more difficult. In order to meet the diverse demands and challenges confronting the networks and to allow for a scalable and manageable growth, the autonomic network paradigm has been proposed to create self-organizing, self-managing and context-aware autonomous net-

works.

Self-management is a high degree requirement for an autonomic network system to enable it to be usable and stable. Self-management is defined as using autonomic principles to provide management functionalities. An important corollary is that the management function itself is an autonomic service, and is therefore subject to the same requirements as other autonomic services. A self-management system can be summarized by four objectives, self-configuration, self-healing, self-protection and self-optimization, and four attributes, self-awareness, environment-awareness, self-monitoring and self-adjusting [50].

1) self-configuration involves automatic incorporation of new components and automatic component adjustment to new conditions to maintain desired functionalities or provide new functionality;

2) self-healing can detect, diagnose and repair hardware, software and firmware problems;

3) self-protection can automatically defend against large-scale attacks or cascading internal failures from permanent damaging valuable information and critical system functions. It may act proactively to mitigate reported problems;

4) self-optimization can continually seek ways to improve their behavior and enable the system to be more efficient.

As one of the FCAPS management functions, automatic fault management is a crucial issue in achieving the goal of self-protection and self-healing. Individual hardware defects, software errors or combinations of such defects and errors in different system components may cause the service degradation of other (remote) components in networks or even their complete failure due to functional dependencies between managed objects. Hence, an effective distributed fault detection method is needed to support quick fault detection in network management and allow for automation of fault management.

Although the Open System Interconnect (OSI) management standard provides a framework for managing faults in heterogeneous open systems, it does not address methodological issues to detect and diagnose faults. In order to fill this gap, a great deal of research efforts in the past decade have been focused on improving management systems in automatic fault detection and diagnosis. Rule-based expert systems have so far been the major approach to alarm correlation in fault detection [40] [57]. This approach suits well-defined problems where the environment is not very dynamic, but they do not adapt well to the evolving network environment [10]. Case-based reasoning [31] [45] and Coding-based methods [55] [30] offer potential solutions for fault identification and isolation, but they cannot deal with uncertain or unstable situations in networks. Finite State Machines (FSMs) are used to model fault propagation behaviors and to execute the fault identification [48] [4] [35]. However, this approach has difficulties in scaling up to large and dynamic networks. Kätker and Geihs provide Model Traversing Techniques for fault isolation in networks [21], but this lacks of flexibility, especially when fault propagation is complex

and not well structured. Lunze and Lamperti et al. investigate the fault diagnosis in discrete-event systems [28][34]. Badonnel exploits the promise theory framework to model voluntary cooperation among network nodes and make them capable of expressing the trust in their measurements during the fault detection process [2]. Most of these solutions are based on certain mechanisms and improve the automatic scheme in fault management. However, they are sensitive to "noise" (such as loss of management information, delay in information collection and response, misunderstanding alarms). That means they are unable to deal with incomplete and imprecise management information effectively in uncertain and dynamic environments.

In networks, due to losses or delays in data collection, it is difficult to obtain full and precise management information. As the complex dependency relationship between managed objects and the cause-effect relationships among faults and alarms are generally incomplete, it is impossible to get a full and exact understanding of the managed system from the viewpoint of systems management.

In daily management, specialist or expert knowledge is very important and useful. However, quantitative expert knowledge is often expressed in imprecise ways, such as "very high," "normal" or "sometimes." When expert knowledge is incorporated into a management system, probabilistic approaches are needed for the quantitative expression of this kind of expert knowledge.

Due to the complexity of networks, it is not always possible to build precise models for fault management. A well-designed strategy for fault management should therefore operate efficiently in the case of redundant, incomplete and unreliable information.

Thus, probabilistic reasoning is another effective approach for fault detection in fault management [17] [52] [49] [6] [46].

Moreover, in real-life networks, dynamic changes are unavoidable due to the enhanced network complexity and the potential degeneration or improvement in system performance. Hence to understand unavoidable changes and to catch the trend of changes in a network will be very important for automatic fault management.

Most of the current commercial management software, such as IBM Tivoli, HP OpenView, SunNet Manager, Cabletron Spectrum and Cisco Works network management software, support the integration of different management domains, collect information, perform remote monitoring, generate fault alarms and provide statistics on management information. However, they lack facilities for exact faults localization, or the automatic execution of appropriate fault recovery actions. From the experience in network management, a typical measurement for on-line fault identification indicates 95% fault location accuracy while 5% of faults cannot be located and recovered in due time [16]. Hence for large networks with thousands of managed components it may be rather time-consuming and difficult to resolve the problems in a short time by an exhaustive search for the root causes of a failure and the exhaustive detection may interrupt important services in the systems.

In large-scale uncertain and dynamic network environments, the autonomic fault management paradigm is an alternative strategy in assisting to achieve the self-management of the networks. In dealing with autonomic fault management systems, there are three major aspects to be considered: 1) to design an architecture to support autonomic behavior; 2) to represent the uncertain and dynamic information that is necessary to an autonomic object to achieve an autonomic behavior; 3) to communicate and to organize the autonomic objects among themselves in a possible large context, particularly to execute probabilistic reasoning between the depended objects.

In this chapter, application issues of probabilistic models for automatic fault management are investigated in order to resolve the fault detection challenges in uncertain and dynamic network environments.

9.2 Probabilistic Inference in Fault Management

Efficient fault management requires an appropriate level of automation and self-management. A serious problem of using deterministic models is their inability to isolate primary sources of faults from uncoordinated network events. Observing that the cause-and-effect relationship between symptoms and possible causes is inherently non deterministic, probabilistic models can be considered to gain a more accurate representation. Bayesian networks are appropriate models for probabilistic management in fault management. From the view of management, a normal operation is to trace the root causes from detected symptoms (alarms). Hence the backward inference based on the probabilistic model from effects to causes is the basis in distributed fault management.

9.2.1 The Characteristics of the Faults in Distributed Systems

Due to the enlarged topology, distributed application and services, and complex dependency between managed systems, the faults in a network are different with those in a centralized system. In general, the characteristics of faults in networks are identified as follows:

- **The sources of faults are distributed.**

  In networks, the topology and distributed services in distributed systems are often expanded in a distributed environment. The managed objects are distributed geographically or logically. The faults which are generated from a network are also scattered in the whole network. Therefore, an efficient fault management system should consider fault detection within the distributed environments.

- **Fault propagation comes from dependency relationship between managed objects.**

  In networks, to finish certain applications or services, managed objects cooperate with each other for the same goal. Often in the cooperation, one managed object depends on another object in transactions. This kind of dependency is denoted by the cause-effect relationship. For example, the disconnection in a switch may interrupt the connection services with other devices which are connected to the switch and on which they depend, Hence the dependency relationship between managed objects is an important factor in fault detection and recovery.

- **The sources of faults are often hidden.**

  In complex networks, due to the fault propagation, the symptoms of a hardware or software fault are often caused by a remote or hidden factor. That means the root of the faults is sometimes difficult to identify directly from the detected symptoms. Consider the following simple scenario: A physical link failure occurs in one of several interconnected networks. The failure is detected and reported by the management component monitoring the physical resource in question. The failure also has sided againt effects on other resources in the network, e.g., connections on the various layers which use the link will experience timeouts. The management components of resources such as, e.g., protocol stacks, will therefore report failures. The result is that the operator's console is literally flooded with reports indicating the existence of some network abnormal condition, making it extremely difficult to determine the real cause of the problem.

- **A managed system holds enough information for fault management.**

  In current networks, lots of devices and application modules keep the records of the numerous states of the operations and services. Most of the information related to faults is also recorded in the managed systems. Hence, it is possible to mine new knowledge from all the recorded information and the statistics of the historical data. For example, the system's event log consists of a large number of individual events sent by all nodes in the system that have event generation capabilities. On a typical day, the log could reach tens of thousands of events [39]. This number is a function of parameters such as the size of the system, the configuration, the type and the amount of traffic being carried, the number and the type of faults that have occurred during that day. Events are collected at a centralized operation and maintenance center (OMC) where the events log is being assembled. It is known that the manual processing of this mass of data tends to become unfeasible as the number of high speed systems in the network increases. However, the

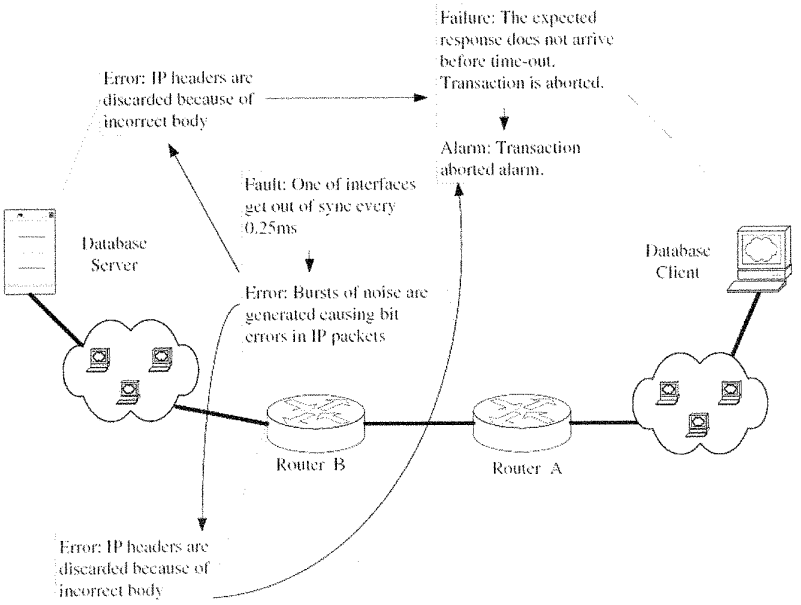


FIGURE 9.1: A model of fault propagation.

statistics in the large volume of history data can be used to retrieve new knowledge which is a potential reference for fault management.

