



**João Paulo
Silva Barraca**

**Estabelecimento de Redes de Comunidades
Sobreponíveis**

Establishment of Stackable Community Networks



**João Paulo
Silva Barraca**

**Estabelecimento de Redes de Comunidades
Sobreponíveis**

Establishment of Stackable Community Networks

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Informática, realizada sob a orientação científica do Doutor Rui Luis de Andrade Aguiar, Professor Associado com Agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Trabalho co-financiado pela FCT através do programa POCTI 2010, do programa POS_C do III Quadro Comunitário de Apoio, e pelo Orçamento de Estado do MCTES, através de bolsa com referência SFRH/BD/32548/2006.

Dedico este trabalho à minha esposa, Rosa, pelo apoio e encorajamento que sempre prestou; à minha filha, Catarina, que adicionou uma dimensão à minha vida; aos meus pais, Noémia e João, pelo esforço que tiveram para que esta tese fosse possível.

o júri / the jury

presidente / president

Doutora Maria Hermínia Deulonder Correia Amado Laurel
Professora Catedrática da Universidade de Aveiro

vogais / examiners committee

Doutor Joaquim Arnaldo Carvalho Martins
Professor Catedrático da Universidade de Aveiro

Doutor Rui Luis de Andrade Aguiar
Professor Associado com Agregação da Universidade de Aveiro (Orientador)

Doutora Teresa Maria Sá Ferreira Vazão Vasques
Professora Associada do Instituto Superior Técnico da Universidade Técnica de Lisboa

Doutor Paulo Alexandre Ferreira Simões
Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

agradecimentos / acknowledgements

Gostaria de agradecer a ajuda de todos os colegas do grupo de investigação (HNG/ATNoG) onde estou inserido no Instituto de Telecomunicações - pólo de Aveiro. Este grupo forneceu-me as oportunidades adequadas para a realização deste trabalho, e para a colaboração com outros grupos de investigação a nível internacional. Fruto disto é a presença de parte deste trabalho nos resultados de projetos internacionais, assim como a sempre construtiva discussão que pude obter junto de um grande número de investigadores influentes na área.

Agradeço ao meu orientador, o Professor Doutor Rui L. Aguiar, sem o qual seria impossível realizar esta tese. O seu empenho na orientação, suporte, discussão e disseminação dos conceitos abordados foram essenciais para a sua elaboração.

Agradeço à Fundação para a Ciência e Tecnologia pelo financiamento que me atribuiu através de uma bolsa de doutoramento.

Agradeço ao João Martins pela dedicação na instanciação de um painel de administração para a plataforma experimental, ao Filipe Ferreira por ter explorado as questões de conectividade em redes sem fios não planeadas, e ao Pedro Fernandes pelo trabalho relativo à descoberta de serviços em comunidades. Estes trabalhos, presentes nas suas dissertações de mestrado, forneceram um contributo importante para o enquadramento desta tese.

Agradeço ao Bruno Santos, ao João Dias, e ao Pedro Francisco pelo contributo em componentes iniciais da plataforma de gestão no contexto do projeto IST-WIP; ao Carlos Gonçalves pelo apoio na manutenção da plataforma experimental; ao Alexandre Brito pelo contributo na criação de interfaces de rede remotos.

Agradeço ao Eduardo Rocha pela revisão cuidada que realizou ao documento final.

Agradeço à minha família que sempre acreditou na realização desta tese, tendo-me continuamente apoiado e incentivado. Uma tese de doutoramento é um caminho longo e por vezes desconhecido, o apoio que prestaram foi imprescindível.

Por fim, agradeço a todas as outras pessoas que direta ou indiretamente forneceram críticas construtivas, e que de alguma forma contribuíram para a concretização deste trabalho. A todos muito obrigado.

Palavras Chave

gestão de redes, redes comunitárias, gestão por políticas, rbac

Resumo

Uma das áreas de investigação em Telecomunicações de interesse crescente prende-se com os futuros sistemas de comunicações móveis de 4ª geração e além destes. Nos últimos anos tem sido desenvolvido o conceito de redes comunitárias, no qual os utilizadores se agregam de acordo com interesses comuns. Estes conceitos têm sido explorados de uma forma horizontal em diferentes camadas da comunicação, desde as redes comunitárias de comunicação (Seattle Wireless ou Personal Telco, p.ex.) até às redes de interesses peer-to-peer. No entanto, estas redes são usualmente vistas como redes de overlay, ou simplesmente redes de associação livre. Na prática, a noção de uma rede auto-organizada, completamente orientada ao serviço/comunidade, integralmente suportada em termos de arquitetura, não existe. Assim este trabalho apresenta uma realização original nesta área de criação de redes comunitárias, com uma arquitetura subjacente orientada a serviço, e que suporta integralmente múltiplas redes comunitárias no mesmo dispositivo, com todas as características de segurança, confiança e disponibilização de serviço necessárias neste tipo de cenários (um nó pode pertencer simultaneamente a mais do que uma rede comunitária). Devido à sua importância para os sistemas de redes comunitárias, foi dado particular atenção a aspetos de gestão de recursos e controlo de acessos. Ambos realizados de uma forma descentralizada e considerando mecanismos dotados de grande escalabilidade. Para isso, é apresentada uma linguagem de políticas que suporta a criação de comunidades virtuais. Esta linguagem não é apenas utilizada para o mapeamento da estrutura social dos membros da comunidade, como para, gerir dispositivos, recursos e serviços detidos pelos membros, de uma forma controlada e distribuída.

Keywords

network management, community networks, policy based management, rbac

Abstract

One of the research areas with increasing interest in the field of telecommunications, are the ones related to future telecommunication systems, both 4th generation and beyond. In parallel, during the last years, several concepts have been developed related to clustering of users according to their interests, in the form of community networks. Solutions proposed for these concepts tackle the challenges horizontally, for each layer of the communication stack, ranging from community based communication networks (e.g. Seattle Wireless, or Personal Telco), to interest networks based on peer-to-peer protocols. However, these networks are presented either as free joining, or overlay networks. In practice, the notion of a self-organized, service and community oriented network, with these principles embedded in its design principles, is yet to be developed. This work presents a novel instantiation of a solution in the area of community networks, with an underlying architecture which is fully service oriented, and envisions the support for multiple community networks in the same device. Considerations regarding security, trust and service availability for this type of environments are also taken. Due to the importance of resource management and access control, in the context of community driven communication networks, a special focus was given to the support of scalable and decentralized management and access control methods. For this purpose, it is presented a policy language which supports the creation and management of virtual communities. The language is not only used for mapping the social structure of the community members, but also to, following a distributed approach, manage devices, resources and services owned by each community member.

Contents

Contents	i
List of Figures	iii
List of Tables	v
List of Code Snippets	vii
Glossary	ix
1 Introduction	1
1.1 Motivation	3
1.1.1 What is a Community?	5
1.2 Contribution	8
1.3 Thesis Structure	13
1.4 Summary	14
2 Building blocks	17
2.1 Networking Technologies	17
2.1.1 Wireless Ad-hoc Networks	17
2.1.2 Wireless Mesh Networks	22
2.2 Network and Systems Management	24
2.2.1 Management Methodologies	25
2.2.2 Policy Based Management	34
2.2.3 Role Based Access Control	42
2.3 Summary	48
3 Scenarios & Solutions for Community Networks	51
3.1 Community Aware Routing Protocols	51
3.2 Community Wireless Networks	54
3.3 Community Management Solutions	60
3.3.1 Rural Ambient Networks	61
3.3.2 WIP: Wireless IP Network	62
3.3.3 Hotspot based communities	67
3.4 Usage Scenarios	68
3.4.1 Spontaneous Ad-hoc Networks	69
3.4.2 Isolated Locations	71
3.4.3 Urban Neighborhoods	74
3.4.4 Interest Driven Overlay Networks	78
3.5 Summary	82

4	Management of Stackable Communities	85
4.1	Integrated Community Management	85
4.1.1	Addressing and Identification	89
4.1.2	Name Resolution	94
4.1.3	Partitioning and Merging	100
4.2	Community Management Policy Language	104
4.2.1	Policy Base Concepts	105
4.2.2	Structural Policy Concepts	108
4.2.3	Managed Elements Concepts	120
4.3	Community Management Framework	126
4.3.1	Architecture and Modules	127
4.4	Summary	139
5	Evaluating Policies in Communities	141
5.1	Selecting an evaluation technique	141
5.1.1	Testbed Description	143
5.2	CMF Prototype Implementation	153
5.2.1	Information Manager Implementation	156
5.2.2	Distributed Storage	162
5.3	Evaluation and Results	164
5.3.1	Distributed service authorization	165
5.3.2	Cross layer gateway selection	171
5.4	Summary	178
6	Conclusion	179
6.1	Review and Discussion of Achievements	179
6.2	Future Work	181
6.3	Final Remarks	183
	References	187

List of Figures

1.1	Examples of networks with the same number of nodes. Left is random, right is scale-free.	4
1.2	Structure of a network with communities between some members.	5
2.1	Ad-hoc network providing connectivity in a multi-hop manner.	19
2.2	AODV Route Discovery process.	20
2.3	Multi Point Relays in a network using OLSR.	21
2.4	Three tier Wireless Mesh Network providing connectivity to a neighborhood.	23
2.5	The feedback loop present in reactive systems.	26
2.6	The feedback loop present in predictive systems.	29
2.7	High level structure of an autonomic based management system (MAPE-K loop).	30
2.8	High level structure of a knowledge based management system.	33
2.9	Common policy decomposition from high level policies to low level policies.	35
2.10	Some of the basic components of the architecture for Policy Based Management.	36
2.11	Simplified representation of a single LED using CIM.	40
2.12	Entities are entitled to roles which both grant access to resources and determine obligations.	44
2.13	RBAC applied to a simple hospital with doctors, specialists, interns and nurses.	45
3.1	The multi-overlay community structure proposed in CLONE.	53
3.2	Personal Telco project deployment map as in March 2012.	56
3.3	Wray's community network initial and final deployment [123].	59
3.4	Higher Layer Community and Basic Community according to WIP.	64
3.5	WIP High Level architecture.	65
3.6	WIP detailed component architecture.	65
3.7	A spontaneous ad-hoc network can provide connectivity at the site of an conference.	70
3.8	A community providing services and connectivity at a remote location.	72
3.9	Two community composed by neighborhoods at urban locations.	75
3.10	A community of interest providing services to external entities.	80
4.1	The different expressions through which User interacts.	86
4.2	Different identifiers in a multi layer community oriented communication system.	90
4.3	Naming hierarchy in a community.	95
4.4	Structure of DNS entities in standard DNS and in mDNS.	97
4.5	Community partitioning (left), incorporation of a new member, and merging (right).	100
4.6	Community partitioning and delegation anchors.	102
4.7	Concept composition diagram of the <i>Community</i> concept.	108
4.8	Concept composition diagram in relation to the User concept.	120
4.9	Component architecture of the CMF prototype.	127
4.10	Tree structure create by the <i>Shape</i> module.	132

4.11	Packet flow inside the Linux netfilter architecture when using the <i>Filter</i> , <i>Gateway</i> and <i>Shape</i> modules.	132
4.12	Process for determining the ownership of a packet and applying filter rules.	134
4.13	Join process with a retry due to missing enrollment requirements.	137
4.14	High level overview of the dual storage solution.	138
5.1	System and deployment view of the AMazING testbed.	146
5.2	Internal (left) and external (right) view of a wireless node.	147
5.3	The result analysis interface provided by the AMazING Panel.	149
5.4	Temperature variation over several days with nodes operating.	151
5.5	Radio level with and without nodes operating.	151
5.6	Throughput and SNR observed as the distance between nodes increases.	152
5.7	Graphical User Interface developed.	155
5.8	Structure of a compressed QUERY message.	158
5.9	XOR supported, key based routing in a DHT network.	163
5.10	PAM architecture depicting services, PAM internals and mechanisms (e.g. CMF).	165
5.11	Flow of invocations related to the authorization of an user.	167
5.12	Service authorization scenario with four access points.	167
5.13	Message sequence diagram of the authorization process for a single user.	169
5.14	Client output of the two access attempts (left fails, right succeeds).	170
5.15	Control traffic produced during the successful authorization of two FTP sessions.	171
5.16	Two gateways provide Internet connectivity to a neighborhood wireless mesh network.	172
5.17	HNA message containing SmartGateway information.	175
5.18	Total cost charged to the gateways considering standard and CMF enabled scenarios, under realistic charging assumptions.	177

List of Tables

3.1	Relevant community wireless networks deployments around the world.	58
3.2	Summary of the roles proposed for each scenario.	82
4.1	Comparison of different addressing solutions.	94
4.2	Collision probability of different identifiers for different community sizes.	103
5.1	Radius of the first Fresnel zones for different distances.	153
5.2	Traffic produced and time spent for authorizing a service through PAM.	170

List of Code Snippets

2.1	Ponder authorization policy.	39
2.2	Example policy using CIM-SPL.	42
2.3	Policy groups in CIM-SPL.	42
2.4	<i>Credential</i> syntax proposed by the authors of dRBAC.	47
2.5	Delegations in dRBAC.	47
2.6	Delegations can include restrictions both on resource usage and further delegation.	48
3.1	High level gateway selection rules for RAN.	62
3.2	WIP policy regarding Leader role.	66
4.1	Description of the <i>CMObject</i> concept.	106
4.2	Description of the <i>Attribute</i> concept.	107
4.3	Instantiation of the <i>Attribute</i> concept setting the state of the Forward service to 1 (on).	108
4.4	Description of the <i>Community</i> concept.	109
4.5	Instantiation of the <i>Community</i> concept and its inner concepts.	110
4.6	Description of the <i>Role</i> concept.	111
4.7	Role with containers for Permissions, Rules, Conditions and Attributes. Only members of HigherRoleName can delegate or modified it.	112
4.8	Description of the <i>Permission</i> concept.	112
4.9	Instantiation of the <i>Role</i> concept and basic permissions.	113
4.10	Description of the <i>Condition</i> concept.	113
4.11	Condition evaluating if subject hasRole Member.	114
4.12	Description of the <i>Rule</i> concept.	115
4.13	Rule stating that the administrative cost associated to routes with outside hosts is tied to the cost reported by the Gateway service (monetary cost).	116
4.14	Description of the <i>RuleResult</i> concept.	116
4.15	RuleResult containing the cost of connecting to the Internet for system JohnsLaptop.	117
4.16	Description of the <i>Delegation</i> concept.	118
4.17	Delegation granting enrollment in a role. The Subject cannot further delegate this role if the delegate field is equal to 0.	119
4.18	Separation of rights and role dependency for enrollment.	120
4.19	Description of the <i>User</i> concept.	121
4.20	Instantiation of the <i>User</i> concept, defining an additional <i>Attribute</i>	122
4.21	Description of the <i>System</i> concept.	122
4.22	Description of the <i>Device</i> concept.	123
4.23	Description of the <i>Service</i> concept.	124
4.24	Instantiation of the <i>System</i> concept, with inner <i>Device</i> , <i>Attribute</i> , and <i>Service</i> instantiations.	125
4.25	ANNOUNCE message informing neighbors about the existence of communities C and C1.	135

4.26	JOIN message with attributes.	136
4.27	STATUS message with Attributes required for enrollment.	136
5.1	PING and PONG messages.	157
5.2	STAT Request message regarding an object with ID (MD5("Community.C")).	159
5.3	GET request message regarding an object with ID (MD5("Community.C")).	159
5.4	STORE request message regarding an object with ID (MD5("Community.C")).	160
5.5	LIST request message regarding Delegation tokens of User B in Community C.	161
5.6	QUERY Request message regarding an object with ID (MD5("Community.C")).	164
5.7	Community policy describing a service named FTP which is available to members.	168
5.8	CMF community definition supporting cross layer Gateway Cost definition.	175

Glossary

AC	Autonomic Computing	DNS-SD	DNS Service Discovery
ACL	Access Control List	dRBAC	Distributed Role Based Access Control
ADSL	Asymmetric Digital Subscriber Line	ECA	Event-Condition-Action
AODV	Ad-hoc On-demand Distance Vector	ETT	Expected Transmission Time
AP	Access Point	ETX	Expected Transmission Count
API	Application Programming Interface	FTP	File Transfer Protocol
BABEL	Babel routing protocol	FTTH	Fiber To The Home
BGP	Border Gateway Protocol	FNPA	Free Networks Peering Agreement
BATMAN	Better Approach To Mobile Adhoc Networking	FSM	Finite State Machine
BER	Bit Error Rate	GPRS	General Packet Radio Service
BBS	Bulletin Board System	GPS	Global Positioning System
CA	Certification Authority	GPU	Graphics Processing Unit
CAPEX	Capital Expenditure	GSM	Global System for Mobile communication
CBR	Content Based Routing	HIP	Host Identity Protocol
CC	Community Cost	HNA	Host and Network Association
CCN	Content-Centric Network	HTB	Hierarchical Token Bucket
ccTLD	Country Code TLD	HTTP	Hyper Text Transport Protocol
CDN	Content Distribution Network	IANA	Internet Assigned Numbers Authority
CIM	Common Information Model	ICT	Information and Communications Technology
CIM-SPL	CIM Simplified Policy Language	IdM	Identity Management
CLONE	Community and Location aware cONtEnt based routing	IEEE	Institute of Electrical and Electronics Engineers
CMF	Community Management Framework	IETF	Internet Engineering Task Force
COPS	Common Open Policy Service	IPFIX	Internet Protocol Flow Information eXport
COTS	Commercial Off-The-Shelf	IPv4	Internet Protocol (usually IP refers to version 4)
CPU	Central Processing Unit	IPP	Internet Printing Protocol
DAD	Duplicate Address Detection	IPv6	Internet Protocol version 6
DisCo	Distributed Coalitions Infrastructure	IRC	Internet Relay Chat
DHCP	Dynamic Host Configuration Protocol	IPsec	Internet Protocol Security
DHT	Distributed Hash Table	JSON	JavaScript Object Notation
DMTF	Distributed Management Task Force	JSR-94	Java Specification Request 94
DNS	Domain Name System		

LAN	Local Area Network	RR	Resource Records
LDAP	Lightweight Directory Access Protocol	RREP	Route Reply
LED	Light Emitting Diode	RREQ	Route Request
LTE	3GPP Long Term Evolution	RAN	Rural Ambient Network
MANET	Mobile Ad-hoc Network	SaaS	Software-as-a-Service
ME	Managed Element	SATA	Serial ATA
MD5	Message-Digest algorithm 5	SHA-1	Secure Hash Algorithm 1
mDNS	Multicast DNS	SCTP	Stream Control Transmission Protocol
MOF	Managed Object Format	SIP	Session Initiation Protocol
MPR	Multi Point Relay	SNMP	Simple Network Management Protocol
NAL	Network Abstraction Layer	SNR	Signal to Noise Ratio
NAPT	Network Address and Port Translation	SQL	Structured Query Language
NAS	Network Attached Storage	SOA	Service Oriented Architecture
NETCONF	NETwork CONFIguration protocol	SOHO	Small-Office, Home-Office
NIC	Network Interface Card	SrcRR	SrcRR routing protocol
OLSR	Optimized Link State Routing	TLD	Top Level Domain
OEDL	OMF Experiment Description Language	TPM	Trusted Platform Module
OMF	Orbit Management Framework	TTL	Time To Live
OpenID	OpenID single sign-on service	UML	Unified Modeling Language
OPEX	Operational Expenditure	UMTS	Universal Mobile Telecommunication System
OS	Operating System	URI	Uniform Resource Identifier
OSPF	Open Shortest Path First	URL	Uniform Resource Locator (see URI)
P2P	Peer-to-Peer	USB	Universal Serial Bus
PaaS	Platform-as-a-Service	UUID	Universally Unique Identifier
PAM	linux Pluggable Authentication Modules	VoIP	Voice over IP
PBM	Policy Based Management	VPN	Virtual Private Network
PC	Personal Computer	WBC	WIP Basic Community
PD	Policy Domain	WBEM	Web-Based Enterprise Management
PDP	Policy Decision Point	WHLC	WIP High Level Community
PEP	Policy Enforcement Point	Wi-Fi	Wi-Fi alliance certified
PHY	Physical layer	WiMAX	Worldwide Interoperability for Microwave Access
OpenPGP	Open Pretty Good Privacy	WMN	Wireless Mesh Network
PKI	Public Key Infrastructure	WMR	Wireless Mesh Router
POSIX	Portable Operating System Interface	WoT	Web of Trust
PR	Policy Repository	WSDL	Web Services Description Language
QoS	Quality of Service	WWW	World Wide Web
RAM	Random Access Memory	XML	Extensible Markup Language
RBAC	Role-Based Access Control		
RBM	Role-Based Management		
RFC	IETF Request For Comments		

INTRODUCTION

This chapter introduces the topics that were further explored during the development of this thesis, as well as the underlying ideas that motivated pursuing this work. Also, it presents a brief description of the main contributions to the state-of-the-art that were published in peer-review conferences, journals and magazines. Finally, this chapter presents an overview of the entire document structure.

Technology arguably has kept changing the way individuals interact and envision the future of their professional, private and social spheres. An analysis of how technology shaped humanity should start together with the appearance of humanity itself, but even nowadays, after thousands of years, technology keeps pushing our hopes and dreams, and shaping our lives.

Recently, the invention of the computer created the vision of ubiquitous environments created to improve our quality of life. At that time, this was considered as science fiction, and actually there was a boom in this kind of literature. Automata to perform heavy, time consuming or otherwise unwanted tasks, computers to do complicated calculations and automate our environment by doing simple, otherwise repetitive tasks. Technology was providing a vision. A dream of a much wanted future, and in turn shaping people hopes and focusing individual prospects. What we see today is that fiction authors were not wrong. Even though some areas evolved more rapidly than others, the future followed the vision provided by the latest technology developments and was constantly reinvented and reshaped with every new discovery.

Computers and other electronic devices are not only used to automate our life, or to solve otherwise time consuming equations, they became one of the pillars of our social and professional life. They are one of the most important links we have with others as

shown by the widespread usage of social networking systems, email, messaging and other collaboration applications, which are used everyday by thousands of millions of individuals. Together with the invention and massive usage of computers, so massive that every citizen in a developed country is probably continuously in the presence of computational devices, one of the innovations that changed our lives even more dramatically was the way computers exchange information. Probably even more dramatically than the invention of the telegraph, which also evolved from using discrete physical carriers (mail envelopes) to electronic signals.

The first computers were enormous systems, built in place, weighting tons, and each system consuming tremendous amount of power [1]. Exchanging data between systems made early use of punched cards later followed by magnetic memory units yielding much higher storage capacity. While the amount of data able to be transferred was large (by the standards of that time), the transaction was too slow to support interactive systems. Data transfers involved loading physical the storage device, sending it to the destination and then loading it into the destination system; any form of reply would follow the inverse path. The process was tedious, prone to errors, insecure, and with very high latency. Some decades later, the idea of improving this process appeared following the same evolution made by the appearance of the telephone: replacing discrete storage units (again, mail envelopes) by a real time duplex channel supported by permanently deployed wires. This evolution, which mimicked the mail to telephone evolution (and actually used the telephone infrastructure to communicate) not only changed the way computers communicated but also changed our world in such a way it seems improbable we will ever go back.

This same infrastructure is now the access fringe of a much larger network. Once again, evolution kept reducing communication barriers between systems and, as a consequence, also reduced communication barriers between individuals. Nowadays, and thanks to the ubiquitous nature of wireless access, our data, friends and business partners are reachable everywhere. The telephone is now getting deprecated and being replaced by a communication device much more powerful, sometimes called a personal digital assistant. At our homes we have wireless access points which provide free connectivity within the household, while other wireless technologies provide similar services across entire countries. These technologies are so ubiquitous that individuals can create their own networks, and even act as network providers to others, or foster the creation of communication oriented communities, by using commonly available tools.

Communities are not simple groups to which individuals state their membership. While a naive view of communities considers them as mere digital groups to where individuals belong, a more through analysis will reveal that at its fabric, and the aspect that keeps them alive, is the value of social interactions and the resulting social ties, commonly named as

social capital [2] ¹. They incorporate social ties, interests and aspirations, and can enable communication networks which are created in a more friendly manner, or that are better suited to the usage requirements of its members. The purpose of this work is the creation of mechanisms facilitating the creation and maintenance of such structures, supported by ubiquitously available tools and technologies.

1.1 MOTIVATION

Phenomena were once analyzed by creating a representation of the world where order and perfect structure was supreme. Due to pioneer work of Erdős and Rényi [4], with the discovery of the characteristics between random graphs, graph theory was developed. The concept was simple: get two random nodes and create a link between them. Then repeat the process until a number of links was reached. While the concept seems simple, the insights it brought were very important. In particular, because these random networks experience a threshold effect, just like some real world situations. While some nodes are not connected, the transition between not connected and highly connected nodes is not linear, but resembles a phase transition phenomenon. It is well understood that users do not interact randomly. Particular structures are present in the relations between individuals, which has motivated several studies regarding graph theory. Real world networks are not fully random, and therefore random graphs are not really appropriate to study all real life processes, but after the initial graph theory studies, two further achievements greatly improved the potential of graph theory for real world networks: the small world phenomena and scale free graphs.

The small world phenomena relates directly to the study of social networks, i.e., the study of who knows, works with, or otherwise interacts with whom. Pioneer studies [5] demonstrated that for any two random persons in the United States, the distance, in terms of number of hops, between these two persons was, in average 6. That is, in average, any two US citizens can potentially get into contact very rapidly because they have a relatively small social distance. Relevant to the application of communities to communication networks, two basic properties from the study of the small world phenomena are pertinent:

- Social networks experience a high clustering effect i.e., in average, the fraction of nodes that are connected to a given node and also have connections between them is high. This reflects our social behavior as we cluster around places, interests, work places, schools, or simply family.

¹This concept is present in other sciences and is attributed to Aristotle in *Metaphysica* as “The whole is more than the sum of its parts”. However it cannot be found in any of his works and actually contradicts Aristotle [3].

- The average distance between any two nodes is small, and according to several studies, around 6 [5].

These findings seem to fit our social behavior, but in practice the difference between a random graph and a graph experiencing a small world phenomena is very small. Later works by Watts *et al.* [6] suggest that simply by rewiring some of the links between nodes, a small world would arise. Therefore, and because we do not exhibit a random behavior, it is rather simple to observe a small world phenomena between any set of individuals.

Scale free graphs complement the properties of the small world phenomena by the observation that social networks are not homogeneous in terms of connectivity (number and nature of links between users) [7]. While the vast majority of the individuals are clustered, they are also poorly connected. The result is that the fraction of connections to the members of the clusters will be much higher than with the exterior. The remaining minority of nodes is highly connected and therefore act as the clustering point for their neighbors. The scale free property also states that this heterogeneous nature of graphs is present independently of the dimension of the network, therefore the term scale free. In practice, it can be observed that a small world always presents scale free properties. Figure 1.1 depicts a network presenting both the small world and scale free properties. As it can be observed, key nodes with high connectivity provide a clustering capability for the remaining, poorly connected nodes.

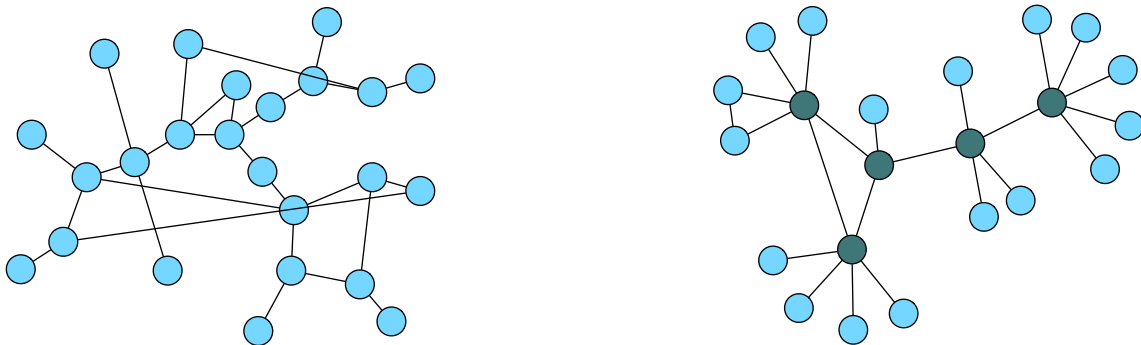


FIGURE 1.1 – Examples of networks with the same number of nodes. Left is random, right is scale-free.

A rather simple observation is that it is highly possible for any two individuals to cluster around a similar shared environment, location or interest. Or in other terms, it is highly probably that some two individuals share some common attribute. Potentially sprouting a community uniting individuals sharing the clustering property. Moreover, communication between individuals is mostly done between the same individuals, the members of the cluster. The probability of two random individuals, which already interact, to interact again in the future is also very high. All this improves the probability that a group of individuals

interacting at some point will interact again in the future, and motivates the creation of long time trust relations. The resulting graph is similar to a scale-free graph, but with the existence of clusters of individuals, with some of the individuals acting as anchors for the community, and/or bridges between different clusters. In this context, each cluster can result in the formation of a community.

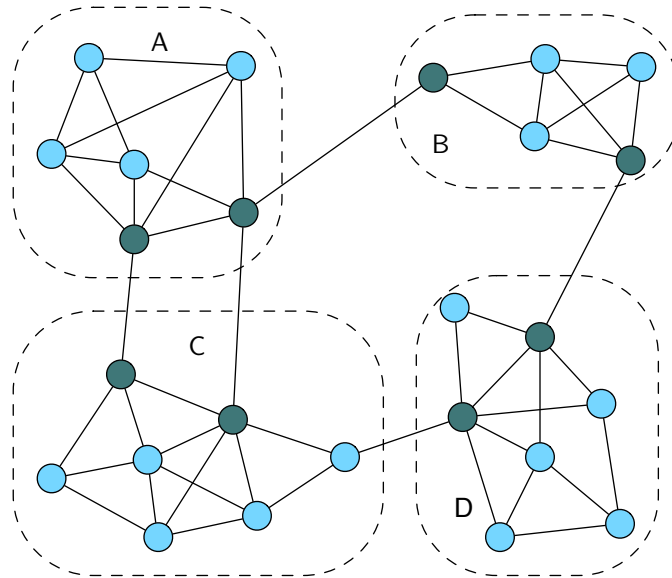


FIGURE 1.2 – Structure of a network with communities between some members.

An important aspect of both small world, scale free networks, and community networks made by individuals is that communication networks can take this structure in consideration and improve its utility, instead of blindly treating all individuals the same way.

1.1.1 WHAT IS A COMMUNITY?

A community is defined as a group of close knit entities (usually individuals) sharing a common environment. Because not all authors agree upon the meaning of community, its specific definition has been changing over time. Different disciplines have different definitions for what is a community, what mechanisms are required and what drives its creation, maturity and destruction. Around the middle 1950s the amount of definitions was already large [8], and with the appearance of the Internet, this number has increased.

The key aspect that gathers individuals in a community is the shared environment, which presents members of the community with specific characteristics, opportunities and challenges. A shared environment makes it easier to create social ties because of the shared

context. Before the advent of the Internet, in the early 1990s, this environment actually meant physical location. Members of a given community would live, work, or otherwise interact inside some physical boundaries, which defined the community environment. One could find, and still does find, communities that developed around small towns, neighboring farms, or people working together like in a university campus.

In many situations, the environment (in this case the physical location) provided a common context for all people living or working in that area. As a consequence, the shared context facilitated interaction by presenting similar challenges, limitations and behavioral constraints to all members. The result from frequent interaction in such a shared context is that individuals create ties, and its personal objectives become more aligned (this doesn't impose perfect harmony), creating a common set of values, and an expected behavior, while strengthening the social ties between members, leading to community cohesion.

Communities were tied to physical location because of the limited propagation speed of information. When a community started to expand, without the modern communication facilities, the common and ubiquitous environment was not possible to be properly maintained, simply because changes in the environment would take a long time to reach all members. Local variations in the environment could be mutually amplified, and without the capability of being dissolved to the remaining environment, the community would eventually suffer fragmentation.

With the advent of the Internet, and its mass adoption by all social strata, it was possible to see that communities are about people interacting in an environment and the environment can be almost anything providing clustering characteristics [9]. Because of our current communication capabilities and range are being constantly enhanced, communities sprout around environments like a music genre or artist (e.g., Folk music or Classic Rock bands), a product (e.g., a particular mobile phone), an interest (e.g., better Internet connectivity), or simply a concept (e.g., Eco-sustainability).

Communities are not created simply because individuals share a common context. Individuals must also interact with each other and create bonds, even if indirectly through *personas* or *avatars* [10]. The main result is the social capital [11], which constitutes one of the main aspect differentiating communities from groups. Under this view, groups are simply the set of all individuals with certain characteristics (e.g., location), while a community is richer because of the existence of the social bonds, and shared interests between individuals. This concept is so important that it reshaped the Internet after the introduction of social networks [12] [13]. Also, community marketing gained strength, and a big number of companies now focus in creating communities around their products. The result is a closely knit consumer base providing self-assistance, motivating others to join (i.e., buy the product, or

help solving problems), discussing the future capabilities of the product, and even fostering business around it.

The reasons why communities are frequently desired rely in the characteristics that authors from social sciences have identified in communities [14]: membership, influence, integration, fulfillment of needs, tolerance [15], reciprocity [2] and trust [16].

Membership relates to the fact that individuals find it attractive to belong to a group, and once in the group, they will behave differently towards members and outsiders. Individuals feel integrated because other individuals in the community have similar prospects and knowledge is openly shared. This has been identified in the social sciences as the feeling of belonging. In turn, they are able to influence the remaining members, gearing the community towards better fulfilling their own needs and feel rewarded by it. In the end, because of the past interactions inside the communities, individuals become more tolerant regarding members and exchange effort (i.e., knowledge or services) not according to strict *tit-for-tat* rules, but in a more altruist manner (for the sake of the community).

The frequent past interactions and close proximity the members feel, also lead to an increased sense of trust towards members. It should be pointed that an increased sense of trust is not the same as the abolition of barriers and policies. In fact, many authors support that, in order to foster the expected behavior, incentives, trust and reputation mechanisms, as well as policies shaping cooperation, should be present [17]. Without this it becomes difficult to identify divergent behavior, and increase efficiency. Then the community stops fulfilling members' needs and members' identification with the community principles decreases, leading to fragmentation or simply lower number of members.

While the characteristics of a community belong to the field of social sciences, these characteristics can provide insight on how to develop and maintain a communication network able to create communities. The inherent characteristics identified in communities make it possible to limit the impact of issues like cooperation selfishness [18], and increase the efficiency of communication in networks scenarios requiring (or benefiting from) strong user commitment.

When applied to the creation and management of networks, communal characteristics can promote both the cooperation between users and better suitability of the network to individual prospects. On the opposite side we can find traditional broadband access networks, where user preferences are just an application or contractual aspect, and individual prospects have little or no impact to network operation and management. When applied to communication networks, communities represent interaction contexts.

The community itself can be described as a set of close-knit entities which have autonomic

behavior, shaped according to a set of principles they feel identified with (the community doctrine). Network elements and services do not feel these principles *per se*, but the individuals that actually activate and configure these elements do. Behavior can be mapped to a common set of rules, permissions and configurations, and is valid for users, services, network elements, and the network layer itself. It should be noticed that this corresponds to a fully specified communication environment with expected parameters and applied to the entire community, in a cross layer manner. While there is some freedom for users to configure their devices, their freedom is always confined to the limits imposed by the community.

One point differentiating this view of communities from a restricted group is that in communities members have the opportunity to propose new rules. Moreover, because interactions are important, a knowledge layer can be available to members, allowing to store all the information internally produced by members, the knowledge of their interactions.

This thesis aims to tackle solutions for scenarios where users wish to create communities of interest for the purpose of collaborating, sharing information, or simply providing shared Internet connectivity. Therefore it will focus on devising the control and policy mechanisms and the semantics allowing users to manage their communication systems in a decentralized manner.

1.2 CONTRIBUTION

The pursuit of this thesis resulted in several contributions to the advance of the state-of-the-art. The thesis document describes an integrated view over the work produced, while several publications support the analysis and results presented here.

The first publication focus in the identification of the mechanisms required for the management of community networks. Titled “Role Based Cross-Layer Communities in WMN” [19], this publication analyses some of the management and access control concepts that are relevant to the envisioned scenarios, and proposes possible solutions for some of the identified problems. It is considered that devices and users are encouraged to cooperate at all levels, sharing their resources both at the application and communication layers. The social relationships existing between users will promote increased trust in local environments, allowing wireless techniques based on individually owned Access Points (APs) to become a trusted communication environment. Mesh networking is particularly adequate to this approach, allowing users to roam freely, while their APs establish long-term communication backbones. These multi-level communities can define their own policies and establish cooperation agreements with other communities — both at the same and at different levels of the

communication stack.

In this work it was proposed that rules are vital to communities as they define the behavior nodes should follow, resources they can use, and the configuration parameters they should apply. The agreement on the same set of rules will create a coherent environment, even if members have different resources or provide different services. Taking in consideration the concepts of Distributed Role Based Access Control (dRBAC) [20], the basic concepts enabling cross layer, community based management were proposed. Users are entitled to roles and may delegate these roles to others as they seem fit. Therefore the network grows organically according to users' social relations, only limited by the requirements for enrolling into a particular role. Users keep delegation information in *Wallets*, which can be queried by other users, or even by other communities. This work also identified the challenges created by the need for verification of delegation chains, both between users, and between communities.

Community management considers that communities are formed across different layers with the possibility of having different objectives, e.g., wireless resource sharing, routing support, exchange of specific distributed application-layer services. In this sense, it was proposed a cross-layer approach for the communities' creation and management: the communication between several nodes requires cross-layer interactions and agreements between different types of communities, where this interaction is addressed in a similar way across layers. Such concept requires a distributed architecture capable of controlling, managing and discovering communities in the considered scenarios. This architecture follows a modular approach aiming at efficiently providing the required functionality, keeping, simultaneously, a low complexity level to cope with nodes' resources and medium limitation issues. In the publication titled "An Architecture for Community Mesh Networking" [21], a draft architecture for management of community networks is first presented, together with the basic concepts driving the management of community works. The architecture developed also considered the integration with a mesh based, and community aware, service location solution named WiNe-SD [22]. Service location and advertising components are vital to efficiently propagate service and community information on the available communities. The mechanism proposed is evaluated against standard service discovery mechanisms [23], showing its benefits, both in terms of control overhead and query delay in the network.

Communities are distinct from a set of individuals. Social bonds are created over the multiple interactions individuals have in their restricted environment. Communities provide the agglutination factor for users to cluster based on their interests, and for users to cooperate in the fulfillment of common goals. Even possibly, some users find new interests due to community dynamics. In wireless communities, users participate with their devices and

exchange services at different layers. Networked devices may route each others traffic in an ad-hoc manner. Files may be exchanged through well known services such as File Transfer Protocol (FTP) [24], and web based collaboration tools can be deployed in order to foster cooperation. Individuals may also exchange services at higher layers, such as help desk tasks or counseling. Traditional incentives are not appropriate for community environments, as they usually focus in one specific layer (e.g., packet exchange). Moreover, due to the multiple interactions individuals may have in communities, reputation and trust cannot be ignored and must be considered when aiming for fairness in resource consumption, or service provisioning. In a publication titled “Community Building over Neighborhood Wireless Mesh Networks” [25], a discussion about the correct incentives and pricing schemes is presented. Also in this work, it is presented a discussion regarding how community networks can bootstrap and be maintained, while keeping community operation fair. The concept of community networking considers integration of an extended collaboration stack, ranging from user interactions (not mediated by devices), to network based services, and to packet exchanges at the level of the Institute of Electrical and Electronics Engineers (IEEE) 802.11 Physical layer (PHY). For each layer between the social and PHY layers, it is described how incentives can be applied, and how cooperation can benefit from such integrated communication stack.

For scenarios of high-density community mesh networks, deployed in urban scenarios, and where part of the equipment belongs to end-users (user-managed mesh networks), it seems to be natural to provide a certain level of control to the equipment owners (e.g., the common bandwidth controls of Peer-to-Peer (P2P) applications). This level of control is desirable due to differences in user behavior, and although common goals can exist in a community, users still keep their individual expectations, and ownership over their equipment. However it is imperative to promote cooperation and agreement between users, otherwise, no communication will exist — which would negate the whole concept of user-managed mesh networks. Thus user based network management, while desirable, comes at a price, and a balance is required to be established between user preferences and the overall network setup. In a publication titled “Managing Community Aware Wireless Mesh Networks”, the challenges and directions found when aiming to develop a system capable of handling this dichotomy are presented. With the solution proposed, the aim is to improve the self-manageability of the network, making it at the same time more robust and user friendly. That is, management concepts developed aimed to be easily comprehended by individuals, and able to foster new methodologies for network management. The management model proposed is based on meta-information from the users social level and other social structures which is then mapped to the self-organization logic of the network, thus creating a natural mapping between users self-interest and the underlying community mesh network. One of

the benefits of such approach is that routing can take in consideration higher layer metrics, instead of the typical, packet oriented ones. In this sense, routes can be pre-established between individuals which present to have ties at the social level, hoping that by this, routes can be provided faster for more frequent communications.

Wireless communities can be deployed over a wide range of scenarios, and for a different set of purposes. When aiming for the use of dynamically evolving policies to govern the expected behavior, it is vital that the policy can be adapted to users' needs. Instead of defining all possible attributes, that role and rule policies can have, it is argued that an ontology describing the policy concepts should be defined. Then, each community, acting as an independent policy domain, can create custom elements, which still adhere to the overall policy ontology. In a publication titled "Ontology driven Framework for Community Networking Management" [26], the reasoning behind this approach is presented, as well as how the ontology can be structured, and what base concepts are required. This work is based on the concept of flexible policies, which adhere to an ontology. Individual managing software instances will process policy objects and are able to operate in environments containing policies with unknown elements. The actual instantiation of the software framework realizing this concept was described in a publication titled "Framework for User-centric Autonomic Management" [27].

When aiming for user driven community wireless networks that sprout in urban scenarios, it is important to consider topology aspects. It is considered that individuals composing a community possess two types of devices: networking routers and terminals. Terminals roam across the community together with individuals (e.g., laptops, tablets, and smartphones). Networking routers exist at citizen premises and are not mobile at all. Connectivity between these devices will be constrained by their location, and characteristics of the surrounding environment (e.g., radio fading and interference). Placing devices in places closer to windows, and with line of sight to other devices may be possible, but frequently, individuals can only increase their antenna characteristics. Even this it is not expected to happen for most of the users as it incurs in additional costs and require a higher level of expertise regarding radio transmission. In a work titled "Characterization of Unplanned Metropolitan Wireless Networks" [28], it was described an analysis of the actual deployment of the potential wireless infrastructure for a community in the Aveiro urban area. The analysis focused in the characterizing existing IEEE 802.11 access points in the main areas of Aveiro. For each access point, and taking in consideration the average signal level received, its location and capabilities were recorded. The resulting information allowed the creation of a topology map of all equipments detected, and the characterization of a wireless community network utilizing those devices. Simulations considered different transmit ranges, and different maximum Time To Live (TTL) values for route queries. As shown, the high density of access

points allows the creation of wireless community networks, and excluding access to the world available Internet a small number of gateways is required in order to connect the identified isolated clusters. The work regarding this approach further present in Filipe Ferreira Masters dissertation [29].

Social communities are not restricted to wireless (mesh) networks, and can actually be found in several other environments. Companies frequently try to drive communities around their products, and users cluster in such structures for the purpose of achieving a goal, even if not relying on wireless technologies. Given the pervasiveness of computation clouds, mainly in the form of public and private clouds, it is expected a rise of community clouds. Private clouds are owned by corporations for their internal use. Public clouds are enabled by corporations and provide Infrastructure, Platform, Network or Software as a Service (IaaS, PaaS, NaaS, and SaaS respectively) to other corporations or individuals. Communities clouds are similar to public clouds, but with one important difference: users act not as individuals and consumers but as communities, where the policy is used to create cohesion between members, so that public services can be provided to other individuals or corporations. The requirements in terms of Identity Management (IdM) are identified, as well as the requirements for a cloud operating system realizing the concept of user centric cloud communities. This vision was presented in a publication titled "User Centric Community Clouds"[30].

The need for a testbed platform with the characteristic of allowing repeatable network experiments was identified and, during this thesis, such testbed was deployed. It served as the evaluation platform for the work presented in this document, as well as for several other masters and doctorate thesis. In a publication titled "AMazING - Advanced Mobile wireless Network playGround" [31], are described the efforts in this direction. This wireless testbed is composed of 24 wireless nodes that can be used to perform a broad range of studies in the area of next generation networks. This publication addresses the difficulties and constrains faced throughout the deployment process. Flexibility and controllability were key concerns driving the testbed design. The testbed can be remotely managed through a series of remotely accessible web services performing low level management. Validation results are presented, showing the interference levels of the testbed as well as its maximum throughput capabilities. A later work under the same scope [32], presented the advanced management interface developed for the purpose of automating testbed operation, while facilitating interaction between researchers and any Orbit Management Framework (OMF) [33] enabled testbed.

Like any other physical testbed, there is always a limitation on the number of nodes and network interfaces available, in particular wireless interfaces. Technologies allowing the instantiation of virtual nodes enable extending the existing testbed resources horizontally.

However, these virtual nodes have also virtual hardware, which is somewhat standardized and lacks support for many technologies. Having this in consideration, it was identified the need for a software component allowing to export physical network interfaces into virtual nodes. This method is of particular importance when applied to network interfaces using particular communication technologies, such as the wireless interfaces available at the AMaZING testbed. The architecture, implementation and evaluation of this concept was published in a publication titled “Network Interfaces Flying over IP Networks” [34].

Mobile robotic systems are a typical example of a distributed environment where agents must collaborate in order to fulfill predetermined goals. Communications for mobile robots can use a wide range of technologies, being IEEE 802.11 [35] and IEEE 802.15.4 [36] two of the most widely used. Both technologies allow multi-hop communications as a way of increasing communication range. In a publication titled “Collaborative Relaying Strategies in Autonomic Management of Mobile Robotics” [37] it was analyzed how collaborative relaying techniques can be used to increase communication characteristics in scenarios considering mobile robots. Collaborative relaying differs from the generalized multi-hop paradigm from ad-hoc networks in the sense that it can still be in infrastructure networks. Moreover, it only considers a single relay between source and destination. The advantage is that while in multi-hop routing, throughput decreases with the increase in the number of hops, in cooperative relaying, throughput can actually increase. Other metrics such as latency, packet loss and energy consumption can also be improved by using this technique. A key aspect of this technique is the algorithm selecting the best relay. This selection must take in consideration aspects such as battery level, available processing capacity, movement speed, and the characteristics of the traffic to be sent (bitrate, packet size, criticality). In this work, it is proposed the existence of a distributed policy and a distributed information system, containing the active policy, as well as the currently active applications, and communication relays used by each robot. With this information, nodes can have information about themselves and their neighbors, and can fulfill their goals following a cooperative approach. Specifically, robots location can be adjusted so that the new location potentiates its neighbors to better fulfill their goals.

1.3 THESIS STRUCTURE

This thesis is structured in 6 chapters: one provides an introduction to the work; two chapters describe the state-of-the-art; two chapters describe and evaluate the work proposed; and one final chapter provides conclusions regarding the work and its future developments.

The first chapter provides an introduction to community related concepts. This briefly

includes some information regarding scale free networks and community structure. Also, the motivation driving this work and the contribution provided to the academic community is described.

The second chapter describes building blocks driving the creation of wireless communities. Wireless communication technologies and web based social networks are described. Management solutions and management principles are also described as these concepts have a strong impact in the work developed.

The third chapter describes solutions which take in consideration community or otherwise social aspects into communication networks. Solutions providing management of systems following community concepts are more deeply described. Also, several scenarios benefiting from the application of community models are described, and the most relevant roles of its participants are identified.

The fourth chapter comprises the main contribution of this thesis. It describes how communities can be created when supported by commonly available communication devices and technologies. A language to describe the policies applied to the management of each community is presented.

The fifth chapter presents a technical description of the experimentation platform developed, as well as some results obtained when using the testbed. In this chapter it are also described some technical details regarding the implementation of the prototype policy management system developed. Evaluation of this prototype focuses in demonstrating the feasibility of the concepts described and the effectiveness of the language developed.

The sixth chapter, the last, draws some conclusions regarding the work developed in this thesis, and presents some insights towards the research challenges which were left open by this work.

1.4 SUMMARY

This chapter provided the motivation for this thesis, and introduced aspects related to the study of graphs, and to the structure of the particular graphs demonstrated by individuals. In particular, the small world phenomena and the scale free properties were presented. They are relevant because they are at the root of the idea that it is rather easy for communities to sprout whenever social relations are considered. The probability of a random set of users to share friends, or to share interests, is not as small as one would think if considering that humans construct random graphs. Human individuals are social beings, which look for structure

and automatically create clusters around their interests and known ones. This chapter also presented a simple introduction to the aspects which define a community, and in particular, why a community is not a simple group of users. Groups of random entities, with no relation or interest (such as computer systems) may present highly random communication patterns. Human individuals create bubbles around them, and interact more often with the individuals inside these bubbles. Also in this chapter are briefly described the most relevant contributions that were published during the development of the work presented in this thesis. These contributions comprise a total of 8 publications in conferences, 2 publications in international journals, and 1 publication in an international magazine. At the date of writing, these publications here already cited more than 15 times by peers, further highlighting the impact of the work in key areas of the research community. Other publications were produced during the development of this work, some considering cooperative behavior between entities, but in those cases the work developed in this thesis was not the main contribution of the publication.

BUILDING BLOCKS

There are several building blocks required for the design and implementation of management solutions for community wireless networks. At the protocol layer bottom, wireless communication solutions provide the fabric for the exchange of information packets between members. These solutions may present to users both cellular based and mesh based topologies, and allow the creation of highly dynamic networks with commonly available hardware. Above this, routing protocols put some order into the shared communication fabric, allowing information to flow in the most optimal way between devices, and make point-to-point, and point-to-multipoint communications effective. A particular aspect of routing protocols for these scenarios, which contrasts with solutions for the general Internet, is the support of near real time adaption to network changes. Operating in a cross-layer manner, management solutions orchestrate all components and make communications as effective and efficient as possible. Distributed communication environments do not exist by themselves and operate efficiently without some form of management support. Therefore, this chapter will describe some of the most relevant management concepts used in distributed environments. Solutions making it possible to employ the approaches described will also be discussed, having a special attention devoted to policy based management, as it represents an important conceptual aspect of current management solutions. Overall, given the wide scope of relevant subjects and approaches, this chapter will focus on technologies and concepts which influenced the work developed in this doctoral thesis.

2.1 NETWORKING TECHNOLOGIES

2.1.1 WIRELESS AD-HOC NETWORKS

Networks composed in ad-hoc manner sprout very rapidly after the technologies around the IEEE 802.11b [38] suite of protocols was standardized. One of these particular types

of networks is the Mobile Ad-hoc Network (MANET) [39]. While the technologies around MANETs saw a great increase in the 90's, together with IEEE 802.11b, the underlying concepts are from the packet radio solutions developed in the 70's. It consists of an unstructured, multi-hop wireless network, composed by mobile terminals, which act at the same time as clients, routers, and may even provide services. These networks are typically used in situations where no infrastructure is present due to monetary reasons, or the time to deploy such infrastructure is not compatible with the purpose of the network (i.e., disaster, military situations), and/or only temporary or no Internet connectivity is required. Ad-hoc networks follow the simple principle that as long as two nodes are within range, or a multi-hop path is possible, they will be able to exchange data.

No predefined structure or fixed size is found in MANETs, because one or many underlying characteristics are not static: the number of users, the communication range of their devices, their location and their movement direction can either be static or dynamic. In fact, considering a wireless communication medium, and mobile users, determining the current structure of a MANET may be a complex task. Users may exchange data between each other, disconnect and connect their devices, and move freely in any manner, as they require. An important scenario employing ad-hoc technologies is a disaster relief situation, where a network must be created instantly to connect all rescue teams (e.g., Physicians, Firefighters, Police Officers), thus increasing coordination and improving rescuing success ratio [40]. In such situations, deploying a planned infrastructure may not be feasible. Relying on higher range cellular technologies may also be impossible in remote locations. The use of anchor nodes, providing a cellular like network connectivity, limits the reach of the rescue teams, and actually imposes the deployment of these nodes right at the start of the operation.

Ad-hoc networks may or may not be disconnected from the Internet. In fact ad-hoc networks can also be connected to an infrastructure based network and be used only as a dynamically adapting radio extension of the existing infrastructure. These are frequently called as hybrid ad-hoc networks, or ad-hoc stub networks. These integrated networks, when used in a disaster relief scenario, would allow physicians to retrieve patient history, or firefighters to retrieve blueprints of the related buildings from external servers, and use this on-line information in the field, while having the flexibility of multi-hop relaying. Other relevant scenarios for ad-hoc networks are conferences and other public gatherings, ad-hoc wireless transport networks, or military driven communication networks.

A major characteristic of any ad-hoc network (mobile or static) is that communication between two nodes can be successfully accomplished even if the two endpoints of the communication are not within each others' radio range. That is, as depicted in Figure 2.1, given two nodes S and D which are not within direct radio range, R can help by acting as a

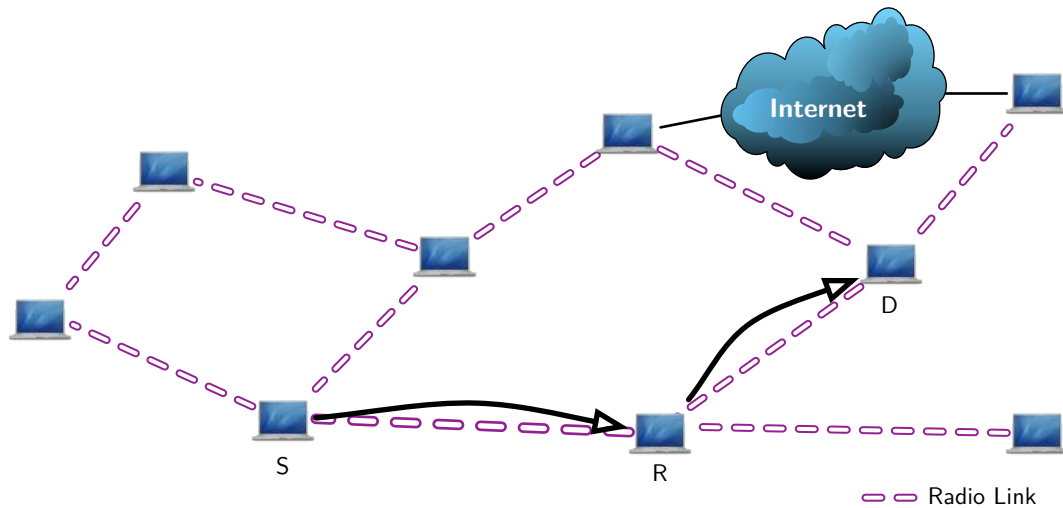


FIGURE 2.1 – Ad-hoc network providing connectivity in a multi-hop manner.

router, forwarding data packets. This multi-hop routing approach enables the network to expand, as more nodes are added. The benefit is that radio coverage of the ad-hoc network adapts dynamically to the actual location of nodes, and radio capabilities, enabling a flexible delivery network to operate without any previous planning, and easily accommodating new users.

In order to provide multi-hop routing capabilities, and even interconnection with wired networks, tens of specially crafted routing protocols were proposed by the research community. Two of the most widely used are Ad-hoc On-demand Distance Vector (AODV) [41] and Optimized Link State Routing (OLSR) [42]. AODV is based on the concept of reactive routing. The process of creating routes between nodes is only triggered when a node *S* wishes to communicate with a node *D*. If no traffic is sent, the AODV protocol spends no resources in determining routes. Before any user data is sent, the route discovery process will make use of a Route Request (RREQ) message to create a route between *S* and *D*. The message is created by *S* and is sent to all its neighbors. When receiving a RREQ, nodes will either reply with a Route Reply (RREP) containing the route, or will increase the RREQ hop count field and rebroadcast it to the next set of neighbors. Also, nodes will keep information about the previous hop from where the RREQ was received. This aspect is vital for nodes to be able to reconstruct the path. Eventually, after a number of broadcasts, the message, or several due to the broadcast method used, reach node *D*. Hopefully, the message going through the shortest path will reach *D* quicker than other messages. *D* will issue a RREP towards node *S*. Because all nodes kept pointers to the previous neighbor, they will be able to forward the message to

S. As long as data packets are exchanged between *S* and *D*, route entries will be kept. After a timeout, they will expire and the process needs to be repeated. Figure 2.2 depicts a simple process of requesting a route in AODV.

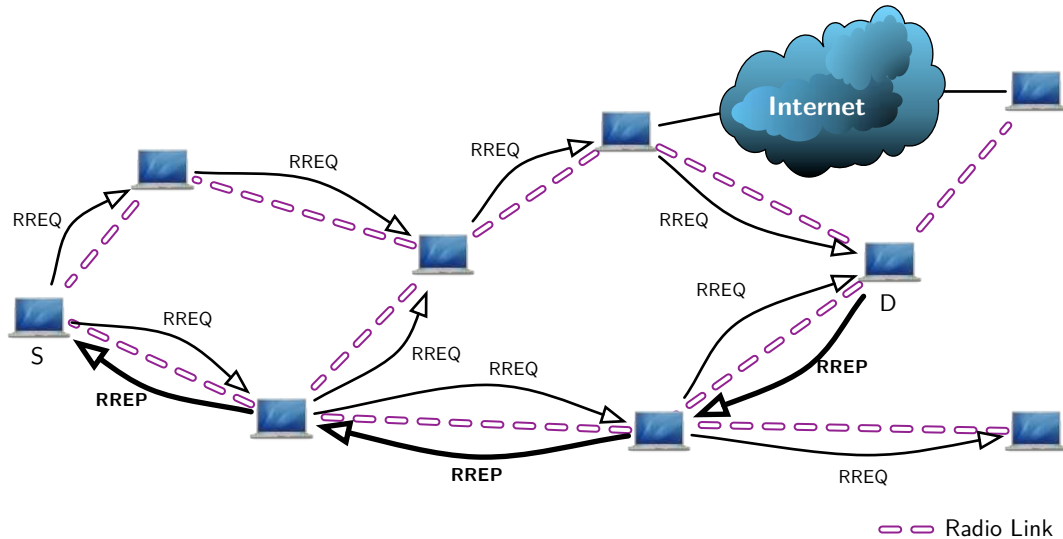


FIGURE 2.2 – AODV Route Discovery process.

The other widely used protocol is OLSR. It greatly differs from AODV because instead of being reactive, OLSR is a proactive protocol. The algorithm employed by OLSR is more complex and gathers information regarding the links and neighbors of each node. To accomplish this, it makes use of HELLO messages which are continuously exchanged between neighbors in the network, even if no user traffic is sent, or no routes are required. HELLO messages are used to determine the status between two nodes, and to exchange information about 2 hop neighbors. Because nodes will advertise all known neighbors to others, it is possible to locally determine which node has a higher number of neighbors. These nodes are candidates for a distribution tree named Multi Point Relay (MPR) (see Figure 2.3). The algorithm converges so that all nodes are connected to a single MPR node. The purpose of creating such structure is to create a distribution tree to route packets from nodes. Only MPR nodes will have the task of relaying others packets, which enables the creation of algorithms that create MPR structures taking in consideration actual node processing capabilities, power source, of simply a high level willingness to relay.

By comparing AODV and OLSR in static scenarios, it is evident that if no traffic exists, AODV will provide lower overhead. The drawback is that while OLSR provides routes between any two destinations instantaneously, AODV will be slower due to the broadcast based discovery process. Nevertheless, both solutions provide multi-hop routing in networks

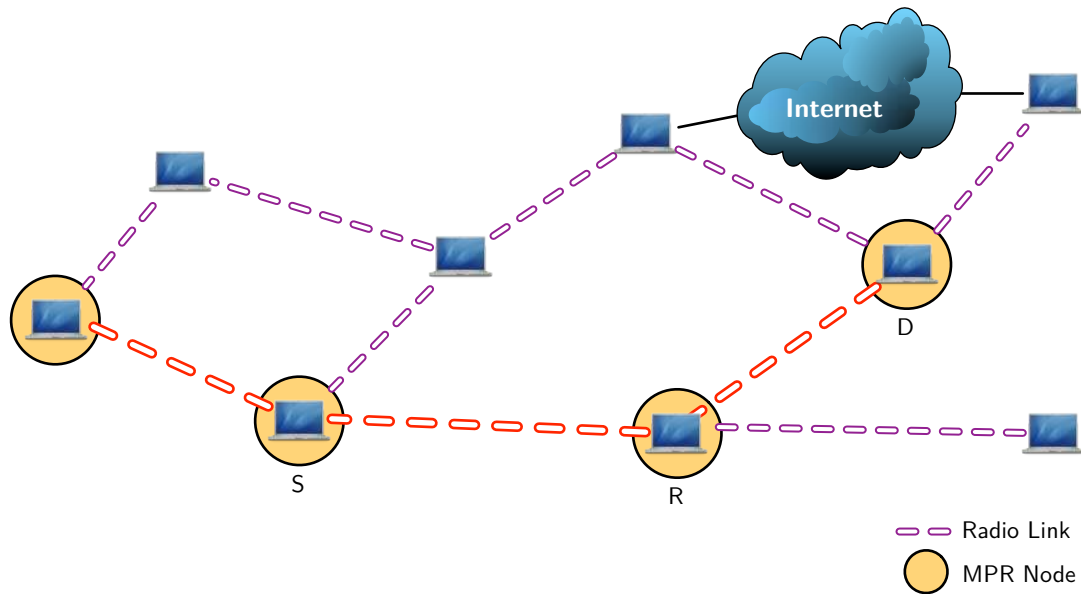


FIGURE 2.3 – Multi Point Relays in a network using OLSR.

with flat addressing schemes, and are able to quickly adapt to changes in network topology or even link quality.

The advantages of a multi-hop approach are clear, however some issues still exist: security and privacy must be maintained so that data is not modified, filtered or even eavesdropped while being routed. In particular no unauthorized user should join the network if sensitive data is being sent [43]; users must be actively motivated to relay others packets, and barriers to prevent selfish behavior must be deployed, even if that implies lower available bandwidth and decreased battery lifetime for some devices [44]; multi-hop routing, as well as the remaining required protocols, must be effective while generating little overhead [45]; routing should be robust so that connectivity to all nodes is maintained without sacrificing a minority of nodes (e.g., by depleting their battery) [46].

Another important issue is that while these networks are ad-hoc by definition, with the large quantity of routing, security, incentive, Quality of Service (QoS), and charging mechanisms proposed and deemed as required, an ad-hoc network providing some basic assurances (e.g., in terms of delivery ratio, privacy, or QoS) in reality is not ad-hoc and requires some sort of planning and configuration. The lack of coordination between solutions, low security, abstract models, management complexity, and eventually unstable or unpredictable performance have lead to the weak deployment rate we find for ad-hoc networks. Also, taking in consideration the availability of cellular and wired based solutions in developed

countries, constructing a reasonable usage model and business model is still a challenge. For undeveloped countries, providing basic connectivity through wireless ad-hoc technologies may have larger reach, and according to some authors present promising prospects [47].

2.1.2 WIRELESS MESH NETWORKS

Wireless Mesh Network (WMN) are very similar to MANETs and were the focus of the research community for scenarios aiming for more stable connectivity requirements. In these networks, nodes can relay messages in a multi-hop manner, and communication can be based in easily available wireless technologies, such as the IEEE 802.11 family of protocols. One clear difference is that while ad-hoc networks are unplanned, mesh networks consider that some planning should be present in order to provide better quality of service. Authors such as Chandra *et al.* [48], clearly demonstrated that a minimum number of gateways to the Internet and some planning is required in order for the network to provide a reasonable level of service, something that MANETs frequently struggle to provide. Therefore, in a WMN not all nodes are expected to be mobile, and it is considered the existence of one or more static backhaul transport cores, Wireless Mesh Routers (WMRs), and an access fringe comprised by end-user terminals such as laptops. The result is that while ad-hoc networks are traditionally flat networks (although, some solutions like [49–51] propose clustering which reduces network flatness in ad-hoc networks), WMNs typically contain several tiers, and at least two can be found. Each tier is optimized towards a specific function, as this is shown to improve total network capacity [52]. Multiple technologies can also be used for the purpose of splitting the WMN in isolated tiers. Figure 2.4 depicts a WMN with three tiers, using multiple technologies, for the purpose of providing network connectivity to a set of buildings.

In the core of the WMN we find WMRs, equipped with higher range, and higher capacity interfaces, frequently relying on technologies such as IEEE 802.11s [53], and even IEEE 802.16j [54]. IEEE 802.16j provides very high capacity links, very high range, and a good set of management solutions, making it ideal for the purpose of aggregating and distributing traffic. In the scenario depicted, a local tier using cheaper, IEEE 802.11n [35] based equipment may be used to route local traffic, while a IEEE 802.16j core provides the communication backbone, and connectivity to the Internet. At the access tier the IEEE 802.11abgn [35, 38, 55, 56] set of protocols still has a strong ground, mainly due to the availability and price of devices based on these technologies. The access tier presents point of attachment to end-user terminals and delivers data either locally or to the backbone interconnect.

An important consideration for WMNs is that the access fringe is presented to terminals as points of attachment (e.g., an Wi-Fi Access Point). End devices send packets towards their

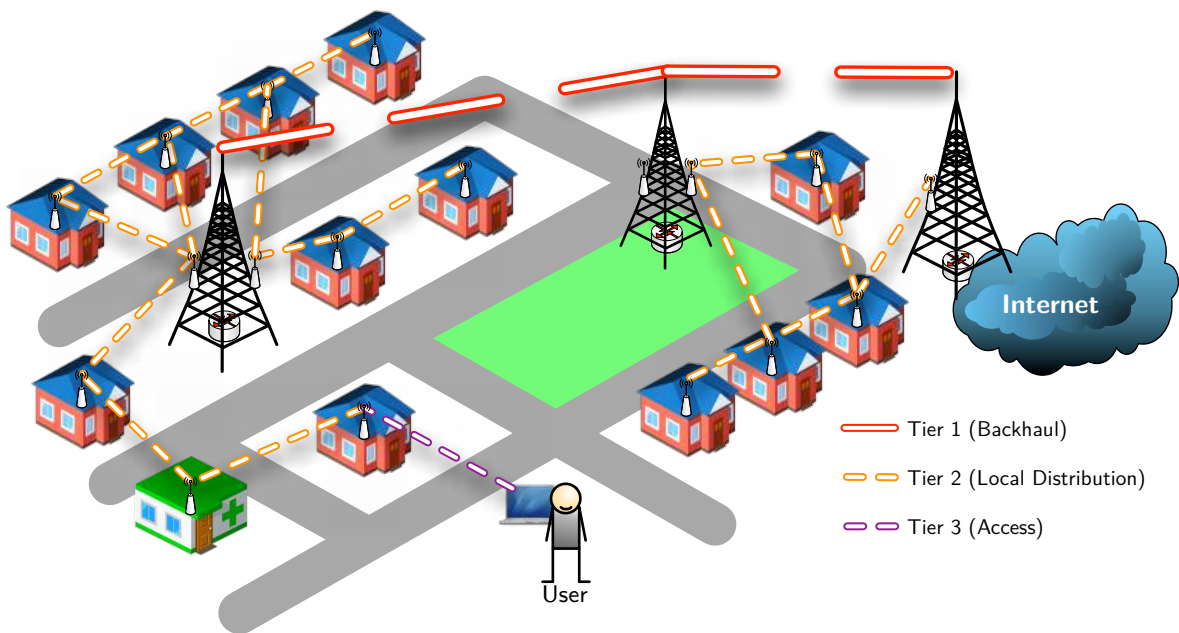


FIGURE 2.4 – Three tier Wireless Mesh Network providing connectivity to a neighborhood.

current point of attachment, their gateway, and do not need to support additional routing solutions. The operation of the routing protocol is confined to the backhaul transport tiers, which are composed by static nodes, in pre-planned locations, thus leading to a more stable network, with higher level of service for users. Some protocols developed for WMNs take in consideration end terminals, and may eventually require specific software support. The objective of such solutions is to provide enhanced service, especially in the case of seamless user mobility [57].

While routing protocols for MANETs are mainly focused in providing routes in highly dynamic situations, routing protocols for WMNs focus in providing better load distribution, or higher resilience to variable link conditions. Both AODV and OLSR can be used in WMNs, and while OLSR is actually a popular solution for WMNs (due to the creation of the MPR distribution backbone), other specialized solutions are also frequently found [57–63]. Due to its use in several important mesh testbeds, a particularly important solution is Better Approach To Mobile Adhoc Networking (BATMAN) [63].

The BATMAN protocol aims to be an evolution from OLSR, with the aim of decentralizing network information. One of the limitations of OLSR, which is inherent of its link state approach, is that all nodes will have full information regarding links, and network structure. For larger networks, this implies more resources required for WMRs so that link information

is calculated and kept. BATMAN nodes will also send periodic advertisement message to their neighbors, and these messages are broadcasted to the entire network. The message can include information regarding multiple interfaces, gateway availability and even the existence of legacy (BATMAN unaware devices). Upon receiving an advertisement message, nodes will now register the originator node, as well as the actual neighbor which sent the message. By using the resulting table, nodes will be able to always forward packets in the correct direction. That is, in the direction from which a matching advertisement was received. Because no global distribution tree is established, BATMAN actually imposes lower computational requirements to devices. Moreover, topology changes will have little impact to overhead. An interesting characteristic is that topology changes, while imposing that the protocol must converge to new optimal routes, the protocol employs progressive conversion spreading from the location originating the change. Nodes which are far from the location originating the topology change will be mostly unaware of it and will route packets like they did in the past. When packet reaches a zone which is already aware of the change, the packet will be routed to the new location of the destination, using new routes. The resulting route will be temporarily sub-optimal. However, packet loss will be minimized [64]. Because BATMAN is to be deployed at the core of the WMN, where little or no mobility will exist, it has been actively used by several mesh deployments. This same idea of progressive convergence is also present in solutions for other protocols [57], also resulting in low packet loss and low routing overhead when dealing with topology changes.

2.2 NETWORK AND SYSTEMS MANAGEMENT

Many methodologies and mechanisms for managing distributed environments were developed in the recent years. These methodologies can be applied to many different systems and environments, and across different management levels, from business processes to systems. The actual instantiation of the methodology potentially results in a different set of mechanisms applied, but a set of concepts remain present across a large amount of methodologies. At all levels we can find long term and short term goals, as well as principles governing the actions of the participating entities, these entities are either systems, organizations, or actual individuals. One of these concepts of primordial importance for system management is the notion of policy, which can be applied either to systems directly, or to a group through the use of roles. The most relevant management methodologies, as well as policy management approaches, and role based concepts will be described in the following paragraphs.

2.2.1 MANAGEMENT METHODOLOGIES

Systems can be managed in different manners. The approach taken for management of a system will vary accordingly with the scenario and the burden accepted for management tasks. Simple systems may require simple administration by a user, while more complex, heterogeneous systems will necessarily make use of other, more capable approaches. Most of the management methodologies make use of policies or rules to some extent, especially in distributed systems. However, these two aspects are frequently orthogonal.

Traditionally it is considered that Static Management is the simplest one, following by Reactive and then Proactive management approaches. The most complex approaches, still presenting many research challenges follow autonomic principles, or manipulate structured information.

2.2.1.1 STATIC MANAGEMENT

Systems may operate in quasi-static environments where little or no changes occur with time, or changes are well known and constrained to a range of metrics. For these systems, management is simple as the entire future of the environment can be known at design time. System designers had always knew what is the expected behavior of a managed element, what is the exact execution environment, and what values or algorithms need to exist so that the element operates. As an example, an Ethernet Hub requires little or no prior configuration and will operate in a highly predictable manner, as long as the environment keeps within a stable operational range. Hubs regenerate and forward packets in a way that is easily modeled and predictable. Therefore, it may be said that static management can be applied to systems where all relevant variables are static within the operational range, and known at design time. Roles and rules may exist but they are implicit in the functionality the device supports (instead of being defined in its operational attributes).

The benefits of statically managed systems are low device complexity, and low device cost, because only a reduced set of possible environmental conditions is supported. The major setback for static management is that devices will continue to operate in the same manner, until reconfigured in order to behave differently (if supported), either by a controlling system or by a human operator. Moreover, the human operator or the controlling system must reason over knowledge and know what configurations to apply (if any configuration is supported). The result is limited reaction to changes or high expertise required for the human operator.

2.2.1.2 REACTIVE MANAGEMENT

Reactive systems enhance statically managed systems by allowing alternative operational modes given predetermined conditions. A reactive system will change its behavior in response to information sensed from within itself, or from the environment. They are event driven as all behavior can be determined by analyzing the inputs and the changes over these inputs. Finite State Machines (FSMs) are frequently used to model reactive systems, as variations in its inputs can be mapped to a state change. Reactive systems are therefore mostly deterministic, just like static systems. However, they can maintain operation even when the surrounding environment changes, as long as changes were predicted at design time and predicted in the existing rules, or some entity modifies the state or conditions. Reactive systems will operate just as static systems as long as they do not change state, but different states may imply different behaviors.

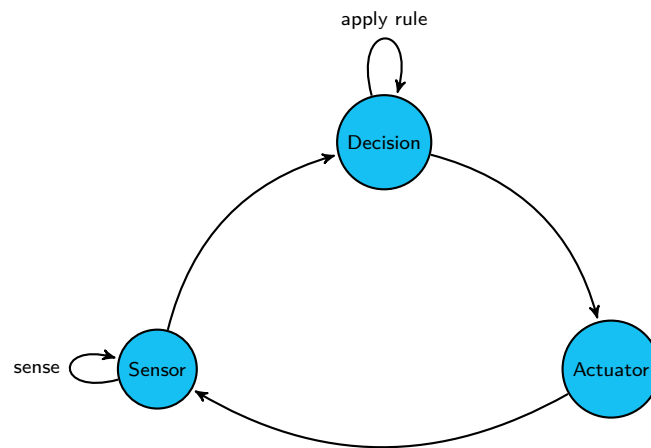


FIGURE 2.5 – The feedback loop present in reactive systems.

The core of reactive systems is composed by four basic constructs: A Control Loop, Sensors, Actuators and Rules (Decision) (see Figure 2.5). There is no notion of past memory, and the system reacts to present events according to the rules defined. The loop describes the state of the system, not actually how information flows. It should be taken in consideration that these rules are short term and do not try to predict future operation. One example of a reactive system is the channel selection algorithm commonly available in Small-Office, Home-Office (SOHO) Wi-Fi alliance certified (Wi-Fi) routers. If interference is detected in the current channel and is above a predefined threshold, devices will simply scan over the entire spectrum and settle at the channel with less interference (estimated by measuring radio power, access delay or other method). This approach is able to solve interference problems caused by a new device added to the neighborhood, when the number of clients is low, or the

source of interference is rather static (e.g., a microwave oven which operates over channels 10-13), but is unable to provide a reasonable solution for a congested area or with variable number of clients. Several other solutions evolving this model are being researched, with the aim of making the process more robust. A thorough review of the existing solutions can be found in [65].

Many real systems are managed following a reactive approach. Even in the field of human behavior, individuals can react instinctively, and manage system in a reactive manner. Many examples can be found every time that problems are solved as they occur, and no further problem avoidance actions are taken. Reactive management has the benefit that manageable elements can still be cost effective and simple. In fact, manageable elements may only allow static configuration (i.e., act as enforcement point). The main difference is that some entity is able to sense the environment and change operational parameters, or take some action, in order to maintain the system operating with the desired behavior.

Reactive functionality can be distributed in all managed resources, which allows the deployment of systems that are at the same time, simple and decentralized, while being able to react to environmental changes. Such systems can even take in consideration the internal state of neighboring managed resources, providing an output that tries to maximize global utility, instead of only maximizing local utility. Moreover, as it is the case of remote sensing applications (e.g., detection of forest fires [66]), or routing solutions for decentralized ad-hoc or mesh networks [41, 67], considering the internal state of neighbor elements is not an optional functionality, and is effectively required for basic operation of a system.

Reactive functionality may not be required for all components of a system, but only at a centralized coordination point. Often this is desired if future scalability is not a concern. Managed elements can still be simple, and data can be sensed directly from the same resources, or from specialized resources. Another benefit is that the decision point can have both global and detailed views of the system, and can more easily correlate different metrics, from different sources. Therefore, complex rules can be created, including different metrics from different systems.

Commercially deployed networked systems frequently follow this approach, by means of a set of specially crafted solutions. Most common solutions are the ones using protocols such as Simple Network Management Protocol (SNMP) [68], which is without doubt *de facto* standard. Web-Based Enterprise Management (WBEM) [69] which is used for querying operational variables, receive event information, and issue state changes. Internet Protocol Flow Information eXport (IPFIX) [70] which is used for monitoring Internet Protocol (IPv4) [71] traffic flows, and NETwork CONFiguration protocol (NETCONF) [72] or Common Open Policy Service (COPS) [73] which are two protocols used for setting policies to distributed

managed systems. Also, NETCONF plans to supersede SNMP, as it provides some of the same functionality. However, because SNMP just works, and NETCONF still has some challenges ahead [74], the transition to NETCONF may suffer some delay.

In some situations, where no automatic reaction is required, a human frequently has the task of checking different metrics at the manageable resources, and modify the current policy as he seems fit. Organizations, groups and communities, can also present reactive behavior. In particular due to the dynamic characteristics of social relations, commercial fluctuations, and individual prospects, the set of rules can evolve continuously in reaction to changes, so that the system is better fit for the current reality.

2.2.1.3 PREDICTIVE MANAGEMENT

Reactive systems are able to change its output given the past, sensed data and the current state. Predictive (or Pro-Active) systems enhance this approach by aiming towards a future state. Instead of simply evolving current operation, by applying the current set of rules in reaction to sensed information, predictive management solutions also target a potential future scenario. Managing pro-actively implies the creation of rules that tackle both the present state and the future self. The purpose is to minimize the need for reaction to changes in the environment, and potentially maximize utility. The feedback loop of a typical predictive system is depicted in Figure 2.6. As observed, the feedback loop is similar to the previous, but enhances the model used in reactive systems by considering a heuristic, which aims at an optimal future state. Therefore, the rules applied are not simply based on reaction to events but also on the current and future states.

To illustrate the difference between reactive and predictive systems, and revisiting routing protocols for MANETs, it may be considered the two most well known routing protocols for mobile ad-hoc networks: AODV and OLSR. AODV is a reactive solution, it only builds local routing information in reaction to the existence of packets waiting to be sent to a destination address. If no packets are to be sent, the protocol will create and maintain neighborhood relations, but it will not determine any route between two points. OLSR, being a pro-active solution, assumes that in the near future, there will be the need for routes between all existing communicating nodes. The result is that, in environments with no mobility AODV takes longer to discover routes (they are created as required). Because OLSR predicted a network which is fully connected, it is able to provide routes almost instantaneously. However, because OLSR aims at a stable network, its performance is lower when factors such as mobility are considered [75].

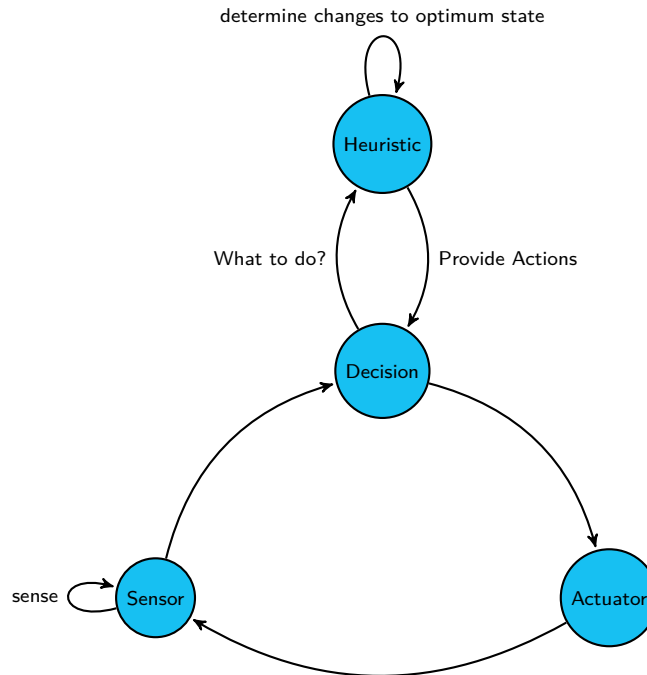


FIGURE 2.6 – The feedback loop present in predictive systems.

2.2.1.4 AUTONOMIC MANAGEMENT

In 2001 IBM coined the term Autonomic Computing (AC) in its manifesto [76]. AC aimed to further improve previous management approaches by proposing systems that are not tied to a particular structure or methodology at development time. That is, systems that have self-regulating capabilities (self-healing, self-configuration, self-protection and self-optimization), and can adapt to very different situations, even if they were not predicted at design time. The primordial purpose of autonomic management is to support increasingly complex and dynamic (software) systems, by mimicking neurological functions in biological systems [77]. Some authors argue that "... the growing complexity of the IT infrastructure threatens to undermine the very benefits information technology aims to provide" [76]. This is particularly relevant given that computational power keeps following Moore's law [78], dragging software design complexity together with the amount of computational power and information available. Instead of proposing components that represent the current structure of information and data flows, autonomic components are more generic and operate over sensed data, reacting to it while taking in consideration an existing policy in a closed loop (see Figure 2.7), and existing knowledge while predicting the future environment.

Autonomic systems can take in consideration past history and predict future events to

some extent, or at least take past events in consideration when reacting to other events, potentially reacting differently, or not reacting at all. Also, instead of abruptly changing behavior, the policy driving an autonomic system evolves: mimicking human behavior, and human collaboration. This policy, which is composed by roles, rules, permissions, attributes and other easily modifiable information constructs (in opposition to compiled software code), can be adapted in small steps following changes to their environment, or following changes to business goals.

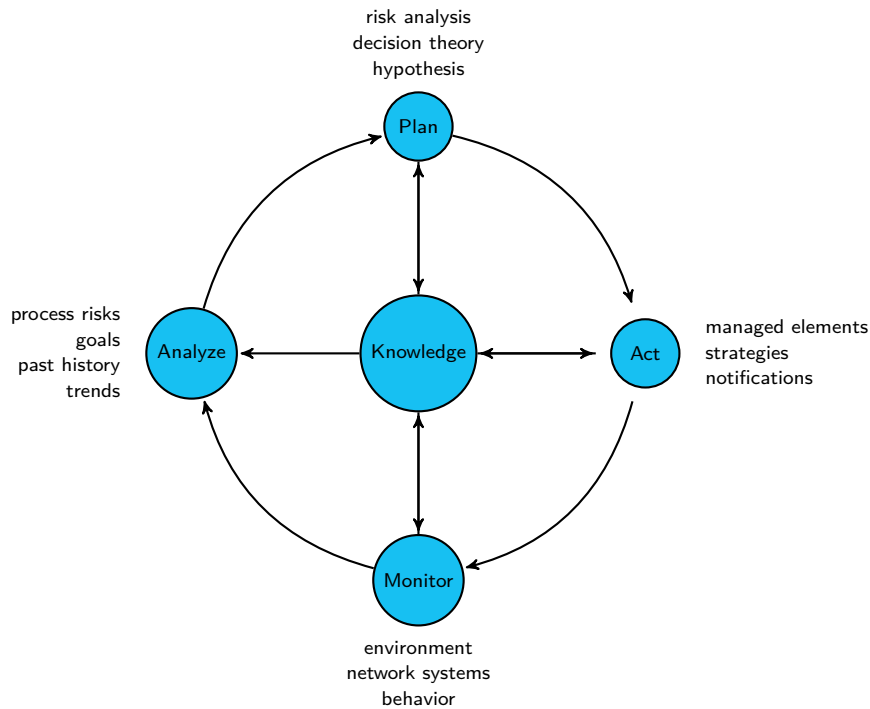


FIGURE 2.7 – High level structure of an autonomic based management system (MAPE-K loop).

There are several basic guidelines for autonomic management, and in particular, autonomic computing. The first guideline, which is of vital importance, states that nodes must know themselves. Without knowledge of the resource structure and attributes, it becomes difficult (if not impossible) to manage it. This does not imply that nodes are self aware and have a conscience, only that they should have a detailed representation of each service and resource they own. This includes: status and other operational information specifying which amount of resources are being consumed and which amount is still free; which services are active; and what is the status of their operational variables. The internal knowledge can also

include administrative or policy aspects, what limits are imposed in terms of resource usage (e.g., quotas), which services can be activated, and to whom they can be provided, under what conditions.

The second guideline is the capability for adaptation to changes in the policy or in the environment. Taking in consideration the knowledge of themselves, that is: the present operational state, and what can be modified under which constraints. Autonomic systems can reconfigure themselves in order to optimize their operation to changing environment conditions. Moreover, nodes should be under constant adaptation. Continual optimization requires both knowledge of themselves, the capability to adapt, but also a feedback loop providing information about how services are being provided and resources being used (see Figure 2.7).

With the increased adaptation capabilities and the existence of a feedback loop, comes the need of a system to maintain their operational state. That is, adaptation to changes in the environment cannot lead to disruption of the basic operations of a node. This would render the node inoperative or lead to improper behavior, which could compromise the entire collective. Therefore, as an additional guideline, autonomic systems should have the capability of healing and protecting themselves. Heal and protect are different concepts, with similar consequences if not present. Continued adaptation can lead to end-points where no further adaptation is possible, even when operation is far from optimal. A more global reassessment of the local conditions must take place, and break the concept of continuous evolution. Protection is important because ambiguous, malformed or otherwise misleading or malicious policies or behaviors can be present. In this case, nodes must be able to sort right from wrong and react against these issues (e.g., ignoring the threat, actively punishing the offender, or correcting the problem).

Another guideline states that autonomic systems are also expected to know their surrounding environment. That is, they are expected to have a representation of their external environment. Which neighbors are present, what services they provide, what resources they have. Taking in consideration this information, nodes can rely on other to optimize global operation or even to provide more complex, composed services. As with other management approaches, reaction to the changing environment can also predict future conditions by identifying patterns, and the collective system can act in a coordinated way towards optimal operation. Solutions for representing and exchanging information such as Extensible Markup Language (XML), JavaScript Object Notation (JSON) [79], or web services through a public Web Services Description Language (WSDL) [80] are explored since a long time and, due to their flexibility and ubiquity, advocated for use in autonomic systems.

The number of solutions aiming to provide autonomic (or self-*) functionalities for net-

work management is vast [81]. However, many solutions self-named as autonomic, do not follow the principles first stated by IBM. One example is [82] which aims at being an “Autonomic Wireless Mesh Network”. In practice, it relies in well known mechanisms such as Domain Name System (DNS) [83], Dynamic Host Configuration Protocol (DHCP) [84], and Network Address and Port Translation (NAPT) [85] together with a predefined configuration in order to allow simple deployment of new nodes to the network, and automatically provide connectivity between mesh deployments and with the Internet. Even though it also supports a fault-resilient chat application supported by [86], there is no support for any of the principles stated previously. Like this solution, many others aiming at providing autonomic behavior fail to create a proper formal description of the system and fail to understand how to put it in practice. It is then clear that there are still several important research challenges regarding autonomic systems, impairing their complete functional realization, such as management, scalability and intelligence. One of these challenges is that artificial intelligence techniques are required for developing autonomic systems, an area that also faces its own research challenges. The complexity of the resulting computational framework, as well as the amount of information generated by the system (its memory) impose the need for powerful computational platforms, and with large amounts of storage. In this manner, scalability must be tackled so that, the system keeps operational at a future time. Another important aspect is the translation from long term goals to effective configurations. This requires automatic (without human intervention) semantic interpretation of the goals, and the adaptation to each specific scenario, devices capabilities and applications, which is still a research challenge.

2.2.1.5 KNOWLEDGE BASED MANAGEMENT

A recent set of approaches aims to add some intelligence and self-learning capabilities to network entities, by means of autonomous semantic interpretation of knowledge. Solutions like [87] propose the creation of a knowledge plane in order to eliminate the traditional layered structure of a standard communication stack. The knowledge layer is considered to exist in parallel to other layers and can be used to shape execution in a completely multilevel, and vertical manner. Given an expected behavior, and the information present in the knowledge layer, the network stack (or any other part of a system) should apply it, choosing the most appropriate set of operational directives. Two similar policies can be applied in a completely different manner, because knowledge of each domain may be very different, and should have no relation to the actual instantiation of the internals of the system. The knowledge system is the sole responsible for translate the high level policy to actual actions, taking in consideration existing knowledge. Considering systems managed in an autonomic manner, knowledge sits in the middle of other functions, impacting how each function operates.

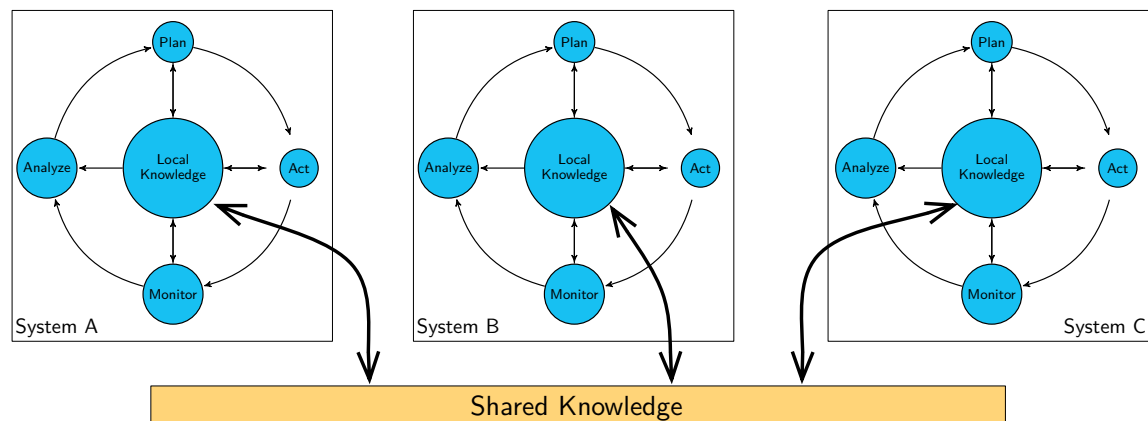


FIGURE 2.8 – High level structure of a knowledge based management system.

A relevant aspect of the knowledge plane is that it is not only a vertical cut inside a single system. The knowledge plane can be shared among a set of systems which are managed under the same context and domain. Therefore, the same high level policies are common to all systems, and are applied according to the specific characteristics of each system, considering its location, owner, or any other attributable characteristic. It is possible to consider the existence of different roles so that different systems apply different policies, or a single policy which is translated to low level instructions on demand. An advantage of the knowledge plane is that it actually constitutes a side-band communication bus that may be used by systems to coordinate execution. Moreover, the intelligence of a system using a knowledge plane can increase over time simply because more knowledge regarding the past is present. Systems gain more experience. Taking in consideration the collective history maintained by all systems, it becomes easier to predict the future and act in a proactive manner, avoiding pitfalls of the past.

One implication of these kind of approaches is the additional effort in order to make the knowledge plane coherent among the different systems. If a system adds a knowledge object to the knowledge plane, this object should be correct, and immediately available to others without errors or without being wrongly manipulated by others. Representation of management information and of knowledge information, as well as all manageable or otherwise interacting entities, in common and unambiguous terms, is vital for knowledge based systems. Moreover, while knowledge represents facts as observed by systems, it is still required to specify how they effect the conversion from business goals to specific actions, and how to reason over the observed facts [88].

Another important aspect of solutions using the concept of a knowledge plane, is that

systems must be able to operate in the presence of partial, inconsistent and even misleading information. A common problematic circumstance occurs when different stakeholders create information leading to goals that are impossible to achieve. Even worse, information may have been created by a malicious user to subvert network operation towards his own objectives. Resorting to asymmetric cryptographic methods, to sign information elements, may assure participants that an information element was created by the correct user, instead of a malicious foreign user. A drawback is that it imposes the existence of a preexisting Public Key Infrastructure (PKI) constantly accessible to all entities. And even if the PKI is available, and messages are signed and authenticated, the use of cryptographic signatures solely ensures that the information element was created on a device with access to a specific cryptographic material (e.g., the private key if using asymmetric cryptography). It says nothing about the intentions of the user or even the correctness of the software and hardware. Without resorting to Trusted Platform Module (TPM) [89] or other hardware based invulnerable module, cryptographic material may still be stolen, resulting in the possibility of forging information. Even if all these issues are considered, the resulting policy can still be incoherent with the high level principles, either at a local or more global scale.

2.2.2 POLICY BASED MANAGEMENT

Together with increasingly higher complexity, the task of managing information, resources and the orchestration of the different modules has increased very rapidly. Initially, management was performed by resorting to careful planning of the deployment scenario, and to static configuration of systems. In some complex scenarios, simple solutions such as SNMP were, and still are, used for distributed management. Later, management of more complex scenarios evolved to also more powerful solutions based on autonomic principles and reasoning over collective knowledge.

A common aspect of all management methodologies is the taxonomy used for management principles. Simple systems do not actually use the same language, mostly due to the low complexity of the management tasks but most systems that are used in recent production environments, and independently of the actual solution they use to manage the inherent complexity, share similar concepts, constructed around the concept of a policy. Therefore, these systems are named policy based management systems. A policy has been traditionally defined as a concept which is "... derived from the goals of management and define the desired behavior of distributed heterogeneous systems and networks" [90]. The base idea is that policies are means to guide managed elements according to management or business goals [91]. A policy defines what is the goal that management wishes to accomplish, and how

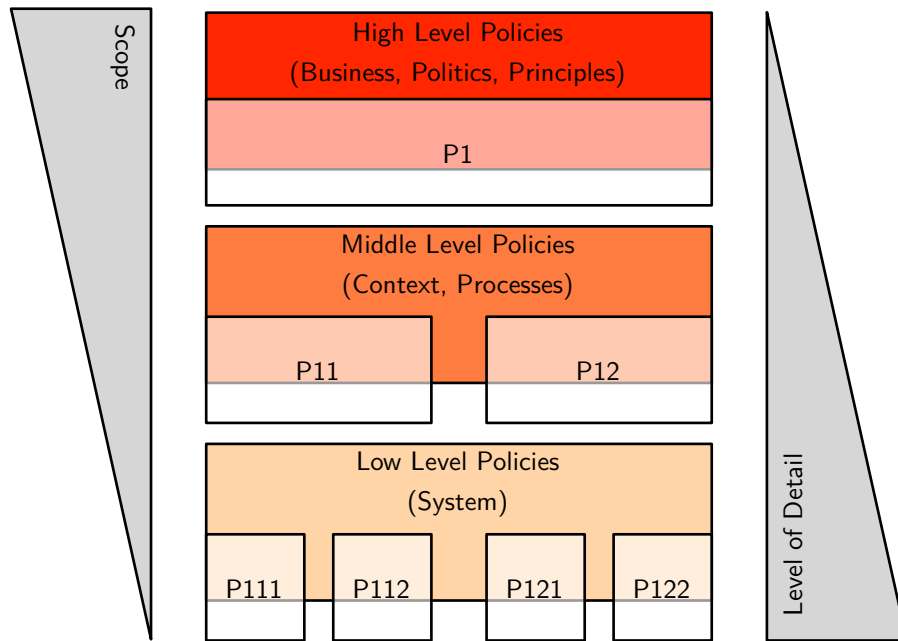


FIGURE 2.9 – Common policy decomposition from high level policies to low level policies.

this goal is to be achieved. The actual protocols, operational details, or specific information models are out of scope of the policy. However, policies must be mapped into some form of practical representation and may become closer to the actual implementation details, or operational model of a given system. In communication systems, this often results in an algorithm and/or manipulation of the operational aspects of an algorithm. Policies allow a system to change their operational behavior without modifying the actual implementation, i.e. the system is able to adapt to novel situations by modification of its operational parameters, while maintaining its underlying implementation.

In the real world, the actual definition of a policy is more complex. Policies can be represented at many different abstraction levels and using many different frameworks. Corporations can define their policy towards long term business goals or regarding client relationship, while policies can be created to define the actual behavior of systems, down to a very low level (see Figure 2.9). Most authors define the existence of high level policies that exist at a business level, and low level policies that exist at systems level [92]. In this case, it is considered that the concatenation of several low level policies will satisfy one high level policy [93]. However, this bipolar categorization may not always be possible due to existence of intermediate levels of policies with increasing detail, or adaptation to specific scenarios.

The Internet Engineering Task Force (IETF) created the Policy Framework charter to

formalize, and standardize policy related concepts. One of the results of this charter was the definition of the terminology related to Policy Based Management (PBM), and how these concepts should be employed in management scenarios. From all the proposed concepts, the ones most relevant to this work are briefly described in the following paragraphs, and depicted in Figure 2.10.

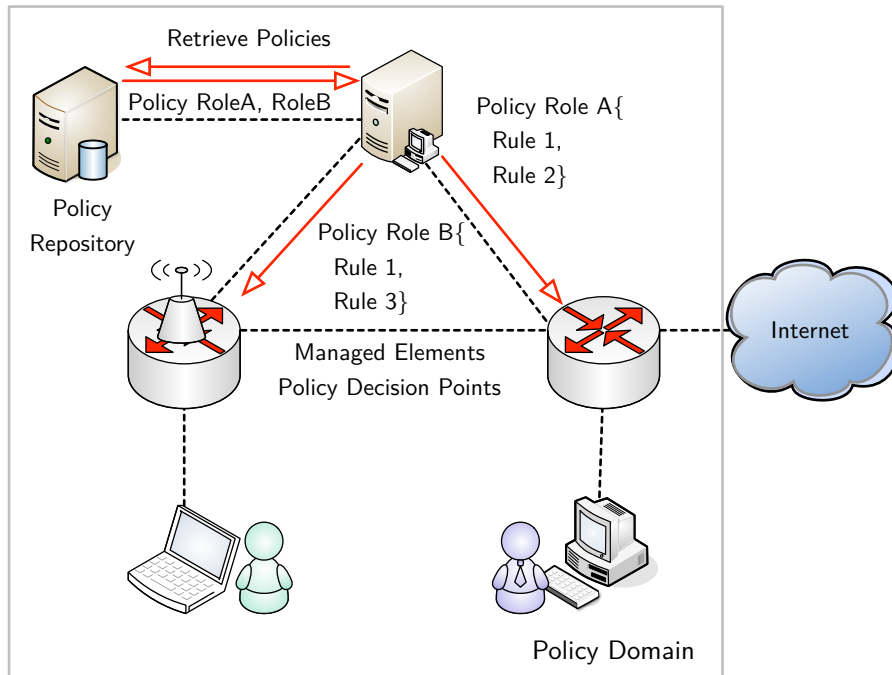


FIGURE 2.10 – Some of the basic components of the architecture for Policy Based Management.

- A Policy Decision Point (PDP) is “A logical entity that makes policy decisions for itself or for other network elements that request such decisions” [94]. According to the definition, the PDP can be any entity (system or human) which decides how to apply an existing policy. That is, the entity that taking in consideration the current environment and the goals described in the policy, decides how others behave. Systems with autonomic capabilities (see Section 2.2.1.4) can decide how to proceed, while simpler systems are unable to convert policies to concrete actions without external support.
- Policy Enforcement Points (PEPs) are logical entities that enforce policy decisions [94]. Policies are converted to decisions, and these decisions are applied at the PEPs elements. From a networking perspective, the PEP is a router which is affected by the policy. Enforcing it results in modifying the behavior so that it complies with the policies (e.g.,

block traffic of a specific type, authorize user). Systems can act at the same time as PDP and PEP. However, simpler systems will only act as PEP, enforcing policies provided by other systems, or directly by a human system administrator.

- A Policy Domain (PD) is “A collection of elements and services, and/or a portion of an Internet over which a common and consistent set of policies are administered in a coordinated fashion” [95]. Policies are valid only in a given domain, and their consistency must only be maintained inside a domain. Therefore, if a system is under a given policy, inherently it belongs to a specific domain, but it may belong to several, given that the policies are compatible. If a two policies from different domains collide in a given system, an administrative action must be taken to solve the collision. One of the solutions may be the removal of the system from one of the PDs.
- A Policy Repository (PR) is “A specific data store that holds policy rules, their conditions and actions, and related policy data” [96]. Because policies must be available to the PDP, the PR is required in order to store all information related to the policy of the current domain. There are many examples of systems providing PR functionalities, ranging from directories and databases, to files in folders in filesystems. Even a logical entity providing storage and retrieval mechanisms can act as a PR. At a very high level, a country’s constitution can be considered a policy repository storing high level policies.
- Roles are “administratively specified characteristics of a managed element. . . . It is a selector for policy rules. . . , to determine the applicability of the rule. . . to a particular managed element” [97]. Therefore, roles enable the categorization of a managed element and the applications of rules depending on this categorization. It allows to have distinct policy sub-domains inside the same PD, according to the attributes of a Managed Element (ME), thus refining the effectiveness of the high level policy to specific targets.
- A Rule is “a basic building block of a policy-based system. It is the binding of a set of actions to a set of conditions — where the conditions are evaluated to determine whether the actions are performed” [97]. Rules are applied to managed elements either through the high level policy, or through the sub-policy defined in a specific role. A common approach for rules is to follow the Event-Condition-Action (ECA) [98] principle which allows reaction to events in a simple manner: an event which is detected by a ME produces quantified values, one or more Conditions are evaluated over these values and may trigger Actions. The event itself may be enough for triggering the action (e.g., disassociation from the Access Point).
- Finally, a Managed Element (ME) is a logical entity or a component of a logical entity which can be managed by a policy. Only MEs can be managed, and they must present some form of interface allowing for local or remote transcription of the proposed actions into actual changes to operational parameters. Also, ME can belong to one or more Roles according to the policy and their characteristics. In this case there is the possibility

of collision between individual policy rules. This is not addressed by the policy itself but is the responsibility of the policy creator

Taking in consideration the above concepts and definitions, many management solutions are proposed in the literature. Each of these solutions is adapted to a particular use case, and focused in a particular interaction model. Two solutions, namely Ponder [99] and CIM Simplified Policy Language (CIM-SPL) [100] are of particular interest as they aim at a broad scope, operate over generic policies, and have a mature specification. In order to illustrate the basic concepts of PBM for management of dynamic communities, network services and resources, both Ponder and CIM-SPL are well suited, and will be described with more detail.

2.2.2.1 PONDER

Ponder was proposed by Damianou *et al.* [99], however it draws upon work covering almost an entire decade [101, 102]. The authors envisioned a set of large scale distributed systems where it was required to enforce a particular behavior. Due to the complexity of the systems, it was also required to modify the expected behavior dynamically, allowing systems to adapt to changes in the environment. The Ponder language is declarative, object oriented, and aims at enforcing behavior using both management and security policies. All policies defined are enclosed in a policy domain or a sub-domain. A domain can have multiple sub-domains, and a single sub-domain can belong to several domains. At each level, domains have explicit references to every object they contain (sub-domain, policy, or subject), but not to the objects belonging to their sub-domains. This is similar to a filesystem where each directory has a reference to all files and sub-directories contained, but not to the objects inside its sub-directories. According to the authors' definition "Policies are rules governing the choices in behavior of system" [103].

Primitive policies are the most basic management elements in Ponder, and define access control and other basic policy constrains. There are several type of primitive policies, all sharing a similar principles: Authorization policies define access control; Information filtering policies apply filters modifying the information exchanged in a given action (e.g., authentication in an TELNET session); Delegation policies limit users to some restriction or grant privileges; Refrain policies indicate the set of actions that a subject mustn't do due to some issue not related to actual authorization. Obligation policies specify actions that must be carried by subjects. Any primitive policy requires the identification of the subject, the action which the subject can or cannot perform, a target and a time constrain. Besides these fields, the policy also states if it is a positive policy (i.e., the subject is explicitly allowed) or a negative policy (i.e., the subject is explicitly forbidden).

The approach followed for enforcing a particular behavior is based on ECA rules. Because individual control over systems is not scalable, and the authors aimed at high scalability, there is a strong focus in using roles as means of grouping subjects according to their tasks, permissions, or even business responsibilities.

```
inst auth- /negativeAuth/testRouters {  
  subject /testEngineers/trainee;  
  action performance_tests();  
  target <routerT> /routers;  
  when time.between('0900','1700');  
}
```

CODE 2.1 – Ponder authorization policy.

Code 2.1 depicts a typical authorization policy (*auth*) stating that subjects of type *trainee* cannot do *performance_tests()* on routers between 9 and 17 hours. The negative aspect of the policy is stated by the “-” operator. The different policy types will use different fields as appropriate for defining the policy. As an example, while Authorization policies follow a structure *subject, doing action, to target, at time*, Obligation Policies are defined following the structure *on event, of subject, to target, do action*. Like in other policy systems, the consistency and meaning of the policy must be assured by the domain administrator.

Beside primitive policies, Ponder also supports complex policies by means of Groups, which aggregate multiple policies in the same object. Roles are similar to Groups with the difference that roles reflect the organizational structure of a business. As an example, *Engineer* would be a role, while *MailUsers* would be a group policy. These concepts have a hierarchical structure, by means of inheritance, and can be used to further specify a policy of a subject. Engineers are also Employees, therefore Engineer would inherit from the Employee Role.

Ponder is very rich, allowing the specification of a large amount of policy constructs. These allow limiting access, enforcing obligations, filters and so on, while at the same time providing clustering of policies to easy management. However the authors of Ponder also considered how the structure and relations between members of an organization could be mapped in the language. For this purpose they proposed two additions to the language: Management Structures and Relationships. Management Structures map the actual organizational branch structure of a business, and can be used to specify to great detail the several branches of a corporation (e.g., departments, or sales points). It becomes possible to define, per branch, which roles are available and what is the structure they follow in terms of relationships between roles. Relationships can further define the behavior inside a domain as they can be

used to define interaction specific policies. That is, policies that only apply when two subjects of two specific roles are interacting.

2.2.2.2 CIM-SPL

CIM-SPL is a simple language for the representation of policies created by the Distributed Management Task Force (DMTF) Policy Workgroup and centered around the Common Information Model (CIM) [104]. CIM is a solution based on the concept of generic managed objects, and is composed by two parts: schema and specification. The CIM Schema represents generic, implementation independent, aspects related to management. CIM Specification describes how CIM is used with the different management models. It makes use of Unified Modeling Language (UML) [105] and Managed Object Format (MOF) to describe the properties, relationships and classes of managed objects, to which is then possible to apply a policy.

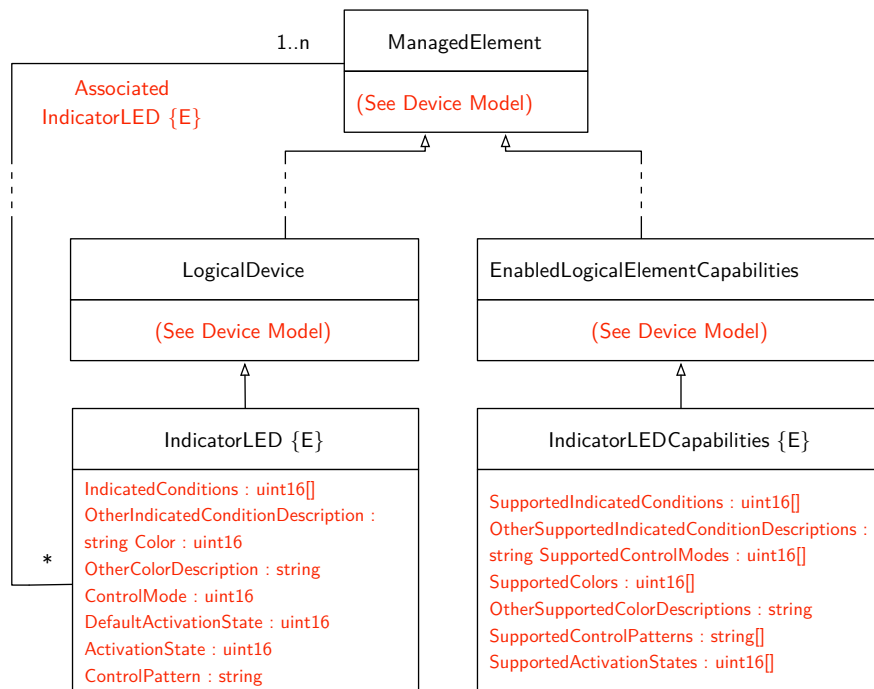


FIGURE 2.11 – Simplified representation of a single LED using CIM.

In CIM there is a Core Model defining the basic concepts, and many specific models for applications, databases, devices and other relevant types of entities. Each model provides a clear description of the methods supported, configuration parameters, resources and available statistics. Because it is object oriented, it supports the traditional concepts of composition,

inheritance, reference, relation and even abstract classes. Figure 2.11 depicts a simplified CIM representation of a Light Emitting Diode (LED) using the CIM Device Model [106] (some more relations of inheritance with upper layer models are missing). Also it should be noticed that a device such as a laptop will have a model for all of its internal components (e.g., all the LEDs, disk drives, etc. . .), making CIM at the same time very complete and complex.

CIM-SPL takes in consideration the previous work developed around Ponder, and other policy languages, and proposes a simple language for creating ECA statements to manage CIM aware systems. The structure of CIM-SPL is much simpler than Ponder. Ponder allows richer policy mapping between an existing organizational structure and managed element. However, this is well balanced by the complexity of and detail level of CIM MOF models. The basic unit of CIM-SPL is the rule, and all policies are described using rules. As it is based on the CIM schema, it allows the creation of simple rules which are then organized in a hierarchical approach to form the complete policy. Rules are divided in four sections: *Import*, *Strategy*, *Declaration* and *Policy*.

- The *Import* section defines the namespace to which is used for look for the objects invoked in the next sections. This is similar to the `import` keyword in the Java language.
- The *Strategy* section defines how the policies should be applied. Two strategies are mandatory: `Execute_All_Applicable` which states that all rules should be applied, and `Execute_First_Applicable` which stops after the first condition is met (ordered by priority).
- The *Declaration* section is used to define custom macros and variables that may be reused by one or more rules. Declarations are not mandatory and are just used to simplify the description of the policy. In the example presented in Code 2.2, it is depicted how an alias (`installDate`) and a macro (`Long Age(DATETIME Born)`) can be defined, which later are used in the definition of a *Policy*
- The most important section is the *Policy* section. Here it is defined the *Condition* which triggers the rule and the action (*Decision*) to take when the *Condition* is met. In Code 2.2 it is depicted how a logical operation is composed by using the macros previously defined in the *Declaration* section, and the action to take (`Upgrade(SKU)`). In the end of each policy it is stated its priority (1 in the example provided). This value must be unique for all policies and is important for defining the order by which they are executed.

```

Import SAMPLE CIM_V_2_8_CIM_Core28-Final::PhysicalElement;

Strategy Execute_All_Applicable;

Declaration{
  InstallDate="ManagedSystemElement.InstallDate";
  Macro {
    Name = Age;
    Type = Long;
    Arguments = Born:DATETIME;
    Procedure = getYear(CurrentDate) - getYear(Born);
  }
}
Policy {
  Condition {
    4 > Age(InstallDate) AND
    VendorEquipmentType == "switch"
  }
  Decision { Upgrade (SKU) }
}:1

```

CODE 2.2 – Example policy using CIM-SPL.

Although not depicted in Code 2.2 another statement can be present: *PolicyGroup*. This is a simple container for multiple policies, allowing nested policies and overall prioritization of multiple policies (see Code 2.3). Like in the case of single policies, the Priority field determines the order they are evaluated, and there cannot be more than only policy group with the same priority.

```

PolicyGroup: [AssociationName(Property1,Property2)]{...}:Priority;
PolicyGroup: [AssociationName(Property1,Property2)]{...}:Priority;
PolicyGroup: [AssociationName(Property1,Property2)]{...}:Priority;

```

CODE 2.3 – Policy groups in CIM-SPL.

2.2.3 ROLE BASED ACCESS CONTROL

Access control models based on roles were proposed since the inception of computers, and used in multi-user environments to provide mechanisms restricting access to systems and resources. One of the most popular formalizations of the concept, applied to modern computer systems is Role-Based Access Control. At the core of Role-Based Access Control (RBAC) [107] is the notion that entities, which may be systems or users, are associated with

roles (just like MEs in a PBM system). Roles are (most importantly) composed by permissions, and entities interact by establishing sessions between themselves. These three aspects put together are able to provide a fine grained access control mechanism, adapted for many real life scenarios. In a given domain there are many roles, and many users (entities). Each user may have many roles and may establish many sessions with other entities. There is also no major issue for using rules in permissions as a mean of providing more refined access mechanisms, or even to enforce communication and behavior. Therefore, the RBAC model fits very well under a PBM solution, and is frequently present in such solutions.

At its inception, roles were envisioned as being the digital representation of the job function and hierarchy existing in corporations. Individuals are assigned to roles based on their position in the company hierarchy, tasks and responsibilities, as well as qualifications [108]. Individuals can easily be assigned to a specific role, and can even have different roles inside the same corporation. In this case, the corporation acts as the domain where roles are valid. In the RBAC model, it is expected that the structure composed by the delegations attributed to individuals will change very frequently, while the events of adding, removing or modifying a role occur less frequently. Delegations are related to the operational aspect of the policy (i.e., which and how many entities are available in a particular role) while roles are related to more highly level policies, and therefore closer to the company goals, principles, and functional structure.

The purpose of RBAC is to facilitate management of environments with heterogeneous and multiple participants, while allowing to review, and have some control over the structure of the resulting environment. As an example, an individual assigned to *Role A* can access a particular resource, but may not be able to modify it (i.e., may only have read only access), and many other individuals can have the same role. One or more individuals can have *Role AC* (Role A Coordinators) which allows them for coordinating the ones with *Role A*, and eventually tweak *Role A* towards the interest of the corporation. An individual belonging to *Role M* can monitor the delegations, and audit who can access which resource, but has no access to resources. Another aspects is that while users can belong to several roles, these roles may not be freely available, as enrollment may require fulfilling constraints present in each role.

The model can be easily extended to support behavioral rules instead of only permissions, as well as behavior auditing [99]. RBAC is neutral to the actual policy and its instantiation, only defines concepts and relations between concepts. Also, RBAC is not a “fits all model”. Because it is based on the concept of shared roles, it tends to homogenize its members, i.e., all members of a role are expected to have the same permissions (or the same behavior). Other models like Access Control Lists (ACLs) may provide finer granularity. As a valuable

characteristic, it should be noticed that RBAC considers systems, devices, individuals, as entities entitled for being assigned to a role. Therefore, the same access control mechanisms can be used to authorize all types of entities. Also, RBAC is not limited to enforcement of discretionary access as roles may contain obligations, actually enforcing behavioral control. Figure 2.12 depicts a situation where three users are entitled to at least one of three existing roles. Taking in consideration the role to which the user is assigned, he may have access to different resources (permissions), but he will also be tied to some obligations.

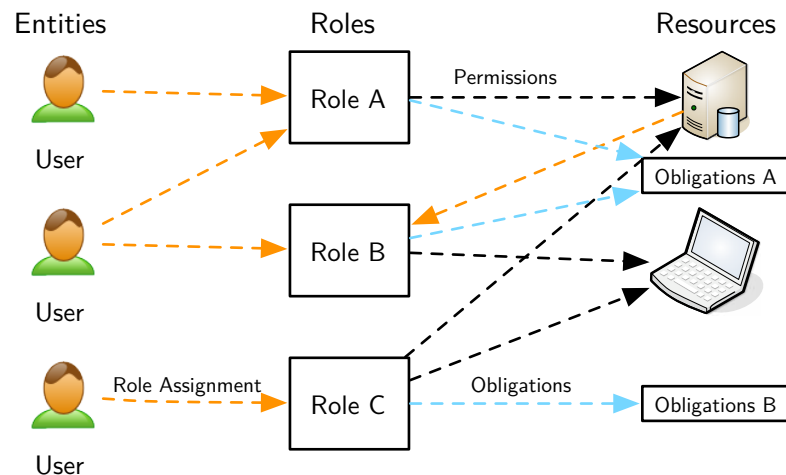


FIGURE 2.12 – Entities are entitled to roles which both grant access to resources and determine obligations.

In RBAC there are three important principles contributing to the relevance and applicability of this model: duty separation, data abstraction and least privilege.

The least privilege principle states that a user will only have access to resources if he is entitled to a specific role. Members of *Role A* can have some kind of access to a system, while members of *Role B* will have full access to the same resources but also to another set of resources (see Figure 2.12). Some information is therefore restricted to members of *Role B*. This is an application of the need-to-know basis, as members of a role will only have access to the resources available to their role(s). Individuals not assigned to a role will have no access to any resource. It should be noticed that while the RBAC model allows the existence of the least privilege principle, and it doesn't grant any permission *a priori* to users, if desired by its designer, an instantiation of the RBAC model can enable free access to one or more resources, violating this same principle.

The data abstraction principle states that the RBAC model should not be targeted towards the traditional permissions such as *can read*, *can write*, *can delete*, *can execute* (e.g., the POSIX

r-w-x permissions). Instead it states that permissions are granted, and they can have any meaning the manager wishes. If the RBAC model is applied to individuals, a permission may be the right to use the photocopier machine, while if the target is a communication system, the permission can state the right to establish a given IPv4 connection. This principle allows the application of the RBAC model across a wide range of scenarios. Also, it enables both the use of mandatory and discretionary access control methods, further increasing the capabilities of the model ¹.

The duty separation principle states that there may be incompatibilities between roles. If an entity has *Role A*, it may be blocked access to *Role B* until it relinquishes membership of *Role A*. The purpose of this concept is to limit abuse by means of collecting roles, and is a concept commonly found in a business environment, usually named conflict of interests. According to this concept, multiple interests can corrupt the motivation to act according to one of the interests. In the same manner, having a *Role A* can corrupt the individual to not respect *Role B*, and if the policy is sensible to this aspect, RBAC allows the definition of mutually exclusive roles.

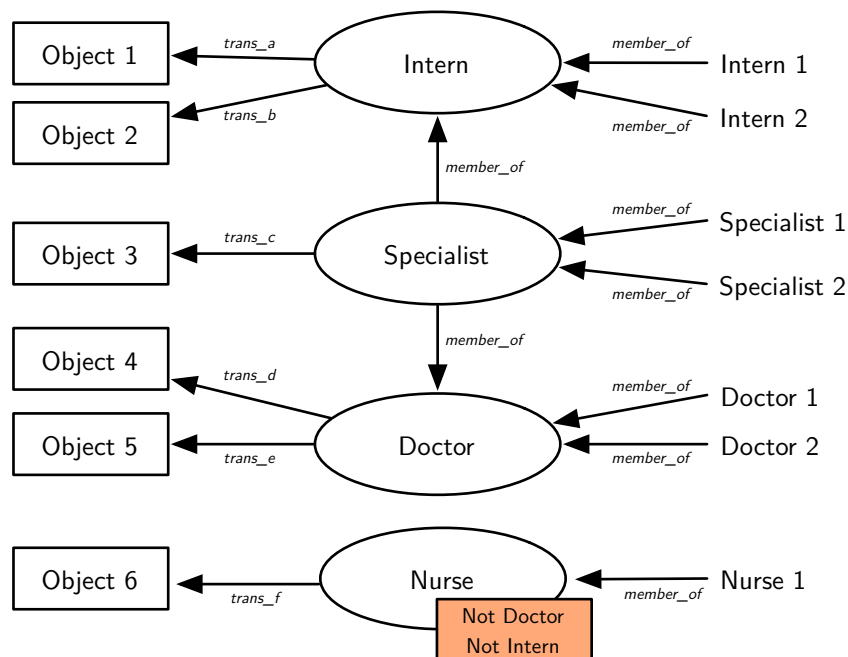


FIGURE 2.13 – RBAC applied to a simple hospital with doctors, specialists, interns and nurses.

¹For more information regarding discretionary and mandatory access models (DAC, MAC), as well as its relation with RBAC please see [109]

This is particularly important in military scenarios, but it can be applied to a wide set of situations [20]. Because in RBAC roles can be hierarchical, the duty separation principle must take in consideration all the parent and child roles related to all roles of a user, which brings complexity to the system. However, hierarchy is required in order to correctly represent real world situations. Figure 2.13 depicts the simple case of doctors, interns, nurses and specialists (e.g., Cardiologist) in a hospital. Individuals entitled to the role *Doctor* will have two specific permissions, while individuals entitled to the role *Intern* will have two other permissions which are not shared with doctors. A *Specialist* belongs both to *Doctor* and *Intern* and will have access to both permissions, additionally to their own permissions. However, specialists, doctors, and interns can't act as *Nurse*. The *Doctor* and *Intern* roles are incompatible with the *Nurse* role, and due to hierarchy, so is the *Specialist* role.

One of the solutions expanding the original RBAC model is Distributed Role Based Access Control (dRBAC) [20]. It aims at providing a scalable, decentralized, trust-management and access-control system for coalition scenarios. A coalition is an alliance of two entities, which cooperate towards a common objective, and is frequently found in military situations. In these, two or more countries, each possessing several different types of forces (i.e., air force, army, marines, etc. . .), decide to act together. The same situation can be found in a group of corporations, or even small groups of individuals, which cooperate while maintaining their identity inside the group. In all cases, there is no common administrative root for all forces, groups, or corporations, as each has its individual domain. During the coalition there is an alternative chain of command that is formed for the purpose of coordinating actions. Also, resources may be shared among members of the coalition. The problem of determining the permission to use the resource, or the authority to trigger an action, arises since the inception of the coalition. The authors of dRBAC, and under the scope of the large Distributed Coalitions Infrastructure (DisCo) [110] framework, propose a simple mechanism centered on a PKI, roles and delegations.

Each structure has all its ranks mapped into roles. Therefore, it can be clearly determined what are the permissions of a given individual, inside a given organization. Besides the permanent roles (e.g., Private, Colonel), transient roles can also be used to provide shorter term permissions (e.g., Private of 1st Battalion of Infantry). A construct named *Credential* (or *Delegation*) is used to associate a subject to an object, and a PKI helps asserting the validity of the *Credentials*. *Credentials* assert that an entity (the *Issuer*) delegates a given role (the *Object*) to another entity (the *Subject*) because this entity meets the conditions of the role. Because the system was designed for coalition scenarios, entities are strictly individuals, while roles are hierarchical positions, group of permissions, or actions. The proofs proving that the subject meets the requirements for the role are also made available for inspection. The syntax of a *Credential* is depicted in Code 2.4.

```
[Subject -> Object'] Issuer
' denotes the right to delegate (optional).
```

CODE 2.4 – *Credential* syntax proposed by the authors of dRBAC.

If an individual has a *Credential* associating it with the role of *Private*, then it will have all the permissions of that role. When interacting using dRBAC, individuals make use of the PKI to verify the identity of the corresponding individual, to prove their authenticity, and to verify the credentials presented. Considering a scenario where *ColonelA* provides a credential of role *Tank.drive* to *PrivateB*, and then *PrivateB* further delegates the same role to *PrivateC* and *SergeantD*, a delegation path is easily identified by following the issuer (see Code 2.6). But, most importantly, an auditor can police the path and (if the correct permissions are in place) revoke the delegation of *PrivateC* and *SergeantD*, just by revoking the credential emitted to *PrivateB*. *Credentials* are considered transient and can be easily created or revoked by its creator. The reasoning behind credentials is that it becomes simple to determine the entitlement of a role by an individual, to track the authorization path, and even to do mass revocation of permissions. It should be noticed that a credential is only considered valid if the entire path is also available and valid. In a distributed, non-structured system, this may impose some delays, however as proposed by Mahesh *et al.* [111] techniques such as cascaded Bloom filters² can provide thousands of verifications per second on SOHO devices.

```
[ColonelA -> Tank.drive'] ColonelA //Self certifying delegation
[PrivateB -> Tank.drive'] ColonelA
[PrivateC -> Tank.Drive'] PrivateB
[SergeantD -> Tank.Drive'] PrivateB
```

CODE 2.5 – Delegations in dRBAC.

dRBAC provides mechanisms to limit both entitlement to a role and re-delegation, by means of conditions and a delegation permission. Using the same example, when *ColonelA* delegates Role *Tank.drive* to *PrivateB*, he may or may not include the right to further delegate the credential (by not including the " ' " character), thus limiting *PrivateB* of creating new credentials for that role (see Code 2.6). Further limiting attributes can be added to the credential, which in the case of the scenario depicted in Code 2.6 would limit the max speed of the Tank to 10 km/h.

²assume an array of m bits and k uniform hash functions H_i returning a position in m , an element e may exist if all bits of computing $H_0(e)...H_k(e)$ in m are 1

```

[ColonelA -> Tank.drive'] ColonelA //Self certifying delegation

[PrivateB -> Tank.drive
  with Tank.speed<=10 ] ColonelA //No delegation, max speed 10km/h.

[PrivateC -> Tank.Drive] PrivateB //Invalid!
[SergeantD -> Tank.Drive] PrivateB //Invalid!

```

CODE 2.6 – Delegations can include restrictions both on resource usage and further delegation.

The dRBAC is simple yet it provides the required mechanisms for entitling access permissions inside the same domain, or even to entities of foreign domains. As depicted in the previous examples, there is no reference to the domain in the roles or entities as it depends on the namespace used. A coalition environment would operate either by creating a new namespace with the existing Entities and Roles at the root of the namespace (e.g., Tank, ColonelA), or add a prefix to all entities (e.g., US.Tank, PT.ColonelA). The important aspect is that the namespace is coherent with the PKI, allowing the verification of the credentials used and the integrity of the delegations created.

dRBAC as well as other RBAC solutions raised much interest in the management community, in particular when associated to military scenarios. Still, it has not been applied only to such scenarios. Other authors propose dRBAC or similar solutions for systems with distributed multiple agents [112], or even for coordinating collaboration inside virtual organizations [113]. Moreover, the work presented in this thesis also inherits some of the concepts described in dRBAC, applied to management of services and resources in communities of interest [21].

2.3 SUMMARY

The second chapter discusses several building blocks that are vital to the comprehension of the work presented. Its structure is further divided according to the knowledge domains of this thesis: networking technologies, and management principles. Several networking technologies that contributed to the possibility of community networks are described. Most of the technologies are the ones responsible for the currently observed ubiquity in communication through wireless solutions. Because of its massive use and low cost of production, wireless devices are affordable for a great amount of the population, and foster communication paradigms such as ad-hoc and mesh networks. The next knowledge domain is related to

management of systems, in particular of networked systems. In this chapter, it is taken in consideration the different management approaches, ranging from simple static management to knowledge based management, as well as some solutions of high impact to history of system management. Also, models such as the RBAC and PBM are described.

SCENARIOS & SOLUTIONS FOR COMMUNITY NETWORKS

Community concepts are present in many different solutions and initiatives. In these situations, it is not the system or the user which plays the most important role, but the aggregate of users and devices, as well as the social relations between users, their interests, and their motivations. Routing protocols can be optimized towards communal interests, with the aim of providing better service, at a reduced cost. Several existing communities took the effort of expanding their reach to the digital world, through the creation of wireless mesh networks. The purpose of these networks can be purely academic, or for the joy of doing it, but it may also aim to solve important problems effecting the community. Internet connectivity is one of the most preeminent problem that wireless communities try to solve. Aiming at future scenarios, some solutions already consider how a community oriented all wireless Internet could be made real. This chapter will tackle these aspects, and then present a set of usage-scenarios where community behavior can be used for the purpose of creating an enhanced communication network.

3.1 COMMUNITY AWARE ROUTING PROTOCOLS

The traditional way of accessing content (a web page, a video, etc. . .) resorts to accessing an Uniform Resource Identifier (URI) [114], which points to a server that provides the desired content or the location where the actual content may be obtained. An alternative approach, championed by many authors [115] considers the existence of providers and consumers (of content), and that access to content is made taking in consideration not its URI but some description of the content. To illustrate the difference between the two approaches, one can consider the case of accessing the front-page of a certain newspaper X. In the traditional approach, users type the URI of the newspaper. In the second approach they state to the

network they wish to access “newspaper X current front-page”. They could also state they wished to access yesterday’s front-page, and so forth. Built into the network there are mechanisms deployed allowing the interpretation of request, and provision of the object requested. Content Based Routing (CBR), provides routing based on content, and enables the creation of Content-Centric Networks (CCNs) where publishers inject content into the network, subscribers (users) state what they want to receive, and content is actually provided without specifically addressing end servers.

One particular aspect of this approach is that, without optimization, all content ever produced should exist in the network ready to be consumed. Brokers receive clients requests and redirect these requests to the appropriate storage server where content is actual stored. In the effort to provide low latency and high bandwidth, content should be placed near its consumers, so that the effort to provide the service is minimized. While this can be done for networks where users are not mobile, and has been done in the context of Content Distribution Networks (CDNs), mobility imposes that the optimal location for content cannot be easily guessed. The result is added overhead, or higher latency, moving data between storage servers closer to users.

As identified by Yoneki *et al.*, [116], routing content in a network is a human centric operation. Individuals cluster in groups, and have particular interests, and behavior, which will effect how content should be stored and delivered. However, this doesn’t mean that all individuals are unique, and therefore have unique behavioral and consumption traits. On the contrary, individuals cluster around communities of interest, having similar desire in the same content. Several cluster based solutions were thus proposed with the purpose of enhancing network performance or adding functionality in these conditions [117, 118]. In these solutions, clusters can be similar to communities, with the exception that no user behavior or interest are considered. Only a clustering metric and the algorithm to create the clusters is considered. Based on these observations, Zhang *et al.* proposed a routing solution [119] which addressed content based routing for highly scalable networks, while taking in consideration user interest, specifically communities of interests. According to this approach, users are organized in communities where one or more interests are shared. An overlay arranges users so that closer users share the same interests, thus forming clusters. The aim is to enhance content publishing and provisioning by further considering that communities have a *community principal*. This node is topologically close to users sharing the same interest, and provides an efficient endpoint for advertising content, and managing subscriptions. Therefore, a two layer network is created: a core distribution network composed by publishers and principals, and an access fringe composed by consumers. It should be noticed that although the task can be offloaded to devices owned by the network providers, users can be *community principals* as well.

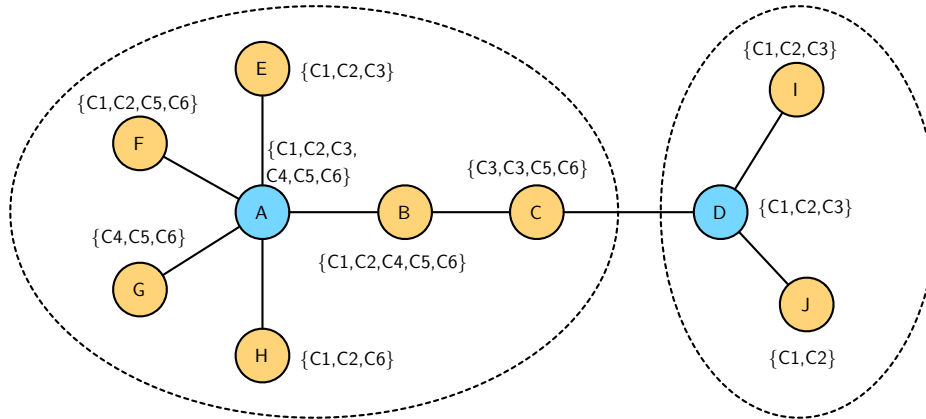


FIGURE 3.1 – The multi-overlay community structure proposed in CLONE.

Communities in Community and Location aware cONtEnt based routing (CLONE) [119] are created by a voting mechanisms which users use to express their interests. The reasoning is that users sharing same interests, will probably access similar content, and therefore will be better candidates for routing of a particular content. Users receive others votes and calculate their similarity in a metric named *Community Principal Weight*. This metric computes the interest similarity between neighbors, and is used both to decide which user is the *community principal* (the one with smallest difference), and what users compose the community. Belonging to a community is simply expressed by having a similarity lower than a predefined value. Because the purpose of this solution is routing optimization, there is no rigid mechanism for stating membership. Users can freely change community by stating different interests. Figure 3.1 depicts how two communities can be formed based on the interests each user advertises ($C1 \dots Cn$). Users *A*, *B*, *C*, and *D* have the most similar interests with their neighbors, therefore being chosen as *community principals*.

When comparing CLONE to the traditional, non structured Content Based Routing (CBR) methods, it is rapidly observed that providing context to the network greatly improves its performance. Even when considering mobile users, latency for publishing and retrieving content is lower (up to 13% of the traditional routing), and the network more easily provides the content requested (up to 90% improvement), with lower overhead (near 50%) [119]. In particular, performance increases as the access pattern becomes more homogeneous. This is explained by the fact that the more specific a community is, more efficient will be the process of bringing content to its members, even if users move from their location. Content is first brought to the community by the *community principal* which then forwards data to the destination. More users requesting the same content results in fewer queries to the service provider, and the provisioning of content directly by the *community principal*.

3.2 COMMUNITY WIRELESS NETWORKS

With the appearance of digital communication technologies in some households, during the late 1980s, and well before the Hyper Text based Internet, Bulletin Board System (BBS) provided the means for local communities to have their digital interaction. People connected to these systems with the purpose of sharing information, knowing people, or discussing relevant topics of interest. Most of these collaboration tools were not globally available. In fact, most existed around small already existing communities. One of the first known references is the Cleveland Free-Net community, founded in 1986 [120]. It included a BBS and evolved around a free medical bulletin board created at the Case Western Reserve University School of Medicine and named St. Silicon's Hospital. People interacted in these systems by dialing to well known phone numbers, but instead of using their telephone, they would use a modem connected to their Personal Computer (PC). The well known phone number was one of the numbers of the person which created or maintained the service. There were many other BBS systems created all around the world, and dedicated to many topics. When the Hyper Text Transport Protocol (HTTP) [121] based Internet appeared, connectivity solutions sprouted, bandwidth increased, and cost decreased. Hosting real time chat systems became easier and most moved to service providers, using software applications capable of hosting thousands of groups. BBS evolved to Internet Relay Chat (IRC) [122], then to forums and are currently almost extinct.

Connectivity increased since the 1980s, but it did not increase in the same way all around the world. In fact, many remote locations are still poorly covered. The same get together feeling that motivated sharing information locally through BBS, IRC and forums, also motivated a number of grass roots movements aiming at providing free Internet connectivity to local communities. Many times, these movements were supported by local government institutions as a way of reducing the digital divide [123]. Connectivity is free in the sense that, for the end user, there is no cost associated with accessing the Internet. Most of the time, the cost of using the network is only related to the network devices users buy and the electricity to power the equipment.

Networking technologies under effective mass-scale deployment, such as 3GPP Long Term Evolution (LTE), and Worldwide Interoperability for Microwave Access (WiMAX), as well as the ever expanding family of the IEEE 802.11 family of protocols clustered around the Wi-Fi initiative, are bringing convergence over the underlying communication media used. The common aspect with all these communication solutions is that they are wireless. Wireless communications, in its most various forms are without doubt the force behind ubiquitous connectivity, and driving novel, decentralized communication scenarios. The appearance of the Wi-Fi initiative, and the tremendous availability of conforming hardware further pushed

adoption of wireless technologies in local networks. Almost all laptops and cell phones currently sold have support for Wi-Fi, and the routing equipment required to cover one or more households can be found for less than 15€¹. Wireless technologies provide users with the means to reduce cost, since network Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) can be distributed over multiple participants. Also, in comparison with wired networks, wireless networks are much easier to setup, while presenting highly reduced cost. Deploying cabled infrastructures requires city permits, heavy equipment, technicians and connectivity medium (such as optical fiber). While presenting lower capacity, wireless technologies require a fraction of the cost to be deployed, and may also require no licensing.

Based on the IEEE 802.11 family of protocols, several community driven networks were created. Many of these networks sprout in an ad-hoc and unplanned manner, while others had some planning and even support by governments and cities. Independently of the success rate of these networks, they represent an effort towards community networking and several lessons can be taken. A very relevant one is that a close integration between the different layers of the communication stack is desirable, and social ties present a large influence to the growth and sustainability of the network.

The FreeNetworks initiative is a umbrella for many of the community networks existing over the globe. Initially it was focused in the United States (Seattle in particular), but now it reaches the entire world. It provides a medium for dissemination of knowledge and experiences regarding the deployment and operation of free wireless networks. A standard agreement stating the rules for creation and participation in free networks was also created, the Free Networks Peering Agreement (FNPA). It tries to establish the minimal language and behavior of free networks, so that a common understanding exists. In particular, it states that a best effort service is provided, equipment is owned by multiple individuals and entities, and no traffic shaping will occur. However, for the purpose of increasing stability and network survivability, QoS can be applied. Also, a set of services exist in the network. Both for the purpose of enabling organization and management, and for the purpose of providing higher level services. Public bulletin boards, or email servers are typical examples.

One of the most notable examples of a FreeNetwork is the Seattle Wireless network [124]. This initiative resulted in one of the oldest community networks, and also one of the most successful projects in this area. The project started not from the desire to connect a remote community, but to simply connect some friends through the Seattle city. In the year 2000, wireless network equipment was still not massively available. Tools, standardization and knowledge about operational aspects was also sometimes lacking. Therefore, the project aimed to develop everything necessary (tools, hardware), to expand connectivity. The

¹As of 2012 in Europe.

inspiration came from other existing hop-by-hop wireless communication strategies such as packet radio and the Guerrilla.net (G.net) free-net movement (now defunct). Later the concept of the project evolved with the purpose of creating a decentralized, city wide, multi-owner, mesh network for the purpose of providing wireless connectivity.

Also allied with the FreeNetworks initiative, a relevant initiative is the PersonalTelco [125] project operating at the city of Portland. The aim is to empower citizens to build a decentralized network operator, based on cheaply available hardware, and using the IEEE 802.11 family of protocols. The underlying concept was that, as Portland has a density of 2500 access points per square mile, if the users cooperated and configured their devices in order to provide connectivity to others, the city would have almost ubiquitous radio coverage. The project is far from its vision, mainly due to lack of participation from the owners of the identified 2500 access points. Nevertheless, it provides connectivity to a reasonable part of the city, and has been operational since the year 2000 (see Figure 3.2), transferring more than 20GiB of data per week between its more than 1300 users.

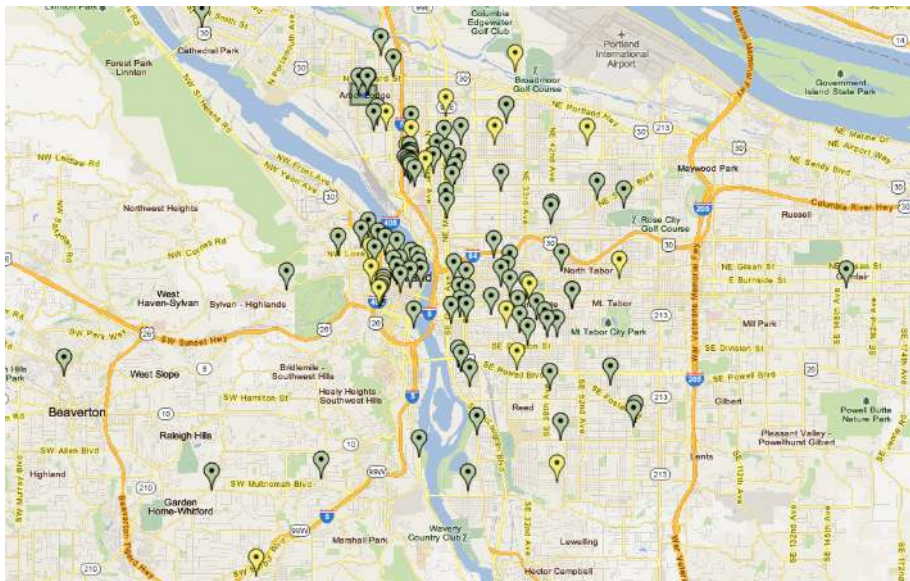


FIGURE 3.2 – Personal Telco project deployment map as in March 2012.

An idea shared by these two projects and most of the other free networks is that a set of services is required in order to provide basic connectivity and basic monitoring mechanisms. This includes DHCP and DNS servers, SNMP agents, as well as a multi-hop routing protocol (OLSR is used). When someone wishes to join the network, configuration of all hardware and software is done manually. This usually requires a reasonable technical knowledge

from users. Maintenance of these networks is also done manually by hand, configuring packet shaping rules when abuse is detected. As a result, connectivity between systems is available, although recovery of problems is limited by manual intervention in the devices. Effective QoS enforcement has been a recurrent question, specially in order to enforce priority based shaping for basic services, and prevent abuse from malicious or otherwise unaware clients. However, due to the effort required to determine the correct QoS rules, and lack of knowledge by the individuals, deployments lack wide usage of QoS mechanisms, resorting to it only in particular situations.

After the initial WMN deployments, several other sprouted in the entire world. Currently, the actual number of community based WMN deployments is unknown as they are created in an ad-hoc manner by local communities. Some, like Guifi.net [126], reached a dimension in the order of tens of thousands of nodes, and provide important hubs for the exchange of information in their regions. Others like Freifunk [127], Funkfeuer [128], and the Athens Wireless Metropolitan Network [129], provided important contributions to the further dissemination of WMN, by developing firmware images and tools to facilitate the deployment and operation of community oriented WMN. Table 3.1 summarizes some relevant deployments around the world, their dimension, the routing protocol used, and how they are managed.

A common aspect of most deployments is that management of network nodes is still done manually by device owners, and there is some lack of coordination. Moreover, especially in large deployments, the level of service achieved may greatly depend on the expertise of the owner of the current WMR, and resulting QoS policies applied.

One WMN deployment is of particular interest, Wray, England. Not due to its dimension, nor due to the technology used, but due to the scientific approach of its deployment, historic documentation, and above all, high level of motivation and engagement of the local community. This network depicts the typical example where a WMN is deployed to provide Internet connectivity in a small, remote located community, fostering development of additional interactions and services for everyone involved.

Network	Location	Start	Size ²	Type	Motivation	Routing	Management
SeattleWireless	Seattle, WA, USA	2000	80 ³	Mesh	Community	OLSR	WMR owner
Guifi.net	Iberian Peninsula	2004	>16K ⁴	Mesh	Community	OLSR, OSPF, BATMAN, BGP	Central registry plus WMR owner
AWMN	Athens, Greece	2002	>3K ⁵	Mesh	Community	OLSR, BGP	WMR owner
Freifunk	Berlin, Germany	2002	390 ⁶	Mesh	Community	OLSR, BATMAN, BABEL	WMR owner
Funkfeuer.at	Austria	2003	>1K ⁷	Mesh	Community	OLSR	WMR owner
Wireless Leiden	Leiden, Netherlands	2002	86 ⁸	Mesh	Community	OSPF	WMR owner
RoofNet	Portland, OR, USA	2007	20 ⁹	Mesh	Community	SrcRR	Centralized
PersonalTelco	Portland, OR, USA	2000	139 ¹⁰	Mesh	Community	OLSR	Central registry plus WMR owner
Wray	Wray, UK	2004	9 ¹¹	Mesh	Community	OLSR	Centralized (manual)
NYCWireless	NY City, USA	2001	142 ¹²	Hotspot	Municipal	None	Centralized
Fon	Global	2006	7M ¹³	Hotspot	Commercial	None	Centralized

TABLE 3.1 – Relevant community wireless networks deployments around the world. Unless otherwise noted, information is as reported in [131].

²The number of nodes refer to backbone nodes only (e.g., routers and access points)

³Circa 2012, <http://map.seattlewireless.net/>

⁴Circa 2012, <http://guifi.net/guifi/menu/stats/nodes>

⁵Circa 2012, <http://wind.awmn.net/>

⁶Circa 2012, <http://map.berlin.freifunk.net/>

⁷Circa 2012, <http://www.funkfeuer.at>

⁸Circa 2012, <http://www.wirelessleiden.nl/en/nodemap>

⁹Circa 2012, <http://pdos.csail.mit.edu/roofnet/doku.php>

¹⁰Circa 2012, <http://map.personaltelco.net/>

¹¹Circa 2008, as referred in [123]

¹²Circa 2012, https://auth.nycwireless.net/hotspots_map.php

¹³Circa 2012, as reported by Fon [130]

The village of Wray, with about 500 citizens, suffered since a long time from poor connectivity to the remaining world [123]. This affected all citizens as they were unable to take part on the Internet, but mostly, it affected local business. In 2003 the village decided to be proactive in the search for a connectivity solution. After a process of researching possible candidate solutions, the citizens decided to create a wireless mesh network, based on IEEE 802.11. The access part of the network uses IEEE 802.11b while the communication backhaul makes use of a dedicated 5.8GHz radio link to a building located in a strategic location. The strategic building was the local school, and because of two aspects: a) the UK government had installed Internet connectivity in most schools, b) the school is located in a high ground, providing line of sight for most households. Figure 3.3 depicts the initial deployment of Wray's community network, and the primary links established between households.

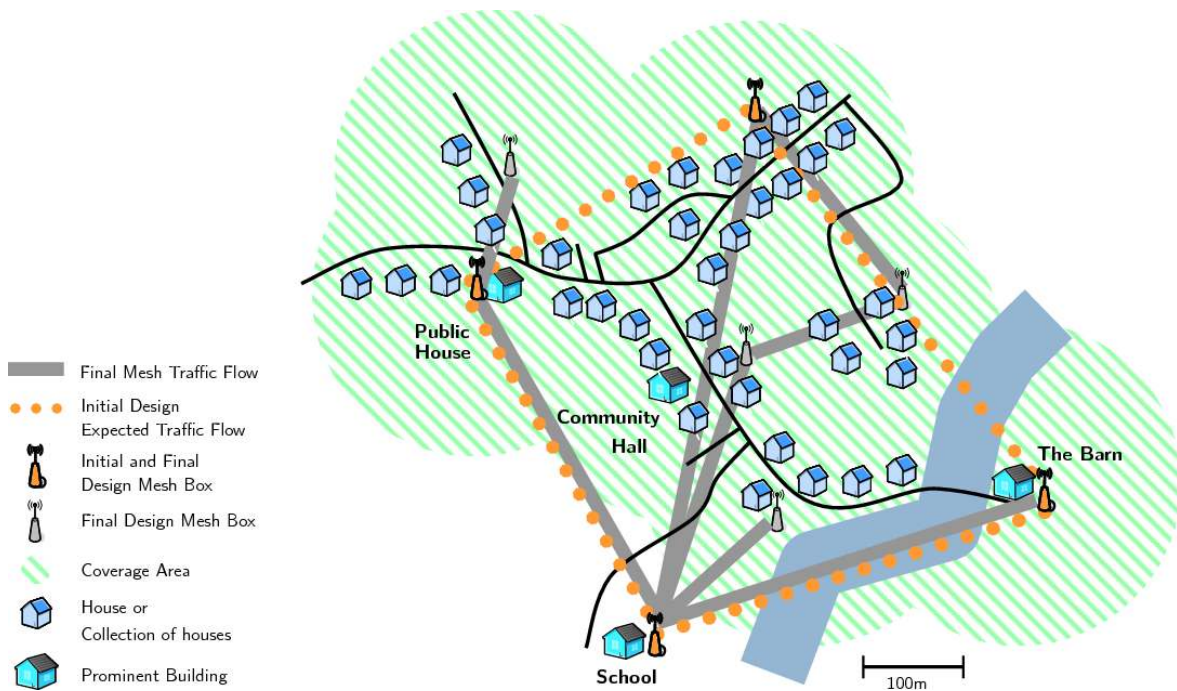


FIGURE 3.3 – Wray's community network initial and final deployment [123].

The network and the associated living lab became a success, but it clearly identified the need for more autonomic, self-* capabilities deployed into the network, in order to reduce the management burden. In practice, configuration was done through simple scripts pre-installed in devices that downloaded a simple policy from a central portal software. This portal was the central management point of the entire network, keeping the individual configurations of each system. Other WMN deployments frequently resort to this method, or simply leave each individual node to be managed by its owner. There was no self-management

capabilities in the system, and all changes were made by humans and pushed through the portal. As the authors refer, this was mostly because of both processing limitations at the devices and software availability. Routes were dynamic, as provided on demand by standard AODV. However, when proving routes between nodes, standard AODV does not take in consideration Signal to Noise Ratio (SNR), link loss rate, or other performance metric specific to wireless links (e.g., Expected Transmission Count (ETX) [132]). Also, the network was suffering from periods of instability due to heavy traffic and variations in foliage between some links. After deploying a monitoring solution, it was identified that a minority of the users were diverting the community from its initial intent by using P2P applications. The result was a high usage of network resources by these applications (a P2P application typically starts tens or hundreds of connections), with detriment of the remaining collaboration tools. QoS was deployed by means of a Leaky Bucket shaping algorithm, with the purpose of avoiding burst and keeping traffic patterns smoother. The next step has been in educating the community members to change their Internet usage behaviors in order to increase the overall performance. In parallel, the policy configured to devices started including a restriction forbidding the download of files larger than 100MiB between 9AM and 9PM.

The case of Wrays' community network is highly relevant because it shows how closely knit citizens, living in a remote location, can improve their condition by acting as a community, and how the network can sprout around existing social communities. But most importantly to this work is the fact that, after the initial deployment, the authors raised concerns about managing such kind of network without restrictions. The management approach used lacked dynamism until a steady state was found by system administrators. Some users also tried to circumvent the purpose of the community network to their own benefit, which required more complex policies and even further user education. This fact increases the need for management tools which take in consideration distributed policies in community driven scenarios. Moreover, Wrays' network shows that policies for community networking are not restricted to networking aspects, such as the configuration of the routing protocol, and which wireless channels should be used. More complex, rule based policies had to be put in place in order to shape and condition traffic and applications.

3.3 COMMUNITY MANAGEMENT SOLUTIONS

Community based networks have an inherently ad-hoc nature, where each member of the community is still an individual, and may have total ownership over some of the communications hardware deployed. While many authors identified the need for incentives [133], and the need for some coordination among participants, policies are frequently distributed

as good practices through web collaboration tools (e.g., forums, wiki pages, documents), and not directly applied to devices and services. WMN deployments such as Guifi [126] make available a single registry where users register their devices, allowing some form of enumeration and monitoring of devices, interfaces and links. Still, participation in the community is uncontrolled, and abusers are dealt with locally by the owner of the nearest router. FON [134] controls access to the network and forces users to choose one of two profiles: Linus and Bills. Linus users must own an Fon device and share their connection to others. They will gain the possibility of connecting, for free, to other access points. Bills do not share devices and must pay for access to the network. Because node density and critical mass are of primordial importance to community networks, access is promoted and, frequently there are no barriers for participation.

3.3.1 RURAL AMBIENT NETWORKS

Several solutions aimed at deploying some form of management to community wireless networks, according to the difficulties found operating the network, and its objectives. Inspired by the Ambient Networks initiative [135], Grampín *et al.* [136] propose the development of a Rural Ambient Network (RAN). The objective is to foster cooperation and digital inclusion of a specific rural community in Uruguay. A major challenge faced in RAN was choosing the best gateway so that Community Cost (CC) was kept at the minimum. By CC the authors refer to the actual cost charged to the community for each billing period. Due to the different providers available, and the non flat nature of its billing plans, load balancing traffic between gateways could actually result in higher cost to the community. Moreover, gateway selection was not addressed globally, in the same manner for the entire community, but according to routing policies for individual nodes and applications, while taking in consideration fault tolerance. The management approach followed in the development of RAN considered policies and the standard, IETF defined components of a PBM system (see Section 2.2.2). A centralized server, stores all policies and behaves as a PDP, while routers and other network devices, as well as clients are the actual PEPs.

One interesting aspect of RAN is the use of the Stream Control Transmission Protocol (SCTP) [137] for the provision of a transparent communication layer inside the RAN domain. Collaboration applications use the management and communication layer and operate unmodified, providing added value to participation in the network.

RAN makes use of the Witmate logic engine [138] and the Java Specification Request 94 (JSR-94) [139] rule description language, to implement an ECA based, routing optimization system. Code 3.1 describes an example routing policy deployed in RAN, which changes the

gateway based on the time of the day in order to minimize CC, while allowing a General Packet Radio Service (GPRS) [140] interface to be used if there is enough demand.

```

<Internet gateway> = g1 from 8 A.M. to 8 P.M.
                    g2 the rest of the time

if <traffic demand> reach threshold th1, or <g1 and g2> are down,
    activate GPRS interface

```

CODE 3.1 – High level gateway selection rules for RAN.

3.3.2 WIP: WIRELESS IP NETWORK

A much different type of initiative was the EU funded IST-WIP project [141]. It aimed at researching mechanisms for an all wireless mobile Internet. Some of the concepts were inherited from the vision presented by Negroponte *et al.* [142], by considering the existence of lily pads and frogs. In the case of WIP, cities are ponds where communities (lilies) exist and messages (frogs) flow across the different communities which interconnect individuals. WIP took a vertical approach at the problem of an all wireless architecture. Related to this thesis, it defined a set of lower level communication solutions, as well as optimizations and studies for existing solutions (e.g. IEEE 802.16) [143], routing protocols optimized for wireless mesh networks [57], incentive mechanisms which surpass the tit-for-tat approach (also named equivalent retaliation) for exchange of services and resources [144], and a set of architectural components for management of wireless communities [21, 145]. An important aspect of WIP is that interaction in wireless community networks also considered social aspects, such as the social links between individuals, or exchange of services not by devices but by individuals or communities.

Traditional incentive solutions for wireless networks considered rewarding and incentive mechanisms in a packet forward basis. In order for traffic to be delivered from source to destination, both source and destination are required to also forward others traffic [146]. Other solutions extended this model by proposing that no instead of relying on direct exchange of credits, banks could actually charge traffic sources and reward forwarding nodes using real money of a virtual currency [147]. An inherent problem with these approaches is the so called *curse of the boundary nodes* [148]. Independently of the willingness to cooperate, node location will play an important role in the effective incentives deployed. Nodes closer to the edges of the network will have no opportunity for relaying others traffic, while nodes placed closer to the center of the network will have higher chance for acting as relays. Without the

proper mechanisms for compensate heterogeneous density and arbitrary node placement, tit-for-tat incentives at the network level may actually be detrimental [148].

In WIP, interactions at the social layer were considered (e.g., training, mentoring, support), and the project identified that user motivation and sense of community are frequently more important than network communication capabilities [25]. Therefore, incentives in WIP, and the network fabric itself, take in consideration social structure and non technology driven interactions between individuals.

WIP defines communities as *a group of entities which cooperate to reach a common goal*, and identifies the existence of two types of barriers for the creation of wireless based communities: technical and social. Social barriers comprise aspects such as motivation for participation, feeling of belong to the community, and resistance to change of social habits. Therefore, communities must provide an integrated environment, where individuals feel motivated to participate and feel like belonging to something. Interaction in the WIP technologically supported communities should not be invasive to social behavior and is should actually take opportunity of it. Technological barriers comprise the need for mesh aware, enhanced medium access solutions, flexible routing and community aware naming and addressing solutions. Management of WIP communities considered each layer, and integration between them.

In the vision proposed by the WIP project, not all communities are the same, and not all communities have the same direct impact in network structure. Some are more related to connectivity aspects, while others were overlays over a transparent network such as the envisioned radio Internet. In particular, two types of communities were defined: WIP Basic Community (WBC) and WIP High Level Community (WHLC). Figure 3.4 depicts how the two types of communities are related [149].

- **WBC:** These communities are composed by entities which aim to create a wireless transport network. Management mechanisms of a WBC have direct impact over network structure, and underlying communication aspects. Members of a WBC are connected through other members of the community, and no external nodes are ever used to link the different devices. WIP points that wireless neighborhood networks are an important instantiation of a WBC due to the clone knit relations between its members and locality. Outside the WIP vision, this type of community translates to a WMN such as the ones described previously.
- **WHLC:** On top of a WBC several WHLC can be deployed as they link users sharing some higher layer interest. That is, an interest in cooperating in order to realize an application or service. WHLC are seen as overlays over WBC and can include members from multiple WBC. Outside the vision of WIP, this type of community is found in

file sharing applications, or other services communicating over distributed P2P Virtual Private Networks (VPNs) or other form of overlays over the Internet.

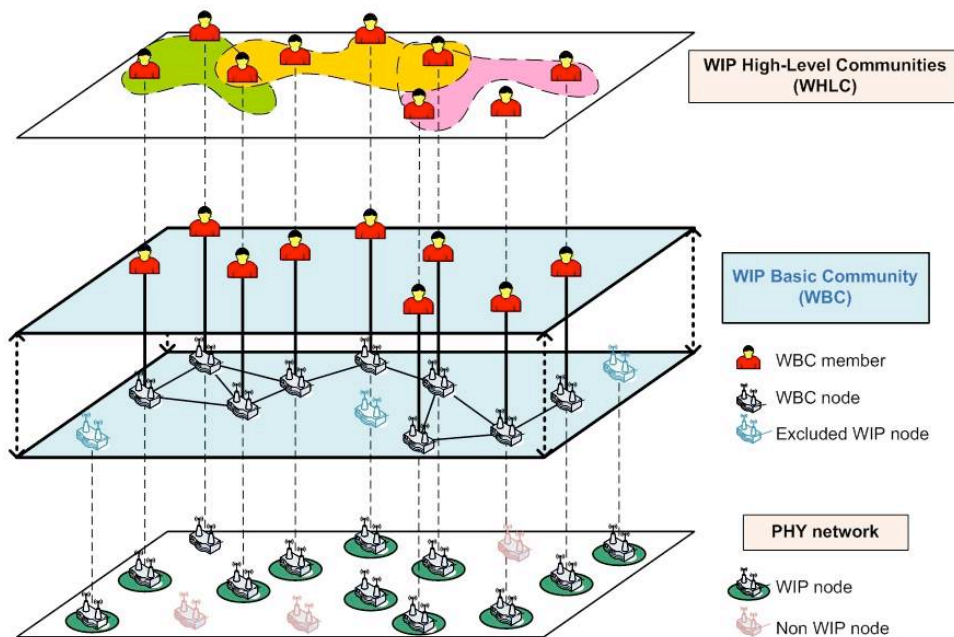


FIGURE 3.4 – Higher Layer Community and Basic Community according to WIP.

From a management perspective, due to direct involvement in the definition of the WIP architecture, the outcome of the WIP project shares some characteristics with this thesis. However, WIP had its own vision and objectives, which resulted in a particular set of management capabilities and global architecture, which in the end differ from the work presented here. The WIP high level architecture for the management of wireless communities [150] is depicted in Figure 3.5.

WIP approach for community management is based on the existence of a cross-layer communication plane binding together three planes considered vital for community aware networked devices: the Community plane, the User plane, and the Connectivity plane.

The Community plane is divided in Community Management and Community Aware-Control sub-planes [149]. The first is responsible for keeping information regarding the current communities, as well as tracking roles, rules and deciding on the final policies to apply. Also, it includes agents that communicate, forming a distributed management system which keeps community information coherent and permanently available. The last is the actual PEP, which transforms policy decisions into actual configurations, and monitors others behavior. The components at this plane also present a user interface through which users can

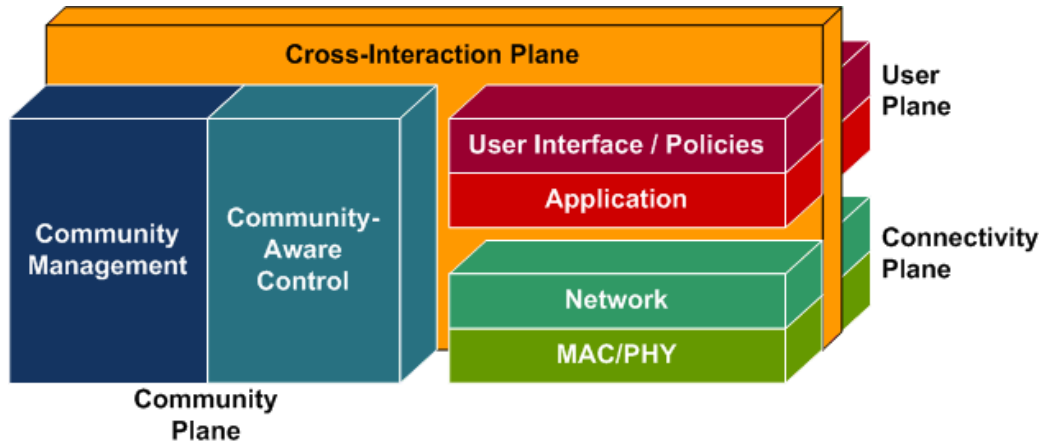


FIGURE 3.5 – WIP High Level architecture.

interact with the community management components (see Figure 3.6).

At the Connectivity plane reside the standard network services providing communication capabilities. Because WIP considers the entire communication architecture of a wireless network, and not only the aspect of wireless communities, it was also found necessary to define enhanced network services such as cooperative communications, monitoring and other low level functions enhancing communication in distributed wireless networks.

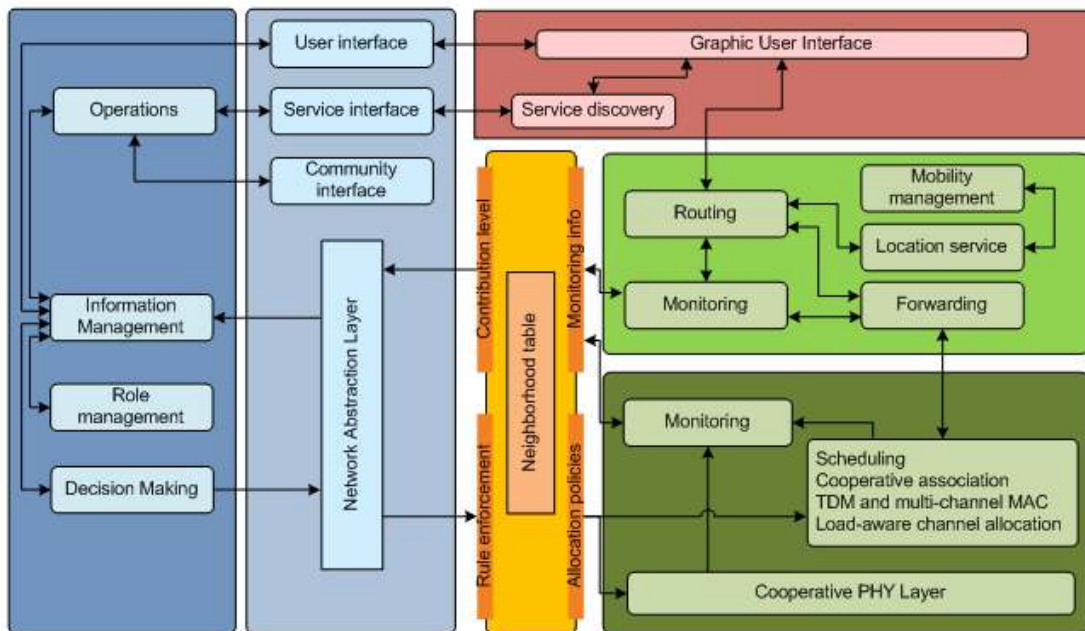


FIGURE 3.6 – WIP detailed component architecture.

The User plane provides users with the actual graphical user interface, allowing the definition of friendship associations, as well as basic exchange of messages between members of the same community. Together with this interface, a service discovery service [21] allows publishing and query of information related to communities. In particular, it allows publishing services that could be accessed by members, as well as advertisement of additional existing communities.

The cross layer interaction plane provides the glue between the different layer, in the form of a neighbor table. In the vision proposed by WIP, nodes are wireless, therefore, they are only able to enforce policies, monitor and otherwise track other nodes at radio distance. It should be noticed that in WIP, information present in the neighbor table is the actual information enforced and is personal for each user. Different users may apply different policies, in particular because WIP considers the existence of unidirectional trust relations between users, and that trust set by a local user towards another remote user, will influence how the local user behaves in relation to the remote.

```
<role id='2534-0745-4574-2364-3234-2344' name='Leader'>
  <creationDate value='17:23:23_10-03-2008' />
  <updateDate value='20:12:10_20-04-2008' />
  <creator id='5445-2466-2367-2342-3467' />
  <conditions>
    <condition id='4534-3423-268f-a2dc-3552' value='kGreaterThanOr'>
      <attribute type=int name='kUser_Reputation' value='80'>
    </condition>
  </conditions>
  <attributes>
    <attribute type=int name='kNet_Rate_Fwr_Min' value='1000' />
  </attributes>
  <rules>
    <rule id='3454-7653-2345-7553-4563' content='partial' />
    <rule id='7654-9742-7834-5745-7643' content='partial' />
  </rules>
</role>
```

CODE 3.2 – WIP policy regarding Leader role.

From a management perspective, WIP considered the existence of Roles, Rules, Conditions and Attributes, which defined rules for participation in the community. Enforcement of the resulting policy was achieved by the direct application of attributes to network services and interfaces. Code 3.2 describes the policy of the “Leader” role in WIP, and shows how conditions could be used to restrict participation to a given role based on user reputation. A serious problem with WIP is that no formal specification was ever formed, which leaves some uncertainties how the attributes created [149], which are depicted in Code 3.2 would

be applied. In particular because trust is, by definition, a personal, unidirectional relation between two users and estimating the global reputation of a user in distributed wireless systems is not trivial.

3.3.3 HOTSPOT BASED COMMUNITIES

The approach of having user collaboration for the purpose of providing network access has been successfully used by several companies, being FON [134] the most popular at this time. Fon is essentially an Wi-Fi network provider using Wi-Fi hotspots for the purpose of providing connectivity to the Internet, but following a not so traditional approach. These hotspot services are popular in public areas such as cafes, shopping centers, hotels and airports as they provide Internet connectivity to users visiting those spaces. The traditional approach followed by similar corporations is to deploy own hardware, based on a shared profit model. Fon innovated by assuming that all individuals with an Internet connection (as opposed to businesses) would be willing to share their Internet access to others. Fon initially provided free Wi-Fi APs to everyone requesting to participate in the network, which rapidly increased the size of the Fon network. Moreover, this approach greatly increased the number of Wi-Fi devices cheaply available for users to experiment with Wi-Fi, as not all devices requested were actually used for its intended purpose. Linux distributions for Wi-Fi APs rapidly added support for the Fonera equipment (name of Fon gateways) which further increased the reuse of these devices for alternative purposes.

Users wishing to provide access according to Fon rules would get free access when accessing through other hotspots of the Fon network, which rapidly provided coverage in most European cities. This type of users, according to Fon naming are the Linus. Another type of users (Bills) are the ones that share their Internet access, but will also get a percentage of the money paid to access through their Fonera. As with Linus users, Bills will also get free access across the entire Fon network. However, due to legal restrictions, this type of user is not available in all countries. Any user which doesn't have a Fonera providing Internet access is considered an Alien. Alien users share no revenue with Fon and are required to pay for Internet access through the acquisition of Fon WiFi Passes. Effectively, the types of users assumed by Fon are mapped into a RBAC model with three distinct roles. They are mutually exclusive, all roles provide rights and impose obligations. Rights are represented in the permission to access the Fon network, or even to have a share of the payment provided by Alien users. Obligations are imposed in two forms: i) the obligation for a minimum of two years providing the service when using a Fonera device; ii) Bill and Linus must keep their Fonera device operating continuously in order to have full access to the rights of their role.

Communication in the Fon network is not done between Fon devices like in a WMN, as they only provide Internet connectivity. Authorization, authentication, accounting, charging and configuration mechanisms are all centralized in Fon central systems. A map service pinpoints the location of all APs and allows users to locate the nearest point of access to the Fon network. After the network gained momentum, other corporations joined the community and began integrating Fon software in the devices used for providing triple play services. This fact greatly increased the visibility of the project, as well as the number of devices active.

From a perspective of community networking, Fon was able to create a community around the devices it provided, and the access model it marketed. Ranging from small business wanting to capitalize Internet access, to Wi-Fi enthusiasts, millions of people wanted to participate by sharing their Internet access. Support forums where users always had a pivotal role in helping other members kept the community alive and growing. Many devices were subverted to other purposes, including the creation of WMNs in local communities. One example is the Hot Mesh initiative in Detroit [151]. Making use of Fonera equipment it aimed at deploying a WMN for the purpose of providing residents with low-cost Internet access, assess mesh wireless hardware performance in real-world conditions, and to advance ongoing, citywide digital justice initiatives. As a key aspect referred by the authors some are very relevant to this thesis: i) the existence of anchor residents providing local relationships; ii) availability of technologist residents in the area; iii) existence of already established local communities for farming. Many other similar initiatives are created which in some form benefited from the Fon approach to Internet connectivity.

At the date of this thesis, Fon reports having more than 7 million devices across the globe [130], and an even greater number of users accessing their network.

3.4 USAGE SCENARIOS

Based on all these previous words, a set of usage scenarios that this thesis must handle can be derived, which benefit from a strong user commitment and adoption of a policy enabled, community oriented behavior. Traditional networks are built according to strict protocols where only some aspects are not imposed [152]. When considering that individuals may influence network operation or topology (constituting the field of user centric networks), so that the utility function of the network is aligned with user prospects, protocol operation (and even design) must be shaped to social aspects [153]. At least, protocols must be able to potentiate user interaction in a humanly natural way, because in many scenarios data flows can have strong bias towards existing links between participants.

As identified by Tasch and Brakel [154], scenarios based in cellular technologies such as Global System for Mobile communication (GSM) [155] or Universal Mobile Telecommunication System (UMTS) [156] are driven by communities and social relations. And this actually reflects what we can see in our daily life: most interactions we make are with family, friends, colleagues or with other business partners. These authors define communities as: “small groups of friends or peer-groups in a city that are mainly communicating via mobile services. Such services enrich their communication and facilitate coordination of common activities”. The community has a purpose and an expected behavior to which participants should comply.

One thing differentiating these policies from a strict, centralized, policy-based system is that the community policy evolves in a dynamic and distributed manner. The expected behavior and purpose of a community is related to a social layer and to how the majority of the individuals believe the community should be used for.

In this section several other scenarios will be presented, in which social, location driven, or even professional ties between individuals, as well as common expectations are present and can be used to increase the effectiveness of the underlying communication infrastructure.

3.4.1 SPONTANEOUS AD-HOC NETWORKS

Users participating in an ad-hoc network may seem to be lacking long-term social ties. However, during the duration of the situation that motivated the creation of the network (e.g., public event or other gathering, disaster, etc. . .), users share a common environment, have similar prospects, and share a social or professional link, or simply an interest. In many cases, and as pointed by Prigent *et al.* [157], in SOHO environments, individuals frequently possess long-term social relations. Also, because of phenomena such as the scale-free property [7] or the small-world effect [5] the probability to find social links in a small group of individuals, that came together to the same location, is significant.

During the time a spontaneous ad-hoc network exists, communal characteristics may be present and may be exploited in order to maximize network utility. That is, individuals have social or professional relations and many are motivated to keep the network operating. If relaying packets is required, individuals are motivated to participate, and with the proper incentives, the resulting ad-hoc network can be stable. A particular important aspect is that configuration is distributed to all participants, and behavior is monitored in order to identify violations. Social or professional relations, such as the ones which link members of specific groups, members of corporations, or simply members of the organization committee can be used to authorize or de-authorize participation. Moreover, social relations can further

refine the actual fabric of the network, either in terms of additional services, topology or QoS provided.

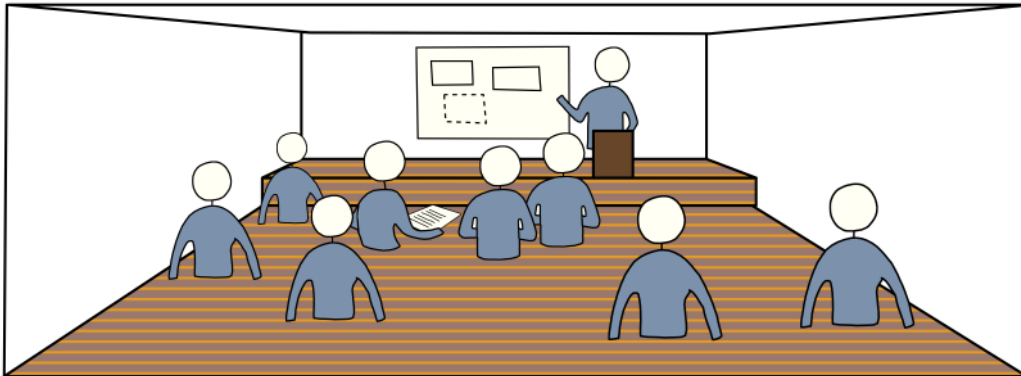


FIGURE 3.7 – A spontaneous ad-hoc network can provide connectivity at the site of an conference.

Like in WIP [141], multiple spontaneous communities can exist, and one or many can encompass the entire ad-hoc network [158]. As an example, a basic community providing network connectivity could be created by the organization committee, providing membership to all participants in the event. Many authors [158–160] made important progress towards the vision of a social aware communication fabric, and observe that by taking in consideration social structure in modeling mobility and traffic patterns, models become more realistic and closer to the flows actually created by individuals. Also, if the existing communication patterns and clusters are identified, and extracted by extrapolating the existing social ties or by considering that peers in the same cluster have high interaction history, network topology and routing mechanisms can adapt to the expected flows [158]. The aim of this effort is to improve network efficiency by creating routes more adapted to current traffic flows, while at the same time predicting future interactions, thus providing better service for future data exchanges.

Distributed policies, with the support for delegation based participation, fit the scenario of a spontaneous ad-hoc network in an event because they fulfill three basic issues of such networks: distributed access control, distributed configuration, and categorization of individuals according to their interests, or affiliation.

It can be considered the existence of 4 essential roles for the scenario of ad-hoc, spontaneous communities (others may exist):

- **Guest:** Individuals which are visiting the location where the community operates but which will leave it after some time. Therefore, the attribution of this role should be time

limited. Guests are not considered as effective members of the community.

- **Member:** Individuals which belong to the community and may access its resources. This role should be incompatible with the *Guest* role, and enrollment can also be time limited. Enrollment can be further restricted according to the requirements for membership of the community. However, because spontaneous communities can frequently be focused in sharing content and providing connectivity, enrollment is expected to be frequently free.
- **Friend:** Members can express their social ties into the community by means of the *Friend* role. While it is not expected that this role can provide better service at a community scale, because spontaneous communities are typically small in size, it can be used to differentiate communications between individual members. Also, applications can take in consideration this links and implement services such as instant messaging or ad-hoc collaboration tools (e.g., Apple Bonjour [161]).
- **Leader:** Leader members are responsible for the management of the community policy to the remaining community members. If using consensus mechanisms such as Paxos [162], individuals enrolled in this role can be considered as Proposers.

3.4.2 ISOLATED LOCATIONS

Many areas around the globe have a very low inhabitant density (e.g., the village of Wray, England), making them unattractive to network operators. The low potential for revenue obtained, combined with the high costs of interconnecting these locations to the operator core network, lowers the interest operators have in extending their access networks to these sites. But even if they select alternative solutions based in wireless technologies, the lack of line of sight due to terrain, long distance, or high tree density may present a real challenge for service provisioning. This is specially relevant if the operator aims at interconnecting all or most households and not just a single point of presence.

The difficulties in providing Internet access to remote locations have been identified since a long time. Based on the knowledge gained by ad-hoc networks, wireless ad-hoc and mesh networks appeared as potential candidates providing connectivity to rural, isolated or otherwise excluded areas [47]. Several service providers already provide network connectivity by relying on such approach [163].

Due to the low capacity or coverage provided, or due to the cost associated to the connectivity service, a common approach has been the creation of self-organized networks by individuals themselves. The low cost of SOHO communication devices much contributed to creation of such initiatives. In particular because the cost of acquisition (laptops, wireless routers and antennas), and the costs of operating the network (mostly just electrical power

consumed by low power devices), can be distributed over all individuals using the service, and paid individually on a pay-as-you-go basis. If a household wishes to join, it just has to acquire the required device and follow the procedures to connect to the network. This scenario is slightly different from broader WMN deployments because in this case, households are not placed in a dense urban environment, but on a highly isolated rural location. Scenarios like this are rarely found in Europe (due to its higher population density [164]), but are specially frequent in other continents such as Africa and Oceania.

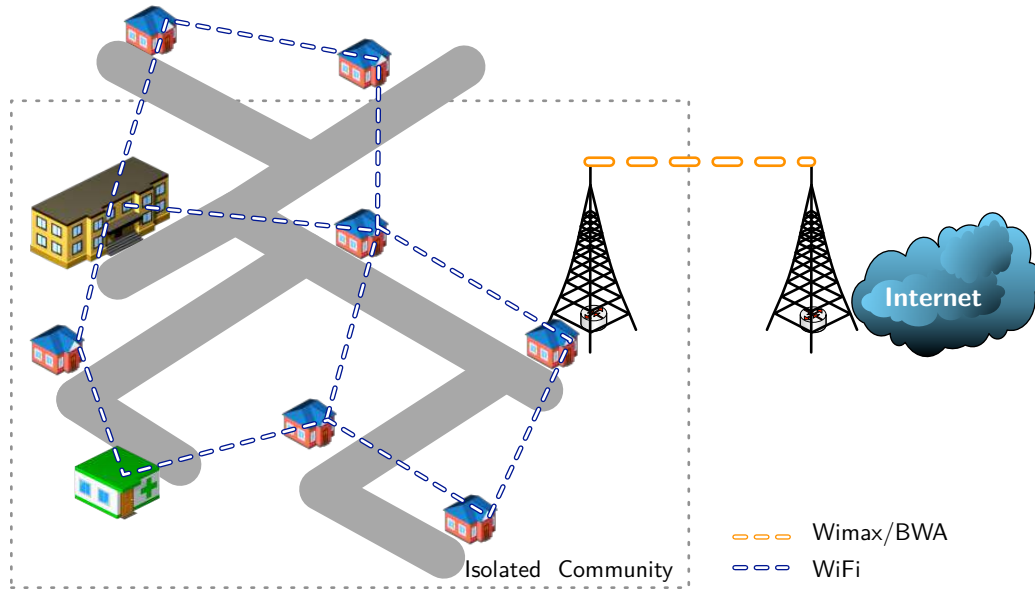


FIGURE 3.8 – A community providing services and connectivity at a remote location.

In the case of neighboring individuals in isolated areas, the community (at the social level) is already present due to the existing social ties, similar environment and common living prospects. Trust relations are already deployed as the result of the past interactions between individuals. The environment is shared due to the remote location and isolation of these places. Prospects are also similar because all individuals are in a disadvantageous situation regarding the same aspects: Internet connectivity and collaboration infrastructures.

All these aspects can be mapped to the policy governing the network so that protocol operation, and network structure, is close to the social structure already present. Existing trust relations may be used to grant membership to new individuals or to grant access to higher roles or particular services. Also, coordination can be improved with benefit for all participants. One example is available in the solution proposed in [165], which considers neighboring homes connected to a wireless network, where only some homes (a minority)

have connectivity to the Internet. The cost of routing traffic through a particular gateway can be propagated in real time to the entire community, and routes can be chosen according to the lowest cost. This is particularly relevant because different access technologies, as well as operator contracts, may present very different cost figures for the same traffic at the same time of the day. Moreover, some contracts specify that cost changes with the time of the day, others that it changes with the volume transferred. A system based on static assignment results in higher cost than a dynamic system in which connectivity is shared, and the gateway is chosen dynamically to the benefit of the community (lower running cost).

After basic connectivity is provided, services can be announced to members of the community, and sub-communities can be found if the proper service discovery infrastructure is deployed [21]. Examples of services provided over the community network are boards, support forums, personal pages and even Voice over IP (VoIP).

It can be considered the existence of 7 essential roles for the scenario of interconnected neighborhoods (others may exist):

- **Guest:** Individuals which are visiting the location where the community operates but which will leave it after some time. Therefore, the attribution of this role should be time limited. Guests are not considered as effective members of the community. Considering the example present in Fon, these users correspond to the Alien role.
- **Member:** Individuals which belong to the community and may access its resources. This role should be incompatible with the *Guest* role, but enrollment can also be time limited. Enrollment can be further restricted according to the requirements for membership.
- **Friend:** Community members can express their social ties into the community by means of the *Friend* role. While it is not expected that this role can provide better service at a community scale, it can be used to differentiate communications between individual members. Also, applications can take in consideration this links and implement services such as instant messaging or ad-hoc collaboration tools (e.g., Apple Bonjour [161]).
- **FamilyMember:** Community members can express their family ties into the community by means of the *FamilyMember* role. While it is not expected that this role can provide better service at a community scale, it can be used to differentiate communications between individual members. As an example, it can be considered that some resources like photos are only available to family members.
- **HardwareProvider:** Community members that provide connectivity equipment in the form of access points, routers or servers. These members may enroll this role as a form of differentiating service access, but enrollment should be limited to users in the *Member* role which actually provide equipment to the community infrastructure.

- **ServiceProvider:** Community members that provide services to the community. These members may enroll this role as a form of differentiating service access, but enrollment should be limited to users in the *Member* role, which actually provide some service to the community. It should be noticed, that it is possible to consider that the service provided relates to exchange of goods, expertise or other non-digital form.
- **ConnectivityProvider:** Members of the *HardwareProvider* role which also provide connectivity to the Internet. These members contribute to the community by providing equipment, supporting the connectivity costs, and being the legal contact point responsible for traffic produced through their devices.
- **Leader:** Leader members are responsible for the management of the community policy to the remaining community members. If using consensus mechanisms such as Paxos, individuals enrolled in this role can be considered as Proposers.

3.4.3 URBAN NEIGHBORHOODS

Urban neighborhoods in developed countries have a large density of IEEE 802.11 access points. As presented in [28], the city of Aveiro was scanned for IEEE 802.11 access points using a methodology that both identified the access point and its location. Analysis considered each neighborhood independently as they are clearly isolated. Results show that the average access point density was 445 devices per square kilometer, with some neighborhoods having up to 775 access points per square kilometer. When these results are correlated with inhabitant density, it was found that the number of inhabitants per device ranges from 3 to 5. It can be easily concluded that most families in the Aveiro scenario make use of IEEE 802.11 enabled devices to communicate. But this density is not specific to Aveiro as other studies [166] provide similar results for other cities.

Initiatives like Fon, aim to exploit the density of wireless devices in urban environments, by promoting that users can enable their devices to provide Internet access. Each user is empowered to provide connectivity by turning their devices into Wi-Fi hotspots, and in retribution will have access across the Fon network of hotspots. While most interaction between members of the Fon community is done at a local scale (friends, neighbors, colleagues), the community has managed to reach a respectable world size.

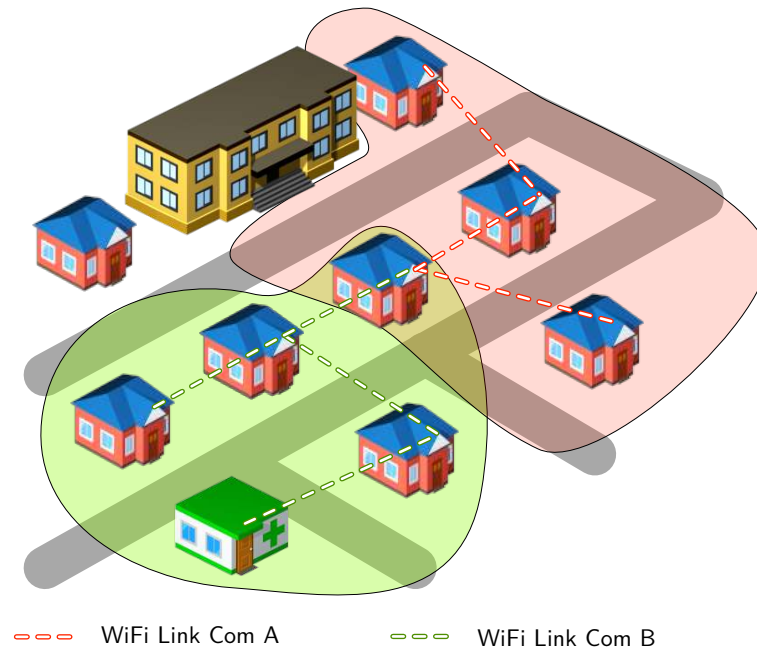


FIGURE 3.9 – Two community composed by neighborhoods at urban locations.

A different approach was followed by the several FreeNetwork initiatives, and municipal WMN deployments. In these initiatives, many operating in densely populated urban areas, the purpose is to provide wireless connectivity outside each users home, while at the same time making a more efficient usage of each Internet access. Without cooperation of individuals, that are connected to WMN deployments, users can roam freely inside their home, but not anywhere in the city. When they go to parks and restaurant, commercially available hotspots, or cellular technologies must be used. The community empowers individuals to create their own network provider. For admission in the community they donate equipment. In return they have a ubiquitous communication in their homes, and outside, across the city. The same mechanism allows individuals to share their Internet connection, thus distributing the price payed every month across a larger base. This aspect is much more relevant today as it is frequent for wired connections to provide between 24Mbits/s (Asymmetric Digital Subscriber Line (ADSL) [167]) and 100Mbits/s (Fiber To The Home (FTTH)). Figure 3.9 depicts a community network over a dense urban environment, where both Internet access is provided to users outside their homes, and where several homes share the same connection to the Internet.

Neighborhood based community networks are composed by individuals which may interact directly, without electronic devices. These individuals are neighbors and it is expected

that they have some knowledge of each other. Roaming across a larger area, or reducing connectivity cost is what drives the clustering of individuals around the purpose of providing Internet connectivity.

The existence of a common policy is vital for such scenarios. It is not expected that individuals can keep the same usage profile over a wireless or wired connection. Sharing an wired connection over a WMN implies making some sacrifices. In particular the use of high demanding P2P applications, and frequent download of large files should be discouraged. Otherwise, the service provided by the WMN will be much reduced. It is also frequent to apply quotas for users, which require some form of membership and identification mechanisms deployed in order to associate IPv4 addresses to actual individuals.

Cost is another factor which should be taken in consideration, and where the community policy can play a role. If in isolated areas, the community members get together to share a single attachment point to the Internet. In urban scenarios, there may be multiple users with devices providing Internet connectivity. From the study presented in [28], a small city had more than 5000 individual access points, most providing Internet connectivity. If a cost agnostic approach is followed, packets are routed to and from the outside from the closest gateway available. Performance wise, this is one of the most efficient ways of distributing connectivity. An improvement of this approach can consider the actual load of the gateways, and reroute some of the flows to gateways with lower load. Routing will be suboptimal (as it leads to routes with higher than optimal number of hops), and local traffic may experience lower quality of service, but in specific scenarios where most traffic is to or from outside networks, total performance may be improved. This is the reason why protocols such as OLSR provide mechanisms for “intelligent” gateway selection.

The problem which is exacerbated in WMN over urban scenarios is that connectivity prices are not homogeneous for all gateways. In fact, the opposite is expected¹⁴. If the charging plan is not considered, individuals with devices located in more popular areas such as parks, may face increased charges if they allow others to use their connection.

A common policy, and the existence of a framework managing systems and user interaction may be used to provide more flexible network operation. One of the aspects is the construction and dissemination of policies enabling cost based gateway selection. With this, it becomes possible for each community to define the policy of gateway selection, and take in consideration cross layer aspects of the communication. The actual cost charged to the gateway owner is one of the parameters.

In neighborhood based communities, the need for differentiated roles for users may be

¹⁴As of 2012, the Portuguese regulatory body registers 83 different charging plans available to end users, across 6 different providers, and using 5 different communication technologies.

unnecessary if all members have similar resources and behavior. A Member role can be used to state membership, restrict participation and configure devices. Participation may be restricted by if the community is actually private to a set of users, or should be limited to a building, or a neighborhood.

In most cases, the existence of differentiated roles will enable better management over community resources and overall evolution of the community. In particular it can be considered that some individuals will provide hardware for the construction of the network, while others will simply use the network for access the Internet or services provided. Differentiated roles can be applied to these users, granting higher differentiated benefits regarding usage of network resources or the consensus processes. Some of these individuals may also own equipments granting access to the Internet (gateways), because of the cost associated with the maintenance of such service, differentiated roles can also exist for them.

It can be considered the existence of 7 essential roles for the scenario of interconnected neighborhoods (others may exist):

- **Guest:** Individuals which are visiting the location where the community operates but which will leave it after some time. Therefore, the attribution of this role should be time limited. Guests are not considered as members of the community. Considering the example present in Fon, these users correspond to the Alien role.
- **Member:** Individuals which belong to the community and may access its resources. This role should be incompatible with the *Guest* role, but enrollment can also be time limited. Enrollment can be further restricted according to the requirements for membership.
- **Friend:** Community members can express their social ties into the community by means of the *Friend* role. While it is not expected that this role can provide better service at a community scale, it can be used to differentiate communications between individual members. Also, applications can take in consideration this links and implement services such as instant messaging or ad-hoc collaboration tools (e.g., Apple Bonjour [161]).
- **FamilyMember:** Members can express their family ties into the community by means of the *FamilyMember* role. While it is not expected that this role can provide better service at a community scale, it can be used to differentiate communications between individual members. As an example, it can be considered that some resources or services are only available to family members.
- **HardwareProvider:** Community members that provide connectivity equipment in the form of access points, routers or servers. These members may enroll this role as a form of differentiating service access, but enrollment should be limited to members which actually provide equipment to the community infrastructure.

- **ConnectivityProvider:** Community members of the *HardwareProvider* role which also provide connectivity to the Internet. These members contribute to the community by providing equipment, supporting the connectivity costs, and being the legal contact point responsible for traffic produced through their devices.
- **LocalLeader:** For larger communities, some Members may have responsibility for the maintenance of a coherent policy across all users of a subset of the community. This subset may be related to their location, or other clustering parameter present in the community.
- **Leader:** Leader members are responsible for the management of the community policy to the remaining community members. If using consensus mechanisms such as Paxos, individuals enrolled in this role can be considered as Proposers.

3.4.4 INTEREST DRIVEN OVERLAY NETWORKS

Another representative scenario where the community semantic is relevant to communicating individuals is the one related to communities of interest: Individuals that share some interest without any restriction on location, and are able to communicate over existing networks, such as existing WMN or even the Internet. Individuals can be located all across the globe and use the Internet as the transport medium, or can be located in the same area such as a university campus. A group of friends, a subgroup of coworkers, or even family members can be examples of such communities. The individuals composing the communities share social or professional ties and may have interest in creating a community for sharing information, increase coordination or simply create an isolated group for gaming.

This scenario differs from the previous scenarios because connectivity is now provided by a transparent medium: the Internet or other network¹⁵. Devices do not create multi-hop self-organized networks, like in the previous cases, and there is no need support management of lower layer protocols, mainly because devices cannot extend their control to the Internet (the network is a pipe, owned by network providers). Local multi-hop clusters can exist where this makes sense, however the important aspect is that most interaction occurs over the network medium, using devices directly attached to an access network, and through technologies such as ADSL or UMTS.

Terminal devices cannot impose control to the community fabric over the Internet, but they can shape and authorize the services they host, and provide services exclusively for the members of the community (e.g., for enhancing collaboration inside an existing community

¹⁵The Internet or the network in a campus is transparent in the sense that a large number of individuals connect to it and communicate without (or with little) restrictions, just like they would communicate orally.

of interest), or provide aggregated services to the remaining world in the form of personal clouds [30]. If required, the resulting overlay can also be used to route traffic between hosts, like in an ad-hoc network. In particular if a service is expected to transfer data under some privacy constraints, and the application does not support privacy mechanisms. An overlay network, or Virtual Network can be used to encapsulate service data packets, and even provide routing over the overlay, similar to the way current privacy aware, file transfer oriented, Peer-to-Peer applications or distributed private networks operate [168].

Communities provide a common context where a common policy can be used and be dynamically refined by its members. As the context and policies are shared among the participants, the community will present a ubiquitous environment where members can interact. Like in the previous scenarios, ties existing at the social layer can be translated to membership and access to particular services or resources, and services can be provided to members according to the active policy and existing roles. Taking as example the case of how services and currently provided, user centric communities over the Internet can be exploited as user centric clouds. A community of users abides to a common set of policy rules and provides a service to their members, or to the outside world, in a transparent manner. For consumers external to the community, members behave like an all and coordinate efforts in order to provide a location independent service. Much like existing Internet grids where users can choose to donate their resources to fight cancer or search for extra-terrestrial life, communities of interest can provide clouds for different purposes. The important fact here is that these new clouds are driven by user requirements and specific interest (which can also be monetary), in opposition to business strategies, and the simple case of providing Internet connectivity.

The incorporation of community self-organization mechanisms as paths to build user centric clouds should result in the need to create primitives allowing users to create, announce, discover, join, leave or destroy communities, as well as participating in a distributed environment composed of several hosts at remote locations. In every infrastructure one or more clouds could be provided, each exporting available resources as services either to external clients or members, according to the community policy. As an example promoting user information privacy, each community can define shared keys for controlling data storage, or system access. These keys, which are secured and only have a local reach, can be used by services to cipher data inside the user cloud. Backups and other data maintenance operations are not affected and can still be performed — because data is secure as they operate over bulk ciphered bytes as well as native data (if we exclude higher entropy which usually results from ciphering data). By using a different key for each service, access to others' data and privacy violations would be more challenging, but more importantly, users could permanently invalidate data (including replicas and backups) simply by invalidating the respective key.

As information and keys are tied to a specific community, destroying a community would result in effective invalidation of all data. Tied with the notion of membership entitlement by delegation, revoking a single user which demonstrates to be a threat to the community will effectively block him from accessing data, as well as all other users relying on his delegation.

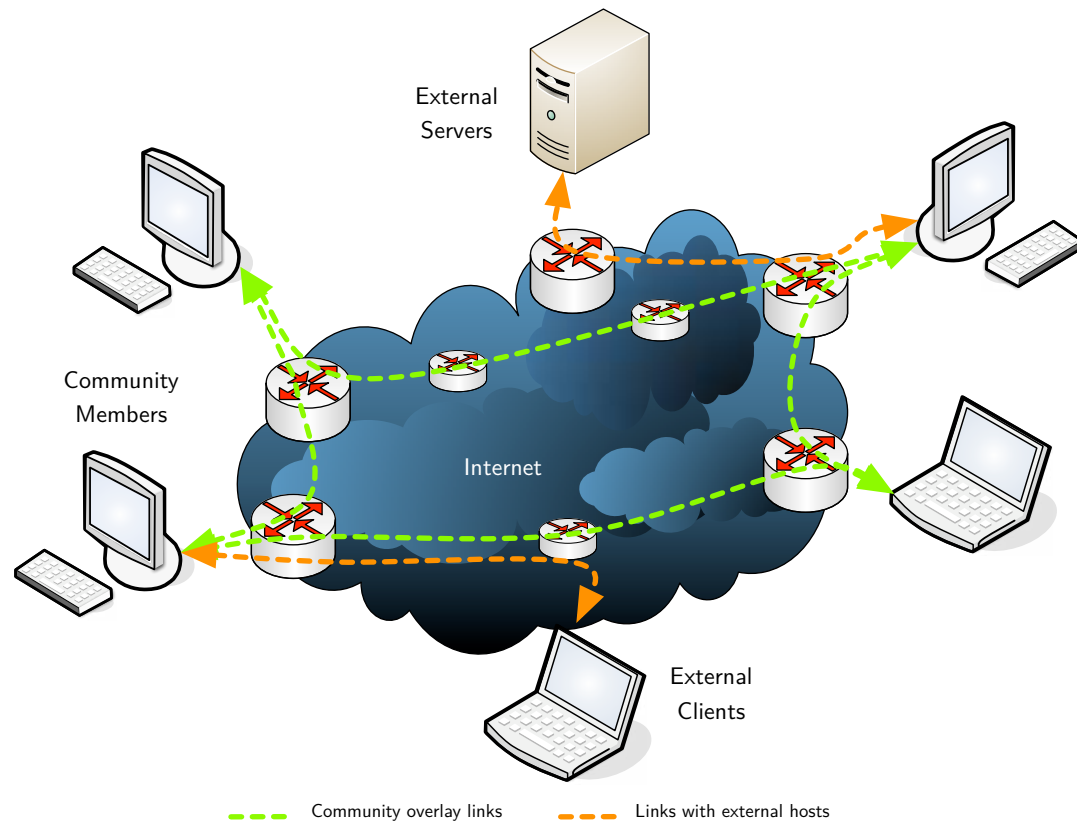


FIGURE 3.10 – A community of interest providing services to external entities.

User interest communities may be more prone for providing a myriad of additional services, mainly due to the focus of its members in more location independent aspects. Also, the social links between users may be of less personal nature than in location based communities. Management of such communities requires stronger identity management processes, resulting in a system that, while being community driven, has a strong notion of user identity. However, not only user identity is important, but also community identity. Considering scenarios where communities provide services to external entities, these communities must be clearly identified so that federation and delegation of services between communities can be possible. As depicted in Figure 3.10, the result is that identity community management mechanisms provide globally available identity management services. In alternative, an existing PKI, together with identity management solutions (e.g., OpenID single sign-on

service (OpenID) [169]), or simply the existing DNS infrastructure may be used to assert membership and identity services between communities and users. The same infrastructure should also be linked with mechanisms providing secure communication tunnels over the (untrusted) communication medium, following methods such as onion routing [170].

Another requirement for effective community based management support is the availability of interfaces with a relevant set of services to externally used solutions such as web, file, database and email services, similar to typical Platform-as-a-Service (PaaS) or Software-as-a-Service (SaaS) approaches. These interfaces are required in order to enable access control and manageability support, both for the services being provided and for the services being used internally. For this purpose, integration of the management system with linux Pluggable Authentication Modules (PAM) for service authorization and authentication, and OpenID for authentication of web oriented services, may be envisioned.

It can be considered the existence of 7 essential roles for the scenario of overlay communities (others may exist):

- **Client:** Individuals which access resources exported by the community can enroll the *Client* role. If required, several other sub roles can be created for the purpose of better applying rights and obligations, as well as to better provide access control (e.g., *StorageClient*)
- **Member:** Individuals which belong to the community and may access its resources. Enrollment can be further restricted according to the requirements for membership. It can be considered that granting the *Member* role immediately provides access to all resources. However, it can be considered that the *Member* role provides no increase in service access permissions, being this the responsibility of the *Client* role.
- **Friend:** Community members can express their social ties into the community by means of the *Friend* role. While it is not expected that this role can provide better service at a community scale, it can be used to differentiate communications between individuals. Also, applications can take in consideration this links and implement services such as instant messaging or ad-hoc collaboration tools (e.g., Apple Bonjour [161]).
- **HardwareProvider:** Community members that provide connectivity equipment in the form of access points, routers or servers. These members may enroll this role as a form of differentiating service access, but enrollment should be limited to community members that actually provide equipment to the community infrastructure.
- **ServiceProvider:** Community members that provide connectivity equipment in the form of access points, routers or servers. These members may enroll this role as a form of differentiating service access. Several other roles may be appropriate according to the service provided (e.g., *StorageProvider*, *LoadBalanceProvider*, etc...)

- **LocalLeader:** For larger communities, some community members may have responsibility for the maintenance of a coherent policy across all users of a subset of the community. This subset may be related to their location, or other clustering parameter present in the community.
- **Leader:** Leader members are responsible for the management of the community policy to the remaining community members. If using consensus mechanisms such as Paxos, individuals enrolled in this role can be considered as Proposers.

Taking in consideration the roles proposed and the scenarios identified, a summary is presented in Table 3.2. As depicted, the *Member* and *Leader* roles are common to all scenarios. This happens because they are required for basic membership and management functionalities. The *Friend* role is also common as it is used to represent social ties between individuals. The remaining roles have higher specificity for the scenarios described.

TABLE 3.2 – Summary of the roles proposed for each scenario.

Role	Spontaneous Networks	Isolated Locations	Urban Neighbourhoods	Interest Networks
Guest	-	✓	✓	✓
Client	-	-	-	✓
Member	✓	✓	✓	✓
Friend	✓	✓	✓	✓
Family Member	-	✓	✓	
Hardware Provider	-	✓	✓	✓
Service Provider	-	✓	-	✓
Connectivity Provider	-	✓	✓	
Local Leader	-	-	✓	✓
Leader	✓	✓	✓	✓

3.5 SUMMARY

This chapter presents an analysis of relevant initiatives and communication solutions. These initiatives and solutions were created in order to take in consideration social structure, or were created due the desire to exploit an existing social structure and shared interest, with the purpose of optimizing network operation, or simply for providing network connectivity. In the set of solutions which take in consideration social structure, it is discussed the example of content delivery networks. They show that by adapting the operation of communication protocols to the behavior of current users, the interests they present, or the social links they have, network performance can be either simpler or present higher levels. Several initiatives

appeared from an opposite perspective, and with different objectives. Wireless networks created around neighborhoods and with many different motivations. Some were created in order to include individuals which were in some way excluded from the Internet due to their remote location, or due to monetary constraints. Others connected individuals simply because they had the desire to create a user owned communication infrastructure, free from traditional network providers. Also presented in this chapter are several projects, and their solutions aiming at the challenge of managing community wireless networks, or which aim at closing the gap between system management and social structure.

MANAGEMENT OF STACKABLE COMMUNITIES

At the social level, communities are created by a set of users which share similar interests. The code of conduct of the community, its doctrine, is in some way shared by most members, even if never actually instantiated in formal set of rules. When applying social interests, and community structure to devices, systems and protocols, there must be some representation of the community structure. This information can be used to optimize system operation towards the goals of the community members. How community structure is represented, and how it is integrated with the existing Internet infrastructure are some of the topics that must be addressed. These topics are addressed in this chapter, together with a formal definition of a policy language and management framework architecture, capable of managing communities of interest.

4.1 INTEGRATED COMMUNITY MANAGEMENT

As discussed previously, several movements appeared around the notion of wireless connectivity, some even with the purpose of deploying networks competing with existing commercial broadband solutions. These solutions offer connectivity to either a city area, or a single neighborhood and appeared from the self interest of the citizens. The result was the deployment of WMN where communication links between systems were related to the existence of social links, or at least a shared interest between the owners of the systems. Some WMN can be seen as a intermediate step between totally ad-hoc networks, and user based networking, with the distinction that the network is not focused on the self, but on the community and on the existing social capital. Users have the characteristic of fearing the

unknown and trusting what they know. Community aware WMNs smooth collaboration by relying on the existing social relations between users.

In the scenario of community networks, a user is an entity which exists at many different layers. The most relevant to this work are the physical world where they interact directly, and the digital world where they interact through electronic systems and systems. When interacting directly, users may provide services to each other in several forms. Counseling or technical support, or simply helping performing a task that required more effort are examples of services which individuals provide to their communities. Through electronic systems, users may also provide services to each other, and otherwise cooperate by participating in forums, sharing files, or exchanging information through the many available web based social networking applications. Even the World Wide Web (WWW), which was in the past dominated by static content, is now an extension of individuals, allowing rich real time interaction. Figure 4.1 depicts how individuals now interact by making use of their devices, and the connectivity they provide.

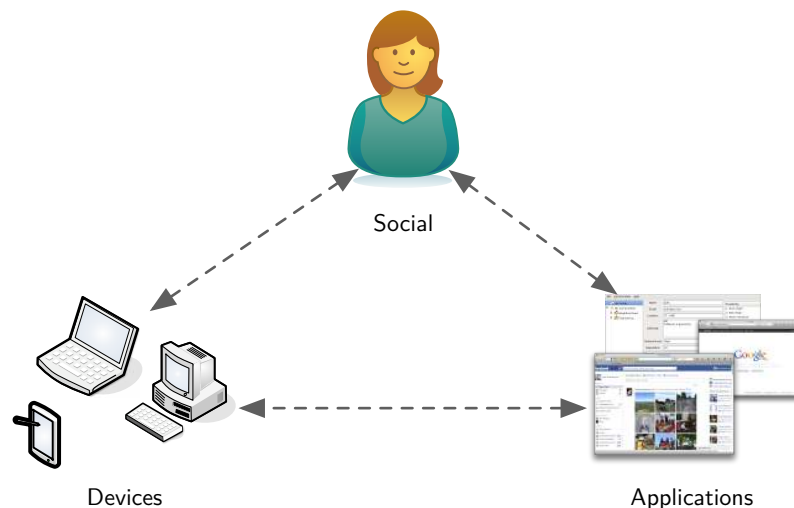


FIGURE 4.1 – The different expressions through which User interacts.

All interactions between individuals strengthen links and increase social capital. The result is enhanced resilience of user driven communities by the increased the cohesion of its members. From a technological point of view, network resilience using wireless technologies and mobile systems considers how easily network operation can cope with aspects such as battery depletion of some of its systems, routing inefficiency, radio interference, or foreign attacks targeting vital mechanisms [171]. However, besides all technical aspects, cohesion of its members, and the ability of the community to resist social changes, maintaining an adequate behavior, is vital. In these networks, communication density and link redundancy

is important, as churning has a high impact in the quality of the service provided. With networks composed by users, to users, link redundancy also means redundancy at the social level, which if existing, will result in a even more robust community.

Community networks require the psychological involvement of users and the desire to contribute to the aggregate, instead of (exclusively) to the self. Incentives based on packet exchanges (e.g., tit-for-tat) are the typical mechanism by which collaboration is fostered in multi-hop networks, and game theory plays an important role in shaping the correct incentives. Using this approach, users keep track of usage by other users, and only provide service if the remote user also has helped him or, alternatively, proves that it helped other user. While this method shows some success in promoting fair behavior, it is usually focused in simple aspects such as packet forwarding and is unable to cope with cross layer service exchanges. Also, lack of proof that users provided a given service does not mean the user is misbehaving. Its physical location or communication conditions may simply make it unsuitable for providing a service, which will degrade or simply block access, even if he intends to participate. This is true for nodes which do not have suitable equipment, or are located in such a location that optimal routes between other nodes do not use them. Users participate in communities because they want to fulfill some purpose, and they share some interest. Therefore, while incentives are relevant [172], the amount of motivation of members and the focus in the social links allows more complex, cross layer incentives, presenting increased effectiveness.

Autonomic systems under the governance of a common policy mostly resemble communities. They can provide aggregated services as a collective to either internal or external entities. The services provided can take the form of Internet connectivity, file sharing, collaboration forums, or any other service typically provided over a communication network. Similarly, the way communities operate has its roots in the fulfilment of needs (both of members and of external clients), the existence of some altruism (or high motivation for sharing), supported by the existence of a common policy that is available to all members (a view of this policy may be provided to clients) [173, 174]. That is, for a community to be lasting, it should be supported by transparency in processes and information, and a clear identification of its members and their roles should exist in a shared knowledge base. This knowledge base can be used to identify expected behavior patterns, as well as sub-groups inside the community, and explored as in autonomic systems. Community management is different from traditional system management, because users have more control over the decisions and the knowledge evolves as the interactions between users occur (which may affect their reputation). Moreover, the policy is dynamic and distributed by nature (both in terms of structure and operation), containing elements which vary with social structure.

Management of communities, when applied to networking scenarios, or when communities are used to foster interactions over the network medium, requires the existence of a set of mechanisms which address the above mentioned aspects. At its core there is a policy stating the purpose of the community. This policy is frequently named a doctrine, or code of conduct, because users not only accept it, but believe in it and choose to follow it. Membership can be verified by consulting repositories storing information about the members, and the structure of the links between users, but that is not required. Many communities, such as the ones formed in an ad-hoc manner, are created considering the relations between individuals. A web of trust is built by acquaintance and the policy can reflect this structure. Individuals can have multiple acquaintances in the community, which work as anchors for the purpose of stating membership. The structure of such network resembles a scale free network as studied in [4]. The closed interaction environment will however create a distinction between the number of links between members of the same community, and the number of links of members of different communities. Effectively, the resulting connectivity graph is the typical community graph (recall Figure 1.2).

The policies in a community should state the guidelines for the correct operation of all member entities. In the scope of a networked community, there should be a strong commitment towards easy configuration and management of networking elements, and services. If the community exists by means of an overlay over the Internet (e.g., a private cloud [30]), the focus should be put towards services and support systems, as well as support for identity management mechanisms.

The fact that communities have well known policies and guidelines does not imply that all users are expected to behave the same way. In fact, it is common for communities to consider multiple clusters of users. These may have different rights and different obligations, taking in consideration their purpose on the community, resources, and individual interest. From the perspective of management, roles are the mechanism which can be used to provide this differentiation. Membership can also be considered a role which members delegate to known (friend) newcomers. Other roles may exist for differentiating systems (routers, laptops) or users (leaders, members). Policies apply rules to attributes of manageable elements and can be used to coordinate operation of the devices users own. Each device is owned by some user, and therefore, may make use of the roles available to its owner.

The community policy is not static as users may propose changes to it. As an example of a situation requiring changes to the policy is the problem of channel assignment over the communication network. Due to interferences from external (even non Wi-Fi devices), the optimal channel may need to be modified from time to time. Moreover, the same channel may be incompatible with some of the devices, requiring local adaptations. In a traditional

network, a central administrator has the capability of imposing a policy to all devices. In a community, some user may have the delegated power to do this, acting as sole administrator after others trusted him the task. However, consensus mechanisms such as Paxos [175] can also be used, presenting many advantages. The benefit of deploying a consensus mechanism is that it can also be used for the purpose of evolving other aspects of the policy. At the same time, it allows members to have an active participation in the direction of the community policy.

Before entering in the specific community management mechanisms that constitute the core of this work, some discussion on the underlying architecture assumptions is required. In particular, how existing mechanisms for addressing (see Section 4.1.1) and naming (see Section 4.1.2) can be integrated into the vision of an Internet containing community driven networks. Also, some considerations regarding the impact of community merging and splitting need to be presented (see Section 4.1.3).

4.1.1 ADDRESSING AND IDENTIFICATION

In a community, communicating entities require the existence of unique identifiers for the purpose of creating communication channels. These identifiers can be considered at different layers, from the link to higher layer entities (e.g., users). The type of identifier used is different across layers, but links between these different identifiers may be deployed through the establishment of relations such as ownership. At the MAC layer, IEEE 802.11 and Ethernet devices have unique 48 bit identifiers, which are statically assigned at manufacture time¹. At the network layer IPv4 or IPv6 addresses identify devices and allow communication over the entire Internet. At a higher layer, other identifiers can exist, as it is normally done for distributed systems. Individuals also have identifiers, in the form of their user names, real names or simply virtual personæ.

Although some form of identifier hierarchy may be present, that is not always true. 48 bit MAC addresses, as well as IPv4 and IPv6 addresses, follow an hierarchical structure. Identifiers attributed to individuals or to higher layer distributed systems, such as in P2P systems are frequently not hierarchical, resulting in flat addressing schemes. Ad-hoc networks also frequently rely on flat addressing networks where all hosts share the same network prefix. Nevertheless, one aspect is common in all systems: each entity must have unique identifiers within the same domain. Otherwise, communication becomes impossible due to the existence of ambiguities in the definition of the actual endpoints.

¹An administrator can overwrite these identifiers, but this is not commonly done, and identifiers are expected to be unique.

For the purpose of addressing entities in communities, it is proposed the use of three different identifiers, each related to a different layer of the communication stack: social, community, system. Figure 4.2 depicts the proposed three layers, and well as the identifiers used. It should be considered that communities may have no authority over the network (i.e., operate as overlays). In this case, network addressing is out of scope for them. In communities mapped to actual network topologies, such as in MANET and WMN, network addressing should also be considered.

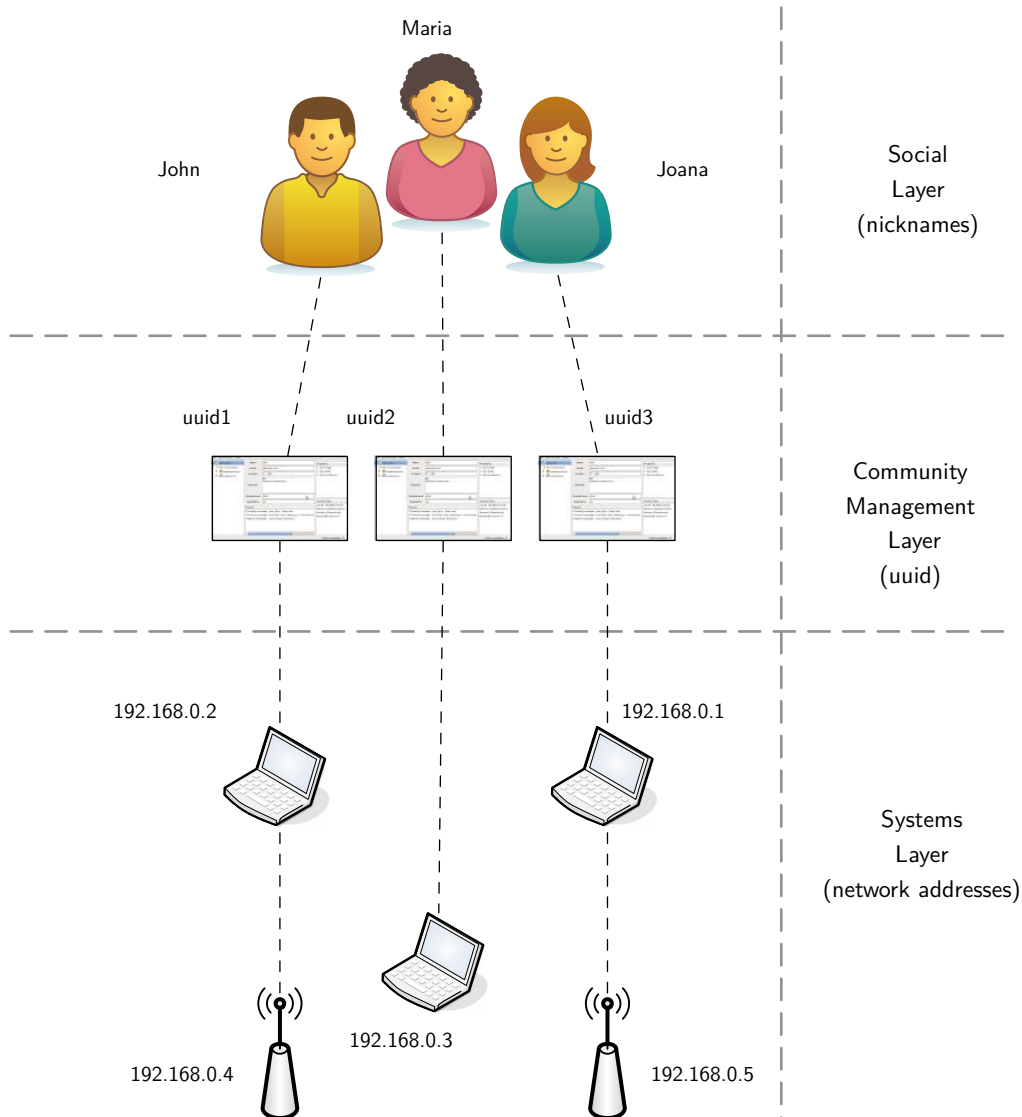


FIGURE 4.2 – Different identifiers in a multi layer community oriented communication system.

At the social layer, individuals interact in the digital world, making use of one or more virtual personæ. A nick name, or other simple identification handle (e.g., email address, phone number), provides a textual representation of this persona. Whether the persona allows the identification of the individual or keeps the information private is irrelevant for addressing, and is solely related to the purpose and design of the system. Initiating communication with a user resorts to indicating his handle. Directories and other referencing systems allow searching the handle by taking in consideration other details such as the email, name, or institution. For the purpose of communities, using the same approach is reasonable, as the mechanism is well known and understood by most users. The management plane of the community must guarantee that identification handles are unique, but there are no latency requirements as the detection of duplicate handles is only required when users first join the system.

Systems also require some form of identification handles. In the Internet, DNS names frequently play this role, and to some extent, considering single-homed environments, IPv4 addresses are frequently used to identify communicating systems. However, the use of IPv4 addresses, while required for communication, provides a bad solution for the purpose of identification. There are several motives for this: a single system may have multiple IPv4 addresses, and IPv4 addresses are linked to network topology. If the system is multi-homed, no single identification handle exists, which creates severe problems when routing messages between systems. Also, if systems are to be managed by some form of policy, having multiple identifications for systems requires the policy to be aware of it. A much worst situation is presented by mobility, or more generically, network topology changes. The way IPv4 subnets operate, when topology changes or systems change their point of attachment (effectively also a topology change), their address may need to be changed in order to provide connectivity in the new location.

To solve this issue where the identifier is also used as a locator [176], several mechanisms have been developed, relying in the existence of globally unique identifiers [177–179]. These entities can be used in indirection architectures providing unique addresses for systems with dynamic locators (IPv4 addresses). Communication is always established between the unique addresses, and an indirection mechanism deployed in key systems enables communications between the actual locators. Other solutions, as globally available P2P networks are example, consider that a unique identifier is associated to each system, which is mapped into a Distributed Hash Table (DHT). Moreover, P2P are designed to operate as overlays over the Internet. Naming, addressing and routing are implemented over the Internet, in overlays, with little dependencies on the underlying mechanisms currently available. Another reason for the adoption of custom indirection solutions, is that P2P systems frequently provide resource location mechanisms by means of highly efficient DHTs: translating identifiers to

IPv4 addresses can be supported with little effort.

Following the same approach, systems participating in a community can also rely in DHT for the creation of unique identifiers, like in P2P networks. When supported by a DHT or a centralized directory, the Duplicate Address Detection (DAD) mechanisms can be used both for detect duplicate system identifiers as well as other conflicting information items such as usernames. The format of such identifiers and the methods for their construction can be the solutions proposed for Universally Unique IDentifiers (UUIDs). In particular UUID version 5 [180] as this scheme results in the generation of constant identifiers for the same system². The advantage of having constant identifiers for each system is that policy information can use that information for aspects such as role delegation.

Addressing network interfaces is achieved through the use of solutions such as IPv4 or IPv6. Due to the scarcity of IPv4 addresses, when using this protocol, communities cannot rely on public addresses, being restricted to private addresses. These type of addresses are reserved by Internet Assigned Numbers Authority (IANA) for the purpose of Local Area Networks (LANs) and are not routed in the global Internet. The use of gateway nodes, making use of NAT is therefore required. Common routers, providing Internet connectivity to households rely on this technique, which maps an internal private network through a single public IPv4 address.

For the purpose of assigning addresses to actual network interfaces of a system, there are several well known and commonly used methods. All are adequate to the purpose of addressing in community networks, but not all present the same functionalities, behave equally, or scale equally.

- The most basic one is static assignment of IPv4 addresses to network interfaces, usually by a network administrator or the user itself. This approach has the advantage of requiring no additional software, but imposes that addresses must be manually re-assigned if systems change their location, or network addressing topology changes. More importantly, there is no method for detecting colliding addresses and the user must both detect and avoid collisions manually.
- Another approach which can be used relies in auto-configuration by stations. In IPv4 this is provided through Link-Local address assignment mechanisms [181], while IPv6 was developed with native auto configuration capabilities through Stateless Address Auto-configuration [182]. In both cases, special prefixes are reserved to be dynamically assigned by stations, followed by a DAD process to find, and ultimately avoid, collisions. A drawback of such solutions is that the DAD process can delay participation in the network. It relies in broadcasting messages to all nodes, looking for the candidate

²Other versions do not present the same property

address. If the network size increases, or more important, if its diameter increases, the delay associated with the DAD process, as well as the number of messages exchanged, will also increase dramatically [183].

- Yet another approach is the DHCP, which is probably the most widely used address assignment method. It relies in the existence of entities which are capable of assigning addresses (as well as other configuration parameters) on demand. Being the most used addressing mechanism, it also found a large acceptance in community based WMN deployments, however, not so much in MANETs. DHCP considers the existence of address pools which are kept by a single DHCP server. Clients request addresses and servers lease free addresses from the pool to clients. A mechanism is defined for keeping leases active. If a client doesn't renew its lease, the address may be provided to other client if required.

WMN deployments make use of DHCP with WMR playing the role of servers, and user terminals (laptops, PDAs, etc. . .) acting as clients. Each WMR provides a different subnet to its clients, and this information is propagated through the WMN routing mechanisms. In some deployments (e.g., Guifi), WMR make use of NAPT which limits interaction between clients. If addresses are to be provided within the same network, a centrally managed pool must be considered. For this purpose, the DHCP Relay agent [184] can be used. It states that access routers (fringe WMR) do not lease addresses but relay requests to other servers which are responsible for managing the address pool. If these central servers fail, no addresses are provided to clients. Due to the need of predefined servers, and the low redundancy of the DHCP system, MANETs frequently rely on auto configuration, or simply static configuration, as it lacks the special node roles (WMR) already present in WMNs. Table 4.1 presents a comparison between these methods.

From the perspective of community management, DHCP presents benefits as it allows rapid address assignment, as well as automatic configuration of critical aspects such as gateways, DNS servers or even proxies. Solutions based on DHT [185] can enable distributed DHCP by keeping a distributed pool, and eventually circumventing the limitations of traditional DHCP.

Solution	Architecture	Scalability	Dynamism	Configuration
Static	User Centric	very poor	low	high
Stateless Auto-conf	Distributed	poor	high	low
DHCP	Centralized Partially Distributed	high	medium	high

TABLE 4.1 – Comparison of different addressing solutions.

4.1.2 NAME RESOLUTION

In this context, naming refers to the structure and mechanisms deployed to attribute human readable identifiers to addresses, services, and other locators. Without names associated to locators, humans will have difficulty to interact in a network and use its services. This problem is aggravated on large networks, and on the Internet, as humans are unable to memorize all addresses and ports for the services they which to use. In dynamic network, such as ad-hoc networks, memory also serves no good because it is impractical to know which hosts are currently available. Names can also be used for the purpose of providing high availability, and mobility, if communications are established between names, instead of being established between locators. Solutions like Host Identity Protocol (HIP) [177] resort to this feature in order to provide host mobility, authentication, and even multi-homing [177]. Because the association between names and locators can be dynamically modified, clients can use the name to resolve its current locator. If a locator changes (e.g., node moves between networks), updating the name will enable new clients to be able to communicate correctly. Different clients can also be provided different locators for the same name, which enables name based load-balancing. Currently, these two situations are vital for the massive scalability and availability of Information and Communications Technology (ICT) cloud infrastructures.

When applied to communities, naming is vital for the purpose of finding communities, systems, services. Because communities imply the existence of users, and interactions are expected to be geared towards users, users should also be present in the naming infrastructure. A hierarchical format is expected as it reflects natural organization inside communities. The community scope naturally, comprises the root of the hierarchy . It has the purpose of grouping all other names which are associated to the community. That is, the users enrolled, systems, and services. Because communities can provide services to external users (or other communities), services are considered to rely directly below the community scope. However, systems and users can also provide services, which requires service names available under each of these scopes. The naming structure proposed in this thesis is depicted in Figure 4.3.

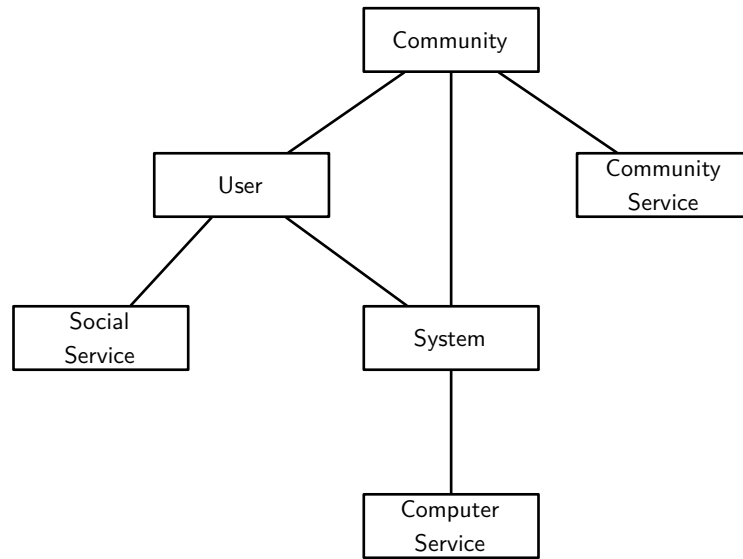


FIGURE 4.3 – Naming hierarchy in a community.

The hierarchy proposed is appropriate for a single community, or multiple communities. If a set of users or systems is common to both communities, each of these names would be available in each community. The result of a query over the name should be a locator, providing information regarding the name queried. Requesting the resolution of the User `John.CommunityA` provide the locators of its systems that are also associated with *CommunityA*³. Requesting `Device Johnslaptop.CommunityA` should provide the locator for communication with the device, in the framework of community *A*. Requesting `Service FTP.Johnslaptop.CommunityA` will provide a locator to the endpoint where the service is provided. `Service FTP.CommunityA` will provide a locator similar to the previous, if the service is being provided by the system named *Johnslaptop*. However, it could be provided by an aggregate of systems relying in load balancing methods.

Locators can include more information than just addressing information. The locator for a user of a system can also provide geo-referenced coordinates (e.g., from Global Positioning System (GPS)). Locators for systems can include information regarding the protocol and port to be used for communication. It should be noted that locators can be provided through multiple systems in order to provide communication with non community aware systems. In this case locators must be mapped to the destination system.

³This same approach is commonly followed by communication protocols such as Session Initiation Protocol (SIP) [186]

4.1.2.1 DOMAIN NAME SYSTEM

The DNS was created as a way to facilitate interaction, by providing the mechanisms to associate identifiers and network addresses (identifiers), as well as identifiers and textual information, and is the underlying resolution mechanism of today's networks. It assumes the existence of clients which issue name resolution queries, and servers which make information available to clients [187]. Following the DNS structure, names have textual representations and are organized in a hierarchical form, written from right to left. At the right most part of the name resides the Top Level Domain (TLD) [188], which denotes the root administrative domain of the name. As an example, each country has an associated Country Code TLD (ccTLD) [188] for the registration of names in their scope. Although not handled by the responsible authority (IANA) the pseudo domain `.local.` is used by Multicast DNS (mDNS) [189] for the purpose of holding names in the current network (other domains can be specified by system administrators). After the domain (from right to left) a set of names may further identify the context of the name, following the format of `identifier.host.top_domain.` (e.g., `ftp.hostA.local.`). DNS names are frequently used to build Uniform Resource Locators (URLs), which also specify the communication protocol to be used (e.g., `http://www.server.local.`). Without relying in URLs, the DNS infrastructure also allows the registration and query of services, by using specially crafted hierarchies and specific entry types. As an example, if a printer is to be made available by its name, and using Internet Printing Protocol (IPP) [190], the respective DNS record would have the format `_ipp._tcp.host.domain.tld.` The record type of these entries also differentiates information between services and general host names. While host entries make use of *IN A* or *IN AAA* types, service entries make use of *SRV* types [191]. In practice, these type of DNS entries are a corner stone of many existing commercially available solutions. Using the URL method, the same printer would be available through `ipp://host.domain.tld.` In this case, the protocol specifier *ipp* instructs the client application about the communication methods to be used for communication with the service.

In standard DNS, servers are connected in a tree shaped structure, with several root servers for the purpose of providing increased performance and high availability. When using mDNS flat topologies can be considered, such as a set of computers in a LAN. The two structures are depicted in Figure 4.4.

mDNS is a standard protocol which provides decentralized naming resolution. It makes use of the same messages as standard DNS but operates over multicast addresses, between neighbors, instead of relying on a pre-existing, and permanently available chain of servers. mDNS answers to the `.local.` TLD, and requests for this TLD should not be propagated to the public DNS infrastructure, as they refer to hosts and services which are not globally

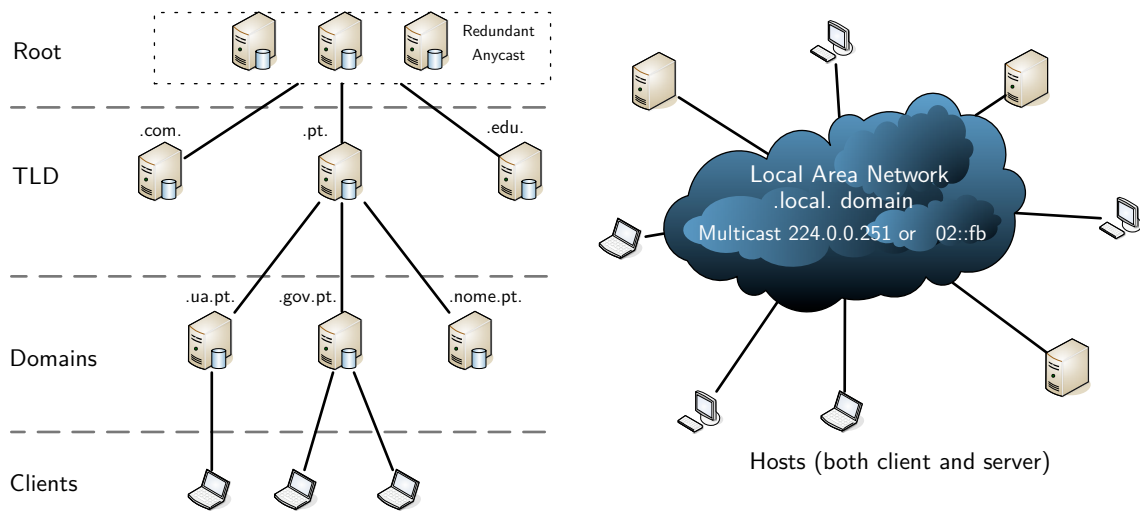


FIGURE 4.4 – Structure of DNS entities in standard DNS and in mDNS.

available. mDNS was developed for LANs, and considers that stations listen to mDNS messages and issue answers both for their entries, as well as for other naming entries which they cached. The message format is shared between DNS and mDNS, which facilitates the existence of hybrid scenarios. However, the semantics associated to each message, in some cases, present some differences.

Considering scenarios where there is no predefined infrastructure, distributed solutions such as mDNS are most probably candidates to provide name resolution. However, supporting mDNS over a WMN, MANET, or more generically, a community of interest, is not straightforward. While in LANs stations in the same IPv4 network are also in the same broadcast domain, in unplanned wireless networks this is not true. The same is valid for communities, where systems may not be direct neighbors (in terms of the underlying network topology). The reason is the limited radio range of wireless transmitters, and/or the need for multi-hop routing in order to reach all entities. Therefore, while mDNS is desired, its support requires specialized solutions. One of this solutions is presented by Proto *et al.* [192], which proposes that mDNS may exploit the MPR tree created by OLSR. mDNS messages are captured by routers and sent to the remaining network hosts through the already established MPR tree. This enables efficient flooding, and allows loop free broadcast of information to neighbors outside the communication range of the sender.

Independently of the information dissemination and retrieval method, community structure can be represented in the existing DNS infrastructure by means of a specific hierarchical structure. At its root ccTLD can be used if the community has a country wide scope, or an

additional `.community.` TLD could be introduced. Using the `.local.` TLD is not advisable because it refers to the local network, which may be disjoint with the existing communities in the same network. After the TLD, the community name can follow, like domains follow the TLD in the current DNS infrastructure. The result will be in the form `CommunityName.community.` Afterwards two approaches can be followed: a) further entities are added just below the community name; b) each entity is placed in a dedicated branch in the DNS tree. Both approaches are reasonable, and can exist simultaneously. Considering an FTP service in system *D* of community *C*, if the service is provided to external entities, it may make sense to simply register the service as `ftp.C.community` because it will be more easily accessed. A URL, or off band information regarding the protocol upon which the service operates must be available. Alternatively, it can be registered in a longer format `_ftp._tcp.C.community.` By extending this approach, the entire community structure can be considered if additional placeholders similar to `_tcp` are used. The following two placeholders are proposed, but others can be added as required.

- **`_sys`**: Denotes a hardware system, such as a router, laptop or any other network enabled host belonging to the community. The primary objective of this placeholder is to support name based, direct communication between hosts. Either they belong to the same community, different communities, or are not community aware. A system *S* mapped into this structure would be represented as `S._sys.communityName.community.` Its services can also be represented by prefixing the service name and protocol (e.g., `_ftp._tcp.S._sys.communityName.community.`).
- **`_user`**: Denotes a user enrolled into the community. The primary objective of this placeholder is to support easily mappable presence information and automatic membership assessment from outside entities. The result of mapping user *U* into this structure would be `U._user.communityName.community.` Ownership of a system can also be represented if the `_sys` placeholder follows the user (e.g., `S._sys.U._user.communityName.community.`)

When community structure is mapped into DNS, several limitations are imposed for the transmission of locators, and other information. DNS considers a fixed set of Resource Records (RR) types, which are documented in a series of IETF Request For Commentss (RFCs) and kept by IANA. Each RR has a determined purpose and specifies how information can be retrieved from the DNS infrastructure. The *TXT* record [187] was initially considered for generic human readable messages (text strings) and can be used for other purposes. Inside the *TXT* record, data is organized in strings of maximum length of 255 bytes. In fact, solutions such as DNS Service Discovery (DNS-SD) [189] make use of the *TXT* record in order to store arbitrary name/value pairs conveying additional information about the named service. Many

other solutions make extensive (abusive?) use of this record type to support custom records. Dissemination of community information can surely use the same approach by defining a custom structure for elements inside the *TXT* record. However, some optimizations can be considered which make better usage of other existing record types. At the same time this allows differentiated queries of the same name, with different results.

The following record types are recommended for mapping community structure to the DNS infrastructure:

- **A**: The *A* record stores an IPv4 address, allowing to directly contact a single host [187]. It can be applied both to systems and services, providing their IPv4 address.
- **AAA**: The *AAA* record stores an IPv6 address, allowing to directly contact a single host [193]. It can be applied both to systems and services, providing their IPv6 address.
- **CNAME**: The *CNAME* record is an alias to other records. In the current scope, it is useful for mapping detailed record entries, such as `_ftp._tcp.S._sys.communityName.community.` to easily available ones, such as `ftp.communityName.community..`
- **RP**: The *RP* record points to the Responsible Person [194]. In this scope, the *RP* record can be used to provide information regarding the ownership of a given system or service, which will be a user. The internal structure of the *RP* field makes it able to both identify the user and the domain, which in this case would be the community.
- **HINFO**: The *HINFO* records provides information regarding the Central Processing Unit (CPU) and the Operating System (OS) of a given system [187].
- **LOC**: The *LOC* record stores information regarding the current geographical location of an entity [195]. Location may be related to a system, a service, or even a user and is composed by latitude and longitude.
- **SRV**: The *SRV* record provides information regarding the location of a particular service [191]. Location is not geographical but logical, in the form of port and destination system (*A* or *AAA* records, but not *CNAME* records). Considering the previous examples regarding FTP. If the community provides an FTP service to the outside, it may provide an entry `_ftp._tcp.communityName.community.` in order to retrieve the address of the actual endpoint providing the service. For many reasons (e.g., load balancing), the actual address provided can be distinct to different querier clients. A feature which can also be used for the purpose of redundancy is the use of the *Priority* and *Weight* fields of *SRV* records. Multiple endpoints can be specified for the same service and clients should connect to the lowest priority endpoint first. If multiple endpoints have the same priority, the *Weight* field is used to select endpoint. While the *Priority* field defines the order for endpoints to be contacted, the *Weight* only specifies

the probability of it being selected. The selection algorithm proposed in [191] defines that higher *Weight* values will have higher probability of being selected.

The above mentioned record types allow services, systems and users to be mapped into the DNS structure. Internal policies are not considered as they relate to the internal operation of the community, and are not expected to be available to other entities. If this behavior is required, in addition to the previous ones, either the *TXT* record or additional, custom defined, records can be used.

4.1.3 PARTITIONING AND MERGING

It is not expected communities to be static and keep their addressing, naming and policy indefinitely. Due to scalability issues, connectivity, or simply due to changes in their administrative structure, communities can both be partitioned and merged. Partitioning a community considers that a single community can be split into two or more other disjoint communities. Merging considers the opposite: two or more communities, disjoint or not, join their members to form a single community. Figure 4.5 depicts the two processes.

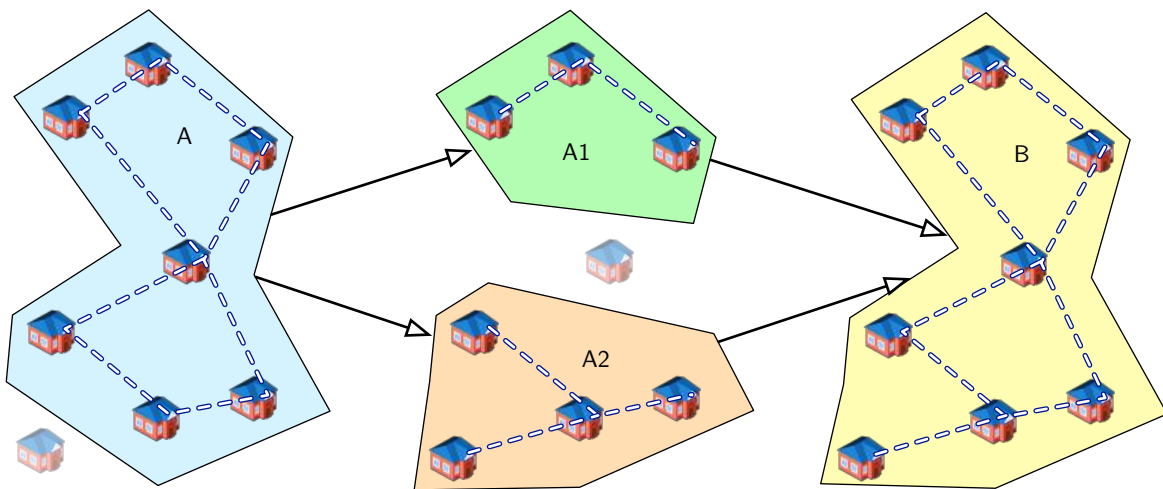


FIGURE 4.5 – Community partitioning (left), incorporation of a new member, and merging (right).

Partitioning can occur due to dynamics in the administrative model of the community or due to scalability concerns. The first may be the result of dissonant policies, while the second due to the need to create policies better adapted to a subset of the community. Network partitioning do not always imply community partitioning, unless network structure

is mapped to community structure and this is a requirement imposed by its members. In fact, due to the characteristics of the wireless medium, it is common for ad-hoc and mesh networks to suffer from temporary partitioning [196]. Communities can encompass several networks and keep operating coherently as long as there is communication between key entities routing communication. If disconnection occurs, the resulting partitions will act as independent entities with its own (similar) policy and set of members. The same will be true for communities.

From the perspective of addressing, considerations should be taken at each layer containing identifiers and locators. User identifiers and system identifiers will not need to change because they are expected to be relative to the community, and most importantly, they can rely in flat addressing schemes (such as when using UUIDs). The assumption that identifiers are unique in each community will lead also to the uniqueness of these identifiers in the resulting partitions. Also, no changes will be required for the addressing scheme of the underlying networks, if private addressing blocks are used. Because private addressing blocks are not routable in the Internet, these addresses can never be propagate outside the community network, and access to internal services must be done through gateway proxy hosts.

If the community structure, systems, and services are mapped into DNS, only the community name needs to be modified. This may have implications in the naming structure if, as used by DNS, naming is hierarchical. If the policy is somehow related to the naming structure present in DNS (e.g., the community name is referred in the policy description), the policy will also need to be adjusted accordingly. After this, the two communities will operate independently of each other, and each policy will evolve independently.

A particular problem with community partitioning arises from the inherent structure of trust and membership in communities. Trust relations created between members may be available to the community, for the purpose of maintaining a web of trust, and verify membership credentials. Each user is linked to the community through one ore more other anchors (also users). If the community is partitioned, users may find in a situation where no anchor remains available in the local community. In practice, because the trust tree is broken, other members will have no way to verify the membership trust of some leaf users. In order to mitigate this, users should be encouraged to get as many anchors as possible. In the scope of the current work, this issue is minimized because the community structure is defined by friendship relations, as stated between users. Interactions between members are expected to increase the number of links between users, therefore increasing the number of anchors available to each user. However, some corner cases will have no clear solution. These occur when all friends of a given user move to another partition and leave the user behind. This

situation is depicted in Figure 4.6. Abandoned users will need to recreate at least one anchor in the local community, so that membership is reasserted.

Solving the issues related to partitioning the web-of-trust would require the existence of a global identity provider, as well as federation functionalities between communities. This would allow verification of the trust chain in foreign communities. However, it would also require the definition of trust relations between communities and a global infrastructure providing identity management functionalities. Considering that communities can be formed in an ad-hoc manner, and can even be disconnected from the Internet, it is not clear how this challenge can be easily overcome. Specially because partitioning of the community can be the result of isolation created by partitioning the network and absence of communication between the two partitions.

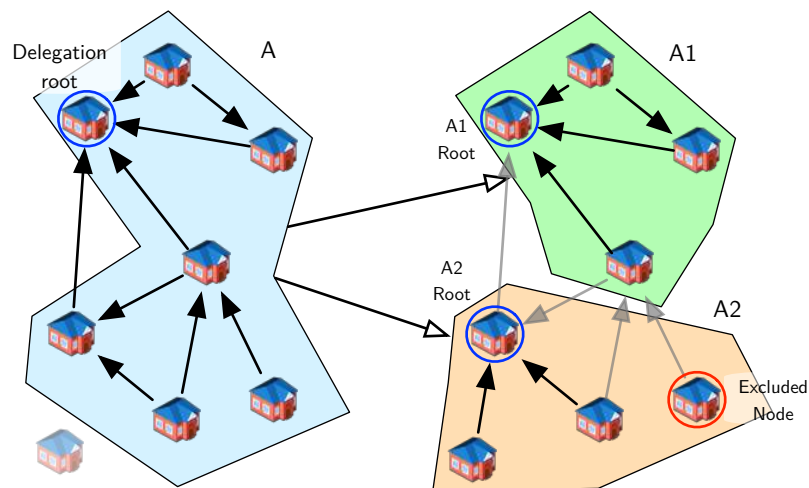


FIGURE 4.6 – Community partitioning and delegation anchors.

The counterpart of community partitioning is merging (see Figure 4.5). Two communities may merge due to administrative reasons in order to also merge the set of users and services they contain. It can also be considered that a community may temporarily be partitioned in several other, which later are re-merged together (e.g., when connectivity is restored). The result of merging two communities may result in a new community, and the destruction of (at least) two, or may result in the assimilation of a community into other. Assimilation will always be simpler than actual merging.

From an addressing and naming perspective there are two aspects which should be considered: structure and collisions. If identifiers are built in a flat space (e.g., UUID or nick names), the same identifiers can be maintained after the merging. However, if they are based

Community Size	nick names ⁴	IPv4 private ⁵	IPv6/64 network	128bit UUID
10	1.77×10^{-11}	2.79×10^{-6}	2.71×10^{-18}	1.47×10^{-37}
100	1.77×10^{-9}	2.79×10^{-4}	2.71×10^{-16}	1.47×10^{-35}
1.000	1.77×10^{-7}	2.76×10^{-2}	2.71×10^{-14}	1.47×10^{-33}
10.000	1.77×10^{-5}	9.38×10^{-1}	2.71×10^{-12}	1.47×10^{-31}
100.000	1.77×10^{-3}	1.00	2.71×10^{-10}	1.47×10^{-29}

TABLE 4.2 – Collision probability of different identifiers for different community sizes.

in some hierarchy, they must be modified to reflect the final structure. If DNS is used to represent the structure of the community, identifiers can be recreated in the final community under the root of the new community name. Network addresses are more vulnerable to such problems, in particular if the community requires some fixed network prefix. However, if no requirement is imposed, the resulting community can operate over two distinct networks as long as routing between networks is provided. If the prefix is equal, it can be maintained after the transition. Considering that in IPv4 the set of private addressing blocks is reduced, the probability of two communities sharing the same network prefix, or different prefixes which can be aggregated, should be considered.

When identifiers of the original communities are compatible with the resulting community, they can be kept. However, this assumes that no collision occurs. Considering two communities *A* and *B*, if they both have a user named *John*, respectively *John.user.A.community* and *John.user.B.community*, when they merge into *C*, there is a collision between the users and only one can keep the same nickname. Otherwise, the reference to *John.user.C.community* would be ambiguous. Identifiers such as UUID or IPv6 addresses have a large number of bits, which results in a reasonably low probability of creating collisions, even for large communities. The actual probability of having at least one collision is calculated according to the birthday problem as presented in Equation 4.1.

$$m \approx 1 - e^{-\frac{n^2}{2t}} \quad (4.1)$$

Where *n* represents the number of identifiers in the given universe of identifiers of size *t*, and *m* represents the resulting collision probability. Applying Equation 4.1 to different sets of identifiers and community sizes (Table 4.2), it is clearly seen that the probability of collision for some of them can be high. Considering a community with 1000 members, and considering that all private IPv4 blocks are available for use, the probability of having at least one collision between addresses is approximately 2.8%.

Colliding identifiers must be regenerated until no collision occurs. Considering that some identifiers can be present in the low level community policies, also policies must be updated in order to include the new identifiers. Still, the process can be done automatically by a merging algorithm, even when considering usernames by the inclusion of numeral suffixes (e.g., John -> John123) or other disambiguation method, policies can also be updated accordingly.

A major issue with merging two communities is the generation of a single policy, able to take in consideration the multiple original policy elements. While collisions in identifiers are easily solved, collisions in the policy domain require highly complex heuristics, that process the original policies at a semantic level, producing a compatible final policy. Even if an optimal algorithm is present, semantic merge cannot be always possible if there are conflicts at the semantic level. Such conflicts occur when there are predicates which state incompatible policies (e.g., states are exclusive), or when different predicates apply to the same attribute of the same manageable object. Moreover, even if the policies are not immediately incompatible, they could lead to a policy that deeply differs from the one desired.

The use of ontologies can lead to better handling of policy objects, and can enable the definition of automatic algorithms to handle conflicts, especially because they reduce the heterogeneity of the policies to be merged. Also, if both community policies to be merged have known ontologies, they can be aligned, mapped and merged [197]. This however still doesn't prevent semantic collisions as policy merging and semantic interpretation still constitute a field of research by itself.

4.2 COMMUNITY MANAGEMENT POLICY LANGUAGE

The policy issue is then a central problem for community management. Management of a set of devices and users requires the existence of a modeling language representing the structure of data elements and relations between these elements. This is how information is represented. The actual information elements could be defined *a priori*, but such approach would limit the expressiveness of the model, and several constraint community actions. In particular it would reduce its scalability to situations different from the ones originally envisioned. Instead, and like many other authors propose [198], a more appropriate method is to define an ontology, creating a taxonomic hierarchy of concepts (often named *Classes*), containing attributes, collections and other constructs. The ontology also includes the definition of the relations between *Classes* and can be extended by other constructs such as *Rules*,

⁴Name composed by 8 letters or digits. Key universe is $(26 + 10)^8$

⁵Total of 17891328 addresses

Axioms, Restrictions or Events. The instantiation of the concepts defined are called *Individuals*, which should be represented and manipulated according to the ontology. These are the actual information elements exchanged between participants and the building blocks for the community knowledge. That is, *Classes* and other concepts do not contain actual information about the policy, only the *Individuals* have this information.

The benefits of using ontologies are two fold: i) each community can use the ontology definition to create domain specific knowledge by generating specific data elements which must agree to the ontology; ii) the task of adding a new data type, and further extend the usefulness of the model is simplified, as the relations and details of the remaining concepts are formally stated and well known.

A description of the most relevant concepts associated with the policies used in this thesis, and instantiations of the concepts, are presented in the following pages. It should be noticed that concepts are built by using object oriented principles, namely inheritance and composition, with a strong favoring of composition over inheritance.

4.2.1 POLICY BASE CONCEPTS

4.2.1.1 CMOBJECT

From the perspective of inheritance, in the base of the data model resides the abstract concept of a *CMObject*. Other concepts must be derived from the *CMObject* concept and can use both inheritance and composition to further define their internal structure. This concept defines the basic attributes that must be supported by all other definitions, namely: *Type, Unique Identifier, Owner, Name, Parent, Modification Date*, and an optional *Signature*. The existence of these common attributes allows the identification of the specific object, even if its content is not understood by a particular implementation. In this case, unknown types can be safely ignored from being processed and new object types can be added in specific communities.

Because the target environment of this work are dynamic communities, which are inherently distributed, objects are organized as individual information elements, which can be transmitted and manipulated individually, while organized in a hierarchical tree. The hierarchy does not reflect inheritance, only composition, and the path from the root to the object is explicitly included in the object. Considering two objects *A, B*, an object *C* with path *A/B/C* does not inherit attributes and relations from *B* and *A*. Instead, the path represents that one or more *B* objects are contained in *A*, and one or more *C* objects are contained in *B*.

From the perspective of composition, two concepts can be considered as roots of all other

concepts: a *Community* and a *User*. That is, while the remaining objects will not inherit from the *Community* and *User* concepts, their instantiation will be tied to one *User* instantiation, one *Community* instantiation or both if the object links a *User* with a *Community*. This topic will be better clarified in the following sub-sections as each concept will be described.

All fields of the *CMObject* concept are depicted in Code 4.1, and described in the following lines. The *Name* and *Parent* fields are representations of the actual object name (in human readable form) and the parent the path (in the hierarchy tree) to where the object belongs, and constructed as the concatenation of the *Name* and *Type* fields of the upper objects.

```
CMObject[
  Type      => Enum
  UID       => UUID   % Digest over Parent:Type:Name
  Parent    => String % Path until root concept
  Owner     => String % Owner Name
  Name      => String % Object Name
  Created   => Integer % Creation Date
  Updated   => Integer % Update Date
  Signature => String % Cryptographic signature (PEM)
]
```

CODE 4.1 – Description of the *CMObject* concept.

The *Unique Identifier* (UID) is a construct that clearly identifies objects and allows establishing relations between different instances. Identifiers must be globally unique inside their scope and can be obtained through many methods, such as a cryptographic grade digest (e.g., Message-Digest algorithm 5 (MD5) [199] or Secure Hash Algorithm 1 (SHA-1) [200]), computed over the complete object composition hierarchy (*Parent* plus *Type* plus *Name* fields). The solution proposed in [180], is also of possible use if considering UUID versions 3 or 5. The basic requirement is that it should be possible to construct the *Unique Identifier* from the remaining metadata, and the value obtained should be constant for the same metadata. Therefore enabling the creation of loose relations between objects with unknown *Unique Identifier* but with known *Name*, *Parent* and *Type*. The result will be optimized management of data in a distributed environment, by making it possible to use solutions based in key retrieval processes such as DHTs.

The *Owner* field is a *Unique Identifier* that unequivocally identifies the user responsible for the creation of the object. Because users cannot create digital information (as they only interact with data through a device), because devices are owned or operated by users, and because objects may be created dynamically without direct user input, the *Owner* field finally corresponds to the owner (a user) of the device that effectively created the object instantiation.

The existence of this field is required in order to later evaluate access permissions, validate information integrity or audit user behavior.

The *Signature* field is optionally present in objects and contains a cryptographically generated authentication code computed over the object, with the *Signature* field blank. A PKI is required for the support of signatures and allow secure manipulation of the *Community* policy, with integrity control. The private key of the object owner should have produced the authentication data present in this field. This work doesn't focus in the cryptographic support and actual security infrastructure. However, many existing solutions can be applied in order to secure the entire process like Open Pretty Good Privacy (OpenPGP) [201].

The *CObject* concept is considered abstract and cannot be directly instantiated. Therefore, the *Type* provides information about the real nature of the information element, and the lower concept must be instantiated instead. If a particular instantiation of the software cannot understand a specific *Type*, it can detect this and skip over it. However, if the policy requires the result obtained by processing an unsupported element, the *User* may be restricted access to services, roles or to the entire community.

4.2.1.2 ATTRIBUTE

The *Attribute* concept represents any existing element that is to be manipulated by the management software. Therefore, *Attributes* are a representation to the management plane of manageable items and some value, and provide a bridge between the physical world and the management policy. *Attribute* objects may represent configuration parameters, sensed data, operational states or even actions. They have a relation to an object, in the sense that they represent elements of an object, typically a device, service or system. As an example, the IEEE 802.11 channel is an operational parameter of a specific IEEE 802.11 enabled device, while the ADSL Bit Error Rate (BER) is considered as sensed data of a specific ADSL enabled device. Attributes may also be used to either set or retrieve the operational status of a component, service or system.

```
Attribute[
  :CObject

  AttributeType => Enum % Value Type (String, Int, Date,...)
  Value         => Data % Data of the Attribute
]
```

CODE 4.2 – Description of the *Attribute* concept.

This concept derives from the *CMObject* concept and inherits all its members. The *Name* of the *Attribute* states where the *Value* is to be applied, or read from (see a description in Code 4.2 and an instantiation in Code 4.3). The actual name of the *Attribute* is not defined *a priori*, and the actual set of names supported by a given implementation is expected to vary.

```
<Attribute name='NetServ_Forward_State' attributeType='Int'>1</Attribute>
```

CODE 4.3 – Instantiation of the *Attribute* concept setting the state of the Forward service to 1 (on).

4.2.2 STRUCTURAL POLICY CONCEPTS

4.2.2.1 COMMUNITY

The *Community* concept is one of the root concepts as it represents a domain where most of the other concepts can be instantiated. The *User* (see Section 4.2.3.1) and *System* (see Section 4.2.3.2) concepts exist independently of the community, in the sense that both users and their devices still exist independently of their membership status to any community.

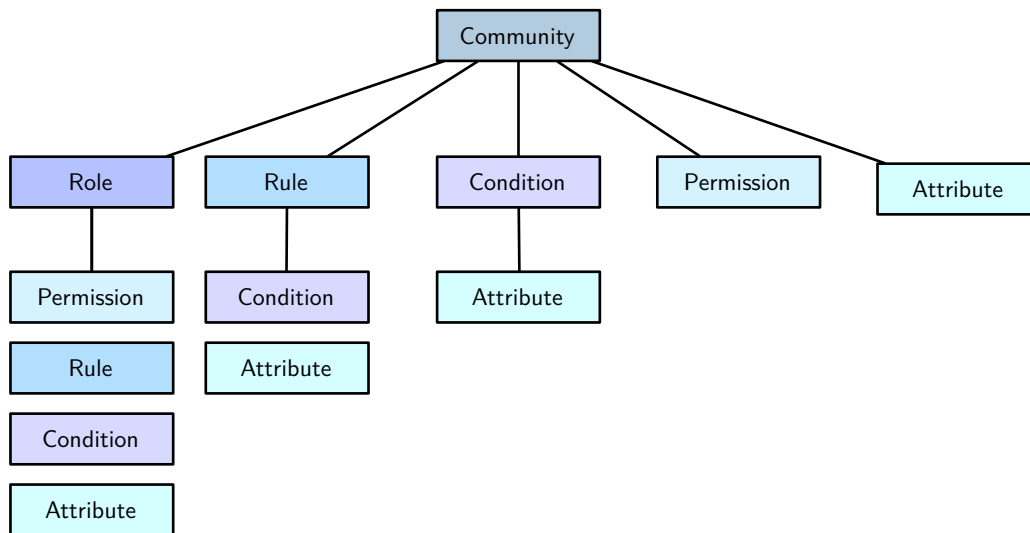


FIGURE 4.7 – Concept composition diagram of the *Community* concept.

The instantiation of the *System* can have associativity relations to one or more communities through its owners (a user). All the remaining concepts are dependent on a given *Community*, the policing domain. The hierarchical structure of the Base and Structural concepts with the

Community concept as its root, from the point of view of concept composition, is depicted in Figure 4.7. The *CMObject* is not represented because it is abstract and cannot be instantiated.

This concept inherits all the field of the *CMObject* concept, adding two additional ones: *Tags* and *Description*. The *Tags* and *Description* fields provide additional textual information to users about the particular community instantiated. This is particularly relevant because communities are expected to be advertised to directories and to the network for users to consult.

```
Community: [
  :CMObject

  Tags          => String    % Words describing Community
  Description    => String    % Textual description of Community
  Permissions{} => Collection % System ID
  Conditions{}  => Collection % Conditions for enrollment in any role
  Attributes{}  => Collection % Set of Attributes to apply
  Rules{}       => Collection % Set of Rules to apply
  Roles{}       => Collection % Set of Roles to make available
  Services{}    => Collection % Set of Service permissions
  Signatures{}  => Collection % Signatures from Owner and modifying User
]
```

CODE 4.4 – Description of the *Community* concept.

Besides *Tags* and *Description*, the *Community* concept may contain several other concepts that are required for proper maintenance of the policy and operation of the community. The most relevant of these concepts are: *Attribute* (see Section 4.2.1.2), *Role* (see Section 4.2.2.2), *Rule* (see Section 4.2.2.5), *Permission* (see Section 4.2.2.3), *Condition* (see Section 4.2.2.4), and *Service* (see Section 4.2.3.4). A particular community may have an undetermined number of instantiations of these concepts, and they are expected to be modified as the community policy evolves. While the *Community* concept represents the policy domain, the remaining concept instantiations specify the behavior expected, rights, permissions and obligations, to be applied to all participants. The structure of the *Community* concept is represented in Code 4.4.

Concepts such as *Attribute*, *Rule* and *Condition* present at the top level of the *Community* are considered as global for all participants. As described previously, if these concepts are contained inside a *Role*, they are only valid for members of that role. Code 4.5 presents an instantiation of the *Community* concept, containing several global concepts and roles.

```

<Community name='Name' created='1328619252' updated='1328619252' owner='User' Name tags='s1,s2,s3'
  description='example'>
  <Permissions> . . . </Permission>
  <Conditions> . . . </Conditions>
  <Attributes> . . . </Attributes>
  <Rules> . . . </Rules>
  <Roles> . . . </Roles>
  <Services> . . . </Services>
  <Signatures> . . . </Signatures>
</Community>

```

CODE 4.5 – Instantiation of the *Community* concept and its inner concepts.

4.2.2.2 ROLE

The concept of role is a well established concept in many policy mechanisms that are constructed around the RBAC and Role-Based Management (RBM) models. In the scope of community management, the *Role* concept groups subjects (either individuals or electronic devices) and can be described as a concept which specifies the behavior, rights and obligations of the ones enrolled. The behavior relates to how subjects must operate in a particular situation. In multi-hop networks this translates to the fact that nodes should favor packet forwarding, sometimes in detriment of their own generated packets. The rights specify which operations, services are resources are available and can be used by the subject, always according to the expected behavior. Finally, the obligations state what operations must be supported, services must be provided or what resources subjects must have in order to enroll a particular role.

In the context of a community, the most basic role available to participants is the *Member* role, which defines the basic properties for membership: what behavior is expected; what services are available; what restrictions should be applied and services must be provided in order for a subject to be considered a member of the community. Higher level roles can exist, each further refining the policy to the ones enrolled, and communities are free to create any number of roles.

The *Role* concept is composed by one or more *Attributes*, *Rules*, *Permissions* and *Conditions* (see Code 4.6). Together, these concepts enable a fully functional RBM model. A list of *Permission* objects enables discretionary access control to limit manipulation of the role (e.g., modification and delegation) and effectively limit which subjects can add new objects to the Role or modify existing. It also limits what subjects are allowed to further delegate this Role to the remaining participants.

Similarly to the case of the *Community* concept, one or more *Condition* objects may be present and are used to limit enrollment, by defining a set of attributes to be validated before a subject is entitled a role . If applied to the *Member* role, conditions can be used to effectively restrict participation in a community, or to enforce separation of duties by applying mutual exclusion of roles. One common example of this discretionary participation are some P2P file sharing applications requiring that in order for a user to have access to the files being shared, he must also share a minimum amount of data. Separation of duties is also relevant in common life scenarios, especially for performance purposes as in the case of restricting the combination of services provided by nodes so that a given service level is maintained. Also, separation of duties creates barriers so that all individuals are restricted on the amount of influence they have towards other members.

Behavior, rights and obligations are stated in the form of *Attribute* objects. These objects can either be global to the *Role*, and thus must be applied independently of the situation, or as a part of a *Rule*, thus only be applied on a specific situation as stated by the *Condition* present inside the *Rule* instantiation.

```

Role: [
  :CObject

  Permissions{} => Collection % System ID
  Conditions{}  => Collection % Conditions for enrollment
  Attributes{}  => Collection % Set of Attributes to apply
  Rules{}       => Collection % Set of Rules to activate
]

```

CODE 4.6 – Description of the *Role* concept.

Code 4.7 depicts a role named “BasicRole” with several permissions over access to roles (specifically “HigherRole” and “BasicRole”), as well as a undefined number of enrollment conditions, static attributes, and rules.

```

<Role prefix='CommunityName' name='BasicRole'>
  <Permissions>
    <Permission read='false' write='false' delegate='true' Role='HigherRole' />
    <Permission read='true' write='false' delegate='false' Role='BasicRole' />
  </Permissions>
  <Conditions>
    <Condition> ... </Condition>
  </Conditions>
  <Attributes>
    <Attribute> ... </Attribute>
  </Attributes>
  <Rules>
    <Rule> ... </Rule>
  </Rules>
</Role>

```

CODE 4.7 – Role with containers for Permissions, Rules, Conditions and Attributes. Only members of HigherRoleName can delegate or modified it.

4.2.2.3 PERMISSION

The *Permission* concept defines the requirements for managing an object, and in the case of a *Role*, the requirements for further issuing *Delegation* tokens. This concept does not inherit its structure from the *CObject* concept. It is composed by boolean permission *Flags*, plus a *Subject* and an *Object* fields. The permission *Flags* are *Write*, *Read*, *Delete*, and *Delegate*, and respectively define if the *Subject* can modify, read, delete the *Object* (see Code 4.8).

```

Permission: [
  Subject => CObject % To whom this permission applies
  Object  => CObject % To what this permission applies
  Read    => Boolean  % Right to Read
  Write   => Boolean  % Right to Modify
  Delete  => Boolean  % Right to Delete
  Delegate=> Boolean  % Right to Delegate (Delegations)
  Access  => Boolean  % Right to Access (services)
]

```

CODE 4.8 – Description of the *Permission* concept.

If the object is a *Role*, the *Delegate Flag* state that the *Subject* can issue *Delegation* (see Section 4.2.2.7) tokens to that *Role*. The *Subject* field points to a *User* by its *Identifier* while the *Object* identifies any existing object by its *Type* and *Name* (e.g., a *Community* or a *Role*).

In the scope of this work, the *Permission* concept effectively limits how the policy can be manipulated and what information is available to its participants.

Code 4.9 depicts the instantiation of the *Permission* concept in the scope of a role named *Member* in a community with at least two roles (*Member* and *Leader*) and one service (FTP). In this case, users assigned to the *Member* role will be able to access the policy specification of the *Member* role, but will be unable to access the specification of the *Leader* role. The FTP service is also restricted to users assigned to the *Leader* role. Users assigned to the *Member* role should not try to use the FTP service, and access from these users will be denied.

```
<Role parent='CommunityName' name='Member'>
  <Permissions>
    <Permission Read='true' Write='false' Delete='false' Subject='Role.Member' />
    <Permission Read='false' Write='false' Delete='false' Subject='Role.Leader' />
    <Permission Access='true' Subject='Role.Leader' Object='Service.FTP' />
    <Permission Access='false' Subject='Role.Member' Object='Service.FTP' />
  </Permissions>
</Role>
```

CODE 4.9 – Instantiation of the *Role* concept and basic permissions.

4.2.2.4 CONDITION

The *Condition* concept is composed by a *Attribute* with two internal fields: *Operator* and a *MatchType* associated to it (see Code 4.10). The *Attribute* identifies what value is to be matched and can relate to an attribute inside any other concept (e.g., relate to an instantiation of an attribute in a device or service). Therefore, it is possible to deploy conditions that cover all aspects of the community policy.

```
Condition: [
  :CObject

  Attribute => Attribute % The Attribute to evaluate
                % and the value to match
  Operator  => Operator  % The Operator to apply
  MatchType => Enum      % The type of match to use

]
```

CODE 4.10 – Description of the *Condition* concept.

The *Operator* represents the standard arithmetic operators (*less than, less or equal than, equal to, greater than, greater or equal than*), and add other mathematical operators such as *exists* and *isEmpty*. The *MatchType* states what is the relevancy of the condition for the calculation of a decision. It can assume the values of *Optional, Sufficient, and Required*:

- An *Optional* condition is evaluated but only results in a decision if it is the only condition evaluated to produce a decision.
- A *Sufficient* is evaluated and if successful, an affirmative result is provided, otherwise the result is ignored.
- A *Sufficient* condition is the only one present, its result will produce the final decision.
- A *Required* condition that is not matched will result in an immediate action. If it matches successfully, the remaining conditions should be evaluated. By default, if no *MatchType* is provided, the *Condition* is assumed to be *Required*.

Code 4.11 depicts a condition used to determine if an user has the Member role. If the opposite condition was to be defined, (e.g. as a way of defining conflicting roles), the operator would have to be modified to “!=”.

```
<Condition name='isMember' operator='=' matchType='optional' >
  <Attribute name='hasRole' type='String'>Member</Attribute>
</Condition>
```

CODE 4.11 – Condition evaluating if subject hasRole Member.

4.2.2.5 RULE

The *Rule* concept provides the mechanism for the definition of conditional behavior, rights and obligations, following an ECA approach. *Rule* is composed by one ore more *Condition* instances and zero or more *Attribute* instances. Whenever the *Condition* is met, the attributes (if any) should be made active. This concept covers the limitation of global attributes in communities and roles, by specifying in which situations and how these attributes should be applied.

Besides conditions and attributes, fields stating how the rule should be evaluated, and how its output should be processed, are also included in the *Rule* concept. Rules should be evaluated when the attributes in its conditions are updated so that the active policy correctly follows real world events. One issue with this approach is that not all attributes can be evaluated reactively when they change because the underlying sub-system may not provide the required event driven interface. One example is the amount of traffic that crosses

an interface. There is no notification to applications (at least in Windows, Linux and BSD systems) when the value changes. In fact, supporting this notification would imply a great amount of overhead, as any packet received or sent through an interface would trigger the event. To cope with this issue, the *Rule* concept has a *Period* field, stating the periodicity between evaluations if not all attributes support modification triggers, thus following a poll approach instead of the preferred event driven, asynchronous, approach.

```

Rule: [
  :CObject

  PublishIfSuccess => Boolean    % Publish result if Condition is true
  PublishIfFail    => Boolean    % Publish result if Condition is false
  Period           => Integer    % Sampling period in seconds
  Condition        => Condition  % The Condition to evaluate
  Attributes{}    => Collection  % Set of attributes to apply
]

```

CODE 4.12 – Description of the *Rule* concept.

Rules can also be used to add information to the knowledge base. Two fields named *PublishIfFail* and *PublishIfSuccess* state if the values contained in the *Condition* should be made public. The first field applies to the situation where the *condition* do are not met while the last applied to the situation where the *Condition* is met. Publishing the result implies instantiating a *RuleResult* object and publishing it to the remaining participants. The structure of the *Rule* concept is described in Code 4.12.

The *Rule* concept also allows the transposition of values from the *Condition*, in the sense that information from the *Attribute* evaluated can be sent to another *Attribute*. This is represented by not providing a value for the *Attribute* to set when the condition is met. As illustrated in the example of Code 4.13, if the cost of interconnecting to the Internet is not 0, the value read is written to the *NetServ_Gateway_GWCost* attribute, thus effecting the cost of the gateway advertised by the routing protocol (e.g., Host and Network Association (HNA) messages in OLSR).

```

<Rule name='InternetHighCost' period='5' publishIfSuccess='1' publishIfFail='0'>
  <Conditions>
    <Condition name='InternetCost' operator='>='>
      <Attribute name='NetServ_Gateway_Cost' type='Int'>0</Attribute>
    </Condition>
  </Conditions>
  <Attributes>
    <Attribute name='NetServ_Routing_GWCost' type='Int'></Attribute>
  </Attributes>
</Rule>

```

CODE 4.13 – Rule stating that the administrative cost associated to routes with outside hosts is tied to the cost reported by the Gateway service (monetary cost).

4.2.2.6 RULERESULT

As described previously, the *Rule* concept enables the collection of sensed data that is published by instantiating one or more *RuleResult*. This concept contains the identification of the system, and a list of *Attribute* fields. It states to others what was the value of the listed attributes when the *RuleResult* object was issued. This concept inherits the structure of the *CObject* concept. Therefore, it also indicates the time in which it was created, and the owner of the system which issued the *RuleResult*. Through the parent field, also inherited from the *CObject*, it is possible to determine the *Rule* which triggered the creation of the *RuleResult*. Code 4.14 describes the structure of the *RuleResult* concept, while Code 4.15 illustrates an instantiation of the concept.

```

RuleResult: [
  :CObject

  System          => System      % System ID
  Attributes{}    => Collection  % Set of attributes to apply
]

```

CODE 4.14 – Description of the *RuleResult* concept.

The motivation behind publishing *RuleResult* objects comes from many different factors, the most relevant are:

- Events may only be noticeable after analyzing results from several participants (e.g., wireless interference), or it may only be relevant if notified by a significant amount of participants.

- Publishing attributes to others may be used to policy the validity of a delegation that is constrained by some rule. Both the delegation and the result of the evaluation will be present. Therefore it is possible for any participant to infer if the reporting node should still possess the delegation.
- Publishing data publicly also enables to forecast required changes or to take in consideration a large number of participants when taking decisions. As an example, if all devices publish the hourly average noise figures for all IEEE 802.11 channels, a coordinating node will be able to modify the policy so that the channel presenting best performance for all the network is used for communication.

```
<RuleResult parent='Rule.GatewayCost' created='1328529486' owner='John' system='JohnsLaptop'>
  <Attributes>
    <Attribute name='NetServ_Gateway_Cost' type='Int'>15</Attribute>
  </Attributes>
</RuleResult>
```

CODE 4.15 – RuleResult containing the cost of connecting to the Internet for system JohnsLaptop.

4.2.2.7 DELEGATION

The *Delegation* concept provides a link between a subject and an object, as stated by an issuer. Under the scope of community networking, both issuer and subject are an *User*, while the object is a *Role*. Therefore, the delegation concept asserts the existence of a trust relation between issuer and subject, as far as the object is concerned. The issuer provides an assurance, both to the subject and to others, that the subject is entitled to that particular role. The result is that the subject will need to comply with the specificity of the policy applied to the role, thus conditioning its behavior, providing rights, and enforcing obligations. Attached to every delegation requiring the validation of one or more attributes are also a list of attributes (*Proofs*) provided by the subject to the issuer, which correspond to the values the issuer considered when creating the delegation.

Besides subject, object, issuer, and proofs, the *Delegation* concept is also composed by several other information elements, namely: *Start Date*, *Expire Date*, *Right to Delegate (RtD)* and *Expired*. The *Start Date* marks the date from which the delegation is valid, while the *Expire Date* mark the date until which the delegation is valid, thus limiting the time interval where the subject is entitled to the role.

```

Delegation: [
  :CObject

  start      => Timestamp % Start time (secs since EPOCH)
  expire     => Timestamp % Expiration time
  revoked    => Boolean   % Revocation flag
  delegate   => Integer   % Delegation count
  Issuer     => User      % Delegation Issuer
  Subject    => User      % Delegation Receiver
  Object     => CObject   % Delegated object
  Proofs{}   => Collection % Attributes provided by Subject
  Signatures{} => Collection % Signature of both Issuer and Subject
]

```

CODE 4.16 – Description of the *Delegation* concept.

The *Expired* flag allows to explicitly state that the delegation is revoked. It can be used to inform others that the conditions that were required for the trust relation are no longer valid, or the issuer no longer trusts the subject. Having no delegation to a role, or a delegation with the revoked flag will yield the same result. A revoked delegation doesn't provide a hint that the issuer doesn't trust the subject in relation to the object. This is true because the subject can revoke any delegation issued to him, therefore breaking the relation and releasing him from the any active obligation. Due to the existence of this flag, dissemination to the other members will also be faster as there is an object clearly stating that the existing association is not longer valid.

The *Right-to-Delegate* provides the permission for the subject to further delegate the role. When the object is the *Member* role, re-delegating this role will effectively grant membership. In order to limit re-delegation, the *Right-to-Delegate* field is not a boolean flag but a counter that is decremented on each new re-delegation, the *Delegate* field. That is, a new delegation should have a *Delegate* value inferior to the one present in the issuers' delegation. Because issuing delegations constructs a tree, it is possible to verify if members of a role actually decremented the counter for new delegations by comparing the value present in a particular delegation with the value present in the delegation of the issuer. When the *Right to Delegate* reaches the value of 0, the subject will be enrolled to the role but he will be unable to provide the same statute to others. By carefully defining the *Right to Delegate*, roles, conditions and permissions, the individual that first created the community is able to limit both the manipulation of the policy as well as its spread. As an example, a community could be deployed so that members are unable to further expand the community, because they are unable to re-delegate the *Member* role, while Leaders (some other role with more permissions) will be able to do so by issuing instantiations of the *Delegation* token.

```

<Delegation prefix='Community.Name' name='SubjectName:RoleName' created='0' updated='0'
  owner='IssuerName' start='0' expire='0' revoked='0' delegate='0'>
  <Issuer>
    <UID name='User.IssuerName' />
  </Issuer>
  <Subject>
    <UID name='User.SubjectName' />
  </Subject>
  <Object>
    <UID prefix='Community.Name' name='Role.Name' />
  </Object>
  <Proofs>
    <Attribute name='name' type=''></Attribute>
  </Proofs>
  <Signatures>
    <signature> '..._owner_...' </signature>
    <signature> '..._subject_...' </signature>
  </Signatures>
</Delegation>

```

CODE 4.17 – Delegation granting enrollment in a role. The Subject cannot further delegate this role if the delegate field is equal to 0.

As the presence of a delegation states that a subject belongs to a role, the number of members of a community can be determined by the unique set of subjects that have a delegation for the most basic role (e.g., Member), without the expire flag active. The existence of *Delegation* instances can be validated by others correlating the *Proof* attributes presented in the *Delegation* and the conditions for enrollment.

Membership information can also be used to enforce separation of rights, by stating that role *A* can only be delegated to subjects that do not belong to role *B*. The opposite can also be defined, by stating that, e.g., role *A* requires subjects that are also entitled to role *B*. Code 4.18 illustrates separation of rights for the role *IT.ResearchMember*. The role requires previous enrollment in a role named *Member* but forbids enrollment if the user is enrolled to a role named *Offender*.

```

<Role prefix='IT' name='ResearchMember'>
  ...
  <Conditions>
    <Condition name='IsMember' operator='='>
      <Attribute name='Role' type='String'>Member</Attribute>
    </Condition>
    <Condition name='IsNotOffender' operator='!='>
      <Attribute name='Role' type='String'>Offender</Attribute>
    </Condition>
  </Conditions>
  ...
  <signatures>
    <signature> '..._owner_...' </signature>
    <signature> '..._other_user_...' </signature>
  </signatures>
</Role>

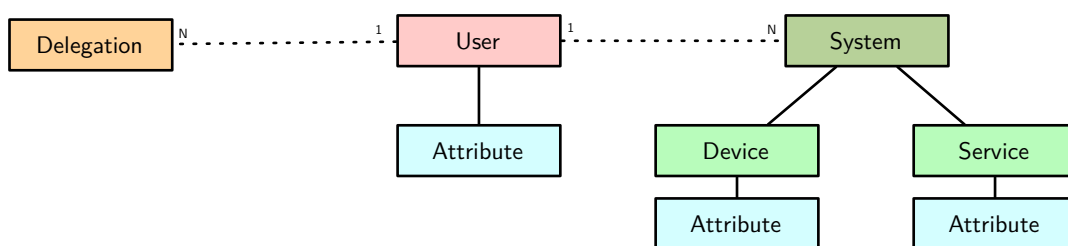
```

CODE 4.18 – Separation of rights and role dependency for enrollment.

4.2.3 MANAGED ELEMENTS CONCEPTS

4.2.3.1 USER

One of the main composed concepts is *User*, as this concept represents participating humans, and provides a root concept for other objects to relate to. While the *User* concept is rather simple, as it only contains identification attributes, all other objects are always tied to a particular *User*, by means of the *CMObject Owner* field. Therefore it is always possible to identify whom instantiated an object, and then search other user related information elements.

FIGURE 4.8 – Concept composition diagram in relation to the *User* concept.

The *User* derives directly from the *CMObject* concept and is further composed by a

list of core (well defined) *Attribute* fields, and list of optional *Attribute* fields. The list of core attributes is comprised by: *FullName*, *Email*, *Address*, *Phone Contact*, and *PublicKey* (see Code 4.19). These fields are the typically identification attributes of an individual and are used to roughly identify the individual by other individuals. They are present but their owners may leave the contents unfilled. The concept composition tree associated with the *User* concept is depicted in Figure 4.8.

```

User[
  :CMObject
  CoreAttributes{ => Collection % Set of Attributes
    FullName      => String  % User full name
    Email         => String  % Email address
    Address       => String  % Postal address
    PhoneContact  => String  % Phone contact
  }

  OptionalAttributes{} => Collection %set of Attributes
  Certificate          => String  % User crypto certificate (PEM)
]

```

CODE 4.19 – Description of the *User* concept.

From all the fields composing the *User* concept, the most relevant is the *User Name* (inherited from the *CMFObject* concept) as it is used to establish ownership relations between objects that were created. That is, it is expected that all objects created have a clear identification to some user, through the addition of the *User Name* field into the *CMObject Owner* field. This way, it becomes possible to trace the creation and update of every component of the policy. The *Certificate* is presented to other members in order to validate signatures made by each user. If cryptographic verifications are required, the *Certificate* attribute must be set. Moreover, the use of a specific Certification Authority (CA) [202] may be required in a particular context.

Additional attributes are optional and could further describe a given individual. As depicted in Code 4.20 the additional attributes enable users to provide a list of interests to others. Also, a policy can also require a specific *Attribute* field to be present, which users must provide in their profile either as core as an optional attribute.

```

<User name='John' created='0' updated='0' owner='John'>
  <Attributes>
    <!-- Core Attributes -->
    <Attribute name='FullName' type='String'></Attribute>
    <Attribute name='Email' type='String'></Attribute>
    <Attribute name='Address' type='String'></Attribute>
    <Attribute name='PhoneContact' type='String'></Attribute>

    <!-- Optional Attributes -->
    <Attribute name='Interest' type='String'>Music</Attribute>

  </Attributes>

  <Certificate> '...' </Certificate>
  <Signature> '...' </Signature>
</User>

```

CODE 4.20 – Instantiation of the *User* concept, defining an additional *Attribute*.

4.2.3.2 SYSTEM

In their normal activities when participating in a standard (non digital) community, users interact directly. In the digital representation of their communities users must interact through electronic systems, here represented by the *System* concept. This concept represents an IEEE 802.11 Access Point, an ADSL router, a laptop or any other electronic equipment that is owned by a user, and interacts in the community (see Code 4.21).

```

System[
  :CObject

  Attributes{}    => Collection % Set of Attributes
  Devices{}       => Collection % Set of Devices
  Services{}      => Collection % Set of Services
]

```

CODE 4.21 – Description of the *System* concept.

From the perspective of participation in a community, this concept is composed by a set of attributes (see *Attribute* concept in Section 4.2.1.2), devices (see the *Device* concept in Section 4.2.3.3) and provides a set of services (see the *Service* concept in Section 4.2.3.4). That is, the *System* concept contains hardware devices such as a VIA CPU, 4GB RAM, a 802.11 Wireless Card and a 200Gb Hard Disk, and can provide services to others such as a File Storage or Sharing, Packet Forwarding capabilities, or Internet Gateway capabilities. It also

exposes some attributes. Such information is presented to others and can be used to shape the community policy. As the remaining concepts, the *System* concept derives from *CMObject*.

Every instantiation of the *System* concept must be related to an *User* through the *Owner* field. If that *User* participates in a *Community* the equipment will also be part of the *Community*. This is important as the obligations and rights delegated to the *User* will be valid to the *System*.

4.2.3.3 DEVICE

The *System* concept is considered a container for instantiations of the *Device* concepts. Considering a laptop, which is an instantiation of the *System* concept, the network cards, storage mediums, or keyboard are all instantiations of the *Device* concept. The *Device* concept thus provides a representation of all discrete components of a computational system and actually describe the system functionality and capabilities, from the point of view of the underlying hardware.

Devices derive from *CMObject*, and extend it by adding a *DeviceType*, specifying the family of the *Device* (e.g., *Processor*, *NetworkInterface*, *Storage*), and a list of *Attribute* fields (see Code 4.22). *Attribute* fields present in devices can be used to enforce rights and obligations through the policy, by reading a device status or issuing configurations. Only the *State* attribute is present at the level of the *Device* concept, and it represents the operational state of the device.

```
Device[
  :CMObject

  DeviceType    => Enum      % Type of device
  Attributes{
    State       => Integer   % Operational State
  }
]
```

CODE 4.22 – Description of the *Device* concept.

Other concepts may be derived from the *Device* concept, providing an accurate representation of specific hardware devices. Moreover, each instantiation of the concept may provide a custom list of attributes. As an example, one can consider that a *NetworkInterface*, which derives from *Device*, will provide information regarding its communication medium such as bitrate or SNR, while a will a *Processor* provide information regarding its cache size, frequency and instruction set. All are instantiations of the *Device* concept, but provide different attributes due to their different purpose and internal mechanisms.

Under the scope of community management for (wireless) networks several devices types are considered. However, any other can be later defined as required for each particular scenario.

- **Processor:** Represents a processing unit such as a CPU or a Graphics Processing Unit (GPU). It may be further derived into other concepts further specializing the management interface provided.
- **NetworkInterface:** Represents a network device such as an Ethernet Network Interface Card (NIC), or ADSL modem. Each of these sub-types will be further specialized in order to provide higher manageability.
- **StorageDevice:** Represents any permanent storage device such as an Hard Disk, or Flash memory.

4.2.3.4 SERVICE

The *Service* concept is the representation of applications providing a some discrete functionality, or an entity providing an higher layer (social) service. The concept is similar to the *Device* concept, however it applies to services provided by entities, and not to hardware resources. Internally, a *System* may have many instantiations of this concept. The *Service* concept applies to discrete applications providing a service to others, such as an FTP, HTTP Server or Messaging server, and also to abstract services such as Storage.

```
Service[
  :CObject

  ServiceType    => Enum      % Type of Service
  Attributes{
    State        => Integer   % Operational State
  }
]
```

CODE 4.23 – Description of the *Service* concept.

It should be noticed that this concept is not expected to expose the service interface to other services, as in a Service Oriented Architecture (SOA) approach. Instead, it only exposes a simple management interface of the service (see Code 4.23), that is, its status, so that the service can be started and stopped.

```

<System name='myLaptop' created='0' updated='0' owner='John'>
  <Attributes>
    <Attribute name='State' type='Int'>1</Attribute>
    <Attribute name='Location' type='Location'>Aveiro</Attribute>
  </Attributes>

  <Devices>
    <Processor type='Generic' name='CPU0'>
      <Attributes>
        <Attribute name='State' type='Int'>1</Attribute>
        <Attribute name='CPUFrequency' type='Int'>1000000</Attribute>
      </Attributes>
    </Processor>

    <NetworkInterface type='IEEE80211' name='Wifi0'>
      <Attributes>
        <Attribute name='State' type='Int'>1</Attribute>
        <Attribute name='Channel' type='Int'>6</Attribute>
        . . .
      </Attributes>
    </NetworkInterface>
    . . .
  </Devices>

  <Services>
    <NetworkService type='Forward' name='NSForward'>
      <Attribute name='State' type='Int'>1</Attribute>
    </NetworkService>
    . . .
  </Services>

  <Signature>'...'</Signature>
</User>

```

CODE 4.24 – Instantiation of the *System* concept, with inner *Device*, *Attribute*, and *Service* instantiations.

Also, internal functionality of the communication stack is exposed as services, and available to the management policy. In this sense, it is considered the existence of a child concept named *NetworkService*, with higher specialization towards management of the network stack. Several instantiations closer to the network stack are envisioned:

- *Route*: Exposes an interface to the routing functionality further provided by a routing daemon.
- *Forward*: Exposes the packet forwarding functionality of the communication stack to the policy.
- *Filter*: Exposes filtering functionality to the policy. It allows the definition of packet

filters allowing or forbidding types of traffic.

- *Shape*: Exposes QoS functionality to the policy. It allows the control of the packet prioritization processes supported by the communication stack.
- *Gateway*: Exposes the functionality of interconnecting an internal network to the Internet, potentially applying translation mechanisms such as NAT.
- *NameResolution*: Exposes the functionality of translating network addresses to the identity of users, and vice versa. Similar to DNS but applied to community information.

Code 4.24 describes a subset of the instantiation of the *System*, *Device* and *Service* concepts. The system represented belongs to user *John* and has one IEEE 802.11g interface, operating in channel 6. Also, it supports forwarding of packets.

4.3 COMMUNITY MANAGEMENT FRAMEWORK

The Community Management Framework (CMF) is proposed as a novel framework enabling distributed management of groups of individuals and their devices, according to the properties of a community of interest. The purpose of CMF is to provide a common set of mechanisms for the creation, distribution and manipulation of the community structure and knowledge. The relations between participants, which are clearly stated and available to other in the form of the community policy, determines the community structure and, as in nature, is highly dynamic. Together with operational data and the policy statements, the structure of these relations composes the community knowledge and is also available to all participants. This prototype instantiation makes use of the concepts described in Section 4.1 and in particular, implements most of the language described in Section 4.2.

While, at its core, the CMF is a PBM system following a distributed RBAC approach, it improves the current state of the art by proposing a system tailored towards distributed, user centric, dynamic communities where users have social ties and interact mainly over a (wireless) network communication medium. Moreover, in this scope, there is a clear association between a device and a user, so that all devices will have rights and obligations inherited from its owner roles. Also, systems can belong to several different policy domains (communities) during their existence, even if simultaneously. Traditional systems are focused in the perspective of the system operator, and the owner is an entity (or individual) managing a campus, a company, or an office.

The entire design and implementation of the prototype was focused towards wireless communities of interest. Therefore, it relies in concepts of distributed storage and P2P

systems. An explicit integration of network functionality and access control mechanisms was also followed.

4.3.1 ARCHITECTURE AND MODULES

The CMF architecture follows a decentralized and service oriented approach with components grouped in several abstract layers, and exchanging data through well-known interfaces. Because of this native modularity, it is possible to add or modify modules according to a particular scenario. Components exist at different layers according to the purpose of the component. Four of these layers are currently devised: the Storage layer is dedicated to storage of policy and community information; the Management layer dedicated to management of roles, rules, communities, other related to objects and its events; the Interface layer provides interfaces with the operating system, users, or services; The Communication layer provides communication between instances running at different hosts. A registration service (Blackboard) is also part of the CMF and is common to all layers.

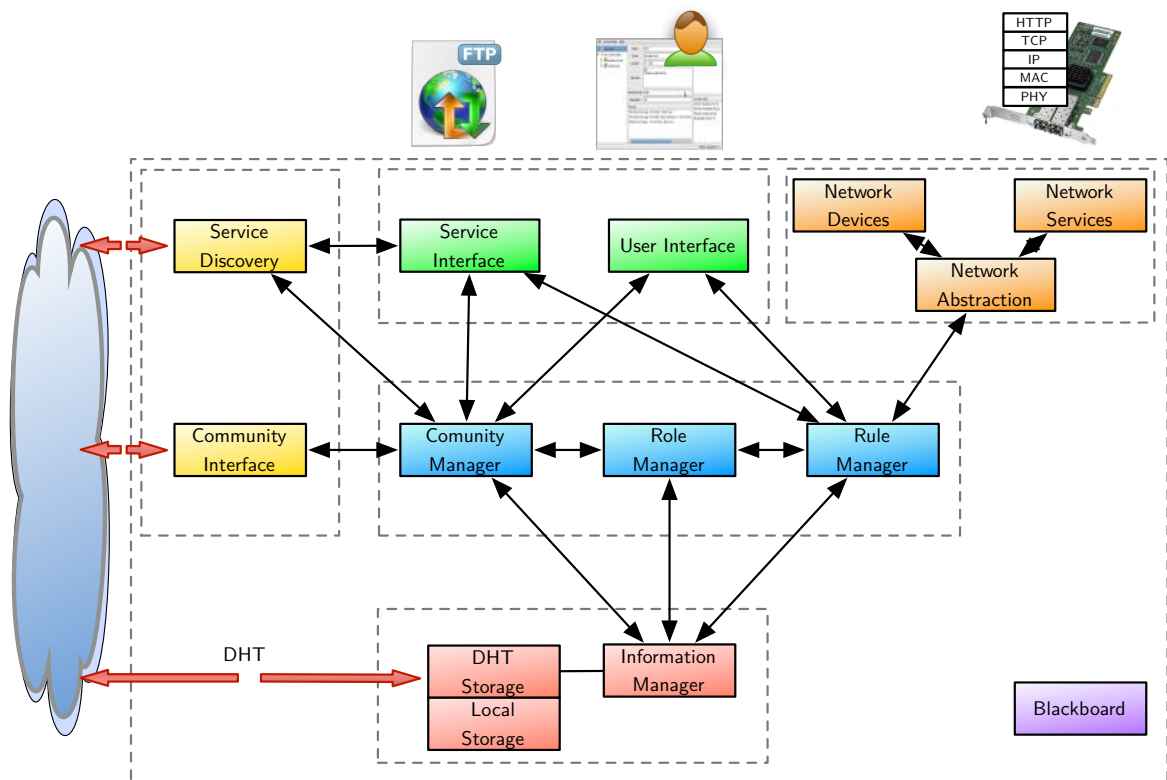


FIGURE 4.9 – Component architecture of the CMF prototype.

With the exception of the Management layer, which contains the core policy processing components, the remaining layers can be composed by an arbitrary number of components with very different behavior, as long as they provide the basic functionality required at that layer (interfacing, storage, communication). The structure described and the associated relationships is depicted in Figure 4.9 and each component is described in the remaining of this section.

4.3.1.1 COMMUNITY MANAGER

The Community Manager component keeps track of the communities available to the current host, and from this list, it identifies which communities are effectively active for the actual user. Therefore, two types of communities are considered: Active Communities and Inactive Communities. The list of available communities comprises all communities known, active plus inactive.

Active communities are those where the user is entitled to participate, as asserted by the existence of a *Delegation* for a role of that community. As shown below, this is asserted by the existence of a specific information element. Belonging to a community may provide some rights, but may also enforce some obligations. Therefore, Active communities are actively tracked to changes in their policy and membership status.

Inactive communities exist when a device has knowledge of some community, but no membership can be asserted. These communities are important as they may be nevertheless required by some active policy element. As an example, a role may state the right to access a service to members of a specific community, which is not the local active community. The Community Manager will consider the first community as active, and will list the second as inactive.

Besides keeping a list of all known communities, and their status, the Community Manager also tracks for changes in the policy of active communities. When a change in the policy is found, and if the current user is effected by the changes, they are pushed to the remaining modules. The process may involve removing currently active rules or roles and reconfigure the remaining modules accordingly.

Upon starting, the Community Manager will scan the local database for any community present. If a delegation is also found, the community is marked as active and the respective policy is loaded. Otherwise, only a reference to the community is added and the entry remains as inactive.

When users create a new community, this module will also initialize the base roles for the community, with names: Member, Leader and Creator. The owner is automatically entitled

to all the Creator and Member roles. The Leader role is created with a condition limiting access to users enrolled in the Member role. Access to the Creator role is forbidden in this setup, however the owner of the community may modify this skeleton and provide access to other members.

4.3.1.2 ROLE MANAGER

The Role Manager component is the core of the PEP inside each instance of the CMF. This component stores all known delegation objects as well as the respective proofs and roles. When a new role is activated by the Community Manager, this module will instruct the remaining modules to act according to the rules and attributes presented. The system behavior is modified by dispatching the *Attribute* objects to the device, service, interface or module responsible for it. Therefore, when a user enrolls into a role, some parameters will be set immediately, unconditionally. Attributes found inside rules will only be activated on a specific event. This information is sent to the Rule Manager module.

The Role Manager is also responsible for creating delegations to other users by the current user. When the user instructs the framework to issue a new delegation, or a remote user requests a new delegation, the Role Manager processes the existing data and, if possible, provides the requested delegation. If the role being requested specifies the existence of conditions for enrollment (constraints) this module will refuse to create the delegation and instead reply that some conditions are not met. The actual attributes required are never specified before enrollment.

4.3.1.3 RULE MANAGER

When a user enrolls to a specific role, it must comply with this subset of the community policy. Therefore, all his devices will need to apply the attributes and rules from the moment its owner enters the new role. The Rule Manager module will keep track of all currently active rules. It will either track the parameters specified in the rule conditions or periodically sample the remaining subsystems for the attributes. When a condition is met, the rule action is triggered.

The Rule Manager module also has the possibility of publishing the result of condition being checked. Rules can specify if the value of the monitored attribute is to be published to other members of the community, and in which cases it should be published: when the condition is met, when the condition is not met, or in all cases.

As an optimization, the Rule Manager module will only process rules that use valid attributes. If a specified attribute is not found, (which may happen if the system lacks a

service, or a device), the rule is temporarily disabled until the set of services available or devices attached is modified.

4.3.1.4 BLACKBOARD

The architecture proposed makes use of components which may freely exchange messages through known programming interfaces. In order for components to exchange messages they must actually know what other components are available. The Blackboard module provides a register service where components may register and announce to others. Therefore, the Blackboard component is the only which must be known at development time. Another reason to adopt a model using a Blackboard is that it allows the broadcast of messages between components, as it acts as an interconnection bus. Moreover, components can communicate with any other component in a pattern that was not determined at design time.

Blackboard systems frequently are used in applications focused towards learning, expert systems, and using artificial intelligence patterns [203], relying in concepts such as knowledge sources, shared repository or shared suggestions. In the case of the CMF, one can actually see the existence of a local Blackboard interconnecting all components, and a shared storage and distribution network interconnecting members. However, the concepts from knowledge based systems (such as iterative or collaborative construction of results) are left unexplored as they constitute a research area by their own.

4.3.1.5 NETWORK ABSTRACTION LAYER

Under the context of self-organized, community aware communication, the Network Abstraction Layer (NAL) is the interface with the networking subsystem present at each node. For this, the CMF resorts to a layer interfacing the management components with the networking stack. The result is that individual functionality of the network stack may be available, and have a representation into the policy, becoming also capable of being managed.

The NAL appears to the remaining framework as an independent module which handles a wide amount of attributes, and further provides additional components to the system in the form of network interfaces and network services. However, from the perspective of attribute handling, the NAL is just a proxy to the actual sub-components. When the NAL is initiated, it scans the current system for all network interfaces and network services, and instantiates a representation of these services and interfaces to the management system. Each service or interface will provide a set of attributes which the NAL automatically exports to the system. Currently provided services are: *Route*, *Filter*, *Forward*, *Gateway*, *Shape* and *Resolve*.

The *Route* service provides a representation of the routing process running in the system.

Most often, the *Route* service will exist in the form of a more specific instantiation such as *OLSRRoute*, *AODVRoute*, or *StaticRoute*. Each will interface with the specific routing software enabled in the system, respectively `olrd`, `aodvd` and the kernel routing tables. Therefore it becomes possible to modify how the routing process operates. One example would be to change the routing cost taking in consideration the role of the next hop.

The *Forward* module represents the forwarding of packets at the communication stack, and will either allow or forbid packet forwarding all together. No further action is performed by this module. However, enforcing forwarding is vital for the operation of multi-hop relay networks, which motivated the instantiation of this specific service.

The *Filter* module has two functions: capture packets for other services, and filter packets based on rules. At its core, the *Filter* module intercepts all packets for the purpose of deciding if a packet is to be allowed or not. This decision is based on the existence of communication constrains imposed by a community or role. One example would be the prohibition to provide an FTP service to hosts not owned by members of a community, or members of a specific role. This would result in blocking all traffic related to FTP at the *Filter* module. The *Filter* module thus interacts with the filter framework provided by the operating system (`netfilter` in the case of the Linux OS). Once a packet is captured for inspection, and in order to not degrade performance any further, it may be sent to other modules before a verdict is issued on its fate (i.e., accepted, rejected, or replaced). Upon starting, any service can register to the *Filter* service and express its intention of receiving packets either entering, leaving or being forwarded.

The *Shape* module represents the QoS differentiating capabilities of modern operating systems. It provides support for differentiating packet flows taking in consideration the protocol, port, and role membership of the destination peer. The actual differentiation is done using a Hierarchical Token Bucket (HTB) [204] queuing discipline, but other queuing disciplines may be used. HTB allows differentiation of flows based on arbitrary classes, together with rate limitation, both for individuals flows or aggregates (e.g., external traffic vs community traffic). When initiated, it sets a tree for the purpose of classifying traffic according to its destination. The actual rules in the policy will then manipulate the entries in the branches of the tree, adding filters matching the particular traffic of a node. The initial structure created is depicted in Figure 4.10. Two main classes of traffic are considered: Internal and External. The first matches traffic internal to the community, while the second matches traffic with outside peers. Then, for each class, traffic can further be classified as High Priority, Normal Priority, or Low Priority. Best Effort Traffic will fall in the Normal Priority category.

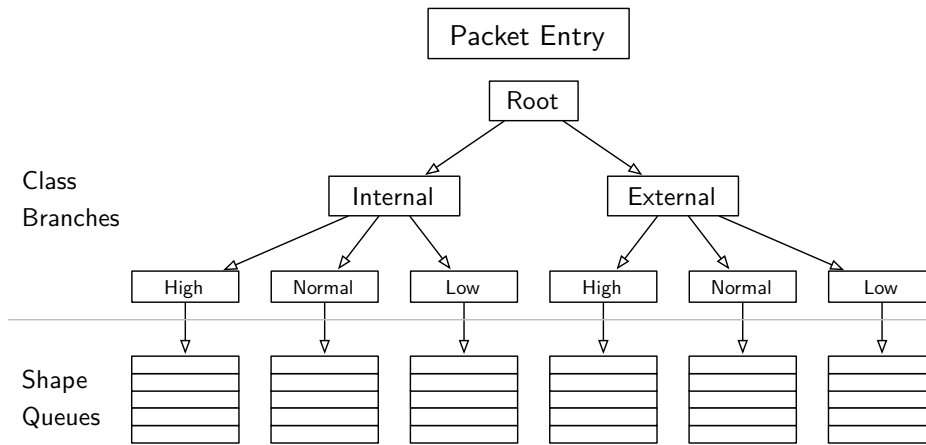


FIGURE 4.10 – Tree structure create by the *Shape* module.

The *Gateway* module is present when the current host allows interconnection of different networks. A typical example would be an ADSL router in a neighborhood wireless community which provides access to the Internet. It is currently used to provide information regarding the traffic exchanged with foreign networks.

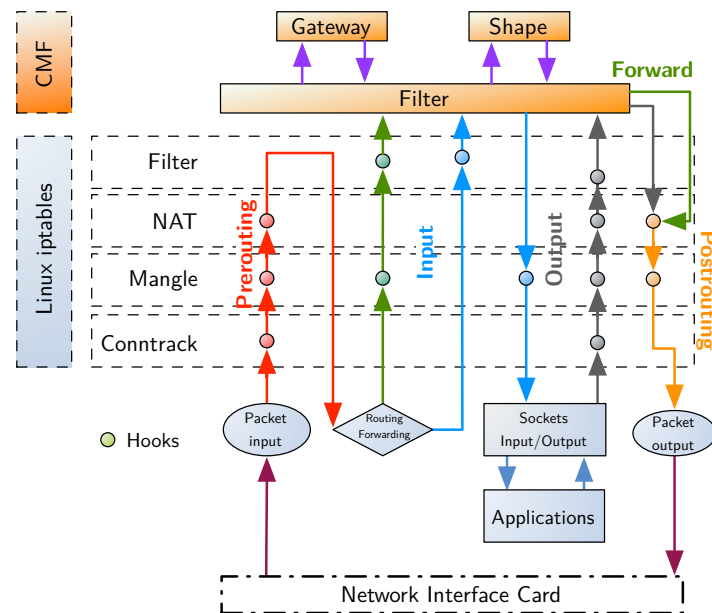


FIGURE 4.11 – Packet flow inside the Linux netfilter architecture when using the *Filter*, *Gateway* and *Shape* modules.

One important metric which is also provided by the *Gateway* module is the actual cost (if

any) of using the current device for communication with a foreign network. This cost is not related to the engineering cost of routing packets, but to the amount of money charged by the service provider. This is calculated from the information provided by users in the CMF configuration. Figure 4.11 depicts how the different modules fit together, and which hooks are set in the networking stack instantiation developed for Linux.

The *Resolution* module is similar to the DNS resolution mechanisms provided to applications, with the difference that it provides information regarding members of the community. By using this service, other services are able to quickly determine who is the owner of the device to which a packet originates or to whom it is destined. Also, it allows to get information about which roles and communities the user belongs (if any user is found). The service relies in distributed storage, providing an interface based on querying information related to IPv4 addresses. The process of getting all the roles of the destination of a packet will imply several steps. First, the common storage is queried for a *System* with a *NetworkInterface* containing that address. Secondly, taking in consideration the owner of the *System* object returned, the storage is queried for the list of delegations available regarding that user. The two steps allow to rapidly identify the destination user as well as the characteristics of the equipment, and list of all its roles in all communities known to the local system. The *Filter* and *Shape* services use this information to apply policy rules.

In order to reduce overhead, and improve performance, the information is cached for a configurable amount of time (see Figure 4.12). It should be noticed that in Linux, only Layer 3 packets can be observed by the *Filter* module due to limitations in the `netfilter` implementation⁶.

No packet authentication mechanism was developed. Therefore, the system will be vulnerable to impersonation attacks based on IPv4 spoofing, like in other insecure network. Mechanisms such as Internet Protocol Security (IPsec) [205] can be deployed to mitigate this issue, but that is a policy configuration issue.

⁶An alternative would be to use `ebtables`, however this mechanisms doesn't provide the required methods of control.

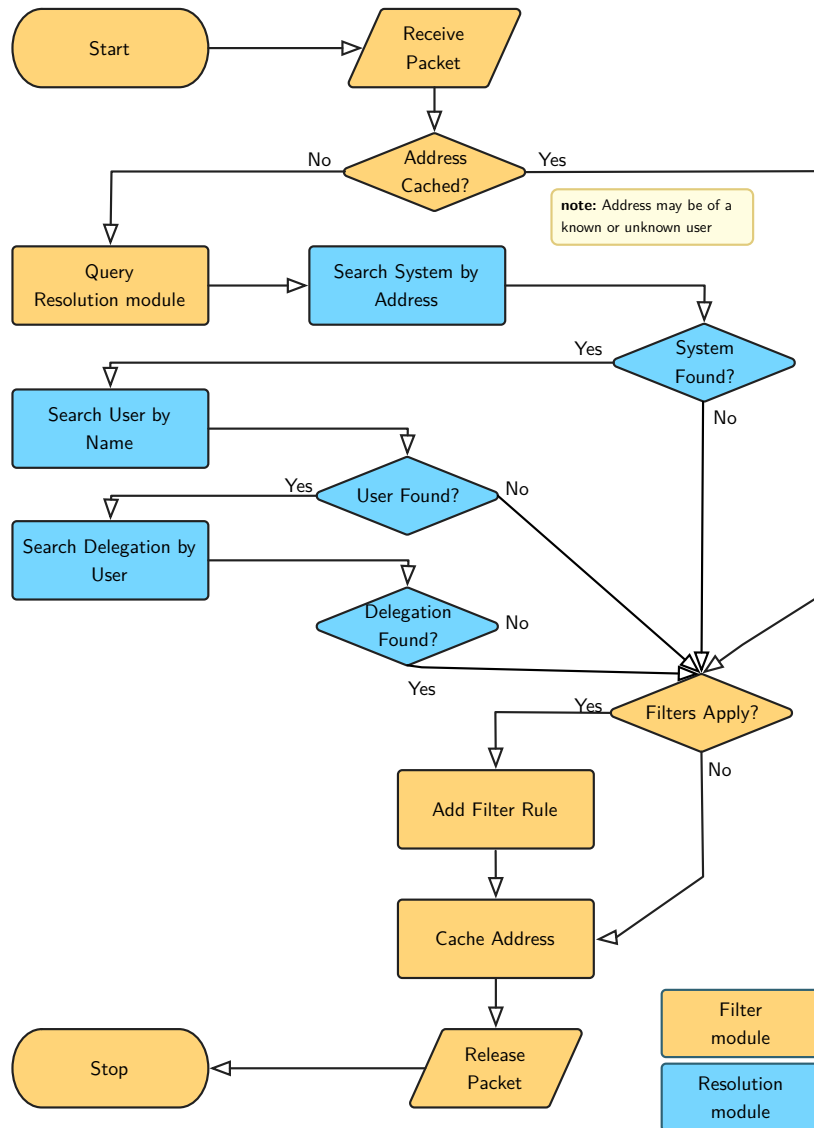


FIGURE 4.12 – Process for determining the ownership of a packet and applying filter rules.

4.3.1.6 INTERFACES

Several interfaces provide interconnection of the management system with services, users, and between instances running at different systems. Interconnection can be provided by means of a dedicated Application Programming Interface (API) or through the exchange of messages across the network.

4.3.1.7 COMMUNITY INTERFACE

Communication between different instances of the CMF, when considering neighbor peers, is achieved through the *Community Interface*. While the *InformationManager* and *Storage* components provide the means for members of a community to store and access knowledge, the *Community Interface* provides the means for neighbor peers to announce their communities, validate the requirements for membership and then join communities. In the current prototype, the only method for communication is through IPv4 multicast and unicast in a local network, however other publish and negotiation methods could be used.

All systems, having one or more communities active, will announce, to its neighbors in the network, the port for the UDP port used by the *Storage* component and the list of communities active. Announcements are sent periodically to the network using UDP packets, destined to a multicast address. Code 4.25 depicts the format of an *ANNOUNCE* packet, announcing the existence of a community *C*, which was last modified in the date represented by the timestamp indicated.

```
ANNOUNCE
PORT 5501
COMMUNITY C:1325376000 C1:1323376000
```

CODE 4.25 – ANNOUNCE message informing neighbors about the existence of communities *C* and *C1*.

When receiving an *ANNOUNCE* message, it is possible to know that a community may be available for joining. Also, because the *ANNOUNCE* message carries a timestamp, it is also possible to assert if the current policy requires to be updated, by comparing the timestamp present in the *ANNOUNCE* message with the timestamp of the community object currently active at the local system. The result is faster dissemination of changes and the ability to easily monitor changes in the policy.

When a User wishes to join a community, it should trigger a join process. The objective is to obtain a *Delegation* object for the *Member* role of the community. For this purpose it should send a *JOIN* message to any member of the community. If the user has a system in the same network and one or more communities are active, it will receive the *ANNOUNCE* messages sent by others and can act on these messages with a *JOIN* message.

JOIN messages express the willingness to participate in a community and carry the identification of the target community, the identification of the user and the communication port used by the *InformationManager*. A typical *JOIN* message is depicted in Code 4.26. As depicted, *JOIN* messages can also include a set of *Attribute* objects with their value. These

objects are used to meet the requirements for enrollment of the *Member* role and will be added to the *Delegation* as proofs of compliance. If the *Attribute* objects or its values do not meet the requirements for enrollment, no delegation is actually created.

```
JOIN C1
USER B
PORT 5501
<Attributes>
  <Attribute name="NetDev_Ethernet_Count" type="int">2</Attribute>
</Attributes>
```

CODE 4.26 – JOIN message with attributes.

The response to a *JOIN* message is a *STATUS* message. It indicates whether the process was successful or, if it failed, what was the case or failure. The most typical ones being provided as a response to a *JOIN* message are: 0 Success, 10 Access conditions are not met, and 12 Object not found. The first indicates that the process was successful; the second indicates that the user doesn't meet the requirements for enrollment; the third indicates that the community object was not found. Because of the specific code 10, the *STATUS* message will also carry a list of *Attribute* objects to which the candidate user must provide values (see Code 4.27).

```
STATUS 10 Access conditions are not met
<Attributes>
  <Attribute name="NetDev_Ethernet_Count" type="int"></Attribute>
</Attributes>
```

CODE 4.27 – STATUS message with Attributes required for enrollment.

After the node is accepted, it should receive a message with code 0 Success, and the *Delegation* just created. The storage components of the framework are now able to start exchanging information. First, basic connectivity is verified, and communication latency is estimated through a *PING* message (see Section 5.2.2), then the *Delegation* object is signed and stored. The new member is now authorized and the community policy he should enforce will be available to him. The exchange of messages of a join process where membership requirements are imposed, is depicted in Figure 4.13.

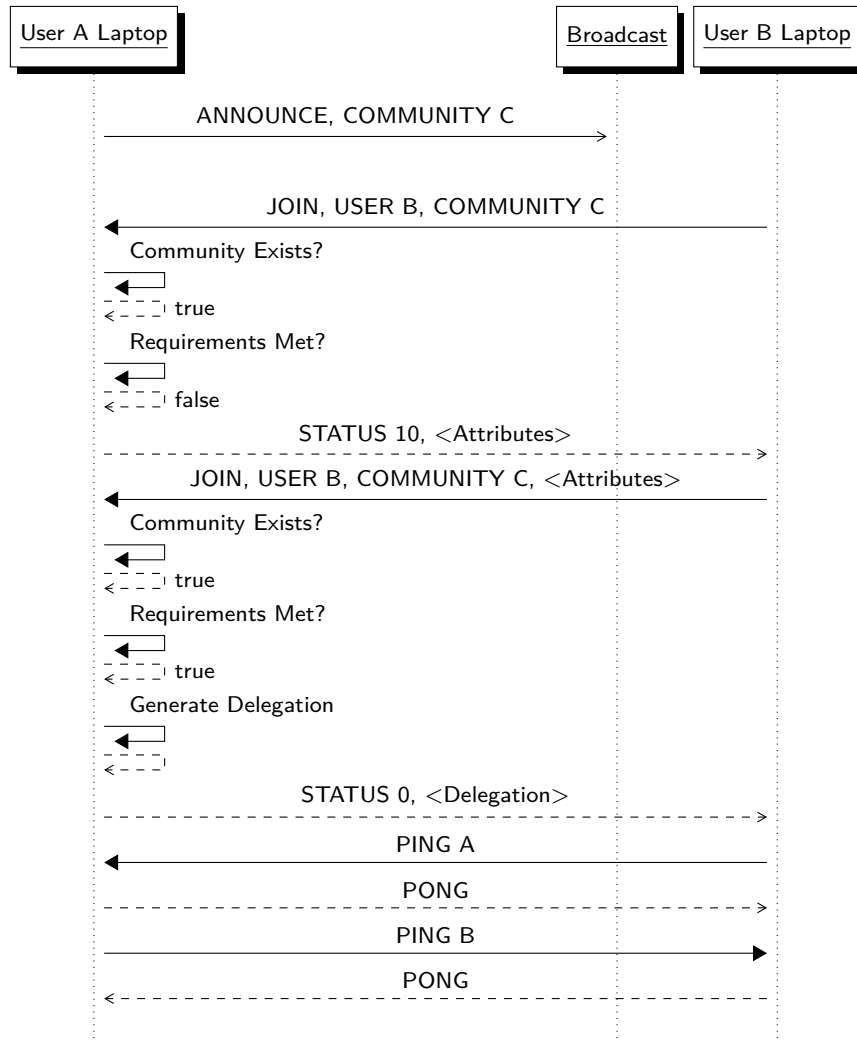


FIGURE 4.13 – Join process with a retry due to missing enrollment requirements.

4.3.1.8 INFORMATION MANAGER

The components at the storage layer provide the primitives to store, load, and search for data objects created by the members of a community. Therefore, the storage provides an out-of-band, indirect, broadcast based, communication channel between participants. Objects created by one user will be available to others, as well as the community policy, which differentiates this layer from a traditional network medium.

In the proposed architecture the primary component of this layer is the *Information Manager*. This component allows management of a private storage and a public storage. The interface made available for the remaining modules follows a *put/get* approach enhanced by *update*, *list by type*, and *search by keyword primitives*, while providing differentiated manage-

ment of one private and one public repositories.

As the component exports the notion of two storage media (private and public), other layers use each storage simply based only on the notion of private and public information. There is no information in the upper layers about how information is actually managed, or even if the storage media are remote or local.

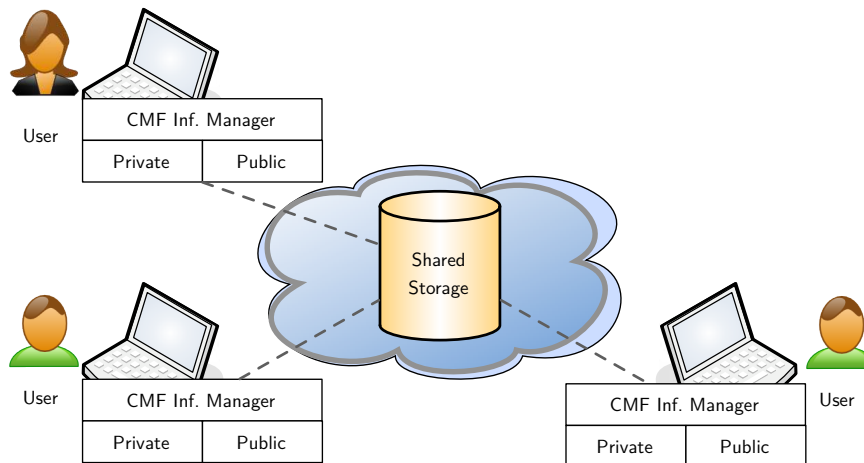


FIGURE 4.14 – High level overview of the dual storage solution.

The private storage is provided in order to store private objects or caching public objects. The public storage is shared by all participants and provides a knowledge repository for all objects published by the community members. Therefore, it is assumed at the level of the Information Manager that the public storage will have higher access penalty, which motivates the use of a local cache. The private repository is based on a hierarchical filesystem, and managed directly by the *Information Management* module, the public repository functionality can be provided by external systems, such as standard network filesystems, databases and directories. Examples of such storage systems that are commonly used by other management systems are Structured Query Language (SQL) databases, directory services supported by Lightweight Directory Access Protocol (LDAP) [206], or other composite service abstracted by a cloud infrastructure. That is, given that connectivity is guaranteed to the storage system, and the infrastructure is available, a large set of different systems can be used (see Figure 4.14).

4.4 SUMMARY

In this chapter it was presented an analysis regarding the impact of communities to the network fabric, and how they can be represented and mapped in an efficient and effective manner. It is also proposed methods for integration of existing solutions that are vital for proper operation of the the Internet, such as the addressing, as well as name resolution using DNS.

A description language is proposed which takes in consideration the resource constrained environment of community networks, composed by user owned SOHO routers connected through wireless links. It is shown how the language can represent several community constructs, and be used for determining membership and role entitlement of members into a community.

Finally, this chapter describes a prototype implementation of a software system which implements part of the language specified previously, its design considerations and modules.

EVALUATING POLICIES IN COMMUNITIES

No work should be properly considered without evaluation of the mechanisms presented. This thesis focused in distributed policies for networked communities, and had a strong focus towards real world use cases. Also included in the work produced for this thesis is the development of a automated experimentation platform, capable of being used for the purpose of evaluating community concepts. This chapter reflect this approach by first describing the evaluation testbed and its management methodology, which is followed by a simple evaluation its operation. A technical description of a prototype implementing the concepts proposed in Chapter 4 is presented, as well as its graphical interface, and the most relevant messages supported. Finally, the prototype is evaluated considering two scenarios which mimic situations found in the real world, in which community concepts and distributed policies present advantages to users.

5.1 SELECTING AN EVALUATION TECHNIQUE

Solutions for (wireless) network environments often start by being first validated in a discrete, event driven simulation environment. After this initial validation by simulation, sometimes a solution can be emulated in a real world prototype and then implemented in a final product (if appropriate). There are many of such simulation platforms, with NS-2 dominating the academia world for a long time, and NS-3 together with OMNET++ [207] appearing in an increasing number of publications [208]. These simulation tools are vital to the advance of the state-of-the-art in most networking areas, because they allow protocol evaluation to be performed in a reasonably controlled platform, unaffected by external variables and thus enabling repeatable experimentation, under very simplistic environments.

Nevertheless, all simulators implement simplified representations of the real world models, such as the underlying characteristics of the transmissions medium, or operating system scheduling effects. These models must be validated and tweaked so that correlation between simulated and real world scenarios is high [209]. However, no model can be perfectly validated: some effects are not linear or too exhaustive to be simulated in real time. The result is that we will always have simplified simulation models of our world, which in turn will always constrain the quality of the results obtained.

With current models and the complexity of existing systems, practice increasingly shows that simulation results should be increasingly taken with a grain of salt, in particular because minor details, which are either unknown or ignored, may result in misleading or incorrect answers. The amount of synthetic validation through simulation increased in modern research, but in many cases without a clear increase in the quality of the results presented [210].

Evaluation of solutions using tools and environments closer to reality, can rapidly exclude many unreal results that while valid in a simulator, are not possible to be ever useful, or require too much optimistic conditions. Therefore, in recent years there has been a major interest in the field of experimental testbeds, in particular involving wireless technologies or large-scale systems.

One of the reasons behind this trend is that both systems are very complex, and are affected by a large number of variables. For example: accurately simulating propagation of radio waves in a office environment, by ray tracing or a similar solution and between a dozen of stations, is so intensive that little processing time is left to the actual simulation [211]. In such cases, specialized hardware is occasionally used [212]. Another reason is that with the constant decrease of device prices, it became financially possible to evaluate solutions in scenarios closer to the real world, by using emulation platforms. With testbeds appearing as a reasonably efficient tool, increased effort in testing, stressing, observing, tuning and validating solutions became a must. For some high profile scientific work oriented towards technical solutions, simulations alone are no longer sufficient to support the results or can cast doubts on the validity of the work. In the light of this trend, several research institutions, enterprises and universities have developed initiatives to build testbeds, aiming at creating a reproducible evaluation environment while providing increased quality of the results.

These initiatives usually have in their core either multi-purpose experimental platforms or simply custom proof-of-concept platforms. The first aim to provide an environment able to evaluate a multitude of solutions and scenarios while proof-of-concept platforms target advancing a particular research objective and exist until the solution reaches market or is abandoned.

The recent history of multi-purpose experimental IEEE 802.11 enabled testbeds starts

in the late 1990s. Ad-hoc networks were on the rise and the MONARCH [213] project was created. It also inspired the creation of larger, and more feature full, experimentation facilities. The Network Research Lab, at University of California - Los Angeles (UCLA), aimed at developing ad-hoc solutions and created a testbed comprised by 20 laptops and 60 PDAs using 802.11b cards. It spans the entire campus and was integrated with a simulation platform able to evaluate hybrid scenarios (simulation plus real world) [214]. Another landmark in the history of testbeds is Grid Roofnet [215], located at the Massachusetts Institute of Technology. It consists of an experimental 802.11b/g network comprised by 20 off-the-shelf nodes providing connectivity to users in Cambridge. Also, the University of California created a wireless testbed aimed at low-level radio research, composed by more than 100 nodes with different capabilities and access technologies. The ORBIT testbed [216], that includes over 400 nodes in a 20 x 20 meter area, is currently one of the most complex wireless testbeds. On a different scale, but also relevant there is the OneLab initiative [217] that interacts with the larger PlanetLab [218], and has extensions to support evaluation of wireless solutions over its general-purpose network infrastructure.

Many other testbeds were created in the past years [219], making use of both physical and virtualized environments with also both real and emulated mobility. A recent initiative aiming at enumerating all experimental testbeds in Europe devoted to Future Internet research, as in 2012 already accounts for 130 of such platforms [220].

In order to obtain results which are closer to reality, and not tied to simulation environments, an experimental approach was taken. These comprised two challenges: the implementation of the CMF, and the development of a test environment where to evaluate this prototype.

5.1.1 TESTBED DESCRIPTION

Due to the experience gained by participating in several other projects with a relevant experimental component, and because of the additional validation provided to this work, it was possible to devise during this PhD a testbed that could fulfill two goals. The primary goal was obviously to provide an environment for evaluating some of the concepts proposed in this thesis in a reproducible, and accurate manner, while taking in consideration real world effects. However, another accessory secondary goal aimed to provide a testbed open to the research community for later use, resulting in a side result from this work. Another outcome of these goals was the development of best practices and advice to others building similar platforms.

The primary goal tackles one of the issues inherent of not using simulation environments:

reproducible and accurate results can be difficult to obtain in experimental network research. This issue is not a flaw of emulation, or testbeds, as measurement limitations are inherent to the real world environment where solutions must ultimately be used. Albeit not completely isolated, a testbed has to be predictable, with small variance in the tests. This could only be obtained in an open environment in which no objects would flow through the node and where radio interference would be set to a very low level (below what could be considered interference). Most testbeds exist in laboratory environments, and are affected by people moving around, have regular electromagnetic interference, and interference from multiple experiences. Work by other authors identified that this environment, in particular people moving, is a major source of variability [221]. Another constraint imposed was that the same experiment has to be repeated several times, so that noise in the result datasets is reduced. Repeating an experiment several times is not a challenge, but keeping all the events synchronized across different runs can become a challenge without specialized software.

The secondary goal arose from the fact that testbeds, providing programmatic capabilities and reproducibility, are still expensive to develop and maintain. Typically, testbeds are used for evaluation and development of a specific solution, and then rearranged to fit the next solution. Its workflow is not clearly defined and they lack accuracy. Most of the existing testbeds are either private, and therefore fully accessible by their owners, or public but provide a limited set of functionality to users. This ultimately means that only a limited set of experiments can be conducted, and some results cannot be reproduced, which is worrisome for research. Keeping the carefully designed testbed that was used in this thesis open to the research community, is a relevant contribution and a way of increasing the impact of the work developed.

According to the before mentioned goals and the objectives of this work, several other aspects had to be taken in consideration when deploying this testbed:

- x86 compatible nodes: Of the several existing computer architectures the most popular among developers is the x86 architecture. This fact is easily realized by the sheer amount of publicly available software implementations in the area of networking, and most research uses network nodes based on x86.
- Support for Multiple Radio interfaces: The focus of the testbed is in creating a heterogeneous wireless testbed, thus potentially covering several wireless technologies, existing and future. Each node must therefore be flexible enough to support different radios. Ultimately this must be translated into mainboards with multiple hardware interfaces, in which the radios can be connected to.
- Access to low-level radio interfaces: In addition to the existence of multiple radios, it is also required that each of the chosen radio interfaces provides programmatic access to low-level aspects such as the MAC (a common research issue). Ultimately the desirable

radio interface should be a fully software-defined radio, but this is not simple to support currently.

- **Extendable:** the nodes processing capabilities must be extendable, and easily connectable to current and future core network solutions. This implies that a smooth interface for the testbed nodes should exist, and that the wireless interfaces of the nodes should be easily accessible from core machines.
- **Low Power and Cost:** The amount of nodes immediately leads to concerns related to power consumption. It is a good practice for each node to consume as less power as possible. To this end, there is a strong concern on the amount of power involved, as well as on the power requirements of the computers and radio interfaces used. Furthermore, it is expected that the testbed should be deployed with low cost Commercial Off-The-Shelf (COTS) devices.

Based on these requirements several hardware-based solutions were evaluated, and a management platform was specified. In the next sections the chosen hardware platform and management platform solutions are detailed, together with the reasons behind each of the choices. The testbed was nicknamed AMazING, from Advanced Mobile networkING playGround.

5.1.1.1 ARCHITECTURE

The key design decision is related with the need to have a clean (in radio terms), predictable environment, in a location of easy access. The solution found was to place the testbed outdoors, in the rooftop of the main building of Instituto de Telecomunicações, which is reasonably insulated from common interference sources, and isolated from interference by human movement. This also had the advantage of providing a reasonable large area ($1200m^2$) for deployment of the testbed. Wireless nodes are distributed across the area, forming a grid with elements 8 meter apart from each other. Figure 5.1 depicts the actual deployment as well as the support communication and computational infrastructure. Wireless nodes are located outside the building, while several support servers are inside. These servers provide: i) the means to host services available to the testbed nodes, ii) programmatic support for specifying and running experiments, iii) storage space to host operating system images and experiment results.

At the core of the testbed, there are several support servers and a redundant storage that serves all files to the testbed. Servers provide processing power to store experiment results and extend the testbed. Also, they configure nodes according to the experiment through the management framework and provide a common time source. Additionally they can be used to run Virtual Machines configured by testbed users in order to support specific experiments.

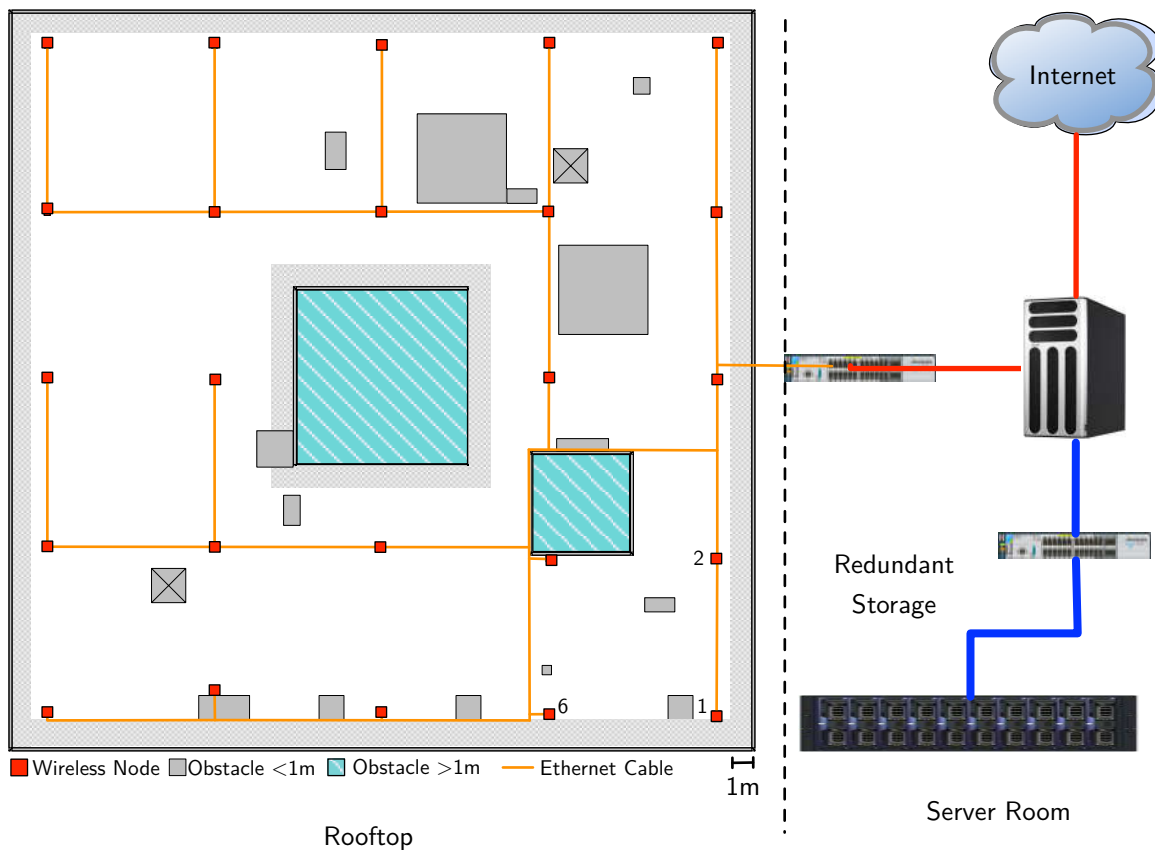


FIGURE 5.1 – System and deployment view of the AMazing testbed.

The redundant storage device provides a 4TiB Network Attached Storage (NAS) in a RAID5 configuration, connected to the servers and nodes using Gigabit Ethernet.

A major issue with the chosen deployment is that nodes are exposed to the weather conditions of a south European coastal region: sunny skies are frequent all around the year but especially from late spring to early autumn, relative humidity is frequently higher than 85%, wind tops at $150\text{km}\cdot\text{h}^{-1}$, and rain is frequent from October to May. Enclosures must be watertight, so that water causes no damage to electrical devices, while still being able dissipate enough heat in days with clear skies and under direct sun light (with corresponding high levels of UV radiation). The final solution devised comprises the several individual aspects, in which the above mentioned requirements were taken in consideration.

Nodes have at its core a Commell LE-365 board, with a VIA EDEN 1GHz CPU and 1GiB of Random Access Memory (RAM). The LE-365 board provides a high number of expansion possibilities such as 8 Universal Serial Bus (USB) [222] ports, a RS-232 port, a 2 ports Serial

ATA (SATA) controller, 2 miniPCI slots and 1 CF interface. This myriad of interfaces makes it possible to easily expand the node interfaces, which is perfect for a multi-radio testbed. New network interfaces can in the future be added to the available interfaces. A Gigabit Ethernet NIC is also available and is used solely for the purpose of managing the nodes and collecting statistics. The availability of a Gigabit Ethernet control network is important due to several aspects: it reduces command delay, supports raw packet collection from multiple nodes, and provides support for NFS with low latency. This interface can also be used for extending the testbed nodes capabilities. The CF interface is occupied by a 4GB CF card while two 802.11 cards occupy the miniPCI interfaces. One of the cards is a Compex WLM54SuperAG, which has an Atheros AR5414 processor at its core. This card can output 20dBm, supporting channel bonding both in the 2.4GHz (802.11bg) and 5GHz (802.11a) ranges. The additional card (Compex WLM200NX) is also Atheros based (AR9220) but besides supporting 802.11, having high sensitivity (Atheros Extended Range), Power Control, and Channel Bonding (40MHz) like the previous card, it also adds support for 802.11n in a 2x2 MIMO spatial multiplexing configuration. The open-source ath5/9k drivers control both cards and provide high flexibility due to almost direct access to the networking ASIC.

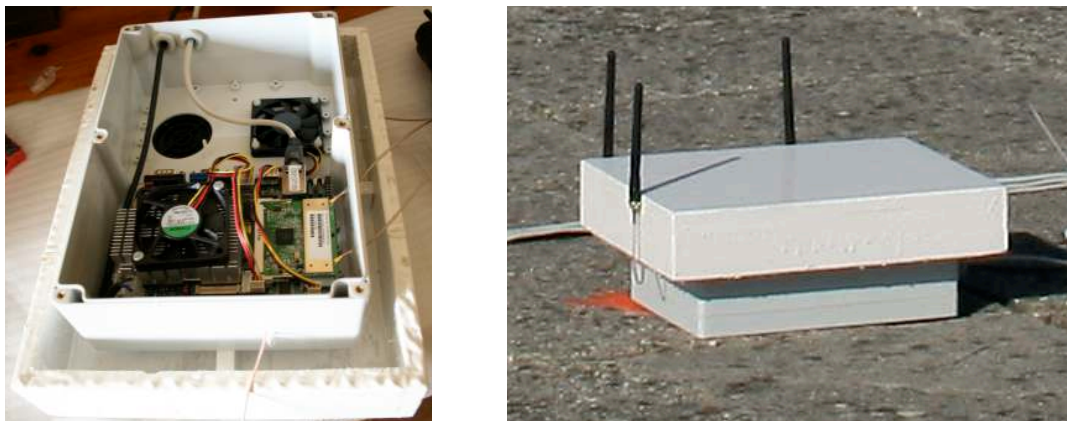


FIGURE 5.2 – Internal (left) and external (right) view of a wireless node.

Enclosures were made of light gray, ultra-violet resistant, IP65 rated [223] polycarbonate. This material is transparent to radio in the 2.4GHz and 5GHz frequencies, while providing water tight, easily manipulated structure for assembling components. The major issue with using polycarbonate is that heat conductivity of this material is very reduced. Enclosures have a total area of 0.22 m^2 and polycarbonate is able to dissipate $0.21 \text{ W}(\text{m}^2\text{K})^{-1}$, which is insufficient for dissipating the 18 watts radiated by the system, while keeping the internal temperature under 70°C . Another issue is solar heat, which further increases the thermal energy in the system in sunny days. Temperature in the Aveiro region can reach more than

40°C during summer time, and sun radiation can add more than 250 W m^{-2} . The solution found (see Figure 5.2) was to add two openings, to the top of the enclosure (protected by particle filters), and force air circulation by using a fan in one of the openings. Also, a custom-built protective hat made of waterproof maritime plywood was added to the top of the enclosure. This hat keeps water in the exterior of the enclosure by protecting the openings, while at the same time, greatly reducing heat gain due to solar radiation.

5.1.1.2 MANAGEMENT INTERFACE

One of the limitations inherent with wireless experimentation facilities is the capability of reproducing experiments, and to repeat the same experiment several times in order to pinpoint odd behavior, or to increase the confidence of the results. In simulation platforms this functionality is easily obtained as the environment is not real.

Several solutions aim at enabling the support of automation and repeatability in experimental testbeds. One of the most popular is the Orbit Management Framework (OMF), that as the name suggests is used to manage the Orbit testbed deployed at the WINLAB, in Rutgers University. OMF automates the entire operation of the testbed.

A scripting language named OMF Experiment Description Language (OEDL) allows researchers to specify the actions for the testbed execute. In each script, researchers may specify the configuration of each communication interface, schedule traffic generators to send specific traffic flows between endpoints, and may even specify which metrics should be collected by the system. Taking in consideration this script, the OMF system also pre-loads the required operating images containing the researchers' software into each of the nodes required for the experiment.

The AMazING testbed makes extensive use of the OMF system, and contributed to the development of the platform by incorporating several improvements. The most relevant is the AMazING panel. OMF presents a command line interface to researchers through which they are able to submit experiments to the platform. If results are to be automatically collected by the framework, a database is created and provided to the researcher which submitted the experiment. While this method is effective, it requires researchers to learn the syntax of the OEDL, and doesn't facilitate reproducibility and incremental experiments. In this sense, the AMazING Panel [32], which has supervised under the scope of this thesis, enables any OMF enabled testbed to be easily controlled, while improving the functionality provided by OMF.

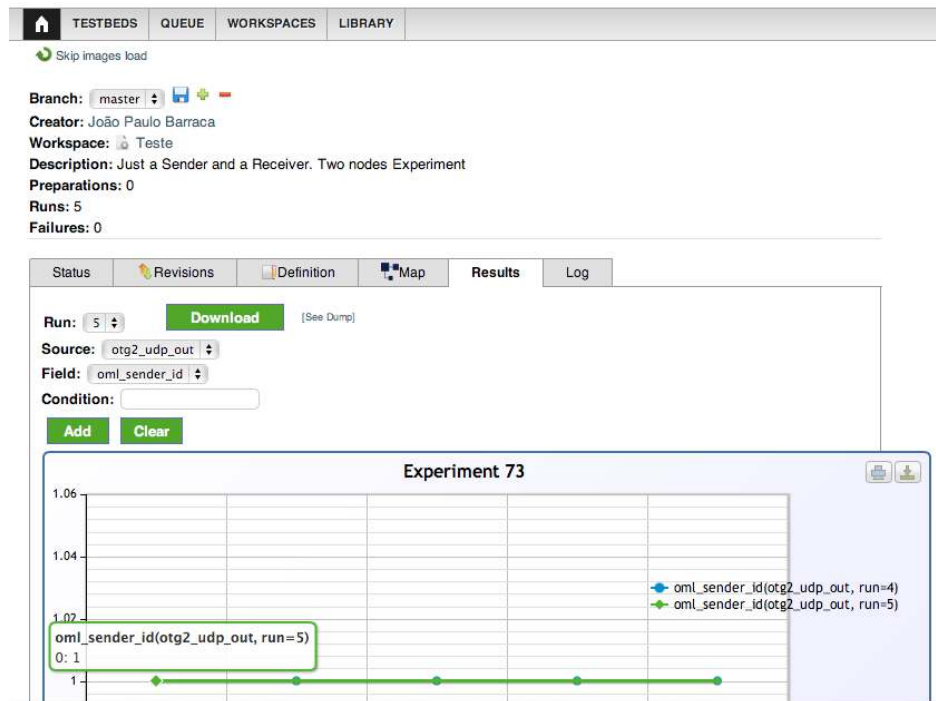


FIGURE 5.3 – The result analysis interface provided by the AMazing Panel.

Supported by web technologies, the AMazing panel provides a web based interface (see Figure 5.3), through which users can interact with the testbed. Many key aspects of the AMazing panel are novel, and some deserve to be highlighted.

- The system allows the creation of workspaces, through which multiple researchers can interact. Each workspace has a series of experiments that can be refined, and scheduled for execution, by any member of the workspace. Therefore it is possible for teachers to create experiments that students can reproduce and analyze, or even change.
- The AMazing panel hides most of the complexity of designing experiments by providing a graphical interface. Through this interface users can specify which nodes to use, which configurations to apply, and which applications to schedule for each node, or to several that are part of a group. The Panel will then automatically create the OEDL describing the experiment. Without this tool, researchers would have to learn the syntax and reserved words of OEDL.
- When experiments are scheduled, researchers have the opportunity to specify how many runs the testbed should execute. Because the testbed is fully automated, it is able to re-create similar scenarios for each of the runs. After the batch is executed, researchers are analyze the results and pinpoint odd results. Moreover, once experiments are

created, they can be reproduced anywhere in time by any other user allowed to access the workspace.

- During the definition of the experiment, researchers are able to specify data sources, and metrics to be collected. While when using OMF researchers are presented with a database containing all results requested, when using the AMazing panel, they are also given the opportunity to quickly view and analyze results through the web interface. Resorting to the latest technologies, it becomes possible to analyze multiple metrics, and to compare their values directly in the application.
- Another relevant aspect is the support for experiment versions. The AMazing panel allows for the continuous refinement of experiments by incremental changes. Experiments can be cloned, slightly modified, and then re-scheduled for execution. Combined with the possibility of analyzing results from multiple experiments, this allows researchers to quickly determine the result of a particular change.

5.1.1.3 TESTBED VALIDATION

In colder days, another problem arises: water condensing in electronic circuits. In order to reduce the danger of water condensing inside the enclosure, the main board was placed in an inverted position and near the top of the enclosure. Heat generated by the electronics (in particular the CPU) always keeps the board above dew point, and accumulation of small amounts of water never touches the electronic components. Figure 5.4 depicts the temperature variation recording for a single node over the span of several days. The values presented are the difference between internal and external temperatures. External temperature was measured using a nearby weather station that is part of the national meteorological weather network.

As part of the initial validation of the testbed it was deployed a measurement device near one of the nodes (Node 1) in order to monitor the level of radio signal over the 2.4GHz band. The device used is an easy to use spectrum analyzer, which while not comparable with laboratory equipments (which are also tens of times more expensive), allows for some basic measurements in the range of -6.3 to -100dBm. The model used has a resolution of 373KHz and sweeps the radio band in the range of 2.400GHz to 2.483GHz. The result of the analysis is depicted in Figure 5.5, representing the level of radio energy received at the given frequency. The two curves compare the level of background radio energy (interference) with the level of a system while transmitting, when Node 1 is acting as an Wi-Fi AP in channel 1. As it is depicted, average signal level on the testbed area is below -92dBm. Peak signal value measured when nodes are active, but with radios idle, is a little higher, reaching -80dBm. If compared with the situation of nodes transmitting in channel 1 (2.412GHz) it can be seen the difference between the signal power of the nodes (at 20dBm) and the environment peak noise

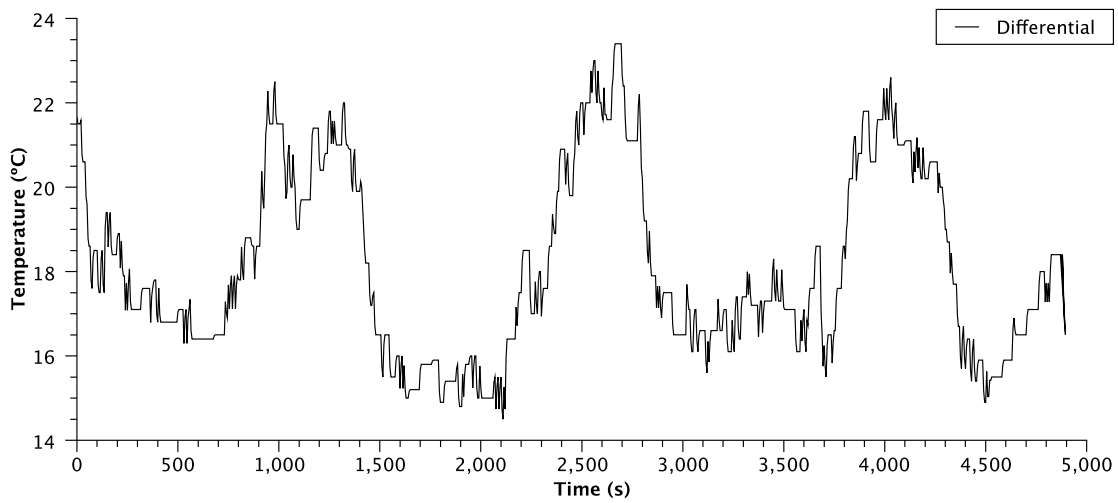


FIGURE 5.4 – Temperature variation over several days with nodes operating.

values.

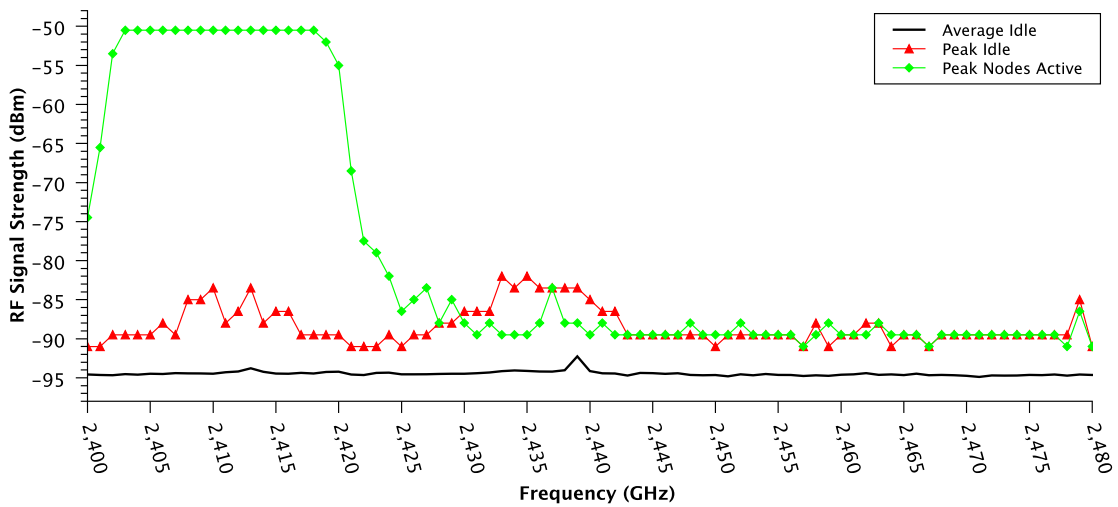


FIGURE 5.5 – Radio level with and without nodes operating.

While Node 1 is configured as an Access Point on channel 1, another experiment was also conducted in order to determine how radio signal and throughput varied with distance along one of the sides of the testbed. For this purpose, Node 1 received TCP flows from Nodes 2, 3, 4 and 5 with the duration of 1 minute. Average throughput values were recorded at

intervals of 10s. All these nodes are located in the right side of the building, actually forming a line, each node distancing 8m from the previous. Node 5 is 34m from Node 1. As depicted in Figure 5.6, neighbor nodes are able to sustain a throughput of 23Mbps/s when using IEEE 802.11g. This value decreases, as expected, with the distance as the receiving power also decreases. The farthest node (Node 5) is able to reach Node 1 and exchange data at less than 4Mbps/s. Interestingly, when considering IEEE 802.11b, rates drop to less than 6Mbps, but values do not decrease with increasing distance until 34 meters. Node 5 is able to exchange data with Node 1 at 6Mbps when using 802.11b, which is higher than when using 802.11g. According to the specification of the Atheros chip, 20dBm of SNR are required for a link rate of 11Mbit, indicating that noise level at the testbed location is indeed very low.

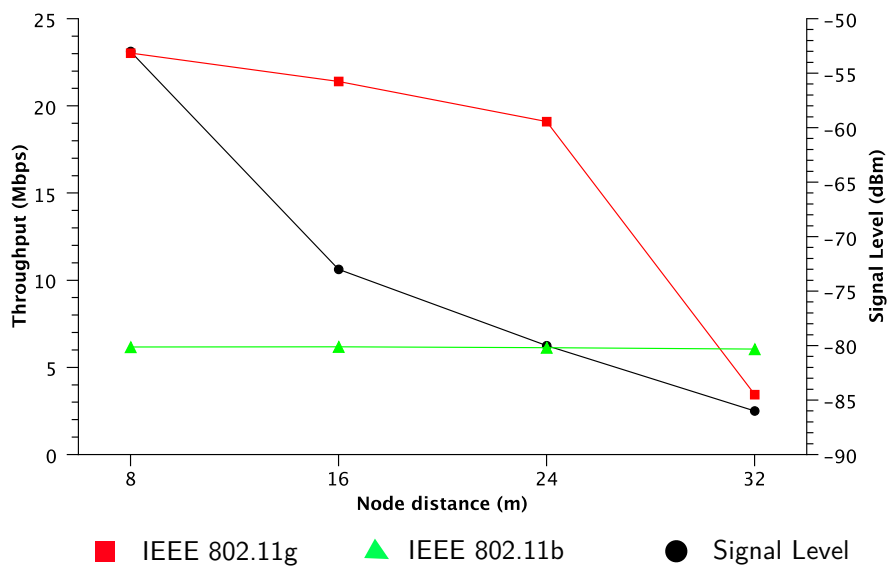


FIGURE 5.6 – Throughput and SNR observed as the distance between nodes increases.

At 8m, Node 2 indicates a receiving power of -55dBm while at 32 meters, Node 5 indicates that the signal generated by Node 1 is received with -86dBm. Considering free space propagation of an omnidirectional antenna, at 32 meters the receiving power should be higher than it is, which suggests additional attenuation. In this case attenuation is intentional and is provoked mainly by the occlusion of the Fresnel zones, which was achieved by placing antennas very near the floor at about 25cm high.

Considering the radius F of the Fresnel zone as given by Equation 5.1,

Frequency	8 m	16 m	24 m	32 m
2.4GHz	0.5	0.71	0.87	1.00
5.0GHz	0.35	0.49	0.60	0.69

TABLE 5.1 – Radius of the first Fresnel zones for different distances.

$$F = \sqrt{\frac{n \times \lambda \times d^2}{2 \times d}} \quad (5.1)$$

where n is number of the Fresnel zone, d is the distance between end-points and λ the wavelength of the signal. The maximum radius of the 1st Fresnel zone, has the values present in Equation 5.1.1.3.

As depicted, even at close range (8m) and for the 5GHz band, the 1st Fresnel zone intercepts the rooftop at a large extent (antenna elevation is 25cm). The minimum occlusion will be of 29% for the node at 8m when using the 5GHz band, while the maximum occlusion is of 85% for nodes at 32m when using the 2.4GHz band. The resulting effect is additional radio attenuation. In the case of the AMazING testbed, this helps by exacerbating the distance between nodes, thus allowing the creation of custom topologies where nodes have no, or only minimized, direct radio reachability.

5.2 CMF PROTOTYPE IMPLEMENTATION

The architecture described in Section 4.3 has been instantiated in a prototype so that the concepts presented could be evaluated, and refined. This work has a strong focus in providing novel concepts but with proved usefulness. Therefore, experimentation was a key aspect that could not be ignored. Moreover, it was considered that devices running a CMF instance would be resource constrained. Examples of the envisioned devices are the Wi-Fi APs commonly available in a common store. Because of the wide range of operating systems deployed in SOHO products, and because some of these operating systems are not publicly available, it was further decided to only consider the Linux operating system. From this analysis a short, but important list of requirements for the prototype was devised:

1. Be able to run on devices with low computational resources. SOHO routers have integrated low processors with less than 500MHz.
2. Be able to run on a wide variety of different processor, namely: MIPS, ARM and X86.
3. Consume very small amounts of resident memory.
4. Have no dependencies with software libraries not available in embedded environments.

5. Require little storage capacity available at most devices.

The requirements related with low computational capacity and low memory usage clearly excluded most popular frameworks such as Java, .Net, Python or Ruby. While Java has achieved a great success in smart phones, SOHO routers are much more memory constrained than a typical smart phone. C and C++ presented as viable candidates, and C++ was chosen for the implementation of the CMF prototype. This choice was made due to the base capabilities of the language, as well as due to the availability of libraries such as the Boost C++ Libraries [224]. Given the fact that C++ is object oriented, the concepts devised could also be mapped almost directly into C++ classes. It should be pointed that the core CMF instantiation is suited for networked devices, and provides no user interface. However, it presented a service oriented interface through which a graphical interface could interact with the prototype.

Implementing the architecture proposed involved following a fully asynchronous operational model. This approach was chosen due to the existence of a core layer processing policy logic, and several other interfaces capable of generating requests. Each interface had its own internal logic and state, running on a separate thread. This allowed interfaces to keep internal timers which were required for retransmission of messages, periodic message advertisement or for the purpose of detecting timeouts. Messages received from interfaces of the CMF are queued and dispatched accordingly. However, because there is no central entry point to serialize and queue requests, nor central locking mechanisms, messages arriving from different interfaces can be processed simultaneously, or even in reverse order. Each module provided an interface to others, and the Blackboard provided the appropriate communication channel and message routing functionalities. Locking mechanisms, internal to each component, assures that from the perspective of each individual component, calls were processed as independent transactions. It is responsibility of the callee to undo any operation if the transaction failed in some part. Still, two calls from independent interfaces, to the same module doest not result in failure of the second. The first call is processed as expected, and the second is hold back until the first releases the internal structures. Then, the second call is processed.

Besides interfaces, other components also keep active threads, namely: *Storage*, *RuleManager*, *NAL* and most other network modules. The *Storage* module must continuously process lookups from other CMF instances running in neighbor devices. The *RuleManager* component keeps active rules and executes their actions periodically or after a monitored aspect is detected. Like described previously, many parameters do not produce events when their value change (e.g., number of packets sent through a network interface), therefore, the *RuleManager* must continuously poll the all required values (which will imply querying attributes from

other components). The *NAL* component, as well as other modules interfacing with network mechanisms (e.g., *Filter*) must also continuously be actively listening to network events, querying important parameters, or waiting for requests from their counterparts applications (e.g., the routing daemon).

An additional, yet vital module was developed in order to enable the architecture proposed, the DHT. While the CMF considers the existence of the *Information Management*, which contains two separate repositories, how the remote repository is kept is scenario dependent. In the scenarios envisioned, the appropriate model is the ones used by DHT solutions. Therefore, a DHT was implemented s integrated with the *Information Management* module, enabling it to distribute information elements to other nodes. Due to its complexity, the actual implementation of the *Information Manager* module is described in Section 5.2.1, while the implementation of the *Distributed Storage* is described in Section 5.2.2.

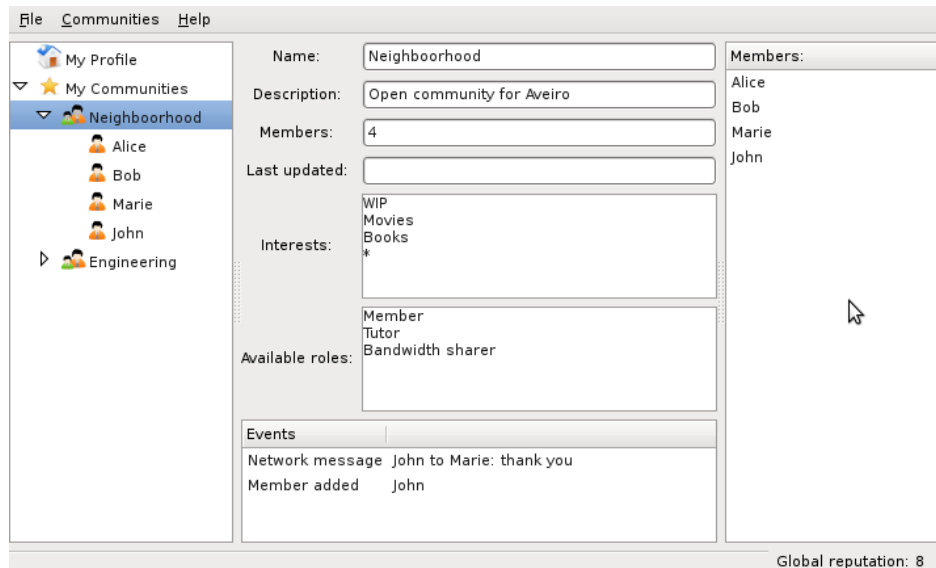


FIGURE 5.7 – Graphical User Interface developed.

A simple implementation of a graphical interface was developed for demonstration purpose of the WIP project [141]. Its development was more limited than the CMF, but it provided a usable interface for preliminary debugging and development of the CMF. The user interface is envisioned to run on laptops and desktops, and remotely control the CMF instance running on the local system, or in user-owned remote systems. By following this approach, CMF instances running on SOHO routers could still be configured by their owners. The requirements for the user interface are much different from the ones applied to the core CMF. In reality, the only requirement is that it should run on a wide range of

desktop platforms, namely: Microsoft Windows, Linux and Apple OSX. Therefore, the .Net framework was used. It has native support in Windows, and can execute under Mono [225] on the remaining environments. Figure 5.7 depicts the graphical interface running on the Linux environment.

A configuration setting allows to configure the credentials for accessing the user interface at the CMF instance. The current prototype supports two users, with the intention of having read-only permissions, while the second having read-write permissions. In this context, write permission relates to the capability of creating new communities, editing rules, and issue delegations (if allowed by the policy). Before the user interface connects to the CMF instance it must complete an authentication process. This process is started by the client which sends an AUTH message indicating the username to use. The CMF will reply with a 128bit random nonce. The client repeats the process by providing the username, and a digest computed over the concatenation of the username, nonce and the password. The CMF is able to validate the request and will only provide access through the user interface after the authentication process is complete.

5.2.1 INFORMATION MANAGER IMPLEMENTATION

The *Information Manager* module is responsible for data coherence and data dissemination among the different participants. When an object is requested, this component will request it from the actual storage medium, taking in consideration the provided key. In the prototype developed, the key is obtained by computing a digest (MD5) over the *Object Path*, *Object Parent*, *Object Type* and *Object Name*, as described previously.

Disconnected operation may render the local cache out of sync, and may present to the local system an inadequate representation of the policy. Therefore, this component will always try to have the latest version of the objects updated. If a new version of an object is found in one of the storage media, the system will consider the one provided directly by the object owner. If the owner cannot be contacted, the system will use the one that is valid and has higher modification timestamp. Validity is accessed by determining if the owner of the object, or the user that modified it, were entitled to do it.

Each peer keeps a list of known neighbors, and these peers are used when forwarding messages or issuing queries. Keeping this list up to date is vital, especially in respect to which peers are reachable and what is the communication latency. Each peer issues *PING* lookups periodically, to all the other peers in its contact list. Upon reception of a *PING* lookup a response is issued and this exchange is useful in order to detect which contacts are still alive. Peers may also add the contact of other peers to these messages. Following this approach,

knowledge about available peers propagates faster and requests are routed with the smaller number of hops. The drawback will be higher processing when creating and receiving *PING* and *PONG* lookups, and higher memory usage due to peer tables.

In wireless networks, bandwidth is a constrained resource. Therefore, in order to reduce bandwidth consumption, and to improve scalability, the reception of any message from a peer will reset the time of the next periodic *PING* lookup to that peer. The result is that in a network with reasonable activity, *PING* lookups will be seldom sent, as any message exchange will update a peer status, and its communication latency. The limitation of this approach is that contacts will not be disseminated rapidly to a relevant number of peers. Code 5.1 depicts *PING* and *PONG* messages with contact dissemination.

```
-- Request --  
PING  
SRC c59caa66d661caaa6a1e2f2167a1d50d  
DST 467d8120dcbf25c9a01be979213bb815  
ID b31ab81a7faefb93b97a8ed75b77c8b1  
Contact 4cc37394d571b0dd1e1ac5c51939f634 192.168.0.212  
Contact d32e91fd1eda65a17129bb56d1a4084b 192.168.0.232\n\n-- Response --  
PONG  
SRC 467d8120dcbf25c9a01be979213bb815  
DST c59caa66d661caaa6a1e2f2167a1d50d  
ID b31ab81a7faefb93b97a8ed75b77c8b1  
Contact ff580bbf975ca2c003fc2cf5f6976538 192.168.0.1\n\n
```

CODE 5.1 – PING and PONG messages.

Peers will also use *PING* lookups to detect colliding identifiers, and take this action both at start time and periodically. The duplicate identification detection process is executed by initiating a *PING* lookup to their own identifier. If a response is received, two peers have the same identification, and a new random identification is chosen. This mechanism mimics other solutions such as the Duplicate Address Detection (DAD) [226] functionality of Internet Protocol version 6 (IPv6) [227].

In order to reduce bandwidth, messages sent to the network are compressed in a transparent and opportunistic manner. Compression is transparent because upper layer applications (in this case, the *InformationManager* component) will be unaware of how messages are coded before transmission to the network. Opportunistic because messages are only sent compressed if the compressed representation results in a smaller number of bytes than the uncompressed representation.

Compressed messages are created by compressing the clear text message, and concatenating the result buffer with a four byte header. As depicted in Figure 5.8 the first byte is named *CID* and identifies the compression algorithm, while the next three bytes specify the size of the compressed message (compressed messages are limited to $2^{(3*8)}$ bytes). Because the communication protocol is text based, and according to the ASCII table, text characters always have integer representations higher than 32, if the first byte is below 32 the message will be assumed as being compressed. The only value currently defined for *CID* is 0, and signals that the message was compressed with the DEFLATE algorithm [228], although 31 other compression schemes can be defined and used.

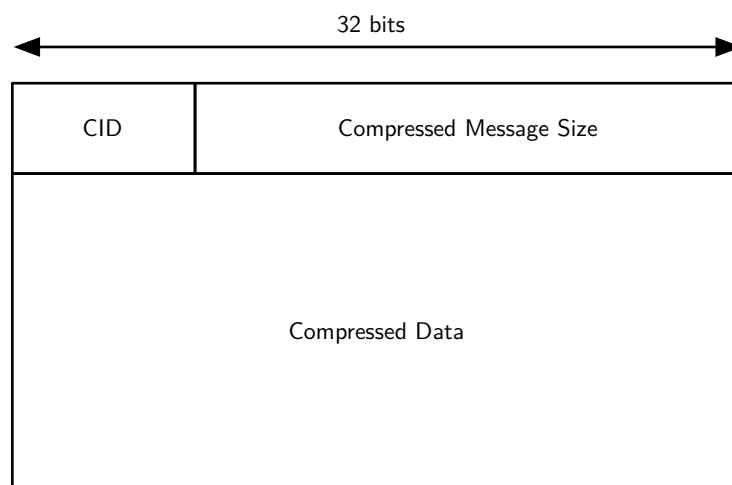


FIGURE 5.8 – Structure of a compressed QUERY message.

Several actions are supported by the *Information Manager* component: Getting an object content, Getting metadata regarding an object, Storing an object, Searching for an object and Listing all objects of a given type (e.g., list all *User* objects, all *Delegation* objects or all specific *RuleResult* objects).

Frequently, it is required to verify the correctness of the local cache, or if a given object was updated by any other peer. Because objects containing policies can be long, and in order to reduce overhead, peers can either get remote objects, or get only their metadata. *STAT* messages are sent to other peers with the ID of the object as parameter. If the object was found, the *STAT* message returns a *ObjectInfo* token with information regarding the object. Code 5.2 depicts the messages sent for requesting the status of community *C* and the response issued. The *ObjectInfo* states that *User.A* owns the community, as well as the modification lock. In this case, to the requesting user the community policy is read only (lockMode="R").

```

-- Request --
ID 066e221f4f5d4e1a0712913c0bb157a7
STAT

-- Response --
ID 066e221f4f5d4e1a0712913c0bb157a7
STAT
<OI s="Community.C" c="1325376000" u="1325376000" o="User.A" l="User.A" lm="R"/>

```

CODE 5.2 – STAT Request message regarding an object with ID (MD5("Community.C")).

Getting the content of an object results in searching the local cache for the specified key, and if the object is still up to date, return it from cache. The cache verification is supported by sending a *STAT* message, and if required, a *GET* message. If the object is not present in the local cache, only *GET* messages will be sent, as no local version exists as term of comparison.

```

-- Request --
ID 066e221f4f5d4e1a0712913c0bb157a7
GET

-- Response --
ID 066e221f4f5d4e1a0712913c0bb157a7
GET
<Community name="C" create="1325376000" update="1325376000" owner="User.A">
. . .
</Community>

```

CODE 5.3 – GET request message regarding an object with ID (MD5("Community.C")).

Upon reception of a request with a *GET* message (see Code 5.3), peers will look in its local database for the object being requested. If the object is found, a copy is returned immediately by sending a *QUERY_RESPONSE* message with the *GET* and the XML representation of the object in the payload. If the object is not found, the peer will order its contact list in relation to the distance to the *Object ID* requested. Peers at the start of the list will present lower distance to the key (using the XOR metric). After sorting, peers will forward the request to a fixed number of neighbors for which the distance between the *Object ID* and the Peer ID is smaller than the distance between the *Object ID* and its own ID.

A value of 5 peers is currently used in the prototype implementation. Higher number of peers will result in faster lookups and higher probability of finding an object at the cost of also higher overhead. If no peer is active or no known peer is closer to the *Object ID*, an

ERROR message is sent back.

Storing objects on the *Information Manager* is supported by the *STORE* action. Objects can be stored either in the private or public storage. If the object already exists, the permission to change it will be verified, and actual storage may be denied. Permissions are also verified if the object was modified by other user besides the owner. Upon reception of a *STORE* message, peers will check the validity of the request and store the object. As depicted in Code 5.4, peers reply with an empty *STORE* message. An *ERROR* message can be also sent towards the sender if the sender is not allowed to modify the object, which can occur if the permissions of the current policy impose limits to the modification of objects (e.g., through the *Permission* concept).

```
-- Request --
ID 066e221f4f5d4e1a0712913c0bb157a7
STORE
<Community name="C" create="1325376000" update="1325376000" owner="User.A">
. . .
</Community>\n\n
-- Response --
ID 066e221f4f5d4e1a0712913c0bb157a7
STORE\n\n
```

CODE 5.4 – *STORE* request message regarding an object with ID (MD5("Community.C")).

When being initiated, the *InformationManager* at each system will execute a series of *STORE* actions regarding their objects. These actions have the purpose of storing locally stored objects in the remote node with closest ID, and to replicate local data to other remote nodes. In the case of a hardware malfunction, or other event leading to disconnection, the information available at the local system will exist replicated in other neighbor systems.

Dissemination of objects for redundancy purposes can typically follow two approaches: the destination requests a neighbor which objects he may store, or this neighbor can proactively suggest the destination node to store an object. The second approach was chosen because it avoids the need for peers to always pool their neighbors for the list of objects they could store, and then issuing individual *GET* messages. Following this approach, a peer that holds an object may decide which neighbors are eligible to also store the object, and neighbors will have no way to know what objects are available without querying all possible keys (e.g., brute forcing keys by generating all possible 128 bits and issuing one *GET* per value). Therefore, peers will observe their list of neighbors, and distribute objects to

neighbors that are close to each *Object ID*. Like in *GET* lookups, before transfer, the sender will verify that the neighbor has an out of date (using a *STAT* message) or simply has no information about the object, and only transfer the object if this is true.

Another functionality provided by the *InformationManager* component consists in searching for objects according to set of search terms. In order to simplify development, search terms consist of a regular expression which is applied to the XML representation of the objects, enhanced by the possibility of searching by object Type. Search tokens are enclosed by parenthesis and the characters `&&` and `||` respectively denote the operators AND and OR. A *LIST* message is issued to all neighbors of a peer, and these neighbors further forward the request to all their neighbors. Peers keep a history of ID values for all *LIST* messages received and this field is used to avoid routing loops. If the peer already replied to the query (e.g., in response to other neighbor) it issues an empty *LIST* message. Otherwise, it provides the objects that match the query. Each peer will be responsible for aggregating responses from their neighbors and provide a single answer to the requester. The identification of the object (calculated by its path, type and name), and the change date is used to decide which object to keep, and which to drop.

```
-- Request --
ID 4c1c0bc2b349e203f3f5ab90e7397762
LIST
(type=Delegation)&&(name="B:[a-zA-Z]*")&&(prefix='Community.C')\n\n

-- Response --
ID 4c1c0bc2b349e203f3f5ab90e7397762
LIST
<Delegation prefix="Community.C" name="B:Member" created="0" updated="0" owner="User.A" start="0"
  expire="0" revoked="0" delegate="0">
. . .
</Delegation>
<Delegation prefix="Community.C" name="B:Student" created="0" updated="0" owner="User.A" start="0"
  expire="0" revoked="0" delegate="0">
. . .
</Delegation>
.
.
.
\n\n
```

CODE 5.5 – LIST request message regarding Delegation tokens of User B in Community C.

5.2.2 DISTRIBUTED STORAGE

Dynamic ad-hoc communities can appear spontaneously. Therefore they may lack the required storage and communication infrastructure, or the infrastructure may be unable to provide the required level of connectivity and reliability. In order to fulfill this aspect, a P2P decentralized system based on a DHT, using Key Based Routing, is proposed as one of the public storage mechanisms for the *Information Management* module, and this was the solution implemented in the CMF prototype.

Following this approach, each instance of the CMF generates a logical address, created in the same way as the *Object Identifiers* but based on a random number ¹, and will track the location of objects that have an identifier close to their logical address. A simple approach was used for storing objects and maintaining integrity of the system: when objects are created, they are sent to the node with smaller distance to the key. This node will take control of the object, and replicate it to his neighbors in the key space. If this node disappears, the one closer to the key will take control of the objects. If a new node appears with smaller distance, all objects for which he is responsible will be sent to him. Other dynamic partitioning approaches such as the one proposed in [86], could also be used.

A basic aspect of searching the key space is calculation of the distance between two keys. The CMF prototype uses the bitwise *XOR* logical operation calculated over the *Object ID* and the *Peer ID*, as also proposed by solutions such as Kademlia [230]. The reason for using this logical operation instead of a standard absolute subtraction is due to the dimension of the key. As these keys are greater than 128bits, this operation is cheaper to compute. Moreover, it shares some interest mathematical properties, such as:

$$a \oplus b = 0, \text{ if } a = b \quad (5.2)$$

$$\forall a, b, c \in \mathbb{R} : a < b < c \Rightarrow a \oplus b < a \oplus c \quad (5.3)$$

The first property makes it possible to match a specific identifier by comparing if the result between the *XOR* of the query identifier and target identifier is 0 (all bits of the result are 0). In this aspect, the property is similar to subtracting two values. The other property is named the triangle inequality, and states that if the distance between *A* and *B* is smaller than the distance between *A* and *C*, the *XOR* distance between *A* and *B* will be smaller than the *XOR* between *A* and *C*. Therefore, if the result of the *XOR* operation is smaller, so will be

¹This identifier can also be created by computing a digest over a description of the hardware present. While this would work for most situations, if some hardware malfunction imposes replacement of a device, the value would change, thus not making it permanent. For other methods of generating an identifier, please see [229].

the logical distance between two keys. This can in fact be used to route packets towards a destination. Figure 5.9 depicts the organization of a DHT with several nodes. Node *A* knows *B*, *G*, *P* and *U*. If he wishes to send a message to *J*, it will sent it to the closest neighbor (*G*), which then forwards the message until it reaches *J*.

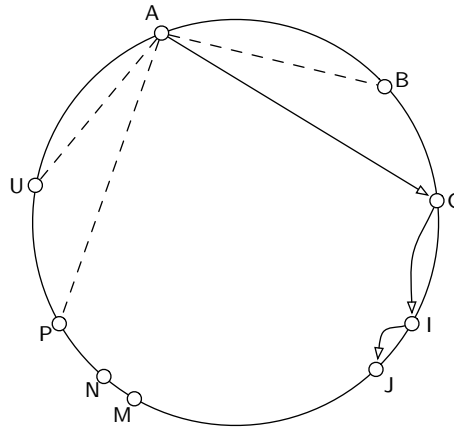


FIGURE 5.9 – XOR supported, key based routing in a DHT network.

Communicating through the P2P network involves creating a query and pushing this query to neighbor nodes. The developed prototype uses directed flooding that forwards a request only to nodes closer (in terms of the *XOR* distance metric) to the object identifier being requested. The resulting P2P network remains unstructured, thus reducing processing requirements. As a consequence, object lookup will be considered as finished only after a predetermined timeout or after all peers issue a result (either success or fail).

Messages are considered to be composed by two parts. The first part identifies the key, type of request and communication end-points. A second part is application dependent and contains the actual data. This second part should be commenced by a command, and then application dependent data.

Queries are text based messages composed by a fixed number of fields, namely: *Query Type*, *Source ID*, *Destination ID*, and *Lookup ID*. Two consecutive End-of-Line characters (ASCII Line Feed, code $0x0A$) delimit the end of a message. All lookups need to be acknowledged by the corresponding peer. Lack of acknowledgment will be interpreted as either the corresponding peer is inactive or some other communication problem exist. The *Source ID* (SRC) identifies the originator of the message, while the *Destination ID* (DST) identifies the destination peer for this lookup. The reserved identifier 0 matches any destination and can be used to provide network wide flooding (broadcast). The *Lookup ID* identifies the lookup and is equal to the object being queried, or a random number if the request refers to no

object. When forwarding a message, peers will limit repetitive forward of messages from the same source, to the same destination, and regarding the same identification. The actual time window used is configurable, the prototype implementation considers the default value of 30 seconds.

Several values for Query Type are considered: *Query Request* (QREQ), *Query Response* (QRES), *Store Request* (SREQ), *Store Response* (SRES), *Ping Request* (PING), *Ping Response* (PONG). Code 5.6 describes a lookup regarding *Community.C* sent from *User.A* to *User.B*. The information present at this level is only used for routing purposes and the final object to be acted upon. The actual action to be performed on the object is not known to this component as it is application dependent.

```

QREQ
SRC c59caa66d661caaa6a1e2f2167a1d50d
DST 467d8120dcbf25c9a01be979213bb815
ID 066e221f4f5d4e1a0712913c0bb157a7
COMMAND
... Remaining IM data here ...

```

CODE 5.6 – QUERY Request message regarding an object with ID (MD5("Community.C")).

Query Request lookups are started when applications wish to obtain information regarding a given object, or the actual object content. It corresponds to the invocation of a procedure with parameters, without any other data being pushed. The destination peer should respond with a *Query Response*, containing the information requested.

When the applications wishes to push data into peers, it should invoke a *Store Request* lookup. This process is the symmetric to the *Query Request*, in the sense that the *Store Request* will carry data to be pushed to remote peers, while the *Store Response* will provide only information about the success of the process.

Ping Request messages and *Ping Response* messages allow peers to keep track of their neighbors are described previously.

5.3 EVALUATION AND RESULTS

The CMF is a generic framework with interfaces to be used by services and other applications wishing to integrate community aspects. Many different scenarios would be required to proper evaluate all possible use cases. In this thesis, two key evaluation scenarios were considered. The first evaluates some of the capabilities of the framework when providing

distributed authorization to services. To further demonstrate the reach of the framework, a legacy service (FTP) is considered. The second scenario considers a WMN with two gateways, and evaluates how a simple policy can be used to change routing cost according to the cost being charged to the available gateways. The result is that traffic can be balanced and reduce the total cost being charged to the community.

5.3.1 DISTRIBUTED SERVICE AUTHORIZATION

Services active in each system have a representation into the policy by means of *Attribute* objects which can be manipulated. Configuration settings can be set when a community is activated or when a rule is triggered. Also, services may restrict its access according to the permissions stated in the community policy. A clear example consists in restricting access to the members of the community, so that access by external users is forbidden. Internal users, or members of a particular role will be allowed to access the service, or will have enhanced QoS.

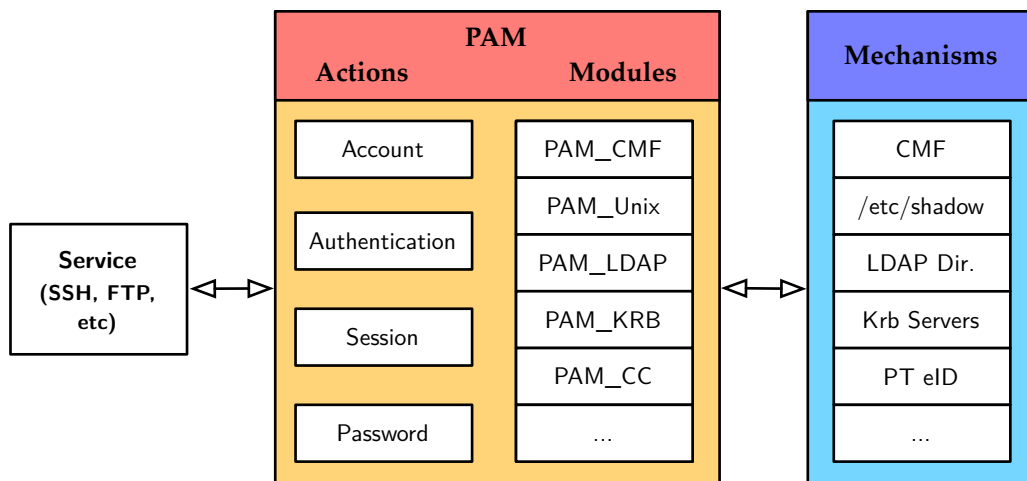


FIGURE 5.10 – PAM architecture depicting services, PAM internals and mechanisms (e.g. CMF).

For the purpose of evaluating the capability for distributed service authorization, an integration with PAM was realized. PAM is the main authorization framework of Linux systems. It allows the enforcement of policies in typical actions of the service delivery process. It supports account management, session management, password management and authentication management (see Figure 5.10). Account management relates to the existence of

users and their permissions to access a system. Session management relates to the definition of the parameters applied to the user environment. Password management relates to the manipulation of authentication tokens, may them be passwords, one-time hardware tokens, biometric credentials or smart cards. Authentication management relates to the validation of the credentials provided by users upon accessing the service.

A key strength of PAM is that it provides a coherent and uniform interface for the definition of policies for service usage, through a clear separation between service, policy and authentication methods. Any service which is integrated with PAM will delegate all the validation of credentials and authorization to the policies configured at the PAM system, and may be completely unaware of the databases and methods used for validation of credentials and storage of policies. Therefore, the development of a PAM module supporting account authorization based on community policies will enable any service to immediately condition its access by the policy decisions produced by the CMF system.

The *PAM-CMF* component developed integrates with the *Service Interface* component of the CMF and is able to support community aware service authorization, through the account management actions of PAM. When a new service request is initiated, PAM enabled services invoke the PAM account management actions and either allow or forbid access based on the decision reported. Internally, the *PAM-CMF* components checks if there is any permission associated with the given service, and if they exist, whether they are met or not.

Authentication is not supported by the prototype, only validation of the user account taking in consideration the name provided to login in the service, and the information presented in the community policy. Authentication is delegated to other PAM modules and may be supported by simple text files, databases or more complex granting systems such as Kerberos [231]. Figure 5.11 depicts the high level message flow triggered when a service wishes to use PAM to authorize a user, and the *PAM-CMF* module is used. As depicted, access to the *Service Interface* is protected by a secret shared between the *PAM-CMF* module and the CMF. The purpose is to restrict users, in a multi-user environment, to freely register new (potentially fake) services.

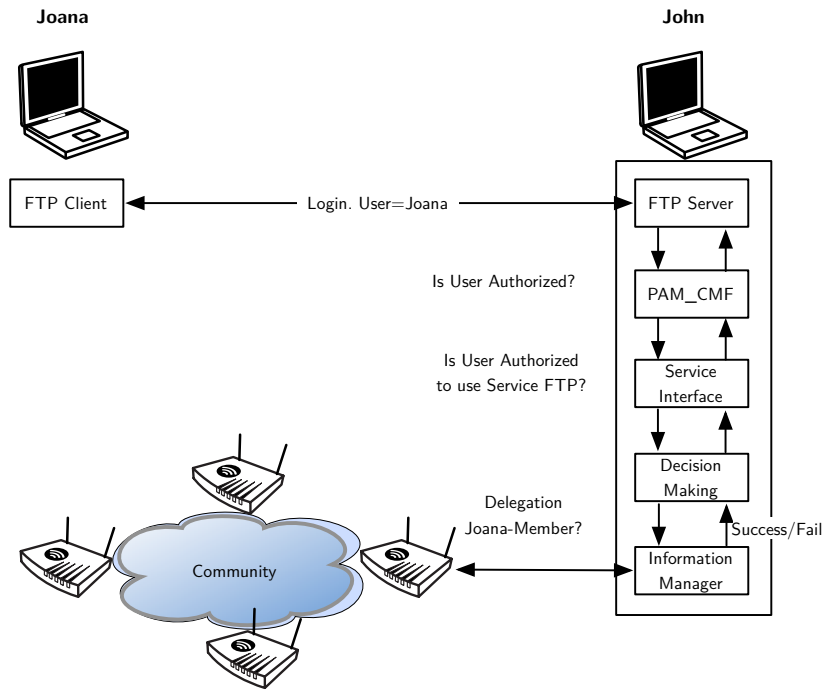


FIGURE 5.11 – Flow of invocations related to the authorization of a user.

A scenario was devised (see Figure 5.12) so that the distributed authorization capabilities of the policy were demonstrated. This scenario consisted of several wireless access points, all belonging to a community *C*, and two clients connected to the access points. One of the clients (*John*) belongs to the community *C*, and provides files through a FTP server.

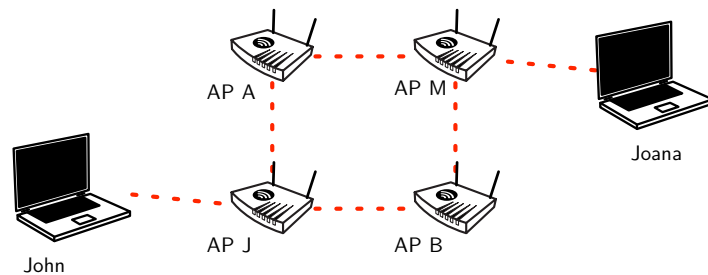


FIGURE 5.12 – Service authorization scenario with four access points.

The management system is running in all systems, except in *Joana*. If no community was active, access to the FTP server would be determined only by its administrator. However, in this particular case, the community imposes that FTP access should be allowed only to

members of the community. Therefore, the service will only be provided to members and external users will be denied access.

This is achieved by describing the FTP service in the community policy, adding a *Permission* element (see Code 5.7). In this element it is stated that the Member role is required for basic access and read operations. It should be noticed that according to this scheme, different roles could be entitled to different permissions, and other roles could have write permissions. Moreover, permissions for individual users could also be stated by means of defining a different *Subject* (e.g., *User.Joana*).

Integration of the CMF with the FTP server was achieved by means of a PAM configuration directive. It stated that validation by the *PAM-CMF* component was required for account validation purposes². Therefore, additionally to all other PAM modules invoked during the authorization, authentication and session establishment processes, also the *PAM-CMF* was to be called. No changes are imposed to the authentication process. Users still require the existence of an account with some authentication token to be validated, only the authorization process was changed. Also, PAM is unable to enforce write and read restrictions for the FTP service, and only authorization is supported.

```
<Community name='C'>
  <Permissions>
    <Permission Read='true' Access='true' Subject='Role.Member' Object='Service.FTP'>
  </Permissions>
  . . .
</Community>
```

CODE 5.7 – Community policy describing a service named FTP which is available to members.

In the scenario described, *Joana* uses a commonly available, unmodified, FTP client and connects to the FTP server (also unmodified) made available by *John*. The result is that FTP invokes PAM for authorizing access by the username provided (*Joana*). *PAM-CMF* has no information regarding the requirements for accessing the FTP service, but it informs the framework about the service being accessed, as well as the username provided. As depicted in Figure 5.13, *PAM-CMF* questions the framework regarding the token `Authorize:<FTP,Joana,access>`. Because the service is constrained to the role Member, and *Joana* has no association with the community C, access will be denied.

The framework is able to rapidly compute the requirements for issuing a delegation to the new member. After this, the required object is searched in the shared information

²account required pam_cmf.so service=FTP,password=xxx

repository by a query towards the node which should hold the object. Because in this case, the community is rather small, *John* may have direct knowledge of all other members, and will issue the request directly to the correct node.

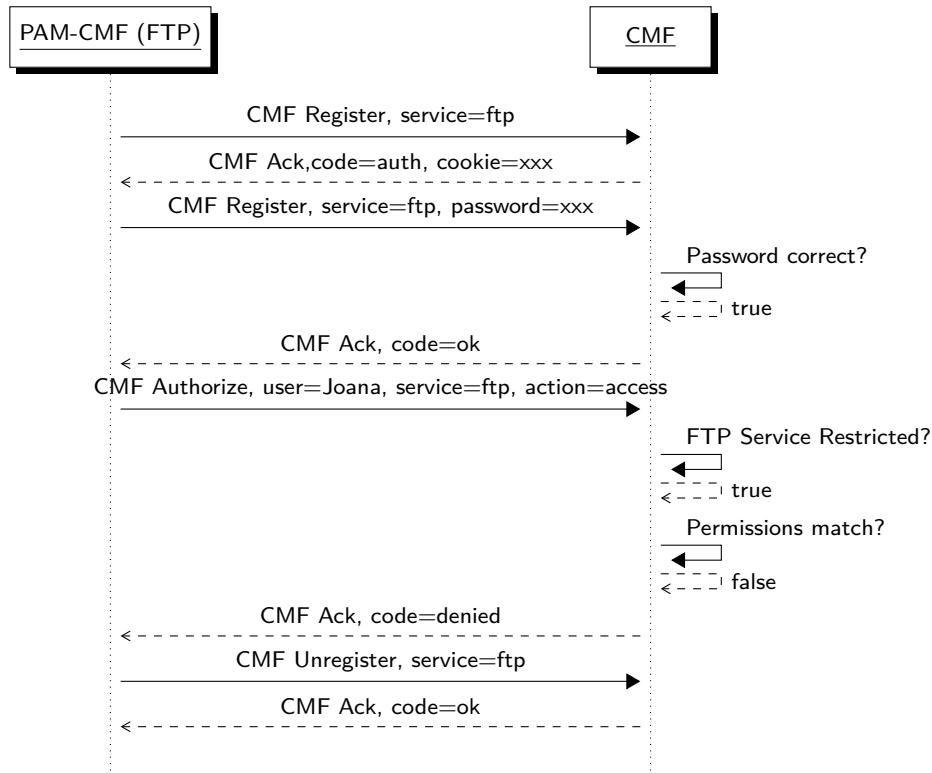
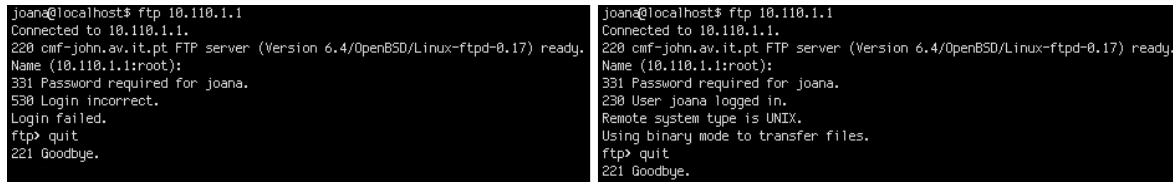


FIGURE 5.13 – Message sequence diagram of the authorization process for a single user.

Joana can enable the CMF instance at her laptop. As *AP-M* is broadcasting the community name through its interfaces, and the community has no membership restriction, *Joana* laptop rapidly joins the *C* community. As described previously, this is achieved by instantiating a *Delegation* token, which is signed and stored at the distributed storage. The same FTP access process is repeated, but this time a *Delegation* is found and service access is granted (*Joana* can access the FTP server owned by *John*). The output in both cases is depicted in Figure 5.14. The message flow is similar to the previous case; with the difference in the result obtained (Authorized). CMF instances can immediately try to issue *Delegation* tokens for new members, or this operation may be handled manually. Users are free to decide how the software instance behaves. It should be considered that the *Delegation* token effectively binds two users and a role in a community. If the new members misbehave, the user which delegated the role may see individuals reduce the trust in him.

Computing the requirements for accessing, querying the information storage and issuing an answer may take a variable amount of time. Three cases can be identified: failed authoriza-



```

joana@localhost$ ftp 10.110.1.1
Connected to 10.110.1.1.
220 cmf-john.av.it.pt FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) ready.
Name (10.110.1.1:root):
331 Password required for joana.
530 Login incorrect.
Login failed.
ftp> quit
221 Goodbye.

joana@localhost$ ftp 10.110.1.1
Connected to 10.110.1.1.
220 cmf-john.av.it.pt FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) ready.
Name (10.110.1.1:root):
331 Password required for joana.
230 User joana logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.

```

FIGURE 5.14 – Client output of the two access attempts (left fails, right succeeds).

Scenario	Time (ms)	Control Traffic (B)
Reject	1003	586
First Accept	2074	1123
Second Accept	1048	690

TABLE 5.2 – Traffic produced and time spent for authorizing a service through PAM.

tion, successful authorization, and repeated successful authorization. When authorization fails because the correct credentials are not found, the process is quick, as the framework only needs to query the node which should have the delegation and issue the verdict.

In the second case, the remote node is queried but the delegation object returned must be transmitted and its delegation tree must be verified. The delegations of all users in the delegation tree are obtained and verified for correctness and validity. For this, they must not be revoked, must have a valid signature, and the current time must fit between the start and expire dates. This step avoids an attack where false delegations are created by nodes in order to authorize others (or even themselves).

The third situation is similar to the previous, with the difference that local caching will speedup the process by avoiding transfer of all objects through the network. Nevertheless, cache consistency processes will incur in some delay. These three scenarios were tested in the described environment and the average time obtained for each case are depicted in Table 5.2 (with confidence of 95%). It should be noticed that the delays experienced by users may be higher than the times reported. This is due to the existence of mechanisms deployed, which keep authorization time almost constant, in order to reduce the probability of a successful time based attack. Also, the rate limiting mechanisms of one lookup invocation per second is the actual culprit of authorization delay. This value is artificially placed and can be removed, however, this is important to increase scalability and reduce management overhead.

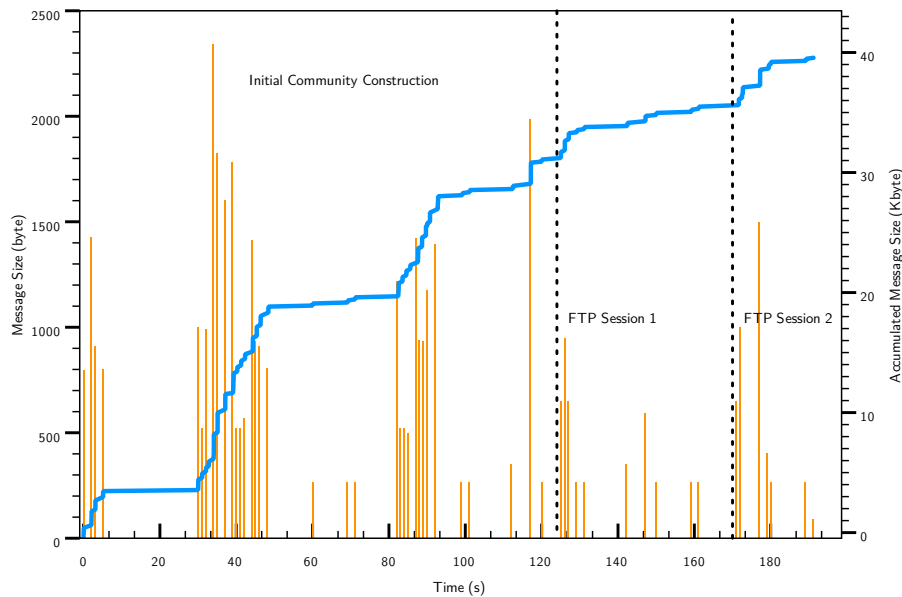


FIGURE 5.15 – Control traffic produced during the successful authorization of two FTP sessions.

Figure 5.15 depicts the traffic in bytes per second, as well as the total accumulated traffic produced during the creation of a community, and acceptance of two FTP sessions for the same user. As it can be seen, the authorization process introduces little traffic in comparison to the initial construction of the community and replication of the data objects. Also, the second FTP session (repeated with success) produces even less traffic.

As shown, a simple policy can enable a community to be created, and to grow by direct dissemination through the contacts of its members. Also, access to services can be tied to the policy of the community, even to legacy services such as FTP.

5.3.2 CROSS LAYER GATEWAY SELECTION

CMF has the possibility of creating simple policies, which can be applied in a cross layer manner, for optimizing network behavior. The only requirement is that the components addressed by the community have a representation in the management software. As described previously, this is achieved through the existence of services and devices with

manageable attributes. An interesting case for community management is the optimization of the cost of Internet access for a neighborhood mesh network. Considering the scenario depicted in Figure 5.16, several houses are interconnected by means of a wireless mesh network. Two of the households have permanent Internet connectivity by means of a ADSL connection to two (different) network operators.

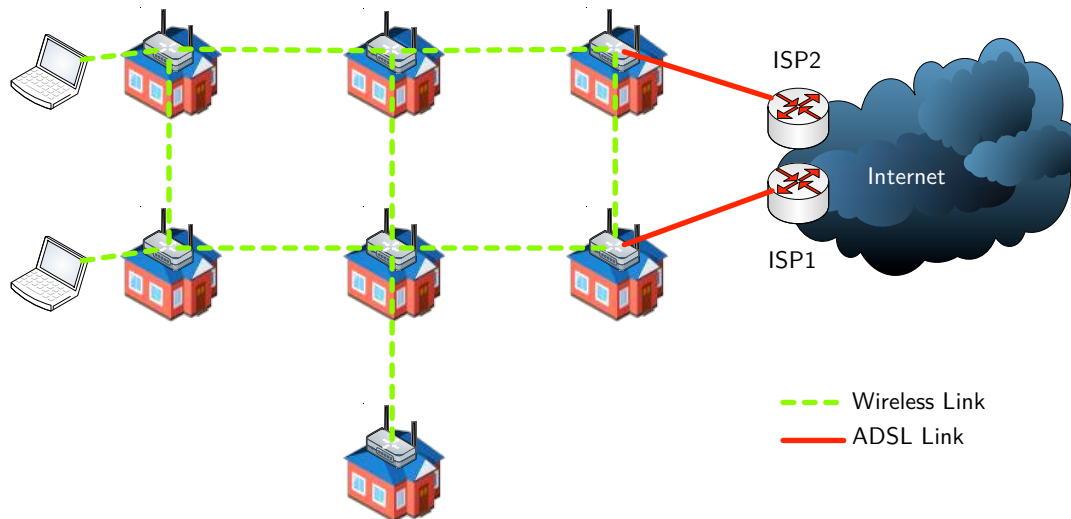


FIGURE 5.16 – Two gateways provide Internet connectivity to a neighborhood wireless mesh network.

Internet connectivity can be provided by either one of the two households connected to the Internet. Providing connectivity has a monetary cost associated, which is charged to the owner of the contract associated with the communication line. If traffic is not balanced among the gateways, one user will be charged most of the traffic produced, while the other will be charged for almost no traffic. Because charging tariffs are not always linear, in the sense that there is not always a linear cost per information unit, the total cost of providing connectivity will be dependent on the actual traffic sent by each one of the communication lines. Therefore, it may be important to distribute traffic in such a way that operational costs are reduced. A strategy can be devised so that traffic is sent through the cheapest gateway, but it should be noticed that this strategy may have an impact on total performance, or other metric.

There is a large amount of solutions aiming for optimizing the gateway selection process [165, 232], but frequently the only metric considered is the cost of the path between a sender and the gateway. The actual metric used to estimate cost can be simply the number of hops, the end-to-end latency, or more complex metrics such as ETX or Expected Transmission

Time (ETT) [132], but not actual monetary cost. Such metrics provide near optimal routes to the outside of the mesh network, and are able to minimize loss and delay, or maximize capacity, depending on the metric. They are important because they enable routing decisions to take in consideration link quality and occupation. However, an important aspect of such networks is the actual cost charged by the operator providing Internet connectivity as this will have impact on the operational cost. Different Internet providers also have different models of charging traffic. Some have flat rate plans, some charge by usage based on time, others charge by usage based on volume. Also, there are some bandwidth constraints in place limiting link capacity to values currently varying between a few hundred of Kib.s^{-1} to several hundreds of Mib.s^{-1} .

In the scenario described, CMF was used to associate routing decisions to actual cost of accessing the Internet. OLSR was used as the base protocol providing routing in the mesh network, as well as advertising and disseminating gateway availability. Inside the wireless mesh network, OLSR was configured to use the ETT metric so that link quality and occupancy aspects were taken in consideration. When selecting a gateway to use, nodes take the standard ETT metric in consideration, and add a cost factor which is advertised by the nodes providing gateway service. Traditionally, this cost factor only took in consideration the capabilities of the outgoing interface, in terms of inbound and outbound bandwidth. Gateways in OLSR advertise the capability of connecting to outside networks using Host and Network Association (HNA) [42] messages. In these messages, nodes advertise their willingness to forward traffic to external networks. HNA messages are very simple, containing only the network address and mask to be advertised to all other OLSR nodes.

By taking gateway information into consideration, stations can now choose the most appropriate route to reach the Internet and get global connectivity. As implemented, and in order to avoid continuously switching between the available gateways, the gateway selection process is conservative and only effects new flows³.

Two network services were developed for this test: *NSGateway* and *NSRoutingOLSR*. The *NSGateway* service manages the connection to the Internet and can effectively control whether the service is available or not, track basis statistics, and allows the configuration of a charging plan. The statistics include the amount of traffic sent and received by the interface, the bandwidth available and the amount of time the interface is operational. A charging plan can be set, thus allowing estimating the cost associated to traffic sent and received to and from the Internet. Two models are supported: Volume based and Time based. In the Volume based model, cost is estimated taking in consideration the amount of traffic transferred in units of 1 MiB, both below and above a threshold. The threshold mimics the typical practice of charging

³Routes providing Internet connectivity to SOHO environments make extensive use of NAT mechanisms, which in its simple form is incompatible with any form of multi-homing or load balancing.

different prices for traffic below and above a certain monthly accumulated amount. Time based models allow charging by taking in consideration the amount of time the interface is connected to the Internet in periods of 1 minute. This module effectively enhances the policy by adding support for the `NetServ_Gateway_Cost` read-only attribute.

The `NSRoutingOLSR` service is a specialization of the `NSRouting` service and provides the means to manage the cost of internal and external routing. The prototype developed is integrated with `OLSRd-0.6.0` [233], and an inter process communication mechanism was implemented, providing communication between the actual routing daemon and the `NSRoutingOLSR` service residing inside CMF. This mechanism allowed policies from the CMF to get or set the cost of the local interfaces. Also, it enabled to set the cost of the gateway route advertised to others. Effectively, this service enabled support for two attributes: `NetServ_Routing_GWCost` read-write attribute, and the `NetServ_Routing_Cost` read-write attribute.

`OLSRd` implements HNA messages, but not as specified in the latest standard. Like referred previously, HNA are supposed to contain the network address and mask to be made available. If the address is equal to `0.0.0.0`, with mask equal to `0.0.0.0` the gateway is actually advertising a default route. An interpretation made by the `OLSRd` development team for the case of gateways is that the mask can be ignored as it will always be equal. The standard implementation of `OLSRd` uses this byte to inform hosts about the bandwidth of each gateway (see Figure 5.17).

When integrated with CMF, through the `NSRoutingOLSR` service, `OLSRd` behavior was once again modified, and this byte is used to propagate an administrative cost. When calculating the cost of the route to all available gateways, the modified implementation of the `OLSRd` computes the route using the number of links, and considering the ETX metric, but then administrative cost is multiplied by 256 (equals to a 8 bit shift to the left) and added to the standard cost metric. The result is that if considering a single gateway, the route selected to reach the gateway is the one presenting the best value for ETX. If multiple gateways are present, the one gateway presenting inferior administrative cost is used. Still, the implementation will choose the best route to the selected gateway.

In this scenario, the administrative cost will be set to the cost reported by the `Gateway` module, thus linking real world currency cost to routing cost.

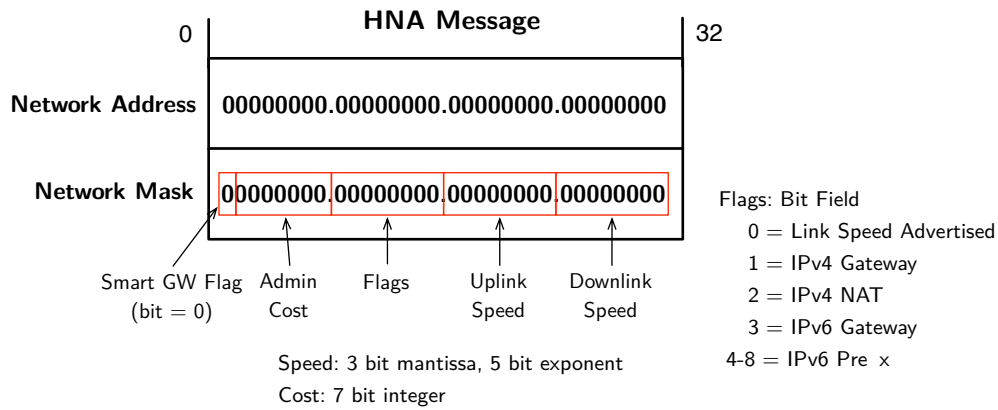


FIGURE 5.17 – HNA message containing SmartGateway information.

```

<Community name='IT' owner='User.John'>
  <Roles>
    <Role name='Member'>
      <Attributes>
        <Attribute name='NetServ_Forward_State' type='Int'>1</Attribute>
        <Attribute name='NetServ_Gateway_State' type='Int'>1</Attribute>
        <Attribute name='NetServ_Routing_State' type='Int'>1</Attribute>
      </Attributes>
      <Rules>
        <Rule name='InternetHighCost' sample='5'>
          <Conditions>
            <Condition name='CondInternetHighCost' operator='>='>
              <Attribute name='NetServ_Gateway_Cost' type='Int'>0</Attribute>
            </Condition>
          </Conditions>
          <Attributes>
            <Attribute name='NetServ_Routing_GWCost' type='Int'></Attribute>
          </Attributes>
        </Rule>
      </Rules>
      <Permissions>
        <Permission read='true' write='true' delegate='true' role='Member' />
      </Permissions>
    </Role>
  </Roles>
</Community>

```

CODE 5.8 – CMF community definition supporting cross layer Gateway Cost definition.

CMF was started in all nodes, and the community described in Code 5.8 was provided to

one of the nodes (its creator). Because no access condition was configured, the community name was set as the default community in every node. As all members could further delegate it, this community was advertised and rapidly propagated to the entire network. The policy states the existence of a community named *IT* where the members of the *Creator* role have full control. There is an additional role named *Member* with specific attributes and rules. The attributes specify that all participants must enable their packet forwarding, gateway and routing services. The existence of those particular services is not mandatory as no access condition is set. Nevertheless, if the service is present, it should be enabled.

A rule object in the *Member* role is evaluated each 5 seconds and links the cost advertised by the *Gateway* service to the cost used by the *Routing* service. This will allow near instant propagation of the cost figures from one layer to the other.

With the scenario and policies described, it becomes possible to assert the usefulness of a cross layer policy solution for balancing traffic according to cost. For this purpose, an experiment was run with the duration of 30 minutes, aiming to simulate a month worth of Internet usage. It was considered that each user with Internet connectivity had 250GiB⁴ of traffic included for the flat rate of 15 currency units. Each GiB above this value would be charged at an additional 1 unit⁵. If the total consumption made by the remaining members is low, it is expected that the total cost of providing connectivity will be of 30 units, 15 units for each of the gateways, because of the flat fee applied. If total traffic exceeds the 250GiB threshold, connectivity cost will increase rapidly, at a much higher rate. Load balancing traffic taking in consideration cost as the most important aspect can reduce the cost of operating the network.

The scenario evaluated depicts a network containing hosts that continuously use the Internet, consuming traffic at a rate of 20GiB per day, which averages to 232KiB.s⁻¹ of continuous traffic. After 30 days, hosts are expected to have consumed 600GiB of information, therefore exceeding the threshold for the flat rate. If load balancing is used, hosts are expected to have consumed 500GiB of data at the flat rate cost, plus 100GiB of data at the inflated cost. If otherwise, the traditional non cost aware method is used, and hosts are expected to have consumed as much as 350GiB of information at an inflated cost.

The experiment considers that time was compressed from 30 days to 30 minutes, which results in a scale factor of $1/(24 * 60)$. Therefore, instead of transferring 600GiB, in the experiment, hosts actually transferred 417MiB at a rate of 161B.s⁻¹. Figure 5.18 depicts how cost evolves under these assumptions: when using a community policy with cost aware load balancing, and in a traditional, non load-balanced situation.

⁴According to IEC, 1GiB equals to 2³⁰ bytes.

⁵In many European countries it is common the practice of charging as little as 15 euros for an broadband

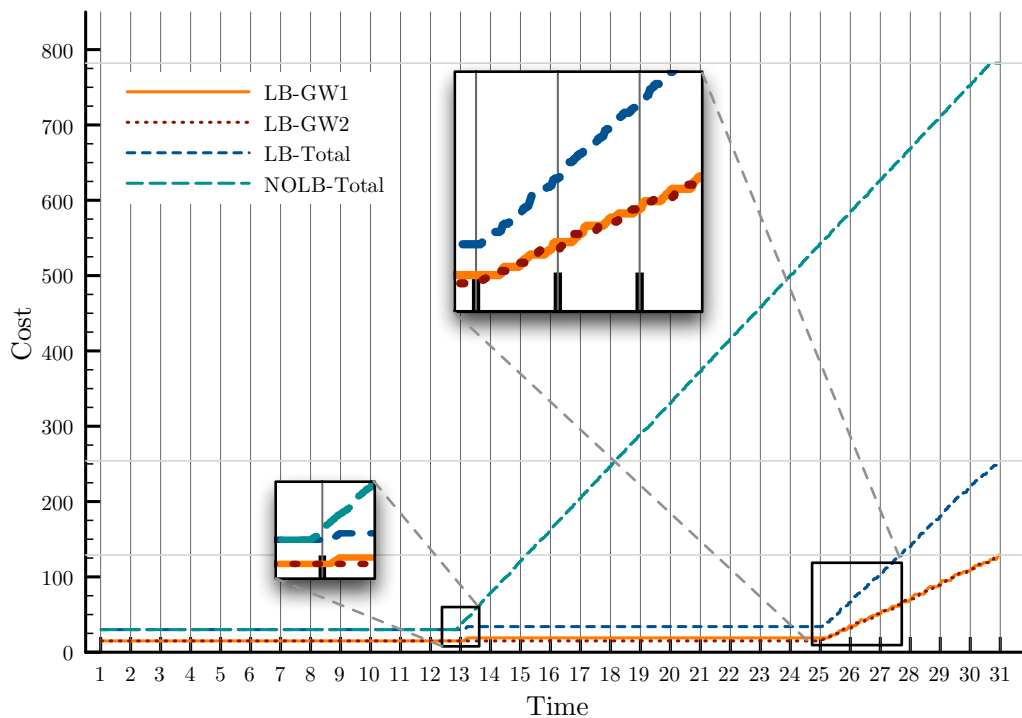


FIGURE 5.18 – Total cost charged to the gateways considering standard and CMF enabled scenarios, under realistic charging assumptions.

If the policy is activated (i.e. the community is active) in the two gateway nodes, traffic will be split among the different gateways according to the cost reported by each gateway. As described previously, OLSR will take cost information in consideration when choosing the most appropriate gateway. Due to the threshold configured for traffic ($15 \text{ units for } 250\text{GiB}\cdot\text{s}^{-1}$), the total cost of operating the network will be much lower using load balanced gateways. Using a simple gateways for routing will imply a cost of 767 currency units, while the sum of the cost charged to both gateways will be of only 272 currency units (135 for one gateway and 137 for the other), achieving savings of around 65%. Without the penalty threshold set at 250GiB (that is, true unlimited charging plan), with flat fee, or using a linear pay as you go charging plan, total cost is expected to be similar in both cases. The difference would be that the cost (and traffic) would be split among the different gateways, instead of being concentrated on only a single gateway.

This example clearly shows that the CMF can correlate attributes from multiple aspects of a managed element. Also it can be used to interact with external components of the connection with up to 250GiB of traffic, and then a few euros per GiB above this threshold.

networking stack by means of simple adapters. These adapters simply map key functionality of the service into policy attributes. From this point beyond, the attributes are available to be managed by the community policy. It is also shown that a simple policy like the one described enables real cost savings in a wireless community.

5.4 SUMMARY

This chapter presented the design guidelines, requirements and actual deployment of the experimental facility used for evaluating the prototype developed, and to serve as a test facility for several other dissertations and thesis dealing with wireless technologies.

Also, details regarding the implementation of the CMF prototype are discussed, including the most important messages exchanged, and the functioning of the distributed storage. A key component of the prototype.

Finally, the key scenarios evaluating important functionality of the CMF prototype are analyzed. The results obtained with the prototype show the feasibility of managing distributed systems, while following community concepts. Integration with external services is demonstrated, as well as integration with network functionality such as shaping and gateway selection. From the results presented, it is perceivable the effectiveness of the system, and its added value for the management of wireless based communities of interest.

CONCLUSION

Thisis need conclusions. It doesn't mean the work is over. It means that the work progressed for some time, some advances were achieved and a situation point should be made. This chapter presents the situation point of the work developed during the elaboration of this thesis. It includes a discussion of the achievements, as well as the most relevant contributions made during the period. Some prospects regarding ways through which the work may be improved are also presented. Some of the topics addressed as future work are of technical nature, but some are conceptual, requiring further reflection. The thesis is completed with some final remarks of the author.

6.1 REVIEW AND DISCUSSION OF ACHIEVEMENTS

The way individuals interact is driven by social relations, personal interests and collective interests. Interactions between individuals are not random, and are shaped by the context where individuals are integrated. Being it family, friends or colleagues, individuals establish social relations, some of them for their entire life. Information follows these links, independently if users communicate orally or by using electronic communication devices. The advent of Internet forums and social web applications didn't violated this rule, only created additional contexts. Also, it facilitated communication between individuals.

In this sense, communities of individuals, established due to the existence of a shared context and interest, sprouted in many ways. This thesis tackled the communities which involved networking devices, and collaboration over digital media. In particular, it focused in scenarios of collaboration between individuals resulting in the formation of wireless community networks, and in one of its main challenges: the infrastructure required to manage its structure, behavior and membership. As presented, the number of community driven initiatives providing Internet connectivity has a reasonable dimension, and gathers the

experience of more than one decade. Some deployments, are already comprised by millions of users.

The area of system management, in particular when considering networked elements considers many different solutions. Taking in consideration the usage scenario and the capabilities of the management elements, a management approach can be as simple as static configuration by human operations, or as complex as knowledge based systems capable of reasoning over abstract knowledge. In the scope of wireless networks, devices are low powered but not static. Solutions for wireless networks must consider the inherent dynamism of the wireless medium, resulting of node mobility and variable transmission characteristics. Therefore, this work focused in a solution that sits between these two extremes. It considers the existence of static configuration attributes, ECA rules, and provides a distributed storage enabling the development of autonomic, or even knowledge based management solutions.

dRBAC is the most important concept adopted by the work presented in this thesis. One of its most important concepts is the concept of role, which can be assigned to users using delegations. Under this assumption, several scenarios capable of benefiting from community concepts were presented, and several key roles were proposed. Any other name can be attributed to the roles proposed (e.g., Visitor instead of Guest), however, the author believes that the basic functionality of an dRBAC system in those environments is the one described. It is not discarded the possibility of finding communities with more roles than the ones identified. Actually the flexibility for defining custom roles was considered in every aspect of this work. From a conceptual point of view, the policy concepts described are not tied to any specific naming scheme for communities and roles.

Part of this work as deeply impacted the global architecture of the EU funded IST-WIP project. The concepts present in IST-WIP share many similarities with this thesis, diverging due to different set of initial scenarios, assumptions and requirements. IST-WIP considered scenarios with an hierarchical structure of communities, with lower level communities dedicated to providing connectivity, and higher layer communities aggregating users sharing similar interests. This thesis doesn't make that consideration as it considers that those two communities are simply two scenarios, from a multitude of other scenarios with user interaction. Therefore, it focused in developing a generic set of concepts and language allowing the creation of those, or other types of communities. IST-WIP vision considered strong assumptions and requirements regarding trust and reputation inside the community, and between communities. The vision of global reputation was never clearly integrated with the community structure paradigm, due to the complexity of the issues involved with calculating absolute values from an unknown number of sources. The author recognizes that reputation and trust are important, and has considered the existence of trust associations in

the form of delegation objects. However, the global reputation aspect was considered as too ambitious and complex for being integrated in this thesis.

Evaluation of the concepts proposed followed an experimental approach, which also resulting in the development of a permanent testbed. Located in the rooftop of the Instituto de Telecomunicações in Aveiro, this testbed has been operating almost continuously since 2009. Besides being the main evaluation platform for the prototype developed in this thesis, the testbed provided invaluable results for several dissertations.

The policy concepts proposed in this thesis were evaluated in two much relevant scenarios, and showed several key features of the approach adopted: i) legacy services can be integrated with community concepts, for the purpose of access control, without modification to its source code; ii) the policy concepts are able to represent information from network services and applications, actually showing that the solutions is cross-layer; iii) users are able to grant access to their friends, and constitute communities united by a single policy, by resorting to delegation of roles; iv) a simple policy can be useful to solve real problems, such is the case of sharing connectivity cost in community aware WMN; v) the distributed management approach followed throughout this thesis is suitable for the purpose of managing sparsely located managed elements.

The concepts presented, and the work developed during the elaboration of this thesis have had high impact in key areas, which is proved by the number of publications, most at international conferences and journals, contribution to EU funded projects, as well as number of citations by research peers.

6.2 FUTURE WORK

The work presented in this thesis is a first step towards the existence of systems capable of taking in consideration user structure, optimizing its operation accordingly. Due to the complexity of the subject, several directions can be pursued for future work.

Future work could focus in increasing the different types of relations presented at a social level to the management of the community. In the solution proposed, Roles can be used to distinguish between the different types of users, and device capability. However, roles are unique to the entire community. The definition of user centric roles, or trust levels between users, could be used to the design and deploy more complex and effective incentive mechanisms. Even without explicit incentives, having more knowledge about the community structure and the strength of relations between its members could be used to further improve community management.

The language proposed to create the community policy is represented in a XML form. It allows the creation of independent policy blocks, capable of being cryptographically signed, exchanged and verified by member of the community. The use of this representation present benefits, in the form of easy comprehension by humans, and faster integration of the prototype developed with other solutions. The drawback is that XML is not a compact representation method. Using XML imposes a penalty to systems, in the form of somewhat increased memory and processing requirements. Also, transmitting XML policies in wireless networks imposes higher overhead than desired. This work shows that those requirements are still small, and the overhead is not much relevant for network operation. Nevertheless, a more compact representation of data could further improve the resulting system. Compression techniques, optimized to the representation format of the policy could further reduce bandwidth, at the cost of higher memory and computational requirements.

Due to the target scenarios envisioned by this work, the prototype developed is focused on the Linux operating system. This also had impact to the naming of some concepts as well as the services and attributes considered. Further work could improve the solution presented by extending the management capabilities to other operating systems, both for desktops and mobile devices.

Integration of several network services was proposed during this work. Also, a PAM was developed, which allows the integration of generic services in a Linux environment. Future work include the extension of the still simple service interface and the integration of more services. Examples of potentially relevant services are file sharing applications, discussion forums and other collaboration tools, distributed social networks, instant messaging applications, and solutions for virtual private networking. These services are especially relevant as they have the capability of fostering interactions inside the community, increasing its social capital. Virtual private solutions are relevant for the case of communities which span over networks or the Internet.

Policy objects, as well as knowledge created inside communities, can be strictly linked to their owner through metadata and verified through cryptographic methods. The work presented suggests the use of existing solutions such as a PKI, or a web of trust such as OpenPGP. As recently proposed by Ulrich *et al* [234], the structure of a Web of Trust (WoT) has some similarities with social networks, and can also be linked to the actual social structure, or even community structure. Developing a cryptographic framework for signature, verification and cipherring of data, integrated with the community policy mechanisms proposed, represents an interesting future step stemming from the solutions proposed.

In the solution proposed, the software framework has access policies, owns sensed data, and the result of rules executed by other members of the community. The amount

of information available allows the development of truly autonomic functionalities, which make use of reasoning engines to process data at a semantic level. Low power nodes such as SOHO devices will mostly be unable to have such complex software instances running. However, devices with high processing capabilities could be deployed in the community network and much improve the autonomic characteristics of the system. Still, it should be considered that autonomic reasoning over inconsistent and incomplete information is still an area of research with many challenges waiting to be overcome.

Another perspective of future work would be to integrate more robust service discovery solutions into management of the community. Preliminary work proposing the integration of an extended service discovery architecture was presented in [21]. It allowed the dissemination of both communities and services inside communities, and using methods adequate to both mesh and ad-hoc scenarios. Future work could extend the solution proposed by designing a management interface capable of taking in consideration privacy and security requirements of a community, or its members.

Additionally to these perspectives that refer to specific areas of improvement to the work developed in this thesis, it is also adequate to consider general improvements to the work. Experimentation using more users, potentially scaling up to hundreds or thousands of users could better demonstrate the scalability of the solutions proposed. However, the scalability of the system is highly related to the scalability of its storage. Being based on a DHT, there is already extensive analysis and proofs of how these systems behave with high number of users. However, higher rates of dynamism, in the form of higher churning rates would also present relevant results for the application of community based management solutions to real world scenarios.

6.3 FINAL REMARKS

A doctoral thesis is not a document which is produced in a short amount of time, and the work required for its production will keep any researcher busy for some years. Therefore it is frequent to observe the world changing together with the work developed. Sometimes in the same direction, sometimes following totally opposite directions.

When the work started, wireless networks were at its most recent boom. Borrowing concepts from the Gartner Hype Cycle, wireless (mesh) technologies using IEEE 802.11 solutions were near the Peak of Inflated Expectations. A huge amount of wireless mesh networks were sprouting and even municipalities started supporting, otherwise citizen supported, wireless communication infrastructures. As with someone familiar with the

Hype Cycle will know, what follows the Peak of Inflated Expectations is the Trough of Disillusionment. User driven wireless networks, were able to foster communities around their networks, and even got support from municipalities, but this was not enough. Radio propagation is not a something to be taken lightly in a crowded urban environment. The amount of interference, multipath attenuation and direct attenuation by building walls, required viable wireless networks to use a greater than envisioned density of wireless routers. Also, the wireless medium can be highly unpredictable and temperamental when considering longer links. This required more effort from the community, and more careful planning regarding location, antennas and radio spectrum.

The issue limiting most wireless networks maintained by municipalities was the lack of a viable business plan. Without a source of income to cover expenses, many wireless community networks were fatally condemned. Solutions allowing for higher throughput such as WiMAX also lost its battle against LTE, and saw its at the time optimistic deployment plan, delayed for a few years. Some municipalities still believed in their community networks, and deployed a WiMAX infrastructure to better provide network access. Some operators pursued the same route, but at the time, they had limited success. Many just quit WiMAX or switched to LTE. While LTE provides similar or even better capacity, it is based on a completely different operation model, which is not so friendly to user initiative. On the opposite, it is better suited to cellular networks, in particular if UMTS is already deployed.

After some stagnation, user driven wireless networks are now observing another sprout. Technology has kept decreasing the cost of communication equipment, and solutions matured by a great amount. Nowadays, all laptops have wireless connectivity, and smart phones represent a respectable percentage of mobile devices. This all makes wireless communications more relevant in today's world.

From the perspective of user centric communication, and community networking, the world is very different. With the massive usage of online social platforms such as Facebook [12], the world changed from a model based on content and producers, to a model self-centered around an individual and its social relations. More and more, users want contextualized content, and contextualized interactions, at a level much higher than the author envisioned when initiating this work. When a user interacts with a social application, information regarding the user contacts is automatically preloaded, ads are personalized, and the entire experience personalized based on the interests of the individual, his friends, and the communities it belongs.

From the point of view of management solutions for wireless networks. Specially those considering user participation, and community structure, the topic is getting much attention nowadays. More research grants cover the topic, especially when applied to smart cities, and

inclusive environments, and more publications are being observed in the area of community structure. Still, wireless mesh networks in operation are being very conservative over the adoption of such technologies. They are more focused in providing a generic communication infrastructure, allowing users to interact over generic online social portals. However, given the impact of social relations in current Internet, it is clear to the author that networks will also stop considering the management approaches where the network has a representation of the user relations. Either in the form of management solutions for communication environments, or solutions for content delivery (e.g., CCN).

REFERENCES

- [1] M. Weik, "Ballistic research laboratories report n°971 — a survey of domestic electronic digital computing systems", S Department of Commerce, Tech. Rep., 1955.
- [2] R. D. Putnam, *Bowling Alone: The collapse and revival of American community*. New York: Simon and Schuster, 2000.
- [3] S. Guberman and G. Minati, *Dialogue about Systems*. Polimetrica S.A.S., 2007.
- [4] P. Erdős and A. Rényi, "On random graphs", *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.
- [5] S. Milgram, "The Small World Problem", *Psychology Today*, vol. 2, no. 1, pp. 60–67, 1967.
- [6] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks", *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.
- [7] A.-L. Barabási and E. Bonabeau, "Scale-Free Networks", *Scientific American*, vol. 288, no. 60–69, 2003.
- [8] G. A. Hillery Jr, "Definitions of Community: Areas of Agreement", *Rural Sociology*, vol. 20, no. 2, pp. 111–123, 1955.
- [9] G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. D. Spyropoulos, "Clustering the Users of Large Web Sites into Communities", in *Proceedings of the 17th International Conference on Machine Learning*, ser. ICML '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 719–726, ISBN: 1-55860-707-2.
- [10] J. Donath, "Identity and Deception in the Virtual Community", in *Communities in Cyberspace*, P. Kollock and M. Smith, Eds. London: Routledge, 1999.
- [11] A. Abdelaal, H. Ali, and D. Khazanchi, "The Role of Social Capital in the Creation of Community Wireless Networks", in *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences HICSS*, IEEE Computer Society, 2009, pp. 1–10, ISBN: 9780769534503.
- [12] *Facebook*, <http://www.facebook.com>, [Online; accessed on December 2012].
- [13] *Orkut*, <http://www.orkut.com>, [Online; accessed on December 2012].
- [14] D. McMillan and D. Chavis, "Sense of Community: A Definition and Theory", *Journal of Community Psychology*, vol. 14, no. 1, pp. 6–23, 1986, ISSN: 1520-6629.
- [15] M. Walzer, *On Toleration*. Yale University Press, 1997, ISBN: 0-268-01897-9.
- [16] J. S. Coleman, *Foundations of social theory*. Harvard University Press, 1990.
- [17] M. Louta, S. Kraounakis, and A. Michalas, "A Survey on Reputation-based Cooperation Enforcement Schemes in Wireless Ad hoc Networks", in *Proceedings of the 2010 International Conference on Wireless Information Networks and Systems*, Jul. 2010, pp. 1–4.
- [18] A. Urpi, M. Bonuccelli, and S. Giodano, "Modelling Cooperation in Mobile Ad Hoc Networks: A Formal Description of Selfishness", in *Proceedings of the Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2003, pp. 3–5.

- [19] J. P. Barraca, S. Sargento, and R. L. Aguiar, "Role Based Cross-Layer Communities in WMN", in *Proceedings of the International Conference on Wireless Information Networks and Systems (WINSYS)*, Barcelona, Spain: INSTICC Press, Jul. 2007, pp. 85–92.
- [20] E. Freudenthal, T. Pesin, P. Lawrence, K. Edward, and K. Vijay, "dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments", *Proceedings of the International Conference on Distributed Computing Systems*, p. 411, 2002, ISSN: 1063-6927.
- [21] J. P. Barraca, P. Fernandes, S. Sargento, and R. Rocha, "An Architecture For Community Mesh Networking", in *Proceedings of the IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2008)*, Sep. 2008.
- [22] P. Fernandes, "Service Discovery for Wireless Networks", Master's thesis, Instituto Superior Técnico / Technical University of Lisbon, Dec. 2007.
- [23] E. Guttman, C. Perkins, J. Veizades, and M. Day, *Service Location Protocol, Version 2*, RFC 2608 (Proposed Standard), Updated by RFC 3224, Internet Engineering Task Force, Jun. 1999.
- [24] J. Postel and J. Reynolds, *File Transfer Protocol*, RFC 959 (Standard), Updated by RFCs 2228, 2640, 2773, 3659, 5797, Internet Engineering Task Force, Oct. 1985.
- [25] P. Antoniadis, B. Le Grand, A. Satsiou, L. Tassiulas, R. Aguiar, J. P. Barraca, and S. Sargento, "Community Building over Neighborhood Wireless Mesh Networks", *Technology and Society Magazine, IEEE*, vol. 27, no. 1, pp. 48–56, 2008, ISSN: 0278-0097.
- [26] J. P. Barraca and R. L. Aguiar, "Ontology-driven Framework for Community Networking Management", in *Proceedings of the International Conference on Telecommunications*, Jun. 2008, pp. 1–7.
- [27] R. L. A. João Paulo Barraca, "A Framework for User-centric Autonomic Management", in *Proceedings of the 11th Conferência sobre Redes de Computadores*, Coimbra, Portugal, Nov. 2011.
- [28] F. Dias, J. P. Barraca, D. Gomes, and R. L. Aguiar, "Characterization of Unplanned Metropolitan Wireless Networks", in *Proceedings of the 10th Conferência sobre Redes de Computadores*, Braga, Portugal, 2010.
- [29] F. Ferreira, "Caracterização de Redes Sem Fios Multihop Não Planeadas", Master's thesis, Universidade de Aveiro, 2009.
- [30] J. P. Barraca, A. Matos, and R. L. Aguiar, "User Centric Community Clouds", *International Journal on Wireless Personal Communications*, vol. 58, no. 1, pp. 31–48, May 2011, ISSN: 0929-6212.
- [31] J. P. Barraca, D. Gomes, and R. L. Aguiar, "AMAZING - Advanced Mobile wireless Network playGround", in *Proceedings of the International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops*, Berlin, Germany, 2010.
- [32] J. Martins, J. P. Barraca, D. Gomes, and R. L. Aguiar, "Experimentation made easy with the AMAZING panel", in *Proceedings of the seventh ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, ser. WINTECH '12, Istanbul, Turkey: ACM, 2012, pp. 11–18, ISBN: 978-1-4503-1527-2.
- [33] D. Raychaudhuri, M. Ott, and I. Secker, "ORBIT Radio Grid Tested for Evaluation of Next-Generation Wireless Network Protocols", in *Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMMunities*, ser. TRIDENTCOM '05, Washington, DC, USA: IEEE Computer Society, 2005, pp. 308–309, ISBN: 0-7695-2219-X.
- [34] J. P. Barraca, A. Brito, and R. L. Aguiar, "Network Interfaces Flying over IP Networks", in *Proceedings of the 2011 IEEE Symposium on Computers and Communications*, ser. ISCC '11, Washington, DC, USA: IEEE Computer Society, 2011, pp. 10 016–10 021, ISBN: 978-1-4577-0680-6.
- [35] "IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for

- Higher Throughput”, *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, p. 502, 2009.
- [36] “IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)”, *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, May 2011.
- [37] J. P. Barraca, R. Sadeghi, and R. L. Aguiar, “Collaborative Relaying Strategies in Autonomic Management of Mobile Robotics”, *International Journal on Wireless Personal Communications*, 2013, (waiting publication).
- [38] “Supplement to IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band”, *IEEE Std 802.11b-1999*, p. 90, 2000.
- [39] S. Corson and J. Macker, *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*, RFC 2501 (Informational), Internet Engineering Task Force, Jan. 1999.
- [40] D. G. Reina, S. L. Toral, F. Barrero, N. Bessis, and E. Asimakopoulou, “Evaluation of Ad Hoc Networks in Disaster Scenarios”, in *Proceedings of the 3rd International Conference on Intelligent Networking and Collaborative Systems*, 2011, pp. 759-764.
- [41] C. Perkins, E. Belding-Royer, and S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*, RFC 3561 (Experimental), Internet Engineering Task Force, Jul. 2003.
- [42] T. Clausen and P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, RFC 3626 (Experimental), Internet Engineering Task Force, Oct. 2003.
- [43] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang, “Security in mobile ad hoc networks: challenges and solutions”, *Wireless Communications, IEEE*, vol. 11, no. 1, pp. 38 - 47, 2004, ISSN: 1536-1284.
- [44] L. Yan and S. Hailes, “Designing Incentive Packet Relaying Strategies For Wireless Ad hoc Networks With Game Theory”, in *Wireless Sensor and Actor Networks II*, ser. IFIP International Federation for Information Processing, A. Miri, Ed., vol. 264, Springer Boston, 2008, pp. 137-148, ISBN: 978-0-387-09440-3.
- [45] M. Almeida, R. Sarrô, J. P. Barraca, S. Sargento, and R. L. Aguiar, “Experimental Evaluation of the Usage of Ad Hoc Networks as Stubs for Multiservice Networks”, *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, no. 1, p. 062 967, 2007, ISSN: 1687-1499.
- [46] S. M. Senouci and G. Pujolle, “Energy efficient routing in wireless ad hoc networks”, in *Proceedings of the IEEE International Conference on Communications*, vol. 7, Jun. 2004, pp. 4057-4061.
- [47] H. Galperin, “Wireless Networks and Rural Development: Opportunities for Latin America”, *Inf. Technol. Int. Dev.*, vol. 2, no. 3, pp. 47–56, Mar. 2005, ISSN: 1544-7529.
- [48] R. Chandra, L. Qiu, K. Jain, and M. Mahdian, “Optimizing the placement of integration points in multi-hop wireless networks”, in *Proceedings of the IEEE International Conference on Network Protocols*, 2004, pp. 271-282.
- [49] M. S. Al-kahtani and H. T. Mouftah, “Enhancements for clustering stability in mobile ad hoc networks”, in *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, ser. Q2SWinet '05, Montreal, Quebec, Canada: ACM, 2005, pp. 112–121, ISBN: 1-59593-241-0.
- [50] S. Karunakaran and P. Thangaraj, “An adaptive weighted cluster based routing (AWCBRP) protocol for mobile ad-hoc networks”, *WTOC*, vol. 7, no. 4, pp. 248–257, Apr. 2008, ISSN: 1109-2742.
- [51] K. T. Mai, D. Shin, and H. Choo, “Connectivity-based clustering with stretching technique in MANETs”, in *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '09, Suwon, Korea: ACM, 2009, pp. 200–206, ISBN: 978-1-60558-405-8. DOI: 10.1145/1516241.1516276.

- [52] S. Zhao and D. Raychaudhuri, "Scalability and Performance Evaluation of Hierarchical Hybrid Wireless Networks", *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1536–1549, Oct. 2009, ISSN: 1063-6692.
- [53] "Draft Amendment to Standard for Information Technology – Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements – Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: Amendment: ESS Mesh Networking, IEEE P802.11s/D1.0", *IEEE 802.11s Task Group*, 2006.
- [54] "IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems Amendment 1: Multiple Relay Specification", *IEEE Std 802.16j-2009 (Amendment to IEEE Std 802.16-2009)*, p. 290, Dec. 2009.
- [55] "Supplement to IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5 GHz Band", *IEEE Std 802.11a-1999*, 1999.
- [56] "IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part Ii: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", *IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11, 1999 Edn. (Reaff 2003) as amended by IEEE Stds 802.11a-1999, 802.11b-1999, 802.11b-1999/Cor 1-2001, and 802.11d-2001)*, p. 67, 2003.
- [57] L. Couto, J. P. Barraca, S. Sargento, and R. L. Aguiar, "FastM: Design and Evaluation of a Fast Mobility Mechanism for Wireless Mesh Networks", *International Journal On Advances in Networks and Services*, vol. 2, no. 9, Feb. 2009.
- [58] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers", in *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, ser. SIGCOMM '94, London, United Kingdom: ACM, 1994, pp. 234–244, ISBN: 0-89791-682-4.
- [59] D. Aguayo, J. Bicket, and R. Morris. (2007). SrcRR: A High Throughput Routing Protocol for 802.11 Mesh Networks (DRAFT).
- [60] R. Draves, J. Padhye, and B. Zill, "The architecture of the Link Quality Source Routing Protocol", Microsoft Research, Tech. Rep. MSR-TR-2004-57, 2004.
- [61] J. Jun and M. Sichitiu, "MRP: Wireless Mesh Networks Routing Protocol", *Computer Communications*, vol. 31, no. 7, pp. 1413–1435, 2008.
- [62] C. Santiv  n  ez and R. Ramanathan, "Hazy Sighted Link State (HSLS) Routing: A Scalable Link State Algorithm", Internetwork Research Department, BBN Technologies, Tech. Rep. 86563005, Rev. March 2003, Aug. 2001.
- [63] D. Johnson, N. Ntlatlapa, and C. Aichele, "A Simple Pragmatic Approach to Mesh Routing Using BATMAN", in *Proceedings of the 2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, Pretoria, South Africa, 2008, ISBN: 978-84-612-5570-2.
- [64] J. Friginal, D. de Andr  s, J.-C. Ruiz, and P. Gil, "Towards benchmarking routing protocols in wireless mesh networks", *Ad Hoc Networks*, vol. 9, no. 8, pp. 1374–1388, Nov. 2011, ISSN: 1570-8705.
- [65] M. H. Rehmani, A. C. Viana, H. Khalife, and S. Fdida, "SURF: A Distributed Channel Selection Strategy for Data Dissemination in Multi-Hop Cognitive Radio Networks", INRIA, Research Report RR-7628, May 2011.
- [66] J. Zhang, W. Li, Z. Yin, S. Liu, and X. Guo, "Forest Fire Detection System based on Wireless Sensor Network", in *Proceedings of the 4th IEEE Conference on Industrial Electronics and Applications*, May 2009, pp. 520 -523.

- [67] D. Johnson, Y. Hu, and D. Maltz, *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*, RFC 4728 (Experimental), Internet Engineering Task Force, Feb. 2007.
- [68] D. Harrington, R. Presuhn, and B. Wijnen, *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*, RFC 3411 (Standard), Updated by RFCs 5343, 5590, Internet Engineering Task Force, Dec. 2002.
- [69] *Web-Based Enterprise Management*, <http://www.dmtf.org/standards/wbem>, [Online; accessed on December 2012], 2011.
- [70] B. Claise, *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*, RFC 5101 (Proposed Standard), Internet Engineering Task Force, Jan. 2008.
- [71] J. Postel, *Internet Protocol*, RFC 791 (Standard), Updated by RFC 1349, Internet Engineering Task Force, Sep. 1981.
- [72] R. Enns, *NETCONF Configuration Protocol*, RFC 4741 (Proposed Standard), Obsoleted by RFC 6241, Internet Engineering Task Force, Dec. 2006.
- [73] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, *The COPS (Common Open Policy Service) Protocol*, RFC 2748 (Proposed Standard), Updated by RFC 4261, Internet Engineering Task Force, Jan. 2000.
- [74] J. Huang, B. Zhang, G. Li, X. Gao, and Y. Li, "Challenges to the New Network Management Protocol: NETCONF", in *Proceedings of the 1st International Workshop on Education Technology and Computer Science*, vol. 1, Mar. 2009, pp. 832 -836.
- [75] A. Al-Maashri and M. Ould-Khaoua, "Performance Analysis of MANET Routing Protocols in the Presence of Self-Similar Traffic", in *Proceedings of the 31st IEEE Conference on Local Computer Networks*, Nov. 2006, pp. 801 -807.
- [76] P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology", *IBM Corporation*, vol. 15, pp. 1-39, 2001.
- [77] J. Kephart and D. Chess, "The Vision of Autonomic Computing", *IEEE Computer*, vol. 36, no. 1, Jan. 2003.
- [78] E. Mollick, "Establishing Moore's Law", *IEEE Ann. Hist. Comput.*, vol. 28, pp. 62-75, 3 Jul. 2006, ISSN: 1058-6180.
- [79] D. Crockford, *The application/json Media Type for JavaScript Object Notation (JSON)*, RFC 4627 (Informational), Internet Engineering Task Force, Jul. 2006.
- [80] D. Booth and C. K. Liu, *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*, World Wide Web Consortium, Recommendation REC-wsdl20-primer-20070626, Jun. 2007.
- [81] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle, "A Survey of Autonomic Network Architectures and Evaluation Criteria", *IEEE Communications Surveys Tutorials*, no. 99, pp. 1-27, 2011, ISSN: 1553-877X.
- [82] J. Lee M.S. Bash Christophe and C. Tschudin, "A Case Study in Designing an Autonomic Wireless Mesh Network", in *Proceedings of the Scandinavian Workshop on Wireless Ad-hoc Networks*, 2007.
- [83] P. Mockapetris, *Domain names - concepts and facilities*, RFC 1034 (Standard), Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, Internet Engineering Task Force, Nov. 1987.
- [84] R. Droms, *Dynamic Host Configuration Protocol*, RFC 2131 (Draft Standard), Updated by RFCs 3396, 4361, 5494, Internet Engineering Task Force, Mar. 1997.
- [85] P. Srisuresh and M. Holdrege, *IP Network Address Translator (NAT) Terminology and Considerations*, RFC 2663 (Informational), Internet Engineering Task Force, Aug. 1999.

- [86] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", in *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, Aug. 2001.
- [87] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet", in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '03, Karlsruhe, Germany: ACM, 2003, pp. 3–10, ISBN: 1-58113-735-4.
- [88] L. Stojanovic, J. Schneider, A. Maedche, S. Libischer, R. Studer, T. Lump, A. Abecker, G. Breiter, and J. Dinger, "The role of ontologies in autonomic computing systems", *IBM Syst. J.*, vol. 43, no. 3, pp. 598–616, Jun. 2004, ISSN: 0018-8670.
- [89] *Trusted Platform Module (TPM) Specifications*, <https://www.trustedcomputinggroup.org/specs/TPM>, [Online; accessed on December 2012], 2007.
- [90] R. Wies, "Using a Classification of Management Policies for Policy Specification and Policy Transformation", in *Proceedings of the IFIP/IEEE International Symposium on Integrated network Management*, Chapman & Hall, 1995, pp. 44–56.
- [91] N. Damianou, "A Policy Framework for Management of Distributed Systems", PhD thesis, Imperial College London, Mar. 2002.
- [92] S. Davy, K. Barrett, B. Jennings, and S. v. d. Meer, "On the use of Policy Based Management for Pervasive m-Government Services", in *Proceedings of the Euro mGov 2005 The First European Mobile Government Conference*, Mobile Government Consortium International LLC, 2005, pp. 110–121.
- [93] Z. J. Fu and S. F. Wu, "Automatic Generation of IPsec/VPN Security Policies In an Intra-Domain Environment", in *Proceedings of the 12th International Workshop on Distributed System Operation & Management (DSOM 2001)*, 2001, pp. 279–290.
- [94] R. Yavatkar, D. Pendarakis, and R. Guerin, *A Framework for Policy-based Admission Control*, RFC 2753 (Informational), Internet Engineering Task Force, Jan. 2000.
- [95] K. Nichols, S. Blake, F. Baker, and D. Black, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, RFC 2474 (Proposed Standard), Updated by RFCs 3168, 3260, Internet Engineering Task Force, Dec. 1998.
- [96] 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*, RFC 3198 (Informational), Internet Engineering Task Force, Nov. 2001.
- [97] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, *Policy Core Information Model – Version 1 Specification*, RFC 3060 (Proposed Standard), Updated by RFC 3460, Internet Engineering Task Force, Feb. 2001.
- [98] K. R. Dittrich, S. Gatzju, and A. Geppert, "The Active Database Management System Manifesto: A Rulebase of ADBMS Features.", in *Rules in Database Systems*, Sellis, Timos K., Ed., ser. Lecture Notes in Computer Science, vol. 985, Springer, 1995, pp. 3-20, ISBN: 3-540-60365-4.
- [99] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "Ponder: A Language for Specifying Security and Management Policies for Distributed Systems", Imperial College, Policy Research Group, Tech. Rep. DoC 2000-1, 2000.
- [100] "CIM Simplified Policy Language (CIM-SPL)", Distributed Management Task Force, Specification Version 1.0.0, 2009.
- [101] J. D. Moffett and M. S. Sloman, "The Representation of Policies as System Objects", in *Proceedings of the Conference on Organizational Computing Systems*, ACM Press, 1991, pp. 171–184.
- [102] D. A. Marriott, "Policy Service for Distributed Systems", in *Proceedings of the IEEE Third International Workshop on Services in Distributed and Networked Environments*, 1997, pp. 2–9.

- [103] "The Ponder Policy Based Management Toolkit", Imperial College, Policy Research Group, Tech. Rep., Aug. 2002.
- [104] Distributed Management Task Force, Inc., *CIM Schema: Version 2.34.0*, DMTF Standard, 2011.
- [105] "Unified Modeling Language 2.4 Super-Structure Specification", Object Management Group, Specification Version 2.4, 2011.
- [106] *CIM Device Model White Paper, CIM Version 2.7*, White paper, Jun. 19, 2003.
- [107] D. Ferraiolo and R. Kuhn, "Role-Based Access Control", in *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, 1992, pp. 554–563.
- [108] B. J. Biddle, *Role Theory: Expectations, Identities and Behaviors*, English. Academic Press, New York, 1979, ISBN: 012095950.
- [109] D. F. Ferraiolo, R. D. Kuhn, and R. Chandramouli, *Role-Based Access Control, Second Edition*. Norwood, MA, USA: Artech House, Inc., 2007, ISBN: 1596931132.
- [110] E. Freudenthal and V. Karamcheti, "DisCo: Middleware for Securely Deploying Decomposable Services in Partly Trusted Environments", in *Proceedings of the IEEE 24th International Conference on Distributed Computing Systems*, ser. ICDCS '04, Washington, DC, USA: IEEE Computer Society, 2004, pp. 494–503.
- [111] M. V. Tripunitara and B. Carburnar, "Efficient access enforcement in distributed role-based access control (RBAC) deployments", in *Proceedings of the Symposium on Access Control Models and Technologies*, 2009, pp. 155–164.
- [112] S. Fugkeaw, P. Manpanpanich, and S. Juntapremjitt, "Agent Computing and Multi-Agent Systems", in A. Ghose, G. Governatori, and R. Sadananda, Eds., Berlin, Heidelberg: Springer-Verlag, 2009, ch. Achieving DRBAC Authorization in Multi-trust Domains with MAS Architecture and PMI, pp. 339–348, ISBN: 978-3-642-01638-7.
- [113] J. Li, J. Huai, C. Hu, and Z. Yanmin, "A Secure Collaboration Service for Dynamic Virtual Organizations", *Information Sciences*, vol. 180, no. 17, pp. 3086 - 3107, 2010, ISSN: 0020-0255.
- [114] T. Berners-Lee, R. Fielding, and L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, RFC 3986 (Standard), Internet Engineering Task Force, Jan. 2005.
- [115] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content", *Communications of the ACM*, vol. 55, no. 1, pp. 117–124, Jan. 2012, ISSN: 0001-0782.
- [116] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft, "A socio-aware overlay for publish/subscribe communication in delay tolerant networks", in *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, ser. MSWiM '07, Chania, Crete Island, Greece: ACM, 2007, pp. 225–234, ISBN: 978-1-59593-851-0.
- [117] L. Ramaswamy, B. Gedik, and L. Liu, "A Distributed Approach to Node Clustering in Decentralized Peer-to-Peer Networks", *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 9, pp. 814–829, Sep. 2005, ISSN: 1045-9219.
- [118] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach", in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, Mar. 2004.
- [119] Y. Zhang, J. Zhao, G. Cao, and C. R. Das, "On Interest Locality in Content-Based Routing for Large-scale MANETs", in *Proceedings of the 6th IEEE International Conference on Mobile Adhoc and Sensor Systems*, Oct. 2009, pp. 178 -187.
- [120] *Cleveland Freent - CaseWesternReserve Wiki*, https://wiki.case.edu/Cleveland_Freent, [Online; accessed on December 2012], 2012.

- [121] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616 (Draft Standard), Updated by RFCs 2817, 5785, 6266, Internet Engineering Task Force, Jun. 1999.
- [122] J. Oikarinen and D. Reed, *Internet Relay Chat Protocol*, RFC 1459 (Experimental), Updated by RFCs 2810, 2811, 2812, 2813, Internet Engineering Task Force, May 1993.
- [123] J. Ishmael, S. Bury, D. Pezaros, and N. Race, “Deploying Rural Community Wireless Mesh Networks”, *Internet Computing, IEEE*, vol. 12, no. 4, pp. 22–29, 2008.
- [124] *Seattle Wireless*, <http://seattlewireless.net>, [Online; accessed on December 2012], 2012.
- [125] *Personal Telco*, <http://www.personaltelco.com>, [Online; accessed on December 2012], 2012.
- [126] *Guifi.net*, <http://guifi.net/>, [Online; accessed on December 2012], 2012.
- [127] *Freifunk*, <http://start.freifunk.net/>, [Online; accessed on December 2012], 2012.
- [128] *FunkFeuer.At*, <http://www.funkfeuer.at>, [Online; accessed on December 2012], 2012.
- [129] *Athens Wireless Metropolitan Network*, <http://www.awmn.net/>, [Online; accessed on December 2012], 2012.
- [130] *Fon Blog*, <http://blog.fon.com/en/>, [Online; accessed on December 2012].
- [131] P. A. Frangoudis, G. C. Polyzos, and V. P. Kemerlis, “Wireless Community Networks: an Alternative Approach for Nomadic Broadband Network Access”, *IEEE Communications Magazine*, vol. 49, no. 5, pp. 206–213, 2011.
- [132] D. S. J. Couto, “High-Throughput Routing For Multi-Hop Wireless Networks”, PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science, Jun. 2004.
- [133] A. Raafat, M. Fathy, A. Yehia, and M. Azer, “Cooperation Incentives in Wireless Ad-hoc Networks”, in *Proceedings of the 2nd International Conference on Education Technology and Computer*, vol. 5, Jun. 2010, pp. 332–337.
- [134] *FON - wifi community*, <http://www.fon.com>, [Online; accessed on December 2012], 2012.
- [135] N. Niebert, M. Prytz, A. Schieder, N. Papadoglou, L. Eggert, F. Pittmann, and C. Prehofer, “Ambient networks: a framework for future wireless internetworking”, in *Proceedings of the IEEE 61st Vehicular Technology Conference*, vol. 5, May 1, 2005, pp. 2969–2973.
- [136] E. Grampin, J. Baliosian, J. Visca, M. Giachino, and L. