Semantic Technologies for User-Centric Home Network Management

Annie Ibrahim Rana, MSc

Department of Computing and Mathematics

Waterford Institute of Technology

Thesis submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy

Supervisors : Dr. Brendan Jennings, Dr. Steven Davy, Dr. Mícheál Ó Foghlú

July 2015

Declaration

I declare that this thesis entitled *"Semantic Technologies for User-Centric Home Network Management"*, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely the result of my own research and has not been taken from the work of others save to the extent that such work has been cited and acknowledged within the text of my work.

Signature	:	Annie Rana
Name	:	Annie Ibrahim Rana
ID	:	20032310
Date	:	July 2015

This thesis is dedicated to my beloved family and teachers, especially my father (late) 'Dr. Muhammad Ibrahim', my mother 'Shehnaz Ibrahim', my husband 'Muhammad Khalid Iqbal Tahir', and my teacher (late) 'Prof. Dr. Ghulam Murtaza' for their endless support, priceless love and undaunted encouragement all the way through out my life.

Abstract

Home area network (HAN) management is problematic for ordinary home users. Lack of user expertise, potential complexity of administration tasks, extreme diversity of network devices, price pressures producing devices with minimal feature sets, and highly dynamic requirements of user applications are some of the main challenges in *HANs*. As networking becomes enabled in many more *HAN* devices, these problems are set to increase. A viable solution to address these challenges lies in various levels of automation in Home Area Network (*HAN*), and at a slightly deeper level of self-governance in general, now often termed as autonomic computing. *HANs* are good candidate for autonomic network management, such as policy-based network management (*PBNM*), to automate network managing tasks. However, a significant challenge here is the transformation of user requirements to a form that is understandable to the *HAN* system. Semantic computing enables a system interpreting semantics of instances at different levels of abstraction (e.g. concepts related to users and network) without requiring it to know the interlinks among different system concepts (e.g. how a user is linked to its networked devices and applications). The research work presented in this thesis proposes a framework for the implementation of user-driven, semantic-aware, policy-based *HAN* management. Our goal is to transform user preferences into network configurations so that we can give control to *HAN* users to make their networks behave as per their requirements.

List of Publications

Annie Ibrahim Rana, Steven Davy, and Brendan Jennings. Semantic aware processing of user defined inference rules to manage home networks. *Journal of Network and Computer Applications (JNCA), Elsevier*, (Submitted).

Annie Ibrahim Rana, Steven Davy, and Brendan Jennings. User-driven certainty factor support model to resolve semantic conflicts for exclusive disjunctive uncertain inference rules in smart homes. *Journal of Systems and Software (JSS), Elsevier*, (Submitted).

Annie Ibrahim Rana and Brendan Jennings. Semantic uplift of monitoring data to select policies to manage home area networks. In *IEEE 26th International Conference on Advanced Information Networking and Applications (AINA)*, IEEE AINA'12, pages 368 –375, Japan, March 2012.

Annie Ibrahim Rana, Brendan Jennings, Mícheál Ó Foghlú, and Sven van der Meer. Autonomic Policy-based HAN Traffic Classification using Augmented Meta Model for Policy Translation. In *8th International Conference on Wireless and Optical Communications Networks (WOCN)*, IEEE WOCN'11, pages 1–8, France, May 2011.

Annie Ibrahim Rana and Mícheál Ó Foghlú. Policy-based network management in home area networks: interim test results. In *Proceedings of the 3rd international conference on New technologies, mobility and security*, NTMS'09, pages 334–336, Egypt, 2009b. IEEE Press.

Annie Ibrahim Rana and Mícheál Ó Foghlú. Policy-based traffic management in home area networks: an elementary testbed model. In *Proceedings of the 7th International*

Conference on Frontiers of Information Technology, FIT'09, ACM, pages 6:1–6:6, Pakistan, 2009.

Annie Ibrahim Rana and Mícheál Ó Foghlú. New role of policy-based management in home area networks: concepts, constraints and challenges. In *Proceedings of the 3rd international conference on New technologies, mobility and security*, IEEE NTMS'09, pages 353–358, Egypt, 2009a. IEEE Press.

Annie Ibrahim Rana and Mícheál Ó Foghlú. Policy refinement for traffic management in home area networks: Problem statement. In *9th Ninth Information Technology and Telecommunication Conference (ITT 2009)*, ITT'09, Ireland, Oct 2009. DIT.

Chamil Kulatunga, Ruban Kandavanam, Annie Ibrahim Rana, Sasi Balasubramaniam, and Dimirti Botvich. Hysac: A hybrid delivery system with adaptive content management for IPTV networks. In *IEEE International Conference on Communications (ICC)*, ICC'11, pages 1–5, USA, June 2011.

Acknowledgements

I owe thank to many people who supported me with their guidance, friendship and love through out different stages of my PhD project. First of all I express my gratitude for the continuous guidance rendered to me by my research supervisor, Dr. Brendan Jennings, whose encouragement, supervision and support from the preliminary to the concluding level enabled me to develop an understanding of the subject. In spite of his difficult official engagements and extreme busy routine, he always made himself available to listen to my problems and provided me with his best guidance and enlightened suggestions. He taught me how to frame my research questions, implementing the research ideas and synthesizing the write up with scientific writing skills. It would not have been possible to complete this project without his invaluable help and guidance. His mentorship was paramount in providing a well rounded experience consistent my long-term career goals. I also want to express my gratitude to Dr. Mícheál Ó Foghlú for his valuable support at the preliminary stage of my PhD project. He provided me utmost moral support at many difficult times of my stay in Ireland. He encouraged me to not only grow as an experimentalist but also as an independent thinker. I am not sure many graduate students are given the opportunity to develop their own individuality and self-sufficiency by being allowed to work with such independence. I am heartily thankful to my other supervisor, Dr. Steven Davy, for his constructive guidance and generous support on my work during the years. His shrewd insights, valued comments and suggestions, have been of such great benefit for improvement of my work, and in writing the manuscripts. I thank Dr. Huaiguo Fu for providing me the opportunity to work at TSSG and pursue my dream to do research.

I would like to thank gratefully and sincerely Dr. Sven van der Meer and Zohra Boudjamil, for their guidance, understanding, and most importantly, their friendship since the day I came here in Ireland, whom I looked up during times of stress and their support gave me a new energy to face the challenges of my studies and project. I would like to thank all the senior researchers of TSSG, especially Dr. Willie Donnelly, Dr. Dmitri Botvich and Eamonn de Leastar, who also helped me time to time with their guidance.

Several people have been instrumental in completion of this thesis report. The emotional support and persuasion of all these people kept me going even when at times it seemed very hard to continue with the demands of the project. My best friend, my father, Dr. Muhammed Ibrahim, who is no longer in this world to see me achieving my target, was the great support I had during all the times. He went against the local traditions and customs, and sent me abroad for higher education. My special thanks are due to my loving husband, Muhammad Khalid Iqbal Tahir, whose sagacious suggestions and motivational speeches inspired and helped me a lot at different tough stages during the project. He was available for me when I needed him in spite of his official commitments. I must express my gratitude towards my friends especially Shaista Jabeen, who is always a great inspiration. I could not imagine completion of my PhD work without a peaceful environment required for this deliberation. I want to express my heartfelt thanks to my beloved family, especially my mother, and my friends for their unwavering love and support for my success. They used all possible ways to provide me with peace and comfort I required to focus on my work. Sometimes unsaid words do wonders.

Above all, I am extremely thankful to Allah almighty for blessing me with loving and caring families, teachers, friends and colleagues as well as providing me with the mediums, courage and health to meet the challenges of the process of PhD. Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

Annie Ibrahim Rana

Contents

List of Figures

List of Tables

Glossary

BFS	Breath First Search
CFSM	Certainty Factor Support model
DEN	Directory Enabled Network
DEN-ng	Directory Enabled Network Next Generation
DFS	Depth First Search
DL	Description Logic
DSL	Digital Subscriber Line
DSSS	Direct Sequence Spread Spectrum
DTD	Document Type Definition
EKI	Enhanced Keyword Interpretation
EUCC	Extended Usage and Change Control
FDE	Filtered Data Element
FHSS	Frequency Hopping Spread Spectrum
FOAM	Framework for Ontology Alignment and Mapping
FTP	File Transfer Protocol
HAN	Home Area Network
HTTP	Hyper Text Transfer Protocol
IDS	Iterative Deepening Depth First Search
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
KI	Keyword Interpretation
LAN	Local Area Network

MAC	Media Access Control
MDA	Model Driven Architecture
MDD	Model Driven Development
NBA	Network Behaviour Analysis
NIET	No Inference Engine Theory
OMG	Object Management Group
OWL	Web Ontology Language
PAN	Personal Area Network
PBNM	Policy-based Network Management
PDP	Policy Definition Point
PEP	Policy Enforcement Point
PM	Plain Old Telephone Service
PM	Probability Model
PSP	Policy Specification Point
QoS	Quality of Service
QoSM	Quality of Service Management
RF	Radio Frequency
RSVP	Resource Reservation Protocol
RuleML	Rule Mark-up Language
SID	Standard Information Model
SIP	Session Initiation Protocol
SWAP	Standard Wireless Access protocol
SWRL	Semantic Web Rule Language
TCP	Transmission Control Protocol
TOS	Type of Service
UCC	Usage and Change Control
UDT	User Datagram Protocol
UIS	Uncertain Inference System
UMLS	Unified Medical Language System
VoIP	Voice over IP
WAN	Wireless Area Network
XML	eXtensible Mark-up Language

Chapter 1

Introduction

Mark Weiser's article on ubiquitous computing [Weiser, 1999], published nearly two decades ago, predicted the "disappearance" of computers as more devices became networked within our homes and other environments. Much progress has been made towards this vision, and many more devices are now networked than previously. In the last decade, convergence of networks enabled devices and complex network services have changed the traditional view of home area networks (*HANs*). Today, a typical *HAN* is a complex network [Laurnén, 2007] of in-home digital devices, such as laptops, desktops, mobile phones, cameras, printers, projectors, gaming consoles, entertainment technology, energy and power control systems, home security systems and smart appliances. Recent research developments in this area [Chetana Sarode, 2012, Gaul and Ziefle, 2009, Sheahen and Skubic, 2015, Wilson et al., 2014] aim to realise the vision of smart home networks with modern devices and advanced interfaces. However, the key challenge today is less about the design of exciting new such devices with non-traditional interfaces that are embedded everywhere; instead, most of us face the more mundane challenge of how to integrate and control the existing devices at our homes that are network enabled but technically challenging from a user control perspective [Chetty et al., 2010, Edwards et al., 2010, Ho et al., 2010, Poole et al., 2009b, Shehan and Edwards, 2007, Wilson et al., 2014]. Essentially, this is a *HAN management* problem. Lack of networking expertise, potential complexity of administration tasks, heterogeneity of network devices and highly dynamic requirements of user applications are some of the main *HAN man-*

agement challenges[1] in a typical *HAN*. As networking becomes enabled in many more *HAN* devices, these problems are set to increase [Edwards et al., 2011, Grinter et al., 2009, Shehan and Edwards, 2007].

HAN users need to be able to configure their networking equipment and applications to ensure that services operate as expected, while also ensuring an adequate access control for both regular users of the network (e.g., family members) and others (e.g., guests). Services such as video conferencing and remote home monitoring have significantly different network forwarding requirements than those of website browsing, yet the home network user is expected to understand how to best configure their network resources to ensure such services operate as required. Furthermore, the users of the *HAN* devices may change frequently, so the users must be able to configure and set appropriate security controls to protect access to home services and devices [Poole et al., 2009a]. In general, it is very difficult to offer a *HAN management* interface that is tailored for the use by non-experts, whilst being highly flexible with respect to a large diversity of network types, end user devices and multimedia applications. Specifically, the challenges relate to understanding the activity of the networking devices, understanding what applications home network users are using, identifying which user is using a device at a given time (often many devices in the home are freely shared between family members) and taking the appropriate actions based on user preferences. These challenges must be addressed whilst coping with a vast diversity of network equipment and emerging use cases for service usage. There are some tools available to monitor and control networks, e.g., [Delaet et al., 2010, Shiravi et al., 2012, Soin, 2012]; however, the expertise level required to use these puts them beyond the access of an ordinary *HAN* user. The process of manual configuration itself and the growing complexity of infrastructure involved in *HAN* have threatened to undermine the very benefits that modern day technologies aim to provide and that is ease of use. Complexity leads to difficulty in management and it potentially leads to unreliability of a system. In short, the diversity of operational scenarios, time and cost constraints, and technical complexity are making *HAN* management more difficult than ever.

[1] `http://newsroom.cisco.com/dlls/2009/ts_051209.htm`

1.1 Motivation

A viable solution to address above mentioned challenges lies in various levels of automation in Home Area Networks (*HANs*), and at a slightly deeper level of self-governance in general, now often termed as autonomic computing [Kephart and Chess, 2003]. One influential articulation of this challenge is IBM's manifesto on autonomic computing that asserts: *"In the evolution of humans and human society, automation has always been the foundation for progress"* [Kephart and Chess, 2003]. Automating systems and processes can itself produce additional complexity in *HANs*. Nevertheless, automation is one potential approach to lessen the growing complexity of existing network enabled *HAN* devices that are interconnected together as, at least in some senses, one big computing system. This requires development of a management system to control the complexity of *HAN* management that will serve as interface between *HAN* users and underlying networked devices. IBM's focus on autonomic computing has been very influential and has migrated to the networking world, where the terms autonomic communications and autonomic network management have been coined to address the issues relating to the self-governance, self-management, self-configuration, self-healing etc. (sometimes abbreviated to self-*), approaches to manage the complex network infrastructures [Davy et al., 2006a, Jennings et al., 2007].

HANs are good candidate for autonomic network management techniques, such as policy-based network management (*PBNM*), a widely accepted and used approach (to achieve some levels of autonomy) in enterprise networks [Verma et al., 2001]. *PBNM* provides the means by which the administration and management processes in a network can be simplified and largely automated [Verma, 2002]. It provides a flexible and robust mechanism to manage network resources and services, e.g., bandwidth allocation, service priority and access control, by using set of rules also known as *policies* to govern the network. In enterprise networks, highly skilled domain experts usually define the network management policies. However, in typical a *HAN*, there is always an acute shortage of expert users who understand their networked devices and related management tasks. Even in the presence of network experts, *HAN management* is tedious and time consuming process. So how we can implement the precepts of *PBNM* in *HAN* without requiring its users to understand their networks systems' complexities?

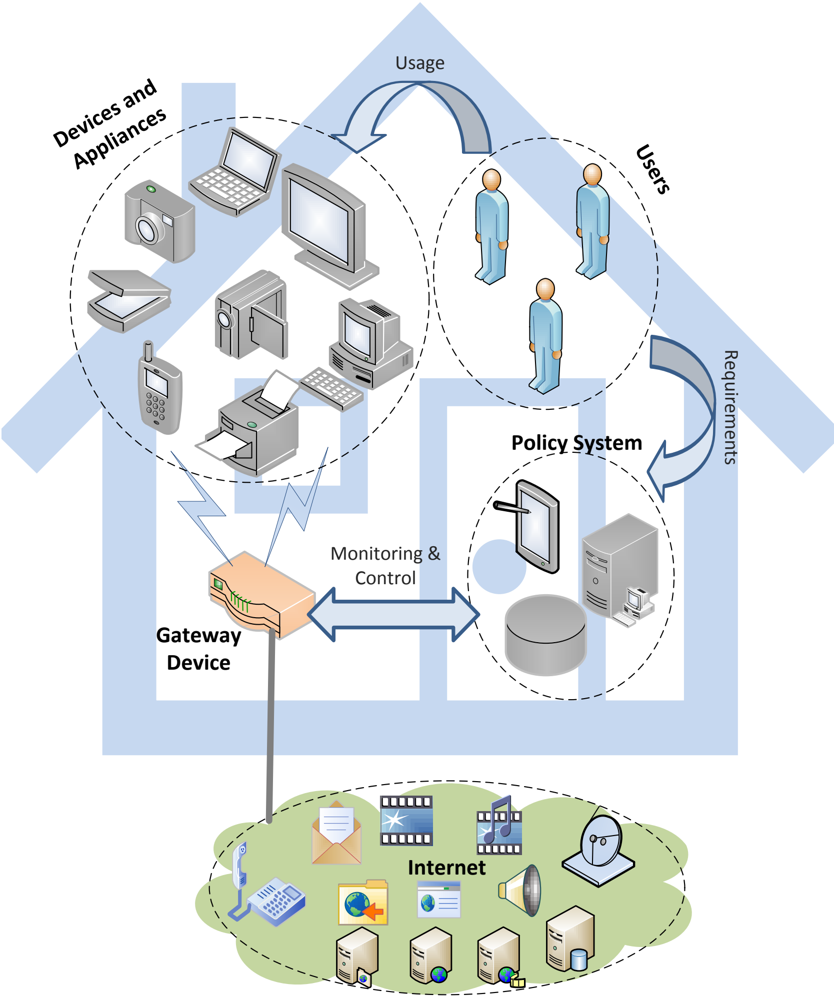


Figure 1.1: User-Centric, Policy-based *HAN* Management: a semi-automated approach to manage and control home devices, applications and systems through network policies that are translated to network configurations.

This question motivated us to explore the idea of using policy rules of different forms to capture user requirements, communicating the rules to *HAN* systems, and then enforcing them in the form of configurations of different devices or applications in a *HAN* system. We aim to establish a *HAN* control loop [Kielthy et al., 2010], where the network events are monitored and policies are triggered based on the events that are significant to *HAN* users. A significant challenge here is to process policy rules across different sub-domains (user domain, device domain, application domain, network domain etc.) of a *HAN* system. Using the concept of various management levels of the network presented in [Davy et al., 2008, Löpez De Vergara et al., 2005], we can establish connections among as many sub-domains of *HAN* through semantic models [Löpez De Vergara et al., 2005] representing the different elements and entities existing in a *HAN* system. A semantic model includes the capability to express information that enables management system to interpret semantics from the instances at different abstraction level of a *HAN* system (e.g., concepts related to users and network devices), without the need to know the meta-model (e.g., how a user correlates to a device or an application within a network system). We wish to explore the role of semantics in the management of *HAN* using an architecture similar to policy continuum [Davy et al., 2008, Löpez De Vergara et al., 2005]. In *HANs*, the networked devices are usually unmanaged and network services work in best effort fashion. Our goal is to transform user preferences into network configurations so that we can give control to *HAN* users to make their networks behave as per their requirements. Figure 1.1 shows a policy system employed in a home network where a *HAN* management system (that is deployed on a *HAN* gateway device) connects *HAN* devices, applications and users with the policy system. The idea is to deploy user policy rules on a *HAN* gateway device with the help of policy system and the gateway device controls the devices, applications and systems attached on *HAN*.

1.2 Hypothesis

The hypothesis of the research work that is presented in this thesis, is formulated from the literature review covering the state of the art in autonomic network management processes in *HANs*, in particular looking at policy-based network management, semantic models and semantic-aware inferencing. It is the primary objective of this thesis to prove the validity of the following hypothesis.

"By leveraging semantic models (that have been enriched with application or domain specific information), processes for policy specification, selection, processing and translation of declarative policies to executable policies can be created, which exploit semantic information and relationships among the entities that have been employed to represent the declarative and executable policies, and their corresponding systems."

This research hypothesis has two parts; the first part focuses on using formal semantic models to assist the policy specification, selection, processing and translation. It is assumed that semantic models can provide an elaboration to a declarative policy using semantic information of entities employed in to represent the policy. The second part focuses on enhanced elaboration of primitive semantic information by using well-defined reasoning approaches to discover relationships to other entities within the semantic models that are used represent the corresponding systems. The baseline assumption is that the elaborated semantic information can help in processing and translating policies.

1.3 Research Questions

The following are the research questions defining the scope of hypothesis:

Q1: *"What semantic information is required to specify and select a declarative policy, and translate it to an executable policy within the scope of HAN?"*

Q2: *"How can semantic information in different sub-domains of HAN be interlinked?"*

Q3: *"How can semantic information provided in the semantic models be used to process declarative policies to transform them into HAN configurations?"*

Q4: *"How semantic information provided in the semantic models can be used to reason over to provide new knowledge to manage HAN?"*

Question 1 defines the scope of our investigation in the area of semantic aware rule composition and its transformation, emphasises identifying the elements of policy rule structure and meta-information that helps translation of a declarative policy rule to an executable form. Discussion around this question can be found in chapters 3 and 6. Question 2 defines the scope of our quest for semantic presentation and arrangement of information in different sub-domains of *HAN*; it is based on an assumption that declarative and executable policies are part of different sub-domains within the *HAN* system. Discussion around this question can be found in chapters 3, 4 and 5. Question 3 defines the scope of our inquiry of semantic information utilization in the areas of policy specification, policy selection for execution, and policy translation. We later extended the scope of this question and also explored the role of semantics in policy conflict resolution. Discussion around this question can be found in chapters 4, 5, 6 and 7. Question 4 defines the scope of our investigation in the area of reasoning over semantic information to produce new knowledge and utilization of knowledge to manage *HAN* devices. Discussion around this question can be found in chapters 4, 5, 6 and 7.

1.4 Main Contributions

This thesis presents processes and algorithms to support semantic aware processing and translation of abstract and declarative policies to network configurations with the help of semantic models. This includes a framework for semantic aware processing of abstract policies, techniques to retrieve semantics from cross layered *HAN* system to enrich the semantics of abstract policies, and a resolution strategy for semantically conflicted rules. The framework describes the components for policy specification, selection, translation and enforcement. The core of the proposed policy processing technique is to utilise the semantic information embodied within the network flows for the selection of abstract policies and further to their transformation into low-level concrete executable policies/configurations with the help of semantic models.

The thesis contributes to the areas of semantic computing for human-centric policy-based home area network management; in particularly, it addresses the areas of semantic uplift of network flows, semantic-driven policy processing, semantic-aware policy translation and semantic-driven conflict resolution. The main contributions are:

1. A human-centric home area network management framework that adapts to dynamic functional changes in a home network;
2. Semantic uplifting and enrichment techniques for monitoring data in the home area networks that extract relevant information from network flows and update semantic models appropriately;
3. A semantic-driven policy processing using enrichment technique that uses semantic information to select and process abstract, user-defined policies;

4. A semantic-aware policy translation technique that demonstrates the role of semantics in translating abstract/declarative user-defined policies to concrete/executable policies/configuration in the *HAN* management system;

5. A semantic-driven conflict resolution of independent exclusive disjunctive rules using a belief support model and user feedback loop to deal with unresolvable conflicted uncertain policy rules.

1.4.1 Thesis Outline

The rest of this dissertation is organized as follows:

Chapter 2 provides an overview of research on *HAN* management techniques, policy-based *HAN* management and use of semantics and inference in the management of home networks. We then present the shortcomings of the existing network management techniques and the challenges that are required to be addressed.

Chapter 3 presents proposed framework for the semantic driven policy-based *HAN* management and framework components. The framework serves following design objectives:

1. Specification of user defined rules and new knowledge in the framework;
2. Linked data management of rules and knowledge in different sub-domains of *HAN*, constructing the *HAN* domain model;
3. Monitoring of network flows and interpretation of network events using the *HAN* domain model;
4. Enforcement of user network requirements.

Chapter 4 discusses the technique and algorithms that are developed to implement semantic uplift of low-level monitoring data as a part of *HAN* management framework. It describes the algorithms for semantic-driven processing of declarative policies that are selected for processing because of the related events captured from the low-level monitoring data. The policy processing involves semantic uplifting for low-level monitoring data and policy selection processes. This chapter also discusses the implementation of the proposed technique using different use cases in *HAN*.

Chapter 5 discusses the technique and algorithms that are developed to implement semantic enrichment of high-level monitoring data as a part of *HAN* management framework. It describes the algorithms for semantic-driven processing of declarative policies that are selected for processing because of the related events captured from the high-level monitoring data. The policy processing involves semantic enrichment for high-level monitoring data, policy selection and policy processing processes. This chapter also discusses the implementation of the proposed technique using different use cases in home area network.

Chapter 6 describes the technique and algorithms that are developed to implement policy translation for user-defined abstract rule using the technique of semantic refinement. The presented technique navigates the *HAN* domain model that is represented in an ontological form. The main contribution of this chapter is the extension of a semantic translation algorithm to map policy concepts in policy languages to meta-model concepts.

Chapter 7 describes the technique and algorithms that are developed to implement conflict resolution of exclusive disjunctive uncertain inference rules and extend a classical conflict resolution technique, Certainty Factor Model, with an intelligent semantic-driven approach to resolve the conflicts. In this chapter, we outlined the theoretical foundation of our approach and described the reasoning capabilities and algorithms for the proposed technique. We demonstrated the perceived effectiveness of our approach through presentation of experimental results in comparison to probabilistic approaches based on real time test scenarios using a test-bed.

Finally, Chapter 8 summarises our results and outlines directions of future research.

Chapter 2

Background and State of the Art

This chapter entails an overview of technologies, trends and challenges in home area networks that motivated our research. The main concepts related to Home Area Network (*HAN*) and network management are briefly discussed. Policy-based management for the use of *HAN* management is explored and other technologies such as ontology-based network management and inference are reviewed to investigate their potential use within the scope of our research problem. State of the art relating to user-centric policy-based home area network management, semantic driven policy processing (specification, selection and translation) and user driven inference are then reviewed. This provided a context for our research on cutting edge processes developed to employ policies to manage home area networks, semantics to interconnect different conceptual models within *HAN* and use of inference to design a user driven intelligent *HAN* management system. Lastly, conclusions are drawn from the literature review and the requirements of the research methodology are outlined.

This chapter is structured as follows: §2.1 gives background knowledge about home area network *HAN*, network management, policy-based network management, semantic technologies, and rule-based reasoning and inference. §2.2 presents cutting edge research in the areas of home network management, semantic aware network management and semantic driven conflict resolution. Lastly, §2.3 presents the analysis, future work related to our research problem and concludes the chapter.

2.1 Background

This section gives a brief background and introduction to home area network, traditional network management, policy-based network management, semantic computing and inference systems, highlighting the standard terminology and methods used in past and present and the mundane challenges in related areas. The main network management concepts are described and reviewed from the perspective of their use in addressing the issues of *HAN* management as discussed in Chapter 1.

2.1.1 Home Area Network

Home area networks emerged in late 1990s fuelled by the rapid growth of the Internet [Coffman and Odlyzko, 2001]. In the literature, there are several different opinions on what exactly constitutes a *HAN* also known as residential or consumer or subscriber network. Over the years, a typical *HAN* has grown from simpler a network with few devices [Laurnén, 2007] to a complex networking system [Kailas et al., 2012] that now connects in-home digital devices, such as laptops, desktops, mobile phones, cameras, printers, projectors, gaming consoles, entertainment technology, energy and power control systems, home security systems and smart appliances, into a common network. Today *HAN* is evolving into more complex networked environment led by the agile advancement of data sharing, communication devices, multimedia applications and Internet technologies. Existing home networks allow home devices to communicate with each other to share resources and often a common connection to the Internet using different networking technologies and models. A few of these technologies and network models are discussed in the following sections.

2.1.1.1 HAN Networking Models

Within homes, network enabled devices, appliances and systems are connected through the combination of wired and wireless technologies [Wang et al., 2011]. The wired technology based *HAN* systems are usually old fashioned bulky, largely point-to-point loop or star-based systems. For the most part, those with any robust capacity are tethered systems limiting the mobility and flexibility of *HAN* users and devices. For wireless communication models, there are several approaches available within *HAN* [Wang et al.,

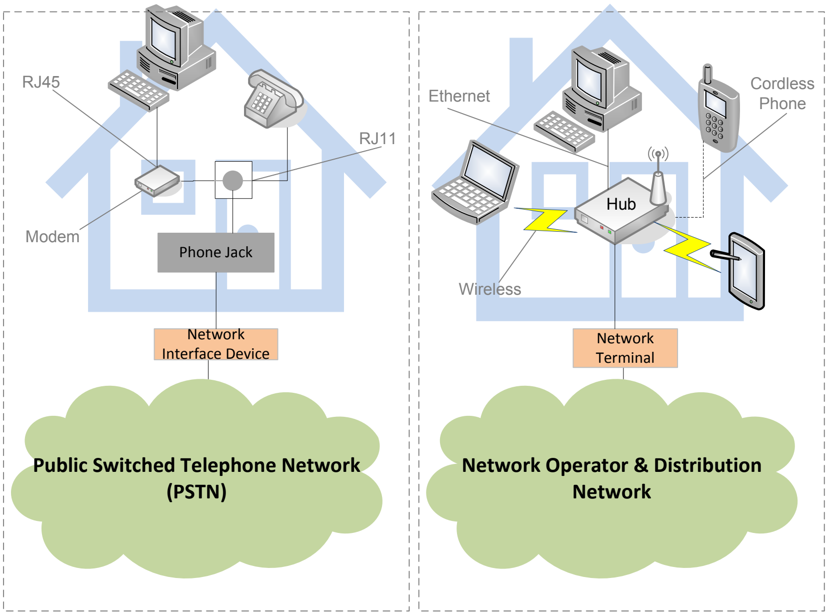


Figure 2.1: Wired and Wireless Connections Home Area Network - Showing the limitations of Wired Network from mobility point of view

.

2011], however, the most popular model is controller-based *HAN* system [Ramanathan and Gusella, 1995]. In controller based system, the digital switch acts as the communications hub, addressing and routing voice data traffic throughout the home. The controller supplies a robust home network for voice and data with high bandwidth capacity. It is the bridge between the transport network element serving the home from the Internet service provider and the wireless home network. Networked devices and appliances require no wires or fixed wired jacks. Data and voice services, including internal device-to-device communications, are commonplace. The controller based switching center is software driven so that new networking requirements in the home can be met without drastic or rudimentary changes [Dixit and Prasad, 2007, Ford et al., 1997]. Figure 2.1 shows the wired and wireless connections with in a typical *HAN* [Henricks, 2000].

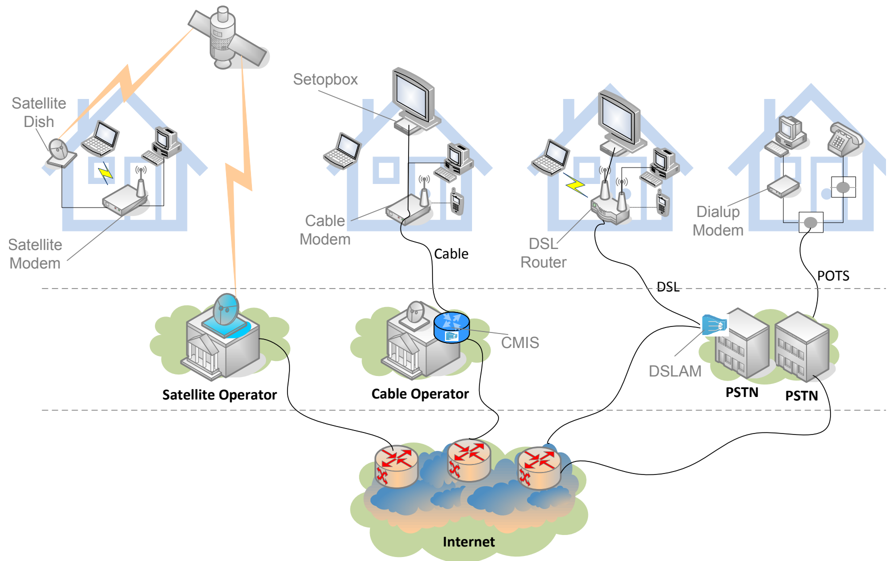


Figure 2.2: POTS, DSL, Cable and Satellite ISP connections for Home Area Networks.

2.1.1.2 HAN Networking Technologies

Popular wired technologies are IEEE 802.3 standard, Integrated Services Digital Network (ISDN), Digital Subscriber Line (DSL) and broadband [Dixit and Prasad, 2007]. Another route to wired networking is power-lines that are readily available as network transport elements throughout a home but wired technologies are rapidly being replaced by Wireless [Vaxevanakis et al., 2003]. Traditional wireless home-networking technology is used for line-of-sight, infra-red, unidirectional, hand-held controller applications. Most commonly used wireless networking technology is radio frequency (RF) focused. In controller-based HAN system, the microprocessor-based digital sends the home network traffic through a powerful on-board RF transceiver. The transceiver is based on patented digital spread-spectrum technology and has an effective reach of several hundred feet from the home. Popular wireless standards include IEEE 802.11, Home-RF, Blue-tooth, and Standard Wireless Access Protocol (SWAP). Two types of spread-spectrum radios are in common use today: Frequency-Hopping Spread-Spectrum (FHSS) and Direct-Sequence Spread-Spectrum (DSSS) radios [O'Driscoll, 2000]. Wired networking technology provides faster and secured connections compared to wireless but has immense mobility and scalability issues.

A *HAN* system can be connected to Internet via Digital Subscriber Line (DSL), Cable, Dialup or Satellite (as shown in Figure 2.2).

Cable Service: It allows *HAN* system to connect with internet via a local cable television line and transmit and receive data over a cable modem into your computer through the wired (cabled) or wireless ethernet card.

DSL Service: It is a family of technologies (Asymmetrical DSL - high download, slow upload speed and Symmetrical DSL - similar download and upload) that provides digital data transmission over local telephone lines. It typically works by dividing the frequencies on a single telephone line into two primary "bands". The internet data is carried over the high frequency band, while the voice is carried over the lower frequency band.

Satellite Service: In a two-way satellite Internet connection, the upstream data is usually sent at a slower speed than the downstream data arrives. Thus, the connection is asymmetric. A dish antenna transmits and receives signals. No phone line or other wired connection is required. A satellite modem in connection with satellite dish connects *HAN* with Internet.

Dial Up Service: Dial up connection dials a number using Plain Old Telephone Service (POTS) to connect with Internet. Dial up technology is no longer suitable for *HAN* and rapidly obsoleting due to its speed limitation.

Establishing a *HAN* system requires management of its operations and resources. Network management is essential to ensuring the day-to-day normal network functionality and security. This requires methods that can detect and respond to network issues in real time, as well as predict possible issues in the future. In following sections, we briefly discuss traditional and some sophisticated network management approaches.

2.1.2 Network Management

Network management is highly diverse field and it has various different meaning in different work domains. In some cases, it involves only a network monitoring activity with some monitoring and analysis tools. In other cases, it involves more complex tasks like profiling, polling of network devices, and generating real-time graphical visualization and meaning of network changes and traffic. In general, network management is a service that employs a variety of tools, applications, and devices to assist network managers in monitoring and maintaining networks [Clemm, 2006]. Network management can be segmented into two categories: Tactical and Strategic. Tactical network management relates to proactive and reactive situations, such as network failures, congestion, and unacceptable service quality. The tasks include troubleshooting, configuration, and adjusting traffic flows. Strategic network management involves a long-term perspective that is oriented toward adequate planning to avoid shortages as the network grows. Strategic tasks use information to adjust operations, optimize quality, and manage facilities to reduce overall operational costs. Therefore network management is considered as an umbrella term that refers to a set of activities for the operation, administration, maintenance, and provisioning (all collectively called as network management processes) of the networking system. The network management processes are interrelated to each other and are usually performed with the help of management tools. The key processes are:

Operations process deals with the functional enabling and monitoring of different devices and services to identify network related problems;

Administration process deals with the assignment of resources in the network and their control;

Maintenance process deals with repairs and upgrades of network devices and services;

Provisioning process deals with configuring resources for a given network service.

Functions that are performed as part of network management include monitoring, planning, allocating, deploying, coordinating, and controlling the resources of a network for authorization, configuration management, fault management, security management, performance management, bandwidth management and accounting management. In this regard, ISO FCAPS (Fault, Configuration, Accountability, Performance and Security) model [Surhone et al., 2010] is a major contributor to network management for outlining the main tasks. A network management system comprises managed entity (devices with software that enables the sending of alerts when network issues are identified; for example, any change of configuration on the network device) and management entity (process that is programmed to react to alerts by executing a predefined set of actions). A typical architecture of a network management platform is made up of a common set of relationships and structure that exists between managed and management entities. A management entity can take on one of the two possible roles: manager or agent. The managed entities are usually monitored by network management agents (to carry network management data and report network transmission problems) and network manager that controls a set of management agents and ensures that these agents collect the appropriate information. Network management protocols carry network management data between the managed and management entities. To simplify network management tasks, quality of service management and network security monitoring are two major tasks in home networks, which are briefly discussed in following sections.

2.1.2.1 Quality of Service Management

Quality of Service Management (QoSM) is network management technique to configure and maintain services and network resources to achieve network quality requirements [Cisco Systems Inc, 1999, Guichard, 1999]. QoSM is usually attained through controlling the traffic and reserving the resources. It uses priority rules to provide a certain level of service based on the priority of different classes of users, applications and traffic flows. For guaranteed services, it allocates resources to particular traffic class. Quality of Service (QoS) is a collective measure of the level of network service provided to a user, which can be characterized by many performance parameters of a network:

1. Timeliness characteristics;
2. Capacity characteristics;
3. Error-related characteristics;
4. Reliability characteristics;
5. Security characteristics;

6. Cost characteristics, etc.

However, there are three most commonly used parameters to quantify the quality of service for video and audio network traffic:

Delay – It refers to a lapse of communication data in terms of time between two points resulting from queuing, processing and congestion;

Jitter – It refers to variations in a data communication resulting from fluctuations in the flow, also called as distortion;

Loss – It refers to loss of the transmitted data packet usually resulting from data congestion at some point along the network path.

QoSM helps to set-up and evaluate QoS goals; QoS goals are transformed into configurations, which act as networking rules. A QoSM methodology entails base lining the network deploying relevant QoS techniques and evaluating QoS results. A QoS level, also referred as service level, is network QoS capability to deliver service needed by network traffic. QoS can be graded into three basic levels [Cisco Systems Inc, 1999]: (a) best-effort service level, this is also known as lack of QoS; best-effort service is basic connectivity with no guarantees; (b) differentiated service level, this also known as soft QoS. Different traffics are classified and treated according to their classification; and (c) guaranteed service level, this also known as also called hard QoS–it reserves network resources for specific traffic.

There are two types of QoS [Guichard, 1999]: provisioned QoS and signalled QoS. Provisioned QoS is statically achieved by configuring network resources for the flow of different types of traffic. Most of QoS approaches are static using priority queues, data flow control and packet marking, etc. In signalled QoS, which is also referred as dynamic QoS, the Internet Protocol (IP) packets contain signalling information describing the specific QoS necessary for the application to function. The Resource Reservation Protocol (RSVP) protocol is mostly used for signalled QoS. QoS manages traffic in two ways [Guichard, 1999]: per-flow QoS, and per-aggregate QoS. Flow is unidirectional stream of data, which receives individual treatment in per-flow QoS. In per-aggregate QoS, two or more unidirectional data streams are put under some classification based on some traffic characteristics, e.g., all packets using Transport Control Protocol (TCP),

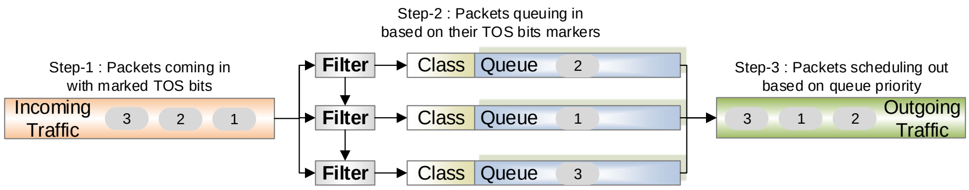


Figure 2.3: IP packets being queued and scheduled on Internet gateway device based on Type of Service (TOS) bits marked in packets' headers.

and the class of different flows receives individual QoS treatment. Provisioned QoS used aggregated QoS traffic management technique and signalled uses per-flow technique. Both QoS techniques can be used in other ways with per-flow and per-aggregate but in that case they may not make much sense. The QoS traffic management has four fundamental elements:

1. Traffic identification scheme;
2. Traffic marking scheme;
3. Traffic filtering scheme;
4. Traffic queuing scheme.

Traffic identification is usually based on the information available in traffic packets and the QoS implementation technique, e.g., source and destination IP addresses, ports, protocols, etc. To provide preferential QoS treatment to a type of traffic, it must be identified first. Traffic marking is not compulsory because traffic can be filtered for QoS treatment even if it is not marked but it really depends on how QoS is implemented. Generally, "type of service" (TOS) bits in IP packet header can be marked for different types of QoS treatments. Traffic identification and marking together called as traffic classification. When the packet is identified but not marked, classification is said to be on a per-hop basis. This is when the classification pertains only to the device that it is on, not passed to the next router. The QoS implementation technique largely depends on the user QoS requirements. User can select priority-based or class-based queues (as shown in Figure 2.3).

2.1.2.2 Network Security Monitoring

The process of a network security monitoring is performed by a wide range of devices belonging into the category of the Intrusion Detection System (IDS). Such devices are focused on identifying and reporting possible security incidents. We can divide IDS technologies into the four groups according to the types of events that they monitor [Tracy et al., 2007]: network-based IDS, wireless IDS, host-based IDS and network behaviour analysis.

In a comparison to the traditional network-based IDS, Network Behaviour Analysis (NBA) system uses statistical information about flows (number of packets, amount of transmitted data, used transfer protocol, etc.) instead of analysing a content of the transmission. This approach allows analysing of unencrypted as well as encrypted data in the same way. IDS technologies use more different methods, usually together, to detect security threats. Generally they can be divided into the following three categories [Tracy et al., 2007]: signature-based detection, stateful protocol Analysis, and anomaly-based detection.

2.1.3 Policy-based Network Management

Policy-based Network Management(PBNM) gained widespread attention in late 1990s when the Internet Engineering Task Force (IETF)[1] formed a Policy Framework Working Group (PFWG) to define architecture and information model for policy-based management of Quality of Service (QoS) in "Internet Protocol" (IP)-based networks [Waters et al., 1999].

The Distributed Management Task Force (DMTF)[2] also developed information models for network and policy management applications and later joined IETF-PFWG to standardise IETF policy information model [Moore et al., 2001]. Many IETF and DMTF standards have been used in policy-based management of networks [Boutaba and Aib, 2007]. This has been formalised as a language for the Common Information Model (CIM) [de Vergara et al., 2005].

[1] `https://www.ietf.org/`
[2] `https://www.dmtf.org/`

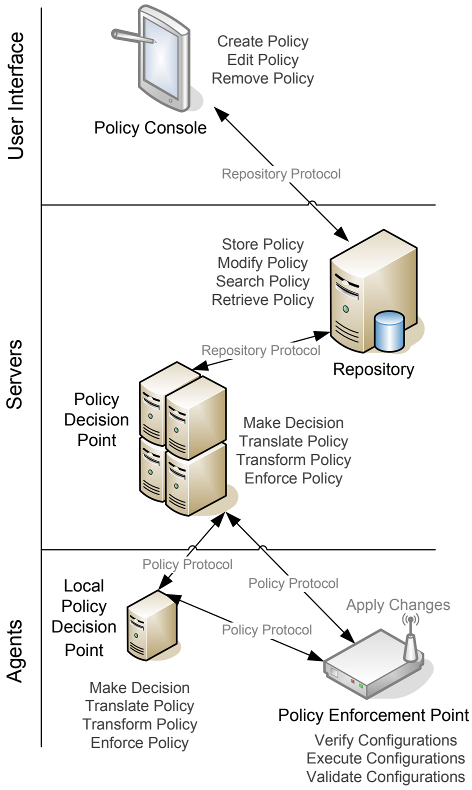


Figure 2.4: Policy-based Network Management Architecture [Waters et al., 1999]: policy console to specify policies; Repository to store policies; Policy Decision Point for evaluation of policies; Policy Enforcement Point to apply policies.

IETF-PFWG has defined a policy management architecture (as shown in Figure 2.4) that is considered as best approach for internet policy-based management. The architecture consists of a Policy Console (PC), Dedicated Policy Repository (DPR), Policy Decision Point (PDP), Policy Enforcement Point (PEP) and policy communication protocols [Boros, 2000, Yavatkar et al., 2000]. According to Davy et al. [2009], the functionality of PEP can be further subdivided into policy execution point (PXP) and policy verification point (PVP). The IETF PBNM architecture works best under the expert intervention. In a typical home area network (*HAN*) expert intervention may not be available, therefore the model components such as policy console lose their usability in the context of *HAN*.

PBNM provides a rules driven system; these rules are called policies (as shown in Fig-

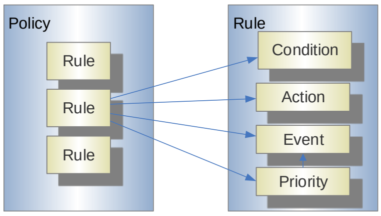


Figure 2.5: Policy Structure: Showing component of a policy rule, containing event (when policy will be triggered), condition (what is the criteria for triggering policy
, action (what needs to be done when) and priority.

ure 2.5). There is no standard way of defining policies but there are some definitions put forward by academic researchers. According to Saperia [2002], a policy is a predetermined action statement for such action patterns that are repeated by entities involved in a network under certain systems conditions. Westerinen et al. [2001] define a policy as a goal or course of action to guide present and future network decisions. More concisely, a policy is set of rules to administer, manage and control the access to network resources and services. "IF Condition-THEN-DO Action" rule structure is simplest form of policy rule which says "on occurrence of an event, if condition is met then do action" also known as Event-Condition-Action (ECA) [Liu, 2009]. The network policies can be classified generally into the following six broad categories [Boros, 2000]:

1. Performance Management Policies;
2. Security/Access Control Policies;
3. Quality of Service Policies
4. Administrative/Configuration Management Policies;
5. Fault Management Policies;
6. Customized/Event Condition Action Policies.

A policy language may use the constructs from an information model to entail the policy compositional elements. The DMTF has defined Common Information Model (CIM) based policy model [de Vergara et al., 2005] with the help of IETF; the data model standardises the ECA-based policy rules [Liu, 2009]. The CIM policy model defines

Table 2.1: Policy Abstraction Levels.

Abstraction	Description
Business Level	These policies are domain, mechanism, device and instance independent. They contain no specification how policy would be realised and no system and network elements are mentioned to support the policy.
Domain Level	These policies are mechanism, device and instance independent, and they are translated into domain specific format. Policies are not assigned to any specific device or network element.
Mechanism Level	These policies are device and instance independent, specified to realize a mechanism. They cover mechanism implementation details.
Device Level	These policies are instance independent. These policies involve device specific parameters and mechanism implementation details.
Instance Level	This is the most specific expression of a policy. All parameters are expanded to all network elements that are involved in enforcement process of this policy.

a very high-level policy model–an ECA policy rule contains four major components: Event, Condition, Action and Priority. The Event indicates the context in which a policy rule is relevant. The Priority indicates the relative importance of the policy rule to avoid policy conflicts. The Condition indicates the state when policy rule will be applicable. The Action part of a policy rule specifies the action to be taken if the rule is applicable. A policy rule can be defined at different levels of network system, which essentially defines policy levels. These levels are sometimes called a policy hierarchy [Damianou, 2002] or abstraction levels as defined in Table 2.1. Abstraction levels represent different views on policies, relationships between policies at different levels of its hierarchy, or abstractions of policies for the purpose of information entailment. Damianou [2002] consider three major levels but Boros [2000] consider five policy abstraction levels and Davy et al. [2008] refer to them as the "policy continuum". Each abstraction level defines policy scope within network. These abstraction levels are interrelated and can be defined in terms of each other.

PBNM communication between PDP and PEP can be realised in many ways, e.g., HTTP, COPS and SNMP, etc. Two commonly used methods are Common Open Policy Service (COPS) [Durham et al., 2000] and Simple Network Management Protocol (SNMP) [Case et al., 1990]. In *HANs*, the network devices are so cheap that they usually do not provide any sophisticated management interface. Moreover, COPS and SNMP have inherent implementation issues i.e. security leaks and slow performance [Boros, 2000]. PBNM is extensively used in Core Networks, where well trained domain experts define policies to manage network services and resources. PBNM, if configured sensibly by *HAN* users through intelligent interfaces that can hide underlying network complexity, there is some hope that *HAN* can be managed according to user requirements.

2.1.4 Semantic Technology and Inference

Semantic technologies [Barabasi, 2002, W3C, 2004] have played a vital role in abstracting the details and lessening the gap between humans and complex network systems. With the help of semantics, *HAN* management systems and related technologies can become more understandable to ordinary *HAN* users and, in the near future, they can also be in a position to help controlling their home systems without requiring them to know the details and complexity of underlying network management systems. However, there is no single ideal knowledge representation technique suitable for all applications when building practical intelligent systems [Gaaevic et al., 2006]. The traditional techniques most frequently used to represent semantics in an intelligent systems include object-attribute-value triplets, uncertain facts, fuzzy facts, rules, semantic networks, and frames.

Object-Attribute-Value (OAV) triplets are a technique used to represent facts about objects and their attributes. More precisely, an OAV triplet asserts an attribute value of an object;

Uncertain Facts are used an as extension of OAV triplets that allow uncertainty of facts to be described. A certainty factor is a numeric value assigned to a statement that represents the degree of belief in the statement;

Fuzzy facts represent uncertainty using the imprecise and ambiguous terms commonly found in natural languages. The fuzzy set representing member groups is defined by a corresponding membership function. This function can be used to calculate the actual numerical value of a membership in the fuzzy set, on a scale from 0 to 1;

Rule-based is a knowledge representation technique and a structure that relates one or more premises (conditions, or antecedents), or a situation, to one or more conclusions (consequent), or actions;

Semantic networks are graphs made up of objects and concepts (the nodes in the graph) in some specific domain of knowledge, connected by some type of relationship/properties (the links/arcs), which could be part-of, type-of, is-a and has-a relationship. Semantic networks contain OAV triplets;

Frames are similar to classes and objects in semantic networks, however unlike semantic networks, frames include procedural knowledge as well, in the form of facets. Facets are typically attached to slots (properties) and contain procedures (methods) that are invoked automatically when the value of the slot is changed, or when it is needed (read).

2.1.4.1 Knowledge Representation Techniques

The central component of any knowledge-based intelligent system is its knowledge base [Gaaevic et al., 2006]. The knowledge base contains a set of sentences - the units of the knowledge represented using one or more knowledge representation techniques and the sentences are expressed in a knowledge representation language. A knowledge representation language can be used to change or query the knowledge based. Knowledge representation languages should be capable of both syntactic and semantic representation of entities, events, actions, processes, and time. There are many forms of knowledge representation languages but the following three are most popular:

Logic-based [van Otterlo, 2009]: (a) A proposition is a logical statement that is either true or false. An OAV triplet is a more complex form of proposition, since it has three distinct parts. Propositional logic is a form of symbolic reasoning. It assigns a symbolic variable to a proposition. The truth value (true or false) of the variable represents the truth of the corresponding statement (the proposition). Propositions can be linked by logical operators (AND, OR, NOT, IMPLIES, and EQUIVALENCE) to form more complex statements and rules. (b) First-order logic extends propositional logic by introducing the universal quantifier and the existential quantifier. It also uses symbols to represent knowledge and logical operators to construct statements. Its symbols may represent constants, variables, predicates, and functions. (c) Description logic contains a terminology (the vocabulary of the application domain) in a part of the knowledge base called the

TBox, and assertions about named individuals (using the vocabulary from the TBox) in a part of the knowledge base called the ABox. The vocabulary consists of concepts and roles. Concepts denote sets of individuals. Roles are binary relationships between individuals. There are atomic concepts and roles (names of concepts and roles) and complex concepts and roles (terms for concepts and roles). The complex concepts are built using descriptions expressed in the corresponding description logic language and are assigned names in the TBox;

Frame-based: The central tenet is a notation based on the specification of frames (concepts and classes), their instances (objects and individuals), their properties, and their relationships to each other. Frame-based languages are usually suitable for representing knowledge that does not change;

Rule-based: Rules are popular in intelligent systems. The Rule Markup Initiative (RMI) has taken steps towards defining "Rule Markup Language" (RuleML) [Antoniou, 2002], a shared language based on XML, contains rule schemas for production rules (the If-Then), integrity rules, reaction rules, derivation rules and transformation rules.

2.1.4.2 Knowledge Representation as Ontology

The ontological engineering, from the viewpoint of computer science rather than the field of philosophy, is called metaphysics [Gruber, 1995]. Informally, an ontology may be described as a representation of a conceptualisation (an abstract viewpoint) of a domain of discourse. Thus, an ontology representation format should include a technique to represent domain concepts and also a technique to represent the relationship between the domain concepts. Many ontologies may be built using different ontology representation formats to portray a single conceptualisation of a domain of discourse. It is imperative that the ontology representation format used is capable of describing the conceptualisation. Conceptuality is understood to be an abstract view of the domain of interest and explicit means that the conceptualisation is clearly defined.

Ontologies have been considered from different, often complimentary, perspectives. Therefore, many ontology classification have been created [Guarino, 1998]. Guarino classifies ontologies into four types according to their generality level. The types of ontologies in this classification are:

Top-level ontologies, which describe general concepts such as space, time, matter and object. These ontologies are independent of a particular problem or domain. Therefore, a well defined top-level ontology has the potential to have a large population of users. These types of ontologies are also called upper ontologies or foundational ontologies;

Domain ontologies, which define the conceptualisation of a generic domain, for example, medicine or geology. Domain ontologies are often specialisations of a top-level ontology. The Gene Ontology is an example of a domain ontology;

Task ontologies, which defined the conceptualisation of a task or activity; for example, troubleshooting a network resource. Task ontologies are often specialisations of a top-level ontology;

Application ontologies, which describe a conceptualisation based on both a domain and a task. Thus, an application ontology is both a specialisation of a domain ontology and a task ontology.

Jurisica et al. [2004] define a classification of ontologies according to the nature of real-world issues modelled by an ontology. The types of ontologies in the classification are:

Static ontologies, which describe conceptualisations of invariable aspects of the real-world. Conceptualisation represented with static ontologies consider the entities that comprise the real world to be unique and immutable; they believe ontology have a lifetime, attributes and relationships with other unique entities;

Dynamic ontologies, which describe conceptualisation of aspects of the real-world which change with time;

Intentional ontologies, which facilitate conceptualisations of goals, intents and beliefs to be expressed and reasoned about;

Social ontologies, which describe social structures, such as organisational structure, affliction networks and interdependences. Actor, role and responsibility are terms likely to appear in a social ontologies.

RDF The "Resource Description Framework" (RDF) [Pulido et al., 2006] is the first language developed especially for the Semantic Web . RDF is developed as a language, realized in XML, for adding machine-readable meta-data to existing data on the Web. RDF Schema extends RDF with basic ontological primitives such as classes, properties and instances. In addition, the instance-of, subclass-of, and subproperty-of relationships have been introduced, allowing class- and property hierarchies. In RDF, all concepts and resources can be specified using Unicode, and uniquely identified using URI's.

While RDF is a language for describing resources with classes, properties and values, it has no way of defining the class hierarchies, property hierarchies and property restrictions. RDF Schema is an extension of RDF that provides a vocabulary for defining the application-specific vocabulary used by RDF. The resources described in a RDF document can be seen as instantiations of definitions in a RDF Schema. A document containing a combination of RDF and RDF Schema is called a RDF-S document.

OWL Ontology Web Language [Pulido et al., 2006] is an expressive ontology language which addresses the limitations of pure RDF-S. OWL serves as an extension of RDF-S and adds more vocabulary for describing properties and classes. The language provides three increasingly expressive sub-languages designed for use by specific communities of implementers and users:

1. OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. Compared with RDF-S it adds local range restrictions, existential restrictions, simple cardinality restrictions (only 0 or 1), equality, and property characteristics (symmetric, transitive, inverse);

2. OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL adds full support for negation, disjunction, cardinality restrictions enumerations, and value restrictions;

3. OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example a class can be treated simultaneously as a collection of individuals and as an individual in its own

right. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

SWRL Semantic Web RuleML Language (SWRL) [Horrocks, 2011] is an extension of OWL-DL which adds the expressive power of rules to OWL. SWRL enables Horn-like rules [Gupta, 1999] to be combined with an OWL knowledge base. However, whereas Horn rules have a conjunction of atomic formulas in the antecedent of the rule and a single atomic formula in the consequent of the rule, SWRL allows any OWL class description, property or individual assertion in both parts. Since SWRL combines the full expressive power of function-free Horn logic with an expressive description logic language, the key inferences tasks (e.g., satisfiability and entailment) are in general undecidable for SWRL. Another rule language for the Semantic Web is F-Logic [Kifer and Lausen, 1989]. Rules in F-Logic are similar to Horn rules, with the distinction that besides atomic formulas, F-Logic rules also allow molecules in place of atomic formulas. The main difference between SWRL and F-Logic is that in SWRL, the rule language is seen as an extension of the ontology language OWL DL, whereas in the F-Logic proposal, ontologies are modelled using rules.

2.1.4.3 OWL based Ontology

An OWL ontology is built up by three components: Classes, Properties, and Individuals. These components are analogous to concepts, relations and instances, respectively. The main components of an owl-based ontology are described as following:

Individuals: Individuals represent objects in the domain we are interested in and can also referred to as instances of classes. It is important to note that OWL does not use the Unique Name Assumption (UNA). This means that two different names could actually refer to the same individual;

Properties: There are two types of properties: Object and data. The object properties are the relation between two individuals, that is a property links an individual to another. For example, the property `hasDevice` can link the two individuals Ben and Mobile together. A property may be functional, symmetric or transitive. If a property is functional there can be at most one individual that is related to the individual via the property. For example the property `hasSoleOwner` is a functional property (you can only have one sole owner). A symmetric property can be defined as follows: If individual A is related to individual B via property P, then, if P is symmetric, B is also related to A via P. A transitive property can be defined as follows: If individual A is related to individual B via property P, and B related to individual C via P - then, if P is transitive, A is related to C via property P. The data properties are the relation between a individuals to literals that characterises a certain trait of individual. For example, `hasIPAddress` is an data property of a device and its value can be string of character that represents an IP address of device;

Classes: OWL classes can be interpreted as sets containing individuals. They are described using conditions that states precisely what requirements needs to be in place in order for an individual to be a member of the class. Classes may be organized in a superclass-subclass hierarchy, known as a taxonomy. Using a reasoner, this taxonomy can be computed automatically;

Rules: Rules are considered to be a major issue in the further development of the Semantic Web . They can be used in ontology languages, either in conjunction with or as an alternative to description logic's, to draw inferences, to express constraints, to specify policies and/or to react to event/changes. With rules one can express knowledge in the form of "if-then".

2.1.5 Rule-based Reasoning

There exist many modes of formal reasoning but one of the most popular kinds is rule-based reasoning [Bryant, 2009, Clark, 1988], which is an obvious choice to accommodate home users in the *HAN* management with the help of inference rules. Rules basically explicate the manner of performing reasoning, which are then used by the control systems within *HAN* to make intelligent decisions at the time of need. Assuming *HAN* management system as an expert system, we can divide it into three main components [Griffin and Lewis, 1989]: knowledge base, inference engine and working memory as shown in Figure 2.6.

An inference rule containing a set of premises (facts/evidences) and a conclusion, represents a presumed rationale behind a piece of knowledge in the knowledge-base. An inference rule has two structural constituents [Poole, 1997]: antecedent and consequent. Antecedent is the left-hand-side of the rule and it uses logical operators to combine propositions in antecedent with the consequent of inference rule. Consequent is the right-hand-side of the rule and specifies a sequence of actions. There exist three main types of assertions for antecedents: conjunctive (AND), disjunctive (OR) and negative.

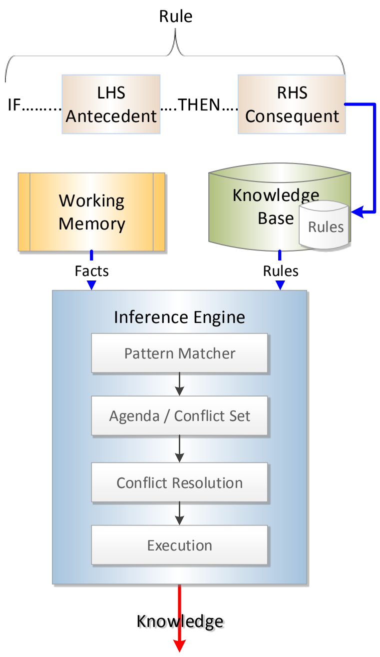


Figure 2.6: An inference engine takes facts/evidences and rules and process them to infer new knowledge using deductive reasoning.

In our research work, we used SWRL [Horrocks, 2011, Horrocks et al., 2004] to represent inference rules; it only supports conjunctive antecedents and single action consequent.

Inference is a rule-based reasoning technique, which is widely adopted in the design of rule-based intelligent systems and works with a semantic reasoner. A rule-based semantic reasoner or more specifically an inference engine renders over knowledge and rules specified in the knowledge base and draws conclusions for an intelligent *HAN* management. We assumed that human provides basic knowledge facts and specifies the inference rules, and based on the given facts and *HAN*-specific knowledge, inference engine can infer new knowledge and can also learn. It is important to note that SWRL does not facilitate to update the existing knowledge in the knowledge base due to its monotonic nature. Jess reasoner [Friedman-Hill, 2003, Laboratories, 2009], a rule based inference engine performs forwarding chaining by default, which is based on deductive reasoning. Inference engine implements the reasoning process by finding rules in the knowledge base that correspond to the facts or data in the working memory. All rules that match the current problem state (criteria) are selected into a conflict set (rules to be executed). A single rule from the conflict set is selected and action part of the selected rule is performed. It may result in changing the working memory and so does the knowledge base if required in an ideal situation.

Typically, most of the formal logic systems adhere to the rules of classical reasoning [Sullivan, 2005] and monotonicity [Truszczynski, 1991] is one of major principles. It is a reasoning property that states that new knowledge facts and rules added to the knowledge base should be admissible and should not affect the state of previously added facts and rules. However, to address the challenges of changing requirements and system adaptability, we require a non-monotonic reasoning system [Egly and Tompits, 1997] in the *HAN* management system so that the inferred knowledge should also be reflected in the knowledge base. However, introducing a non-monotonic logical system in *HAN* can induce many levels of other logical and semantic conflicts as well as inconsistencies in the knowledge base. On the other side, a monotonic logical system loses its usefulness in the *HAN* management system otherwise. The literature indicates that there are a number of successful attempts of using non-monotonic reasoning approach with the combination of ontology-based knowledge systems [Antoniou, 2002, Esposito, 2007].

SWRL follows the monotonicity principle; hence, SWRL rules cannot be used directly to modify existing information in the knowledge base. In *HAN* management system, at the time of rule specification, knowledge rules can be very abstract, which makes it quite difficult to analyse the conflicts with already specified rules. A conflict in the inference rules can be caused by the presence of false premises, which can induce wrong conclusions and rules may become in contradicting state to each other. However, this problem is out of the scope this thesis and we mainly focused on the problem when established premises are correct and inference rules still result in contradicting state, making the logical system inconsistent. It is important to understand the inference conflict before we can discuss our solution strategy. When an inference engine encounters several rules that match the working memory (triggering facts) but only one has to be selected, is termed as an inference conflict. There are several strategies to resolve the inference conflict [Sanborn, 1987], for example:

- Refraction: once the rule has been read, it is not used again;
- Recency: use the rule that has been used recently in such situation;
- Specificity: use the rule with the more specific condition (more facts);
- Priority: assign priority to rules (e.g., based on rank, utility, probability, cost, etc.) and choose the one with the highest priority;
- Parallel: read all rules with separate lines of reasoning.

Firstly, it is important to note here that traditional inference conflict resolution strategies focus on execution pattern of all selected rules, which is not as significant as the execution of right inference rule only among the conflict set. Secondly, none of above mentioned conflict resolution strategies address the problem of semantic conflicts. In semantically conflicted rules, it may also be required to defer the execution of other rules that may cause inconsistency or wrong inference in *HAN* management system. Thirdly, this is a problem of reasoning with uncertainty (predicting which rule is most appropriate for execution). Therefore we emphasize developing a conflict resolution strategy that first learns the context and then helps in selecting an appropriate inference rule for execution from the conflict set using some intelligent way for reasoning with uncertainty. Most of the conflict resolution strategies mentioned above are impractical in this scenario, e.g., refraction and recency may cause execution of a non significant rule, which may lead to wrong inference state. Similarly, parallel execution may cause contradicting state of working memory and knowledge base in non-monotonic logical system. The priority-based strategy has some potential but it may not work in complex situations, e.g., when both rules have equal priority.

2.2 State of the Art

In this section, we review the work that has been done in the area of user-centric *HAN* management using policies and use of semantics for *HAN* network management and policy processing. The research challenges that we want to address are:

1. Framework blueprint for User-centric Policy-based *HAN* Management;
2. Involving user in *HAN* network management process;
3. Taking user requirements in the form of policies;
4. Transforming user policies to network configurations with the help of semantics;
5. Retrieval of policy semantics from *HAN* system semantic model;
6. Mapping of policy semantics from one level of *HAN* system to another level;
7. Making user as part of *HAN* control loop;
8. Resolving semantic conflicts caused by user defined rules

The use of policies and semantics in home area network management is described to present the state of the art and changing trends in *HAN*.

2.2.1 User-Centric Home Network Management Using Policies

In last decade, convergence of network enabled devices and complex network services have changed the traditional view of home area networks (*HANs*). Home area networks have evolved to become complex networks; however, available management tools are still primitive [Sventek et al., 2011]. A study conducted to review *HAN* management tools [Yang and Edwards, 2010] showed that the tools provided by vendors (of devices used in *HAN*) have enormous usability problems. The current monitoring tools have limited device management features, which mostly can not be customized to meet users' requirements. A similar study [Grinter et al., 2009] is conducted in the US and the UK to identify complexities that *HAN* users face when setting up their network infrastructure. The study elicited some major requirements for new management tools with the radical reconsideration of current management issues in *HANs*. Recent research developments are aiming to realise the vision of smart home networks in next decade. However, most of the current attempts for well connected home networks are still far behind a reality. Many of the proposed approaches [Chetana Sarode, 2012, Gaul and Ziefle, 2009, Meyer and Rakotonirainy, 2003] lack substantial user involvement in their proposed solutions.

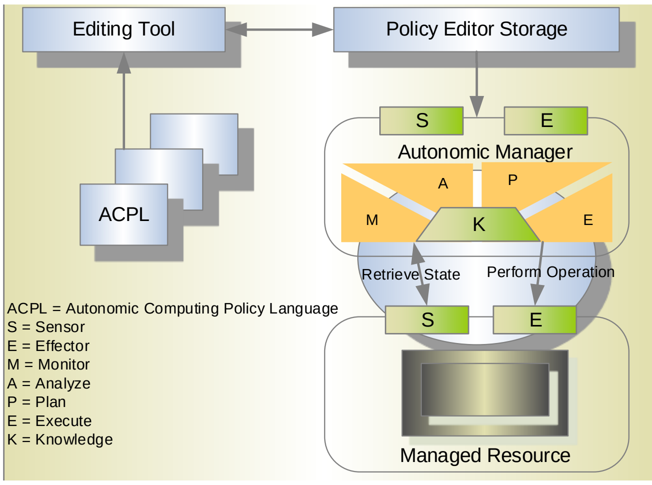


Figure 2.7: Policy Management for Autonomic Computing (PMAC) abstract model for creating and enforcing policies and automating these business scenarios. PMAC runs on observations collected from different sources to achieve high-level business objectives by accomplishing system goals dynamically.

These management systems (lacking fine grained user control) most of the time tend to make decisions on the behalf of home users, some times disregarding the actual requirements [Brennan et al., 2009, Han and Lim, 2010, Liu et al., 2006, Park et al., 2006], which results in losing viability in typical smart management scenarios. Hence these systems also become inadequate to adapt to changing user requirements.

As described in $2.1.3, Policy-based Network Management (PBNM) is a promising technique that is widely accepted and is being used in enterprise networks [Verma et al., 2001]. The use of policies in HANs is at a very preliminary stage, mostly focusing on network access control and some network quality features. There are some proposals for user-centric *HAN* management using policies [Sventek et al., 2011], but most techniques are in the incubation phase.

Autonomic computing is integral part of PBNM [Westerinen et al., 2001]. In an autonomic policy-based network, one controlling element in the network must be capable of autonomic computing, however, it is not an appropriate criterion to classify a network as an autonomic. Many strategies have been suggested by researchers for self-managing networks, most noticeable work model in autonomic PBNM is PMAC [ibm, 2005] (as shown in Figure 2.7). The Policy-based Management for Autonomic Computing (PMAC) is an abstract middleware platform, which determines behaviour of managed systems and guides the managed resources to follow appropriate rules. The managed resources are configured dynamically to achieve the certain network goals and to make working environment adaptive. Agrawal et al. [2005] also defines another abstract policy middleware architecture that uses a concrete information model to specify the semantics of policy operations. The used information model is based on CIM [de Vergara et al., 2005] defined by DMTF policy working group.

There have been quite recent attempts to simplify the process of *HAN* management for ordinary home network users. Pediaditakis et al. [2012b] propose a management framework for home networks that uses "comic strip" story styled policy specification. This approach succeeds in abstracting *HAN* complexity and holds great potential for the future but it does not address the challenges of semantics computation for the abstract concepts that are familiar to the ordinary home network users. The same authors proposed a policy-based configuration service [Pediaditakis et al., 2012a] for home management, which potentially facilitates the development of new user-centric management services for home networks. The configuration service works on a model that provides higher level views of home network to user to control network behaviour. However, the employed model does not elucidate the process of correlating the *HAN* system with users. Another policy-based service management framework is proposed by Brennan et al. [2009] for the management of end-to-end communications services in federated *HAN*; however the architecture does not focus on automating the configuration process within a single *HAN* environment. Other similar *HAN* management issues are addressed in [Bae et al., 2003, Berl et al., 2009, Bertran et al., 2009, Fallon and OSullivan, 2012, Ricquebourg et al., 2006, Siddiqui et al., 2009] but they also do not address user requirements or seek to engage non-technical users in controlling the *HAN* systems.

2.2.2 Semantic-Aware Network Management Using Policies

The use of semantics in managing networks is quite new [Löpez De Vergara et al., 2009]. The semantic technologies [W3C, 2004] have played a vital role in abstracting the details and lessening the gap between humans and complex network systems. With the help of semantics, the *HAN* management systems and underneath technologies can become more understandable to ordinary *HAN* users and in near future, they can also be in a position to help controlling their home systems without requiring them to know the details and complexity of underneath managing systems. However, establishing an accurate user-driven control system as a part of *HAN* management system (that can take intelligent decisions) is extremely challenging. However, the use of semantics in managing networks is quite new [Fallon and O'Sullivan, 2014, Löpez De Vergara et al., 2009] and to the best our knowledge, no semantic work has been done yet to manage *HANs* using network monitoring data. However, there is fairly sizeable of work on semantics in other network domains. Semantics have been applied to monitoring end-to-end services [Keeney et al., 2011], analysis of network payloads [Krueger et al., 2011] and the management of network infrastructure [Xiao and Xu, 2006]. Some papers also discuss techniques to use semantics in ontological [Tran et al., 2007] and non-ontological forms [Krueger et al., 2011] but OWL-based semantic models are more prevalent having been used extensively for analysing online social networks [Ereteo et al., 2009] and underlying network infrastructure [Fuentes et al., 2006, Yang and Chang, 2011].

The vision of semantic-aware policy engineering provided by Lewis et al. [2005] has been manifested in many forms in recent attempts (e.g., [Löpez De Vergara et al., 2009]). On similar lines, Strassner et al. [2009b] proposes a technique to use context-aware policies and ontologies to facilitate business-aware network management. Ontology-based approaches are also used for policy specification [Nejdl et al., 2005], anomaly analysis [Hu et al., 2011] and refinement [Guerrero et al., 2006] and translation. However, little work has been done in the field of semantic-driven policy processing in terms of generating events and selecting related policies based on semantics of network flow data. The articles [Kodeswaran et al., 2011, 2007] propose a packet level semantic tagging framework that enables intermediary routers in the network to reason over semantic tags to retrieve related policies but the framework is designed for the refined semantics of packet level data. We want to use a similar approach for semantic information retrieval

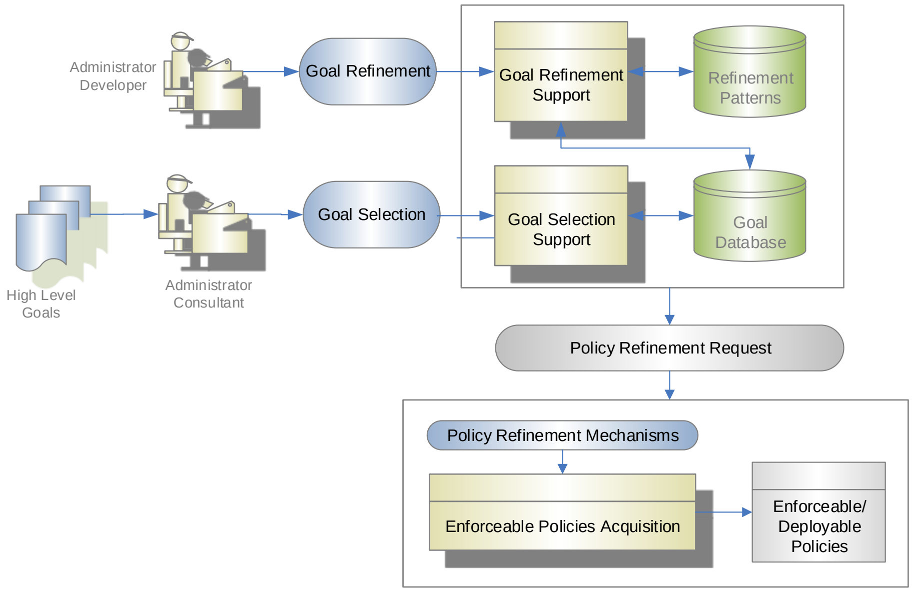


Figure 2.8: Goal-based approach for policy refinement using the domain experts' knowledge to select and refine the goals using the expert-defined support and refinement models.

that is described in articles [Fernandez et al., 2011, Tran et al., 2007] but we require a more sophisticated approach to search the semantics and integrate operational parameters to address the problems at hand: the scoping of search and complex hierarchy of information retrieval.

Many policy translation techniques and approaches have been proposed and developed by researchers. However most of the techniques focus on the syntactical translation without explicit consideration of policy semantics [Kaviani et al., 2007]. Some of the techniques that use semantic models for policy translation are abstractly defined. Most of research work done in this area can be divided into several categories; most prominent are: template-based, classification-based, ontology-based, model-based and goal-based translation. The work presented by Trastour et al. [2009] is domain specific–it attempts to explain an automated planning-based approach to manage the change requests in Information and Technology systems. Klie et al. [2007] and Davy et al. [2006b] present model-based automated policy translation techniques for service composition and conflict analysis, respectively. The POWER prototype [Mont et al., 2000] is a template-based technique that requires significant interaction and system knowledge from a human operator to set up the refinement templates. The goal-based techniques for policy translation [Bandara et al., 2004, Rubio-Loyola et al., 2006] use event calculus and temporal logic in conjunction with abductive reasoning techniques to derive sequence of operations to achieve a desired goal. The goal-based approach, depicted in Figure 2.8, is extremely manual and requires domain expert knowledge. The classification-based policy translation approach also requires system administrators or experts to use domain knowledge to implicitly map bounds of lower-level metrics such that the high-level performance goals are met. The multi-layer policy translation approach [Porto de Albuquerque et al., 2005] addresses the network security domain. The model driven approach for the policy refinement is used for policy conflict analysis technique defined by Davy et al. [2006b]; the proposed model mainly focused on conflict analysis and prevention techniques. The automated policy decomposition [Su et al., 2005] defined a technique for resource management in distributed systems. The expertise knowledge-based policy translation approach [Rochaeli and Eckert, 2007] uses automated work flow for policy translation process supported by the domain expert knowledge. The ontology-based policy refinement model [Guerrero et al., 2006] uses a semantic manager that works with system ontologies and policies defined at different abstraction levels for policy refinement.

2.2.3 User Driven Semantic Conflict Resolution in Inference Rule

By taking home user inputs (as governing rules) in the *HAN* control loop may increase viability of a system in practical manner but it also increases the chances of imprecise knowledge flow if rules are logically inaccurate and can lead erroneous system behaviour if not handled properly. Even if user defined rules are logically sound, there can be situations where two independent rules may end up in a conflict because of run-time system parameters. Such situations can also occur if control system relies on the input of faulty devices or systems. Under these circumstances, conflicts in decision control systems are inevitable. For rule based management within a *HAN*, Chen et al. [2008] propose an event calculus-based logical framework for behaviour reasoning leading to personalised just-in-time behaviour assistance within a smart home. The concepts of a compound action and its hierarchical construction mechanism enable the managing systems to incorporate activities of daily living heuristics and users profiles and hence achieve a degree of personalised assistance. The approach avoids the assumptions of users rationality and the time-consuming planning processes of traditional approaches.

For rule processing and conflict resolution, Mei and Paslaru [2005] present a comparison of rule engines Jess reasoner [Friedman-Hill, 2003, Laboratories, 2009] and Seseme to solve the problem of undecidability of SWRL rules; the authors found that SWRL in Sesame is more flexible compared to Jess, however, SWRL in Jess addresses the implementation of OWL semantics directly. Calero et al. [2011] show the necessity of a new expressiveness extension to SWRL language for non-monotonic reasoning. Such an extension is aimed to define rules, which could contain a Not-Exists quantifier which enables to ask about the non-existence of facts in the knowledge base or remove knowledge from it. Same authors presented a common taxonomy of semantic conflicts in ontology in an earlier article by Calero et al. [2010]. From the list of semantic conflicts, the article pays special attention to four types that can be considered as more usual when dealing with advanced information systems: conflict of interests, self-management conflict, conflict of duties, and multiple-managers conflict. Then, five different strategies for conflict detection are presented to the reader and exemplified using a realistic conflict of interests scenario. Hicks [2007] propose the "No Inference Engine Theory" (NIET) for rule-based systems designed for conflict resolution in the operation of an inference engine. The author described the NIET application as the resulting development environment for performing conflict resolution while eliminating the inference engine and using propositional logic; however, this approach is not feasible in many traditional rule-based reasoning systems. [Yuan et al., 2012] propose to use a secondary inference engine in the presence of the primary, which finds and solves the conflict using priority and matching degree criteria. However, this solution is not practically feasible; moreover, the proposed solution does not solve the issue of conflicting rules of equal importance. Hantry et al. [2011] propose a temporal logic based conflict solver that uses unit rule propagation method combining watcher and classical conflict learning techniques. Linear temporal logic is used with disjunctive and negation operator, however, SWRL only supports conjunctive operators. Another powerful conflict resolution strategy is put forward by Bikakis et al. [2011] that uses the simplest form of defeasible logic using preference ranks of inference rules. We work on a similar idea but in our technique, the preference ranks are dynamically set and they may change with time for every rule.

For uncertain inference rule processing, Nickles and Sottara [2009] present a short survey of prominent approaches to rule frameworks and formal rule languages for the representation of and reasoning with various kinds of uncertain, imprecise or ambiguous information. The survey mainly focuses on probabilistic approaches. Heckerman and Shortliffe [1992] argue that the belief-network (probabilistic models) representation has overcome many of the limitations of the certainty factor model, and provides a promising approach to the practical construction of expert systems. Peng et al. [2010] presents work on Bayesian network belief update with uncertain evidences. They define two types of uncertain evidences: the virtual evidence and soft evidence. The virtual given as a likelihood ratio, represents uncertainty one has for an observation. The soft evidence, given as a distribution over one or more variables, represents the uncertainty of an event one is observing and it requires the distribution be preserved in the updated Bayesian Network. Wlodarczyk et al. [2010] presents SWRL-F as an extension to SWRL that allows constructing fuzzy rules using lexical variables described it OWL-based ontology. It provides a general design that is based on fuzzy control system approach and together with proper construction of SWRL-F ontology that allows to avoid conflicts between Fuzzy logic and Description Logic in the ontology. However, the SWRL-F API is not yet available for experiments. The relevant work done in the domain of uncertainty measurement (entropy) of inference rules are presented by Wise [1986], Wise and Henrion [1985], Wise et al. [1987] from mid to late 80‘s. In these articles, the author proposed a framework for comparing uncertain inference systems (UIS) to probability, and presented his results based on the evaluation of only one uncertain rule. Another related but quite recent work published by Wasserkrug et al. [2012] that proposes a probabilistic technique to resolve uncertain inference rule. The article covers an abstract description of technique, and event and rule models.

2.3 Summary

In this chapter, we have provided a brief but significant background to Home Area Network, Network Management concepts, Policy-based Network Management and Semantic technologies. We also explained state of the art related to the research challenges that we addressed in this thesis. Social and technical advances in communication technology and the rapid growth of the web have strained modern-day *HANs* [Edwards et al., 2011]. Due to ongoing change in our social behaviour towards technology and technical advancement, which is now very much part of our daily life and homes, have made our work more challenging. The increasing complexity of underlying network infrastructure has not only put *HAN* users in a challenging position but has also raised many vulnerability risks [Poole et al., 2009a]. There are some mediums available to monitor and control networks, e.g., WireShark[1]; however, the expertise level required to use these puts them beyond the access of an ordinary *HAN* user. In short, the diversity of operational scenarios, time and cost constraints, and technical complexity are making *HAN* management more difficult than ever. The literature review helped us formulating following assumptions and designing our research methodology:

1. Most of the proposed approaches for autonomic *HAN* management lack substantial non-expert user involvement in their proposed solutions;

2. There exists no formal processes or standards on how to get non-expert users involved in *HAN* management process.

An optimal solution so far is to use policies to automate *HAN* management tasks. Monitoring and control are key components of any autonomic system [Strassner, 2009]. Ideally, a monitoring process should be context-aware to understand the dynamics and semantic technologies [W3C, 2004] can help to leverage the context by highlighting valuable information about network events. By and large, available monitoring tools and techniques provide information of limited value [Scheirer and Chuah, 2008] to an ordinary *HAN* user. In this thesis, we aim to develop a framework and techniques for user-centric *HAN* management using semantic-driven policies. The research methodology is empirical and rigorous. A repetitive and incremental approach is adapted for experimentation. We started our research by surveying *HAN* management issues and experimenting with the use of policies in a test-bed. A preliminary framework is proposed and initial results led us to work on development of techniques for policy translation using semantic model. We expanded and refined the framework by involving *HAN* user in *HAN* management control loop [Kielthy et al., 2010] and developed techniques for the semantic uplifting and enrichment. Lastly, we developed techniques to build *HAN* user driven robust *HAN* management decision system.

[1] http://www.wireshark.org/

Chapter 3

HANmanager Framework

In this chapter we provide a detailed description of our proposed home area network (*HAN*) management framework, which we call *HANmanager*. The framework uses semantic technologies to construct a dynamic model around the *HAN* domain to implement users' preferences for how devices and services are managed and accessed. This chapter gives an insight into all of the main components that constitute the framework and also describes other components that are minimally developed but are significant for proper functioning of framework. The components are described in a chronological manner based on the order of their operations.

This chapter is structured as follows: §3.1 outlines our *HAN* domain model; §3.2 presents an overview of *HANmanager*; §3.3 describes the components of the framework; In §3.4 and §3.5, we propose a deployment architecture and describe the employed test-bed for the implementation of framework; and §3.6 presents the analysis, future work related to the framework and finally, makes a conclusion for this chapter.

3.1 HAN Domain Ontology

Ontology is most prevalent semantic representation technique as it provides (a) vocabulary, (b) taxonomy, and (c) a complex semantic network. An ontology development methodology comprises a set of established principles, processes, practices, methods, and activities used to design, construct, evaluate, and deploy ontologies. Several such methodologies have been reported in the literature (as discussed in Chapter 2). One

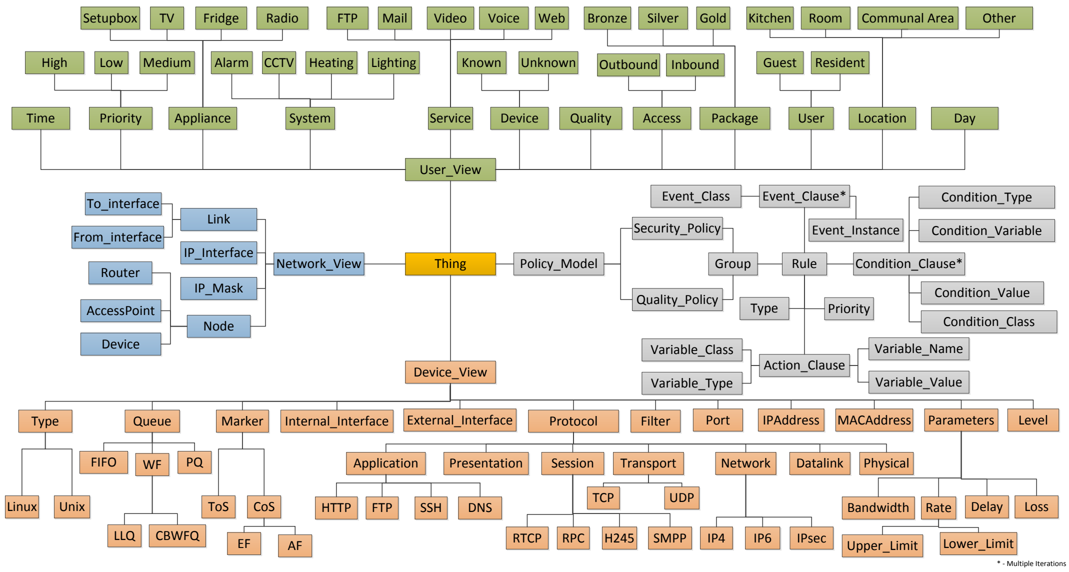


Figure 3.1: HAN Domain Entity Graph - showing entities related to "user-view" (green), "network-view" (blue), "device-view" (orange) and policy model(lilac).

example is the simple methodology proposed by Noy and McGuinness [2001] to start development by defining the domain scope and class hierarchy. Others advise specific ontology development processes such as the one proposed by van der Vet and Mars [1998] for bottom-up construction of ontologies. Model Driven Development (MDD) is being constructed in parallel with the Semantic Web [Selic et al., 2006]. The MDD approach to software development suggests that one should first develop a model of the system under study, which is then transformed into the real domain. The most important research initiative in this area is the Model Driven Architecture (MDA) [Kleppe et al., 2003], which is being developed under the umbrella of the Object Management Group (OMG)[1].

We used basic MDA elements to construct our *HAN* domain ontology and *HAN* management framework. OWL-DL is employed to construct a domain model of a typical (*HAN*). Our *HAN* domain model assigns enriched semantics to the entities belonging to different sub-domains; it also defines that how the entities in a sub-domain of *HAN* (e.g., service, ports, protocols, devices) are related to other sub-domain entities in *HAN* (e.g., applications, users). Different entities/concepts involved in *HAN* are shown in

[1] `http://www.omg.org/`

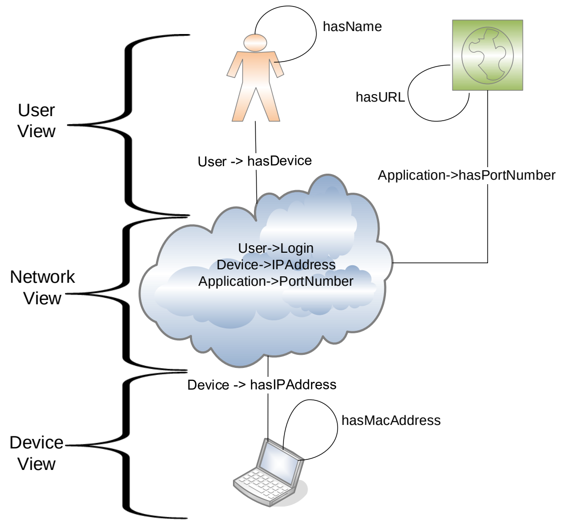


Figure 3.2: Relationship among different but related entities of "user view", "network view" and "device view".

Figure 3.1. The links among the real world and *HAN* system entities/concepts are discovered at ontology design time by determining how concepts within the domains or sub-domains coexist and are related to each other. The domain model also provides an articulation of the meaning behind entities with the help of other entities in different functional layers of *HAN* domain (e.g., an IP address of a device connected the *HAN* can be related to a user in real world hence the IP address is represents a proxy for the user). Therefore, a domain model acts as a formalised context behind the nature of the entities in different layers/sub-domains, which otherwise may stay unconnected. Model-driven architecture is most optimal choice that focuses on the use of models as the central artefact in the development of any complex system, providing a hierarchy of models with the purpose of separating the entities.

Therefore to deal with the complexity of the *HAN* domain, it is divided into three sub-domains/layers: "user view", "network view" and "device view" (as shown in Figure 3.2). The "user view" has the entities that are closely related to the real world, which are abstract in nature with respect to the network system and exist at the higher level of *HAN* domain. Precisely, these are the entities that are more closer and meaningful to the *HAN* users. The "network view" and "device view" have (operational) entities that are related to the network systems and gateway device respectively, which are concrete (less abstract) in nature with respect to *HAN* system and exist at lower levels

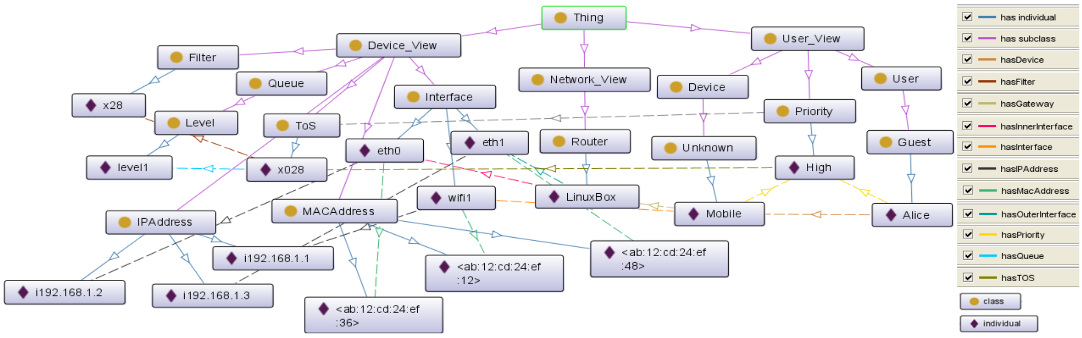


Figure 3.3: HAN Domain Ontology Concepts Graph: showing the interconnections of entities across the three sub-domains of *HAN*.: "user view", "network view", and "device view". This ontology concept graph instance is based on a scenario that assigns highest priority to the traffic generated by a guest *HAN* user.

of *HAN*. An ontological approach within the context of the *HAN* domain can provide the necessary depth in developing and consorting these three sub-domain models within the *HAN*.

The *HAN* domain ontology establishes a conceptual model that describes the entities, their attributes, properties and relationships, and the constraints that govern the integrity of entities. The domain ontology contains the relationships among all different entities including the constraints within the scope of the *HAN* domain. The ontology-based structuring and abstraction help to maintain the complexity and integrity of the *HAN* domain model. Layering the *HAN* entities also simplifies the design, eases scoping the search and gives more visibility to inter-domain relationships. The scope of search can limit itself to select specific views of the domain model to improve efficiency (e.g., selection of "user view" entities for user policy specification). The associative semantics of the ontological entities are defined in terms of associations with other entities, thus these associative semantics are closely connotative semantics [Leech, 1974]. The "user view" has the entities that closely mirror users' perception of the structure and operation of the *HAN*. The "network view" and "device view" have (operational) entities that are related to the network system and gateway device respectively (as shown in Figure 3.3).

Using these ontological concepts, the network related preferences of users can also be

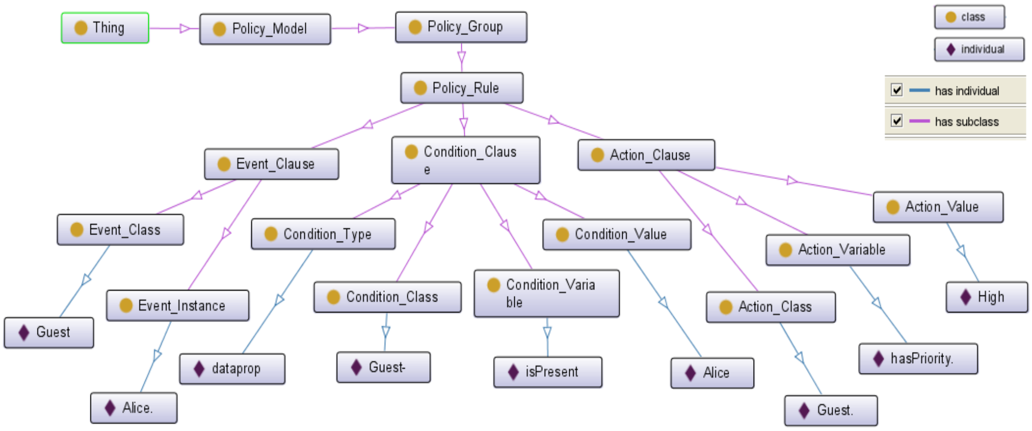


Figure 3.4: Policy Model - Showing the structure of policy framework that is used as a meta model to retrieve policy related semantics for the *HAN* domain entities. In the diagram, a user preference is saved in the policy model in the form of a declarative policy rule using the "user view" entities of *HAN* domain ontology.

saved in the ontology in the form of abstract rules. To deal with the user rules, a policy model is also incorporated in the *HAN* domain ontology; it uses the "user view" entities to specify user defined rules (as shown in Figure 3.4). For example, consider following policy rule:

```
Guest(?x) ^ isPresent(?x, true) -> hasPriority(?x, ''High'')
```

... *R0*

This user rule *R0* implicitly says that all the traffic flows generated by a guest user get high priority, provided the guest is present in the home. In this rule, `Guest` is an managed event entity (that signifies that network activity related to the entity `Guest` is monitored actively by *HAN*manager); `isPresent(?x, true)` is a condition clause (where `isPresent` is a condition variable/property and `true` is a condition value); and `hasPriority(?x, ''High'')` is an action clause (where `hasPriority` is a action variable/property and `High` is a action value). When a home network user specifies its network related preferences, the user's preferences are decoded and the parsed constituents are saved in the policy model as event, condition and action as shown in Figure 3.4. The policy information model acts as a meta-model for the *HAN* domain model to classify domain entities and their properties into the event-condition-action classification. The meta-data about the *HAN* domain entities also helps the *HAN*manager components use *HAN* domain entities in different ways, e.g., user rule specification, translation of user rules to different rule formats.

3.2 *HANmanager* Overview

In this section, we describe constituent components of the *HANmanager* framework and specify the methods for realisation of management control loop as described in [Jennings et al., 2007, Kielthy et al., 2010] for *HAN*. The *HANmanager* provides a flexible mechanism to manage network resources and services without requiring the *HAN* users to understand details of their networks. It not only offers an intelligent and convenient approach to the network resources requirements but also offers a mechanism to manage requirements related to *HAN* users, e.g., giving high priority or access to certain *HAN* user, which makes *HANmanager* unique from other available approaches that significantly lack human-centric network management features. Along with other benefits of the framework, e.g., increase in quality of service, efficiency, adaptability, coherent network behaviour, and flexibility, it mainly imparts ease of use to perform different network management operations (performance, configuration, security and quality of service) in order to achieve *HAN* users requirements. It simplifies the management tasks for ordinary *HAN* users by abstracting network system configurations with the help of semantic models [Antoniou and van Harmelen, 2003] and helps in managing network behaviour as per the requirements in the form of policies [R. Yavatkar, 1999].

Recalling the *HAN* management challenges and requirements from Chapter 1 (§ 1.1), the *HANmanager* offers a promising solution for managing *HAN* by taking into account of *HAN* management issues described in Chapter 2. With the help of semantic models, the *HANmanager* can give more visibility to the network operations and understanding to the network monitoring data, which could highlight troubling issues, for example blocking an unknown device or malicious application that is trying accessing the network.

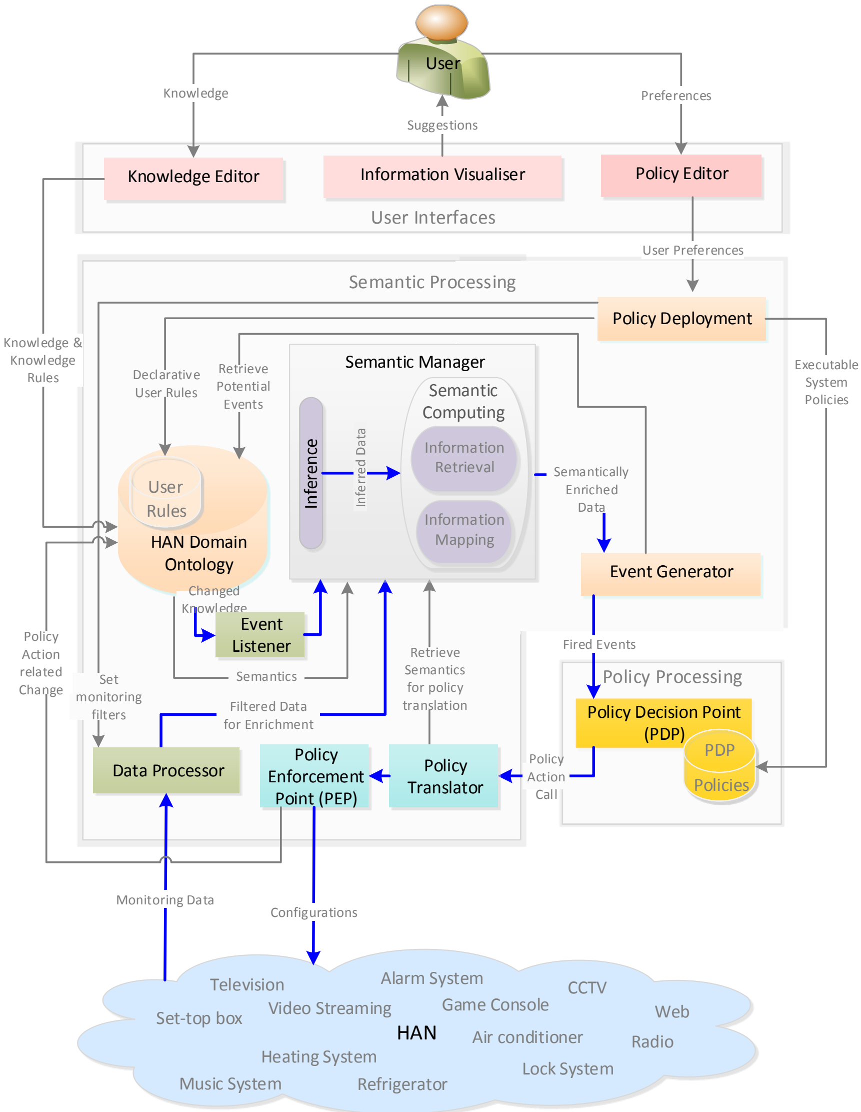


Figure 3.5: User-Centric, Policy-based *HAN* Management System: a semi-automated approach to manage and control home devices, applications and systems through system policies that are translated to network configurations. The figure shows the main components of the *HANmanager*, depicting the control loop with blue arrows. With the help of appropriate interfaces, a *HAN* user can define system rules; devices and services are monitored and significant data is collected from them; collected data is further processed and enriched semantically; and based on the semantics of data, related policy rules are applied to the system to control *HAN* system. The orange coloured subunits of framework are the plug-and-play third-party components.

Via intuitive user-centric management interfaces (visual, voice or graphical), *HAN* users can supply high-level information and requirements related to *HAN*, for example the names of users and devices, whether guests are present in the home, does a guest get access to internet, what application gets more bandwidth and what rooms certain applications can be accessed in and by whom.

The network flows data is semantically enriched by analysing its semantics and dynamically linking to instances of user importance (relevant to user preferences) in the domain ontology (that represents *HAN* domain model). The inferred linkages are determined using inference rules contained in the ontology, which take into consideration the contextual information and may also produce new knowledge. The inferred knowledge helps the *HANmanager* triggering system policies (semantic interpretations of user preferences in the form of *HANmanager* rules). To trigger system policies, related policy events are fired and execution process is initiated at a Policy Decision Point, causing reconfiguration of networking devices, applications and services in order to ensure that users preferences are consistently met. Flexibility to adapt to changing requirements is one of the most significant features of *HANmanager* as it can be modified by:

- the home network user, by reflecting new concepts in the "user view";
- the network equipment by introducing new features, or relationships to "user view" concepts; and
- the terminal devices by bringing new terms of raw data for inference.

With in the *HANmanager* system, user requirements can be expressed using formal rule-based expressions [Bonatti et al., 2009] supported by many policy languages [Tonti et al., 2003]. Since user requirements can be abstracted from the *HANmanager* perspective, a declarative policy [Hinrichs et al., 2009] is most appropriate choice to express user requirements in a semi formal format. If user requirements are specified using a declarative policy language, the policies are mostly translated to an executable form to enforce them on the network. The semantic models can assist the translation process of policies as discussed by Bonatti et al. [2009].

Figure 3.5 shows the different functional components of *HANmanager*. The *semantic manager* is core component that is the prime focus of the research work presented in this thesis. The *semantic manager* supplies semantics to different network data flows and ontology inferred data, and establishes connections among different network data elements to provide more visibility to different interrelated parts of the *HAN* system. The semantic information can be utilised in lots of different ways, for example, making the network base-view data more understandable to *HAN* users by connecting user related data to network related data and vice a verse. Other components of *HAN-manager* are either partially developed or taken off-the-shelf. Essentially, the *semantic manager* is required to provide an interpretation to different network data flows to capture significant network events that are related to *HAN* users requirements and assist the *HANmanager* for enforcing those requirements on the network. For example, the *HANmanager* may infer that an unknown device attempting to send traffic through the gateway router belongs to a guest in the home that *HAN* user has indicated is visiting during a weekend and so may automatically configure the router to allow traffic from that device. Thus, the *HANmanager* balances the trade off between fine-grained management versus the benefits of providing users with an intuitive and abstracted view of the network management process.

3.3 Framework Components

This section describes different components of the framework architecture. In Figure 3.5, we have highlighted some the components that are main contributions of this thesis. However, other components are equally important but they are minimally developed and defined to explain their role with in the scope of the proposed framework. In the current implementation of our framework, we attempted to define and develop *information visualiser*, *policy editor*, *knowledge editor*, *event listener*, *data processor*, *semantic manager* and *policy translator*. However, the primary and contributory component of the framework is the *semantic manager*, which is comprehensively defined and developed. In this chapter, we also explain *policy selector* that is developed in the previous implementation of our framework but it is replaced by a third party policy system in the final version of the framework.

3.3.1 Information Visualiser

Monitoring and controlling are key components of any autonomic system [Strassner, 2009]. Analysis of monitoring data should highlight valuable information about network events that are significant to *HAN* users. Typically, available monitoring tools and techniques use syntax-based data analysis techniques that provide information of limited value [Scheirer and Chuah, 2008] to an ordinary *HAN* user. In contrast, semantic-based analysis not only makes data interpretation more meaningful but can also help in managing different network events. The *information visualiser* provides a basic mechanism to visualise semantically uplifted high-level monitoring information and also suggests points of improvement in *HAN* system to bring additional value to the end user.

The first step is meaningful interpretation of network flows and then management of network by using useful bits of information extracted from monitored data packets. The *information visualiser* also enables the *HAN* users to select a monitoring view under a certain semantic attribute combining the semantically uplifted information and related low level information gathered from the raw monitoring data, e.g., internet access timings for a particular user. The information visualiser can also work as an alert based system for the *HAN* users.

3.3.2 Policy Editor

Home users need to be able to configure and dimension their networking equipment and devices to ensure that services operate as expected. There are many approaches for interfacing the *HAN* users with *HAN* system and many advanced user interfaces (voice-based, zero-input, natural language-based, touch screen) are trying to have a breakthrough [Schaffer and Minge, 2012]. However, human-centric interfaces provide additional value any such interfaces. The *policy editor* is an intuitive and interactive web-based tool used for specifying user preferences related to the behaviour of home network. The user preferences are high-level requirements to manage and control home networks. From the network point of view, these user preferences are abstract declarative policies that are related to "user-view" entities in the *HAN* domain model. The "user-view" entities abstracts the network system and lessens the underlying complexity of managing the *HAN* system for *HAN* users. Users can define their network related requirements via policy editor that might be highlighted by the *information visualiser*, without requiring to understand the *HAN* system completely.

Policies impose behavioural restrictions on properties of the *HAN* system. For an ontological description of policies these restrictions have to be expressed in an underlying knowledge representation of domain. This way stakeholders of *HAN* system can describe their requirements with respect to a common ontology in terms of meaningful concepts and relations without requiring to know how *HAN* system works. By referring to common domain ontology, the usage of an agreed terminology is assured. One substantive way of representing these user-defined policies is to define them in a rule based expression such as Event-Condition-Action (ECA) format[1]. The ECA format is an instinctive way of specifying declarative policies. The *policy editor* can use "user-view" entities as potential network event related entities and their properties for the specification of potential conditions and required actions. For example, consider the following policy rule:

if $User.hasName == \text{“}Ben''$ **then**
  $User.hasPriority = High$
end if

This policy rule says that all the network flows generated by a user, whose name is "Ben", get high priority (over the other network flows generated by other users). In this rule, $User$ is a an event entity (that means any network flow related to entity $User$ is monitored actively by the system at the high level, also keep in mind that $User$ entity is not cognized yet by the system at the low level), `hasName==‘‘Ben’’` is a condition clause (where `hasName` is condition variable/property and *Ben* is condition value) and `hasPriority = High` is an action clause (where `hasPriority` is action variable/property and *High* is action value). The specified rules can be translated into any other ECA format supporting policy language using a standard policy information model as an interlingua.

The *policy editor* provides an intuitive and interactive medium used for specifying user preferences related to the behaviour of the home network. The user preferences are high-level network rules that specify how to manage network users, devices and service. Thus, the user can specify "what needs to be done" by the *HAN* system without requiring them

[1] `http://ruleml.org/reaction/eca/`

to understand and specify "how it needs to be done". In the *HANmanager*, these high-level rules are added to the *HAN* domain ontology as abstract declarative rules that are based on "user-view" entities. A semantics-driven *information visualiser* can also suggest some improvements into the *HAN* system after further analysis of semantically enriched inferred data,however, it is out of the scope of this thesis.

3.3.3 Policy Deployment

The *policy deployment* component has three main roles in the *HAN*manager: (i) deployment of the user policy rules; (ii) translation of user policy rules (that are declarative) for the *policy system* (also referred as PDP - Policy Decision Point — see §3.3.10 for more details) in the form of system policy rules (that are executable); and(iii) configuration of the *data processor* (see §3.3.6) to capture only managed events that are related to user preferences. When a user specifies the preferences using the *policy editor* (see §3.3.2), the *policy deployment* decodes them using the policy model in *HAN* domain ontology and generates an equivalent event-condition-action rule representation. With the help of the *semantic manager*(see §3.3.7), the semantics of abstract managed events (that retrieved from the user policy rules) are further refined to a form, which is intelligible to the *HAN*manager. To process user policy rules, a third-party *policy system* is used and the user policy rules are translated using the ontology to a *policy system* (see §3.3.10) specific format. The translated policies are called as system policies that are deployed in the *policy system.* This component is minimally developed as part of the *HANmanager.*

3.3.4 Knowledge Editor

Ideally, an autonomic *HAN* system should be self-learning but there are certain types of knowledge, where user input is indispensable. The *knowledge editor* is a user-centric management interface through which a home user can supply high-level information about the *HAN* system, for example the names of users and devices, whether guests are present in the home, and what rooms certain services can be accessed in and by whom. The *knowledge editor* is very similar to the *policy editor*; however, the purpose of its use is different. With the help of the *knowledge editor*, *HAN* user can add new knowledge about the "user view" entities or instances. The nature of knowledge can be simple addition or change in characteristic, behavioural or relational property of an entity or of its instance. We can generally classify knowledge rules into three categories:

1. reclassification knowledge;
2. characteristic knowledge;
3. behavioural knowledge.

The specified knowledge is either specific about an instance of an entity or can be general relating to an entity (meaning for all of its instances) but we deal with only concrete knowledge (containing information about instances that is made available after reasoning over the knowledge rules). The general form of knowledge further goes through processing/inference and new knowledge of specific nature is produced with the help of reasoners. The specified knowledge rules are saved in the *HAN* domain model and once they are reasoned over, the resulting changes are monitored closely to capture information related to managed events.

3.3.5 Event Listener

In addition to low-level data monitoring, *HANmanager* also needs to monitor the *HAN* domain model for high-level monitoring data. Any change in the *HAN* domain model is considered potentially a network event. The *event listener* takes all the changes in the *HAN* domain model after the reasoning over the knowledge rules and each piece of information is analysed according to its category. We observe that broadly there can be three types of changes in the *HAN* domain model: reclassification, characteristic or behavioural change. Reclassification is a change that reclassifies an entity under different entity group, e.g., putting a device under unknown category. The characteristic change is related to the modification of a data property of an entity, e.g., changing current location of user. The behavioural change is related alteration in the object property of an entity, e.g., change of ownership of a device to another user. For any change, the target entity and its value are cleaved for further analysis. All of the three types of *HAN* model changes can result in different sort of inferred knowledge. In case of reclassification change, the name of reclassified entity is to be examined. For characteristic or behavioural knowledge change, the name of property/variable that has been changed is to be taken out along with its value for analysis and processing. The retrieved information is imparted to the *semantic manager* to get the semantics of inferred information. This component is minimally developed as part of the *HANmanager*.

3.3.6 Data Processor

The *data processor* retrieves monitoring data from the gateway device (i.e. router). The retrieved information comprises low-level details of *HAN* activity. The *data processor* parses the monitoring data and gets the key-value pairs of filtered data (*FDEs*) that characterises a potential network event, e.g., a device or user accessing the Internet; however, filtered data represents only low-level information. The filtered monitoring data is ported to the *semantic manager* (see §3.3.7) for semantic enrichment. Many different approaches are proposed to deal with diverse data formats from different data sources [Sakka et al., 2012, Sventek et al., 2011] but this issue is out of the scope of this thesis. In our framework, we assumed that context of monitoring data is known but the semantics of monitoring data are unknown. That means that *data processor* knows what is being monitored and what is the source of data. One way to achieve it is through knowing what needs to be monitored from user-defined policies especially by analysing the abstract events in the user-defined policies. The semantics of monitored data are retrieved through *semantic manager*.

3.3.7 Semantic Manager

The *semantic manager* is core to the *HANmanager*. It enriches the semantics inferred data by mapping it to the ontological concepts in the *HAN* domain. Initial mappings of data to the ontology provides primitive semantics, which are further extended for detailed and enriched semantics. To fetch the primitive semantics of network flow data, the *semantic manager* uses the *HAN* domain ontology. A network flow can be monitoring data collected from gateway or inferred data collected from *HAN* domain ontology after the reasoning.

For the inferred data, the *semantic manager* monitors the *HAN* domain ontology changes retrieved after the reasoning. Any change in the *HAN* domain ontology is considered potentially a network event. It processes all the changes in the *HAN* domain ontology after the reasoning over the knowledge rules. As discussed earlier in $3.3.4, there can be three types of changes in the *HAN* domain ontology: reclassification change, characteristic change or behavioural change. Each type of change can be a potential managed event. The *semantic manager* enriches the semantics of the potential managed event by retrieving related concepts and creates a semantic graph. The entities in the *HAN* domain ontology can be linked to each other in many different ways (syntactically, morphologically or semantically), however, we only focus on semantic relationships. The *semantic manager* finds the leads of inferred data in the *HAN* domain ontology using a lexical semantic search technique and once the primitive semantics are found, the semantics are enriched to the next level depending on the depth of ontology graph. A semantic graph of interlinked entities is created by exploiting the relational properties of entities with the other entities in the *HAN* domain ontology.

The semantic enrichment process works in recursive manner to identify transitive entity relationships (indirect entity relationships). Using the values of entities in the network flow data, the semantic graph is also instantiated in recursive manner. The instantiation of semantic graph is a process of retrieving instance related information of the entities. We only support one-to-one entity-instance relationships.

For the low level monitoring data, the semantics of filtered data elements are uplifted by mapping the data to the ontology. Once the mapping is found, a semantic graph is created and instantiated. The instantiated semantic graph is later used for different purports (semantic-aware monitoring and monitoring data visualization, policy processing and translation).

3.3.8 Policy Selector

This is an alternative component to policy system in our framework. Instead of using *policy selector*, we can replace it with a third party policy system, which is explained in the later section. This component is an attempt to built a prototype of policy system using a rule-based language. As we discussed in *policy editor* and *knowledge editor*, a rule-based language is used to specify policy and inference rules. A policy rule represents a user requirement in the system and an inference rule adds new knowledge to facilitate the decision making process. When the domain model is updated and inferred using the values of inferred and filtered monitoring data, the related policy and inference rules in the domain model should also be selected for processing. The *policy selector* uses the instantiated semantic graph (that is created from the monitoring data) and selects the related policies by exploiting the ECA (event, condition and action) semantic rudiments. The policy rules are selected for translation if their respective events, conditions and actions are analysed and verified successfully. This component is minimally developed as part of the *HANmanager*.

3.3.9 Event Generator

Once the semantics of inferred data are fetched in the form of an instantiated semantic graph, the *event generator*(see §3.3.9) iterates over the list of abstract managed events, also retrieved from *HAN* domain ontology, and searches for event related information in the semantic graph. If relevant information is available, e.g., a guest is present or a certain user is trying to access Internet, the associated *policy system* event is fired so that the *policy system* (see §3.3.10) can process related system policies. The *HAN*manager uses a third-party policy system (see §3.3.10) for the evaluation of system policies. The *event generator* fires the event by porting the event related information to the *policy system*. Thus, the fired events are managed by the *policy system* for further processing of system policies.

3.3.10 Policy System

The *semantic manager* (see §3.3.7) interfaces with the *policy system* through the *event generator* (see §3.3.9). When an event is fired by the *event generator*, the Policy Decision Point (*policy system*) matches it with the list of events of active system policies (saved in the PDP policy repository). If a related system policy exists and meets the criteria, then the selected policy is evaluated. The system policy is executed by the *policy system* after successful evaluation. The execution of system policy triggers its translation to system configurations using the semantic graph. The *policy enforcement point* enforces the system policy and apply the system configuration on target devices.

3.3.11 Policy Translator

The *policy translator* translates a user policy rule to a device level policy. It requires four knowledge constituents:

- domain knowledge of *HAN* entities;
- syntactic knowledge of source and target policy languages;
- semantic knowledge of source and target policy languages;
- pragmatic knowledge of source and target policy languages in relation with *HAN* entities.

These knowledge constituents can be specified in the *HAN* domain model. Using the *HAN* domain model, the *policy translator* can translate user policies and generate device policies by exploiting relational properties among different knowledge constituents.

The aim of *policy translator* is to translate user-level policies in the *HAN* domain to the device-level policies that can be enforced on router. The user-level policies are defined by the *HAN* users and those polices are translated to device-level policies by the *policy translator*. We assume that a device-level policy can affect *HAN* devices and services in any numbers. The execution of a device-level policy is associated with the occurrence of particular network event that stipulates the execution of the network-level policy (also referred as system policy). The information about affected devices or services may not be available readily from the inferred or filtered monitoring data but we can determine them from the semantic graph generated from the *HAN* domain using monitoring data. The a higher level semantic map may only help in selection of declarative policies that are required to be enforced but the translation process requires deeper level of information from the "device view" in the domain. For the translation of user-level policies to device-level policies, we need low-level information about the device specification, network topology, target policy language, and also the applications running on devices that assist in enforcing the configurations. Therefore we require an extended semantic graph from "device view" in the *HAN* domain.

3.4 Framework Deployment Model

This section describes the framework architecture from the deployment view point. The framework components are divided in five layered architecture from processing perspective: User Interface Layer (UIL), Semantic Retrieval Layer (SRL), Policy Processing Layer (PPL), Data Monitoring Layer (DML) and Device Interface Layer (DIL). From deployment view point, policy management architecture [Yavatkar, 2000] proposed by IETF Policy Framework Working Group is vitally significant. The architecture consists of a policy console (PC), Dedicated Policy Repository (DPR), Policy Decision Point (PDP), Policy Enforcement Point (PEP) and policy communication protocols.

UIL is deployed on Policy Specification Point (PSP), SRL and PPL are deployed on Policy Decision Point (PDP), and DML and DIL are deployed on Policy Enforcement Point (PEP). The PDP decides authorization decisions over the policies where PEP can communicate with PDP and fetch decisions. Ideally, there should not be a tight coupling between the PEP and PDP but currently there is no means of defining PEP and PDP communication in a standard manner. The communication between PDP and PEP can be realised in many ways, e.g., HTTP, COPS and SNMP. In a *HAN*, the network devices are mostly cheap that they often do not provide sophisticated management interface. For our framework, we used HTTP based communication between PEP and PDP.

3.5 Test-bed Implementation

The test-bed used to implement the *HAN*manager comprised a single Ubuntu Linux router connecting a *HAN* to the Internet. Two network interface cards are used for configuring the Linux machine as a router. The *HAN* has one Ethernet client machine (a Windows XP desktop) and two wi-fi client machines (one Windows XP laptop and one HTC smart phone). As depicted in Figure 3.6, the *HAN*manager functional components are deployed across the *HAN* gateway router and a server. On the client machines, we used the Web Traffic Generator [Technologies, 2007] tool to generate background TCP web traffic; the Traffic Emulator [Kankanyan, 2009] tool to generate background UDP traffic, and XLite [Xlite, 2006] is used to make VoIP calls.

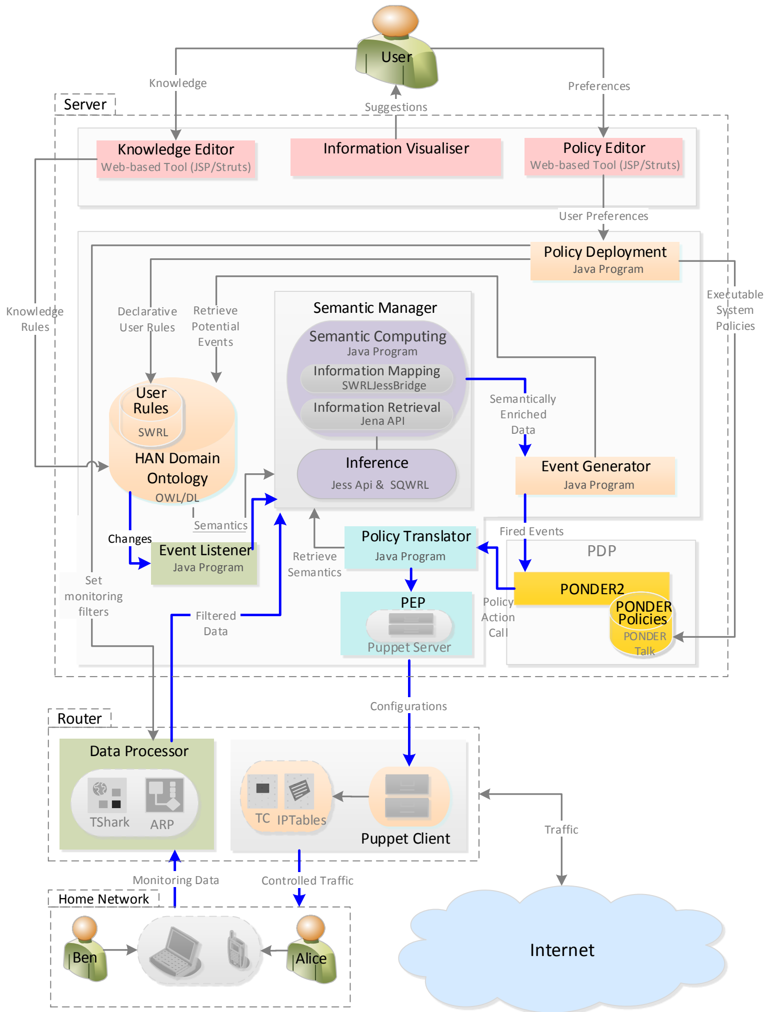


Figure 3.6: The *HANmanager* Deployment Overview: Illustrating the technology and equipment used for deploying the *HANmanager*. The *semantic manager* component adds a layer of abstraction between home network users and the *HAN* infrastructure, hiding the management complexity of *HAN* devices and services from typical home network users. The policy-driven router acts as a controller gateway between *HAN* (users, devices and applications) and Internet. The blue arrow connectors show the implementation of *HAN* control loop and the orange coloured subunits of framework are the plug-and-play third-party components.

The network traffic is monitored (as shown in Appendix D) using Perl[1] scripts (as shown

[1] `https://www.perl.org/`

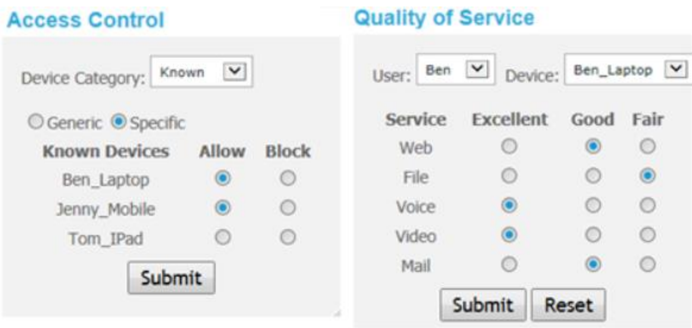


Figure 3.7: Sample of Web-based Policy Editors: Interactive tools for home network users to specify their network preferences using the "user view" entities, e.g., allowing certain "known" devices in the network to access Internet or setting the quality priority for certain applications.

in Appendix A). The server executed the GUI web editors to allow users specify the policy and knowledge rules. For the implementation of policy and knowledge editors, we developed JSP [Oracle, 2007] based web interfaces (using the STRUTS [Apache, 2006] framework). The editors (as shown in Figure 3.7) are tied in synchronously with the OWL-DL [W3C, 2004] based *HAN* domain ontology via an OWL-API [Horridge and Bechhofer, 2011] based implementation. The *HAN* domain ontology is constructed using Protégé [Gennari et al., 2003] tool and the ontology is traversed using OWL-API and JENA API [Carroll et al., 2004]. Semantic Web Rule Language (SWRL) [Horrocks, 2011] is used as a language for specifying user rules in the *HAN* domain ontology with the help of SWRLJessBridge [University, 2010]. Inference over the SWRL rules is achieved via Jess[1] reasoner. Finally, for ontology queries, we use the Semantic Query-Enhanced Web Rule Language (SQWRL) [Connor and Das, 2009].

We used the PONDER2 [Twidle and Lupu, 2007] policy system to maintain system policies (as shown in Figure 3.8). PONDER2 provides with a general-purpose policy management system with variety of policy types based on ECA rule structure [Liu, 2009]. This makes easier for us to translate user defined rules to PONDER policies. The PONDER policies are generated from the populated policy rule template in the *HAN* domain ontology with the help of simplified CIM based policy model that is

[1]`http://www.jessrules.com/`

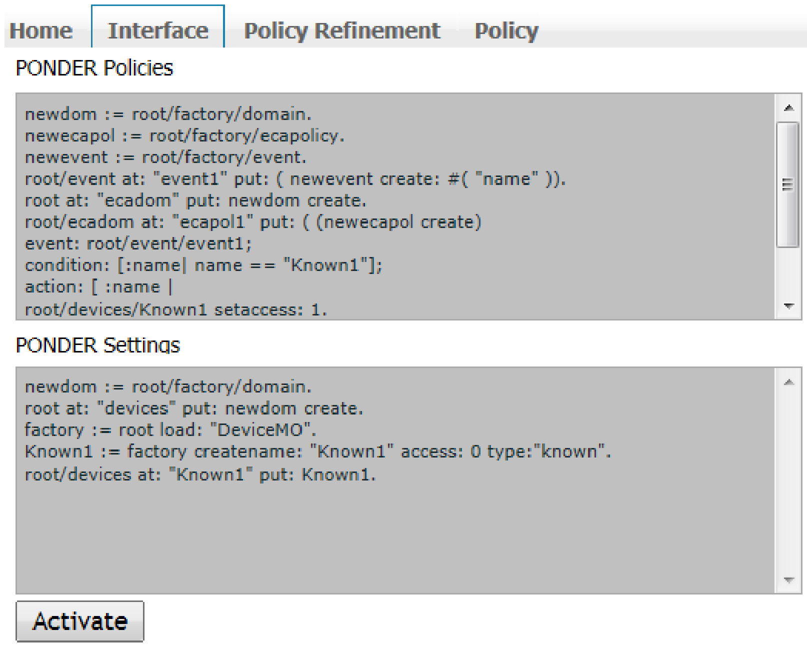


Figure 3.8: Automated generation of PONDER2 policies and settings: when a user specifies network preference, an equivalent PONDER policy is generated into the *HAN* system and set active. The PONDER policy is executed when the related event is fired by the *HANmanger*.

used as policy-semantics meta model. When the PONDER policies are triggered, the device specific configurations files are generated and enforced on the the router using the Puppet framework [Loope, 2011]. The Puppet framework realises a PEP (Policy Enforcement Point) to enforce configurations on the router. The semantic enrichment, policy processing and policy translation algorithms are implemented using Java.

On the router, we used deep packet inspection technique for network monitoring using TShark [Orebaugh et al., 2006] and ARP [Plummer, 1982] applications (as shown in Appendix D and Appendix E). IPtables [Purdy, 2009] are used for generating device specific configurations (as shown in Appendix B and Appendix C) and implementing IPv4 NAT; tc-ng[1] is used for implementing QoS traffic control (three levels quality of service based on service type); tcpdump [Fuentes and Kar, 2005] is also used for capturing monitoring data for subsequent analysis (as shown in Appendix A); Perl-based

[1] `http://tcng.sourceforge.net/`

scripts are used for monitoring the traffic queues and generating descriptive statistics; and bash shell scripts are used to manage the configuration (as shown in Appendix B).

3.6 Summary

We have presented a high level description of the *HANmanager* framework for *HAN* management. This work is inspired by the recent advances in smart homes and by the introduction of many new smart devices and applications. We discussed the main components of the framework emphasising the role of semantic models and policies. The ethos of this framework is to capture real network data and feed this up to the *HANmanager* system for intelligent home management. This offers the possibility of identifying substantial network activities and allowing the *HAN* user to manage their network without requiring them to perform complex network administrative and management tasks. The *HANmanager* monitors and controls home networks in two ways: top down and bottom up. In "bottom up" approach, *HANmanager* gathers the low-level network monitoring data collected from the router device and searches for the semantics of the data using the *HAN* domain ontology. The assumption is that low-level monitoring data usually belongs to the "device-view" in *HAN* domain ontology. Once the primitive semantics are found in "device view", the related entities in other views of domain ontology ("user view" and "network view") are discovered using the *HAN* domain ontology. This way the primitive semantics of monitored data are uplifted (abstracted) and by using this information, *HANmanager* searches related high-level user policies to control network as per defined policies. The "bottom up" approach is further discussed in Chapter 4.

In "top down" approach, *HANmanager* monitors significant changes in *HAN* domain ontology made by the user explicitly or by system itself. Only "user view" entities are exposed to the *HAN* user and users can only make changes in the "user view". Any change in the domain ontology is considered as a potential event. *HANmanager* further goes on and searches for the enriched semantics of the changed entity in other views of domain ontology ("network view" and "device view"). This way the primitive semantics of monitored data are refined and by using this information, *HANmanager* searches related high-level user policies to control network as per defined policies. The "top down" approach is further discussed in Chapter 5.

One of the limitations of proposed framework is the *HAN* domain model itself. Due to the diversity of network related concepts and variety of *HAN* layouts, a standard domain model can not be achieved. Though the domain model is capable of enhancements and systematic growth but it has to be in place at the design time of the *HAN* system. Moreover, the proposed framework does not support any self-learning features at the moment, which makes it dependent on *HAN* users or domain modellers for the information feed. Another challenge is lack of sophisticated device management interfaces. Many of the *HAN* devices are inexpensive and they are usually available with minimal management features. Initial proposal was to control individual devices and the services, however, our *HANmanager* implementation currently can only be used for IP-enabled network communication on a open source router, controlling only the network traffic generated by different connected devices and services, that goes through the gateway router.

Chapter 4

Semantic Uplift of Monitoring Data to Process Policies to Manage Home Area Networks

Tools and processes for management of network infrastructure typically assume that network administrators are technically literate, with a willingness to devote significant time to ensuring that devices are correctly configured to behave as desired. Whilst this is largely the case for service provider and enterprise networks, it is typically not the case for home area network users. The growing complexity and heterogeneity of *HANs* mean that management and configuration tasks are becoming increasingly complex. In this chapter, we present an approach for the semantic uplift of monitoring data from a *HAN* gateway router and the selection of appropriate policies to drive the (re-)configuration of the Internet gateway to provide desired behaviour in response to monitored and managed network events. We outline the use of an ontology-based semantic model to contextualize monitoring data and select the policies to apply. The chapter entails the algorithms developed for semantic uplift and policy selection. The approach is explained and evaluated via two example scenarios that have been realised on our *HAN* test-bed.

This chapter is structured as follows: §4.1 gives a brief overview of semantic uplift of monitoring data to process policies; §4.3 presents the techniques and algorithm developed for semantic enrichment of monitoring data; §4.4 gives a description of test-bed used to conduct experiments and §4.5 explains the evaluation experiments using two test scenario; Lastly, §4.6 summarises our findings and outline areas for further work.

4.1 Introduction

Social and technical advances in communication technology and the rapid growth of the web have strained modern-day *HANs* [Edwards et al., 2011]. The increasing complexity of underlying network infrastructure has not only put *HAN* users in a challenging position but has also raised many vulnerability risks [Poole et al., 2009a]. There are some tools available to monitor and control networks, e.g. WireShark[1]. Monitoring and control are key components of autonomic *HAN* management. Ideally, a monitoring process should be context-aware to understand the dynamics of underneath system and control system should be able act appropriately according to managed events, e.g. allocating higher bandwidth to a high priority network service. Furthermore, analysis of monitoring data should highlight valuable information about network events that are significant to *HAN* users, e.g. an unknown application capturing an unfair share of the Internet bandwidth. Typically, available monitoring tools and techniques use syntax-based data analysis techniques that provide information of limited value [Scheirer and Chuah, 2008] to an ordinary *HAN* user. In this chapter, we show how this form of analysis can be usefully augmented by the use of a *HAN* domain ontology that allows the *uplift* of monitoring data into a form that is understandable to *HANmanager* in terms of the impact of monitored events on users and the devices and services they use. In particular, we show how this form of uplift can be used to automatically trigger policies, expressed by users that result in device (re-)configuration.

We specify a process for semantic uplifting of real-time, low-level monitoring data and selection of policies based on extracted information to manage *HANs*. The process provides a partial realization of autonomic control loop [Kielthy et al., 2010, Strassner et al., 2009a] in *HANs*, in which the system monitors changes in the network states, analyses the data, and manages the network to ensure user requirements are being met.

[1] `http://www.wireshark.org/`

4.2 Semantic Uplifting Technique for Monitoring Data

In this section, we specify a generic technique describing the process of semantic uplifting of monitoring data and selection of policies based on extracted information based on the old implementation of framework. Monitoring systems can, in general, be classified in different ways, based on either data or user perspective.

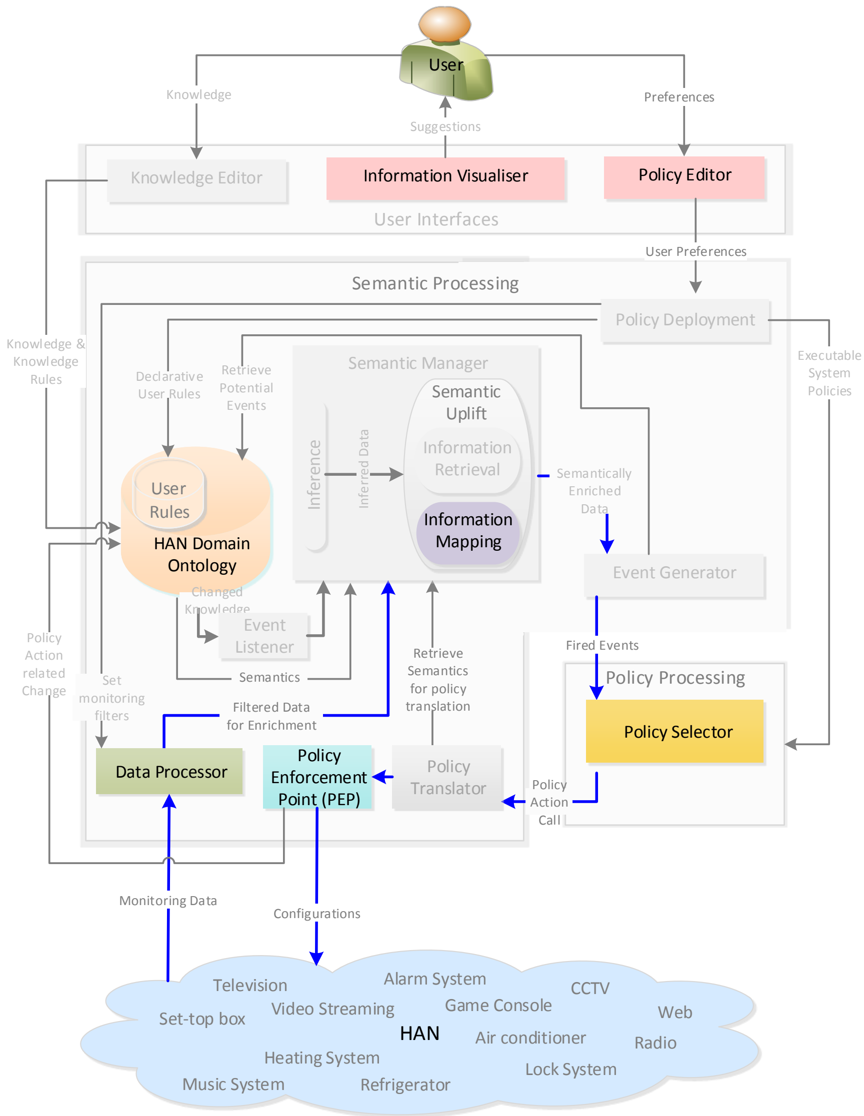


Figure 4.1: The HANmanager Framework - Highlighting the role of Semantic Manager interfacing with Data Processor to uplift monitoring data. The blue arrow connectors show the implementation of *HAN* control loop.

From the data perspective, there are two broad types of monitoring processes: active and passive [Anagnostakis et al., 2002]. Our technique, as shown in Figure 4.1, is based on an old implementation of the framework presented in Chapter 3; it can only be used for real-time passive monitoring and it analyses one packet at a time. From the user perspective, monitoring can be supervised, semi-supervised or unsupervised. The supervised monitoring is user-controlled and the unsupervised is system-controlled. Our technique uses a semi-supervised solution, where the semantics of monitoring data are ambiguous but context of monitoring is known. Furthermore, the user policies are selected based on extracted information from monitoring data after the semantic uplift.

The *data processor* is to collect data from network gateway machine (router). When system senses a packet flow on network, the *data processor* parses the packet data and filters out unnecessary data elements from packet header. Later, it formulates a vector space (*Vspace*) of filtered data elements (*FDEs*). The *FDEs* are semantically uplifted with the help of the *semantic manager* and later mapped to data properties in the *HAN* domain ontology. Using the *FDEs*' values, new information is made available from the *HAN* domain ontology. After semantic enrichment of *Vspace*, it is passed to the *policy manager* for selection of applicable policies. The *policy manager* has two subcomponents: *policy selector* and *policy translator*. The *policy selector* (an obsoleted component replaced by the *policy system* in newer version of our framework as described in Chapter 3) seeks an appropriate policy based on the information extracted from monitoring data and the *HAN* domain ontology. After successful verification process, the policy is selected and passed to the *policy translator* to generate required configurations. These generated configurations are enforced on the router by the *policy manager*. The functional detail of the framework is discussed in the following sections.

4.2.1 Low-level Data Processing

The *data processor* retrieves monitoring data from the router using deep packet inspection (DPI) tools. The retrieved information comprises low-level details of a packet header and payload. The *data processor* parses packet data and formulates a vector space (*Vspace*) of filtered data elements (*FDEs*) e.g. source and destination IP (Internet Protocol) Addresses.

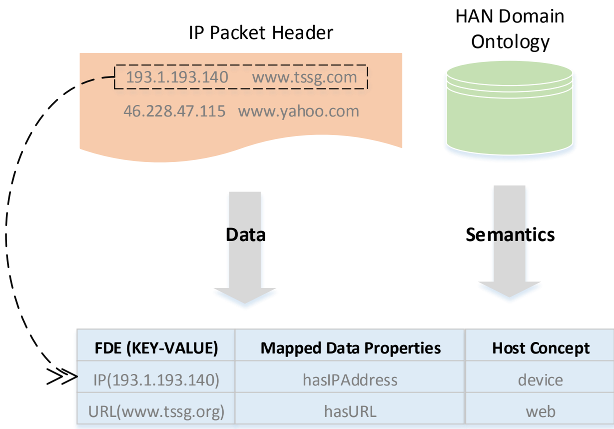


Figure 4.2: Formulation of Filtered Data Elements(*FDEs*) VSpace from low level monitoring data gathered from the router's IP traffic logs. The semantics of *FDEs* are retrieved from the *HAN* domain ontology.

A *FDE* is a key-value pair that characterizes a potential network event of user interest, e.g., a device or user accessing the Internet; however, the *FDEs* represent only low-level information. Here the context of monitoring process is known but the semantics of monitoring data are unknown. Therefore, the semantics of *FDEs* are discovered with the help of the *semantic manager*. Figure 4.2 shows the process of *Vspace* formulation.

4.2.2 Semantic Uplifting of Monitoring Data

The *semantic manager* finds matching ontological entities to map all *FDEs* in the *Vspace*. Note that *FDEs* can be mapped to Individuals or data properties in the *HAN* domain ontology because they represent data values, however, we mapped them to data properties in this technique.

The *semantic manager* retrieves a list of all Individuals and data properties from the *HAN* domain ontology and compares each *FDE*'s key against the retrieved lists to find potential mapping (by using fuzzy lexical matching technique[Schoolderman, 2012]). For mapping of *FDEs*, we used only the data properties. When a matching data property is available for a *FDE*, its corresponding *URI* is saved in the *Vspace*. When all *FDEs* in *Vspace* are mapped to the ontological data properties, the resultant *Vspace* is further

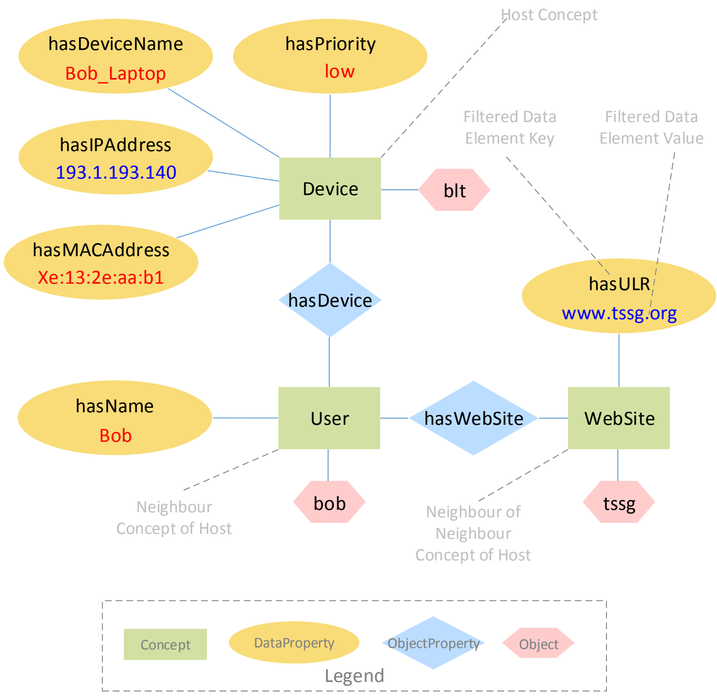


Figure 4.3: Instantiated semantic map of filtered data elements *FDEs* then they are mapped to *HAN* domain ontology; a semantic map contains different concepts, their properties and values based on relationships to each other.

processed to refine primitive semantics of *FDEs* as shown in the Figure 4.2.

This process first determines the host concept of mapped entities so that we can link all *FDEs* with their host concepts. A host concept is the domain class of the mapped data property that holds that the individual or data property. To simply the process of mapping, we assume that data properties have one-to-one mapping with their corresponding classes. The *HAN* domain ontology is queried again to find host concepts of all mapped individuals and data properties and neighbour concepts of their hosts. A semantic map is created that contains host and neighbour concepts along with their data and object properties. The semantic map is a subset of *HAN* domain ontology. Using the given data values of *FDEs*, the *HAN* domain ontology is queried and all other values of related data properties are retrieved from the *HAN* domain ontology, hence instantiating the semantic map. The motive behind making a semantic map and its instantiation is explained in the next section. Finally, the updated *Vspace* along with the instantiated semantic map is passed to the *policy selector* for further processing. The Figure 4.3 shows an instantiated semantic map of some *FDEs* that are mapped to data properties in the *HAN* domain ontology.

4.2.3 Selection of Policies based on Semantics

The *policy selector* generates separate lists of all possible events, conditions and actions using the semantic map of *FDEs*. All the host and neighbour concepts are placed in the events list. The object and data properties (along with their values) populate the conditions list and only data properties are put in the actions list. Built-in functions as conditions in SWRL rules are not currently handled. The *policy selector* fetches all policy rules and decodes each rule one by one in ECA (event, condition and action) semantic rudiments. Note that, in the antecedent part of the SWRL rule, the class atoms represent abstract events clauses; the object and data properties atoms represent condition clauses. Similarly, in the consequent part, a data property atom represents an action clause. When defining a SWRL policy using ECA format, multiple events or conditions clauses can occur in one rule body but only one action clause can occur in rule head.

After the decoding of policy rule, the policy variables (events, conditions and action) are compared with the *FDEs*' lists of variables. If all policy variables exist in their respective *FDEs*' lists, the values of condition and action variables are further analysed and verified. The SWRL policy rule is selected for translation if both action and event clauses are verified successfully.

Table 4.1: Comparison of variables and values of *FDEs* and A SWRL policy.

	FDE		SWRL	
	Variable	Value	Variable	Value
Event	`Device`	blt	`Device`	-
	`User`	bob	`User`	-
	`Web`	tssg	-	-
Condition	`hasIPAddress`	193.1.193.140	-	-
	`hasMacAddress`	xe:13:e2:aa:b1	-	-
	`hasDeviceName`	Bob_Laptop	-	-
	`hasPriority`	low	-	-
	`hasUserName`	Bob	`hasUserName`	Bob
	`hasUrl`	webmail.tssg.org	-	-
	`hasDevice`	bob	`hasDevice`	-
	`hasWeb`	tssg	`hasWeb`	-
Action	`hasIPAddress`	193.1.193.140	-	-
	`hasMacAddress`	xe:13:e2:aa:b1	-	-
	`hasDeviceName`	Bob_Laptop	-	-
	`hasPriority`	low	`hasPriority`	high
	`hasUserName`	Bob	-	-
	`hasUrl`	webmail.tssg.org	-	-

Table 4.1 shows comparison of *FDEs* and SWRL policy variables and their values.

4.3 Algorithms for Semantic Uplift and Policy Selection

In this section, we present two algorithms that together provide a generic technique to uplift semantics of monitoring data and select SWRL policy rules based on the extracted information. In later sections, we explain the use of these algorithms in two practical scenarios. The Figure 4.4 shows the flow diagram of the two algorithms.

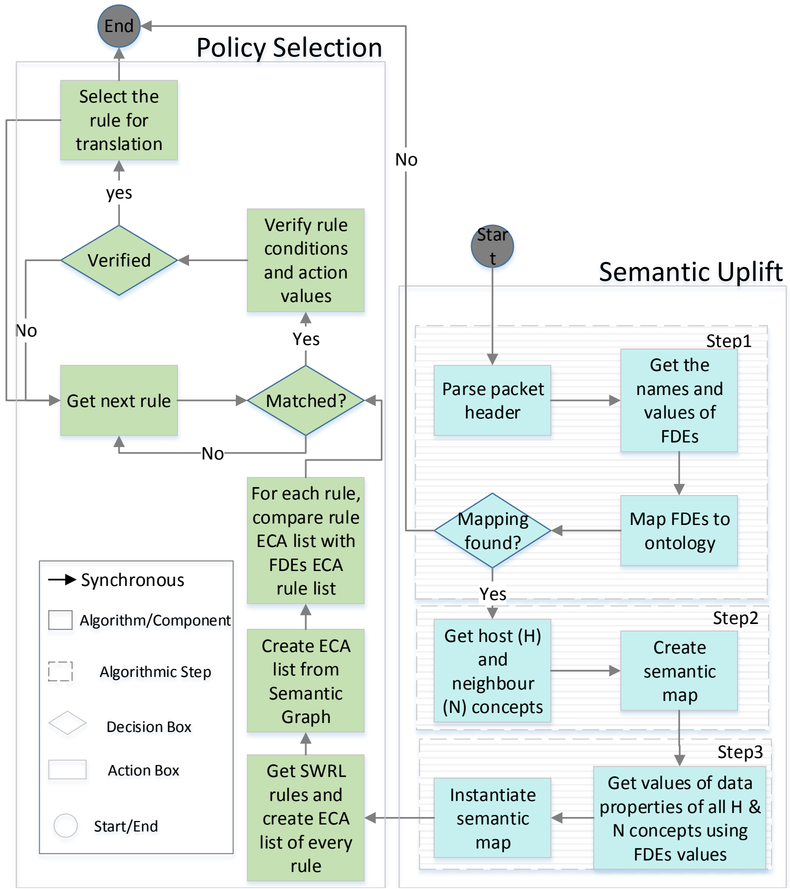


Figure 4.4: Flow diagram of Algorithms for Semantic Uplift and Policy Selection, describing different steps from mapping to selection of rules for execution.

4.3.1 Semantic Uplift Algorithm

The semantic uplift process has three main steps:

1. mapping of *FDEs* to ontology;
2. formulation of semantic map for *FDEs*; and
3. instantiation of semantic map using the values of *FDEs*.

Algorithm 1 Semantic Uplift Algorithm

Step 1: Map elements of $\mathcal{V}_s$ (*FDEs*) to $\mathcal{O}_s$

if $\mathcal{V}_s.size \neq 0$ **then**
  foreach *element* e_v *in* $\mathcal{V}_s$ **do**
    if $e_v == e_o \wedge e_o \in l_{\mathcal{P}_d})$ **then**
      $e_v.name \leftarrow e_o$
      $e_v.host \leftarrow e_o.domain$
      $e_v.host.uri \leftarrow e_o.domain.uri$
    return $\dot{\mathcal{V}}_s$

Step 2: Create semantic map for the elements in $\dot{\mathcal{V}}_s$

if $\dot{\mathcal{V}}_s.size \neq 0$ **then**
  foreach *element* e_v *in* $\dot{\mathcal{V}}_s$ **do**
    call $createMap(e_v.host)$
function $createMap$ $(ClassInfo\ info)$
foreach *object property* r *in* $info.Relations$ **do**
      $info.l_{\mathcal{N}}$.add($r.range$)
   foreach *object property* r *in* $l_{\mathcal{P}_o}$ **do**
      if $info == r.range$ **then**
         $info.l_{\mathcal{N}}$.add($r.range$)
   foreach *class* n *in* $info.l_{\mathcal{N}}$ **do**
      return $createMap(n)$
return $\ddot{\mathcal{V}}_s$

Step 3: Instantiate the semantic map $\ddot{\mathcal{V}}_s$

if $\ddot{\mathcal{V}}_s.size \neq 0$ **then**
  foreach *element* e_v *in* $\ddot{\mathcal{V}}_s$ **do**
    call $instantiateMap(e_v.host)$
function $instantiateMap$ $(ClassInfo\ info)$
foreach *relation* r *in* $info.l_{\mathcal{P}_d}$ **do**
      $info.r.value \leftarrow e_o.value$ given $info.value$
   foreach *neighbour* n *in* $info.l_{\mathcal{N}}$ **do**
      foreach r *in* $n.R \wedge r \in l_{\mathcal{P}_d}$ **do**
         $n.r.value \leftarrow e_o.value$ given $info.value$
   foreach *class* n *in* $info.l_{\mathcal{N}}$ **do**
      return $instantiateMap(n)$
return $\bar{\mathcal{V}}_s$

In the first step, the algorithm parses the received packet header and extracts useful data elements that characterize a network event. The filtered data elements (*FDEs*) are saved in a vector space $\mathcal{V}_s$ with their keys and values for further analysis. Every element e_v in $\mathcal{V}_s$ has a potential mapping to a data property e_o within the *HAN* domain ontology $\mathcal{O}_s$. To make the matching function more efficient, the algorithm maintains separate lists of ontological concepts $l_{\mathcal{C}}$, object properties $l_{\mathcal{P}_o}$, data properties $l_{\mathcal{P}_d}$ and instances $l_{\mathcal{I}}$.

Every e_v is compared with ontological data properties in $l_{\mathcal{P}_d}$. The matching process compares the key of a *FDE* with data properties list using a fuzzy lexical matching technique. It may return more than one potential mapping due to syntactical variants of e_v in the the *HAN* domain ontology $\mathcal{O}_s$. To avoid any ambiguity, we used distinguishable elements names in $\mathcal{O}_s$ and used URI of e_o for exact mapping. Step one results in enriched $\dot{\mathcal{V}}_s$, containing primitive semantics of $\mathcal{V}_s$ elements.

Now the host concepts of all $\mathcal{V}_s$ elements are determined but they could be in a disconnected state. In other words, a relation between two *FDEs* may exist in the ontology through the mapped entities (data properties) but this is not yet apparent. The second step is about creating a semantic map, which results in connecting all the *FDEs* together. In step two, the algorithm determines the neighbour concepts of all *FDEs* with respect to the host concepts of their mapped data properties in a recursive manner. A class is considered as neighbour of host concept if (a) it is a parent or child class; (b) its a range class in the object property; (c) or it is a domain class that holds host concept as a range class in its object property. In step three, when neighbour concepts are discovered, the algorithm instantiates the semantic map using the values of *FDEs* in $\mathcal{V}_s$. The instantiation process is also completed in recursive manner. By doing so, algorithm defines the semantics of *FDEs* as shown in Algorithm 1.

4.3.2 Policy Selection Algorithm

The semantic uplift algorithm generates lists of all possible events, conditions and actions using the *FDEs*' semantic map $\bar{\mathcal{V}}_s$. In the policy selection algorithm the host and neighbour concepts of all e_v in $\bar{\mathcal{V}}_s$ that belong to $l_{\mathcal{C}}$ are added to the events list $l_{\mathcal{E}_{FDE}}$. The object and data properties are added to the conditions list $l_{\mathcal{C}_{FDE}}$ and finally only data properties are added to the actions list $l_{\mathcal{A}_{FDE}}$.

All SWRL policy rules are also fetched from the replica ontology $\mathcal{O}'_s$ (to use SWRL in non-monotonic form as presented by Calero et al. [2011]; otherwise, using SWRL in its current form to make changes in domain ontology makes ontology inconsistent due to SWRL monotonicity issue). Each rule is parsed in ECA rudiments (event, condition, action) and lists of events $l_{\mathcal{E}_{pol}}$, conditions $l_{\mathcal{C}_{pol}}$ and action variables $l_{\mathcal{A}_{pol}}$ are generated.

Algorithm 2 Policy Selection Algorithm

```
if V̄_s.size ≠ 0 then
    foreach element e_v in V̄_s do
        if e_v.host ∈ l_C ∧ e_v.host.l_N.size ≠ 0 then
            foreach policy p in l_Pol do
                if p.l_E_pol ⊂ l_E_FDE ∧ p.l_C_pol ⊂ l_C_FDE ∧ p.l_A_pol ⊂ l_A_FDE then
                    foreach condition c in p.l_C_pol do
                        cFlag= true
                        if c.value ≠ l_C_FDE.r.value then
                            cFlag= false
                    if cFlag == true then
                        if a ∈ p.l_A_pol ∧ r ∈ l_A_FDE ∧ a == r ∧ a.value ≠ r.value then
                            l_SPol.add(p)
    return l_SPol
```

The variables lists of each policy rule are compared with their respective *FDEs*' variables lists. If all policy variables exist in the *FDEs* lists, the values of policy conditions and action are further verified. For the selection of policy, the policy condition values should be equal to *FDEs* condition values but action value has to be different, otherwise the algorithm assumes that policy has been executed already. The policy p is selected for translation if conditions and action values are verified successfully as shown in Algorithm 2.

4.4 Implementation and Test-bed

The complete test-bed implementation details are given in Chapter 3.

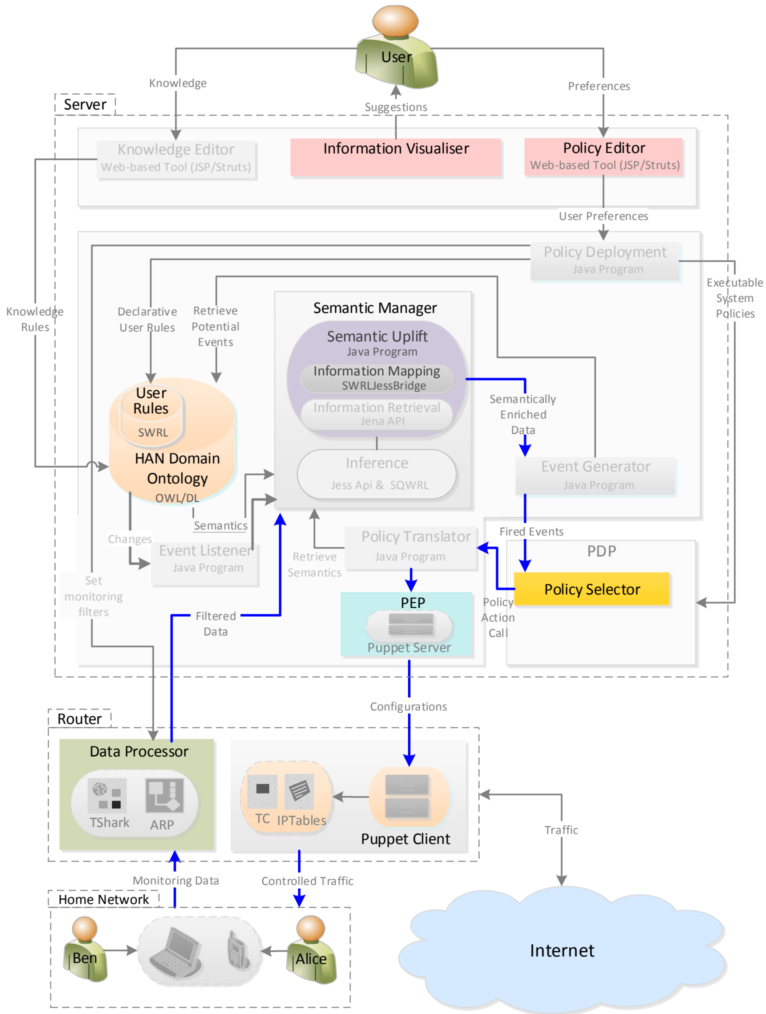


Figure 4.5: HANManager Test-bed: Shows the collection of monitoring data by Data Processor for the semantic uplifting by Semantic Manager. Based on enriched monitoring data provided by Semantic Manager, related policies are selected by Policy Selector.

This chapter covers the components that are related to this chapter only as shown in Figure 4.5. The test-bed architecture used for conducting the experiment discussed in later section. As part of our experiments, we developed a web-based interactive policy editor to specify user policies. Figure 4.5 shows the test-bed architecture used for monitoring experiments in *HAN*. SWRL is used as user policy language that is supported by OWL-based *HAN* domain ontology. The Protégé[1] tool is used for designing the ontology, the OWL-API[2] to interact with the ontology model and the Jena-API[3] to infer the *HAN* domain ontology.

The SWRL rules are processed using the Jess-API[4]. For ontology queries, we used the SQWRL[5]. The Puppet framework[6] is used for setting up PDP (Policy Decision Point) and PEP (Policy Enforcement Point) architecture in *HAN*. The semantic uplift and policy selection algorithms are developed in JAVA using the IBM Eclipse platform. For deep packet inspection of monitoring data, we used the TShark[7] application and TCPDump.

4.5 Evaluation

To demonstrate the power of our semantic uplift technique, we present two example scenarios that have been realised using the *HANmanager* test-bed presented in 4.4.

4.5.1 Test Scenarios

To illustrate the operation of our *HAN* management framework we describe two use cases. The first relates to the detection that a new device attached to the *HAN* belonging to a house guest, which in line with user preferences should be given access. The second relates to the identification, based on browsing profiles, of which one of a number of possible users is using a given device at a given time.

4.5.1.1 Test case 1: Identifying Unknown Devices

In this use case, we assumed that a device attempts to send traffic through the *HAN* router. When a data packet is parsed, we apply our semantic uplift technique to identify the source device. Based on the extracted information, a device profile is assigned to the

[1] `http://protege.stanford.edu/`
[2] `http://owlapi.sourceforge.net/`
[3] `http://jena.sourceforge.net/`
[4] `http://www.jessrules.com/`
[5] `http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL`
[6] `http://www.puppetlabs.com/`
[7] `http://www.wireshark.org/docs/man-pages/tshark.html`

traffic flow. Later with additional information, we selected applicable policies related to the source device. The following section describes the operation of the semantic uplift and policy selection algorithms in this scenario.

Description We assumed that there is a policy in place saying that all unknown devices have their traffic blocked (even if they have the access point password):

```
Device(?x)^Unknown(?x)^Guest(?y)
^isGuestPresent(?y,"no")->hasDeviceAccess(?x,"no")
(Description: if guest is absent, unknown devices have no
 access to the Internet and HAN.)
```

... *P1*

There are also policies saying that all guests visiting the home have access to internet and get high priority for the traffic they generate:

```
Device(?x)^Guest(?y)^isGuestPresent(?y,"yes")
^hasDevice(?y,?x)->hasDeviceAccess(?x,"yes")
(Description: if guest is present and device belongs to guest
 then device has access to the Internet and HAN.)
```

... *P2*

```
Device(?x)^Guest(?y)^hasDevice(?y,?x)
->hasPriority(?x,"high")
(Description: The device belonging to guest
has priority.)
```

... *P3*

Let us assume there would be guest in the house for a weekend. Assume the home owner has access to an interface that allows him/her to indicate that a guest named Alice would be in the house on Saturday and Sunday. Let us also assume that the following inference rules are in place in the *HAN* domain ontology:

```
Device(?x)^hasDeviceName(?x,?y)
^swrlb:stringEqualIgnoreCase(?y,"Unknown")
->Unknown(?x)
```

```
(Description: if device has unknown name, classify
device as unknown.)
                                                        ... I1

Device(?x)^Unknown(?x)^Guest(?y)
^isGuestPresent(?y,"yes")->hasDevice(?y, ?x)
(Description: If device is unknown and guest is present
then consider device belongs to quest.)
                                                        ... I2
```

If Alice is given the password to access the WIFI access point and starts an Internet browsing session, then we expect the following behaviour. The *data processor* passes the MAC address of the source device for the new flow to the *semantic manager*. The semantic manager does not find the MAC address in the *HAN* domain ontology and it infers that the MAC Address belongs to an unknown device based on the inference rule *I1*. The inference rule *I2* is also executed because guest is present and it further clarifies that the flow belongs to the guest. Since Alice is identified as a guest, the system infers that the device belongs to the guest. Therefore, *policy selector* selects *P2* and *P3* for translation rather than *P1*. Following is an excerpt of IPTables configurations generated for policy *P3*:

```
1) iptables -t mangle -I FORWARD -i ${LAN} -o ${WAN}
-s 192.168.22.4 -j TOS --set-tos 0x28

2) tc class add dev ${WAN} parent 1:1 classid 1:11 htb
rate 60kbps ceil 90kbps prio 1

3) tc filter add dev ${WAN} parent 1:0 prio 1 protocol
ip u32 match ip tos 0x28 0xff classid 1:11
```

The above configurations can be described as:

1. Type of service (TOS) bits of data packets are marked for highest priority;
2. A priority queue created with optimal bandwidth;

3. A packet filter is also created that enqueues high priority packets in the matching priority queue.

In the application of rules, here is a trade off between the system uncertainty in terms of scenarios and its usability in a typical *HAN*, which may not be the case in enterprise networks.

4.5.1.2 Test case 2: Identifying the user of a known device

In this test case, we exploited web surfing history of a user to take into account user characteristics and ultimately selecting a user profile based on those characteristics. The following section describes the proposed technique in this test case.

Description We assume that multiple family members may use the same device for the Internet browsing. Since different family members have different policies (preferences) governing how their traffic is prioritised, it is desirable to identify a device's current user. We assume that individual users visit Internet sites that others in the home do not visit; hence, these sites are reliable indicators of the device user. For example, parents in a house may access their work webmail servers, where as children may access ClubPenguin.com.

Let us assume the following policies are in place:

```
Device(?x)^User(?y)^hasDevice(?y,?x)
^hasUserName(?y,"Bob"->hasPriority(?x,"high")
(Description: Any device belonging to Bob
 gets high priority..)
```

... P4

```
Device(?x)^User(?y)^hasDevice(?y,?x)
^hasUserName(?y,"Mary"->hasPriority(?x,"high")
(Description: Any device belonging to Mary
 gets high priority.)
```

... P5

```
Device(?x)^User(?y)^hasDevice(?y,?x)
^hasUserName(?y,"Joey"->hasPriority(?x,"low")
(Description: Any device belonging to Joey
 gets low priority.)
```

... P6

Let us also assume that the following inference rules are in place in the *HAN* domain ontology:

```
Web(?x)^hasUrl(?x,"http://clubpenguin.com")
^User(?y)^hasUserName(?y, "Joey")->hasWeb(?y, ?x)
(Description: Joey uses clubpenguin.com.)
```

... I3

```
Web(?x)^hasUrl(?x,"https://webmail.tssg.org")
^User(?y)^hasUserName(?y, "Bob")->hasWeb(?y,?x)
(Description: Joey uses tssg.org.)
```

... I4

```
Web(?x)^hasUrl(?x,"http://vogue.co.uk")
^User(?y)^hasUserName(?y, "Mary")->hasWeb(?y,?x)
(Description: Joey uses clubpenguin.com.)
```

... I5

We assume that the home owner has access to a user interface that allows him/her to specify these identifier URLs. If we assume that Bob has been working from home then system would have identified him as he has been accessing his office email. Whilst Bob is conversing on a SIP-phone, he allows Joey to use his laptop to play games on clubpenguin.com. When Joey first accesses clubpenguin.com to login, the system infers that the user of the device “Bob-laptop" has changed from “Bob" to “Joey" using inference rule *I3*. Consequently, policy *P6* are selected and enforced on the system.

4.5.2 Experimental Results

We conducted some experiments to observe the impact of changes in selected and deployed policies on *HAN* traffic. Let us assume that our home users are conducting different internet activities simultaneously. Joey is playing a game on ClubPenguin.com, Mary is uploading her research backup using a FTP application and Bob is having a conversation over his SIP-Phone. In the background, there were a number of other active network streams downloading software updates, email and media files for other users at home. Without an explicit QoS management, the service quality of SIP call and FTP upload is extremely poor. After enforcing policies *P4*, *P5* and *P6* to prioritize and classify different users' traffic in different priority queues for better bandwidth, quality of intended services improved significantly. The Figure 4.6 shows the impact of policies on different *HAN* traffic flows.

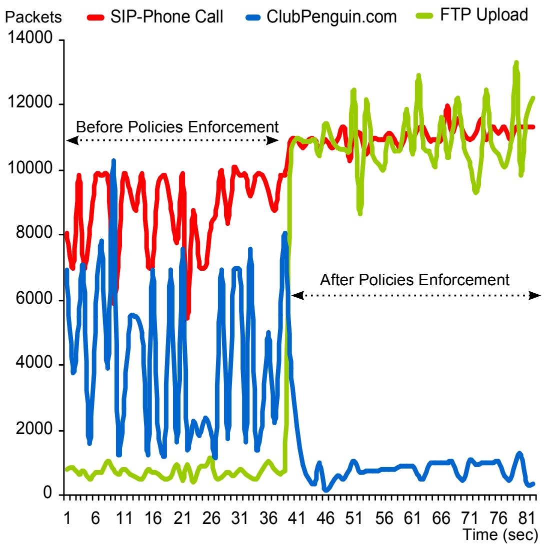


Figure 4.6: HAN Traffic Bandwidth Control using Policies: Before the application of policies, FTP upload is extremely slow and SIP-call quality is also not good because clubpenguin.com taking unfair share of bandwidth but after applying policies, the FTP upload and SIP-call quality is boosted.

4.6 Summary

We have presented a generic technique for semantic uplifting of monitoring data and selection of policies based on extracted information. In particular, we presented algorithms for implementation of our proposed approach. We believe that in home area network scenarios, many users will prioritize ease-of-configuration over guaranteed functional correctness of applied policies—the opposite holds true for typical enterprise networks. Our policy selection process, wherein inferences that may potentially be inaccurate, are used to automatically select and apply policies reflects this prioritisation of ease-of-configuration.

The proposed technique works for real-time monitoring to analyse one data packet at a time. For future work, we plan to extend this work to analyse larger network logs for *HAN* optimization. In the light of our experiments, we have argued the significance of proposed technique for interpretation of real-time, low-level monitoring data and selection of policies for user-centric *HAN* management.

Chapter 5

Semantic aware Processing of User defined Inference Rules to manage Home Networks

Increasing complexity makes it difficult for users to manage their home networks in a way that optimises their experience when using rich multimedia services. Current network management systems are not designed for ordinary network users—they do not seek to abstract the configuration details of network devices and services that need to be managed, requiring instead editing of configuration files with specific syntax and semantics. We investigated the use of semantic technologies to improve the ability of typical users to manage their network by capturing their preferences using concepts familiar to them, and applying inference techniques to link monitored network events to these preferences so that appropriate configurations can be automatically applied. The *HAN-manager* framework abstracts the detail of managing the network access and various multimedia services consumed in homes into an ontological descriptions augmented by inference rules (derived from users' interaction with system via intuitive interfaces). In this chapter, we specify semantic enrichment algorithms that analyse user supplied information and apply a reasoning process to identify events with user significance. These events are forwarded to a Policy Decision Point, triggering system policies that result in configuration actions. We demonstrate the power of our solution by implementing a set of use cases, and show that the semantic enrichment algorithms are flexible to suit a wide range of typical scenarios and performs better against a popular semantic search technique based on keyword interpretation.

The chapter is structured as follows: §5.1 gives an introduction to semantic enrichment problem addressed in this chapter. In §5.2, we describe the technique to address the problem. In §5.3 section, we specify algorithms for semantic enrichment and policy processing. §5.4 section describes the implementation details and an evaluation of our test-bed. §5.5 section presents use cases used to demonstrate different operations of the *HANmanager* and provides empirical results for evaluation of its performance. Finally, in §5.6 we conclude and summarise our findings and outline further work.

5.1 Introduction

The *HANmanager* framework (discussed in Chapter 3), which uses semantic technologies to construct a model of home area network and illustrates that how semantic model can be used to control *HAN* systems, allows home users to specify their preferences through intuitive and easy to understand interfaces. These preferences describe how devices and application will be accessed within *HAN*. Users can also supply high-level basic information that the framework needs to be fully useful and functional; for example, names of other users and devices, whether guests are present in the home, and what rooms certain services can be accessed in and by whom. The *HANmanager* starts its work by capturing raw monitored data either from network flows at bottom level (which is discussed in Chapter 4) or from changes made by the user at top level of the *HAN* system. The framework enriches the semantics of captured data to make sense of it. The process of semantic enrichment of user defined inference rules is described in this chapter.

The chapter builds upon Chapter 4) with three new significant additions. Firstly, the version of the framework presented in this chapter decouples the semantic analysis approach from the policy management system, allowing the use of different policy management systems (in the newer implementation we use PONDER2, [Twidle and Lupu, 2007]). Secondly, we implement new semantic analysis techniques and algorithms for high-level network related information added into the system by the home network users. The proposed technique in this chapter has also overcome many of the shortcomings of our previous approach that doesn't separate system policies (as rules) from configurations (as settings). We compare our enhanced semantic analysis techniques in terms of accuracy with a well known semantic search approach [Tran et al., 2007] that is based on the keyword interpretation.

5.2 Semantic Enrichment of Inferred Data Technique

In this section, we give an overview of Semantic Enrichment technique for inferred data to monitor events of user's interest and apply the changes to *HAN* accordingly. We use *HANmanager* framework discussed in Chapter 3 for the realisation of management control loop as described in [Jennings et al., 2007, Kielthy et al., 2010] for *HAN*. A subset of the *HANmanager* framework, used for semantic enrichment of inferred data, is shown in Figure 5.1.

The frameworks works as follows: When user specify knowledge rules into the system, the generated network events are captured using the inferred data. The captured data is semantically enriched by analysing its semantics and dynamically linking to instances of user importance (relevant to user preferences) in the domain ontology (that represents *HAN* domain model). The inferred linkages are determined using inference rules contained in the ontology, which take into consideration the contextual information and may also produce new knowledge. The inferred knowledge helps the *HANmanager* triggering system policies (that are actually interpretations of user preferences in the form of system rules). To trigger system policies, related policy events are fired and execution process is initiated at Policy Decision Point (the router in our case) causing reconfiguration of networking devices, applications and services in order to ensure that users preferences are consistently met.

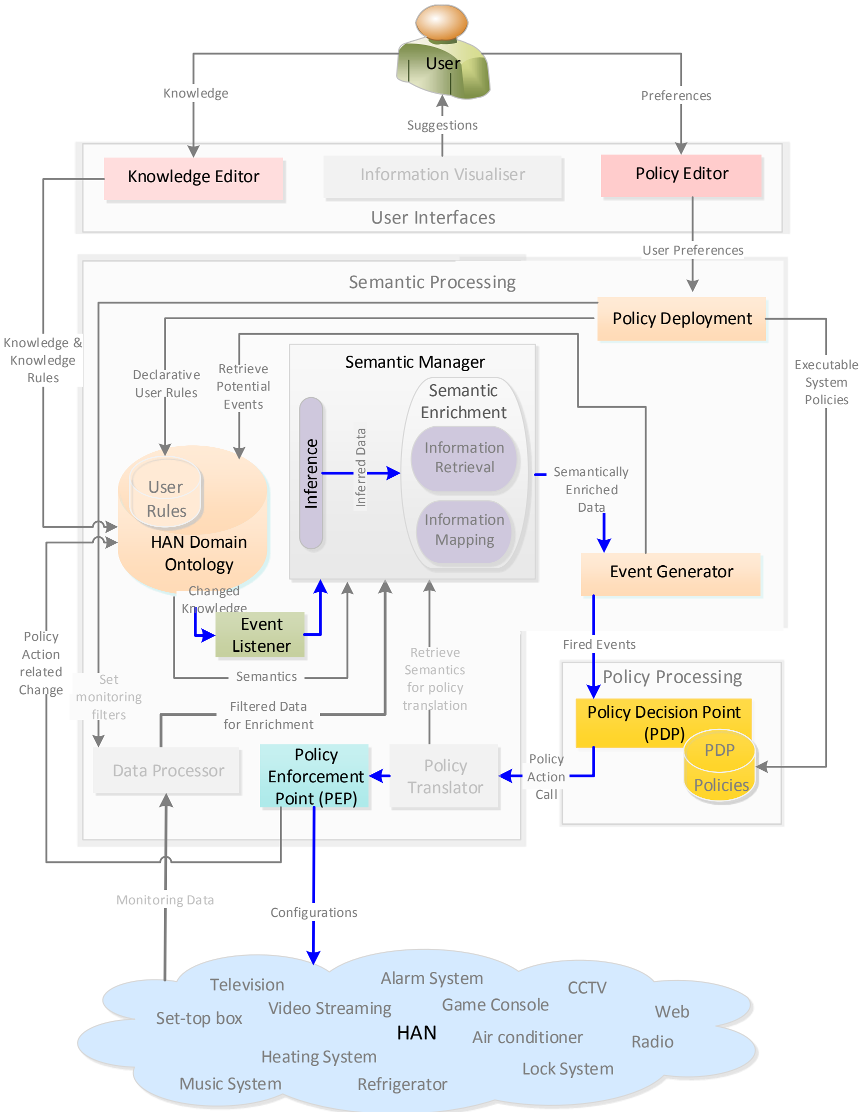
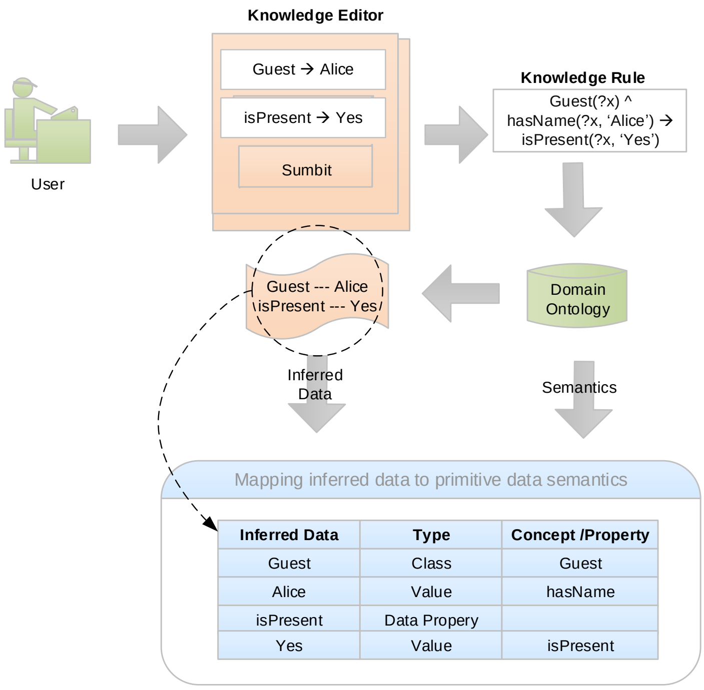


Figure 5.1: The HANmanager Framework - Highlighting the role of Semantic Manager interfacing with Policy Decision Point via Event Generator. The blue arrow connectors show the implementation of *HAN* control loop and the orange coloured subunits of framework are the plug-and-play third-party components.

5.2.1 Specification of Knowledge Rules

The *knowledge editor* (see §3.3.4 in Chapter 3) is a user-centric management interface through which a home network user can supply high-level information about the *HAN* system; for example, the names of users and devices, whether guests are present in the home, and what rooms certain services can be accessed in and by whom. The *knowledge editor* is very similar to the *policy editor* (see $3.3.2 in Chapter 3); however, the purpose of its use is different and knowledge rules are directly saved into the *HAN* domain ontology in the "user view" in contrast to user policy rules (which are saved in the policy model). With the help of the *knowledge editor* (see $3.3.4 in Chapter 3), home network user can add new knowledge about the "user view" entities or instances. The nature of knowledge can be simple addition or change in characteristic, behavioural or relational property of an entity or of its instance in the *HAN* domain ontology. The specified knowledge is either specific about an instance of an entity or can be general, being applicable to all instances of entry type. The specified knowledge rules are saved in the *HAN* domain ontology and once they are reasoned over, the resulting inferred data is closely monitored and used by the *semantic manager* (see §3.3.7 in Chapter 3) to capture information related to managed events in *HAN*.



Inferred Data	Type	Concept /Property
Guest	Class	Guest
Alice	Value	hasName
isPresent	Data Propery	
Yes	Value	isPresent

Figure 5.2: Parsing of inferred data and mapping to primitive semantics

5.2.2 Semantic Enrichment of Inferred Data

The *semantic manager* enriches the semantics of inferred data by mapping it to the ontological concepts in the *HAN* domain. Initial mappings of data to the ontology provides primitive semantics as shown in Figure 5.2, which are further extended to next levels for detailed and enriched semantics. To fetch the primitive semantics of network flow data, the *semantic manager* uses the *HAN* domain ontology. A network flow can be monitoring data collected from gateway or inferred data collected from *HAN* domain ontology after the reasoning. In this chapter, we mainly focus on the enrichment of inferred data.

The *semantic manager* monitors the *HAN* domain ontology for the inferred data retrieved after the reasoning. Any change in the *HAN* domain ontology is considered potentially a network event.

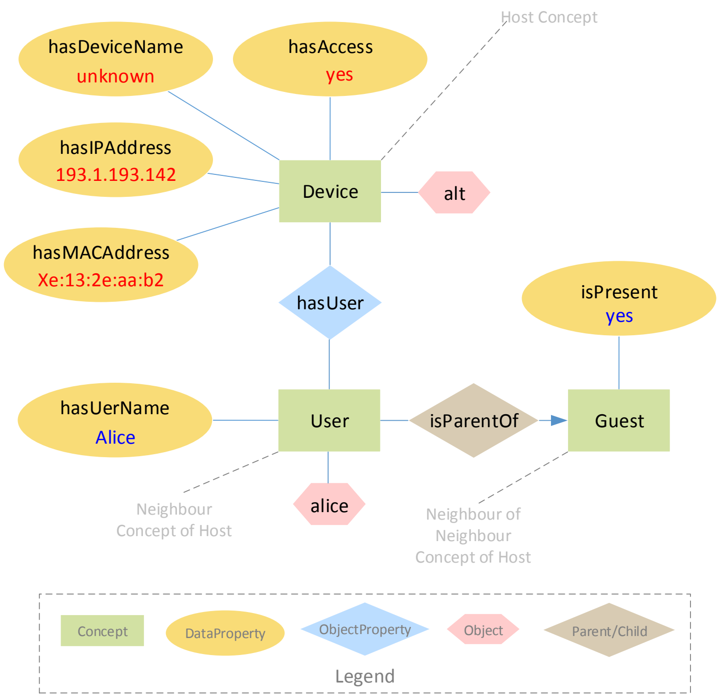


Figure 5.3: Semantic enrichment of inferred data in blue and retrieved semantic information in red

It processes all the changes in the *HAN* domain ontology after the reasoning over the knowledge rules. There can be three types of changes in the *HAN* domain ontology: reclassification change, characteristic change or behavioural change. Reclassification is a change that reclassifies an entity under different entity group, e.g., classifying a device as a mobile handset. A characteristic change is related to the modification of a data property of an entity, e.g., changing the current location of user. The behavioural change is related to an alteration in the object property of an entity, e.g., change of ownership of a device to another user. For any change, the target entity and its value are taken for further analysis. All of the three types changes can result in different kinds of inferred knowledge.

The entities in the *HAN* domain ontology can be linked to each other in many different ways (syntactically, morphologically or semantically), however, we only focus on semantic relationships. The *semantic manager* finds the leads of inferred data in the *HAN* domain ontology using a lexical semantic search technique based on cosine similarity [Tata and Patel, 2007] and once the primitive semantics are found, the semantics are enriched to the next level depending on the depth of ontology graph. A semantic graph (an example is in Figure 5.3) of interlinked entities is created by exploiting the relational properties of entities with the other entities in the *HAN* domain ontology. The semantic enrichment process works in recursive manner to get transitive entity relationships (indirect entity relationships). Using the values of entities in the network flow data, the semantic graph is also instantiated in recursive manner. The instantiation of semantic graph is a process of retrieving instance related information of the entities. In this thesis, we only support one-to-one entity-instance relationships. The instantiated semantic graph is later used for different purports (semantic-aware monitoring and monitoring data visualisation, policy processing and translation).

5.2.3 Event Generation from Inferred Knowledge

Once the semantics of inferred data are fetched in the form of an instantiated semantic graph, the *event generator* (see §3.3.9 in Chapter 3) iterates over the list of abstract managed events, also retrieved from *HAN* domain ontology, and searches for events' related information in the semantic graph. If relevant information is available e.g., a guest is present or a certain user is trying to access the Internet, the associated policy event is fired so that the *policy system* (see §3.3.10 in Chapter 3) can process related system policies. The *event generator* fires event by porting the event related information to the *policy system*. Thus, the fired events are managed by the *policy system* for further processing of system policies.

5.2.4 Policy Execution for Fired Events

The *semantic manager* interfaces with the *policy system* through the *event generator*. When an event is fired by the *event generator*, the Policy Decision Point (*policy system*) matches it with the list of events of active system policies (saved in the PDP policy repository). If a related system policy exists and meets the criteria, then the selected policy is evaluated. The system policy is executed by the *policy system* after successful evaluation. The execution of system policy triggers its translation to system configurations using the semantic graph. The *policy enforcement point* enforces the system policy and apply the system configuration to the router. In Chapter 4, we translated user policies directly to system configurations using the domain ontology. It simplified the system work flow, however, the policy management (event generation, policy evaluation and execution) was poorly accomplished. Rather than building our own policy management system based on declarative user defined rules, here we use a third party policy management system as explained in §5.4.

5.3 Algorithms for Semantic Enrichment of Inferred Data

This section explains the techniques and related algorithms that are developed to implement the *HANmanager* framework. The following sub-sections describe the algorithms for semantic enrichment, policy processing and policy translation. The notations used for the algorithms are described in Table 5.1.

Table 5.1: Algorithmic Notations

Notation	Description
IG	Inferred Graph
FDE	Filtered Data Element
$\mathcal{V}_s$	Vector - contains *IGs* or *FDEs*
$\dot{\mathcal{V}}_s$	Mapped $\mathcal{V}_s$ - contains mappings for *IGs* or *FDEs*
$\ddot{\mathcal{V}}_s$	Instantiated $\dot{\mathcal{V}}_s$ - contains Semantic Graphs for *IGs* or *FDEs*
e_v	Vector Element - either *IG* or *FDE*
$\mathcal{O}$	Domain Ontology
$\mathcal{O}_u$	User View of Domain Ontology
e_u	Element of User View in Domain Ontology
$l_{\mathcal{C}}$	List of Classes
$l_{\mathcal{P}_o}$	List of Object Properties
$l_{\mathcal{P}_d}$	List of Data Properties
$l_{\mathcal{I}}$	List of Individuals
d	Iteration Depth
$l_{\mathcal{CH}}$	List of Children Classes
$l_{\mathcal{P}}$	List of Parent Classes
$l_{\mathcal{N}}$	List of Neighbour Classes
R	Set of relations - Power set of $l_{\mathcal{P}_o}$ and $l_{\mathcal{P}_d}$
E_m	Managed Events - stored in $\mathcal{O}$
G_e	Entity Graphs - contains managed events related information
g_e	An Instance of Entity Graph - representing a managed event
n_o	Root Node of an Entity Graph
n_l	Left Leaf Node of an Entity Graph
n_r	Right Leaf Node of an Entity Graph
e_m	A Managed Event
p_u	User Rule
p_n	Management Policy
p_d	Device specific Configuration
R_p	Policy Repository
e_a	Action Entity - an entity whose property is being changed
$\mathcal{P}_{d_a}$	Action Property - an property whose value is being changed
d_v	Data Value - a value for action property
p_t	Configuration Template
$\bar{p}_t$	Filled Configuration Template

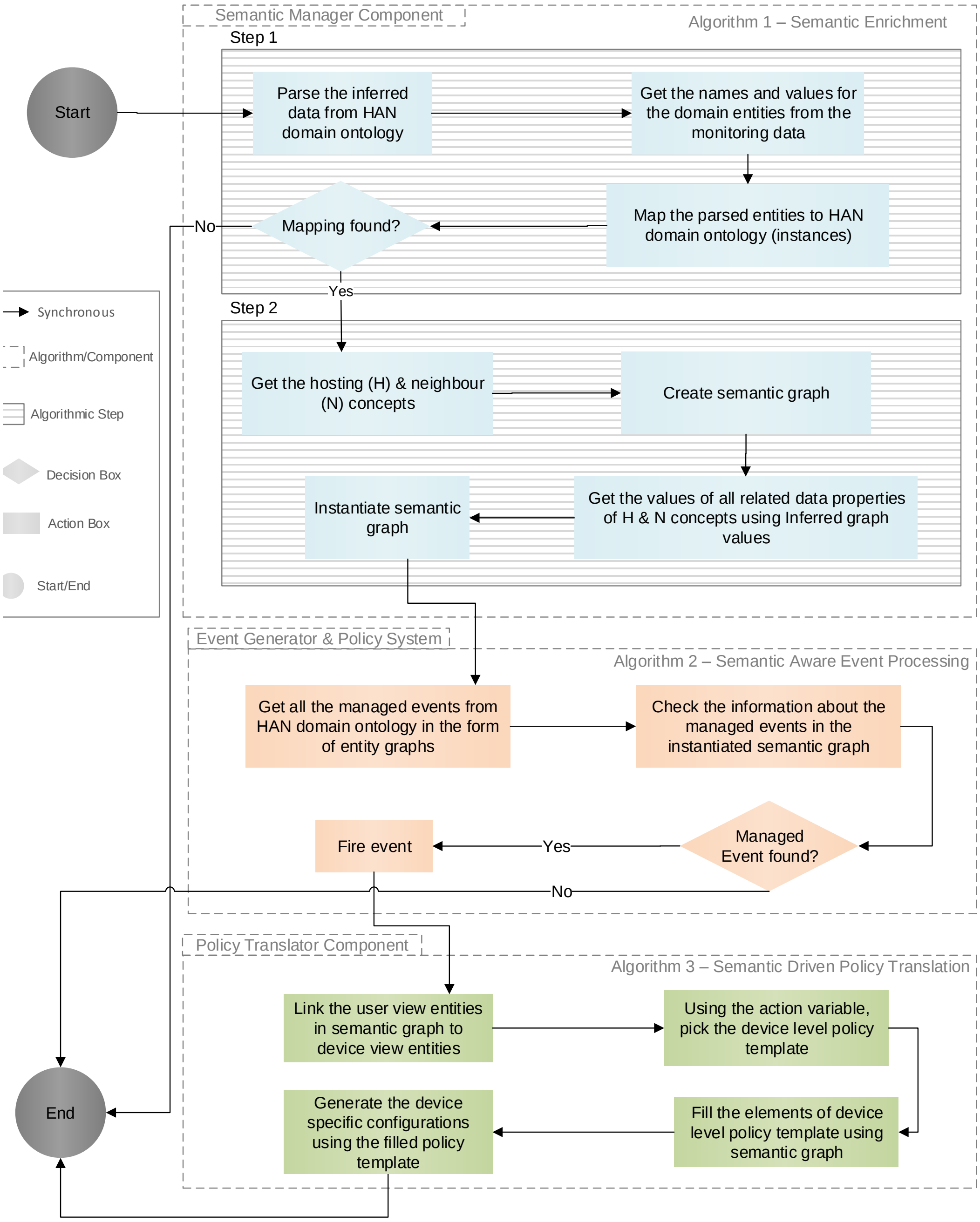


Figure 5.4: Activity diagram of Semantic Enrichment Algorithm - Stepwise explanation of Semantic Enrichment process starting from mapping of inferred data to the *HAN* domain ontology and ending at creation of instantiated semantic graph.

In this thesis, the semantic enrichment is regarded as elaborated concepts (mainly from network and device views of *HAN* domain ontology), their relationships and properties entailment to primitive information (gathered from inferred data) using querying the domain ontology. The discovered semantics (concepts) are used for the analysis of inferred data and then used for selecting and translating the user rules to manage and control home networks. If the inferred data is related to user rules that exist in the *HAN* domain ontology then the inferred information is utilised for firing associated *policy system* events. Figure 5.4 shows the activity diagram of semantic enrichment algorithm.

5.3.1 Semantic Enrichment as a Graph Search Problem

The key assumption of the proposed technique is that the *HAN* domain entities (that exist at different sub-domains within the *HAN* domain) are interlinked with each other through different relational properties. Through these relational properties, we can enrich the primitive semantics of an entity in a sub-domain by discovering other related entities in the same or other sub-domains. Essentially it is an associative (more strictly connotative) semantics search problem.

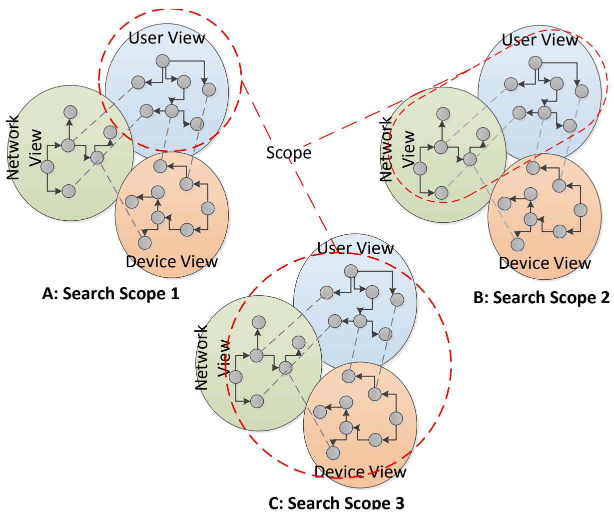


Figure 5.5: Scope of Semantic Graph Search - Search scope 1 deals with “user view”, search scope 2 deals with “user view” and “network view”, and search scope 3 covers all three views in semantic search.

To define semantic enrichment as a graph search problem, assume that ontology is represented as ontology graph $G = (V, E)$ (where V is the set of concepts (classes) in the ontology and E is the set of relationships between concepts) of order $n > 0$ and h is a host vertex from which we start creating a semantic graph G_s (representing the semantics of inferred data) such that semantic graph $G_s \subseteq G$ (ontology graph). The semantic graph G_s is a multiple arcs graph but one of the shortcomings of our approach is that we assumes only one instance per concept. The vertices are numbered from 1 to $n = |V|$, i.e. $V = \{c_1, c_2, \ldots, c_n\}$.

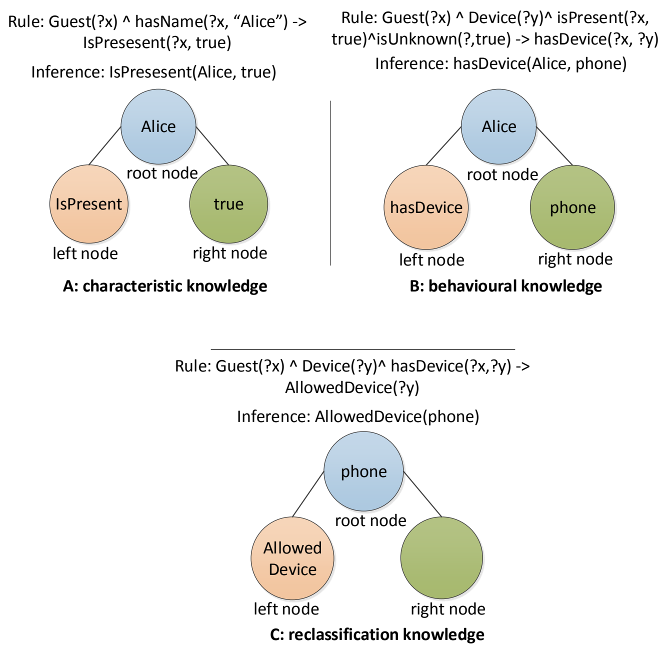


Figure 5.6: Inferred Graph for Different Knowledge Rules - Characteristic Knowledge (when value of a data property of an instance of an entity is changed), Behavioural Knowledge (when the value of an object property of an instance of an entity is changed) and Reclassification Knowledge (when class of an instance of an entity is changed).

Now, the problem is that how can we retrieve the semantics of inferred data from ontology graph G? First we require a list of all adjacent vertices of distance 1 from host vertex h. The adjacent vertices of host vertex h can be parent, child or neighbour (connected via a relational property, i.e., object or data property) in the ontology graph

G. Depending on the required depth d (level of exploration) for the semantic graph G_s, we keep on discovering distance 1 vertices of all discovered vertices until required depth d is reached. A search scope can also be defined to delimit the search to a certain view S in the ontology graph G, where $S = \{x, y, z\} \wedge x, y, z \subseteq V \wedge x < y < z$ as shown in Figure 5.5. The search scope S and depth d affirm that resultant semantic graph G_s rooted at host h is of depth d and its all vertices are within the scope of S are subset of V.

5.3.2 Semantic Enrichment of Inferred Graph

In this section, we describe the process of semantic enrichment for the inferred data gathered from the *HAN* domain ontology when it is reasoned over by a reasoner. The inferred data is about capturing any change that occurs in the *HAN* domain ontology. In this chapter, we focus on the changes made to the "user view" entities that are exposed to home users for the specification of their network related preferences. A change in the *HAN* domain ontology can be either implicit (caused by inference) or explicit (added by the user or by the *HANmanager* itself). The change can be an addition, update or deletion to "user-view" entities and their properties. In this thesis, we only support addition or update related changes.

The inferred data represents information about a potential managed event occurring at the high-level of *HAN* system when new information is added or existing information is changed in the *HAN* domain ontology. For each piece of inferred data, an inferred graph *IG* is created. An *IG* is a three node graph that is populated depending on the category of inferred knowledge. For the reclassification, characteristic or behavioural knowledge, the root node of the *IG* always contains the instance of an entity that is changed in the *HAN* domain ontology. In case of reclassification, the left leaf node of *IG* keeps the name of reclassified entity class, and for characteristic or behavioural knowledge, it contains the name of property/variable that has been changed. The right leaf node in *IG* remains empty for reclassification knowledge and for characteristic or behavioural change, it keeps the value of the instance that has been set for property/variable in left node of *IG* as shown in Figure 5.6. An *IG* is a representation of information about a potential network event in the *HAN* system. The list of inferred graphs is further processed to get refined semantics of inferred data.

Algorithm 3 Semantic Enrichment Algorithm

Step 1: Map elements of $\mathcal{V}_s$ ($\boldsymbol{IGs}$) to $\mathcal{O}_u$

if $\mathcal{V}_s.size \neq 0$ **then**
  foreach *element* e_v *in* $\mathcal{V}_s$ **do**
    if $e_v.rootnode.name \approx e_u.name \wedge e_u \in l_{\mathcal{I}}$ **then**
      $e_v.host \leftarrow e_u$
  return $\dot{\mathcal{V}}_s$

Step 2: Create semantic graph for the elements in $\dot{\mathcal{V}}_s$

if $\dot{\mathcal{V}}_s.size \neq 0$ **then**
  foreach *element* e_v *in* $\dot{\mathcal{V}}_s$ **do**
    call $createGraph(e_v.instance, d, \mathcal{O}_u)$
function *createGraph* (*individual* i , *Depth* d, *Ontology* $\mathcal{O}_u$))
if $d >= 1 \wedge i.visitedAlready == false \wedge i \in \mathcal{O}_u.l_{\mathcal{I}}$ **then**
  foreach *child class* c_i *in* $i.l_{\mathcal{CH}}$ **do**
    $i.l_{\mathcal{N}}$.add(c_i)
  foreach *parent class* pa_i *in* $i.l_{\mathcal{P}}$ **do**
    foreach *object property* r *in* $pa_i.R$ **do**
      $i.l_{\mathcal{N}}$.add($r.range$)
    foreach *object property* r *in* $l_{\mathcal{P}_o}$ **do**
      if $pa_i == r.range$ **then**
        $i.l_{\mathcal{N}}$.add($r.range$)
    $i.l_{\mathcal{N}}$.add(pa_i)
  foreach *object property* r *in* $i.R$ **do**
    $i.l_{\mathcal{N}}$.add($r.range$)
  foreach *object property* r *in* $l_{\mathcal{P}_o}$ **do**
    if $i == r.range$ **then**
      $i.l_{\mathcal{N}}$.add($r.range$)
  foreach *neighbour* n *in* $i.l_{\mathcal{N}}$ **do**
    get $n.instance$ given $i.value$ in query
  $i.visitedAlready == true$
  foreach *neighbour class* n *in* $i.l_{\mathcal{N}}$ **do**
    call $createGraph(n.instance, d-1, \mathcal{O}_u)$
  return $\ddot{\mathcal{V}}_s$

In Algorithm 3 as a first step, the list of "inferred graphs" (IGs) is saved in a vector $\mathcal{V}_s$. Every element e_v in vector $\mathcal{V}_s$ may have a potential mapping to an instance e_u within the "user view" $\mathcal{O}_u$ of the *HAN* domain ontology $\mathcal{O}$. To make the mapping function more efficient, the algorithm maintains separate lists of ontological classes $l_{\mathcal{C}}$, object properties $l_{\mathcal{P}_o}$, data properties $l_{\mathcal{P}_d}$ and instances (individuals) $l_{\mathcal{I}}$.

As a second step, once appropriate mapping is found in the "user view" part of $\mathcal{O}_u$, a graph $\dot{\mathcal{V}}_s$ of related entities within the $\mathcal{O}_u$ is built around the mapped instance i of mapped concept (class) using its entity relationships. Children (sub-classes) and parents (super-classes) nodes are added as neighbours of instance i. The neighbours of parents are also added in the neighbour list of instance i. Firstly, entity related classes are added and then using the value of mapped instance i, the semantic graph $\dot{\mathcal{V}}_s$ is instantiated. The instantiated semantic graph contains the instances of neighbours and their related data properties, however, not every entity class may contain an instance. In that case semantic enrichment may not work properly as the instance level information provides specificity to an entity relationship. The semantic graph depth/height is adjustable; for each level of depth, semantic graph goes through further extension by exploring not fully visited entities in the semantic graph. By exploiting other related entities through all possible relationships, semantic graph is stretched forth and instantiated in recursive manner until all related entities at desired depth level are part of the instantiated semantic graph $\ddot{\mathcal{V}}_s$.

5.3.3 Policy Processing by Policy System

In this section, we explain the algorithm for semantic-driven policy processing. The proposed technique replaces the *policy selector* as presented in Chapter 4. When an instantiated semantic graph $\ddot{\mathcal{V}}_s$ is available, the policy processing algorithm retrieves all managed events E_m from the *HAN* domain ontology $\mathcal{O}$ in the form of entity graphs G_e. An entity graph g_e is a three node graph structure similar to inferred graph IG, containing root node n_o, left leaf node n_l and a right leaf node n_r. However, unlike inferred graphs, entity graphs are populated with the managed events saved in the policy model of *HAN* domain ontology $\mathcal{O}$.

In Algorithm 4, for each managed event e_{m_i}, the instantiated semantic graph $\ddot{\mathcal{V}}_s$ is searched if the data related to the root node n_{o_i} is available. A lexical matching technique using cosine similarity [Tata and Patel, 2007] is used for the searching lexicons in the ontology $\mathcal{O}$. If mapping for the root node n_{o_i} is found and then the value of n_{l_i} is fetched from the $\mathcal{O}$ and set into the right node n_{r_i} of the entity graph g_{e_i} and an event

e_{m_i} is fired for the *policy system*. Firing an event indicates that an event e_{m_i} has occurred in the *HAN* system and associated system policy p_{n_i} in the policy repository R_p of the *policy system* should be processed for execution. The *policy system* evaluates the selected policy p_{n_i} and if the evaluation process is successful then policy p_{n_i} is executed and translated to generate a device specific configurations p_{d_i}.

5.3.4 Semantic-Aware Policy Translation

In this section, we explain the algorithm for the semantic-aware policy translation of user policy rule to generate network configurations. When a system policy p_{n_i} is executed, from its pertained user policy rule p_{u_i}, the information about the action entity e_a and the action property $\mathcal{P}_{d_a}$ with its newly set value d_v, are retrieved.

We assumed that every action property is reflected in the other sub-domains of the *HAN* domain ontology, therefore using the action property $\mathcal{P}_{d_a}$ as a lead to the "device view" $\mathcal{O}_d$ of ontology $\mathcal{O}$, a new semantic graph $\bar{\mathcal{V}}_s$ is created and instantiated using the value d_v of action property $\mathcal{P}_{d_a}$ of action entity e_a. The semantics of action data property $\mathcal{P}_{d_a}$ of "user view" $\mathcal{O}_u$ are encapsulated through either data property or entity in the "device view" $\mathcal{O}_d$ (it depends on how the *HAN* domain ontology is designed). In this thesis, the action data properties of "user view" $\mathcal{O}_u$ are perceived as entities in "device view" $\mathcal{O}_d$.

In Algorithm 5, using lexical matching, action data property $\mathcal{P}_{d_a}$ is matched with the classes enlisted in $l_{\mathcal{C}}$ or with the list of data properties $l_{\mathcal{P}_d}$ in the "device view" $\mathcal{O}_d$. When a match is found, related information is saved in instantiated vector $\ddot{\mathcal{V}}_s$. Using the key and values of recently saved information in $\ddot{\mathcal{V}}_s$, an extended semantic graph $\bar{\mathcal{V}}_s$ is created and instantiated from the "device view" $\mathcal{O}_d$. When semantic graph $\bar{\mathcal{V}}_s$ is

Algorithm 4 Policy Processing Algorithm

if $\ddot{\mathcal{V}}_s.size \neq 0$ **then**
  foreach *element* q_{e_i} *in* G_e *where* $q_{e_i} \equiv e_{m_i} \wedge e_{m_i} \in E_m$ **do**
    if $q_{e_i}.n_o$ *exists in* $\ddot{\mathcal{V}}_s \wedge q_{e_i}.n_o == e_v \wedge e_v \in \ddot{\mathcal{V}}_s \wedge q_{e_i}.n_l == e_v.dataproperty_{name}$ **then**
      $n_{r_i} \leftarrow e_v.dataproperty_{value}$
      call PolicySystem.fire(e_{m_i})

Algorithm 5 Policy Translation Algorithm

if $e_a \in \ddot{\mathcal{V}}_s \wedge e_a.\mathcal{P}_{d_a} \approx e_d \wedge e_d \in \mathcal{O}_d$ **then**
    if $e_d \in \mathcal{O}_d.l_{\mathcal{C}}$ **then**
        get $e_d.instance\ using\ d_v\ of\ \mathcal{P}_{d_a}\ and\ e_a.value$ in query
        $e_a.l_{\mathcal{N}}.add(e_d)$
        $i \leftarrow e_d.instance$
        $mapped\_intance \leftarrow e_d.instance$
    if $e_d \in \mathcal{O}_d.l_{\mathcal{P}_d}$ **then**
        get $e_d.host.instance\ using\ e_a.value$ in query
        $e_a.l_{\mathcal{N}}.add(e_d.host)$
        $i \leftarrow e_d.host.instance$
    call $createGraph(i, d, \mathcal{O}_d)$
    return $\bar{\mathcal{V}}_s$
if $\bar{\mathcal{V}}_s.size \neq 0$ **then**
    $p_t \leftarrow PolicyTemplateRepo(mapped\_intance)$
    fill *elements* of p_t using $\bar{\mathcal{V}}_s$
    return $\bar{p_t}$

available, an appropriate device specific configuration template structure p_t is fetched from the template repository with the help of *mapped_intance* of action entity e_a in the "device view" $\mathcal{O}_d$. Using the extended instantiated semantic graph $\bar{\mathcal{V}}_s$, the configuration template p_t is filled and transformed into the device specific configuration that is later enforced on the device.

5.3.5 Complexity Analysis

In this section, we present an asymptotic complexity analysis (time and space) of Algorithms 3, 4 and 5 using the notion given in Table 5.2.

Table 5.2: Semantic Enrichment Algorithms Complexity Analysis Variables

Variable	Name	Explanation
b	Branching Factor	the number of different new states generated from a state
d	Depth of a Solution	the shortest length from the initial state to one of the goal states
k	Search Level	Level of the search in the graph or tree
n	Size of a Problem	Size of input in a problem that needs to be processed
s	Search Scope	Search scope divides graph nodes

The semantic enrichment algorithm (Algorithm 3) is based on "iterative deepening depth-first search" (*IDS*) [Korf, 1985], however, it does not repeat expanding already visited graph nodes and, secondly, if a graph node does not belong to the target search scope, the node is not expanded further. In our algorithm, we use depth limit as a base function to halt the algorithm along with the search scope to delimit graph expansion. *IDS* works by running "depth-first search" (*DFS*) repeatedly with a growing constraint on how deep to explore the semantic graph. This gives a search that is effectively "breadth-first search" (*BFS*) with the low memory requirements of *DFS*. Thus it combines the advantages of both search strategies, taking the completeness and optimality of *BFS* and the minimal memory space of *DFS*. The completeness and optimality features are discussed briefly later in this section.

Semantic enrichment may cause extra computation by visiting nodes multiple times, however, the wasted computation does not affect the asymptotic growth of the run time for exponential searches and also it helps in completeness for semantic search. Let us assume that we are generating a semantic graph and it has reached to a depth d; *IDS* expands nodes at depth d once, nodes at depth $d - 1$ twice, nodes at depth $d - 2$ thrice and so on, until depth 1 is reached and nodes are expanded d times; the time complexity in the worst case scenario is: $(d)b^1 + (d - 1)b^2 + ... + (2)b^{d-1} + (1)b^d = \mathcal{O}(b^d)$ where b is (fixed) branching factor and d is depth. However this is not an actual time complexity of Algorithm 3. The $\mathcal{O}(b^d)$ represents the complexity for creating semantic graph in Algorithm 3 only. For worse case scenario, the complexity of Step 1 in

Algorithm 3 is $n(c) = \mathcal{O}(n)$ where n is the size of the problem. Combining all two steps together, the time complexity becomes $\mathcal{O}(b^d + n)$. If the semantic graph instantiation takes same amount of time as does its creation then the actual time complexity becomes $\mathcal{O}(2(b^d) + n)$.

The semantic enrichment is based on *IDS*, so, at any given time it is performing as *DFS*, and never searches deeper than depth d. The space it uses is $\mathcal{O}(k)$ at each iteration level k (where k is search level) so it has linear space complexity $\mathcal{O}(bk)$ unlike *BFS*, which is exponential. Moreover, the semantic enrichment algorithm is complete (it does not engage in loops), unlike *DFS*, when the branching factor b is finite. This means, it finds a solution if it exists and does not get engaged in infinite loops for the loop containing semantic graphs [Greenlaw, 1990]. The semantic enrichment algorithm is also optimal when the steps of node exploration are of the same cost for each node, however, results may vary for different graphs depending on the branching factor.

The policy processing algorithm (Algorithm 4) contains only one "foreach" loops with "ifelse" statements. In worse case scenario, the time and space complexity of the algorithm is $c * (n) = \mathcal{O}(n)$. The policy translation algorithm(Algorithm 5) uses semantic enrichment algorithm with some additional "ifelse" statements. Therefore in worse case scenario, the time and space complexity of the algorithm is same as of the semantic enrichment $\mathcal{O}(2(b^d) + n)$.

5.4 Implementation and Test-bed

The complete test-bed implementation details are given in Chapter 3. The test-bed used to implement the *HANmanager* comprised a single Ubuntu Linux router connecting a *HAN* to the Internet. Two network interface cards are used for converting the Linux machine to a router. The *HAN* has one Ethernet client machine (a Windows XP desktop) and two wi-fi client machines (one Windows XP laptop and one HTC smart phone). As depicted in Figure 5.7, the *HANmanager* functional components are deployed across the *HAN* gateway router and a server.

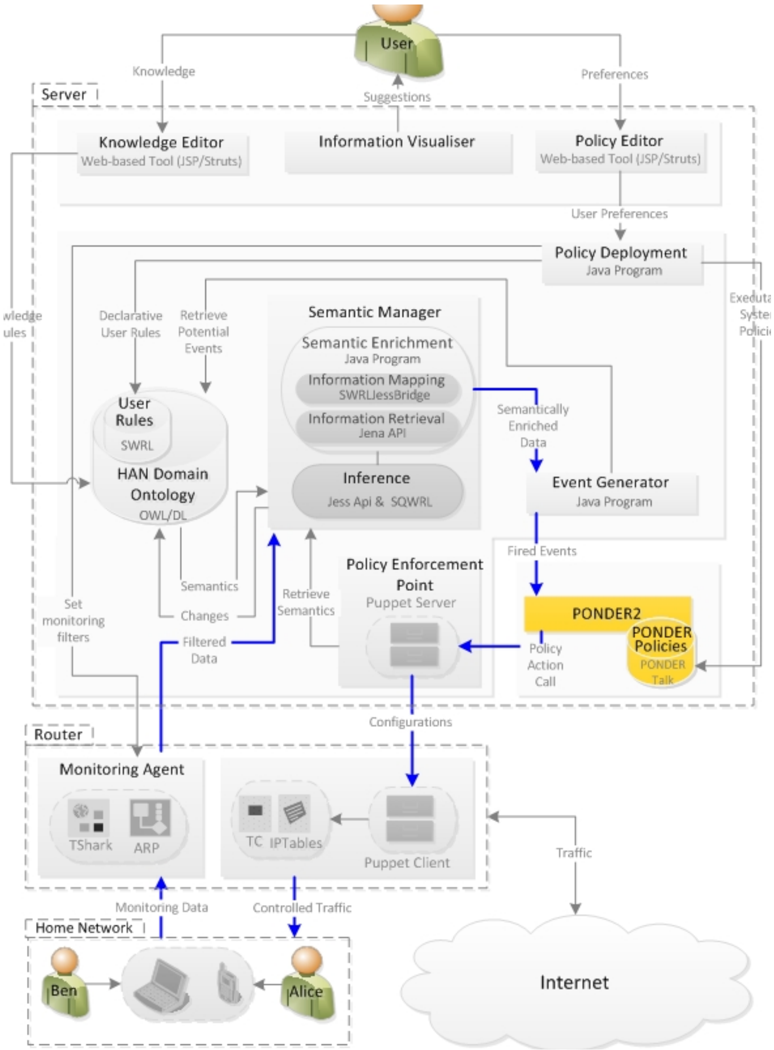


Figure 5.7: HANmanager Test-bed - Illustrating the technology and equipment used for setting up the test-bed. The *semantic manger* component adds a layer of abstraction between home network users and *HAN* infrastructure, hiding the manageability complexity of *HAN* devices and services from the typical home network users. The policy-driven router acting as a controller gateway between *HAN* (users, devices and applications) and the Internet. The arrow connectors show the implementation of *HAN* control loop and the orange coloured subunits of framework are the plug-and-play third-party components.

On the client machines, we used the Web Traffic Generator [Technologies, 2007] tool to generate background TCP web traffic; the Traffic Emulator [Kankanyan, 2009] tool to generate background UDP traffic, and XLite [Xlite, 2006] is used to make VoIP calls. The server executed the GUI web editors to allow users specifying the policy and knowledge rules. For the implementation of policy and knowledge editors, we developed JSP [Oracle, 2007] based web interfaces (using the STRUTS [Apache, 2006] framework). The knowledge is tied in synchronously with the *OWL-DL* [W3C, 2004]-based *HAN* domain ontology via an *OWL*-API [Horridge and Bechhofer, 2011] based implementation. The *HAN* domain ontology is constructed using Protégé [Gennari et al., 2003] tool and the ontology is traversed using *OWL*-API and JENA API [Carroll et al., 2004]. Semantic Web Rule Language (*SWRL*) [Horrocks, 2011] is used as a language for specifying user rules in the *HAN* domain ontology with the help of SWRLJessBridge [University, 2010]. Inference over the *SWRL* rules is achieved via Jess[1] reasoner. Finally, for ontology queries, we use the Semantic Query-Enhanced Web Rule Language (SQWRL) [Connor and Das, 2009].

We used PONDER2 [Twidle and Lupu, 2007] policy system to maintain system policies. PONDER2 facilitates with a general-purpose policy management system with variety of policy types based on *ECA* rule structure [Liu, 2009]. This makes easier for us to translate user defined rules to PONDER policies. The PONDER policies are generated from the populated policy rule template in the *HAN* domain ontology with the help of simplified CIM [de Vergara et al., 2005] based policy model that is used as policy-semantics meta model. When the PONDER policies are triggered, the device specific configurations files are generated and enforced on the the router using the Puppet framework [Loope, 2011]. The Puppet framework implements PEP (Policy Enforcement Point) to enforce configurations on the router. The semantic enrichment, policy processing and policy translation algorithms are implemented using Java.

On the router, we used deep packet inspection technique for network monitoring using TShark [Orebaugh et al., 2006] and ARP [Plummer, 1982] applications. IPtables [Purdy, 2009] are used for generating device specific configurations and implementing IPv4 NAT; tc-ng[2] is used for implementing QoS traffic control (three levels quality of service based on service type); tcpdump [Fuentes and Kar, 2005] is also used for capturing monitoring data for subsequent analysis; Perl-based scripts are used for monitoring the traffic queues and generating descriptive statistics; and bash shell scripts are used to manage the configuration. In this chapter, we mainly focus on the inferred data monitoring.

[1] `http://www.jessrules.com/`
[2] `http://tcng.sourceforge.net/`

5.5 Evaluation

To demonstrate the power of our semantic enrichment technique, we first present three example scenarios that have been realised using the *HANmanager* test-bed presented in §5.4. In the second part of this section, we present quantitative experimental results, which compare the accuracy of our Semantic Enrichment (*SE*) approach over a semantic search technique termed as Keyword Interpretation (*KI*) [Tran et al., 2007]– a popular lexicon-based semantic search algorithm.

5.5.1 Test Scenarios

For the test scenarios, we assume there are three users in a *HAN* with names: Ben (father), Jenny (mother), Tom (son). Ben and Jenny manage the network and indicate preferences for when and where Tom and guests access the Internet.

5.5.1.1 Test case 1: Identifying unknown devices

Let us assume, there is a guest named Alice, who is visiting Ben's family for a weekend and she would like to access the Internet via her smart phone. Ben has already set-up policy and knowledge rules using the HANmanger editors as follows:

```
Device(?x)^Unknown(?x)^Guest(?y)
^isGuestPresent(?y,false)->hasDeviceAccess(?x,false)
(Description: if guest is absent, unknown devices have no
 access to the Internet and HAN.)
                                                    ... R1

Device(?x)^Unknown(?x)^Guest(?y)
^isGuestPresent(?y,true)->hasDeviceAccess(?x,true)
(Description: if guest is present, unknown devices have
access to the Internet and HAN.)
                                                    ... R2

Device(?x)^Unknown(?x)^Guest(?y)
^isGuestPresent(?y,true)->hasDevice(?y,?x)
```

```
(Description: if guest is present, unknown devices any
unknown device belongs to Guest.)
                                                                 ... R3

Device(?x)^Guest(?y)^hasDevice(?y,?x)->hasPriority(?x,1)
(Description: Any device that belongs to Guest
has highest priority.)
                                                                 ... R4
```

Whenever an unknown device attempts accessing HAN, the *HANmanager* checks if this device specifications (MAC address) are available in the *HAN* domain ontology. If the device does not exist in the domain ontology, it is added under the unknown devices category. Due to policy rule *R1*, the HANmanger changes the device access to the Internet and *HAN*.

Suppose on the arrival of Alice, Ben changes the status of the guest from absent to present in the ontology using a *knowledge editor*. When Alice accesses the network using her smart phone, the *HANmanager* adds Alice's phone under the unknown device category as the device does not exist in the *HAN* domain ontology. However, due to the inference based on rule *R3* and *R2*, Alice's smart phone is granted access automatically to the Internet and the network traffic generated by her device gets highest priority.

At the network level, when user rules are specified using the editor, the *HANmanager* picks up the information about managed events entities (`Guest`, `Device`) from the user rules (*R1*, *R2* and *R4*) and related data properties (`isGuestPresent`, `hasDevice`) that are set to be monitored. For rule *R1* and *R2*, all instances of guest entity are monitored for the data property presence; for *R4*, all devices are monitored. The list of managed events is saved in the *HAN* domain ontology under the policy information model and user policy rules are translated to PONDER policies; for instance the PONDER policy for *R1* is:

```
newdom := root/factory/domain.
newecapol := root/factory/ecapolicy.
newevent := root/factory/event.
root/event at: "event1" put: ( newevent create: #( "isPresent" )).
root at: "ecadom" put: newdom create.
```

```
root/ecadom at: "ecapol1" put: ( (newecapol create)
event: root/event/event1;
condition: [:isPresent| isPresent == "false"];
action: [ root/devices/Router setUnknownaccess: 0.];
self).
root/ecadom/ecapol1 active: true.
```

Upon Alice's arrival, new knowledge is produced by the inference engine and after semantic enrichment of inferred data and related events are fired that trigger PONDER policies. On a PONDER policy triggering, the related user policy rule is translated to IPTable rules (device specific configurations) and enforced on the gateway router. The IPTable rule for *R1* is given below:

```
iptables -A OUTPUT -s 192.168.22.1 -j DROP
iptables save
```

Here Ben only had to include presence of a guest, everything else including IP address of Alice's device as inferred automatically.

5.5.1.2 Test case 2: Time and location based policies

Suppose, after 8:00 pm, Tom has no permission to use the Internet from his bedroom. However, from the communal area, he can access the Internet but not after 10:00 pm. If we assume that the household contains two access points (one in the bedroom area, the other in the communal area) then the access point through which a user connects can provide a coarse form of location detection. The following rules can be applied in this scenario:

```
User(?x)^Location(?y)^Time(?z)^hasLocation(?x,?y)^hasAccessTime(?x,?z)
^swrlb:stringEqualIgnoreCase(?y,"Room2")^swrlb:greaterThan(?z,"T20:00")
->hasAccess(?x,false)
(Description: if location is room2 and current time is greater than
20:00, then Tom has no access to the Internet.)
                                                                 ... R1
```

```
User(?x)^Location(?y)^Time(?z)^hasLocation(?x,?y)^hasAccessTime(?x,?z)
^swrlb:stringEqualIgnoreCase(?y,"Room3")^swrlb:greaterThan(?z,"T22:00")
->hasAccess(?x,false)
(Description: if location is room3 and current time is greater than
22:00, then Tom has no access to the Internet.)
                                                              ... R2
```

At the network level, the *HANmanager* picks up information about managed events from the rules and saved them in the *HAN* domain ontology. For rule *R1*, the user name is monitored along with its Internet access time and location. A related PONDER policy is generated and set active in the PONDER as shown below:

```
newdom := root/factory/domain.
newecapol := root/factory/ecapolicy.
newevent := root/factory/event.
root/event at: "event3" put: ( newevent create: #( "name" "location"
"time" )).
root at: "ecadom" put: newdom create.
root/ecadom at: "ecapol3" put: ( (newecapol create)
event: root/event/event3;
condition: [:name :location :time| name == "Tom" location=="Room2"
time>2000];
action: [ root/devices/Router setUseraccess: 0.];
self).
root/ecadom/ecapol3 active: true.
```

At the device level, when a related managed event is fired (either Tom accesses Internet from his room after 8:00 pm or from communal area after 10:00 pm), related PONDER policies are triggered. The managed event related user policy rules are translated to generate IPTable rules, and enforced on the gateway router. IPTable rule for *R1* is given below:

```
iptables -A OUTPUT -i eth3 -o eth1 -s 192.168.22.2 -m time
--timestart 20:00 --timestop 08:00 -j DROP
```

5.5.1.3 Test case 3: Application specific policies

Ben is making a VoIP call and Jenny is watching a movie on-line. The VoIP quality is quite poor as most of the network bandwidth has been taken by the video stream. If the system has following policies in the *HAN* domain ontology:

```
VoiceApplication(?x)^hasPort(?x,"sip")
^isActive(?x, true)-> hasBandwidth(?x,"High")

VideoApplication(?x)^VoiceApplication(?y)^hasPort(?x,"rtp")
^isActive(?x, true)^isActive(?y, true)->hasBandwidth(?x,"Medium")

WebApplication(?x)^VideoApplication(?y)^VoiceApplication(?z)
^hasPort(?x,"http")^isActive(?x, true)^isActive(?y, true)
^isActive(?z, true)->hasBandwidth(?x,"Low")

VideoApplication(?x)^VoiceApplication(?y)^hasPort(?x,"rtp")
^isActive(?x, false)^isActive(?y, true)->hasBandwidth(?x,"High")

WebApplication(?x)^VideoApplication(?y)^VoiceApplication(?z)
^hasPort(?x,"http")^isActive(?x, true)^isActive(?y, true)
^isActive(?z, false)->hasBandwidth(?x,"Medium")

WebApplication(?x)^VideoApplication(?y)^VoiceApplication(?z)
^hasPort(?x,"http")^isActive(?x, true)^isActive(?y, false)
^isActive(?z, false)->hasBandwidth(?x,"High")

(Description: if applications being accessed are VoIP,
video stream and web, then give highest priority to VoIP, medium
priority to video stream and low priority to web and ftp up/download.)
                                                             ... R1-R7
```

The required bandwidths are allocated to the active applications based on above specified rules. The *HANmanager* monitors network and makes adjustments whenever it is required e.g., re-allocating maximum of bandwidth for video when VoIP call ends. At the network level, the *HANmanager* picks up information about the managed events from the specified user rules; for all the rules above, applications' ports are monitored and related PONDER policies are generated and set active in PONDER. The following are the PONDER policies for VoIP, Video and Web traffic:

```
newdom := root/factory/domain.
newecapol := root/factory/ecapolicy.
newevent := root/factory/event.
root/event at: "event4" put: ( newevent create: #( "port" )).
root at: "ecadom" put: newdom create.
root/ecadom at: "ecapol4" put: ( (newecapol create)
event: root/event/event4;
condition: [:port | port == "sip"];
action: [ root/devices/Router setAppBandwidth: "sip" "High".];
self).
root/ecadom/ecapol4 active: true.

newdom := root/factory/domain.
newecapol := root/factory/ecapolicy.
newevent := root/factory/event.
root/event at: "event5" put: ( newevent create: #( "port" )).
root at: "ecadom" put: newdom create.
root/ecadom at: "ecapol5" put: ( (newecapol create)
event: root/event/event5;
condition: [:port | port == "rtp"];
action: [ root/devices/Router setAppBandwidth: "rtp" "Medium".];
self).
root/ecadom/ecapol5 active: true.

newdom := root/factory/domain.
newecapol := root/factory/ecapolicy.
newevent := root/factory/event.
root/event at: "event6" put: ( newevent create: #( "port" )).
root at: "ecadom" put: newdom create.
root/ecadom at: "ecapol6" put: ((newecapol create)
event: root/event/event6;
condition: [:port | port == "http"];
```

```
action: [root/devices/Router setAppBandwidth: "http" "Low".];
self).
root/ecadom/ecapol6 active: true.
```

At the device level, when a managed event is fired, related PONDER policies are triggered and related user policy rules are further translated to IPTable rules. Following are TC and IPTable rules for VoIP, Video and Web traffic:

```
tc class add dev eth1 parent 1:1 classid 1:10 htb rate 1Mbps ceil 2Mbps
 prio 1
tc class add dev eth1 parent 1:1 classid 1:11 htb rate 500kbps ceil
1Mbkbps prio 2
tc class add dev eth1 parent 1:1 classid 1:12 htb rate 200kbps ceil
500kbps prio 3

tc qdisc add dev eth1 parent 1:10 handle 20: sfq perturb 10
tc qdisc add dev eth1 parent 1:11 handle 30: sfq perturb 10
tc qdisc add dev eth1 parent 1:12 handle 40: sfq perturb 10

tc filter add dev eth1 parent 1:0 prio 1 protocol ip u32 match ip tos
0x28 0xff classid 1:10
tc filter add dev eth1 parent 1:0 prio 2 protocol ip u32 match ip tos
 0x48 0xff classid 1:11
tc filter add dev eth1 parent 1:0 prio 3 protocol ip u32 match ip tos
0x68  0xff classid 1:12

iptables -t mangle -I FORWARD -o eth1 -p udp --sport sip -j TOS
--set-tos 0x28
iptables -t mangle -I FORWARD -o eth1 -p udp --sport rtp -j TOS
--set-tos 0x48
iptables -t mangle -I FORWARD -o eth1 -p udp --sport http -j TOS
--set-tos 0x68

iptables save
```

5.5.2 Experimental Results

In this section, we report and discuss the observed results for above mentioned test cases using our Semantic Enrichment (*SE*) approach in comparison with the Keyword Interpretation (*KI*) approach presented by Tran et al. [2007]. The *KI* algorithm translates keyword queries to *DL* conjunctive queries using background knowledge available in ontologies. It uses keyword interpretation for exploring asserted knowledge and for a semantics-based declarative query answering process. However, the *KI* approach does not take into account the complex hierarchy of relationships among different ontological concepts/entities; for example, object relationships of parent classes for a child entity, which eventually results in less rich semantics. Instead it uses direct entity relationship (object properties) of mapped instance to the keyword. However, entities can be linked together in a more complex manner sometimes through indirect relationships—object properties inherited from parent classes. Therefore, the *KI* algorithm drops many relevant entities in the search algorithm that are indirectly linked. Another difference to our *SE* approach is that the *KI* algorithm searches keyword semantics to build *DL* query; in contrast *SE* uses dynamic *DL* queries to build semantic graph in progressive manner, giving a comprehensive semantics search.

In Table 5.3, we show different levels of qualitative performance of both algorithms using search depth 1 and search scope 1 (“user view”). We initially observed the results for *KI* algorithm for above mentioned test cases and found some indirect relationships are missing in the final results causing failure to trigger relevant policies. On the other hand, *SE* returns sufficient semantics of inferred data that result in triggering the correct policies.

Table 5.3: Comparison of Keyword Interpretation (KI) and Semantic Enrichment (SE) - A qualitative evaluation of both semantic search algorithms (in relation to triggering of system/ponder policies) using search depth 1 and search scope 1 parameters.

	Test Case 1		Test Case 2		Test Case 3		Test Case 4	
	KI	SE	KI	SE	KI	SE	KI	SE
No. of Entities	1	2	1	3	1	4	1	2
Policy Triggered	No	Yes	No	Yes	No	Yes	Yes	Yes

Table 5.4: Comparison of Basic Keyword Interpretation (KI), Enhanced KI (EKI) and Semantic Enrichment (SE) Algorithms - Another qualitative evaluation of three semantic search algorithms (in terms of effectiveness) using different graph depth and search scope parameters.

		KI			EKI			SE		
	Depth	R1	U1	M1	R2	U2	M2	R3	U3	M3
Scope 1	1	1	0	2	3	2	0	3	0	0
	2	1	0	2	3	4	0	3	0	0
	3	3	0	1	3	4	0	3	0	0
Scope 2	1	1	0	4	5	2	0	5	0	0
	2	1	0	4	5	2	0	5	0	0
	3	3	0	1	5	2	0	5	0	0
Scope 3	1	1	0	6	7	0	0	7	0	0
	2	1	0	6	7	0	0	7	0	0
	3	3	0	3	7	0	0	7	0	0

R:Related,U:Unrelated, M:Missing

We modified the *KI* algorithm (by adding capability to integrate indirect relationships in semantic search graph) to observe the effects of search depth and search scope parameters. The Enhanced Keyword Interpretation (*EKI*) algorithm traversed the ontology iteratively to a specified depth level. We applied all three algorithms to a larger, multi-domain *HAN* ontology. The results are tabulated in Table 5.4, which shows the number of related, unrelated and missing entities (within a search scope) for each of the algorithm with different depth and search scope parameters. We observed that the basic *KI* algorithm did not return expected results due to its inability to recognise the complex relation hierarchy of entities; thus there are high number of missing entities in semantics search result. The *EKI* algorithm overcame the issue of missing entities but returned a high number unrelated entities in the semantics search result, which are not related to the defined search scope. Once again *SE* algorithm provided the appropriate search results, without missing any related entity or providing unrelated entities in its results.

The *SE* algorithm found the semantics of inferred data within the defined search scope. By limiting the search scope, it processed entities more efficiently. With search scope 1, the *SE* algorithm searched the semantics within the "user view". The "user view" search scope is used for finding the semantics of inferred data because we assumed that inferred knowledge is related to high-level entities (knowledge rules are specified by home network users through the knowledge editor, which only deal with "user view" entities). Similarly, the *SE* algorithm searched in the "device view" for the translation of user policy rules. With the *EKI* algorithm it is not possible to limit the scope of semantics search, therefore causing extra processing of unrelated entities in ontology graph. It is also notable that both *EKI* and *SE* performed equally (in terms of time and memory consumption) when search scope is set to 3.

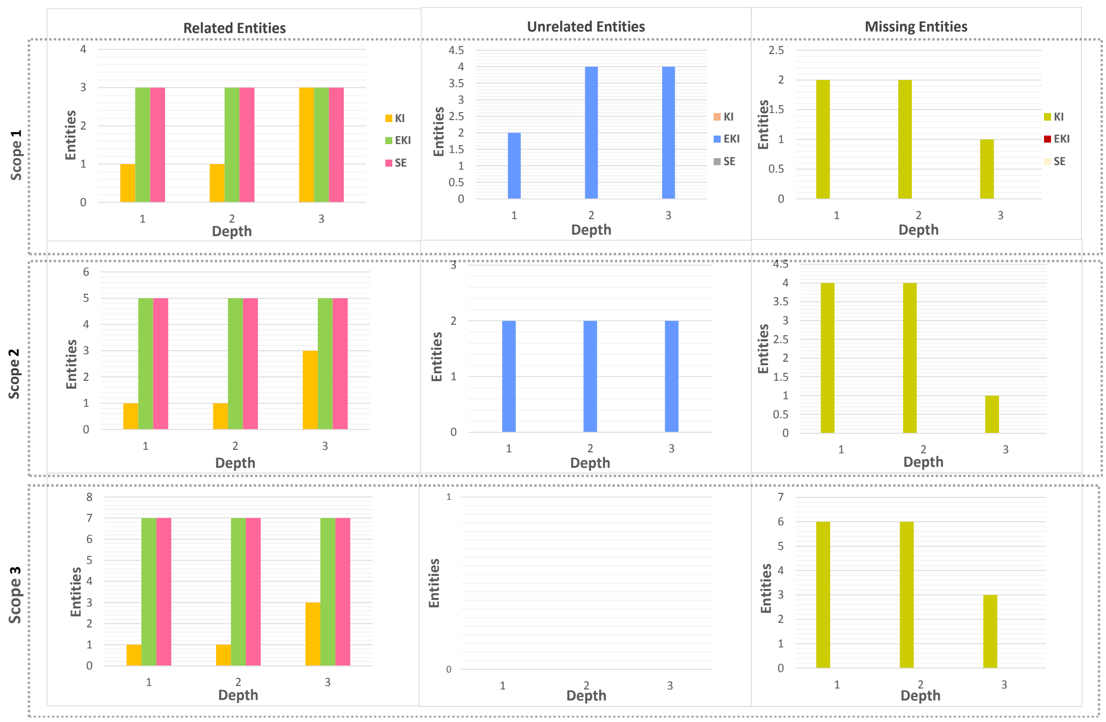


Figure 5.8: Multi charts showing the power of Semantic Enrichment algorithm over Keyword Interpretation and Enhanced Keyword Interpretation algorithms for the retrieval of entities from ontology in terms of related, unrelated and missing entities.

The measurements presented in Tables 5.3 and 5.4 are subjective and limited, yet indicative of the degree of significance and efficiency in retrieval and processing of semantics. Moreover, the experiments also showed that by defining search scope at different levels, relevance of searched entities is higher for Semantic Enrichment algorithm as showin in Figure 5.8.

5.6 Summary

In this chapter, we presented the *HANmanager*, a framework for management of home area networks that incorporates generic techniques for semantic enrichment of inferred data from *HAN* domain ontology, and processing of selected policies based on extracted information from the inferred data. The *HANmanager* alleviates ordinary, non-technical home network users from specifying complex system policies and writing configuration scripts to set-up their network devices. Our policy processing technique automatically enforces user-defined preferences, and facilitates ease-of-configuration for the ordinary users without requiring them to understand the network system and network management processes. Our semantic enrichment algorithms are shown to retrieve the semantics better compared to the other popular semantic retrieval techniques. Thus we have addressed the list of challenges that we presented earlier in this chapter.

Chapter 6

Policy Translation using Ontology-based Meta Model

In user driven complex control systems, translating user rules is desirable if a system needs to map user objectives to system configurations. Considering policies as rules, the process of policy translation is difficult due to the inherent complexity of transforming abstract concepts into something concrete and entirely meaningful to a system. Formal semantic models can harness this translation process by connecting the concepts at different abstraction levels but existing policy languages do not provide semantic models that capture concepts' semantics embodied in policy languages.

In this chapter, we attempt to explain the process of semantic translation of user-defined abstract rules. It is important to note the basic definition of policy translation used in this thesis report before we proceed any further. We consider semantic-aware policy translation as a process of mapping concepts of a policy language in one domain or a sub-domain to the concepts of another policy language in another domain or sub-domain, with an assumption that the information in both source and target policy languages and their domains or sub-domains are related to each other in some manner. The inferred information that are extracted through the translation technique is later used applying user policies in the form of system configurations to manage Home Area Networks (*HANs*). The presented technique manipulates a *HAN* domain model that is represented in an ontological form, for translation of user policies to system configurations. The *HAN* domain model is a knowledge base that contains the information of different entities of domain (users, devices, and applications etc.) in a structural manner. The ontology-based domain model is core of our technique for policy translation; further details can be found in Chapter 3.

The main contribution of this chapter is the extension of the semantic translation algorithm, Usage and Change Control (UCC) [Barret, 2009], which is used to map policy concepts in policy languages to policy meta-model concepts and meta-model helps in translation of a policy language to another policy language that also uses the meta-model.

6.0.1 Introduction

There exists a gap between the users and their actual network systems. This gap can be explained in terms of lack of understanding of the systems by their users. If user and HAN systems represents two domains, the gap can be filled by determining how concepts within the two domains or sub-domains coexist and relate to each other. Domain modelling, as explained in Chapter 3, is a method used for capturing and representing the domain concepts. It provides an articulation of the meaning behind concepts, which are present in a domain. The resulting domain model acts as a formalised context behind the nature of the elements in the working system based on real world concepts.

Lack of formal policy semantic models for existing policy languages is one of the major hurdles in translation of policies for a policy-based HAN management. Particularly, the translation of declarative policies to executable policies with respect to the policy continuum [Davy et al., 2008] is extremely difficult. The policy continuum has different policy definition and abstraction levels with respect to a network management system. The declarative policy languages at high-levels of network systems are usually abstract in nature and therefore employ a flexible and uncomplicated syntax model to incorporate a wider domain of application [Damianou et al., 2001]. By contrast, the executable policy languages used to configure network devices are often domain specific with concrete syntax format and limited application. Policy translation is complex due to the missing connections between abstract and concrete policies. To simplify policy translation, we consider two levels of policy abstractions; user-level and device-level. The policies at user-level are mostly declarative and the policies at device level are executable.

In this chapter, we use a simplified version of a policy model (e.g. DEN-ng [Jennings et al., 2007] and *CIM* [de Vergara et al., 2005]) to link two different policy languages defined at different abstraction levels. We used *SWRL* [Horrocks, 2011] as high-level policy language to define user requirements and translated them to *IPTables* rules with the help of an ontological model. The ontological model contains the syntax, semantic and domain models of both policy languages. The generated policies, using the ontological model, aimed to classify *HAN* traffic in different priority queues to improve user experience. We used the Puppet framework [Turnbull, 2007] to enforce policies on a *HAN* gateway that manages the *HAN* devices and the services.

The major contribution of this chapter is the extension of Usage and Change Control (*UCC*) algorithm [Barret, 2009]. The *UCC* algorithm is initially proposed to define semantic mapping of policy languages and their translation. However, the employed policy languages have to be of equal abstraction levels otherwise *UCC* fails the viability step (see [Barret, 2009] for further details). We extended the *UCC* algorithm to address this problem and used the algorithm for translation of policy languages of different abstraction levels.

6.1 Policy Translation Technique using Meta Transition

In this section, we give an overview of Policy Translation technique for transforming user defined rules into system configurations for *HAN* management. A subset of the *HANmanager* framework, used for policy translation, is shown in Figure 6.1. The framework works as follows: when the *policy system* fires the policies to be executed, the *policy manager* takes the user defined policy and by using the semantic graph and policy meta-model, it translates the policy into system configurations. The generated configurations are applied on the target system or device.

Policies play an imperative role in rule-based *HAN* management as they can formalise the concepts of rules and decision making. The policies at each level of *HAN* system may at first appear disparate in syntax but they can be linked via semantics [Barret, 2009]. However, variations in syntax and their applications have made translation extremely complex. The translation process requires four major knowledge components to link policy concepts in source and target policy languages:

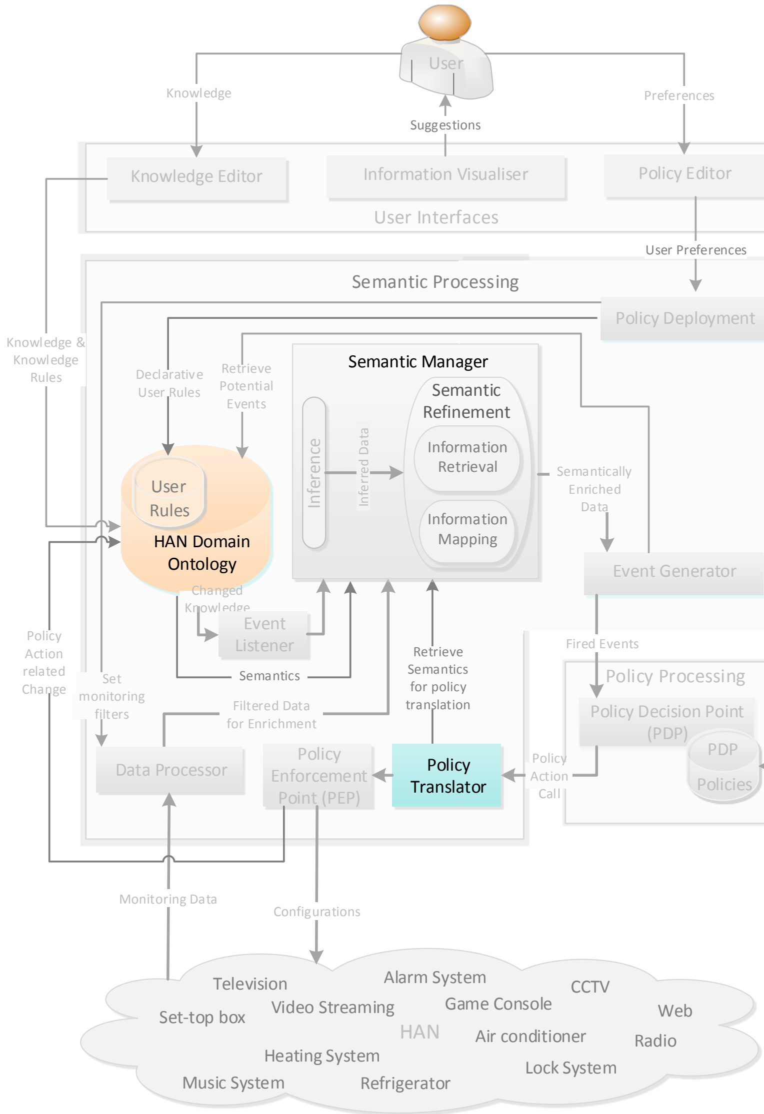


Figure 6.1: The HANmanager Framework - Highlighting the role of Policy Translator transforming user defined policies into system configurations. Policy Translator uses a policy meta model to translate a policy in a policy language to another.

1. domain knowledge of entities in policy domains;
2. syntactic knowledge of policy languages;
3. semantic knowledge of policy concepts embodied in policy languages and entities in the domains;
4. pragmatic knowledge of policy concepts in relation to entities in respective domains.

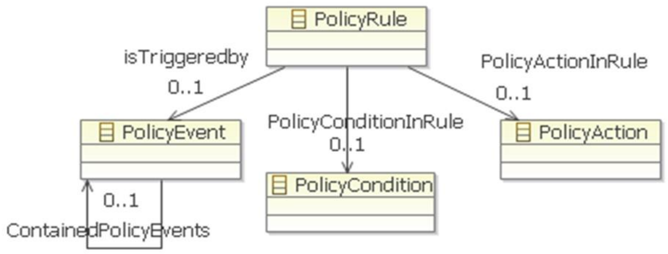


Figure 6.2: Simplified Policy Model - Showing the structure of policy model that is used as a meta model to retrieve policy related semantics for the *HAN* domain entities. In the diagram, a user preference is saved in the policy model in the form of a declarative policy rule using the "user view" entities of *HAN* domain ontology.

The domain knowledge specifies entities roles, properties and their relationships with each other. A domain defines the entity model, where source and target policy languages can be applied. In our experiments, we use user, network/system and device as three policy domains, each domain having its own policy language. However, we used a simple syntactic translation of user level policy to network/system level policy (to make user policy executable when relevant event occurs) and semantic translation of user level policy (when is executed with the help of system/network level policy) to device level policy. In this chapter, we only focus on semantic translation of user level policy to device level policy that would require syntax and semantic knowledge. The syntactical knowledge defines the grammar rules for a policy language. The semantic knowledge defines meaning of policy concepts and entities in policy domains. The pragmatic knowledge defines the rules governing policy concepts in relation to the entities involved in policy domains. In our approach, we used an OWL-DL based ontology to define syntactical, semantic, pragmatic, and domain knowledge for source and target policy languages. To relate policy concepts of two different policy languages, we require a semantic model that contains semantics for the vocabulary used in policy languages and definition of how policy concepts are interrelated. The policy information models (e.g. *CIM*, SID, DEN-ng) can be used as semantic models to formulate policy concepts from the domain. This chapter argues for the use of a similar technique to map abstract policy concepts (in one domain or sub-domain) to concrete concepts (in another domain or sub-domain) by using semantic models. The *CIM* policy rule model is illustrated in Figure 6.2. The PolicyRule class defines the "event-condition-action" semantics that form a *CIM* policy rule. Following example shows that how *CIM* policy model can be used to define semantics of policy concepts in a policy language.

Let us take a simple example of two network traffic flows "x" and "y". The *HAN* user wants to give high priority to traffic "x" over traffic "y".Representing the user requirement in *SWRL*, using the concepts and properties of *HAN* domain model, will look like as shown in (a):

```
NetworkService(?x) ^ hasLevel(?x ,?a ) ^ hasPriority(?a , ?b ) -> hasPriority(?x ,?b)
```

... 1

The semantic representation of policy rule (a) in a simplified *CIM* ECA format is shown in (b):

```
Event: NetworkService(x)
Condition: hasLevel(a)
Condition: hasPriority(b)
Action: hasPriority(b)
```

In a similar fashion, the semantics of policy concepts in the *IPTables* language can be defined in terms of *CIM* policy model. The proposed policy translation technique in this chapter uses a sub-set of the *CIM* policy model in ontological form. The translation process involves mapping of high-level and low-level policy concepts to *CIM* and it leads to mapping of high-level policy concepts in the source language to low-level policy concepts in target language. This technique is called meta-transition and is shown in Figure 6.3. The policy concepts in *SWRL* and *IPTables* are represented in ontological form and mapped to *CIM* policy model for semantic translation. The mappings information of *SWRL* and *IPTables* policy concepts to *CIM* policy model is further utilized to discover connections between both policy languages. The technique is explained below:

1. Define ontological models for source and target policy languages in one domain ontology containing syntax, semantics, and domain entities;

2. Define high-level requirements into *SWRL* rules and apply them on domain ontology;

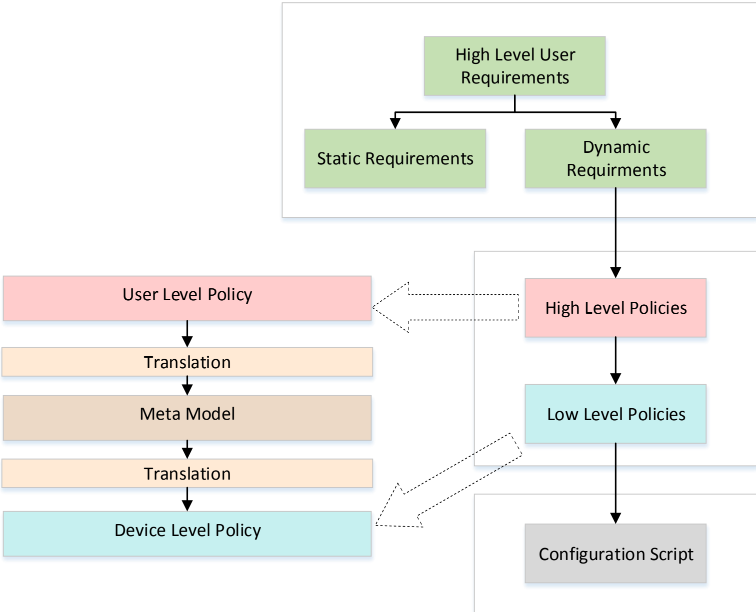


Figure 6.3: Policy Translation - Translation of User Requirements in the form of High Level Policy Language to Device Configuration with the help of Meta Model

3. Execute the mapping algorithm to map high-level and low-level policy concepts to meta-model for semantic definition;
4. Execute the discovery algorithm to determine the relations among high-level and low-level policy concepts;
5. Generate the low-level policies and parse them into network configuration.

6.2 Usage and Change Control Algorithm

This section gives an overview of Usage and Change Control (*UCC*) algorithm [Barret, 2009] for meta-transition technique. Typically, policy translation approaches concentrate exclusively on the syntactical translation of policy. Consequently, the motivation for *UCC* algorithm is to assist in the realisation of semantic-aware policy translation. The *UCC* algorithm is not coupled with any particular syntactical translation approach. Instead, it can be executed to augment any existing or future syntactical translation approach. Table 6.1 presents abbreviations and their meaning used in *UCC* and extended *UCC* (*EUCC*) algorithms.

Table 6.1: Algorithmic Abbreviations

Abbreviations	Explanation
PL	PolicyLanguage
PLO	PLOntology
PLC	PLConcept
PLP	PLProperty
PLI	PLIndividual
PLOR	PLORepository
OMR	OntologyMappingRepository
PLIO	PLInterlinguaOntology (Meta-Model)
MAP	Mapping of PLC1 to PLC2.
MMD	MapMetadata
OMD	OntologyMetadata
MF	MappaingFormat
IO	Interlingua Ontology
IOS	Interlingua Ontology Syntax
OF	Ontology Format
IOF	Interlingua Ontology Format
KRF	Knowledge Representation Format

6.3 Semantic Translation Algorithm

The existing ontological mapping techniques do not support mapping between the concepts at different abstraction levels and therefore we propose a meta-transition technique that translates abstract policies to concrete policy policies. We use the ontological concept matching and mapping modules of the Usage and Change Control (*UCC*) algorithm [Barret, 2009] for meta-transition technique. The *UCC* algorithm is initially proposed for semantic translation of policies at the same abstraction level. We extend the *UCC* algorithm to map policies of different abstraction levels. Initially, we develop a domain ontology containing policies concepts for *SWRL*, *IPTables*, *HAN* topology and Linux applications (*IPTables* and Traffic Control) but later we redesign the ontology to keep policies separate from the domain ontology.

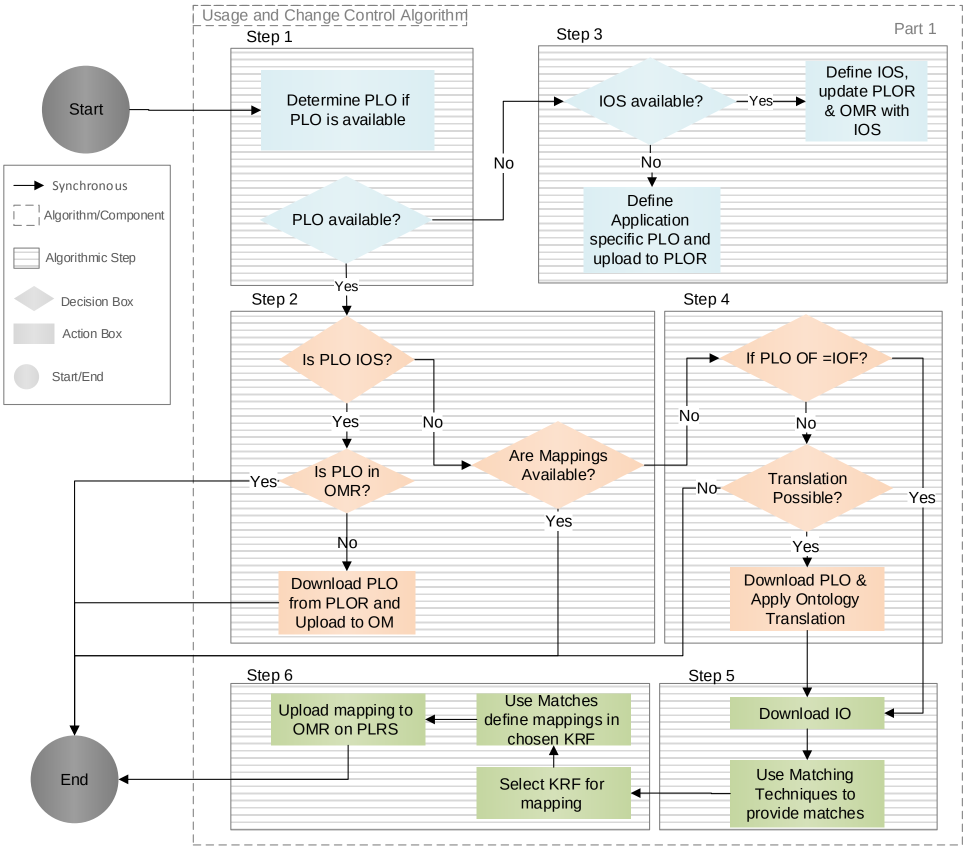


Figure 6.4: Usage Change and Control Algorithm Part 1 - Showing the different steps of UCC algorithm.

The *UCC* algorithm is divided into 2 parts; part 1 is responsible for ensuring that an ontology that defines the semantics of the policy concepts embodied in a particular policy language exists and that mappings from the policy language ontology to the Interlingua ontology (policy semantic model) are established correctly. Part 2 is responsible for determining if semantic translation from a source policy language to a target policy language is viable. Part 2 is independent of Part 1 and vice versa. Therefore, *policy manager* aiming to determine if the semantic translation policy language concept is viable may commence the investigation at Step 1 in Part 2. However, we only explain the part 1 that is extended and used for semantic translation algorithm and assumed that policy translation is viable. The part 1 of *UCC* algorithm is shown in in Figure 6.4. A meta-model is defined manually by adding object and data properties to link *SWRL* and *IPTables* policy concepts in the domain ontology.

Table 6.2: Algorithmic Notations

Notation	Explanation
$a \in S$	Element a belongs to S
$S \cup a$	Set S union a
$S \cap a$	Set S intersection a
PT	Power Set T
$b : PT$	b is element of PT
$\exists! i \in S,\ i-1=0$	There is exactly one i, which belongs to S where i-1=0
$\exists i \in S,\ i-1=0$	There exists one or more i, which belongs to S where i-1=0
$\forall i \in S,\ i-1=0$	For all i, which belongs to S where i-1=0
$(R \times S \times T)$	A tuple consisting of an entity from each set specified
$funct : (A) \doteq (B)$	A function specification, the parameter is from the set A and the result if from the set B
$(f \circ g)(x)$	Apply f after g to x
aLS	It behaves like set union but it is only defined for disjoint pairs of sets

Our initial meta-model that we designed, lacked a standardized approach for linking the policy concepts and then we used a subset of *CIM* policy model as a meta-model. The *CIM* policy model defines all the elements of a policy language and their relations to each other in a hierarchical manner. In this chapter, we present the new and modified modules of *EUCC* as shown in Algorithms 6, 7 and 8. Table 6.2 provides the explanation of notions used in *EUCC* algorithm. We used the relational hierarchy of ontology to determine the connections between abstract policy concepts in *SWRL* (a user level policy) and concrete policy concepts in *IPTables* (a device level policy).

The extended *UCC* (*EUCC*) algorithm defines an approach to map policy concepts to meta-model concepts based on concept-type attributes. The extended *UCC* algorithm matches concepts type using a set of attributes. The concept-type attributes can be added as data properties for all policy concepts of source and target policy languages in the domain ontology. Unfortunately, existing ontology tools do not support matching of concepts based on attribute value. Therefore, *UCC* or *EUCC* algorithms are not currently supported by any ontology mapping tool. Nevertheless, manual mappings can be defined using the Protégé PROMPT plug-in[1].

[1] `http://protegewiki.stanford.edu/wiki/PROMPT`

6.3.1 Define Type Module

Algorithm 6 Semantic Translation Algorithm: Define Type Module

function $defineSameType/DiffType\ (PLO_1 \times PLIO_1)$
$DiffType_1 = idDiffType(PLO_1, PLIO_1)$
$SameType_1 = createNewType()$
if $\exists i \in PLConcept.SameType_1 \cap \{i\} = null \wedge isPartOfPLOntology(i, PLO_1)$ **then**
$SameType_1 =$
$SameType_1 \cup \{\ (checkTypeAttributes(i, \forall j \in PLConcept.isPartOfPLOntology\ (j, PLIO_1)))\}$
$SameType_1 =$
$SameType_1 \cup \{(checkTypeAttributes(\forall j \in PLConcept.isPartOfPLOntology\ (j, PLIO_1), i))\}$
$uploadType\ (PLOR_1 \times SameType_1)$
$uploadTypeMetadata\ (MMD_1 \times PLOR_1)$
$return\ definedMapping$

This is the new module in the *EUCC* algorithm, which defines policy concepts in terms of the meta-model concepts based on concept type attribute. This module only covers the policy concepts and will be extended for properties and individuals. Algorithm 6 shows the module description in Vienna Development Method (VDM) notation [Bjørner and Jones, 1978].

The type attributes form a set of data and object properties associated with each entity and define the semantic information to classify the entities broadly into: event, condition, action, variable, operator, value, source, target etc.

6.3.2 Find Match or Mismatch Module

This is a modified module of the *UCC* algorithm, which finds matching policy concepts. Ontology matching involves the identification of semantically related concepts. The *EUCC* algorithm is application and implementation independent, and any concept matching technique (e.g. lexical, fuzzy matching) can be used. We used lexical matching and semantic similarity functions to match the policy concepts. Algorithm 7 shows the *UCC* find match/mismatch module.

Algorithm 7 Semantic Translation Algorithm: Find Match/Mismatch Module

function $findMatch/Mismatch/DiffType\ (PLO_1 \times PLIO_1)$
$Mismatch_1 = idMismatch(PLO_1, PLIO_1)$
$Match_1 = createNewMatch()$
if $\exists i \in PLConcept.Match_1 \cap \{i\} = null \wedge isPartOfPLOntology(i, PLO_1)$ **then**
$Match_1 =$
$Match_1 \cup \{\ (match(i, \forall j \in PLConcept.isPartOfPLOntology\ (j, PLIO_1)))\}$
$Match_1 =$
$Match_1 \cup \{(match(\forall j \in PLConcept.isPartOfPLOntology\ (j, PLIO_1), i))\}$
if $\exists i \in PLProperty.Match_1 \cap \{i\} = null \wedge isPartOfPLOntology(i, PLO_1)$ **then**
$Match_1 =$
$Match_1 \cup \{\ (match(i, \forall j \in PLProperty.isPartOfPLOntology\ (j, PLIO_1)))\}$
$Match_1 =$
$Match_1 \cup \{(match(\forall j \in PLProperty.isPartOfPLOntology\ (j, PLIO_1), i))\}$
if $\exists i \in PLIndividual.Match_1 \cap \{i\} = null \wedge isPartOfPLOntology(i, PLO_1)$ **then**
$Match_1 =$
$Match_1 \cup \{\ (match(i, \forall j \in PLIndividual.isPartOfPLOntology\ (j, PLIO_1)))\}$
$Match_1 =$
$Match_1 \cup \{(match(\forall j \in PLIndividual.isPartOfPLOntology\ (j, PLIO_1), i))\}$
$uploadType\ (PLOR_1 \times Match_1)$
$uploadTypeMetadata\ (MMD_1 \times PLOR_1)$
$return\ definedMapping$

The similarity σ between two entities i and j is a function: $o \times \acute{o} \rightarrow \Re$, which can be described as:

$$\forall i \in o, \forall j \in \acute{o}, \sigma(i,j) \geq 0\ \dots\ positiveness \tag{6.1}$$

$$\forall i \in o, \forall j \in \acute{o}, \sigma(i,i) \geq \sigma(i,j)\ \dots\ maximality \tag{6.2}$$

$$\forall i \in o, \forall j \in \acute{o}, \sigma(i,j) \geq \sigma(j,i)\ \dots\ symmetry \tag{6.3}$$

Where o and $\acute{o}$ are two ontologies. The similarity can be expressed using real numbers $\Re \rightarrow [0,1]$. The similarity rules for entities i and j are given below:

$$\forall i \in o, \forall j \in \acute{o}, \sigma(i,j) = 1 \leftrightarrow i = j\ \dots\ identical \tag{6.4}$$

$$\forall i \in o, \forall j \in \acute{o}, \sigma(i,j) < 1, \sigma(i,j) > 0 \leftrightarrow i \approx j\ \dots\ similar/dissimilar\ to\ a\ certain\ degree \tag{6.5}$$

$$\forall i \in o, \forall j \in \acute{o}, \sigma(i,j) = 0 \leftrightarrow i \neq j\ \dots\ Non\ identical \tag{6.6}$$

6.3.3 Define Mapping Module

Algorithm 8 Semantic Translation Algorithm: Define Mapping Module

function $defineMap\ (PLO_1 \times PLIO_1, Match_1, OMR_1)$
$MF_1 = selectMappingFormat(PLO_1, PLIO_1)$
$MAP_1 \wedge MAP_2 = createEmptyMap(MF_1)$
if $\exists i \in PLConcept.AttemptedPLO2PLIO \cap \{i\} = null$ **then**
$MAP_1 =$
$MAP_1 \cup \{\ (mapElement(i, Match_1, MF_1, PLOnt2InterlinguaOnt\}$
$AttemptedPLO2PLIO = AttemptedPLO2PLIO \cup i\}$
if $\exists j \in PLProperty.AttemptedPLO2PLIO \wedge \{j\} = null \wedge isPartOfPLOntology(i, PLO_1)$ **then**
$MAP_1 =$
$MAP_1 \cup \{\ (mapElement(j, Match_1, MF_1, PLOnt2InterlinguaOnt\}$
$AttemptedPLO2PLIO = AttemptedPLO2PLIO \cup j\}$
if $\exists j \in PLIndividual.AttemptedPLO2PLIO \wedge \{k\} = null \wedge isPartOfPLOntology(i, PLO_1)$ **then**
$MAP_1 =$
$MAP_1 \cup \{\ (mapElement(j, Match_1, MF_1, PLOnt2InterlinguaOnt\}$
$AttemptedPLO2PLIO = AttemptedPLO2PLIO \cup k\}$
if $\exists i \in PLConcept.AttemptedPLO2PLIO \cap \{i\} = null$ **then**
$MAP_2 =$
$MAP_2 \cup \{\ (mapElement(i, Match_1, MF_1, PLOnt2InterlinguaOnt\}$
$AttemptedPLO2PLIO = AttemptedPLO2PLIO \cup i\}$
if $\exists j \in PLProperty.AttemptedPLO2PLIO \wedge \{j\} = null \wedge isPartOfPLOntology(i, PLO_1)$ **then**
$MAP_2 =$
$MAP_2 \cup \{\ (mapElement(j, Match_1, MF_1, PLOnt2InterlinguaOnt\}$
$AttemptedPLO2PLIO = AttemptedPLO2PLIO \cup j\}$
if $\exists j \in PLIndividual.AttemptedPLO2PLIO \wedge \{k\} = null \wedge isPartOfPLOntology(i, PLO_1)$ **then**
$MAP_2 =$
$MAP_2 \cup \{\ (mapElement(j, Match_1, MF_1, PLOnt2InterlinguaOnt\}$
$AttemptedPLO2PLIO = AttemptedPLO2PLIO \cup k\}$
$uploadType\ (OMR_1 \times MAP_1)$
$uploadTypeMetadata\ (MMD_1 \times OMR_1)$
$uploadType\ (OMR_1 \times MAP_2)$
$uploadTypeMetadata\ (MMD_2 \times OMR_1)$

This is a modified module of the *UCC* algorithm, which maps one policy concept to another policy concept based on the find match/mismatch criteria. Once a mapping format has been decided, two new mapping containers will be created. One mapping container holds mappings from policy language ontology elements (concepts, properties and individuals) to the meta-model (Interlingua) ontology elements and a second mapping container holds mappings from meta-model ontology elements to policy language ontology elements. The module is explained in Algorithm 8. The mapping of concepts in two policy languages is performed on the instantiated semantic graph.

6.4 Evaluation

For the policy translation experiments, a network traffic classification test case is conducted. In a typical *HAN*, there can be several types of network traffic e.g. VoIP, Audio and Video on demand, Web and many more. Usually, the *HAN* traffic works in best effort fashion, meaning QoS is not guaranteed. The *HAN* traffic quality can suffer due to bursty traffic, which usually gets most of the network bandwidth at the expense of other network traffics. Moreover, sometimes users' applications and systems connected to *HAN* require network resources more than the network capability. In such situations no service gets satisfactory share in network resources.

This leads the network into a state of congestion, which sometimes chokes network traffic flow and results in poor quality of network services. Mostly, the solution to resolve congestion issue is to get more bandwidth for the network but logically it alleviates the issue temporarily but does not provide any long lasting remedy for healthy networking operations. The provisioned QoS is statically achieved by configuring network resources for different types of network traffic flows. Most of QoS approaches are static using priority queues, data flow control and packet marking etc. We configured provisioned QoS for *HAN* traffic. The autonomic traffic classification experiments are based on a test scenario that assumed a *HAN* user is conducting a VoIP call and other internet activity simultaneously. Without QoS management, the VoIP call quality is adversely affected (including packet loss, delay and jitter). After using policies to prioritize and classify different traffic flows, packet loss decreased to 30% and VoIP quality improved dramatically. The user policies defined using SWRL are translated to IPTables using

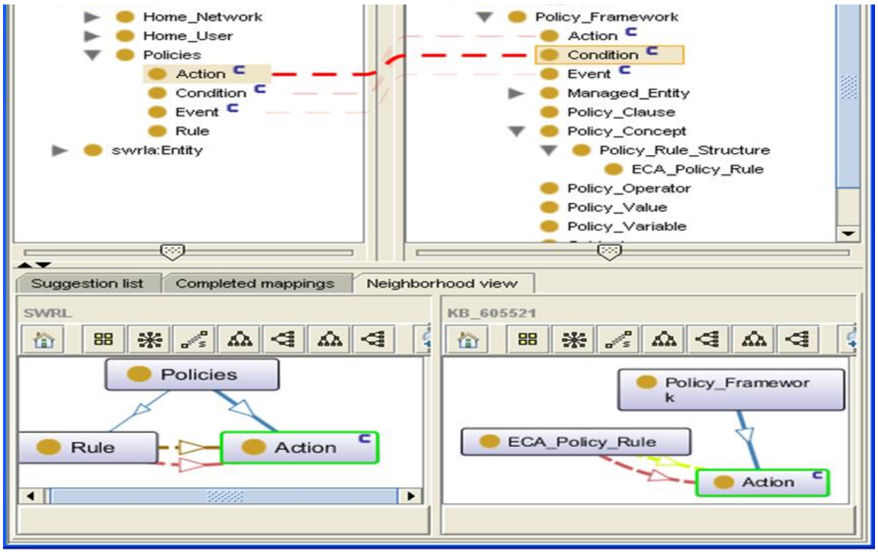


Figure 6.5: Ontological Concepts Mapping - Showing the process of concept mapping to for translation of concepts in an ontology using using PROMPT.

meta-transition technique of high-level requirements to network configurations without user intervention.

The Protégé plug-in PROMPT [Malik et al., 2010] is used to find matches between concepts represented in the domain ontology. For the convenience of experiments, we broke the domain ontology into two ontological sub-models: domain model of *HAN* and *CIM* policy model as meta model. The simulations are conducted with PROMT supported mapping techniques -e.g. UMLS [Lomax and McCray, 2004](Unified Medical Language System), Lexical, and FOAM [Ehrig and Sure, 2005](Framework for Ontology Alignment and Mapping) along with *UCC* and *EUCC*. Due to the difference of abstraction levels of policy concepts and difference in ontological alignment, only manual mapping is achievable. The process of ontological mapping is shown in Figure 6.5 using the meta-transition technique. The conventional mapping techniques failed because of the differences in taxonomy and lexicons terminologies in the ontologies. Similar results have been presented in the thesis by Barret [2009]. FOAM employs a concept alignment formula that uses a combination of syntactical and lexical matching but most the above evaluated techniques including FOAM highly rely on lexicon matching for mapping the concepts in the ontology so they suffered badly due lack of semantic correlation information among concepts.

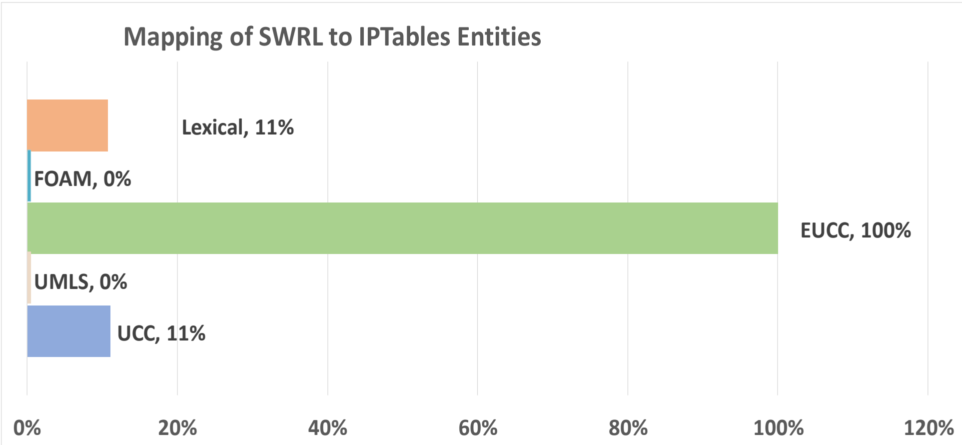


Figure 6.6: Analysis of different mapping techniques - Showing the results of mapped concepts in percentage using *UCC*, *UMLS*, EUCC, FORM, and Lexical mapping technique with help of PROMPT plugin in Protégé.

The Figure 6.6 shows the comparison of different mapping techniques for policy concepts at different abstraction levels, showing *EUCC* with the highest performance rate. The *UCC* used lexical matching technique so the results are same as for the lexical-based mapping technique. The analysis is conducted only for the frames (classes), which does not cover the slots (properties) and instances. Table 6.3 shows the details of mapped frames of *SWRL* and *IPTables* to *CIM* policy model using the meta-transition technique. Lexical and *UCC* managed to map few frames but *EUCC* managed to map all of the frames compared to other techniques.

Table 6.3: Mapping of SWRL and IPTables Policy Concepts to CIM (Policy Meta-Model)

SWRL to CIM Mapping							
Lexical/UCC		FOAM		EUCC		UMLS	
*M**	*NM***	*M*	*NM*	*M*	*NM*	*M*	*NM*
7	5	0	12	12	0	0	12
IPTables to CIM Mapping							
Lexical/UCC		FOAM		EUCC		UMLS	
M	*NM*	*M*	*NM*	*M*	*NM*	*M*	*NM*
9	8	0	17	17	0	0	17
M*= Mapped, NM**=Not Mapped							

6.5 Summary

The meta-transition technique presented in this chapter supports the semantic translation of policies at different abstraction levels. Most of the existing translation techniques focus on the syntactical translation without explicit consideration of policy semantics and *UCC*, a semantic based technique, does not cater the difference of semantic abstraction levels of different policy languages. We extended the *UCC* algorithm and explains the approach for policy translation. The proposed technique can also be used for simple syntactical policy translation. For future work, advance ontology tools can be developed to simplify the translation process and to support the meta-transition technique. We use *SWRL* and *IPTables* for the practical reasons because they both have different semantic levels. Further experiments can be conducted to test the technique with other policy languages at different levels of semantic difference.

Chapter 7

User-driven Certainty Factor Support Model to Resolve Semantic Conflicts

In this chapter, we present our investigation of the problem related to conflicting exclusive disjunctive uncertain inference rules. We extend a classical conflict resolution technique, the Certainty Factor Model [Dan and Dudeck, 1992, Heckerman, 1990], with an intelligent user driven approach to resolve the conflicts in inference rules. We outline the theoretical foundation of our approach and describe the reasoning capabilities and algorithms for the proposed technique. We demonstrate the perceived effectiveness of our approach through presentation of experimental results in comparison to probabilistic approaches based on real time test scenarios using a test-bed. We envision the application of our approach in smart homes that have evolved as new challenging environment for user driven intelligent systems. Most of the existing management solutions struggle due to inflexibility to adapt to new changing requirements in our homes and scarcely involve home users in the management process.

7.1 Introduction

This chapter builds upon our previous Chapter 5 with following three main contributions. Firstly, we extend the Certainty Factor Model [Dan and Dudeck, 1992, Heckerman, 1990] to deal with independent exclusive disjunctive uncertain rules. Secondly, we propose a belief support model based on supporting rules that can help to resolve conflicted uncertain rules. Lastly, we propose user feedback loop to deal with dead-locks caused by unresolved conflicted uncertain rules. The chapter is structured as follows: in §7.1, we introduce the role of users in decision systems and the *HANmanager* framework to support users in decision systems. In §7.2, we outline the main research challenge addressed in this chapter. §7.3 explains a problem scenario that will be used in rest of chapter to explain the construction of proposed technique. In §7.4, we explain the limitations of two major approaches to resolve uncertainty and present a support model that layouts the crux of our technique. In §7.5, we present the abstract algorithms to explain our proposed technique. §7.6 provides empirical results for evaluation of our technique in comparison of other most popular technique. Finally, in §7.7, we conclude and summarise our findings and outline further work.

7.1.1 User-driven Decision System in Smart Homes

In the last decade, convergence of network enabled devices and complex network services have changed the traditional view of home area networks (HANs). Recent research developments are aiming to realise the vision of smart home networks in next decade. However, most of the current attempts for well connected smart homes are still far behind a reality. Many of the proposed approaches [Chetana Sarode, 2012, Gaul and Ziefle, 2009, Meyer and Rakotonirainy, 2003] lack substantial user involvement in their proposed solutions. These management systems (lacking fine grained user control) most of the time tend to make decisions on the behalf of home users, some times disregarding actual user requirements, which results in losing viability in typical smart management scenarios e.g., power and energy control, and security. Hence, these systems also become inadequate to adapt to changing user requirements. Taking home user inputs (as governing rules) in the *HAN* control loop [Jennings et al., 2007, Kielthy et al., 2010] may increase viability of a system in a practical manner, but it also increases the chances of imprecise knowledge flow if rules are logically inaccurate and can lead erroneous system behaviour if not handled properly. Even if user defined rules are logically sound, still there can be situations where two independent rules may end up in a conflict because of dynamic contextual changes–such situation can also occur if control system relies on the input of faulty devices or systems. Under these circumstances, conflicts in decision control systems are inevitable. This chapter revolves around the problem of conflicted user rules, conflicts that can be detected at run time when the rules are set to be executed but difficult to resolve because of the associated evidential uncertainty.

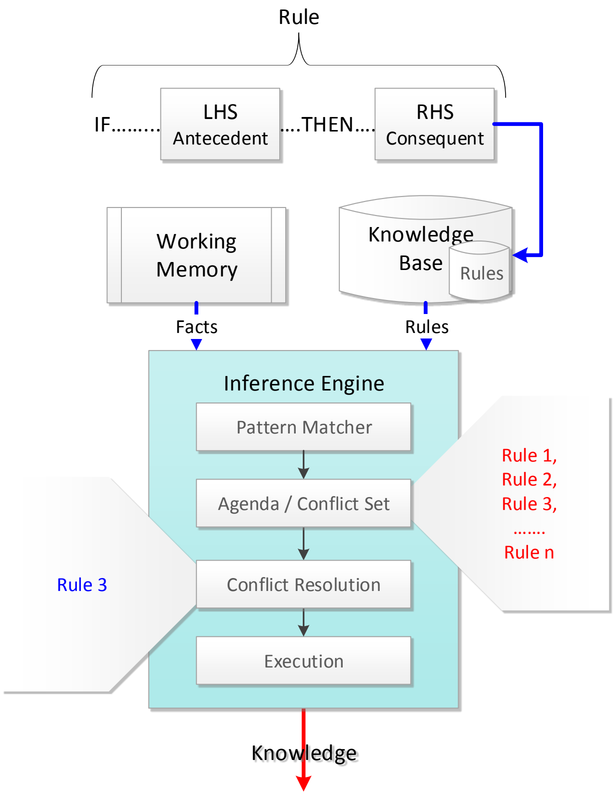


Figure 7.1: An inference engine takes facts/evidences and rules and processes them to infer new knowledge using deductive reasoning. Sometimes there can be conflicting rules that might interfere with inference. Conflict resolution is to resolve the conflicts and pick most appropriate rule for execution.

7.1.2 Inference Rules and Inference Engine

Logic is central to our *HANmanager* framework. Logic basically explicates the manner of performing reasoning, which is then used by the control systems within the Home Area Network (*HAN*) system to make intelligent decisions at the time of need. Considering a *HAN* logical system as an expert system, we can divide it into three main components

[Griffin and Lewis, 1989]: knowledge base, inference engine and working memory as shown in Figure 7.1. There exist many modes of formal reasoning but one of the most popular kinds is rule-based reasoning [Bryant, 2009, Clark, 1988], which is an obvious choice in case of the *HANmanager* to accommodate home users in the *HAN* control loop with the help of inference rules.

An inference rule containing a set of premises (facts/evidences) and a conclusion, represents a presumed rationale behind a piece of knowledge in the knowledge-base. The inference rules used in our framework have two structural constituents [Poole, 1997]: antecedent and consequent. There exist three main types of antecedents: conjunctive, disjunctive and negative. In our work, we use *SWRL* [Horrocks et al., 2004] to represents inference rules, and it only supports conjunctive antecedents and single action consequent to our best knowledge. The rule-based semantic reasoner in our framework renders over knowledge and rules specified in the knowledge base, and draws conclusions for an intelligent *HAN* management system. We assume that the *HAN* user provides basic knowledge facts and specifies the inference rules through intuitive interfaces; based on the given facts and *HAN*-specific knowledge, the inference engine can infer new knowledge and can also learn. It is important to note that *SWRL* does not facilitate updating the existing knowledge in the knowledge base due to its monotonic nature (discussed later in this chapter). The Jess reasoner [Friedman-Hill, 2003, Laboratories, 2009], a rule based inference engine, is used in our test bed (discussed later in this chapter); it performs forwarding chaining by default, which is based on deductive reasoning. Jess implements the reasoning process by finding rules in the knowledge base that correspond to the facts or data in the working memory. All rules that match the current problem state (criteria) are selected into a conflict set (rules to be executed). A single rule from the conflict set is selected based on the employed conflict resolution strategy and action part of the selected rule is performed. It may result in changing the working memory and so does the knowledge base if required in an ideal situation.

7.1.3 HANManager- Rule-driven HAN Management System

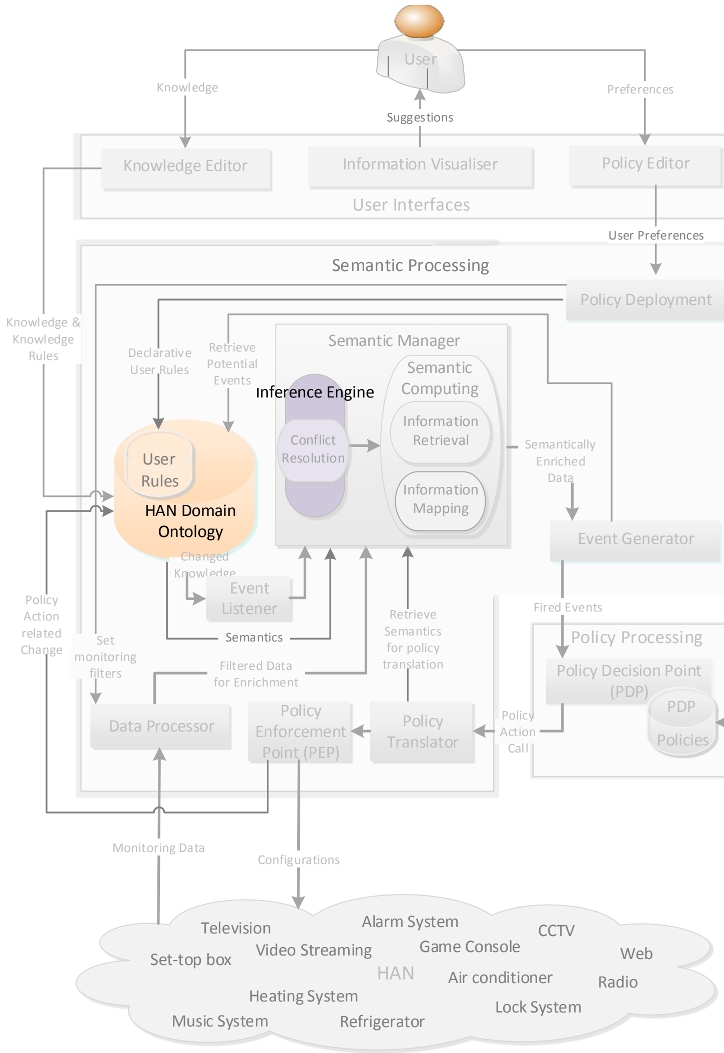


Figure 7.2: User-Centric, Policy-based *HAN* Management System: a semi-automated approach to manage and control home devices, applications and systems through system policies that are translated to network configurations. The figure shows the main components of the *HANmanager*. The inference engine plays a vital role in a rule based system and one of the main components of an inference engine is conflict resolution that is highlighted above.

The idea is to involve ordinary home users in the control loop of their home networks, which essentially helps them managing their network resources and other controllable entities involved in the *HAN* system -e.g., people, devices and services, according to their preferences with the help of rules. In the *HANmanager*, we have distinguished the user-defined rules into three categories: knowledge, policy and inference rules. The policy rules are user instructions for the system to act in certain way; inference rules are to aid thinking process and knowledge rules provide grounds for thinking. Our previous work (presented in Chapters 4 and 5) led us towards the classical problem of inference conflict that causes the *HANmanager* to work abnormally in the presence of imprecise information.

7.2 Problem Statement: Inference Conflict and Resolution

Typically, most formal logic systems adhere to the rules of classical reasoning [Sullivan, 2005], where monotonicity [Truszczynski, 1991] is one of major principles. Monotonicity is a reasoning property that states that new knowledge facts and rules added to the knowledge base should be admissible and should not affect the state of previously added facts and rules. However, to address the challenges of changing requirements and system adaptability, we require a non-monotonic reasoning system [Egly and Tompits, 1997] in the *HANmanager* so that the inferred knowledge should also be reflected in the knowledge base. However, introducing a non-monotonic logical system in *HAN* can induce many levels of other logical and semantical conflicts as well as inconsistencies in the knowledge base. On the other side, a monotonic logical system loses its usefulness in the *HANmanager* otherwise. The literature depicts that there are a number of successful attempts of using non-monotonic reasoning approach with the combination of ontology-based knowledge systems [Antoniou, 2002, Esposito, 2007].

SWRL follows the monotonicity principle, and hence, *SWRL* rules cannot be used directly to modify existing information in the knowledge base. In the *HANmanager*, at the time of rule specification, knowledge rules can be very abstract, which makes it quite difficult to analyse the conflicts with already specified rules. A conflict in the inference rules can be caused by the presence of false premises, which can induce wrong conclusions and rules may contradict each other. However, this problem is out of the scope this thesis and we mainly focused on the problem when established premises are correct and inference rules still result in contradicting state, making the logical system inconsistent. It is important to understand the inference conflict before we can discuss our solution strategy. When an inference engine encounters several rules that match the working memory (triggering facts) but only one has to be selected, is termed as an inference conflict. There are several strategies to resolve the inference conflict [Sanborn, 1987] e.g.,

- Refraction : once the rule has read, it is not used again;
- Recency : use the rule that has been used recently in such situation;
- Specificity : use the rule with the more specific condition (more facts);
- Priority : assign priority to rules (i.e. rank, utility, probability, cost, etc.) and choose the one with the highest priority;
- Parallel : process all rules with separate lines of reasoning.

Firstly, it is important to note here that traditional inference conflict resolution strategies focus on execution pattern of all selected rules, which is not as significant as the execution of right inference rule only among the conflict set. Secondly, none of above mentioned conflict resolution strategies addresses the problem of semantic conflicts. In semantically conflicted rules, it may also be required to defer the execution of other rules that may cause inconsistency or wrong inference in the *HANmanager*. Thirdly, this is a problem of reasoning with uncertainty (predicting which rule is most appropriate for execution). Therefore, we emphasize developing a conflict resolution strategy that first learns the context and then helps in selecting an appropriate inference rule for execution from the conflict set using some intelligent way for reasoning with uncertainty. Most of the conflict resolution strategies mentioned above are impractical in this scenario e.g., refraction and recency may cause execution of a non significant rule, which may lead to wrong inference state. Similarly, parallel execution may cause contradicting state of working memory and knowledge base in a non-monotonic logical system. The priority-based strategy has some potential but it may not work in complex situations e.g., when both rules have equal priority.

There is no as such preferred approach to evaluate semantic conflict resolution techniques for uncertain rules. We use three metrics to evaluate existing conflict resolution

Table 7.1: Comparison of Different Conflict Resolution Techniques for Uncertain Rules

Technique	Ability*	Decidability**	Preciseness***
Refraction	No	Yes	Circumstantial
Recency	No	Yes	Circumstantial
Specificity	Yes	Yes	Circumstantial
Priority	No	No	Circumstantial
Parallel	No	No	None
Probability	Yes	Yes	Partial
Certainty Factor Model	Yes	Yes	Partial

*Semantic Conflict Resolution Ability **Decidability Under Uncertainty ***Preciseness of Results

techniques for uncertain rules: ability to resolve semantic conflict, decidability and preciseness. Table 7.1 shows different techniques with their decidability, preciseness and conflict resolution abilities for uncertain rules. Later in this article, we use decidability and preciseness for evaluation of our proposed technique.

7.3 Running Example: Energy Saving and Security in Smart Homes

This section aims to provide a running example used in the rest of this chapter. The scenario highlights an inference conflict and emphasises the necessity of user-driven conflict resolution strategy using reasoning under uncertainty. Let us suppose, Ben's family is going on vacation and they have set-up the *HANmanager* control system. The following are some policy, knowledge and inference rules set-up for the control of heating, light and security systems:

(a) Run heating system at specific hours of the day only in the rooms where average house temperature is below 14 °C and family is not on vacation. (Policy Rule);

(b) Family is on vacation from 1st June to 30th June. (Knowledge Rule);

(c) If somebody is at home, family is not on vacation. (Inference Rule 1);

(d) If today's date is less than 30th June (vacation over date), family is on vacation. (Inference Rule 2).

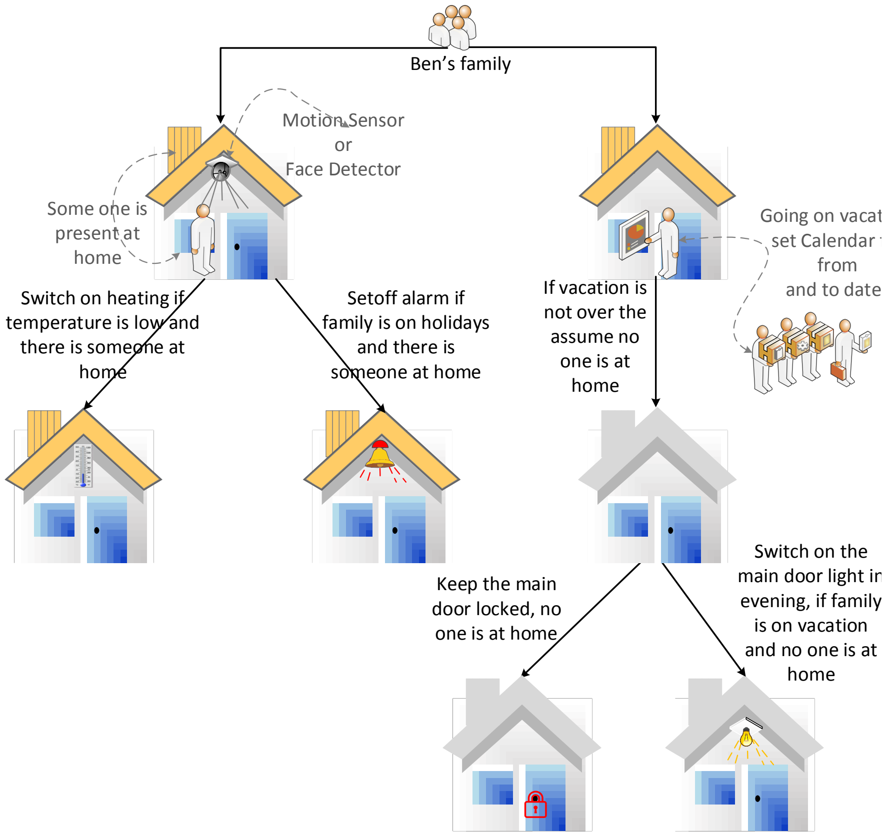


Figure 7.3: Smart Home Management Scenario: Mr. Ben's family is going on vacation for a month and they have set-up the HANmanager to control energy and security automatically using some intelligent rules.

Now assume that family is on vacation and will be back on 30th of June. If someone (guest, mechanic, house care taker or any inhabitant) arrives home earlier than 30th of June or motion sensor picks up presence of an intruder, the *HANmanager* is required to take appropriate actions. However, the *HANmanager* system can fall into an uncertain state after reasoning over Inference Rules 1 and 2 because the state of some one presence will become inconsistent. Here it is important to highlight that reasoning under uncertainty is what we require to deal with in this situation in order to resolve conflicting state of system as shown in Figure 7.3). None of the conflict resolution strategies discussed earlier in §7.2 can help solving this problem. There are two most popular approaches that we can use for reasoning rules under uncertainty e.g., certainty factor model [van der Gaag, 1994] and probabilistic reasoning [Pearl, 1988]. Suppose, the following are two *SWRL* rules that may end in a conflicting state:

```
Sensor(?x) ^ Home(?y) ^ hasSensor(?x,?y) ^
sensePresence(?x, true) -> isSomeOnePresent(?y, true)
(Description: If sensor senses presence of some one,
then there is some one at home.)
```

... *R1*

```
Calendar(?x) ^ Home(?y) ^ hasCalendar(?x,?y)
^ isOnHolidays(?x, true) -> isSomeOnePresent(?y, false)
(Description: if calendar says that family is on holidays,
then there is no one at home.)
```

... *R2*

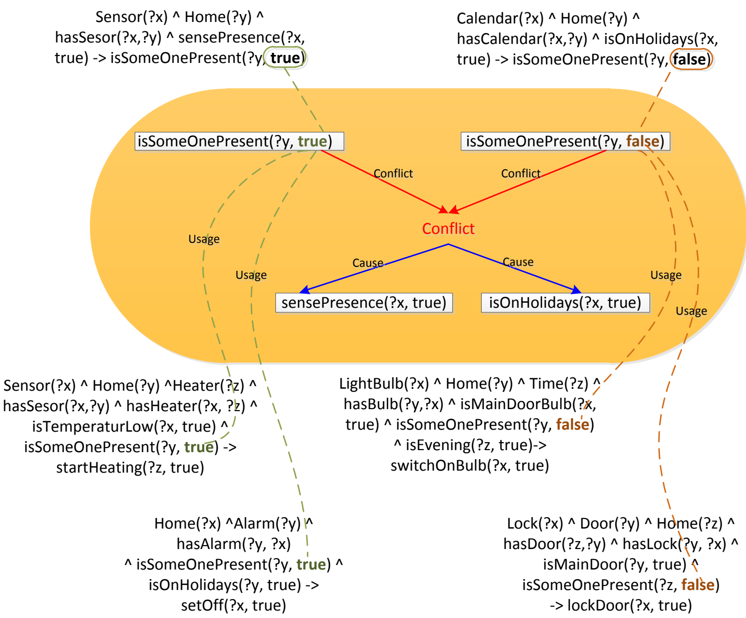


Figure 7.4: Smart Home Management Scenario: Mr. Ben's family is going on vacation for a month and they have set-up the HANmanager to control energy and security automatically using some intelligent rules.

A conflict of two inference rules is shown in Figure 7.4. The conflicting property is `isSomeOnePresent` due to two different types of values "true" and "false" are being set by two different rules *R1* and *R2* to same instance of Home. There would have been no conflict if either *R1* or *R2* is active one at a time but not both. Now we are required to measure the certainty or belief value associated to both rules in order to determine the exact situation that might have occurred. Causing property (observed evidence) for the activation of *R1* is `sensePresence` and similarly for *R2* is `isOnHolidays`. Therefore, the related certainty or belief value linked to both properties will also affect the certainty or belief values of their related rules.

```
Sensor(?x) ^ Home(?y) ^Heater(?z) ^ hasSensor(?x,?y)
^ hasHeater(?x, ?z) ^ isTemperatureLow(?x, true)
^ isSomeOnePresent(?y, true) -> startHeating(?z, true)
(Description: if some one is present at home and
 temperature is low then start heating)
```

... R3

```
Home(?x) ^Alarm(?y) ^ hasAlarm(?y, ?x)
^ isSomeOnePresent(?y, true) ^ isOnHolidays(?y, true)
-> setOff(?x, true)
(Description: if some one is present at home
and family is on vacation the set off the burglar's alarm.)
```

... R4

```
LightBulb(?x) ^ Home(?y) ^ Time(?z) ^ hasBulb(?y,?x)
^ isMainDoorBulb(?x, true) ^ isSomeOnePresent(?y, false)
^ isEvening(?z, true)-> switchOnBulb(?x, true)
(Description: if some one is not present at home
and it is evening then switch on the main door light.)
```

... R5

```
Lock(?x) ^ Door(?y) ^ Home(?z) ^ hasDoor(?z,?y) ^
hasLock(?y, ?x) ^ isMainDoor(?y, true) ^ isSomeOnePresent(?z, false)
-> lockDoor(?x, true)
(Description: if some one is not at home,
keep the main door locked.)
```

... R6

Table 7.2: Different examples of Contextual Semantic Conflicts

Scenario Set A	Conflicting Scenario Set B
(1A) Temperature Sensor: extreme temperature, it must be fire;	(1B) Faulty Fire Sensor: normal temperature recorded, there is no fire.
(2A) Faulty Motion Sensor: no motion detected, there is no intruder;	(2B) Camera Sensor: presence detected, it must be an intruder.
(3A) Timer: it is not evening yet, there must be light outside;	(3B) Weather Sensor: it is dark cloudy outside, there is no light.
(4A) Door Lock Sensor: door lock is open, it is a security risk;	(4B) Motion Sensor: no presence detected, there is no security risk.
(5A) Voice Detection Security: voice not recognised, unknown Person is present;	(5B) Face Recognition Sensor: face recognised, Mr. Ben is present.

Moreover, *R3* and *R4* are dependent on *R2*, and *R5* and *R6* are dependent on *R3*. If a wrong decision is reached over *R3* and *R4*, it may trigger erroneous behaviour in the system. Table 7.2 presents few other abstract examples of contextual semantic conflicts of similar nature.

7.4 Technique: User Driven Certainty Factor Support Model for Semantic Conflict Resolution

In this section we discuss our proposed solution for conflict resolution; we briefly touched the certainty factor model algebra used in the proposed approach. Table 7.3 presents model notations based on the running example rules *R1* and *R2*. Lastly, we also discuss two other supportive models and their limitations in HAN management reasoning system.

7.4.1 Certainty Factor Model Algebra for Inference Rules

Certainty factor model is a heuristic based reasoning approach, where subjective uncertainty measures are used to make an intelligent decision. Using the certainty factor model, we can compute a change in belief in any rule or evidence in HAN reasoning system. We do so by calculating certainty factor for an evidence of a inference rule in question. The certainty factor of an evident/condition is calculated by subtracting the measure of disbelief from the measure of belief and the subjective certainty measures

Table 7.3: Algorithmic Notations

Notation	Description
t	Truth, the property value.
f	False, the property value.
v	Action/Effect, an action data property.
e	An event with true value.
$\grave{e}$	an event with false value.
c	Class, an entity related an event.
i	Instance of sensor class
R	*e(t/f)* $\rightarrow$ *v(t/f)*, an inference rule
MB	Measure of belief.
MD	Measure of disbelief.
CF	Certainty factor associated with a rule or evidence
a	A positive number > 0 but < 1
b	A negative number < 0 but > -1

are set by a *HAN* user. Our approach is based on Certainty Factor Model and Certainty Factor Algebra used in our approach is discussed below:

$$CF = MB - MD \tag{7.1}$$

where $1 \geq MB \geq 0, 1 \geq MD \geq 0$, and $1 \geq CF \geq 1$ and CF= Certainty Factor, MB= Measure of Belief, MD= Measure of Disbelief.

For a single rule R given multiple evidences e_x and e_y, as shown in the Figure 7.3, the certainty factor algebra is:

$$\begin{aligned} MB[R|e_x \wedge e_y] =& MB[R|e_x] + MB[R|e_y] - (MB[R|e_x] \times MB[R|e_y]) \\ =& MB[R|e_x] + MB[R|e_y](1 - MB[R|e_x]) \\ MD[R|e_x \wedge e_y] =& MD[R|e_x] + MD[R|e_y] - (MD[R|e_x] \times MD[R|e_y]) \\ =& MD[R|e_x] + MD[R|e_y](1 - MD[R|e_x]) \end{aligned} \tag{7.2}$$

$$CF[R|e_x \wedge e_y] = MB[R|e_x \wedge e_y] - MD[R|e_x \wedge e_y]$$

For multiple rules R_x and R_y given a single evidence e, the certainty factor algebra is:

$$\begin{aligned}
MB[R_x \wedge R_y|e] &= min(MB[R_x|e], MB[R_y|e]) \\
MB[R_x \vee R_y|e] &= max(MB[R_x|e], MB[R_y|e]) \\
MD[R_x \wedge R_y|e] &= min(MD[R_x|e], MD[R_y|e]) \\
MD[R_x \vee R_y|e] &= max(MD[R_x|e], MD[R_y|e])
\end{aligned} \tag{7.3}$$

$$\begin{aligned}
CF[R_x \wedge R_y|e] &= MB[R_x \wedge R_y|e] - MD[R_x \wedge R_y|e] \\
CF[R_x \vee R_y|e] &= MB[R_x \vee R_y|e] - MD[R_x \vee R_y|e]
\end{aligned}$$

7.4.2 Limitations of Certainty Factor Model

In the Certainty Factor Model, the law of rules combination should be independent of the way rules are fired or executed. This means the combined certainty factor should obey associative and commutative rules. The certainty factors for *R1* and *R2* given *e1* and *e2* respectively are:

$$\begin{aligned}
CF[R_1|e_1] &= MB[R_1|e_1] - MD[R_1|e_1] \\
CF[R_2|e_2] &= MB[R_2|e_2] - MD[R_2|e_2]
\end{aligned} \tag{7.4}$$

Where *e1* is `sensePresence`, *e2* is `isOnHolidays`, *R1* is *e1(t)* $\rightarrow$ *v1(t)* and *R2* is *e2(t)* $\rightarrow$ *v1(f)*. The evidences *e1* and *e2* are not causally related, hence, their effects on *R1* and *R2* are independent of each other. Unfortunately, the combined certainty factor can only work for causally related evidences or rules. Therefore, the relative measures of belief or disbelief for an independent evidence or rule can not be calculated straight forwardly in this case. Taking the example of *R1* and *R2*, the conflicting property is *v1* and with two different values (t, f) and it leads to a different conclusion (*v1(t)* or *v1(f)*) for given condition/evidence *e1* or *e2* respectively. The *e1* and *e2* are causally independent evidences so relative Certainty Factor Model can not be applied here. However, *v1(t)* is actually $\neg$*v1(f)* and *v1(f)* is $\neg$*v1(t)*, which means they are mutually exclusive. This is a semantic relationship between two mutually exclusive conditions. Using this idea, we can say if *v1(t)* is most likely to happen then *v1(f)* is equally unlikely to happen. However, we can not claim that *R1* is $\neg$*R2*.

We use following representation for calculating Certain Factor of exclusive disjunctive rules R_x and R_y:

$$\begin{aligned} MB[R_x|e \veebar R_y|\grave{e}] =& max(MB[R_x|e], MB[R_y|\grave{e}]) \\ MD[R_x|e \veebar R_y|\grave{e}] =& max(MD[R_x|e], MD[R_y|\grave{e}]) \\ CF[R_x|e \veebar R_y|\grave{e}] =& MB[R_x|e \veebar R_y|\grave{e}] - MD[R_x|e \veebar R_y|\grave{e}] \end{aligned} \tag{7.5}$$

Now the challenge is that how the belief network is created for exclusive disjunctive rule, this is the exact problem that we have addressed in this chapter by proposing a supporting model.

7.4.3 Limitations of Probabilistic Models

In probabilistic reasoning with uncertain evidences, a regular evidence is termed hard evidence. However, it is not always possible to observe the complete value of an evidence or to have a complete trust on a claimed observation, thus bringing uncertainty to the evidences and consequently to the related rules. The evidence with uncertainty is termed soft evidence [Pan et al., 2006, Peng et al., 2010]. There are also two types of probabilities [Wallsten et al., 1997]: experienced and subjective; experienced probability is calculated and subjective probability is based on experts view. Traditional probabilistic models can be used for uncertain evidences rules, however the Bayesian Model is a promising approach. Considering the same rules *R1* and *R2* given evidences *e1* and *e2*, the Bayesian probability is:

$$P(R1|e1) = \frac{P(e1|R1).P(R1)}{P(e1)} \tag{7.6}$$

$$P(R2|e2) = \frac{P(e2|R2).P(R2)}{P(e2)} \tag{7.7}$$

Here *P(R1|e1)* and *P(R2|e2)* are posterior probabilities; *P(e1|R1)* and *P(e2|R2)* are called likelihood; *P(R1)* and *P(R2)* are called a priori probabilities; and *P(e1)* and *P(e2)* are called marginal probabilities [Berger, 1985]. Bayesian inference requires pre-calculated probabilities to calculate the posterior probability. In the *HANmanager*, the initial calculations for the Certainty Factor Model and Bayesian inference are subjective; once the initiatory inference model is set-up, the experienced calculations take place to make inference. However, there are too many values to be pre-set for the subjective parameters -e.g., *(P(e1|R)*, *P(e2|R2)*, *P(R1)*, *P(R2)*, *P(e1)*, and *P(e2)* required for Bayesian-based inference calculations, which makes it ineffectual in *HAN* decision system, where *HAN* user input is unavoidable to make decision system keep rolling.

7.4.4 User-driven Certainty Factor Support Model

Due to limitations of certainty factor and probabilistic models, we propose an extension of User-driven Certainty to calculate the certainty factor of conflicted and exclusive disjunctive inferences rules. We also propose a support model that guesses on preliminary certainty factor model, however, it extends the model by creating a belief model on top of it with the help of user inputs and beliefs of other existing inference rules . We initially used user input for creating a support model, however, this approach failed in many cases. Suppose, if measure of belief and disbelief for a certain evidence or rule is set to be zero initially, then we can take a user input for the conflicting evidences or rules and later the *HANmanager* can use previously set belief measurement to make decision in future. This approach only works if there is certain behavioural pattern in the system events. However, the uncertainty factor associated with uncertain events would remain the same every time even if we manipulate the employed certainty factor model for the uncertain rules. Suppose *R1* and *R2* are triggered by two independent causes *e1(t)* and *e2(t)* respectively but they result in mutually exclusive effects *v1(t)* and *v1(f)* respectively. Even if the system keeps on learning with the help of user input (say user favours *R1*), then the chances are that model can go wrong on the occurrence of a similar situation next time when *R2* is required to be triggered but it goes for *R1* because user favoured it last time. In this case, learning based on user input is not a good option here as:

- behavioural pattern can not be learned when there is not historic data available to train the model;
- an event can be random, not necessarily occurring in a specific pattern always.

To deal with this issue, we add another step to get support of the calculating certainty by matching each conflicting rule with other active inference rules that are set to be executed. If the conditional part of a conflicted rule is matched with any of active rules, then we can safely assume that condition of conflicted rule is also stand true. However, if both rules conditions are found to be true from active rules and both rules have same certainty factors, then user input is taken as an ultimate solution. Suppose, the certainty factor for *R1* is a_i and for *R2* is a_j. If a_i is equal to a_j, a user input is taken to get confirmation on *R1* if it is a potential case that is going to happen in system. Rather than getting an affirmation on the effect, we seek an affirmation for the cause *e1*, the observed evidence. If user input is positive then certainty factor for *e1* is incremented and certainty factor for *e2* is decremented. Consequently, a change in belief in evidence changes the belief in rule as: $CF[R1|e1] = CF[e1] \times CF[R1]$, where $CF[e1]$ is changed after the user input. Further to this, *R1* is passed to the inference engine and remove the *R2* from the conflict set. However, if user input is negative then certain factor for *e1* is decreased and certainty factor of *e2* is increased.

7.5 Algorithms

In this section, we describe the detailed steps and related algorithms developed for conflict resolution using Certainty Factor Support Model.

7.5.1 Conflict Analysis and Rule Classification

As a first step, we require to classify inference rules into two categories based on the type of conflict: defeasible and indefeasible. In this chapter, we used the extended definition of defeasible reasoning [Moodley et al., 2012]. If two or more rules, which are exclusive disjuctive that are in conflict with each other, then they are considered as defeasible rules because one of them will be inactivated as the process of reasoning non demonstrative. In our approach, we mainly focus on mutually exclusive non-compatible inference rules, e.g., *R1* and *R2*. The consequent parts of both *R1* and *R2* are contradictory to each other. Here, it is important to note that the consequent parts of both rules should hold at least one conflicting property with two different values as previously discussed. However, this conflict may or may not occur because of late binding of instances of the class that holds the conflicting property. If the conflicting property is related to two different instances of same class then they may not be in a conflict in the first place. The detection of conflict between semantically inverted properties is out of the scope of this thesis.

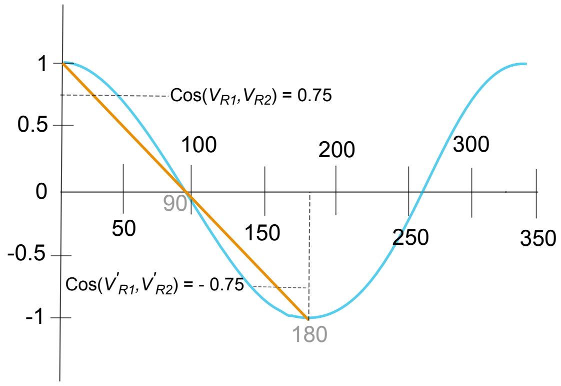


Figure 7.5: Two extreme values of cosine similarity for V_{R_1} and V_{R_2}. If cosine value is *1*, it means 0° angle between V_{R_1} and V_{R_2} and hence highest rank of similarity but if cosine value is *-1*, it means 180° angle between V_{R_1} and V_{R_2} and hence lowest rank of similarity.

In our approach, we measure the similarity of consequent of rules through the binary (0-1) vector space model [Salton et al., 1975], where the consequent of every specified rule is represented as a vector of similar or non-similar literals. When a new rule is specified in the *HANmanager*, it is ranked according to its proximity with other rules, where proximity is the similarity of consequent. In our case, Euclidean distance [Danielsson, 1980] is an inappropriate choice for classification to measure the proximity because the euclidean distance among vectors can be misleading [Gower, 1985], hence it is an inaccurate measure of proximity. Therefore, we used difference of angle (cosine similarity) [Cha, 2007] among the vectors to measure the similarity. Suppose the consequent of *R1* is V_{R_1} and for *R2* is V_{R_2}, then the cosine similarity for *R1* and *R2* is:

$$cos(V_{R_1}, V_{R_2}) = \frac{(V_{R_1}.V_{R_2})}{\|V_{R_1}\|.\|V_{R_2}\|} = \frac{\sum_{j=0}^{n} V_{R_{1j}} \times V_{R_{2j}}}{\sqrt{\sum_{j=0}^{n}(V_{R_{1j}})^2} \times \sqrt{\sum_{j=0}^{n}(V_{R_{2j}})^2}} \tag{7.8}$$

Where $V_{R_{1j}}$ and $V_{R_{2j}}$ represent the literals of vectors V_{R_1} and V_{R_2}, and cosine is a monotonically decreasing function for the interval/angle between the V_{R_1} and V_{R_2} as shown in the Figure 7.5. It means if cosine value is 1, there is 0° angle between V_{R_1} and V_{R_2} and hence highest level of proximity and also highest chances of being defeasible. Note that the rule classifier is to classify inference rules. Similarly, cosine value -1 means 180° angle between V_{R_1} and V_{R_2} and hence lowest level of proximity and lowest chances of being defeasible rules.

Table 7.4: Algorithm Notations for Algorithm 9

Notation	Description
V_{R_1}	Consequent part of *R1*
V_{R_2}	Consequent part of *R2*
r_n	New rule instance
V_r	Existing Rule Set
r_i	An instance of rule from existing rule set
V_{r_n}	Consequent part of new rule r_n
V_{r_i}	Consequent part of an instance of r_i from V_r
V_{dp_n}	Dot product vector for new rule r_n
V_{ab_n}	Absolute value vector for new rule r_n
V_{dp_i}	Dot product vector for an instance of r_i from V_r
V_{ab_i}	Absolute value vector for an instance of r_i from V_r

However, the analysed conflict may or may not exist at the time of execution because of dynamic late binding of instance with in a rule. Therefore, marking rules as defeasible only serves purport of notifying the *HANmanager* about the presence of potentially conflicted rules. The rule classification using cosine similarity is shown in Algorithm 9 using the notation given in Table 7.4.

Algorithm 9 Rule classification using cosine similarity model

Step 1: Rule classification in $\mathcal{V}_r$

if r_n *is new Rule* **then**
  foreach *element* r_i *in* $\mathcal{V}_r$ **do**
    if *calculateSimilarity*$(r_n,\ r_i) \geq THRESHOLD$ **then**
      r_n.defeasible $\leftarrow$ true
      r_i.defeasible $\leftarrow$ true
      r_i.addToConflictSet(r_n)
  return $\dot{\mathcal{V}}_r$

Step 2: Get cosine similarity for r_n and r_i

function *calculateSimilarity* (*Rule* r_n, *Rule* r_i)
if $r_n \neq NULL \wedge r_i \neq NULL$ **then**
  $\mathcal{V}_{r_n} \leftarrow r_n$.ConsequentVector
  $\mathcal{V}_{r_i} \leftarrow r_i$.ConsequentVector
  foreach *element* e_n *in* $\mathcal{V}_{r_n}$ *and* e_i *in* $\mathcal{V}_{r_i}$ *where* $\mathcal{V}_{r_n}.size == \mathcal{V}_{r_i}.size$ **do**
    $\mathcal{V}_{dp}$.add($e_n \times e_i$)
  foreach *element* e_n *in* $\mathcal{V}_{r_n}$ **do**
    $\mathcal{V}_{ab_n}$.add(e_n^2)
    $\mathcal{V}_{ab_n} = \sqrt{\mathcal{V}_{ab_n}}$
    return $\mathcal{V}_{ab_n}$
  foreach *element* e_i *in* $\mathcal{V}_{r_i}$ **do**
    $\mathcal{V}_{ab_i}$.add(e_i^2)
    $\mathcal{V}_{ab_i} = \sqrt{\mathcal{V}_{ab_i}}$
    return $\mathcal{V}_{ab_i}$
  return $\frac{\mathcal{V}_{dp}}{\mathcal{V}_{ab_n} \times \mathcal{V}_{ab_i}}$

7.5.2 Creating Certainty Factor Support Model

The Certainty Factor Support model (in vector form $\dot{\mathcal{V}}_r$) is created with the help of active rules $\mathcal{V}_a$ that are set to be executed. The exclusive disjunctive rules with conflicts, once marked defeasible, are taken further to find their support from the active rules set. List of classes $l_{\mathfrak{cs}_e}$, objects $l_{\mathfrak{cs}_i}$, data properties $l_{\mathfrak{cs}_c}$ and data properties values $l_{\mathfrak{cs}_v}$ are retrieved from the antecedent part of the conflicted rule. Similarly, same lists of items are fetched from the antecedent part of rules in the active rule set. If the conflicted rule has support from any of the active rules based on its match ratio of list of classes,

Table 7.5: Algorithm Notations for Algorithm 10

Notation	Description
$\dot{\mathcal{V}}_r$	Existing Rule Set with rules marked defeasible
$\mathcal{V}_a$	Set of active rules that are set to be executed
r_i	An instance of rule from existing rule set
r_n	An instance of rule from conflict set
r_a	An instance of rule from active rule set that are to be executed
$l_{\mathcal{CS}_c}$	List of data properties of an instance of rule from existing rule set
$l_{\mathcal{CS}_v}$	List of data properties values of an instance of rule from existing rule set
$l_{\mathcal{CS}_e}$	List of classes of an instance of rule from existing rule set
$l_{\mathcal{CS}_i}$	List of class objects of an instance of rule from existing rule set
$l_{\mathcal{P}_d}$	List of data properties from the antecedent of an instant of rule
$l_{\mathcal{C}}$	List of classes from the antecedent of an instant of rule
$l_{\mathcal{I}}$	List of classes objects from the antecedent of an instant of rule
MD	Measure of Disbelief

objects, data properties and data properties values. Match ratio is determined based on the similarity of corresponding items. As per our assumption (also discussed earlier under section Running Example and section Algorithms), the data properties in the antecedent part of conflicting rules are the causes of conflict so affirming the status of causing data properties by matching them with data properties of active rules can potentially resolve the conflict. However, the problem will remain the same if both data properties find support in active rule set. In that case, human input to resolve the conflict will be required. The creation of Certainty Factor Support Model is shown in Algorithm 10 using notation given in Table 7.5.

7.5.3 Using Certainty Factor Support Model for Conflict Resolution

After Certainty Factor Support Model $\dot{\mathcal{V}}_r$ is created, it is used to resolve the conflict. For each rule r_i, certainty factor CF with calculated with the help of its Measure of Belief MB and Measure of Disbelief MD. If the CF of r_i is greater than CF of conflicted rule r_n, r_i is added to active rule set $\mathcal{V}_a$ and r_n is discarded. And if the conflicted rule r_n has higher CF then r_n is added to active rule set and r_i is discarded. However, if CF of r_i and r_n is equal then human input is taken to select most appropriate rule. The use of Certainty Factor Support Model is shown in Algorithm 11 using notation given in Table 7.6.

Algorithm 10 Creating Certainty Factor Support Model

foreach *element* r_i *in* $\dot{\mathcal{V}}_r$ **do**
  if r_i*.ConflictSet.size* $\geq$ *0* **then**
    $l\mathcal{CS}_{c_1} \leftarrow r_i$.AntecedentVector.$l\mathcal{P}_d$
    $l\mathcal{CS}_{v_1} \leftarrow r_i$.AntecedentVector.$l\mathcal{P}_d$.Values
    $l\mathcal{CS}_{e_1} \leftarrow r_i$.AntecedentVector.$l\mathcal{C}$
    $l\mathcal{CS}_{i_1} \leftarrow r_i$.AntecedentVector.$l\mathcal{J}$
    Where $r_i.l\mathcal{C}$ and $r_i.l\mathcal{J}$ belong to $r_i.l\mathcal{P}_d$
    foreach *element* r_n *in* r_i*.ConflictSet* **do**
      $l\mathcal{CS}_{c_2} \leftarrow r_n$.AntecedentVector.$l\mathcal{P}_d$
      $l\mathcal{CS}_{v_2} \leftarrow r_n$.AntecedentVector.$l\mathcal{P}_d$.Values
      $l\mathcal{CS}_{e_2} \leftarrow r_n$.AntecedentVector.$l\mathcal{C}$
      $l\mathcal{CS}_{i_2} \leftarrow r_n$.AntecedentVector.$l\mathcal{J}$
      Where $r_n.l\mathcal{C}$ and $r_n.l\mathcal{J}$ belong to $r_n.l\mathcal{P}_d$
    foreach *element* r_a *in* $\mathcal{V}_a$ **do**
      $l\mathcal{CS}_{c_3} \leftarrow r_a$.AntecedentVector.$l\mathcal{P}_d$
      $l\mathcal{CS}_{v_3} \leftarrow r_a$.AntecedentVector.$l\mathcal{P}_d$.Values
      $l\mathcal{CS}_{e_3} \leftarrow r_a$.AntecedentVector.$l\mathcal{C}$
      $l\mathcal{CS}_{i_3} \leftarrow r_a$.AntecedentVector.$l\mathcal{J}$
      Where $r_a.l\mathcal{C}$ and $r_a.l\mathcal{J}$ belong to $r_a.l\mathcal{P}_d$
    if $l\mathcal{CS}_{c_1} \cong l\mathcal{CS}_{c_3} \wedge l\mathcal{CS}_{e_1} \cong l\mathcal{CS}_{e_3} \wedge l\mathcal{CS}_{i_1} \cong l\mathcal{CS}_{i_3} \wedge l\mathcal{CS}_{v_1} \cong l\mathcal{CS}_{v_3}$ **then**
      if r_i*.MD* $>$ *-1* **then**
        r_i.MD $\leftarrow r_i$.MD - 0.1
      if r_n*.MD* $<$ *1* **then**
        r_n.MD $\leftarrow r_n$.MD + 0.1
    else if $l\mathcal{CS}_{c_2} \cong l\mathcal{CS}_{c_3} \wedge l\mathcal{CS}_{e_2} \cong l\mathcal{CS}_{e_3} \wedge l\mathcal{CS}_{i_2} \cong l\mathcal{CS}_{i_3} \wedge l\mathcal{CS}_{v_2} \cong l\mathcal{CS}_{v_3}$ **then**
      if r_n*.MD* $>$ *-1* **then**
        r_n.MD $\leftarrow r_n$.MD - 0.1
      if r_i*.MD* $<$ *1* **then**
        r_i.MD $\leftarrow r_i$.MD + 0.1
return $\ddot{\mathcal{V}}_r$

Table 7.6: Algorithm Notations for Algorithm 11

Notation	Description
$\ddot{\mathcal{V}}_r$	Existing Rule Set with Certainty Factor Support Model
$\mathcal{V}_a$	Set of active rules that are set to be executed
r_i	An instance of rule from existing rule set
r_n	An instance of rule from conflict set
MB	Measure of Belief
MD	Measure of Disbelief
CF	Certainty Factor

Algorithm 11 Conflict resolution using Certainty Factor Support Model

foreach *element* r_i *in* $\ddot{\mathcal{V}}_r$ **do**
    if r_i*.ConflictSet.size* $\geq$ *0* **then**
        foreach *element* r_n *in* r_i*.ConflictSet* **do**
            r_i.CF $\leftarrow$ r_i.MB - r_i.MD
            r_n.CF $\leftarrow$ r_n.MB - r_n.MD
            if $r_i.CF > r_n.CF$ **then**
                $\mathcal{V}_a$.add(r_i)
            else if $r_i.CF < r_n.CF$ **then**
                $\mathcal{V}_a$.add(r_n)
            else if $r_i.CF == r_n.CF$ **then**
                getUserInput(r_n, r_i)
                if r_i*.isSelected* $==$ *true* **then**
                    if $r_n.MB > -1$ **then**
                        r_n.MB $\leftarrow$ r_n.MB - 0.1
                    if $r_i.MB < 1$ **then**
                        r_i.MB $\leftarrow$ r_i.MB + 0.1
                else if r_n*.isSelected* $==$ *true* **then**
                    if $r_i.MB > -1$ **then**
                        r_i.MB $\leftarrow$ r_n.MB - 0.1
                    if $r_n.MB < 1$ **then**
                        r_n.MB $\leftarrow$ r_n.MB + 0.1
                $\mathcal{V}_n$.remove(r_i)
        r_i.ConflictSet.remove(r_n)

If user selects r_i to be executed, then MB of r_n is decremented after checking MB lower bound for r_n and MB of r_i is incremented after checking MB upper bound for r_i. And r_i is added to active rule set $\mathcal{V}_a$ and r_n is removed from its conflict set. If user selects r_n to be executed, then MB of r_i is decremented after checking MB lower bound for r_i and MB of r_n is incremented after checking MB upper bound for r_n. And r_n is added to active rule set $\mathcal{V}_a$ and it is removed from the conflict set of r_i.

7.6 Evaluation

For the qualitative analysis, we compare our technique with probabilistic model in terms of validity, decidability and performance. Below we present the test theories

Table 7.7: Measures of Belief and Probability **??**

Value	Certainty Factor Model	Probabilistic Model
1	cf(x) = 1 means true, believed to be the case	p(x) = 1
0	cf(x) = 0 means false, no evidence	p(x) = $\grave{p}$(x), presumingly the prior
-1	cf(x) = -1 means definitely not the case	p(x) = 0

that we used to set-up the experiments, and discuss the results of the evaluation of the two modified strategies. It is important to note that the construction of technique is progressional based on the results of experiments. For example, the user input is initially set-up to build the support model, however, it is later also used to verify the decisions to improve the end results. We believe that our proposed model works better in random situations of events, even in the absence of historic data, where typical machine learning algorithms can not work straightforwardly.

7.6.1 Comparison of Models

Certainty Factors are similar to conditional probabilities, but rather representing the degree of probability of an outcome, it represents a measure of belief in the outcome. In article Wise and Henrion [1985], author presents a framework to compare certainty factor model and probability model. Presumingly, if two different representations of uncertainty lead to making the same decision, then they are operationally equivalent. However, these approaches do not provide agreed upon decision strategies, and so direct comparison is impossible. Therefore, we used piecewise interpolation of points of correspondence between certainty factor and probability models as given below in Table 7.7.

We used following test cases to compare the two models:

- No historic data, no supporting rules (output R1)
- 1 historic record in favour of R1, no supporting rules (output R1)
- 2 historic records in favour of R1, 1 supporting rule for R2 (output R2)
- 3 historic records, 2 in favour of R1, 1 for R2; 2 supporting rule for R2 (output R2)

- 4 historic records, 2 in favour of R1, 2 for R2; 1 supporting rule for R1 (output R1)
- 5 historic records, 3 in favour of R1 and 2 for R2; 0 supporting rule (output R2)
- 6 historic records, 3 in favour of R1, 3 for R2; 0 supporting rule (output R1)
- 7 historic records, 4 in favour of R1, 3 for R2; 1 supporting rule for R2 (output R2)
- 8 historic records, 4 in favour of R1, 4 for R2; 1 supporting rule for R2 (output R2)

We evaluate the models using random test cases with an assumption that there are no historic records available at the beginning to support the algorithms. The historic records are built step by step and keeps on adding in timely fashion. In above test cases, no historic data means that the system has encountered a first case of conflicted rules and supporting rule means the matched active rules. Let us consider the inference rules *R1* and *R2* again for an evaluation. For each test case, we compute measure of belief(MB) and measure of disbelief(MD) for *R1* and *R2* . Each measurement is impacted by four main factors:

- User Input if measures of certainty factor CF for *R1* and *R2* are equal.
- In the presence of supporting rule, measure of disbelief (MD) is affected.
- In the presence of supporting rule, user input is not taken even if measures of certainty factor CF for *R1* and *R2* are equal.
- For previously successful rule, the measure of belief(MB) is affected positively.

Result Analysis

We also calculate conditional probabilities for *R1* and *R2* by measuring prior probability ($\grave{P}$), likelihood (L) and posterior probability (P). The posterior probability is greatly affected by the absence of prior or likelihood probabilities. Following graphs show decideability of Certainty Factor Support Model (CFSM) and Probability Model (PM).

Decidability shows the power of the model for taking out HAN decision system from the state of undecidability as shown in Figure 7.6, where 1 means decidable, and 0 means undecidable state.

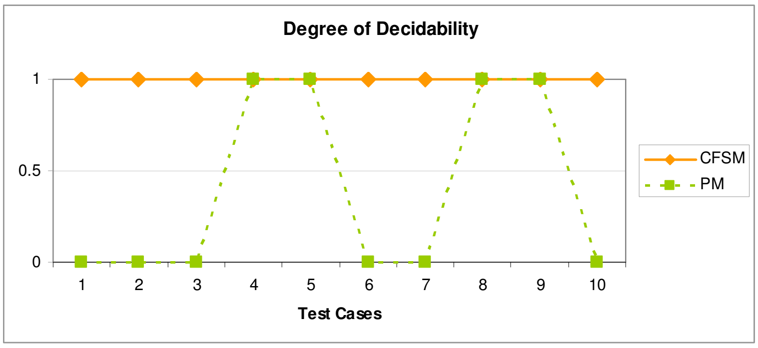


Figure 7.6: The graph showing degree of decidability of Certainty Factor Support Model (CFSM) and Probability Model (PM) where 1 means decidable, and 0 means undecidable.

As we can see from Figure 7.6 that CFSM gives 100% results mainly because of user input. On the other hand, probability model suffers due unavailability of prior or likelihood probabilities, which is expected for starting test cases. We also measure the validity of reached decisions by both models as shown in Figure 7.7:

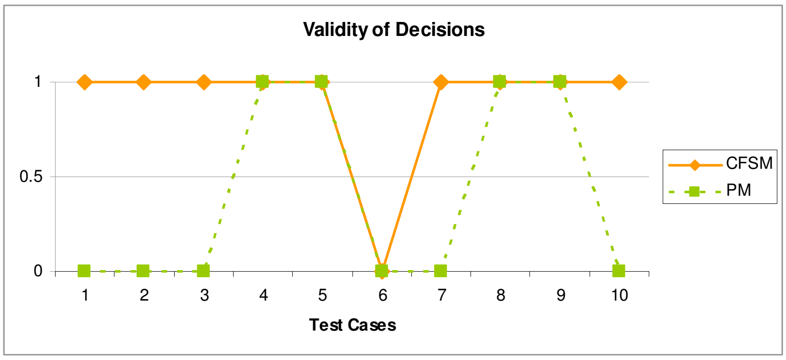


Figure 7.7: The graph showing validity of decisions made by Certainty Factor Support Model (CFSM) and Probability Model (PM) where 1 means correct, and 0 means false.

For CFSM, the correctness ratio remains above 85%, and for PM, the resulting ratio remained below 40%. The CFSM validity drops to 0 for test case 6 because there is no supporting rule, however, it recovers in next test case even in the absence of supporting rule. The PM performs low because of the unavailability of prior or likelihood probabilities. We also speculate the performance scores for both models based on the validity and decidability ratios as shown in Figure 7.8.

Figure 7.8: The graph showing performance scores for Certainty Factor Support Model (CFSM) and Probability Model (PM) calculated for each test case based on validity and decidability ratios.

If validity and decidability results are combined together as a truth-function of logical conjunction and using the amplified result values of truth and false (by adding 10 in true value and 1 in false value), the performance scores can be calculated by adding a constant along with previous score to every value for each case. The performance scores are simulated prediction, not the actual scores of models. Based on the simulated scores data, performance of CFSM remains steady and consistent. However, with the availability of historic data, PM performance may become equivalent to CFSM at some stage. Other learning algorithms may also be applicable in this situation if there are patterns of events and availability of historic data, otherwise, learning may not be straightforward due to random nature of events.

7.7 Summary

In this chapter, we present an extension of Certainty Factor Model to deal with independent, exclusive disjunctive and conflicted uncertain rules. We also propose a support model to resolve conflicts using Certainty Factor Model that is based on an idea of getting the support of other active inference rules (that are either set to be executed or recently executed with out any conflict). We also introduce an idea of user validation to construct support model and to verify the results, essentially helping in resolving the conflicts. We believe that proposed support model and user validation loop can also help to improve performance of other approaches as well e.g., Bayesian model, in order to resolve uncertain conflicted rules. The historic data and presence of event patterns in data are essential for the functioning of any modern day learning technique but our proposed model works fine even in the absence of historical data. The proposed model can be used in any real time, decision critical rule based system -i.e., health assistance system, shopping assistance system, automobile diagnostic system. The user feedback loop system can be used to improve human computer interaction based applications. For future work, we plan to use support model and user validation in combination with conditional probability and compare the results with Certainty Factor Support Model. The result shows that our proposed model works better in random situations of events, even in the absence of historic data, when compared with other predictive models. We have presented a technique that integrates uncertain non-monotonic reasoning with the use of quantitative information and user feedback. This differentiates it with other existing approaches, which normally fail to reach to any decidable state of a system accurately. However, there is a great margin of improvement in our proposed technique by incorporating other elements of contextual information other than just availability of supporting rules.

Chapter 8

Conclusion

In this chapter, we discuss contributions, limitations and future work related to our research. We also discuss ideas to extend our work and suggest pointers for the improvements. This chapter is structured as follows: §8.1 gives a detailed analysis of contributions of our research work. §8.2 presents shortcomings and limitations of our research work; and lastly §8.3 talks about future work.

8.1 Discussion

This thesis contributes to the areas of semantic computing for human-centric, policy-based home area network management, in particularly, it addresses the areas of semantic uplift of network flows, semantic-driven policy processing, semantic-aware policy translation and semantic-driven conflict resolution. The main contributions are:

1. A human-centric home area network management framework that adapts to dynamic functional changes and requirements in a home network;

2. Semantic uplifting techniques of monitoring data in the home area networks that extract relevant information from network flows and update semantic models appropriately;

3. A semantic-driven policy processing using semantic enrichment technique that uses semantic information to select and process abstract user-defined policies;

4. A semantic-aware policy translation technique that demonstrates the role of semantics in translating abstract/declarative user-defined policies to concrete/executable policies/configuration in the *HAN* management system;

5. A semantic-driven conflict resolution of independent exclusive disjunctive rules using a belief support model and user feedback loop to deal with unresolvable conflicted uncertain policy rules.

In this thesis, we presented a framework, the *HANManager*, as a solution to manage home area networks according to home users' requirements. We discussed the main components of the framework emphasising the role of semantic models and policies in managing *HANs*. We explained the framework's monitoring and controlling techniques: "top down" and "bottom up". We presented a generic technique (bottom up) for semantic uplifting of monitoring data and selection of policies based on extracted information. We also presented algorithms and a framework for implementation of our proposed (top down) technique for semantic enrichment of inferred data from a *HAN* domain ontology, and processing of selected policies based on extracted information from the inferred data. We also discussed techniques to process and translate user requirements into network configurations and used the semantics model to resolve inference rules conflict analysis.

Taking into account the first contribution, a framework that adapts to changing requirements of a home area network. Many frameworks have been suggested in the literature to manage home area networks and to automate home systems. Many of the proposed approaches [Chetana Sarode, 2012, Gaul and Ziefle, 2009, Meyer and Rakotonirainy, 2003] lack substantial user involvement in their proposed solutions. These management systems (lacking fine grained user control) most of the time tend to make decisions on the behalf of home users, some times disregarding users' requirements, which results in losing viability in a typical *HAN* management scenarios. This contribution partially answers our first and second research questions explained in Chapter 1. In Chapter 3, we explain a technique to interlink semantic information in different sub-domains of *HAN*.

The second and third main contributions of our thesis is the development of techniques to uplift and enrich the network flows information to select appropriate action policies to manage the monitored events. There are some tools available to monitor and control networks that provide some level of automation as well but, by and large, available monitoring tools and techniques use syntax-based data analysis techniques that provide information of limited value [Scheirer and Chuah, 2008] to an ordinary *HAN* users. This contribution answers our second, third and fourth research questions by explaining the required semantic information and the methods to specify, select and translate declarative policy for execution in Chapters 4, 5 and 6.

The fourth main contribution is extension of Usage and Change Control (*UCC*) algorithm [Barret, 2009]. The *UCC* algorithm is initially proposed to define semantic mapping of policy languages and their translation. However, the employed policy languages have to be of equal abstraction levels otherwise *UCC* fails the viability step (please see [Barret, 2009] for further details). We extended the *UCC* algorithm to address this problem and used the algorithm for translation of policy languages of different abstraction levels. This contribution answers our third research question by explaining the technique of transforming user defined policies to system configuration with the help of semantic models in Chapter 6.

The fifth main contribution is an extension of Certainty Factor Model [Dan and Dudeck, 1992, Heckerman, 1990] to deal with independent, exclusive disjunctive and conflicted uncertain rules. We also propose a support model to resolve conflicts using Certainty Factor Model that is based on an idea of getting the support of other active inference rules (that are either set to be executed or recently executed without any conflict). The main issue is that the behavioural pattern about an uncertain situation cannot be learned when there is no historic data available and most of the predictive models, e.g., Bayesian model, also do not work well in the absence of historic data. Our proposed technique overcomes this problem by calculating the certainty by matching each conflicting rule conditions states with other active inference rules that are set to be executed. This contribution mainly answers the fourth research question by explaining the use of inference to resolve the semantic conflicts of user defined policies in Chapter 7.

8.2 Limitations

We experimented with different implementations of our framework and also experimented our proposed techniques with other proposed approaches. Aside from our implementation efforts, there are several scenarios, where our approaches did not work as expected and in some cases we did not have time to experiment with other approaches. Some of the major limitations of research work are presented below.

Framework and Domain Model

1. The *HANmanager* works well in managing IP-enabled network devices, applications and systems in *HAN*. However, we did not try our framework with other home automation technologies [Chetana Sarode, 2012], e.g., Zigbee, X10;

2. Many of the components of *HAN* framework are minimally developed, especially the semantically enriched information visualiser.

3. Despite our earlier efforts, we could not succeed in developing a generic *HAN* domain model. Due to diversity of network related concepts and variety of *HAN* layouts, a standard domain model cannot be achieved. Though the domain model is capable of enhancements and systematic growth but it has to be in place at the design time of *HAN* system;

4. The framework does not support any self-learning features at the moment, which makes it dependent on *HAN* users or domain modellers for the information feed;

5. Another challenge is lack of sophisticated device management interfaces. Most *HAN* devices are cheap and they are usually available with minimal management features. Our initial intention is to control all types of home devices and the systems, however, the *HANmanager* currently can only be used for IP-enabled network communication on a open source router, controlling only the network traffic generated by different connected devices and systems, that goes through the gateway router.

Semantic Uplift of Monitoring Data

1. The proposed technique works only for real-time monitoring to analyse one data packet at a time, which is extremely slow;
2. The policy manager is minimally developed on a design of a policy system, lacking sophisticated policy authoring, policy validation, and policy execution components;
3. The monitoring data is initially mapped to data properties only in the data domain ontology, which could also be mapped to classes and individuals;
4. We assumed that the monitoring data maps to only one mapping object in the *HAN* domain ontology, which could result in multiple objects mappings.

Semantic Enrichment of Inferred Data

1. SWRL at its sole is not an adequate choice for specifying user policy rules in the *HAN* domain ontology as they are monotonic and undecidable under certain conditions [Wise, 1986, Wise and Henrion, 1985, Wise et al., 1987];
2. If SWRL policy rules are directly executed to change the values of data properties in the *HAN* domain ontology, it makes the model inconsistent. And with out domain model modifications, the *HANmanager* loses it viability in practical manner;
3. When a semantic graph is created, the individual and class binding assumes one-to-one relationship. This is a significant limitation of our approach;
4. Ontologies are great in modelling complex domains but they are not very well suited in real time large scale systems because of the extremely slow processing and inference capabilities.

Semantic Translation of Policy Rules

1. Most of the syntactical and grammatical rules employed in the process of policy translation are hard coded in our implementation;

2. Our technique heavily relies on pre-defined meta model for the process of translation;

3. The technique is only tested with limited scenarios using only few policy language constituents (e.g., priority, group).

Semantic Aware Conflict Resolution

1. Our technique is only valid for disjunctive semantic conflicts;

2. In the presence of historic data, our technique produced similar results as of the other probabilistic models;

3. The techniques results are based on simulation, we did not manage to deploy the technique on the real test bed;

4. Our technique does not define how it will be employed by a rule engine at real time to resolve conflicted rules.

8.3 Future Work

Framework Extension The *HANmanager* framework has a great potential to evolve; we plan to introduce self-learning feature for the future work. A seamless mechanism of self-learning leads towards self-managing intelligent home network that pro-actively acts on certain operational conditions to achieve optimal system state. In today's homes, it is desirable for network devices to automatically configure themselves based on the context of the environment and user preferences for both convenience and security purposes. For the future, our hope is if the collective efforts of either a commercially oriented solution (backed by big industry players) or of an open source community effort could actually help building the framework, the open source router projects can provide, perhaps, the clearest path to immediate deployment, if any solution developed on our generic testbed could be pared down to fit the manageable disk space and memory requirements of these cheap devices (e.g., Buffalo[1] and Linksys[2] devices that can be re-flashed with

[1] `http://www.buffalo-technology.com`

[2] `http://www.linksys.com/`

these open source alternative operating systems). We also plan to extend this work to federate *HAN* with service providers so that those service requirements related to application bandwidth, which can not be fulfilled due to network limitations, can also be addressed as shown in Figure 8.1. There is also great potential for work in area of *HAN* privacy. There has been some recent work done in this area by Brennan et al. [2014] but it requires further investigation. Figure 8.1 explains a scenario when there is a *HAN* service (High Definition Live Streaming) that requires guaranteed service level (such as bandwidth) for certain time of period and *HANmanager* makes a request to ISP through Federation Manager. ISP can dedicate resources based on the service level request through the Federation Manager.

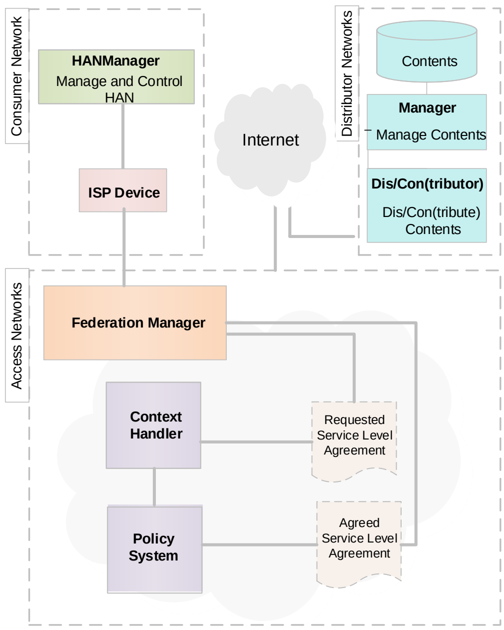


Figure 8.1: Federation of HAN with ISP provider: for the futuristic *HAN* management requirements, such as higher service quality for certain time period, *HAN* can submit a service level request and ISP can provide requested service level using Federation Manager.

To enable the communication of the *HANManager* with other autonomic elements, such as intelligent devices within HAN, there is an autonomic framework (FOCALE) proposed by Jennings et al. [2007] that defines general rules of communication among different autonomic elements in network systems. In the context of HAN, one of the main challenges is that HAN may contain many different devices. These devices may have different vendor specific programming models and may provide different management data, describing the same or similar concepts but in different manner. This makes it imperative to harness information model and ontology within HAN to abstract away vendor specific functionality to facilitate a standard way of reconfiguring the managed autonomic elements (devices). FOCALE proposes to incorporate an autonomic manager (AM) on top of every managed autonomic element. The AM is independent of the vendor specific functionality/data of the underlying autonomic element, which facilitates easier communication among the different elements for coordination of management decision making. Each AM realises the autonomic management functionality via an event manager, a state manager, an action manager, a reasoner, a learner, and a policy decision point (PDP). All these sub-components can communicate with each other and have access to the information model, an object model reflecting the current state of the autonomic elements, the system ontology, and the set of deployed policies governing the autonomic elements. Different AMs have the ability to communicate with other AMs to coordinate activities such as analysis of global network state or introduction of new policies or policy enforcement. Now considering *HANManager* as an AM for the HAN router, to coordinate with other autonomic network elements, there has to be AMs for all coordinating autonomic elements and *HANManager* can act as a master AM (can override decisions of other AMs) to make coordination simpler and robust. To optimise the performance of *HANManager*, in particularly, the processing of monitoring of data, there are many approaches that can be used easily. Nandy et al. [2014] propose to use optimized topology synthesis algorithms that are devised to minimize the computational effort. Bar et al. [2014] propose to use DBStream, which is an SQL-based system that explicitly supports incremental queries for rolling data analysis in comparison with Apache parallel data processing engine (Spark) [Zaharia et al., 2010]. Hoplaros et al. [2014] propose to use data summarization for network traffic monitoring using data mining approach.

Techniques Extension For the semantic uplift, enrichment and translation techniques, we plan to use Boolean Matrix and Graph Theory concepts to make the processing of policies more formalised and faster. Figure 8.2 depicts an excerpt of a *HAN* domain model based on a scenario: it contains classes (User, Device etc.), object properties (`hasDevice`, `hasApplication` etc.) and data properties (`hasUserName`, `hasIPAddress` etc.). An object property connects two different classes together trough a relation and a data property defines an attribute of a class.

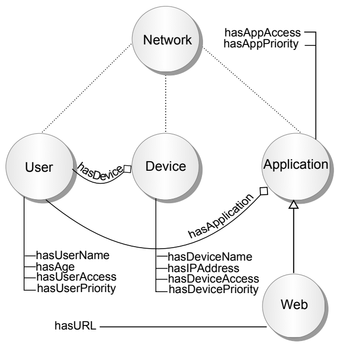


Figure 8.2: HAN domain concepts and their relationships and properties based on a scenario

If all the classes, object properties and data properties in the *HAN* domain model are elements in a set called domain D, then D can be defined as $D = (\mathcal{C}, \mathcal{P}_o, \mathcal{P}_d, \mathcal{I})$, where

- $\mathcal{C}$ is a finite set, called the classes of domain D;
- $\mathcal{P}_o$ is a finite set, called the object properties of D;
- $\mathcal{P}_d$ is a finite set, called the data properties of D;
- $\mathcal{I}$ is a finite set, called the instances of D.

If the elements of domain D can be represented as $\{e_1, e_2, e_3, ..., e_n\}$ then the elements of *HAN* domain model can be written as given in table 8.1.

Now using the ECA formalism, we a set S of semantic rudiments (s_1= event clause, s_2 = condition clause and s_3= action clause) then we can redefine the elements of domain D in boolean matrix format: $D^s =$

$$\begin{array}{c c}
 & \begin{array}{cccccccccccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} & e_{12} & e_{13} & e_{14} & e_{15} & e_{16} & e_{17} & e_{18} \end{array} \\
\begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}
\end{array}$$

If we have user defined policy:

Table 8.1: HAN domain elements naming convention based on a scenario

Name	Element		
`Network`	e1	`hasDeviceAccess`	e10
`Device`	e2	`hasDevicePriority`	e11
`User`	e3	`hasUserName`	e12
`Application`	e4	`hasAge`	e13
`Web`	e5	`hasUserAccess`	e14
`hasDevice`	e6	`hasUserPriority`	e15
`hasApplication`	e7	`hasAppAccess`	e16
`hasDeviceName`	e8	`hasAppPriority`	e17
`hasIPAddress`	e9	`hasUrl`	e18

```
Device(?x)^User(?y)
^hasDevice(?y,?x)^hasUserName(?y,"Alex")->hasDevicePriority(?x,"high")
(Description: Device belonging to User Alex has priority.)
                                                                    ... R0
```

Now using D^s, we can rewrite above SWRL policy *R0* as: $P^s =$

$$
\begin{array}{c}
 \\ s_1 \\ s_2 \\ s_3
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} & e_{12} & e_{13} & e_{14} & e_{15} & e_{16} & e_{17} & e_{18} \end{array} \\
\begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

Now to check if any policy if applicable for the given semantic graph, we will apply Hadamard product[Zhan, 1997], entry wise multiplication of semantic graph matrix and policy matrix. The entry wise multiplication takes two binary bits and performs the logical AND operation on each pair of corresponding bits. For each pair, the result is 1 if the first and second bits are 1, otherwise the result is 0. Suppose we have only one policy matrix P^s defined in our system. Consider if P^s and M^s are two $m \times n$ matrices then The Hadamard Product of P^s and M^s is defined by $\left[P^s \circ M^s\right]_{ij} = \left[p^s\right]_{ij}\left[M^s\right]_{ij}$ for all $1 \leq i \leq m, 1 \leq j \leq n$ be compared with row e^p_{ij} of p^s, where $i \neq j$. If resultant matrix of Hadamard Product $\tilde{R}$ is equal to P^s then policy will be selected for translation. The policy selection formula for a policy P_x^s is: $\left[P_x^s \circ M^s\right]_{ij} = \left[\tilde{R}\right]_{ij} = \left[P_x^s\right]_{ij}$. However, instances information is not include so this approach is incomplete with out instance level information. This work can be further extended to include enrichment and translation techniques, and further work can be done if it produces better results by using less processing time.

References

An architectural blueprint for autonomic computing. Technical report, IBM, June 2005. 35

D. Agrawal, Kang-Won Lee, and J. Lobo. Policy-based management of networked computing systems. *Communications Magazine, IEEE*, 43(10):69–75, Oct 2005. 35

K.G. Anagnostakis, S. Ioannidis, S. Miltchev, M. Greenwald, J.M. Smith, and J. Ioannidis. Efficient packet monitoring for network management. In *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 423 – 436, 2002. 68

Grigoris Antoniou. A nonmonotonic rule system using ontologies. In Michael Schroeder and Gerd Wagner, editors, *RuleML*, volume 60 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2002. 25, 31, 139

Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. Massachusetts Institute of Technology Press, 2003. 47

Apache. The apache struts web framework, 2006. URL `http://struts.apache.org`. 61, 105

Changseok Bae, Jinho Yoo, Kyuchang Kang, Yoonsik Choe, and Jeunwoo Lee. Home server for home digital service environments. *IEEE Trans. on Consum. Electron.*, 49(4):1129–1135, nov 2003. ISSN 0098-3063. doi: 10.1109/TCE.2003.1261207. 35

Arosha K Bandara, Emil C Lupu, Jonathan Moffett, and Alessandra Russo. A goal-based approach to policy refinement. In *Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY '04. IEEE Computer Society, 2004. ISBN 0-7695-2141-X. 38

A. Bar, A. Finamore, P. Casas, L. Golab, and M. Mellia. Large-scale network traffic monitoring with dbstream, a system for rolling big data analysis. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 165–170, Oct 2014. doi: 10.1109/BigData.2014.7004227. 169

Albert-Laszlo Barabasi. *Linked the new science of networks*. Perseus Pub., 2002. URL `http://www.amazon.com/Linked-The-New-Science-Networks/dp/0738206679`. 23

Keira Barret. *A Framework for the Semantic Translation of Policy Language Concepts*. PhD thesis, Waterford Institute of Technology, 2009. URL `http://www.tssg.org/publication/`. 118, 119, 123, 124, 131, 164

OJames Berger. *Statistical decision theory and Bayesian analysis*. Springer series in statistics. Springer, New York, NY [u.a.], 2. ed edition, 1985. 149

Andreas Berl, Roman Weidlich, Michael Schrank, Helmut Hlavacs, and Hermann Meer. Network virtualization in future home environments. In *Proceedings of* *the 20th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Integrated Management of Systems, Services, Processes and People in IT*, DSOM '09, pages 177–190, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04988-0. doi: 10.1007/978-3-642-04989-7_14. 35

Benjamin Bertran, Charles Consel, Patrice Kadionik, and Bastien Lamer. A sip-based home automation platform: an experimental study. In *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, pages 1 –6, oct. 2009. doi: 10.1109/ICIN.2009.5357075. 35

Antonis Bikakis, Grigoris Antoniou, and Panayiotis Hasapis. Strategies for contextual reasoning with conflicts in ambient intelligence. *Knowl. Inf. Syst.*, 27(1): 45–84, apr 2011. ISSN 0219-1377. doi: 10.1007/s10115-010-0293-0. 39

Dines Bjørner and Cliff B. Jones, editors. *The Vienna Development Method: The Meta-Language*, London, UK, UK, 1978. Springer-Verlag. ISBN 3-540-08766-4. 127

Piero Andrea Bonatti, Juri Luca De Coi, Daniel Olmedilla, and Luigi Sauro. Semantic techniques for the web. chapter Rule-based policy representations and reasoning, pages 201–232. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 3-642-04580-4, 978-3-642-04580-6. 49

S. Boros. Policy-based network management with snmp. In *Proceedings of the EUNICE 2000 Summer School*, number 16 in CTIT technical report series, Enschede, The Netherlands, 2000. University of Twente, Centre for Telematics and Information Technology. 20, 21, 22

Raouf Boutaba and Issam Aib. Policy-based management: A historical perspective. *J. Netw. Syst. Manage.*, 15(4):447–480, dec 2007. 19

R. Brennan, Z. Etizoni, K. Feeney, D. O'Sullivan, W. Fitzgerald, and S. Foley. Consumer-managed federated homes. *Communications Magazine, IEEE*, 52(6): 194–201, June 2014. ISSN 0163-6804. doi: 10.1109/MCOM.2014.6829964. 168

Rob Brennan, David Lewis, John Keeney, Zohar Etzioni, Kevin Feeney, Declan OSullivan, Jose A. Lozano, and Brendan Jennings. Policy-based integration of multiprovider digital home services. *Netwrk. Mag. of Global Internetwkg.*, 23(6):50–56, nov 2009. ISSN 0890-8044. doi: 10.1109/MNET.2009.5350353. 34, 35

Michael Forrester Bryant. *A Comparison of the Rule and Case-based Reasoning Approaches for the Automation of Help-desk Operations at the Tier-two Level.* PhD thesis, Nova Southeastern University, 2009. 29, 137

Jose M. Alcaraz Calero, Andres Munoz Ortega, Gregorio Martinez Perez, Juan A. Botia Blaya, and Antonio F. Gomez Skarmeta. Editorial: Detection of semantic conflicts in ontology and rule-based information systems. *Data Knowl. Eng.*, 69(11): 1117–1137, nov 2010. ISSN 0169-023X. doi: 10.1016/j.datak.2010.07.004. 39

Jose M. Alcaraz Calero, Andres Munoz Ortega, Gregorio Martinez Perez, Juan A. Botia Blaya, and Antonio F. Gomez Skarmeta. A non-monotonic expressiveness extension on the semantic web rule language. *J. Web Eng.*, 11(2):93–118, June 2011. ISSN 1540-9589. 39, 76

Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers and posters*, WWW Alt. '04, pages 74–83, New York, NY, USA, 2004. ACM. ISBN 1-58113-912-8. doi: 10.1145/1013367.1013381. 61, 105

Jeffrey Case, Mark Fedor, Martin Schoffstall, and James Davin. A simple network management protocol (snmp). Technical report, IETF Network Working Group, 1990. 22

Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007. URL `http://www.gly.fsu.edu/~parker/geostats/Cha.pdf`. 151

Liming Chen, Chris Nugent, Maurice Mulvenna, Dewar Finlay, Xin Hong, and Micheal Poland. A logical framework for behaviour reasoning and assistance in a smart home. *International Journal of the Atmospheric Sciences*, 2008. International Journal of ARM,VOL 9, NO.4 December 2008. 38

Prof Chetana Sarode. Smart home implementation techniques: A survey approach. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 1 (6):57–61, dec 2012. ISSN 2278-7798. 1, 34, 135, 163, 165

Marshini Chetty, Richard Banks, Richard Harper, Tim Regan, Abigail Sellen, Christos Gkantsidis, Thomas Karagiannis, and Peter Key. Who's hogging the bandwidth: the consequences of revealing the invisible in the home. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 659–668, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753423. 1

CORPORATE Cisco Systems Inc. *Cisco IOS 12.0 Quality of Service*. Cisco Press, 1999. ISBN 1578701619. 16, 17

Peter Clark. A comparison of rule and exemplar-based learning systems. In *IN INTERNATIONAL WORKSHOP ON MACHINE LEARNING, META-REASONING AND LOGICS*, pages 69–81, 1988. 29, 137

Alexander Clemm. *Network Management Fundamentals*. Cisco Press, 2006. ISBN 1587201372. 15

Kerry G. Coffman and Andrew M. Odlyzko. Growth of the internet. Technical report, 2001. 11

Martin J. O Connor and Amar K. Das. Sqwrl: A query language for owl. In Rinke Hoekstra and Peter F. Patel-Schneider, editors, *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2009), Chantilly, VA, United States, October 23-24, 2009*, volume 529 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. 61, 105

Nicodemos Damianou. A policy framework for management of distributed systems. Technical report, Imperial College of London, 2002. 22

Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *LECTURE NOTES IN COMPUTER SCIENCE*, pages 18–38. Springer-Verlag, 2001. 118

Qiu Dan and J. Dudeck. Certainty factor theory: Its probabilistic interpretations and problems. *Artif. Intell. Med.*, 4(1):21–34, February 1992. ISSN 0933-3657. doi: 10.1016/0933-3657(92)90035-N. URL `http://dx.doi.org/10.1016/0933-3657(92)90035-N`. 134, 164

P. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248, November 1980. ISSN 0146664X. doi: 10.1016/0146-664x(80)90054-4. URL `http://dx.doi.org/10.1016/0146-664x(80)90054-4`. 151

S. Davy, K. Barrett, S. Balasubramaniam, S. van der Meer, B. Jennings, and J. Strassner. Policy-based architecture to enable autonomic communications - a position paper. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 1, pages 590 – 594, jan. 2006a. doi: 10.1109/CCNC.2006.1593092. 3

Srteven Davy, Brendan Jennings, and John Strassner. On harnessing information models and ontologies for policy conflict analysis. In *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, pages 821–826, June 2009. 20

Steven Davy, Brendan Jennings, and John Strassner. Conflict prevention via model-driven policy refinement. In *Proceedings of the 17th IFIP/IEEE International Conference on Distributed Systems: Operations and Management*, DSOM'06, pages 209–220, 2006b. ISBN 3-540-47659-8, 978-3-540-47659-7. 37, 38

Steven Davy, Brendan Jennings, and John Strassner. The policy continuum-policy authoring and conflict analysis. *Comput. Commun.*, 31:2981–2995, August 2008. ISSN 0140-3664. 5, 22, 118

Jorge E. López de Vergara, Víctor A. Villagrá, and Julio Berrocal. On the formalization of the common information model metaschema. In *Proceedings of the 16th IFIP/IEEE Ambient Networks International Conference on Distributed Systems: Operations and Management*, DSOM'05, pages 1–11, 2005. ISBN 3-540-29388-4, 978-3-540-29388-0. 19, 21, 35, 105, 119

Thomas Delaet, Wouter Joosen, and Bart Vanbrabant. A survey of system configuration tools. In *Proceedings of the 24th International Conference on Large Installation System Administration*, LISA'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association. 2

Sudhir . Dixit and Ramjee Prasad. *Technologies for Home Networking.* Wiley, 2007. ISBN 9780470196526. URL `http://books.google.ie/books?id=6ZtV-kS9LyMC`. 12, 13

David Durham, Ron Cohen, and Jim Boyle. The cops (common open policy service) protocol. Technical report, IETF Network Working Group, 2000. 22

W. Keith Edwards, Mark W. Newman, and Erika Shehan Poole. The infrastructure problem in hci. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 423–432, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753390. 1

W. Keith Edwards, Rebecca E. Grinter, Ratul Mahajan, and David Wetherall. Advancing the state of home networking. *Commun. ACM*, 54:62–71, June 2011. ISSN 0001-0782. 2, 40, 66

Uwe Egly and Hans Tompits. Is non-monotonic reasoning always harder? In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *LPNMR*, volume 1265 of *Lecture Notes in Computer Science*, pages 60–75. Springer, 1997. 31, 139

Marc Ehrig and York Sure. Foam — framework for ontology alignment and mapping; results of the ontology alignment initiative. In *Proceedings of the Workshop on Integrating Ontologies*, volume 156, pages 72–76. CEUR-WS.org, October 2005. 131

Guillaume Ereteo, Michel Buffa, Fabien Gandon, and Olivier Corby. Analysis of a real online social network using semantic web frameworks. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 180–195, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04929-3. 36

M. Esposito. An ontological and non-monotonic rule-based approach to label medical images. In *Signal-Image Technologies and Internet-Based System, 2007. SITIS '07. Third International IEEE Conference on*, pages 603–611, Dec 2007. 31, 139

Liam Fallon and Declan OSullivan. Using a semantic knowledge base for communication service quality management in home area networks. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 43–51, 2012. doi: 10.1109/NOMS.2012.6211881. 35

Liam Fallon and Declan O'Sullivan. The aesop approach for semantic-based end-user service optimization. *IEEE Transactions on Network and Service Management*, 11 (2):220–234, 2014. doi: 10.1109/TNSM.2014.2321784. URL `http://dx.doi.org/10.1109/TNSM.2014.2321784`. 36

Miriam Fernandez, Ivan Cantador, Vanesa Lopez, David Vallet, Pablo Castells, and Enrico Motta. Semantically enhanced information retrieval: An ontology-based approach. *Web Semant.*, 9(4):434–452, dec 2011. ISSN 1570-8268. doi: 10.1016/j.websem.2010.11.003. 37

Merilee Ford, Tim Stevenson, H. Kim Lew, and Steve Spanier. *Internetworking Technologies Handbook*. Macmillan Publishing Co., Inc., Indianapolis, IN, USA, 1997. ISBN 1562056034. 12

Ernest Friedman-Hill. *Jess in Action: Java Rule-based Systems*. Manning, Greenwich, CT, 2003. ISBN 1-930-11089-8. URL `http\protect\kern+.2222em\relax//www.manning.com/friedman-hill/`. 31, 39, 137

Felix Fuentes and Dulal C. Kar. Ethereal vs. tcpdump: a comparative study on packet sniffing tools for educational purpose. *J. Comput. Sci. Coll.*, 20(4):169–176, apr 2005. ISSN 1937-4771. 62, 105

Jose Maria Fuentes, Jorge E. Lopez de Vergara, and Pablo Castells. An ontology-based approach to the description and execution of composite network management processes for network monitoring. In *DSOM*, pages 86–97, 2006. 36

Dragan Gaaevic, Dragan Djuric, Vladan Devedzic, and Bran Selic. *Model Driven Architecture and Ontology Development*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 3540321802. 23, 24

Sylvia Gaul and Martina Ziefle. Smart home technologies: Insights into generation-specific acceptance motives. In *Proceedings of the 5th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for e-Inclusion*, USAB '09, pages 312–332, Berlin, Heidelberg, 2009. Springer-Verlag. 1, 34, 135, 163

John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubezy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. The evolution of protege: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, jan 2003. ISSN 1071-5819. doi: 10.1016/S1071-5819(02)00127-1. 61, 105

John C. Gower. Properties of euclidean and non-Euclidean distance matrices. *Linear Algebra and its Applications*, 67: 81–97, June 1985. ISSN 00243795. doi: 10.1016/0024-3795(85)90187-9. URL `http://www.convexoptimization.com/TOOLS/Gower1.pdf`. 151

R. Greenlaw. *Breadth-depth Search is P-complete*. Technical report series. Department of Computer Science, University of New Hampshire, 1990. URL `http://books.google.ie/books?id=JMeWGwAACAAJ`. Last visited on 10/05/2013. 103

N. L. Griffin and F. D. Lewis. A rule-based inference engine which is optimal and vlsi implementable. In *Tools for Artificial Intelligence, 1989. Architectures, Languages and Algorithms, IEEE International Workshop on*, pages 246–251, Oct 1989. 29, 137

Rebecca E. Grinter, W. Keith Edwards, Marshini Chetty, Erika S. Poole, Ja-Young Sung, Jeonghwa Yang, Andy Crabtree, Peter Tolmie, Tom Rodden, Chris Greenhalgh, and Steve Benford. The ins and outs of home networking: The case for useful and usable domestic networking. *ACM Trans. Comput.-Hum. Interact.*, 16: 8:1–8:28, June 2009. ISSN 1073-0516. 2, 33

Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43: 907–928, December 1995. ISSN 1071-5819. doi: 10.1006/ijhc.1995.1081. 25

Nicola. Guarino. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1998. ISBN 9051993994. 25

Antonio Guerrero, Víctor A. Villagrá, Jorge E. López de Vergara, Alfonso Sánchez-Macián, and Julio Berrocal. Ontology-based policy refinement using swrl rules for management information definitions in owl. In *Proceedings of the 17th IFIP/IEEE International Conference on Distributed Systems: Operations and Management*, DSOM'06, pages 227–232, 2006. ISBN 3-540-47659-8, 978-3-540-47659-7. 36, 38

J. Guichard. A primer on policy-based network management. Technical report, Hewlett-Packard Company, 1999. 16, 17

Gopal Gupta. Horn logic denotations and their applications. In KrzysztofR. Apt, VictorW. Marek, Mirek Truszczynski, and DavidS. Warren, editors, *The Logic Programming Paradigm*, Artificial Intelligence, pages 127–159. Springer Berlin Heidelberg, 1999. 28

Dae-Man Han and Jae-Hyun Lim. Smart home energy management system using ieee 802.15.4 and zigbee. *IEEE Trans. on Consum. Electron.*, 56(3):1403–1410, August 2010. ISSN 0098-3063. 34

Francois Hantry, Mohand-Said Hacid, and Romuald Thion. Detection of conflicting compliance rules. In *Proceedings of the 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops*, EDOCW '11, pages 419–428, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4426-7. doi: 10.1109/EDOCW.2011.57. 39

D. Heckerman. Readings in uncertain reasoning. In Glenn Shafer and Judea Pearl, editors, *Readings in uncertain reasoning*, chapter Probabilistic Interpretations for MYCIN's Certainty Factors, pages 298–312. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-125-2. URL `http://dl.acm.org/citation.cfm?id=84628.85334`. 134, 164

David E. Heckerman and Edward H. Shortliffe. From certainty factors to belief networks. *Artif. Intell. Med.*, 4(1):35–52, February 1992. ISSN 0933-3657. doi: 10.1016/0933-3657(92)90036-O. 40

Mark Henricks. *Mastering Home Networking*. Sybex, 2000. 12

Richard C. Hicks. The no inference engine theory - performing conflict resolution during development. *Decis. Support Syst.*, 43 (2):435–444, March 2007. ISSN 0167-9236. doi: 10.1016/j.dss.2006.11.001. 39

Timothy L. Hinrichs, Natasha S. Gude, Martin Casado, John C. Mitchell, and Scott Shenker. Practical declarative network management. In *Proceedings of the 1st ACM workshop on Research on enterprise networking*, WREN '09, pages 1–10, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-443-0. doi: 10.1145/1592681.1592683. 49

Justin T. Ho, David Dearman, and Khai N. Truong. Improving users' security choices on home wireless networks. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, SOUPS '10, pages 12:1–12:12, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0264-7. doi: 10.1145/1837110.1837126. 1

Demetris Hoplaros, Zahir Tari, and Ibrahim Khalil. Data summarization for network traffic monitoring. *Journal of Network and Computer Applications*, 37:194 – 205, 2014.

ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2013.02.021. URL `http://www.sciencedirect.com/science/article/pii/S1084804513000593`. 169

Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semant. web*, 2:11–21, January 2011. ISSN 1570-0844. 61, 105

Ian Horrocks. SWRL Language FAQ, 2011. URL `http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ`. 28, 30, 61, 105, 119

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. Swrl: A semantic web rule language combining owl and ruleml. W3c member submission, World Wide Web Consortium, 2004. URL `http://www.w3.org/Submission/SWRL`. 30, 137

Hongxin Hu, GailJoon Ahn, and K. Kulkarni. Ontology-based policy anomaly management for autonomic computing. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, pages 487 –494, oct. 2011. 36

B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M. O Foghlu, W. Donnelly, and J. Strassner. Towards autonomic management of communications networks. *Communications Magazine, IEEE*, 45:112 –121, october 2007. ISSN 0163-6804. 3, 47, 87, 119, 135, 168

Igor Jurisica, John Mylopoulos, and Eric Yu. Ontologies for knowledge management: An information systems perspective. *Knowl. Inf. Syst.*, 6(4):380–401, July 2004. ISSN 0219-1377. 26

Aravind Kailas, Valentina Cecchi, and Arindam Mukherjee. A survey of communications and networking technologies for energy management in buildings and home automation. *Journal Comp. Netw. and Communic.*, 2012. 11

Varuzhan Kankanyan. Network traffic emulator, 2009. URL `http://www.nsauditor.com`. 59, 104

N. Kaviani, D. Gasevic, M. Hatala, and G. Wagner. Web rule languages to carry policies. In *Policies for Distributed Systems and Networks, 2007. POLICY '07. Eighth IEEE International Workshop on*, pages 188–192, June 2007. 37

John Keeney, Owen Conlan, Viliam Holub, Miao Wang, Laetitia Chapel, and Martín Serrano. A semantic monitoring and management framework for end-to-end services. In *Proceedings of 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011)*, 2011. 36

Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, jan 2003. ISSN 0018-9162. doi: 10.1109/MC.2003.1160055. 3

Jesse Kielthy, Kevin Quinn, Raquel Toribio, Pablo Arozarena, Sidath Handurukande, Marc Garcia Mateos, and Martin Zach. Design of a han autonomic control loop. In *Proceedings of the 5th IEEE international conference on Modelling autonomic communication environments*, MACE'10, pages 1–11, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-16835-3, 978-3-642-16835-2. 4, 41, 47, 66, 87, 135

Michael Kifer and Georg Lausen. F-logic: A higher-order language for reasoning about objects, inheritance, and scheme. *SIGMOD Records*, 18(2), June 1989. 28

Anneke G. Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 032119442X. 43

Torsten Klie, Benjamin Ernst, and Lars Wolf. Automatic policy refinement using owl-s and semantic infrastructure information. In *Proc. 2nd IEEE Int. Workshop on Modelling Autonomic Communications Environments (MACE)*, 2007. 37

Palanivel Kodeswaran, Sethuram Balaji Kodeswaran, Anupam Joshi, and Tim Finin. Enforcing security in semantics driven policy based networks. *Comput. Stand. Interfaces*, 33(1):2–12, jan 2011. ISSN 0920-5489. doi: 10.1016/j.csi.2010.03.010. 36

Sethuram Balaji Kodeswaran, Olga Ratsimor, Anupam Joshi, and Filip Perich. Utilizing semantic tags for policy based networking. In *GLOBECOM*, pages 1954–1958, 2007. 36

Richard E. Korf. Depth-first iterative-deepening: an optimal admissible tree search. *Artif. Intell.*, 27(1):97–109, September 1985. ISSN 0004-3702. 102

Tammo Krueger, Nicole Krämer, and Konrad Rieck. Asap: automatic semantics-aware analysis of network payloads. In *Proceedings of the international ECML/PKDD conference on Privacy and security issues in data mining and machine learning*, PSDML'10, pages 50–63, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-19895-3. 36

Sandia National Laboratories. Jess, the rule engine for the java platform, September 2009. URL `http://www.jessrules.com`. 31, 39, 137

Robin Laurnén. User centred access control for home networks. Master's thesis, Helsinki University of Technology (TKK), 2007. URL `mask.teknolog.fi/~llauren/thesis-llauren-100-remastered.pdf`. 1, 11

G.N. Leech. *Semantics*. A Pelican original. Penguin, 1974. URL `http://books.google.ie/books?\id=KHccAQAAIAAJ`. 45

David Lewis, Kevin Feeney, Kevin Carey, Thanassis Tiropanis, and Simon Courtenage. Semantic-based policy engineering for autonomic systems. In *Proceedings of the First international IFIP conference on Autonomic Communication*, WAC'04, pages 152–164, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-27417-0, 978-3-540-27417-9. doi: 10.1007/11520184_12. 36

Guoliang Liu, Shanquan Zhou, Xi Zhou, and Xin Huang. Qos management in home network. In *CIMCA '06: Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation*, page 203, Washington, DC, USA, 2006. IEEE Computer Society. 34

Ling Liu. Event-condition-action rules. In *Encyclopedia of Database Systems*, page 1068. Springer US, 2009. 21, 61, 105

Jane Lomax and Alexa T. McCray. Mapping the gene ontology into the unified medical language system: Research papers. *Comp. Funct. Genomics*, 5(4):354–361, June 2004. ISSN 1531-6912. doi: 10.1002/cfg.v5:4. URL http://dx.doi.org/10.1002/cfg.v5:4. 131

J. Loope. *Managing Infrastructure with Puppet*. OReilly Media, 2011. ISBN 9781449313227. URL http://books.google.ie/books?id=hYb2U-ZZByMC. 62, 105

Jorge E. Löpez De Vergara, Antonio Guerrero, Vïctor A. Villagrá, and Julio Berrocal. On the formalization of the common information model metaschema. In *Proceedings of the 16th IFIP/IEEE Ambient Networks International Conference on Distributed Systems: Operations and Management*, DSOM'05, pages 1–11, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-29388-4, 978-3-540-29388-0. doi: 10.1007/11568285_1. 5

Jorge E. Löpez De Vergara, Antonio Guerrero, Víctor A. Villagrá, and Julio Berrocal. Ontology-based network management: Study cases and lessons learned. *J. Netw. Syst. Manage.*, 17:234–254, September 2009. ISSN 1064-7570. 36

Sanjay Kumar Malik, Nupur Prakash, and S. A. M. Rizvi. Ontology merging using prompt plug-in of protégé in semantic web. In *Proceedings of the 2010 International Conference on Computational Intelligence and Communication Networks*, CICN '10, pages 476–481, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4254-6. doi: 10.1109/CICN.2010.151. URL http://dx.doi.org/10.1109/CICN.2010.151. 131

J. Mei and Paslaru. Reasoning paradigms for SWRL-enabled ontologies. In *Proceedings of International Workshop on Protege with Rules*, July 2005. 39

Sven Meyer and Andry Rakotonirainy. A survey of research on context-aware homes. In *Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003 - Volume 21*, ACSW Frontiers '03, pages 159–168, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc. ISBN 1-920682-00-7. 34, 135, 163

Marco Casassa Mont, Adrian Baldwin, and Cheh Goh. POWER prototype: towards integrated policy-based management. In *The Networked Planet: Management Beyond 2000, 7th IEEE/IFIP Network Operations and Management Symposium, NOMS 2000, Honolulu, HI, USA, April 10-14, 2000. Proceedings*, pages 789–802, 2000. 37

Kody Moodley, Thomas Meyer, and Ivan José Varzinczak. A defeasible reasoning approach for description logic ontologies. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, SAICSIT '12, pages 69–78, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1308-7. doi: 10.1145/2389836.2389845. URL http://doi.acm.org/10.1145/2389836.2389845. 150

Bob Moore, John Strassner, Andrea Westerinen, and Ed Ellesson. Policy core information model. Technical report, IETF Network Working Group, 2001. 19

B. Nandy, N. Seddigh, R.S. Makkar, and P.S. Pieda. Real-time network analyzer, May 27

2014. URL https://www.google.com/patents/US8737235. US Patent 8,737,235. 169

Wolfgang Nejdl, Daniel Olmedilla, Marianne Winslett, and Charles C. Zhang. Ontology-based policy specification and management. In *Proceedings of the Second European conference on The Semantic Web: research and Applications*, ESWC'05, pages 290–302, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-26124-9, 978-3-540-26124-7. doi: 10.1007/11431053_20. 36

Matthias Nickles and Davide Sottara. Approaches to uncertain or imprecise rules - a survey. In *Proceedings of the 2009 International Symposium on Rule Interchange and Applications*, RuleML '09, pages 323–336, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04984-2. doi: 10.1007/978-3-642-04985-9_30. 40

Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. online, 2001. URL http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html. 43

Gerard O'Driscoll. *The Essential Guide to Home Networking*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000. ISBN 0130198463. 13

Oracle. Java server pages, 2007. URL http://docs.oracle.com/javaee\/5/tutorial/doc/bnagx.html. 61, 105

Angela Orebaugh, Gilbert Ramirez, Josh Burke, and Larry Pesce. *Wireshark and Ethereal Network Protocol Analyzer Toolkit*. Syngress Publishing, 2006. ISBN 1597490733. 62, 105

Rong Pan, Yun Peng, and Zhongli Ding. Belief update in bayesian networks using uncertain evidence. In *Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on*, pages 441–444, Nov 2006. doi: 10.1109/ICTAI.2006.39. 148

Choon-Gul Park, Jae-Hyoung Yoo, Seung-Hak Seok, Ju-Hee Park, and Hoen-In Lim. Intelligent home network service management platform design based on osgi framework. In *Proceedings of the 9th Asia-Pacific International Conference on Network Operations and Management: Management of Convergence Networks and Services*, APNOMS'06, pages 550–553, Berlin, Heidelberg, 2006. Springer-Verlag. 34

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 1558604790. 143

Dimosthenis Pediaditakis, Anandha Gopalan, Naranker Dulay, and Morris Sloman. A configuration service for home networks. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1048 –1053, april 2012a. doi: 10.1109/NOMS.2012.6212028. 35

Dimosthenis Pediaditakis, Anandha Gopalan, Naranker Dulay, Morris Sloman, and Tom Lodge. Home network management policies: Putting the user in the loop. In *Policies for Distributed Systems and Networks (POLICY), 2012 IEEE International Sym-*

posium on, pages 9 –16, july 2012b. doi: 10.1109/POLICY.2012.12. 35

Yun Peng, Shenyong Zhang, and Rong Pan. Bayesian network reasoning with uncertain evidences. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 18(5):539–564, 2010. 40, 148

D. C. Plummer. Ethernet address resolution protocol, 1982. URL `https://tools.ietf.org/html/rfc826`. 62, 105

David Poole. Probabilistic partial evaluation: Exploiting rule structure in probabilistic inference. In *IJCAI*, pages 1284–1291. Morgan Kaufmann, 1997. 30, 137

Erika Poole, W. Edwards, and Lawrence Jarvis. The home network as a socio-technical system: Understanding the challenges of remote home network problem diagnosis. *Computer Supported Cooperative Work (CSCW)*, 18:277–299, 2009a. ISSN 0925-9724. 2, 41, 66

Erika Shehan Poole, Marshini Chetty, Tom Morgan, Rebecca E. Grinter, and W. Keith Edwards. Computer help at home: Methods and motivations for informal technical support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 739–748, New York, NY, USA, 2009b. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518816. 1

Joao Porto de Albuquerque, Heiko Krumm, and Paulo Licio de Geus. Policy modeling and refinement for network security systems. In *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*, pages 24–33, June 2005. 38

J. R. G. Pulido, M. A. G. Ruiz, R. Herrera, E. Cabello, S. Legrand, and D. Elliman. Ontology languages for the semantic web: A never completely updated review. *Know.-Based Syst.*, 19(7):489–497, November 2006. ISSN 0950-7051. 27

G.N. Purdy. *Linux iptables Pocket Reference*. OReilly Media, 2009. ISBN 9781449378981. URL `http://books.google.ie/books?\id=CCk6xpGeYjwC`. 62, 105

R. Guerin R. Yavatkar, D. Pendarakis. A framework for policy-based admission control. Technical report, Internet Engineering Task Force, 1999. 47

S. Ramanathan and R. Gusella. A home network controller for providing broadband access to residential subscribers. *IEEE Trans. on Consum. Electron.*, 41(3):859–868, August 1995. ISSN 0098-3063. doi: 10.1109/30.468073. URL `http://dx.doi.org/10.1109/30.468073`. 12

Vincent Ricquebourg, David Menga, David Durand, Bruno Marhic, Laurent Delahoche, and Christophe Loge. The smart home concept : our immediate future. In *E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on*, pages 23 –28, dec. 2006. 35

Taufiq Rochaeli and Claudia Eckert. Expertise knowledge-based policy refinement process. *Policies for Distributed Systems and Networks, IEEE International Workshop on*, 0:61–65, 2007. doi: http://doi.ieeecomputersociety.org/10.1109/POLICY.2007.23. 38

Javier Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, and G. Pavlou. A methodological approach toward the refinement problem in policy-based management systems. *Communications Magazine, IEEE*, 44(10):60–68, Oct 2006. ISSN 0163-6804. doi: 10.1109/MCOM.2006.1710414. 38

Mohamed Amin Sakka, Bruno Defude, and Jorge Tellez. A semantic framework for the management of enriched provenance logs. In *Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, AINA '12, pages 352–359, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4651-3. doi: 10.1109/AINA.2012.9. 55

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18 (11):613–620, 1975. The paper where vector space model for IR was introduced. 151

James C. Sanborn. A modifiable approach to expert systems development. In *Proceedings of SPIE 0786, Applications of Artificial Intelligence*, volume 99 of *SPIE 0786*, pages 99–106, FL, USA, 1987. SPIE. doi: http://dx.doi.org/10.1117/12.940607. 32, 140

J. Saperia. Ietf wrangles over policy definitions. Technical report, IETF Policy Framework Working Group, 2002. 21

Stefan Schaffer and Michael Minge. Error-prone voice and graphical user interfaces in a mobile application. *Speech Communication; 10. ITG Symposium; Proceedings of*, pages 1 –4, sept. 2012. 51

Walter Scheirer and Mooi Choo Chuah. Syntax vs. semantics: competing approaches to dynamic network intrusion detection. *Int. J. Secur. Netw.*, 3:24–35, December 2008. ISSN 1747-8405. 41, 51, 66, 164

Marc Schoolderman. *Fuzzy Lexical Matching*. PhD thesis, Radboud University Nijmegen, 2012. 69

B.V. Selic, D. Gaševic, D. Djuric, and V. Devedžic. *Model Driven Architecture and Ontology Development*. Springer Berlin Heidelberg, 2006. ISBN 9783540321828. URL `https://books.google.ie/books?id=Tv7SSrXm5OQC`. 43

Mary Sheahen and Marjorie Skubic. Design and usability of a smart home sensor data user interface for a clinical and research audience. In *Smart Homes and Health Telematics*, pages 13–20. Springer, 2015. 1

Erika Shehan and W. Keith Edwards. Home networking and hci: what hath god wrought? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 547–556, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: 10.1145/1240624.1240712. 1, 2

A. Shiravi, H. Shiravi, and A. A. Ghorbani. A survey of visualization systems for network security. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1313–1329, 2012. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.144. 2

Muhammad Shoaib Siddiqui, Syed Obaid Amin, and Choong Seon Hong. A set-top box for end-to-end qos management and home network gateway in ims. *Consumer Electronics, IEEE Transactions on*, 55(2):

527 –534, may 2009. ISSN 0098-3063. doi: 10.1109/TCE.2009.5174417. 35

Chakchai Soin. A survey of network traffic monitoring and analysis tools. In *Cisco Netflow*. Cisco Press, 2012. 2

John Strassner. Context-aware, policy-based seamless mobility using the focale autonomic architecture. In *Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, MDM '09, pages 568–573, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3650-7. 41, 50

John Strassner, Sven Van Der Meer, Brendan Jennings, and Miguel Ponce De Leon. An autonomic architecture to manage ubiquitous computing networks and applications. In *Proceedings of the first international conference on Ubiquitous and future networks*, ICUFN'09, pages 116–121, Piscataway, NJ, USA, 2009a. IEEE Press. ISBN 978-1-4244-4215-7. 66

John Strassner, Sven van der Meer, Declan OSullivan, and Simon Dobson. The use of context-aware policies and ontologies to facilitate business-aware network management. *Journal of Network System Management*, 17(3):255–284, September 2009b. ISSN 1064-7570. doi: 10.1007/s10922-009-9126-4. 36

Linying Su, David W. Chadwick, Andrew Basden, and James A. Cunningham. Automated decomposition of access control policies. In *Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY '05, pages 3–13, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2265-3. doi: 10.1109/POLICY.2005.10. 38

Scott M. Sullivan. *An Introduction To Traditional Logic: Classical Reasoning For Contemporary Minds*. BookSurge Publishing, 2005. 31, 139

L.M. Surhone, M.T. Tenoe, and S.F. Henssonow. *FCAPS: Telecommunications Management Network, Network Management, Alert Standard Form*. Betascript Publishing, 2010. ISBN 9786134568432. URL `http://books.google.ie/books?id=hStaYgEACAAJ`. 15

Joe Sventek, Alexandros Koliousis, Oliver Sharma, Naranker Dulay, Dimosthenis Pediaditakis, and Morris Sloman. An information plane architecture supporting home network management. In *Proceedings of 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011)*, 2011. 33, 34, 55

Sandeep Tata and Jignesh M. Patel. Estimating the selectivity of tf-idf based cosine similarity predicates. *SIGMOD Rec.*, 36:7–12, jun 2007. ISSN 0163-5808. doi: 10.1145/1328854.1328855. 91, 99

UTD Technologies. Web traffic generator, 2007. URL `http://download.cnet.com/Website-Traffic-Generator`. 59, 104

Gianluca Tonti, Jeffrey M. Bradshaw, Renia Jeffers, Rebecca Montanari, Niranjan Suri, and Andrzej Uszok. Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder. In *International Semantic Web Conference*, pages 419–437, 2003. 49

Miles Tracy, Wayne Jansen, Karen A. Scarfone, and Theodore Winograd. Sp 800-44 version 2. guidelines on securing public web servers. Technical report, Gaithersburg, MD, United States, 2007. 19

Thanh Tran, Philipp Cimiano, Sebastian Rudolph, and Rudi Studer. Ontology-based interpretation of keywords for semantic search. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 523–536, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-76297-3, 978-3-540-76297-3. 36, 37, 87, 106, 113

David Trastour, Robert Fink, and Feng Liu. Changerefinery: Assisted refinement of high-level it change requests. In *Proceedings of the 10th IEEE International Conference on Policies for Distributed Systems and Networks*, POLICY'09, pages 68–75, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-4670-4. 37

M. Truszczynski. Embedding default logic into modal nonmonotonic logics. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *Logic Programming and Non-Monotonic Reasoning: Proc. of the First International Workshop*, pages 151–165. MIT Press, Cambridge, MA, 1991. 31, 139

James Turnbull. *Pulling strings with Puppet : configuration management made easy.* FirstPress. Apress New York, NY, Berkeley, CA, 2007. ISBN 978-1-590-59978-5. URL http://opac.inria.fr/record=b1125928. 119

Kevin Twidle and Emil Lupu. Ponder2 - policy-based self managed cells. In *Proceedings of the 1st international conference on Autonomous Infrastructure, Management and Security: Inter-Domain Management*, AIMS '07, pages 230–230, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-72985-3. doi: 10.1007/978-3-540-72986-0_33. 61, 86, 105

Stanford University. Swrl jess bridge, 2010. URL http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessBridge. 61, 105

Linda C. van der Gaag. A pragmatic view of the certainty factor model. *Int. J. Expert Syst.*, 7(3):289–300, September 1994. ISSN 0894-9077. URL http://dl.acm.org/citation.cfm?id=196248.196260. 143

Paul E. van der Vet and Nicolaas J. I. Mars. Bottom-up construction of ontologies. *IEEE Trans. on Knowl. and Data Eng.*, 10(4):513–526, July 1998. ISSN 1041-4347. doi: 10.1109/69.706054. URL http://dx.doi.org/10.1109/69.706054. 43

M. van Otterlo. *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for Adaptive Sequential Decision Making Under Uncertainty in First-order and Relational Domains.* Frontiers in artificial intelligence and applications. Ios Press, 2009. ISBN 9781586039691. 24

K. Vaxevanakis, Th. Zahariadis, and N. Vogiatzis. A review on wireless home network technologies. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(2):59–68, April 2003. ISSN 1559-1662. 13

Dinesh Verma. Simplifying network administration using policy-based management. *Netwrk. Mag. of Global Internetwkg.*, 16

(2):20–26, March 2002. ISSN 0890-8044. doi: 10.1109/65.993219. 3

Dinesh C. Verma, Mandis Beigi, and Raymond Jennings. Policy based sla management in enterprise networks. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, POLICY '01, pages 137–152, London, UK, 2001. Springer-Verlag. ISBN 3-540-41610-2. 3, 34

W3C. Owl web ontology language, 2004. URL `http://www.w3.org/TR/owl-guide`. 23, 36, 41, 61, 105

Thomas S. Wallsten, David V. Budescu, and Ido Erev. Evaluating and combining subjective probability estimates. *Journal of Behavioral Decision Making*, pages 243–268, 1997. 148

Wenye Wang, Yi Xu, and Mohit Khanna. Survey paper: A survey on the communication architectures in smart grid. *Computer Networking*, 55(15):3604–3629, October 2011. ISSN 1389-1286. 11

Segev Wasserkrug, Avigdor Gal, and Opher Etzion. A model for reasoning with uncertain rules in event composition systems. *CoRR*, abs/1207.1427, 2012. 40

Glenn Waters, Robert Moore, Andrea Westerinen, and Lee Rafalow. Policy framework architecture. Technical report, IETF Network Working Group, 1999. ix, 19, 20

Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, July 1999. 1

A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. Terminology for policy-based management. Technical report, United States, 2001. 21, 34

Charlie Wilson, Tom Hargreaves, and Richard Hauxwell-Baldwin. Smart homes and their users: a systematic analysis and key challenges. *Personal and Ubiquitous Computing*, pages 1–14, 2014. 1

Ben Wise. Experimentally comparing uncertain inference systems to probability. In *Uncertainty in Artificial Intelligence 2 Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, pages 89–99, Amsterdam, NL, 1986. Elsevier Science. 40, 166

Ben Wise and Max Henrion. A framework for comparing uncertain inference systems to probability. In *Uncertainty in Artificial Intelligence Annual Conference on Uncertainty in Artificial Intelligence (UAI-85)*, pages 69–84, Amsterdam, NL, 1985. Elsevier Science. 40, 157, 166

Ben Wise, Bruce Perrin, David Vaughan, and Robert Yadrick. Evaluation of uncertain inference models iii: The role of tuning. In *Uncertainty in Artificial Intelligence 3 Annual Conference on Uncertainty in Artificial Intelligence (UAI-87)*, pages 55–62, Amsterdam, NL, 1987. Elsevier Science. 40, 166

Tomasz Wiktor Wlodarczyk, Martin J. O'Connor, Chunming Rong, and Mark A. Musen. Swrl-f - a fuzzy logic extension of the semantic web rule language. In Fernando Bobillo, Rommel N. Carvalho, Paulo Cesar G. da Costa, Claudia dAmato, Nicola Fanizzi, Kathryn B. Laskey, Kenneth J. Laskey, Thomas Lukasiewicz, Trevor Martin, Matthias Nickles, and Michael Pool, editors, *URSW*, volume 654 of *CEUR Workshop Proceedings*, pages 97–100. CEUR-WS.org, 2010. 40

Debao Xiao and Hui Xu. An integration of ontology-based and policy-based network management for automation. In *Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce*, pages 27–, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2731-0. doi: 10.1109/CIMCA.2006.43. 36

Xlite. Xlite 4.0, 2006. URL `http://www.counterpath.com/x-lite`. 59, 104

Jeonghwa Yang and W. Keith Edwards. A study on network management tools of householders. In *Proceedings of the 2010 ACM SIGCOMM workshop on Home networks*, HomeNets '10, pages 1–6, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0198-5. 33

Sheng-Yuan Yang and Yi-Yen Chang. An active and intelligent network management system with ontology-based and multi-agent techniques. *Expert Systems with Applications*, 38(8):10320 – 10342, 2011. ISSN 0957-4174. 36

R. Yavatkar. A framework for policy-based admission control, 2000. URL `https://tools.ietf.org/html/rfc2753`. 59

R. Yavatkar, D. Pendarakis, and R. Guerin. RFC 2753: A Framework for Policy-based Admission Control. Technical report, IETF, 2000. URL `www.ietf.org/rfc/rfc2753.txt`. 20

Wang Yuan, Ji Wen, and Wang Jianhui. Design and implementation of inference engine for conflict resolution. In *Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on*, pages 220–223, Jan 2012. 39

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1863103.1863113`. 169

Xingzhi Zhan. Inequalities for the singular values of hadamard products. *SIAM J. Matrix Anal. Appl.*, 18(4): 1093–1095, October 1997. ISSN 0895-4798. doi: 10.1137/S0895479896309645. URL `http://dx.doi.org/10.1137/S0895479896309645`. 171

Appendices

Appendix A

Perl Script to Monitor Packet Queues

Following script is used to monitor sent, dropped and backlog IP packets for "before and after" policy application experiments.

```perl
#!/usr/bin/perl -w
use strict;

my (%sent, %oldsent, %comp, %drop, %olddrop);
while (1) {
  open FILE, "/sbin/tc -s qdisc show dev eth2 |";
  my $queue;
  while (<FILE>) {
    $queue = $1 if ($_ =~ /qdisc ([^:]+):/);
    $sent{$queue} = $1 if ($_ =~ /Sent (\d+) bytes/);
    $comp{$queue} = $1 if ($_ =~ /backlog (\d+)p/);
    $drop{$queue} = $1 if ($_ =~ /dropped (\d+),/);
  }
  close FILE;
  foreach my $q (keys %sent) {
    print "  $q",
          "\t\t",
          scalar ($sent{$q} -
                  ((defined $oldsent{$q}) ? $oldsent{$q} : 0)),
          "\t\t",
          "Backlog: ",
          (defined $comp{$q}) ? $comp{$q} : 0,
          "p\t\tDropped: ",
          scalar ($drop{$q} -
                  ((defined $olddrop{$q}) ? $olddrop{$q} : 0)),
  "\t\t";
  }
  %oldsent=%sent;
  %olddrop=%drop;
  sleep(1);
  print "\n";
}
```

Appendix B

Puppet Recipe for Creation of Packet Queues

Following script shows a puppet recipe for the creation of packet queues.

```
iptables { "rule1":
  task => "tc",
  object => "qdisc",
  command => "add",
  dev => "eth2",
  level => "root",
  handle => "1:",
  queue => "htb",
  number => "12:",
}
iptables { "rule2":
  task => "tc",
  object => "class",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:",
  classid => "1:1",
  queue => "htb",
  rate => "50kbps",
  max_rate => "50kbps",
}
iptables { "rule3":
  task => "tc",
  object => "class",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:1",
  classid => "1:10",
  queue => "htb",
  rate => "50kbps",
  max_rate => "50kbps",
  priority => "1",
}

iptables { "rule4":
  task => "tc",
  object => "class",
```

```
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:1",
  classid => "1:11",
  queue => "htb",
  rate => "50kbps",
  max_rate => "50kbps",
  priority => "2",
}

iptables { "rule5":
  task => "tc",
  object => "class",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:1",
  classid => "1:12",
  queue => "htb",
  rate => "50kbps",
  max_rate => "50kbps",
  priority => "3",
}

iptables { "rule6":
  task => "tc",
  object => "qdisc",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:10",
  handle => "20:",
  queue => "sfq",
  number => "10:",
}

iptables { "rule7":
  task => "tc",
  object => "qdisc",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:11",
  handle => "30:",
  queue => "sfq",
  number => "10:",
}

iptables { "rule8":
  task => "tc",
  object => "qdisc",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:12",
  handle => "40:",
  queue => "sfq",
  number => "10:",
}

iptables { "rule9":
  task => "tc",
  object => "qdisc",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:13",
  handle => "50:",
  queue => "sfq",
```

```
  number => "10:",
}

iptables { "rule10":
  task => "tc",
  object => "filter",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:0",
  priority => "1",
  protocol => "ip",
  filter_type => "u32",
  filter_match => "ip",
  tos => "null",
  classid => "1:10",
}

iptables { "rule11":
  task => "tc",
  object => "filter",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:0",
  priority => "2",
  protocol => "ip",
  filter_type => "u32",
  filter_match => "ip",
  tos => "0x48 0xff",
  classid => "1:11",
}

iptables { "rule12":
  task => "tc",
  object => "filter",
  command => "add",
  dev => "eth2",
  level => "parent",
  parentid => "1:0",
  priority => "3",
  protocol => "ip",
  filter_type => "u32",
  filter_match => "ip",
  tos => "0x68 0xff",
  classid => "1:12",
}

iptables { "rule13":
  task => "iptables",
  table => "mangle",
  chain => "FORWARD",
  iniface => "eth0",
  outiface => "eth2",
  proto => "udp",
  sport => "rtsp",
  dport => "rtsp",
  jump => "TOS",
  tos => "null",
}

iptables { "rule14":
  task => "iptables",
  table => "mangle",
  chain => "FORWARD",
  iniface => "eth0",
  outiface => "eth2",
  proto => "tcp",
  sport => "ftp",
  dport => "ftp",
```

```
  jump => "TOS",
  tos => "0x68 0xff",
}

iptables { "rule15":
  task => "iptables",
  table => "mangle",
  chain => "FORWARD",
  iniface => "eth0",
  outiface => "eth2",
  proto => "udp",
  sport => "sip",
  dport => "sip",
  jump => "TOS",
  tos => "0x48 0xff",
}

iptables { "rule16":
  task => "iptables",
  table => "mangle",
  chain => "FORWARD",
  iniface => "eth0",
  outiface => "eth2",
  proto => "tcp",
  sport => "http",
  dport => "http",
  jump => "TOS",
  tos => "0x68 0xff",
}
```

Appendix C

Ruby Script to Translate Puppet Recipe to IPTables

Following script is used to translate Puppet recipe into IPTables.

```
require "ipaddr"

module Puppet

  @@rules = {}

  @@current_rules = {}

  @@ordered_rules = {}

  @@total_rule_count = 0

  @@instance_count = 0

  @@table_counters = {
    'filter' => 1,
    'nat'    => 1,
    'mangle' => 1,
    'raw'    => 1
  }

  @@usecidr = nil

  @@finalized = false

  # pre and post rules are loaded from files
  # pre.iptables post.iptables in /etc/puppet/iptables
  @@pre_file  = "/etc/puppet/iptables/pre.iptables"
  @@post_file = "/etc/puppet/iptables/post.iptables"
  @@shell_file  = "/etc/QoS/QoS.sh"

  # location where iptables binaries are to be found
  @@iptables_dir = "/sbin"

  # order in which the differents chains appear in iptables-save's output. Used
  # to sort the rules the same way iptables-save does.
  @@chain_order = {
    'PREROUTING'   => 1,
    'INPUT'        => 2,
```

```
    'FORWARD'     => 3,
    'OUTPUT'      => 4,
    'POSTROUTING' => 5,
  }

  newtype(:iptables) do
    @doc = "Manipulate iptables rules"

    newparam(:name) do
      desc "The name of the resource"
      isnamevar
    end

    newparam(:chain) do
      desc "holds value of iptables -A parameter.
                  Possible values are: 'INPUT', 'FORWARD', 'OUTPUT', 'PREROUTING', '
    POSTROUTING'.
                  Default value is 'INPUT'"
      newvalues(:INPUT, :FORWARD, :OUTPUT, :PREROUTING, :POSTROUTING)
      defaultto "INPUT"
    end

    newparam(:table) do
      desc "one of the following tables: 'nat', 'mangle',
                  'filter' and 'raw'. Default one is 'filter'"
      newvalues(:nat, :mangle, :filter, :raw)
      defaultto "filter"
    end

    newparam(:proto) do
      desc "holds value of iptables --protocol parameter.
                  Possible values are: 'tcp', 'udp', 'icmp', 'esp', 'ah', 'vrrp', 'igmp', '
    all'.
                  Default value is 'all'"
      newvalues(:tcp, :udp, :icmp, :esp, :ah, :vrrp, :igmp, :all)
      defaultto "all"
    end

    newparam(:jump) do
      desc "holds value of iptables --jump target
                  Possible values are: 'ACCEPT', 'DROP', 'REJECT', 'DNAT', 'LOG', 'MASQUERADE
    ', 'REDIRECT'."
      newvalues(:ACCEPT, :DROP, :REJECT, :DNAT, :LOG, :MASQUERADE, :REDIRECT, :TOS)
      defaultto "TOS"
    end

    newparam(:source) do
      desc "value for iptables --source parameter"
    end

    newparam(:destination) do
      desc "value for iptables --destination parameter"
    end

    newparam(:sport) do
      desc "holds value of iptables [..] --source-port parameter.
                  If array is specified, values will be passed to multiport module.
                  Only applies to tcp/udp."
      defaultto ""
    end

    newparam(:dport) do
      desc "holds value of iptables [..] --destination-port parameter.
                  If array is specified, values will be passed to multiport module.
                  Only applies to tcp/udp."
      defaultto ""
    end

    newparam(:iniface) do
      desc "value for iptables --in-interface parameter"
```

```
    end

    newparam(:outiface) do
      desc "value for iptables --out-interface parameter"
    end

    newparam(:todest) do
      desc "value for iptables '-j DNAT --to-destination' parameter"
      defaultto ""
    end

    newparam(:reject) do
      desc "value for iptables '-j REJECT --reject-with' parameter"
      defaultto ""
    end

    newparam(:log_level) do
      desc "value for iptables '-j LOG --log-level' parameter"
      defaultto ""
    end

    newparam(:log_prefix) do
      desc "value for iptables '-j LOG --log-prefix' parameter"
      defaultto ""
    end

    newparam(:icmp) do
      desc "value for iptables '-p icmp --icmp-type' parameter"
      defaultto ""
    end

    newparam(:state) do
      desc "value for iptables '-m state --state' parameter.
                 Possible values are: 'INVALID', 'ESTABLISHED', 'NEW', 'RELATED'."
    end

    newparam(:limit) do
      desc "value for iptables '-m limit --limit' parameter.
                 Example values are: '50/sec', '40/min', '30/hour', '10/day'."
      defaultto ""
    end

    newparam(:burst) do
      desc "value for '--limit-burst' parameter.
                 Example values are: '5', '10'."
      defaultto ""
    end

    newparam(:redirect) do
      desc "value for iptables '-j REDIRECT --to-ports' parameter."
      defaultto ""
    end

    newparam(:task) do
      desc "The name of the service"
      newvalues(:tc, :iptables)
      defaultto "iptables"
    end

    newparam(:number) do
      desc "Number of applications/users for quality service"
      defaultto 0
    end

      #Usage: tc [ OPTIONS ] OBJECT { COMMAND | help }
    #where OBJECT := { qdisc | class | filter }
    #OPTIONS := { -s[tatistics] | -d[etails] | -r[aw] }
    newparam(:object) do
      desc "holds value of tc object types.
                 Possible values are: 'qdisc', 'class', 'filter'."
```

```
    newvalues(:qdisc, :class, :filter)
    defaultto "qdisc"
  end

  #Usage: tc qdisc [ add | del | replace | change | get ] dev STRING
  #[ handle QHANDLE ] [ root | ingress | parent CLASSID ]
  #[ estimator INTERVAL TIME_CONSTANT ]
  #[ [ QDISC_KIND ] [ help | OPTIONS ] ]
  newparam(:command) do
    desc "holds value of object command functions.
            Possible values are: 'add', 'del', 'replace', 'change', 'get'."
    newvalues(:add, :del, :replace, :change, :get)
    defaultto "add"
  end

  newparam(:device) do
    desc "network device to which we want attach a queue."
  end

  newparam(:level) do
    desc "indicates that the queue is at what level of network device.
            Possible values are: 'root', 'ingress', 'parent'."
    newvalues(:root, :ingress, :parent)
    defaultto "root"
  end

  newparam(:handle) do
    desc "represents the unique handle that is assigned by the user to the queuing
 discipline."
  end

  #this is incomplete lits of supported queues
  newparam(:queue) do
    desc "indicates queueing discipline, which is able to enqueue and dequeue packets.
            Possible values are: 'fifo', 'prio', 'tbf', 'htb', 'sfq', 'cbq', 'red', '
 gred', 'atm'."
    newvalues(:fifo, :prio, :tbf, :htb, :sfq, :cbq, :red, :gred, :atm)
    defaultto "root"
  end

  newparam(:parentid) do
    desc "represents the handle of the parent queuing discipline."
  end

  newparam(:classid) do
    desc "network device to which we want attach a queue."
  end

  newparam(:rate) do
    desc "defines the minimum threshold value in bytes"
  end

  newparam(:max_rate) do
    desc "defines the maximum threshold value in bytes."
  end

  newparam(:priority) do
    desc "identifies the virtual queue priority."
  end

  newparam(:tos) do
    desc "value of type service marker."
  end

  newparam(:filter_type) do
    desc "to classify (map) packets based on certain properties of the packet."
    newvalues(:rsvp | :u32 | :fw | :route)
    defaultto "u32"
  end
```

```
  newparam(:filter_match) do
    desc "contains definition of the pattern, that will be matched to the currently processed packet."
    defaultto "ip"
  end

  # Parse the output of iptables-save and return a hash with every parameter
  # of each rule
  def load_current_rules(numbered = false)
    if( numbered )
      # reset table counters to 0
      @@table_counters = {
        'filter' => 0,
        'nat'    => 0,
        'mangle' => 0,
        'raw'    => 0
      }
    end

    table         = ''
    loaded_rules  = {}
    table_rules   = {}
    counter       = 1

    `#{@@iptables_dir}/iptables-save`.each { |l|
      if /^\*\S+/.match(l)
        table = self.matched(l.scan(/^\*(\S+)/))

        # init loaded_rules hash
        loaded_rules[table] = {} unless loaded_rules[table]
        table_rules = loaded_rules[table]

        # reset counter
        counter = 1

      elsif /^-A/.match(l)
        # matched rule
        chain = self.matched(l.scan(/^-A (\S+)/))

        table = self.matched(l.scan(/-t (\S+)/))
        table = "filter" unless table

        proto = self.matched(l.scan(/-p (\S+)/))
        proto = "all" unless proto

        jump = self.matched(l.scan(/-j (\S+)/))
        jump = "" unless jump

        source = self.matched(l.scan(/-s (\S+)/))
        if source
          ip = IpCidr.new(source)
          if @@usecidr
            source = ip.cidr
          else
            source = ip.to_s
            source += sprintf("/%s", ip.netmask) unless ip.prefixlen == 32
          end
        end

        destination = self.matched(l.scan(/-d (\S+)/))
        if destination
          ip = IpCidr.new(destination)
          if @@usecidr
            destination = ip.cidr
          else
            destination = ip.to_s
            destination += sprintf("/%s", ip.netmask) unless ip.prefixlen == 32
          end
        end

```

```
      sport = self.matched(l.scan(/--sport[s]? (\S+)/))
      sport = "" unless sport

      dport = self.matched(l.scan(/--dport[s]? (\S+)/))
      dport = "" unless dport

      iniface = self.matched(l.scan(/-i (\S+)/))
      iniface = "" unless iniface

      outiface = self.matched(l.scan(/-o (\S+)/))
      outiface = "" unless outiface

      todest = self.matched(l.scan(/--to-destination (\S+)/))
      todest = "" unless todest

      reject = self.matched(l.scan(/--reject-with (\S+)/))
      reject = "" unless reject

      log_level = self.matched(l.scan(/--log-level (\S+)/))
      log_level = "" unless log_level

      log_prefix = self.matched(l.scan(/--log-prefix (\S+)/))
      log_prefix = "" unless log_prefix

      icmp = self.matched(l.scan(/--icmp-type (\S+)/))
      icmp = "" unless icmp

      state = self.matched(l.scan(/--state (\S+)/))
      state = "" unless state

      limit = self.matched(l.scan(/--limit (\S+)/))
      limit = "" unless limit

      burst = self.matched(l.scan(/--limit-burst (\S+)/))
      burst = "" unless burst

      redirect = self.matched(l.scan(/--to-ports (\S+)/))
      redirect = "" unless redirect

      data = {
        'chain'       => chain,
        'table'       => table,
        'proto'       => proto,
        'jump'        => jump,
        'source'      => source,
        'destination'=> destination,
        'sport'       => sport,
        'dport'       => dport,
        'iniface'     => iniface,
        'outiface'    => outiface,
        'todest'      => todest,
        'reject'      => reject,
        'log_level'   => log_level,
        'log_prefix'  => log_prefix,
        'icmp'        => icmp,
        'state'       => state,
        'limit'       => limit,
        'burst'       => burst,
        'redirect'    => redirect,
      }

      if( numbered )
        table_rules[counter.to_s + " " +l.strip] = data

        # we also set table counters to indicate amount
        # of current rules in each table, that will be needed if
        # we decide to refresh them
        @@table_counters[table] += 1
      else
        table_rules[l.strip] = data
```

```
        end

        counter += 1
      end
    }
    return loaded_rules
  end

  # Small helper used in load_current_rules()
  def matched(data)
    if data.instance_of?(Array)
      data.each { |s|
        if s.instance_of?(Array)
          s.each { |z|
            return z.to_s
          }
        else

          return s.to_s
        end
      }
    end
    nil
  end

  # Fix this function
  def load_rules_from_file(rules, file_name, action)
    if File.exist?(file_name)
      counter = 0
      File.open(file_name, "r") do |infile|
        while (line = infile.gets)
          next unless /^\s*[^\s#]/.match(line.strip)
          table = line[/-t\s+\S+/]
          table = "-t filter" unless table
          table.sub!(/^-t\s+/, '')
          rules[table] = [] unless rules[table]
          rule =
            { 'table'         => table,
              'full rule'     => line.strip,
              'alt rule'      => line.strip}

          if( action == :prepend )
            rules[table].insert(counter, rule)
          else
            rules[table].push(rule)
          end

          counter += 1
        end
      end
    end
  end

  # finalize() gets run once every iptables resource has been declared.
  # It decides if puppet resources differ from currently active iptables
  # rules and applies the necessary changes.
  def finalize
    old = false
    if old
      # sort rules by alphabetical order, grouped by chain, else they arrive in
      # random order and cause puppet to reload iptables rules.

      @@rules.each_key {|key|
        @@rules[key] = @@rules[key].sort_by {|rule| [rule["chain_prio"], rule["name"]] }
      }

      # add numbered version to each rule
      @@table_counters.each_key { |table|
        rules_to_set = @@rules[table]
        if rules_to_set
```

```ruby
          counter = 1
          rules_to_set.each { |rule|
            rule['numbered rule'] = counter.to_s + " "+rule["full rule"]
            rule['altned rule']    = counter.to_s + " "+rule["alt rule"]
            counter += 1
          }
        end
      }

      # On the first round we delete rules which do not match what
      # we want to set. We have to do it in the loop until we
      # exhaust all rules, as some of them may appear as multiple times
      while self.delete_not_matched_rules > 0
      end

      # Now we need to take care of rules which are new or out of order.
      # The way we do it is that if we find any difference with the
      # current rules, we add all new ones and remove all old ones.
      if self.rules_are_different
        # load new rules
        benchmark(:notice, self.noop ? "rules would have changed... (noop)" : "rules have changed...") do
          # load new rules
          @@table_counters.each { |table|
            rules_to_set = @@rules[table]
            if rules_to_set
              rules_to_set.each { |rule_to_set|
                if self.noop
                  debug("Would have run 'iptables -t #{table} #{rule_to_set['alt rule']}' (noop)")
                  next
                else
                  debug("Running 'iptables -t #{table} #{rule_to_set['alt rule']}'")
                  `#{@@iptables_dir}/iptables -t #{table} #{rule_to_set['alt rule']}`
                end
              }
            end
          }

          # delete old rules
          @@table_counters.each { |table|
            current_table_rules = @@current_rules[table]
            if current_table_rules
              current_table_rules.each { |rule, data|
                if self.noop
                  debug("Would have run 'iptables -t #{table} -D #{data['chain']} 1' (noop)")
                  next
                else
                  debug("Running 'iptables -t #{table} -D #{data['chain']} 1'")
                  `#{@@iptables_dir}/iptables -t #{table} -D #{data['chain']} 1`
                end
              }
            end
          }
        end

        @@rules = {}
      end
    else
      # TOS functionality
      f = File.new(@@shell_file, "w+")
      @@table_counters.each { |table|
        rules_to_set = @@rules[table]
        if rules_to_set
          rules_to_set.each { |rule_to_set|

            if self.noop
              debug("Would have run '#{rule_to_set['full rule']}' (noop)")
              next
```

```
            else
              debug("Writing the rule '#{rule_to_set['alt rule']}'")
              if table != "tc"
                f.puts(rule_to_set['full rule'])
                f.puts(rule_to_set['alt rule'])
              else
                f.puts(rule_to_set['full rule'])
              end
            end
          }
        end
        f.close
        debug("Executing the script now...")
        shell_script_output = `sudo chmod +x /etc/QoS/QoS.sh`
        puts shell_script_output
      }
    end
    @@finalized = true
  end

  def finalized?
    if defined? @@finalized
      return @@finalized
    else
      return false
    end
  end

  # Check if at least one rule found in iptables-save differs from what is
  # defined in puppet resources.
  def rules_are_different
    # load current rules
    @@current_rules = self.load_current_rules(true)

    @@table_counters.each_key { |table|
      rules_to_set = @@rules[table]
      current_table_rules = @@current_rules[table]
      current_table_rules = {} unless current_table_rules
      if rules_to_set
        rules_to_set.each { |rule_to_set|
          return true unless current_table_rules[rule_to_set['numbered rule']] or
 current_table_rules[rule_to_set['altned rule']]
        }
      end
    }

    return false
  end

  def delete_not_matched_rules
    # load current rules
    @@current_rules = self.load_current_rules

    # count deleted rules from current active
    deleted = 0;

    # compare current rules with requested set
    @@table_counters.each_key { |table|
      rules_to_set = @@rules[table]
      current_table_rules = @@current_rules[table]
      if rules_to_set
        if current_table_rules
          rules_to_set.each { |rule_to_set|
            full_rule = rule_to_set['full rule']
            alt_rule  = rule_to_set['alt rule']
            if    current_table_rules[full_rule]
              current_table_rules[full_rule]['keep'] = 'me'
            elsif current_table_rules[alt_rule]
              current_table_rules[alt_rule]['keep']  = 'me'
            end
```

```
          }
        end
      end

      # delete rules not marked with "keep" => "me"
      if current_table_rules
        current_table_rules.each { |rule, data|
          if data['keep']
          else
            if self.noop
              debug("Would have run 'iptables -t #{table} #{rule.sub('-A', '-D')}' (noop)")
              next
            else
              debug("Running 'iptables -t #{table} #{rule.sub('-A', '-D')}'")
              `#{@@iptables_dir}/iptables -t #{table} #{rule.sub("-A", "-D")}`
            end
            deleted += 1
          end
        }
      end
    }
    return deleted
  end

  def evaluate
    @@ordered_rules[self.name] = @@instance_count
    @@instance_count += 1

    if @@instance_count == @@total_rule_count
      self.finalize unless self.finalized?
    end
    return super
  end

  # Reset class variables to their initial value
  def self.clear
    @@rules = {}

    @@current_rules = {}

    @@ordered_rules = {}

    @@total_rule_count = 0

    @@instance_count = 0

    @@table_counters = {
      'filter' => 1,
      'nat'    => 1,
      'mangle' => 1,
      'raw'    => 1
    }

    @@finalized = false
    super
  end


  def initialize(args)
    super(args)

    if @@usecidr == nil
      iptablesversion = `#{@@iptables_dir}/iptables --version`.scan(/ v([0-9\.]+)/)
      iptablesversion = iptablesversion[0][0].split(".")
      if iptablesversion[0].to_i < 2 and iptablesversion[1].to_i < 4
        @@usecidr = false
      else
        @@usecidr = true
      end
    end
```

```
    invalidrule = false
    @@total_rule_count += 1

    full_string = ""

    if value(:task).to_s == "iptables"
      full_string = "iptables "
      table = value(:table).to_s
      @@rules[table] = [] unless @@rules[table]

      if value(:table).to_s == "filter" and ["PREROUTING", "POSTROUTING"].include?(value(:chain).to_s)
        invalidrule = true
        err("PREROUTING and POSTROUTING cannot be used in table 'filter'. Ignoring rule.")
      elsif  value(:table).to_s == "nat" and ["INPUT", "FORWARD"].include?(value(:chain).to_s)
        invalidrule = true
        err("INPUT and FORWARD cannot be used in table 'nat'. Ignoring rule.")
      elsif  value(:table).to_s == "raw" and ["INPUT", "FORWARD", "POSTROUTING"].include?(value(:chain).to_s)
        invalidrule = true
        err("INPUT, FORWARD and POSTROUTING cannot be used in table 'raw'. Ignoring rule.")
      else
        iptables += "-t " + value(:table).to_s + " "
        full_string += "-A " + value(:chain).to_s
      end

      source = value(:source).to_s
      if source != ""
        ip = IpCidr.new(source)
        if @@usecidr
          source = ip.cidr
        else
          source = ip.to_s
          source += sprintf("/%s", ip.netmask) unless ip.prefixlen == 32
        end
        full_string += " -s " + source
      end

      destination = value(:destination).to_s
      if destination != ""
        ip = IpCidr.new(destination)
        if @@usecidr
          destination = ip.cidr
        else
          destination = ip.to_s
          destination += sprintf("/%s", ip.netmask) unless ip.prefixlen == 32
        end
        full_string += " -d " + destination
      end

      if value(:iniface).to_s != ""
        if ["INPUT", "FORWARD", "PREROUTING"].include?(value(:chain).to_s)
          full_string += " -i " + value(:iniface).to_s
        else
          invalidrule = true
          err("--in-interface only applies to INPUT/FORWARD/PREROUTING. Ignoring rule.")
        end
      end
      if value(:outiface).to_s != ""
        if ["OUTPUT", "FORWARD", "POSTROUTING"].include?(value(:chain).to_s)
          full_string += " -o " + value(:outiface).to_s
        else
          invalidrule = true
          err("--out-interface only applies to OUTPUT/FORWARD/POSTROUTING. Ignoring rule.")
        end
      end

      alt_string  = full_string
```

```
if value(:proto).to_s != "all"
  alt_string  += " -p " + value(:proto).to_s
  full_string += " -p " + value(:proto).to_s
  if not ["vrrp", "igmp"].include?(value(:proto).to_s)
    alt_string += " -m " + value(:proto).to_s
  end
end

if value(:dport).to_s != ""
  if ["tcp", "udp"].include?(value(:proto).to_s)
    if value(:dport).class.to_s == "Array"
      if value(:dport).length <= 15
        full_string += " -m multiport --dports " + value(:dport).join(",")
        alt_string += " -m multiport --dports " + value(:dport).join(",")
      else
        invalidrule = true
        err("multiport module only accepts <= 15 ports. Ignoring rule.")
      end
    else
      full_string += " --dport " + value(:dport).to_s
      alt_string  += " --dport " + value(:dport).to_s
    end
  else
    invalidrule = true
    err("--destination-port only applies to tcp/udp. Ignoring rule.")
  end
end
if value(:sport).to_s != ""
  if ["tcp", "udp"].include?(value(:proto).to_s)
    if value(:sport).class.to_s == "Array"
      if value(:sport).length <= 15
        full_string += " -m multiport --sports " + value(:sport).join(",")
        alt_string += " -m multiport --sports " + value(:sport).join(",")
      else
        invalidrule = true
        err("multiport module only accepts <= 15 ports. Ignoring rule.")
      end
    else
      full_string += " --sport " + value(:sport).to_s
      alt_string  += " --sport " + value(:sport).to_s
    end
  else
    invalidrule = true
    err("--source-port only applies to tcp/udp. Ignoring rule.")
  end
end

if value(:icmp).to_s != ""
  if value(:proto).to_s != "icmp"
    invalidrule = true
    err("--icmp-type only applies to icmp. Ignoring rule.")
  else
    full_string += " --icmp-type " + value(:icmp).to_s
    alt_string += " --icmp-type " + value(:icmp).to_s
  end
end

# let's specify the order of the states as iptables uses them
state_order = ["INVALID", "NEW", "RELATED", "ESTABLISHED"]
if value(:state).class.to_s == "Array"

  invalid_state = false
  value(:state).each {|v|
    invalid_state = true unless state_order.include?(v)
  }

  if value(:state).length <= state_order.length and not invalid_state

    # return only the elements that appear in both arrays.
```

```
            # This filters out bad entries (unfortunately silently), and orders the entries
            # in the same order as the 'state_order' array
            states = state_order & value(:state)

            full_string += " -m state --state " + states.join(",")
            alt_string  += " -m state --state " + states.join(",")
          else
            invalidrule = true
            err("'state' accepts any the following states: #{state_order.join(", ")}. Ignoring rule.")
          end
        elsif value(:state).to_s != ""
          if state_order.include?(value(:state))
            full_string += " -m state --state " + value(:state).to_s
            alt_string  += " -m state --state " + value(:state).to_s
          else
            invalidrule = true
            err("'state' accepts any the following states: #{state_order.join(", ")}. Ignoring rule.")
          end
        end

        if value(:limit).to_s != ""
          limit_value = value(:limit).to_s
          if not limit_value.include? "/"
            invalidrule = true
            err("Please append a valid suffix (sec/min/hour/day) to the value passed to 'limit'. Ignoring rule.")
          else
            limit_value = limit_value.split("/")
            if limit_value[0] !~ /^[0-9]+$/
              invalidrule = true
              err("'limit' values must be numeric. Ignoring rule.")
            elsif ["sec", "min", "hour", "day"].include? limit_value[1]
              full_string += " -m limit --limit " + value(:limit).to_s
              alt_string += " -m limit --limit " + value(:limit).to_s
            else
              invalidrule = true
              err("Please use only sec/min/hour/day suffixes with 'limit'. Ignoring rule.")
            end
          end
        end

        if value(:burst).to_s != ""
          if value(:limit).to_s == ""
            invalidrule = true
            err("'burst' makes no sense without 'limit'. Ignoring rule.")
          elsif value(:burst).to_s !~ /^[0-9]+$/
            invalidrule = true
            err("'burst' accepts only numeric values. Ignoring rule.")
          else
            full_string += " --limit-burst " + value(:burst).to_s
            alt_string += " --limit-burst " + value(:burst).to_s
          end
        end

        full_string += " -j " + value(:jump).to_s

        if value(:task).to_s == "iptables"
          alt_string += " -j RETURN"
        else
          alt_string += " -j " + value(:jump).to_s
        end

        full_string += "--set-tos " + value(:tos).to_s + " "

        if value(:jump).to_s == "DNAT"
          if value(:table).to_s != "nat"
            invalidrule = true
            err("DNAT only applies to table 'nat'.")
```

```
        elsif value(:todest).to_s == ""
          invalidrule = true
          err("DNAT missing mandatory 'todest' parameter.")
        else
          full_string += " --to-destination " + value(:todest).to_s
          alt_string += " --to-destination " + value(:todest).to_s
        end
      elsif value(:jump).to_s == "REJECT"
        if value(:reject).to_s != ""
          full_string += " --reject-with " + value(:reject).to_s
          alt_string += " --reject-with " + value(:reject).to_s
        end
      elsif value(:jump).to_s == "LOG"
        if value(:log_level).to_s != ""
          full_string += " --log-level " + value(:log_level).to_s
          alt_string += " --log-level " + value(:log_level).to_s
        end
        if value(:log_prefix).to_s != ""
          # --log-prefix has a 29 characters limitation.
          log_prefix = "\"" + value(:log_prefix).to_s[0,27] + ": \""
          full_string += " --log-prefix " + log_prefix
          alt_string += " --log-prefix " + log_prefix
        end
      elsif value(:jump).to_s == "MASQUERADE"
        if value(:table).to_s != "nat"
          invalidrule = true
          err("MASQUERADE only applies to table 'nat'.")
        end
      elsif value(:jump).to_s == "REDIRECT"
        if value(:redirect).to_s != ""
          full_string += " --to-ports " + value(:redirect).to_s
          alt_string += " --to-ports " + value(:redirect).to_s
        end
      end

      chain_prio = @@chain_order[value(:chain).to_s]
    elsif value(:task).to_s == "tc"
      full_string += "tc "
      if value(:object).to_s != ""
        if value(:object).to_s == "qdisc"
          full_string += "qdisc "
        elsif value(:object).to_s == "class"
          full_string += "class "
        elsif value(:object).to_s == "filter"
          full_string += "filter "
        end
        if value(:commad).to_s == "add"
          full_string += "add "
          if value(:device).to_s != ""
            full_string += "dev " + value(:device).to_s + " "
            if value(:level).to_s != ""
              if value(:level).to_s == "root"
                full_string += "root "
              elsif value(:level).to_s == "parent"
                full_string += "parent " + value(:parentid).to_s + " "
              end
              if value(:handle).to_s != ""
                  full_string += "handle " + value(:handle).to_s + " "
              end
              if value(:queue).to_s != "" and value(:object).to_s != "class"
                full_string += value(:queue).to_s + " "
              end
              if value(:level).to_s == "root"
                full_string += "default " + value(:number).to_s
              elsif value(:level).to_s == "parent" and value(:object).to_s != "class"
                full_string += "perturb " + value(:number).to_s
              end
            end
          end
          if value(:object).to_s == "class"
```

```
        if value(:classid).to_s != ""
          full_string += "classid " + value(:classid).to_s + " "
        end
        if value(:queue).to_s != "" and value(:object).to_s != "class"
          full_string += value(:queue).to_s
        end
        if value(:rate).to_s != ""
          full_string += "rate " + value(:rate).to_s + " "
          if value(:max_rate).to_s != ""
            full_string += "ceil " + value(:max_rate).to_s + " "
          end
        end
      end
      if value(:object).to_s != "qdisc"
        if value(:priority).to_s != ""
          full_string += "prio " + value(:priority).to_s + " "
        end
        if value(:object).to_s != "class"
          if value(:proto).to_s != ""
            full_string += "protocol " + value(:proto).to_s + " "
          end
          if value(:filter_type).to_s != ""
            full_string += value(:filter_type).to_s + " "
          end
          if value(:filter_match).to_s != ""
            full_string += "match " + value(:filter_match).to_s + " "
          end
          if value(:tos).to_s != ""
            full_string += "tos " + value(:tos).to_s + " "
          end
          if value(:classid).to_s != ""
            full_string += "classid " + value(:classid).to_s
          end
        end
      end
    end
  end
end
debug("iptables param: #{full_string}")

if invalidrule != true and value(:task).to_s == "iptables"
  @@rules[table].
    push({ 'name'            => value(:name).to_s,
           'chain'           => value(:chain).to_s,
           'task'            => value(:task).to_s,
           'table'           => value(:table).to_s,
           'proto'           => value(:proto).to_s,
           'jump'            => value(:jump).to_s,
           'source'          => value(:source).to_s,
           'destination'     => value(:destination).to_s,
           'sport'           => value(:sport).to_s,
           'dport'           => value(:dport).to_s,
           'iniface'         => value(:iniface).to_s,
           'outiface'        => value(:outiface).to_s,
           'tos'             => value(:tos).to_s,
           'todest'          => value(:todest).to_s,
           'reject'          => value(:reject).to_s,
           'redirect'        => value(:redirect).to_s,
           'log_level'       => value(:log_level).to_s,
           'log_prefix'      => value(:log_prefix).to_s,
           'icmp'            => value(:icmp).to_s,
           'state'           => value(:state).to_s,
           'limit'           => value(:limit).to_s,
           'burst'           => value(:burst).to_s,
           'chain_prio'      => chain_prio.to_s,
           'full rule'       => full_string,
           'alt rule'        => alt_string})
elsif if invalidrule != true and value(:task).to_s == "tc"
    table = "tc"
    @@rules[table].
```

```
          push({ 'name'          => value(:name).to_s,
                 'task'          => value(:task).to_s,
                 'object'        => value(:object).to_s,
                 'command'       => value(:command).to_s,
                 'device'        => value(:device).to_s,
                 'level'          => value(:level).to_s,
                 'handle'        => value(:handle).to_s,
                 'queue'         => value(:queue).to_s,
                 'parentid'      => value(:parentid).to_s,
                 'classid'       => value(:classid).to_s,
                 'rate'          => value(:rate).to_s,
                 'max_rate'      => value(:max_rate).to_s,
                 'priority'      => value(:priority).to_s,
                 'tos'           => value(:tos).to_s,
                 'filter_type'   => value(:filter_type).to_s,
                 'filter_match'  => value(:log_level).to_s,
                 'number'        => value(:number).to_s,
                 'full rule'     => full_string})
      end
    end
  end
end


class IpCidr < IPAddr

  def netmask
    _to_string(@mask_addr)
  end

  def prefixlen
    m = case @family
    when Socket::AF_INET
      IN4MASK
    when Socket::AF_INET6
      IN6MASK
    else
      raise "unsupported address family"
    end
    return $1.length if /\A(1*)(0*)\z/ =~ (@mask_addr & m).to_s(2)
    raise "bad addr_mask format"
  end

  def cidr
    cidr = sprintf("%s/%s", self.to_s, self.prefixlen)
    cidr
  end

end
```

Appendix D

Unparsed and Parsed Monitoring Data Collected from Router

Following is unparsed monitoring data collected from Router.

```
17:22:21.816251 IP (tos 0x28, ttl 127, id 41740, offset 0, flags [DF], proto TCP (6), length 48) 10.37.2.253.3864 > 163.5.255.9.65458: S, cksum 0xa704 (correct), 814017946:814017946(0) win 65535 <mss 1460,nop,nop,sackOK>
17:22:21.816761 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 200) 10.37.84.53.15726 > 10.37.2.253.34722: UDP, length 172
17:22:21.833184 IP (tos 0x28, ttl 127, id 41741, offset 0, flags [none], proto UDP (17), length 200) 10.37.2.253.34722 > 10.37.84.53.15726: UDP, length 172
17:22:21.836582 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 200) 10.37.84.53.15726 > 10.37.2.253.34722: UDP, length 172
17:22:21.839172 IP (tos 0xa0, ttl 51, id 61276, offset 0, flags [DF], proto TCP (6), length 48) 163.5.255.9.65458 > 10.37.2.253.3864: S, cksum 0x9387 (correct), 3598925289:3598925289(0) ack 814017947 win 65535 <mss 1460,sackOK,eol>
17:22:21.839449 IP (tos 0x28, ttl 127, id 41742, offset 0, flags [DF], proto TCP (6), length 40) 10.37.2.253.3864 > 163.5.255.9.65458: ., cksum 0xbf4a (correct), 1:1(0) ack 1 win 65535
17:22:21.839517 IP (tos 0x28, ttl 127, id 41743, offset 0, flags [DF], proto TCP (6), length 46) 10.37.2.253.3043 > 163.5.255.9.21: P, cksum 0x6e96 (correct), 50760:50766(6) ack 132647 win 65110
17:22:21.853279 IP (tos 0x28, ttl 127, id 41744, offset 0, flags [none], proto UDP (17), length 200) 10.37.2.253.34722 > 10.37.84.53.15726: UDP, length 172
17:22:21.856823 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 200) 10.37.84.53.15726 > 10.37.2.253.34722: UDP, length 172
17:22:21.862620 IP (tos 0xa0, ttl 51, id 61278, offset 0, flags [DF], proto TCP (6), length 118) 163.5.255.9.65458 > 10.37.2.253.3864: P 1:79(78) ack 1 win 65535
17:22:21.862705 IP (tos 0x80, ttl 51, id 61277, offset 0, flags [DF], proto TCP (6), length 79) 163.5.255.9.21 > 10.37.2.253.3043: P 132647:132686(39) ack 50766 win 65535
17:22:21.862724 IP (tos 0xa0, ttl 51, id 61279, offset 0, flags [DF], proto TCP (6), length 40) 163.5.255.9.65458 > 10.37.2.253.3864: F, cksum 0xbefb (correct), 79:79(0) ack 1 win 65535
17:22:21.863064 IP (tos 0x28, ttl 127, id 41745, offset 0, flags [DF], proto TCP (6), length 40) 10.37.2.253.3864 > 163.5.255.9.65458: ., cksum 0xbf49 (correct), 1:1(0) ack 80 win 65457
17:22:21.863629 IP (tos 0x28, ttl 127, id 41746, offset 0, flags [DF], proto TCP (6), length 40) 10.37.2.253.3864 > 163.5.255.9.65458: F, cksum 0xbf48 (correct), 1:1(0) ack 80 win 65457
17:22:21.873268 IP (tos 0x28, ttl 127, id 41747, offset 0, flags [none], proto UDP (17), length 200) 10.37.2.253.34722 > 10.37.84.53.15726: UDP, length 172
17:22:21.876545 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 200) 10.37.84.53.15726 > 10.37.2.253.34722: UDP, length 172
```

```
17:22:21.886666 IP (tos 0x80, ttl 51, id 61280, offset 0, flags [DF], proto TCP (6), length
     64) 163.5.255.9.21 > 10.37.2.253.3043: P, cksum 0x53dc (correct), 132686:132710(24) ack
     50766 win 65535
17:22:21.886829 IP (tos 0xa0, ttl 51, id 61281, offset 0, flags [DF], proto TCP (6), length
     40) 163.5.255.9.65458 > 10.37.2.253.3864: ., cksum 0xbefb (correct), 80:80(0) ack 2 win
     65534
17:22:21.887133 IP (tos 0x28, ttl 127, id 41748, offset 0, flags [DF], proto TCP (6), length
     40) 10.37.2.253.3043 > 163.5.255.9.21: ., cksum 0x1b46 (correct), 50766:50766(0) ack
     132710 win 65047
17:22:21.891457 IP (tos 0x28, ttl 127, id 41749, offset 0, flags [DF], proto TCP (6), length
     131) 10.37.2.253.3043 > 163.5.255.9.21: P 50766:50857(91) ack 132710 win 65047
17:22:21.893419 IP (tos 0x28, ttl 127, id 41750, offset 0, flags [none], proto UDP (17),
     length 200) 10.37.2.253.34722 > 10.37.84.53.15726: UDP, length 172
17:22:21.896418 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 200)
     10.37.84.53.15726 > 10.37.2.253.34722: UDP, length 172
17:22:21.915364 IP (tos 0x80, ttl 51, id 61282, offset 0, flags [DF], proto TCP (6), length
     77) 163.5.255.9.21 > 10.37.2.253.3043: P 132710:132747(37) ack 50857 win 65535
17:22:21.916073 IP (tos 0x28, ttl 127, id 41751, offset 0, flags [DF], proto TCP (6), length
     45) 10.37.2.253.3043 > 163.5.255.9.21: P, cksum 0x7c79 (correct), 50857:50862(5) ack
     132747 win 65010
17:22:21.916788 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 200)
     10.37.84.53.15726 > 10.37.2.253.34722: UDP, length 172
17:22:21.917933 IP (tos 0x28, ttl 127, id 41752, offset 0, flags [none], proto UDP (17),
     length 200) 10.37.2.253.34722 > 10.37.84.53.15726: UDP, length 172
17:22:21.936720 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 200)
     10.37.84.53.15726 > 10.37.2.253.34722: UDP, length 172
17:22:21.938044 IP (tos 0x28, ttl 127, id 41753, offset 0, flags [none], proto UDP (17),
     length 200) 10.37.2.253.34722 > 10.37.84.53.15726: UDP, length 172
17:22:21.939510 IP (tos 0x80, ttl 51, id 61283, offset 0, flags [DF], proto TCP (6), length
     132) 163.5.255.9.21 > 10.37.2.253.3043: P 132747:132839(92) ack 50862 win 65535
17:22:21.940012 IP (tos 0x28, ttl 127, id 41754, offset 0, flags [DF], proto TCP (6), length
     51) 10.37.2.253.3043 > 163.5.255.9.21: P, cksum 0x3e6c (correct), 50862:50873(11) ack
     132839 win 64918
17:22:21.955830 IP (tos 0xc0, ttl 1, id 0, offset 0, flags [none], proto UDP (17), length 48)
      10.37.2.249.1985 > 224.0.0.2.1985: [udp sum ok] HSRPv0-hello 20: state=active group=32
     addr=10.37.2.251 hellotime=1s holdtime=3s priority=200 auth="cisco^@^@^@"
17:22:21.956515 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 200)
     10.37.84.53.15726 > 10.37.2.253.34722: UDP, length 172
```

Following is parsed monitoring data collected from Router.


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Packets>
    <Packet>
        <Time>17:22:21.816251</Time>
        <Tos>0x28</Tos>
        <Id>41740</Id>
        <Protocol>TCP</Protocol>
        <Source_IP>10.37.2.253.3</Source_IP>
        <Source_Port>3864</Source_Port>
        <Destination_IP>163.5.255.9.6</Destination_IP>
        <Destination_Port>65458</Destination_Port>
    </Packet>
    <Packet>
        <Time>17:22:21.833184</Time>
        <Tos>0x28</Tos>
        <Id>41741</Id>
        <Protocol>UDP</Protocol>
        <Source_IP>10.37.2.253.3</Source_IP>
        <Source_Port>34722</Source_Port>
        <Destination_IP>10.37.84.53.1</Destination_IP>
        <Destination_Port>15726</Destination_Port>
    </Packet>
    <Packet>
        <Time>17:22:21.839172</Time>
        <Tos>0xa0</Tos>
```

```
        <Id>61276</Id>
        <Protocol>TCP</Protocol>
        <Source_IP>163.5.255.9.6</Source_IP>
        <Source_Port>65458</Source_Port>
        <Destination_IP>10.37.2.253.3</Destination_IP>
        <Destination_Port>3864</Destination_Port>
    </Packet>
    <Packet>
        <Time>17:22:21.839449</Time>
        <Tos>0x28</Tos>
        <Id>41742</Id>
        <Protocol>TCP</Protocol>
        <Source_IP>10.37.2.253.3</Source_IP>
        <Source_Port>3864</Source_Port>
        <Destination_IP>163.5.255.9.6</Destination_IP>
        <Destination_Port>65458</Destination_Port>
    </Packet>
    <Packet>
        <Time>17:22:21.839517</Time>
        <Tos>0x28</Tos>
        <Id>41743</Id>
        <Protocol>TCP</Protocol>
        <Source_IP>10.37.2.253.3</Source_IP>
        <Source_Port>3043</Source_Port>
        <Destination_IP>163.5.255.9.2</Destination_IP>
        <Destination_Port>21</Destination_Port>
    </Packet>
    <Packet>
        <Time>17:22:21.853279</Time>
        <Tos>0x28</Tos>
        <Id>41744</Id>
        <Protocol>UDP</Protocol>
        <Source_IP>10.37.2.253.3</Source_IP>
        <Source_Port>34722</Source_Port>
        <Destination_IP>10.37.84.53.1</Destination_IP>
        <Destination_Port>15726</Destination_Port>
    </Packet>
```

Appendix E

Network Activity Log

Following is network activity log showing what devices are connected to Home network. This information is used to detect new devices connecting to the network for our experimentations.

```
arana@puppet-client:~$ arp -a
puppet-server.local (192.168.22.3) at 00:15:c5:24:e3:ac [ether] on eth1
puppet (192.168.22.2) at 00:25:64:49:ee:38 [ether] on eth1
Annie-htc(192.168.22.4) at 7C:71:93:00:15:C2 [wifi] on wlan0
? (10.37.2.251) at 00:00:0c:07:ac:20 [ether] on eth0
```

Following is network activity log showing what websites are being visited by different devices (users) for our experiments.

```
arana@puppet-client:~$ sudo tshark -i eth0 -nn -e frame.date -e frame.time -e ip.src -f 'port
    80' -l -t ad -n -R 'http.request' -T fields -e http.host
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
  Jul 25, 2011 19:09:15.201387000 10.37.2.190 static.bbc.co.uk
  Jul 25, 2011 19:09:15.201787000 10.37.2.190 www.bbc.co.uk
  Jul 25, 2011 19:09:15.209476000 10.37.2.190 static.bbc.co.uk
  Jul 25, 2011 19:09:15.381112000 10.37.2.190 static.bbc.co.uk
  Jul 25, 2011 19:09:15.393457000 10.37.2.190 static.bbc.co.uk
  Jul 25, 2011 19:09:15.413320000 10.37.2.190 js.revsci.net
  Jul 25, 2011 19:09:15.421587000 10.37.2.190 static.bbc.co.uk
  Jul 25, 2011 19:09:15.444972000 10.37.2.190 static.bbc.co.uk
  Jul 25, 2011 19:09:15.542968000 10.37.2.190 node1.bbcimg.co.uk
  Jul 25, 2011 19:09:15.700906000 10.37.2.190 static.bbc.co.uk
  Jul 25, 2011 19:09:15.804782000 10.37.2.190 sa.bbc.co.uk
  Jul 25, 2011 19:09:15.811104000 10.37.2.190 ad.doubleclick.net
  Jul 25, 2011 19:09:15.877124000 10.37.2.190 sa.bbc.co.uk
  Jul 25, 2011 19:09:15.954675000 10.37.2.190 static.bbc.co.uk
```

Following is network activity log showing the location proximity of devices for our experiments.

```
First Floor : Kitchen
arana@puppet-client:~$ iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: B4:14:89:1B:4E:D0
                    Channel:8
                    Frequency:2.447 GHz (Channel 8)
                    Quality=62/70  Signal level=-48 dBm
                    Encryption key:on
                    ESSID:"TSSG"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                              11 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=00000cf7b80cd178
                    Extra: Last beacon: 24416ms ago
                    IE: Unknown: 000454535347
                    IE: Unknown: 010882848B0C12961824
                    IE: Unknown: 030108
                    IE: Unknown: 2A0100
                    IE: Unknown: 2D1A6E181BFFFF000000000000000000000000000000000000000000
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: 32043048606C
                    IE: Unknown: 3D160800070000000000000000000000000000000000000000
                    IE: Unknown: 851
    E0B0095000F00FF0319003131343241474E303200000000000000002000036
                    IE: Unknown: DD180050F2020101870003A4000027A4000042435E0062320000
                    IE: Unknown: DD06004096010103
                    IE: Unknown: DD050040960305
                    IE: Unknown: DD050040960B09
                    IE: Unknown: DD050040961401
First Floor : Communal Area
wlan0     Scan completed :
          Cell 01 - Address: B4:14:89:1B:4E:D0
                    Channel:8
                    Frequency:2.447 GHz (Channel 8)
                    Quality=57/70  Signal level=-53 dBm
                    Encryption key:on
                    ESSID:"TSSG"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                              11 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=00000cf9496d1177
                    Extra: Last beacon: 8068ms ago
                    IE: Unknown: 000454535347
                    IE: Unknown: 010882848B0C12961824
                    IE: Unknown: 030108
                    IE: Unknown: 2A0100
                    IE: Unknown: 2D1A6E181BFFFF000000000000000000000000000000000000000000
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: 32043048606C
                    IE: Unknown: 3D160800070000000000000000000000000000000000000000
                    IE: Unknown: 851
    E0D0095000F00FF0319003131343241474E303200000000000000002000036
                    IE: Unknown: DD180050F2020101870003A4000027A4000042435E0062320000
                    IE: Unknown: DD06004096010103
                    IE: Unknown: DD050040960305
                    IE: Unknown: DD050040960B09
                    IE: Unknown: DD050040961401
```

```
arana@puppet-client:~$
Second Floor: Room R2.102
arana@puppet-client:~$ iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: B4:14:89:1B:59:F0
                    Channel:4
                    Frequency:2.427 GHz (Channel 4)
                    Quality=34/70  Signal level=-76 dBm
                    Encryption key:on
                    ESSID:"TSSG"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                              11 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=000010f663686177
                    Extra: Last beacon: 9784ms ago
                    IE: Unknown: 000454535347
                    IE: Unknown: 010882848B0C12961824
                    IE: Unknown: 030104
                    IE: Unknown: 2A0100
                    IE: Unknown: 2D1A6E181BFFFF000000000000000000000000000000000000000000
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: 32043048606C
                    IE: Unknown: 3D16040007000000000000000000000000000000000000000000
                    IE: Unknown: 851
    E0C0095000F00FF0319003131343241474E30340000000000000007000036
                    IE: Unknown: DD180050F2020101870003A4000027A4000042435E0062320000
                    IE: Unknown: DD06004096010103
                    IE: Unknown: DD050040960305
                    IE: Unknown: DD050040960B09
                    IE: Unknown: DD050040961401

Second Floor : Room R2.104
wlan0     Scan completed :
          Cell 01 - Address: B4:14:89:1B:59:F0
                    Channel:4
                    Frequency:2.427 GHz (Channel 4)
                    Quality=37/70  Signal level=-73 dBm
                    Encryption key:on
                    ESSID:"TSSG"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                              11 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=000010f6c1b68178
                    Extra: Last beacon: 56864ms ago
                    IE: Unknown: 000454535347
                    IE: Unknown: 010882848B0C12961824
                    IE: Unknown: 030104
                    IE: Unknown: 2A0100
                    IE: Unknown: 2D1A6E181BFFFF000000000000000000000000000000000000000000
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: 32043048606C
                    IE: Unknown: 3D16040007000000000000000000000000000000000000000000
                    IE: Unknown: 851
    E120095000F00FF0319003131343241474E3034000000000000000A000036
                    IE: Unknown: DD180050F2020101870003A4000027A4000042435E0062320000
                    IE: Unknown: DD06004096010103
                    IE: Unknown: DD050040960305
                    IE: Unknown: DD050040960B09
                    IE: Unknown: DD050040961401
```

Index