In Figure 9.1, a simple network is presented in which a client accesses a remote database server. An interface of one of the routers between the client and server gets intermittently out of sync causing bursts of bit errors in transmitted IP datagrams. As a result, many IP datagrams passing through the router are rejected by the next router due to header errors, or by the server due to the corrupted datagram body. The client does not receive any response to its query and times out. This example illustrates how a seemingly invisible fault manifests itself through a failure at a location distant from the location of the fault. Since most faults are not directly observable, the management system has to infer their existence from information provided by the received alarms. The information carried within reported alarms may include the following: the identity of the object that generated the alarm, type of failure condition, time stamp, alarm identifier, measure of severity of the failure condition, a textual description of the failure, etc. [15] [51].

Hence, in a network, a single fault may cause a number of alarms to be delivered to the network management center. Multiple alarms may be a result of (1) fault re-occurrence, (2) multiple invocations of a service provided by a faulty component, (3) generating multiple alarms by a device for a single

fault, (4) detection of and issuing a notification about the same network fault by many devices simultaneously and (5) error propagation to other devices causing them to fail and, as a result, generate additional alarms [15]. It may be argued that typical networks provide plenty of information necessary to infer existence of faults [55].

9.2.2 Bayesian Networks for Fault Management

Bayesian networks are appropriate for automated diagnosis due to their deep representations and precise calculations. A concise and direct way to represent a system's diagnostic model is as a Bayesian network constructed from relationships between failure symptoms and underlying problems. A Bayesian network represents cause and effect between observable symptoms and the unobserved problems so that when a set of symptoms are observed the problems most likely to be the cause can be determined. In practice, the fault management system is built from descriptions of the likely effects for a chosen fault.

The development of a diagnostic Bayesian network requires a deep understanding of the cause and effect relationships in a domain, provided by domain experts. One advantage of Bayesian networks is that the knowledge can provide clear inner relationship between effects and causes. It is not represented as a black box, as in Artificial Neural Networks. In addition, compared with other logic, Bayesian networks also provide more fine-grained quantitative evaluation in probabilistic models. Thus, humanly understandable explanations of diagnoses can be given.

When a network fault management system is modelled as a Bayesian Network, two important processes need to be resolved:

1. **Ascertain the Dependency Relationship between Managed Objects.**

   A network consists of a number of managed objects. An object is a "part" of the network that has a separate and distinct existence. At the physical level, an object can be a network, a node, a switch, a layer in a protocol stack, a virtual link, a physical element like an optical fiber, a piece of cable, a hardware component, etc. At the logic level, an object can be a software service, such as a process, a piece of code, a URL, a servlet or a service request. Objects in a network consist of other objects down to the level of the smallest objects that are considered indivisible. An indivisible object is defined as a terminal object. The concept of division and appropriate level of division are system-dependent and application-dependent.

   Objects in a network are dependent upon each other in rather complex ways. These dependencies are very important for the alarm correlation and fault identification process. In most cases a failure in one object

has sided effects on other objects that depend on it. For example, a link failure has an effect on other resources in the network, e.g., connections on the various layers that use the link will experience timeouts. The knowledge of these dependencies gives us valuable information for alarm correlation and fault localization.

Dependencies: When one object requires a service performed by another object in order to execute its function, this relationship between the two object is called a dependency.

Consider any two objects, say A (such as a service, an application component in software or hardware) and B. A is said to be dependent on B, if B's services are required for A to complete its own service. One weight may also be attached to the directed edge from A to B, which may be interpreted in various ways, such as a quantitative measure for the extent to which A depends on B or how much A may be affected by the nonavailability or poor performance of B, etc. Any dependency between A and B thus arises from an invocation of B from A, which may be synchronous or asynchronous.

Dependency analysis explores causal dependencies among objects and data items, with the goal to trace the fault symptom back to the cause. This is an often used trouble-shooting technique, applicable to any system that is based on collaboration of independent or distributed entities. For instance, deadlocks in databases may be diagnosed by following transactions that are blocked waiting for other transactions.

In computing, there exist many different kinds of dependencies. However, not all references and interactions actually represent causal dependencies that are relevant for diagnosis. Hence the dependencies, which are pertinent to the purpose of the management, are taken into account.

The dependencies among distributed entities can be assigned probabilities to the links in the dependency or causality graph [25] [29]. This dependency graph can be transformed into a Bayesian Network with certain special properties [18].

In networks, the notion of dependency can be applied at various levels of granularities. Sometimes the dependencies that occur between different system components should be defined carefully. For example, the maintenance of an email server obviously affects the service 'email' and thus all the users whose user agents have a client - server relationship with this specific server; however, other services (news, WWW, FTP) are still usable because they do not depend on a functioning email service. Therefore, the inter-system dependencies are always confined to the components of the same service.

Two models are useful to get the dependency between cooperating entities in networks.

- Functional model (from the view of users)

  The functional model defines generic service dependencies and establishes the principle constraints to which the other models are bound. A functional dependency is an association between two entities, typically captured first at design time, which says that one component requires some services from another.

  The functional dependence between logical objects is determined by the implementation and functional support relationships and originates in a graph, from which it is possible to correlate a set of state changes (which may be considered as a "signature" of a problem) to the original cause of the problem.

  The functional model is utilized by a "network state estimator" to correlate the changes in the network state. The state changes are reported by the received alarms, to which information exogenous to the network (such as those related to climatic situations) is added.

- Structural model (from the view of system implementers)

  The structural model contains the detailed descriptions of software and hardware components that realize the service. A structural dependency contains detailed information and is typically captured first at deployment or installation time.

2. **Obtaining the Measurement of the Dependency.**

   The faults and anomalies in networks can be identified based on the statistical behavior of the Management Information Base (MIB) variables and the recordings in log files. When Bayesian networks are used to model networks, Bayesian networks represent causes and effects between observable symptoms and the unobserved problems, so that when a set of evidences is observed, the most likely causes can be determined by inference technologies.

   Single-cause (fault) and multi-cause (fault) are two kinds of general assumptions to consider the dependencies between managed entities in network management.

   In Bayesian networks, a nonroot node may have one or several parents (causal nodes). Single-cause means any of the causes must lead to the effect. Therefore, the dependencies between causes and effect for single-cause are denoted as:

$$P(\overline{e} \mid c_1, \ldots, c_n) = \begin{cases} 100\%, & c_i = F(False), \exists i, i \in [1, n]; \\ 0, & \text{otherwise.} \end{cases} \tag{9.1}$$

   e denotes the effect node, c_i denotes the cause of e, $i = 1, \ldots, n$.

   The existence of multiple causes means that one effect is generated only when more than one cause occurs simultaneously. Therefore, the mea-

surement of the dependencies has various possibilities based on the particular problem domain. In the above description, the states of the objects are identified as T(*True*) or F(*False*). In complex systems, it is possible that managed objects hold more than two states.

In networks, the measurement of dependencies between managed objects can be obtained from the following methods:

- Management information statistics are the main sources to get the dependencies between the managed objects in networks.
- The empirical knowledge of experts is another important source to determine the dependency between managed objects.
- For particular dependencies, an experiment provides a way to retrieve the dependencies between the managed objects.

Hasselmeyer [14] argues that the dependencies among distributed cooperating components should be maintained and published by services themselves, and he proposes a schema that allows these dependencies to be obtained.

Some researchers have performed useful work to discover dependencies from the application view in networks [13] [20] [12].

Despite all the methods cited in this section, it has to be observed that obtaining dependency information in an automatic fashion is still an open research problem. In obtaining dependency information, it needs to use available and suitable techniques to deal with every system, layer or type of device separately.

In terms of precision, the behavior of a Bayesian network reflects the quality and the detailing level of its structure, which stems from the object system model. Another factor which affects the Bayesian network model is the precision of the value of the conditional probabilities.

9.2.3 Probabilistic Inference for Distributed Fault Management

The semantics of a Bayesian network determines the conditional probability of any event given any other event. When computing such a conditional probability, the conditioning event is called the evidence, while the event for which we want to determine its conditional probability given the evidence is called the query. The general capability of a Bayesian network to compute conditional probabilities allows it to exhibit many particular patterns of reasoning (inference).

- Causal reasoning is the pattern of reasoning that reasons from a cause to its effects.

- Evidential reasoning is the reasoning from effects to its possible causes.
- Mixed reasoning combines both causal and evidential reasoning.
- Intercausal reasoning involves reasoning between two different causes that have an effect in common.

In case of fault management in networks, we only consider the backward inference (evidential reasoning), which is the basic operation of fault diagnosis.

In our previous research, a Strongest Dependency Route algorithm (SDR) for backward inference in Bayesian networks is presented in [7].

Various types of inference algorithms exist for Bayesian networks, such as exact inference [33] [42] [43] and approximate inference [38]. Each class offers different properties and works better on different classes of problems, but it is very unlikely that a single algorithm can solve all possible problem instances effectively.

In the early 1980's, Pearl published an efficient message propagation inference algorithm for polytrees [24] [41]. The algorithm is exact and works only for singly connected networks. Pearl also presented an exact inference algorithm for multiply connected networks called loop cutset conditioning algorithm [41]. A straightforward application of Pearl's algorithm to an acyclic digraph comprising one or more loops invariably leads to insuperable problems [27] [38].

Another popular exact Bayesian network inference algorithm is Lauritzen and Spiegelhalter's clique-tree propagation algorithm [33]. The clique propagation algorithm works efficiently for sparse networks, but still can be extremely slow for dense networks. Its complexity is exponential in the size of the largest clique of the transformed undirected graph.

In general, the existent exact Bayesian network inference algorithms share the property of run time exponentiality in the size of the largest clique of the triangulated moral graph, which is also called the induced width of the graph [33]. It is also difficult to record the internal nodes and the dependency routes between particular effect nodes and causes. In distributed systems management, the states of internal nodes and the key routes, which connect the effects and causes, are important for management decisions. Moreover, the sequence of localization for potential faults can be a reference for system managers and thus very useful. It is also important for system performance management to identify the relevant key factors. Few algorithms give satisfactory resolution for this case.

Compared with other algorithms, the SDR algorithm belongs into the class of exact inferences and it provides an efficient method to trace the strongest dependency routes from effects to causes and to track the dependency sequences of the causes. The computing complexity of the SDR algorithm is $O(n^2)$. It is useful in fault location, and it is beneficial for performance management. Moreover, it can treat multiple connected networks modelled as DAGs.

9.3 Prediction Strategies for Fault Management in Dynamic Networks

For complex networks, it is important that the fault management be proactive, that is, to detect, diagnose and mitigate problems before they result in severe degradation of system performance. Proactive fault management depends on monitoring networks to obtain the data on which a manager's decisions are based. Fault prediction is to predict a failure in advance based on the current information about the system. It is especially true for large systems that have some components failing all the time, for such a prediction can be done by an analysis of the historical information.

Dynamic changes in networks raise higher barriers for exact fault location. Hence, for large networks with thousands of managed components, it may be rather time-consuming and difficult to locate the unknown causes of faults in due time by the exhaustive search for the root causes of a failure and this process may interrupt or impair important system services. Dynamic updates bring even more challenges in the fault detection.

Systems, whose behavior is not fully understood, are often modeled by Bayesian networks (BNs). However, the BN paradigm does not provide direct mechanisms for modeling temporal dependencies in dynamic systems [1] [56].

We apply Dynamic Bayesian Networks (DBNs) to address temporal factors and model the dynamic changes of managed entities and the dependencies between them.

9.3.1 Dynamic Characteristics in Networks

In real-life networks, dynamic changes in networks are correlated to hardware, software and the dependencies between those components in implementing certain functions. Hence changes in networks demonstrate some particular characteristics.

1. **Hard Changes and Soft Changes in Networks**

Dynamic updates in networks can be classified into either hard or soft changes [8].

A hard change refers to a change that happens abruptly and most of the time is generated on purpose by the system owner. This kind of change does not depend on the system's history. For example, a router being added or removed from the distributed system may cause an abrupt change in the system topology and behavior. Some intended operations also generate this kind of hard change, such as a change of the configuration of a distributed system. Generally, a hard change does not happen so often, but it depends on the intention of the system manager.

A soft change, in contrast, refers to a change that happens gradually and depends on the system history. A soft change typically results from changes

of system properties such as performance degradation, application degeneration, dependency modifications and so on. A soft change may bring some potential problems, such as the network traffic gets slower or certain network services decrease in efficiency. The roots of these kinds of problems are aging devices, conflicted applications or unknown hidden factors in the systems. From the experience of system management, lots of unknown or unlocated causes of faults are triggered by a soft change, which is related to the potential changes and updates of the system. Compared with a hard change, a soft change keeps going on all the time in networks and it is hard to predict using a straightforward approach. In our research we focus on soft changes in networks.

When networks are modelled as graph structures, hard changes can be treated by structure modifications in the models based on the abrupt changes in the systems or based on the intentions of system managers. Soft changes (such as the improvement or degradation in performance of a hardware component or a software component) will not effect in the topology of the network, but will update the weights (dependencies) between the components in the network.

Considering soft changes in networks, one kind of change comes from individual networks entities; another arises from updating dependencies between managed entities. From the viewpoint of management, an entity can be a hardware device, a software component or a certain application.

In networks, real-life dynamic systems are often rife with nonlinearities, many of which are expressed as discrete failure modes that can produce discontinuous jumps in system behavior.

2. Characteristics of Dynamic Changes in Networks

In networks, dynamic changes are often identified as a discrete nonlinear time series.

A time series is a chronological sequence of observations on a particular variable. Time series data are often examined in hope of discovering a historical pattern that can be exploited in the preparation of a forecast. In order to identify this pattern, it is often convenient to think of a time series as consisting of several components: trend, cycle and irregular fluctuations.

Prediction of system faults, anomalies and performance degradation forms an important component of network management. The advent of real-time services on networks creates a need for continuous monitoring and prediction of systems performance and reliability. Although faults are rare events, they have enormous consequences when they do occur. However, the rareness of faults in distributed systems makes their study difficult. Performance problems occur more often and in some cases may be considered as the indicators of an impending fault [37]. Efficient handling of these performance issues may help eliminate the occurrence of severe faults.

9.3.2 Dynamic Bayesian Networks for Fault Management

When a dynamic network is modeled, a time dimension will be considered. Because observations and evidence can be updated over time, a management system should capture the evolution of the system as it changes over time.

DBNs provide a way to model a dynamic system, which describes a system that is dynamically changing or evolving over time [22]. DBNs will enable users to monitor and to update the system as time proceeds, and even to predict further behavior of the system. As there is no standard definition for DBNs, researchers may use different descriptions to accommodate their research requirements. Current literature tends to use the terms "dynamic" and "temporal" interchangeably.

The temporal approaches can be divided into two main categories of time representation models:

- as points or instances or
- as time intervals.

DBNs have various definitions in different application areas. In fault management and dissident systems, DBNs possess a time related function:

$$BN(t) = (V(t), L(t), P(t)) \tag{9.2}$$

For a soft change in networks, dynamic changes only happen in individual components and on the dependency between components. Under these kinds of changes, the topology of the Bayesian Network keeps stable; hence the time parameter can be omitted in nodes and edges:

$$BN(t) = (V, L, P(t)) \tag{9.3}$$

DBNs can represent large amounts of interconnected and causally linked data as well as the dynamic properties when they occur in networks. Thus DBNs can model time-related changes in the dependencies between managed objects in networks.

DBN is an extension of BN that models a time series [11]. A DBN is a way to extend Bayesian networks to model the possible distributions over a time series. We only consider the discrete-time stochastic processes, so we increase the index t by one every time a new observation arrives. The observation could indicate that something has changed in networks. Note that the term 'dynamic' means not that the topology of the network changes over time, but that a dynamic system is modeled.

9.3.3 Prediction Strategies for Network Management

Predictive management plays a crucial role in networks. The ability to predict service problems in networks, and to respond to those warnings by

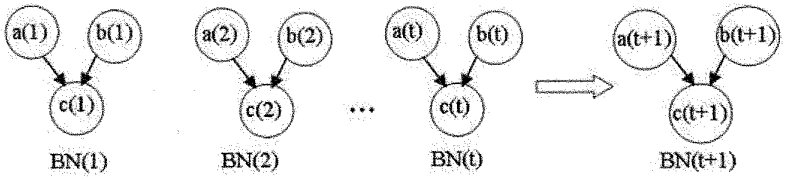


FIGURE 9.2: Model of dynamic Bayesian network.

applying corrective actions, brings multiple benefits. Firstly, the detection of system failures on a few servers can prevent those failures from spreading to the entire distributed system. For example, slow a response time on a server may gradually escalate technical difficulties on all nodes in the attempt to communicate with that server. Secondly, a prediction can be used to ensure the continuous provision of networks services through the automatic implementation of corrective actions.

Correlation serves to diminish the number of alarms presented to the operator in network management, yet ideally, the approach should be able to facilitate the fault prediction, which can predict the faults that have occurred from the alarms and warn the operator before severe faults may happen.

Considering the model of DBN in Fig. 9.2, two possible changes, which are updated over time, are presented in DBNs:

- the possible updates of the nodes (variables)
- the possible updates of the links (dependency between nodes).

When a network is modeled as a DBN, one important task is to capture the trends for the evolution in the network. This amounts to obtain $BN(t+1)$ based on the data set $BN(1), BN(2), \ldots, BN(t)$. Here $BN(t)$ denotes the updated BN at time t.

In DBN, the prediction can be denoted as

$$BN(1), BN(2), \ldots, BN(t) \Rightarrow BN(t+1) \tag{9.4}$$

In DBNs, the following prediction tasks are to be considered as a result of the management requirements.

- **Prediction per individual component.** The state of an individual component in a network can change over time due to the degradation or improvement of the component. The prediction of the individual component's change of states can be denoted as:

$$P(v(1)), P(v(2)), \ldots, P(v(t)) \rightarrow P(v(t+1)), v \in V. \tag{9.5}$$

$P(v(t))$ represents the probability of the state of component v at time t.

- **Prediction of the dependency between components.** The modification of dependencies between managed objects derives from the update of the system performance and changes in the correlation between objects. This can be denoted as:

$$P(v(1)|\pi(v(1))), P(v(2)|\pi(v(2))), \ldots, P(v(t)|\pi(v(t))) \rightarrow P(v(t+1)|\pi(v(t+1))) \tag{9.6}$$

  $v \in V$, $P(v(t)|\pi(v(t)))$ represents the probability of the dependency between node v and its parent $\pi(v)$ at time t.

- **Prediction for potential faults based on backward inference.** When the future state of the effect nodes is estimated, a promising prediction is to trace the causal nodes based on the estimated state of the effect nodes. The prediction from effects to causes is considered as the backward inference:

$$E(t+1) \rightarrow C(t+1) \tag{9.7}$$

  $E(t)$ denotes the set of effects at time t, and $C(t)$ denotes the set of causes at time t.

Dynamic Bayesian Networks (DBNs) are applied in fault management in order to address the temporal factors and to model the dynamic changes of managed entities and the dependencies between them. Our related work [DLJ+06] investigated the prediction capabilities by means of the relevant inference techniques when the imprecise and dynamic management information occurs in the distributed system. To identify the dynamic changes in networks as discrete nonlinear time series, Least Square Fit (LSF) is used for the polynomial regression. It is workable in a large scale distributed system with thousands of nodes and links. Based on the given prediction strategies, not only the prediction in an individual entity and the dependency relationship between managed entities are considered, but also the potential reasoning from effects to causes in DBN can be obtained.

To evaluate the approach of probabilistic backward inference and prediction strategy, we design a simulation scheme, which is reported in [5], to construct the simulation in Bayesian networks for probabilistic inference and prediction, so that the simulation in Bayesian networks is close to real life networks and, further, the intelligent decision in management of networks can be obtained.

9.4 Application Investigations for Probabilistic Fault Management

The probabilistic fault management strategies do not intend to replace traditional network management, but to complement the deficiency of traditional management systems in uncertain situations.

9.4.1 Architecture for Network Management

Network management takes place between two major types of systems: those in control, called managing systems, and those observed and controlled, called managed systems. The most common managing system is called a network management system (NMS). Managed systems include hosts, servers and network devices such as routers or intelligent repeaters. Here, we use "online network devices" to represent the term "managed system."

9.4.1.1 Components of Network Management System

As specified in Internet RFCs [47] and other documents, a typical distributed management system comprises:

- **Manager**: A manager generates commands and receives notifications from agents. There usually are only a few managers in a system.
- **Agents**: Agents collect and store management information such as the number of error packets received by a network element. An agent has local knowledge of management information and transforms that information into the form compatible with SNMP. An agent responds to commands from the manager, and sends notification to the manager. There are potentially many agents in a system.
- **Managed object**: A managed object is a vision of a feature of a network, from the point of view of the management system [19]. For example, a list of current active TCP circuits in a particular host computer is a managed object. Managed objects differ from variables, which are particular object instances. Managed objects can be scalar (defining a single object instance) or tabular (defining multiple and related instances). In literature, "managed object" is sometimes used interchangeably with "managed element."
- **Management Information Base (MIB)**: A MIB is a formal description of a set of network objects that can be managed by using the Simple Network Management Protocol (SNMP). The format of the MIB is defined as part of the SNMP.

- **Management protocol**: A management protocol is used to convey management information between agents and network management stations (NMSs). Simple Network Management Protocol (SNMP) is the Internet community's de facto standard management protocol.

Interactions between NMSs and managed devices can be any of four different types of commands: *read*, *write*, *traverse* and *trap*.

9.4.1.2 Protocols for Network Management

1. **Simple Network Management Protocol (SNMP)**

SNMP (Simple Network Management Protocol) is a communication protocol that has gained widespread acceptance since 1993 as a method of managing TCP/IP networks, including individual network devices, and devices in aggregate. SNMP was developed by the IETF (Internet Engineering Task Force), and is applicable to any TCP/IP network, as well as other types of networks. It defines management entities, typically the NMS, and agent entities, typically the network devices, or more accurately, the processes that run on the NMS and network devices. The information available through SNMP is organized into a Management Information Base (MIB). The structure of this information is defined in the Structure of Management Information (SMI). One or more MIB files define the MIB supported by a given SNMP agent. The bottom line is that SNMP provides management information in a structured manner that is well suited to retrieval and modification via applications.

2. **Common Management Information Protocol (CMIP)**

Common Management Information Protocol (CMIP) is an OSI-based network management protocol that supports information exchange between network management applications and management agents.

CMIP is a well-designed protocol that defines how network management information is exchanged between network management applications and management agents. It adopts an ISO reliable connection-oriented transport mechanism and has built up security that supports access control, authorization and security logs. The management information is exchanged between the network management application and management agents through managed objects.

CMIP is widely used in the telecommunication domain and telecommunication devices that typically support CMIP.

By far the largest advantage of SNMP over CMIP is its simple design, so it is easy to use on a small network as well as on a large one, with ease of setup, and lack of stress on system resources. Also the simple design makes it simple for the user to program system variables that they would like to monitor. Another major advantage of SNMP is its wide use today around the world. Because of its development during a time when no other protocol of this type existed, it became very popular, and has built in protocol supported by most major vendors of networking hardware, such as hubs, bridges and

routers, as well as major operating systems. SNMP is by no means a perfect network manager, but it can fix these flaws due to its simple design.

3. **Other Related Protocols for Network Management**

Besides the standard protocols SNMP and CMIP, there are still some other miscellaneous protocols which can help in network management.

- **Ping:** Ping is commonly used to check connectivity between devices. It is the most common protocol used for availability polling. It can also be used for troubleshooting more complex problems in the network. Ping uses the Internet Control Message Protocol (ICMP) Echo and Echo Reply packets to determine whether one IP device can talk to another. Most implementations of ping allow users to vary the size of the packet.

- **Traceroute:** Traceroute is most commonly used to troubleshoot connectivity issues. If we know that we cannot reach a host from another host, traceroute will show whether the connectivity loss exists at one of the intermediate routers. Traceroute determines that the destination has been reached when it receives an ICMP destination port unreachable message. Note that we are actually discovering the path that the ICMP timeout messages are taking when they come back.

- **Terminal Emulators:** Terminal emulators are used for many purposes in network management, including users' accesses to network devices. Obviously, access is useful for configuring and troubleshooting devices. There are also times when information or operations on network devices is not available through SNMP and scripts must be written to access this information or capability through terminal access. Telnet is the traditional way of obtaining terminal emulation access to network devices.

- **Syslog:** The syslog protocol was first defined as part of the UNIX operating system to log messages within the OS. Syslogs allow a computer or device to deliver messages to another computer. Syslog messages have a particular format that associates a facility and a severity or priority with a message. The facility code allows syslog to group messages from different sources and take actions based on this facility or group.

9.4.1.3 Extended Architecture for Probabilistic Fault Management

In an extended architecture, a module of Fault Diagnosis Agent (FDA) is added to execute the management tasks for probabilistic fault diagnosis. A detailed model is denoted in Figure 9.3.

The FDA is designed to operate in parallel with a SNMP manager or a SNMP agent. It can communicate with a SNMP manager and get the needed information (such as the states of the interfaces or ports of devices) from the SNMP manager, in emergent situations or special requirements, and it can

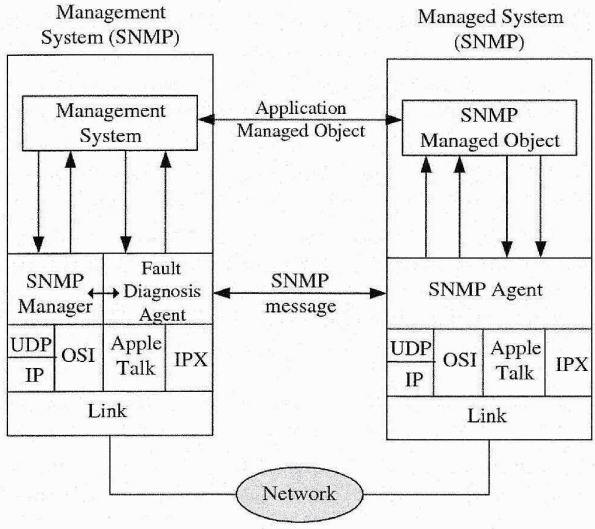


FIGURE 9.3: Detailed model of network management with FDA.

also communicate with agents which reside on the managed devices to retrieve information by SNMP.

In ordinary operation, FDA gets information or fault events from the SNMP manager, so that it saves the traffic of network communication. FDA has interfaces with the logfile analyzer and system managers so that additional events can be integrated into the fault management systems.

9.4.2 The Structure and Function of Fault Diagnosis Agent

Figure 9.4 depicts the inner structure of FDA and the process for fault management. Some important components and their functions in FDA are described in the next subsections.

9.4.2.1 Data Collection and Analysis for Fault Management

The following components are related to data collection and analysis:

1. **Event Configurator**

   Event configurator can process event configuration based on certain management tasks or managers' purpose. Thus we need to determine when and which events to be triggered by taking the following steps:

   - Select the objects or variables.
   - Select the devices and interfaces.

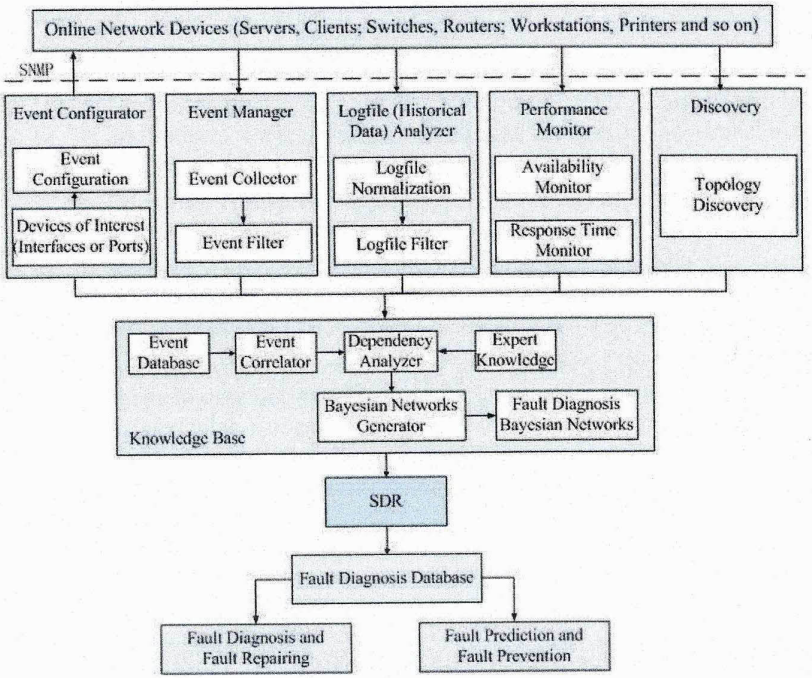


FIGURE 9.4: The structure of fault diagnosis agent.

- Determine the trigger values for each object or object and interface type.
- Determine the severity for the event.

Event configuration can:

- Determine what information is of interesting and what states or levels are normal for the network.
- Evaluate the events. The devices will automatically generate and make sure that the desired events are turned on and the undesired ones are turned off or filtered out.

Configuring Events

There are two ways to configure events:

(a) The easiest and the least costly way is to have the network device check the trigger points and generate the event. The data we want to query already exist in the device on which we are configuring the event.

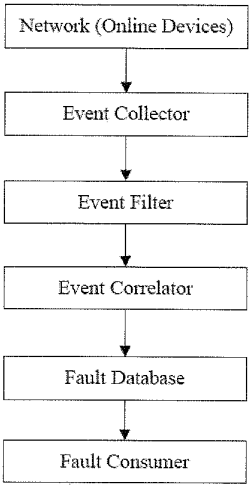


FIGURE 9.5: The procedure of event management.

(b) The other way is to collect data at a management station and analyze that data against thresholds there. Thresholds on devices, also known as agent-based thresholds, allow the network devices to directly generate events when something interesting happens in the network.

2. **Event Manager**

The event manager component can execute event collecting, event filtering and event correlating. Figure 9.5 denotes the procedure of event management in the fault management system. Some components (online devices, event collector, fault consumer) in Figure 9.4 are standard or defined by the network device vendors and they can be configured to meet the management requirements. The other components are own developments (such as event filter, event correlator, fault database) which are organized for certain functions in dealing with the event management. Most of the components related to event management are actually software components which are embedded in the management system.

Event Producers

Any device on the network can produce an event for a variety of reasons. In addition, events can be produced by the network management systems or call-tracking systems. Network engineers (expert) can identify issues and directly enter them as events.

Some examples of components that produce events and the types of events they might generate are as follows:

- Devices that provide network connectivity such as hubs, switches, and routers can generate events such as the loss or re-establishment of connectivity between two devices.
- Computers providing services can generate events such as a database or print server becoming available or unavailable.
- Network management systems can generate events such as an event that is triggered if, after calculating the number of collisions versus the network load, a sample falls outside the expected curve.
- An event could be produced by an end user of the network calling the help desk and reporting a problem such as the inability to reach a particular printer. The help desk technician may generate an incident report that would trigger an event describing the incident reported.

Continuous and Discrete Data Sources of Events

There are two kinds of data sources for events: continuous data source and discrete data sources.

(a) **Continuous data sources** present a continuously changing curve of values. Examples of continuous data sources include:

- The number of packets received on a network interface,
- The CPU utilization on a device,
- The number of calls completed on an ISDN interface.

These data sources are sampled at some rate to produce a data stream. The type of trigger applicable to continuous data sources is a continuous threshold. This threshold type generates an event when the most recent value in a continuous data stream becomes interesting.

(b) **Discrete data sources** present data that can have several discrete states. Discrete event is a main character in computer systems. In networks, most of the events comes from discrete data sources. Examples of discrete data sources include:

- The state of a network interface (*up, down, testing*),
- The operational state of a device (*operational, faulty, reloading*),
- The environmental status (*normal, warning, critical, shutdown, notPresent*).

Discrete data can be represented by data types, including enumerated data, Boolean data or text data fields with a fixed range of text values or states.

Types of Events

Events can be divided into two broad categories: state change events and performance events. State change events are triggered when something in the network changes state. Performance events are generated when a possible performance issue is noticed by some component of the network. Most performance events are generated by thresholds that are defined by the network manager on the network devices or NMS stations.

Event Monitoring

Event monitoring can be divided into two types: time-driven monitoring and event-driven monitoring. The former involves periodically obtaining snapshots of the network state, e.g., by polling for the values of MIB variables. The later involves asynchronous receipts of notifications about "interesting" events.

Consider trap messages generated by agents in response to state changes in managed objects. Traps may be the result of completely unpredictable events, such as the loss of an underground line to a backbone, a hardware failure in a router or a software error in a user's application.

Event Collection, Normalization and Filtering

The goal of event management is to collect all event information and determine what, if any, actions need to be taken as a result of each event. There are several steps that must be taken on receipt of an event:

(a) Events must be collected. Usually, events will come in from a variety of sources through several different methods or protocols. There are two methods for collecting event data: active polling and event reporting.

   i. **Active polling** involves a management station actively obtaining specific management data from network devices. The collected data are then stored in a database and used later for reporting.

   ii. **Event reporting** (polling by exception) denotes that the managed device or agent generates a trap or event which is received and logged by the manager. For event reporting, events are generated when preconfigured thresholds are exceeded, or a change of state or an unusual event such as a fault occurs.

(b) Upon receipt, the events should be normalized to facilitate their processing. Normalization means to format the events in a consistent way, regardless of the delivery mechanism.

(c) Next, this event must be determined if it can be filtered or deleted. Because the volume of events to be processed can be very high, it is important to eliminate undesired events as early as possible. Filtering means to eliminate undesirable events by comparing them to

some pattern and eliminating those events that match the pattern. T. Koch provided efficient approaches in automated event filtering and event correlation [23].

(d) Next, the management system should correlate events and determine the faults that exist in the network. Correlation in this context means to examine events to determine the preceding cause of an event or to determine the root cause of a fault. Event correlation will provide help in investigating the dependency between events or between managed components based on certain service.

3. **Logfile Analyzer**

Logfiles are used by many event producers to record the events they generate. Sometimes, the only way to determine the occurrence of these events is to parse the logfiles and to extract the event information. Some logfiles only have event information. Others have a variety of data mixed with event information.

In networks, logfiles keep a record of historical events of a system. Efficient analysis and mining of logfiles will help to retrieve the events and to evaluate the system performance.

Some examples of log files that might have event information include the following:

- syslog logfiles
- system console logfiles
- application message files

Many systems log messages to files. Windows NT has an event log that records system events. Many UNIX systems use the syslog protocol to log system events. Network devices such as routers and switches can record system events on a syslog server. Applications may also log messages to files. Some of these messages are the result of applications finding that a threshold has been crossed, such as a database management system noticing that a database partition is within 90 percent of capacity.

Logfiles are often recorded to be less-structured or half-structured. Before the information can be processed further, the logfile should be normalized and filtered in terms of the management requirements.

For example, consider a syslog message that indicates that an interface went down:

> Sep 19 16:51:07 10.29.2.1 79: Sep 19 16:50:24: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state to down

The example syslog message would be normalized as follows:

Source = 10.29.2.1
Time = Sep 19 16:50:24 (may want to convert to UTC)
Priority = 5
Type = linkDown
Variables = Interface Ethernet0

Then the event can be appended into a structured relational database which might contain the elements such as Event ID, Source (IP addresses), Time Stamp, Priority, Event Type, Variables (States of Interfaces or Ports), Destination (IP address) and so on. Event analysis can be executed by the operation on the database.

4. **Performance Monitor**

Availability is the measure of time for which a network or application is available for a user. From a network perspective, availability represents the reliability of the individual component in a network. The availability can be an important parameter in measuring the dependency between network events or between network devices.

Measuring availability requires coordinating real-life measures with the statistic collected from the managed devices.

Measuring Availability

According to Stallings [53], availability is expressed by the Mean Time Between Failures (MTBF) with the following formula:

$$Availability = \frac{MTBF}{MTBF + MTTR} \tag{9.8}$$

where MTTR is Mean Time To Repair.

ICMP pings are the easiest to use and report on measuring availability. The following equation shows the relevant formula:

$$Avail = \frac{(Total\#\ of\ PINGs\ received)}{(Total\#\ of\ PINGs\ sent)} \tag{9.9}$$

Availability Statistics

Availability can be defined as the probability that a product or service will operate when needed. In networks, this can be defined as the average fraction of connection time that the product or service is expected to be in operating condition. For a network that can have partial as well as total system outages, availability is typically expressed as network availability:

$$Availability = 1 - \frac{total\ connection\ time_{outage}}{total\ connection\ time_{in-service}} \tag{9.10}$$

The availability of objects or services can be an important source to evaluate the reliability of objects.

9.4.2.2 Dependency Analysis for Events

Dependency analysis for events includes the integration work of the topology analysis and event correlation analysis and eventually to ascertain the precedence events for each event.

1. **Topology Discovery**

   Physical network topology refers to the characterization of the physical connectivity relationships that exist among entities in a network. Discovering the physical layout and interconnections of network elements is a prerequisite to many network management tasks, including reactive and proactive resource management, server siting, event correlation and root-cause analysis [3].

   SNMP-based algorithms for automatically discovering network layer (i.e., layer-3) topology are featured in many common network management tools, such as HP's OpenView, IBM's Tivoli, Actualit's Optimal Surveyor and the Dartmouth Intermapper. Recognizing the importance of layer-2 topology, a number of vendors have recently developed proprietary tools and protocols for discovering physical network connectivity. Examples of such systems include Cisco's Discovery Protocol, Bay Networks' Optivity Enterprise and Loran Technologies' Kinnetics network manager.

   The topology of a network can help in investigating the dependency relationship between managed objects.

2. **Event database**: is an integrated database to record all received events and knowledge related to fault management. The events come from a variety of sources such as SNMP notifications, syslog messages, entries in log files, NMS events or event expert knowledge. Refined expert knowledge can be manually appended into the database. Historical data are another possible source to enlarge the database. Some historical data (such as logfiles, historical records of fault management) are available in a network, but most of them are not structured, large-scale and intermingled. Hence data mining technologies are useful tools to help the data classification, refining and integration [44] [54]. Temporal events and period data collection and statistics can be organized as temporal databases, so that the prediction operation can be implemented in the dynamic environments of networks.

The event database should be a formatted and structured database. A typical example structure of event database is depicted as Table 9.1. The data in the event database can be used for event-correlation analysis, dependency calculation and statistics analysis.

Table 9.1: The Structure of Event Database.

Event ID	$0x003831f49700000055$	...
Event Type	*LinkDown*	...
Priority	*Critical*	...
Time Stamp	1125302050	...
Source IP	10.29.2.1	...
Variables	$InterfaceEthernet0$	...
Precedence Event	$0x003831f49700000000$	...

Event ID is the only identification of an event. It is generated by the event collection system automatically.

Event Type is to identify the type of the event, such as *Link Down*, *system error* or *notification of link overload*, etc.

Priority identifies the level of the events. In event database, ordinal number is used to denote the priority of an event, such as *Information* (4), *Major Warning* (3), *Serious* (2) and *Critical* (1).

Time Stamp records the time when the event is generated. In Event Database, UTC (Coordinated Universal Time) is used to replace GMT (Greenwich Mean Time). For example, GMT: *7:54:10, Aug. 29*, is equal to UTC: *1125302050.*

Source IP denotes the location (IP address) where the event is generated.

Variables depict the characteristics of a concrete event. MIB defines some standard variables for distributed systems management. The values of variables or their collaboration analysis may help one to determine the real value of a fault event. For example, in order to detect a *down* in an interface, there are three primary methods, all using RFC 2233 variables:

(a) Watch for *linkDown* traps from a device.

(b) Watch for interface status change messages in syslog.

(c) Poll the *ifOperStatus* and *ifAdminStatus*. If *ifAdminStatus* is up and *ifOperStatus* is down, the interface is intended to be up.

Precedence Events identify the preceding events (cause events) of a current event. The value of precedent events is the result of event correlation and can be used further for dependency analysis between events. An event may have one or more precedence events, or else the event tends to be a root event if the precedence event is *null*.

3. **Event Correlation and Dependency Analyzer**

   Event correlation is an automated process that enables administrators to find, among many events, those revealing critical problems that cannot be ascribed to other issues (root cause analysis). Of particular importance is the detection of a few problems that have an adverse effect on the stability and performance of critical services (e.g., an e-business application server for an online retail shop).

   Event correlation is the "smart" part of a management application. It relies on a number of techniques [32]: state transition graphs (finite state machines), rule-based reasoning, binary coding (code-books), case-based reasoning, probabilistic dependency graphs (Bayesian networks), model-based reasoning, neural networks, etc. In today's management platforms, several techniques are often used in conjunction.

   A number of researches have already investigated the distribution of event correlation. Some proposals focus on correlating network events with the network topology [4] or intrusion detection [26]. Others propose general-purpose languages [36].

   The basic premise underlying dependency models is to model a system as a DAG in which nodes represent system events or system components and weighted directed edges represent dependencies between them. A dependency edge is drawn between two nodes only if a failure or problem with the node at the head of the edge can affect the node at the tail of the edge; if the dependency edge presents, the weight of the edge represents the impact of the failure's effects on the tail node. Heavier edges represent more significant dependencies and therefore more likely main causes.

9.4.2.3 Bayesian Networks for Fault Management

The following components depict the practice details how Bayesian networks are applied to probabilistic fault management.

1. **Bayesian Networks Generator (BNG)** is designed to generate a BN model from the event database by event correlation and dependency analysis. In order to generate a BN for network management, the following steps should be taken:

   (a) Identify the **event family**. Here an event family is defined as the set of one event and all its parents events. Based on the event

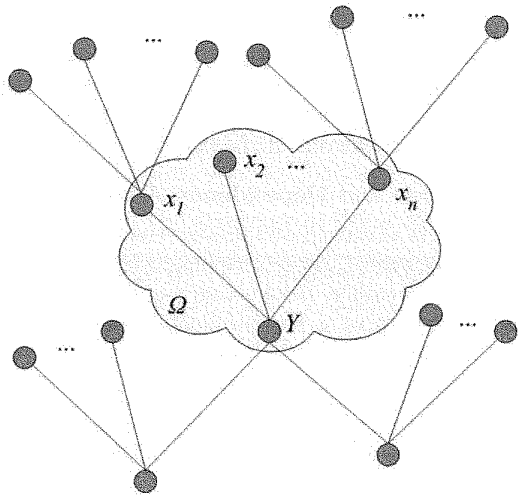


FIGURE 9.6: Dependency analysis in events.

database and correlation techniques, the direct precedence of an event will be identified and recorded. Then an event family will be set up. Figure 9.6 depicts the event family (sub-BN) Ω and the dependency between nodes.

(b) In BNs, the dependency is denoted by a JPD, that means the effect node (son) is effected by all its joint causal nodes (parents), even if some are strong or weak. See Figure 9.6, for an event family Ω; suppose node Y has n parents $(x_1, x_2, \ldots, x_n)$. Table 9.2 depicts how the JPD is obtained between every pair of parent nodes and their son node.

Table 9.2: The JPD obtaining between every pair of parent nodes (X) and their son node (Y).

i	$\overline{Y}$	$x_1 x_2 \ldots x_n$	τ_i	$P(\overline{Y} \mid X_i)$
1	1	0 0 ... 0	τ_1	P_1
2	1	0 0 ... 1	τ_2	P_2
⋮	⋮	⋮ ⋮ ⋮ ⋮	⋮	⋮
2^n	1	1 1 ... 1	τ_{2^n}	P_{2^n}

The JPD should have 2^n data items: $P(\overline{Y} \mid X_i), i \in [1, 2, \ldots, 2^n]$, X_i denotes the status of parent nodes of Y at $No.i$. Here 1 denotes the state is in order, while 0 denotes the state is out of order. For example,

$$X_1 = x_1 x_2 \ldots x_{n-1} x_n = 0\ 0\ \ldots\ 0\ 0$$
$$X_2 = x_1 x_2 \ldots x_{n-1} \overline{x_n} = 0\ 0\ \ldots\ 0\ 1$$
$$\vdots$$
$$X_{2^n} = \overline{x_1 x_2} \ldots \overline{x_{n-1} x_n} = 1\ 1\ \ldots\ 1\ 1$$

For each accepted event, it is possible to get the value of precedence events by polling or trap record. All the events in the event family at the time will become a record for the JPD.

In Table 9.2, $\overline{Y} = 1$ denotes Y is out of order. τ_i depicts the number of the statistics of the events where $\overline{Y} = 1$ and X_i occur. τ_i can be also denoted as the statistics of time duration for fault events. Then

$$P_i = P(\overline{Y} \mid X_i) = \frac{\tau_i}{\sum_{j=1}^{2^n} \tau_j} \tag{9.11}$$

Thus we can determine the JPD of dependencies between parent nodes and their son. All these data are the prerequisites for further SDR inference.

(c) Each event family can be bridged by the correlation and the dependency between conjunction events, and as a result a global BN for a distributed system can be created. Both the structure of the BN and the values of dependencies between nodes are recorded as fault diagnosis Bayesian Networks.

The BNG can generate dynamic Bayesian networks based on the temporal data in event database for a given period. Then a temporal JPD will be obtained.

2. **Strongest Dependency Route (SDR) Calculator** is a key component to execute the calculation for pruning and backward inference in BNs. The results after the SDR are:

   (a) a spanning tree with strongest dependency routes, which can be acquired by the depth-first search in spanning tree, between effect nodes and cause nodes;

   (b) a strongest dependency sequence of candidate causes for specified effects. Normally, a SDR module is started by the manager when the management system cannot locate the root cause of faults, and it can act as an assistant in evaluating the system performance.

The SDR module can work on a dynamic Bayesian network model, and it can result in a temporal spanning tree and temporal strongest dependence sequences.

3. **Fault diagnosis database (FDD)** is a refined database to record the results (both the spanning tree and the strongest dependency sequence of causes) of the SDR operation. A FDD is actually a knowledge base in guiding the probabilistic fault diagnosis in networks.

   In a static model, FDD can be the basis to execute fault diagnosis and fault repairing. In a dynamic model, FDD can provide fault prediction and help for fault prevention.

   Normally a repairing action will be executed on the basis of the knowledge of FDD in an even-driven style. The action module can be controlled by the system manager. Moreover, a human manager can do some repair work to assist system maintenance.

   Another important function for an action module is to send feedback to the FDD and fault diagnosis BN, and further to update FDD and diagnosis BN. The results of both success and failure results will be accepted by the FDA itself, so that it is possible to improve the accuracy and performance for future inference and prediction in fault management.

All in all, the FDA will involve the fault management processes in networks, such as collecting alarms (events), filtering correlated alarms (events), dependency analysis, diagnosing faults, verifying and eliminating faults, taking action to ensure customer satisfaction and developing and implementing corrective plans.

9.4.3 Discussion of Application Issues

For probabilistic fault management of networks, there are still some related topics that should be investigated carefully.

- Efficiency and accuracy are important factors in evaluating a management system. For probabilistic fault management, the scale of the data set, the accuracy of the BN models, the filtering and refining data and iterative correction are main stages to improve the efficiency of a management system. BN models should be defined carefully in appropriate levels to resolve the core problems. The granularity definition of managed objects should be consistent with the management tasks. Hence a managed object may denote a subset, a server or a function module in different levels based on different management goals and tasks.

- The complexity of computation and operation in fault management is related to the complexity of the model, data collection, analysis and the complexity of the algorithm. Building a Bayesian network requires a careful tradeoff between the desire for a large and rich model on the one hand and the costs of construction, maintenance and inference on the other hand. Actually, building a Bayesian network is a creative and

iterative process. With the advance of iterative procedures and associated graphical tools for supporting the overall construction process, the quantification task will be addressed within its proper context, which hopefully will contribute to reducing its burden.

- The growth of applications and network services extends a network and suggests network management to migrate to service-oriented application management. Application services are generally composed of a set of distributed components, each of which contributes a specific function to the total services provided by the application. Thus, a fault or malfunction occurring in any of the software components can lead to a problem in the end-to-end service that the customer perceives. In addition, application components function in close cooperation with the elements that comprise the environments like the communication network, the operating system, various middle-ware such as databases, message functions and their services. Thus, determining the source of a problem involves the hunt for the root cause which may lie in any of the components and services that contribute to the end-to-end service to the customer. In service-oriented networks, probabilistic fault management is still an important research topic in maintaining service efficiency. The strategies of fault management which is discussed in this chapter can be also used in application services of the network. Although the sources and the process of data collection are different, the core components and functions of the Fault Diagnosis Agent (FDA) presented in this chapter are still workable in the scenario of application services. Some related research is necessary to discover dependencies from the application view in distributed services [13] [12].

- Security is another important topic in network management. Security not only focuses on protecting the systems itself, but also relates to the protection of management information. In our approaches to fault management, the FDA is designed on the basis of traditional management frameworks. Hence it will not bring new risks to the security of management information and communication data. The key computing and operation of the components are executed locally in network management stations. Management information is more sensitive than other data in networks; thus most management systems are running in reliable and private environments.

9.5 Conclusions

Some unavoidable uncertain factors and dynamic changes in networks raise higher barriers for fault management. Bayesian network is an appropriate tool in modeling the probabilistic environment and is efficient in executing the backward inference, which is an important task in tracing the root causes when faults are detected. In modeling dynamic networks, the time factor should be taken into account, since even the standard Bayesian Networks paradigm does not provide direct mechanisms for modeling the temporal dependencies in dynamic systems. For this reason, Dynamic Bayesian Networks (DBNs) are applied in network management in order to address the temporal factors and to model the dynamic changes of managed entities and the dependencies between them. Furthermore, the prediction capabilities are investigated by means of the relevant inference techniques when the imprecise and dynamic management information occurs in the network.

Application issues of probabilistic fault management are investigated to demonstrate how the probabilistic fault management can be brought into practice. The software architecture of the fault diagnosis agent (FDA) is designed to implement the main tasks of probabilistic fault management, such as data collection, data filter and refinement, Bayesian network generation, SDR algorithm application, inference of causes and fault prediction operations.

Acknowledgement

This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Program. This research was also partly supported by the Chinese national program of high-tech research and development 863 project: 2006AA01Z169 and Chinese national 973 project: 2007CB316506. The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] C. F. Aliferis and G. F. Cooper. A structurally and temporally extended bayesian Belief network model: Definitions, properties, and modeling techniques. Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, pages 28-39, 1996. Morgan Kaufmann.

[2] Remi Badonnel and Mark Burgess: Fault Detection in Autonomic Networks Using the Concept of Promised Cooperation. DSOM 2007, pages 62-73, 2007.

[3] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri and A. Silberschatz. Topology Discovery in Heterogeneous IP Networks. Proceedings of IEEE INFOCOM, pp. 265-274, 2000.

[4] C. S. Chao, D. L. Yang and A. C. Liu. An automated fault diagnosis system using hierarchical reasoning and alarm correlation. Journal of Network and Systems Management, 9(2):183-202, 2001.

[5] Jianguo Ding, Ningkang Jiang, Xiaoyong Li, Bernd Krämer, Franco Davoli and Yingcai Bai. Construction of Simulation for Probabilistic Inference in Uncertain and Dynamic Networks Based on Bayesian Networks. Proceedings of the 6th International Conference on ITS Telecommunications, pages 983-986, IEEE Communication Society Press, June 2006.

[6] Jianguo Ding, Bernd J. Krämer, Yingcai Bai and Hansheng Chen. Probabilistic Inference for Network Management. M.M. Freire, P. Chemouil, P. Lorenz (Eds.) Universal Multiservice Networks: Third European Conference, ECUMN 2004, Lecture Notes in Computer Science, Volume 3262, pages 498-507, Springer-Verlag Heidelberg, 2004.

[7] Jianguo Ding, Bernd Krämer, Yingcai Bai and Hansheng Chen. Backward Inference in Bayesian Networks for Distributed Systems Management. Journal of Network and Systems Management, 13(4), pages 409-427, Dec 2005.

[8] Jianguo Ding, Bernd Krämer, Shihao Xu, Hansheng Chen and Yingcai Bai. Predictive Fault Management in the Dynamic Environment of IP Networks. Proceedings of 2004 IEEE International Workshop on IP Operations & Management, pages 233-239, Oct. 2004.

[9] Jianguo Ding, Xiaoyong Li, Ningkang Jiang, Bernd J. Krämer and Franco Davoli. Prediction Strategies for Proactive Management in Dynamic Distributed Systems. Proceedings of the International Conference on Digital Telecommunications, IEEE Computer Science Society Press, p.74(6 pages), August 2006.

[10] A.S. Franceschi, L.F. Kormann and C.B. Westphall. Performance Evaluation for Proactive Network Management. Proceedings of the ICC'96 International Conference on Communications, vol. I, pages 22-26, 1996.

[11] N. Friedman. Learning the Structure of Dynamic Probabilistic Networks. Proceedings of the 14th Annual Conference on Uncertainty in AI, pages 139-147, 1998.

[12] J. Gao, G. Kar and P. Kermani. Approaches to Building Self Healing Systems using Dependency Analysis. Proceedings IEEE/IFIP Network Operations and Management Symposium (NOMS'04), Vol. 1, pages 119-132, 2004.

[13] M. Gupta, A. Neogi, M. K. Agarwal and G. Kar. Discovering Dynamic Dependencies in Enterprise Environments for Problem Determination. Proceedings of 14th IEEE/IFIP International Workshop on Distributed Systems Operations and Management (DSOM 2003), LNCS 2867, pages 221-233, 2003.

[14] Peer Hasselmeyer. Managing Dynamic Service Dependencies. 12th International Workshop on Distributed Systems: Operations & Management (DSOM 2001), Nancy, France, ISBN 2-7261-1190-4, pages 141-150, 2001.

[15] K. Houck, S. Calo and A. Finkel. Towards a practical alarm correlation system. A.S. Sethi, F. Faure-Vincent and Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London pages 226-237, 1995.

[16] C. Hill. High-availability systems boost network uptime: Part 1. http://www.eetasia.com/ARTICLES/2001JUL/2001JUL01_NTEK_ST_QA_TA.PDF. Motorola Telecom Business Unit, 2001.

[17] C. Hood and C. Ji. Proactive Network Fault Detection. IEEE Transactions on Reliability, Vol. 46, No.3, pages 333-341, Sept. 1997.

[18] D. Heckerman and M. P. Wellman. Bayesian networks. Communications of the ACM, 38(3):27-30, Mar. 1995.

[19] ITU-T. Recommendation X. 700: Management Framework for Open Systems Interconnection (OSI) for CCITT applications, September 1992.

[20] A. Keller, U. Blumenthal and G. Kar. Classification and Computation of Dependencies for Distributed Management. Proceedings of 5th IEEE Symposium on Computers and Communications. Antibes-Juan-les-Pins, France, July 2000.

[21] S. Kätker and K. Geihs. A Generic Model for Fault Isolation in Integrated Management System. Journal of Network and Systems Management, Special Issue on Fault Management in Communication Networks, Vol. 5, No. 2, June 1997.

[22] T. Koch, B. Kramer and G. Rohde. On a Rule Based Management Architecture. Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environment, pages 68-75, June 1995.

[23] Thomas Koch. Automated management of distributed systems. Ph. D thesis, FernUniversität in Hagen. Shaker Verlag, ISBN: 3-8265-2594-9, 1997.

[24] Jin H. Kim and Judea Pearl. A computational model for combined causal and diagnostic reasoning in inference systems. Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83), pages 190-193, 1983. Morgan Kaufmann.

[25] I. Katzela and M. Schwarz. Schemes for Fault Identification in Communication Networks. IEEE/ACM Transactions on Networking, vol. 3, pages 753-764, 1995.

[26] C. Krugel, T. Toth and C. Kerer. Decentralized event correlation for intrusion detection. Proceedings of Information Security and Cryptology (ICISC), pages 114-131, December, 2001.

[27] F. L. Koch and C. B. Westphall. Decentralized Network Management Using Distributed Artificial Intelligence. Journal of Network and Systems Management, Vol. 9, No. 4, December 2001.

[28] J. Lunze. Diagnosis of quantized systems based on a timed discrete-event model. Systems, Man and Cybernetics Part A, IEEE Transactions 30 (3)(2000) 322-335.

[29] S. Kliger, S. Yemini, Y. Yemini, D. Oshie and S. Stolfo. A Coding Approach to Event Correlation. Proceedings of the Fourth IEEE/IFIP International Symposium on Integrated Network Management, pages 266-277, Chapman and Hall, London, UK, May 1995.

[30] C. Lo, S. H. Chen and B. Lin. Coding-based schemes for fault identification in communication networks. Journal of Network and Systems Management, 10(3), pages 157-164, 2000.

[31] L. Lewis. A case-based reasoning approach to the resolution of faults in communication networks. Integrated Network Management, III, pages 671-682. Elsevier Science Publishers B.V., Amsterdam, 1993.

[32] Lundy Lewis. Managing Business and Service Networks. Kluwer Press, ISBN 0306465590, 2001.

[33] S. L. Lauritzen and D. J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. Journal of the Royal Statistical Society, Series B 50:157-224, 1988.

[34] Gianfranco Lamperti and Marina Zanella. Diagnosis of discrete-event systems from uncertain temporal observations. Artif. Intell. 137(1-2): 91-163, 2002.

[35] R. E. Miller and K. A. Arisha. FaultManagement Using Passive Testing for Mobile IPv6 Networks. Proceedings of 2001 IEEE Global Telecommunications Conference. Vol. 3, pages 1923-1927, 2001.

[36] M. Mansouri-Samani. Monitoring of Distributed Systems, Ph.D. thesis, Imperial College, UK, 1995.

[37] R. Maxion and F. Feather. A Case Study of Ethernet Anomalies in a Distributed Computing Environment. IEEE Transactions on Reliability. Vol. 39, No. 4, pages 433-443, October, 1990.

[38] R. M. Neal. Probabilistic Inference Using Markov Chain Monte Carlo Methods, Tech. Rep. CRG-TR93-1, University of Toronto, Department of Computer Science, 1993.

[39] Y. A. Nygate. Event Correlation Using Rule and Object Based Techniques. Proceedings of IFIP/IEEE International Symposium on Integrated Network Management, pages 278-289, May 1995.

[40] A. Osmani and F. Krief. Model-Based Diagnosis for Fault Management in ATM Networks. Proceedings of International Conference on ATM ICATM 99, pages 91-99, 1999.

[41] J. Pearl. A constraint-propagation approach to probabilistic reasoning, Uncertainty in Artificial Intelligence. North-Holland, Amsterdam, pages 357-369, 1986.

[42] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, CA, 1988.

[43] J. Pearl. Causality: Models, Reasoning, and Inference. Cambridge, England: Cambridge University Press. New York, NY, 2000.

[44] Wei Peng, Tao Li and Sheng Ma. Mining log files for data-driven system management. Natural language processing and text mining. ACM SIGKDD Explorations Newsletter archive, Volume 7, Issue 1, June 2005.

[45] G. Pemido, J. Nogueira, and C. Machado. An Automatic Fault Diagnosis and Correction System for Telecommunications Management. Proceedings of 6th IFIP/IEEE International Symposium on Integrated Network Management, pages 777-791, 1999.

[46] Shunshan Piao, Jeongmin Park and Eunseok Lee. Problem Localization for Automated System Management in Ubiquitous Computing. LNCS 4809, pages 158-168, 2007.

[47] http://www.rfc-editor.org

[48] I. Rouvellou and G. W. Hart. Automatic Alarm Correlation for Fault Identification. Proceeding of IEEE INFOCOM95, pages 553-561, 1995.

[49] R. Sterritt and D. W. Bustard. Fusing hard and soft computing for fault management in telecommunications systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C, Vol. 32, No. 2, pages 92-98, 2002.

[50] R. Sterritt and D. W. Bustard. Autonomic Computing - a Means of Achieving Dependability? Proc. IEEE Int. Conf. Engineering of Computer Based Systems, pages 247-251, 2003.

[51] P.H. Schow. The Alarm Information Base: A Repository for Enterprise Management. Proceedings of Second IEEE International Workshop on Systems Management, pages 142-147, 1996.

[52] M. Steinder and A. S. Sethi. Non-deterministic Diagnosis of End-to-end Service Failures in a Multi-layer Communication System. Proceedings of ICCCN, Scottsdale, pages 374-379, 2001.

[53] William Stallings. SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, 3rd Edition, Addison-Wesley, ISBN: 0-201-48534-6, 1998.

[54] J. Stearley. Towards informatic analysis of syslogs. Proceedings of 2004 IEEE International Conference on Cluster Computing, pages 309-318, 2004.

[55] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini and D. Ohsie. High speed and robust event correlation. IEEE Communications, 34(5):82-90, May 1996.

[56] J. D. Young and J. E. Santos. Introduction to Temporal Bayesian Networks. Online Proceedings of the 1996 Midwest Artificial Intelligence and Cognitive Science Conference. http://www.cs.indiana.edu /event/maics96/Proceedings/Young/maics-96.ps.

[57] J. Zupan and D. Medhi. An Alarm Management Approach in the Management of Multi-Layered Networks. 3rd IEEE International Workshop on IP Operations & Management, pages 77-84. 2003.

Computer Science/Computer Engineering/Computing

Bringing together an extensively researched area with an emerging research issue, **Context-Aware Computing and Self-Managing Systems** presents the core contributions of context-aware computing in the development of self-managing systems, including devices, applications, middleware, and networks. The expert contributors reveal the usefulness of context-aware computing in developing autonomous systems that have practical application in the real world.

The first chapter of the book identifies features that are common to both context-aware computing and autonomous computing. It offers a basic definition of context-awareness, covers fundamental aspects of self-managing systems, and provides several examples of context information and self-managing systems. Subsequent chapters on context-awareness demonstrate how a context can be employed to make systems smart, how a context can be captured and represented, and how dynamic binding of context sources can be possible. The chapters on self-management illustrate the need for "implicit knowledge" to develop fault-tolerant and self-protective systems. They also present a higher-level vision of future large-scale networks.

Through various examples, this book shows how context-aware computing can be used in many self-managing systems. It enables researchers of context-aware computing to identify potential applications in the area of autonomous computing. The text also supports researchers of autonomous computing in defining, modeling, and capturing dynamic aspects of self-managing systems.

Features

- Illustrates the scope and usefulness of context-aware computing
- Identifies context-aware aspects in autonomous systems
- Describes the essentials of self-managing systems, including the capability to perceive one's computing environment
- Investigates the various roles context-aware computing plays in developing self-managing devices, middleware, applications, and networks, such as self-protecting and self-healing networks

CRC Press
Taylor & Francis Group
an informa business
www.crcpress.com

ISBN 978-0-367-38584-2
9 780367 385842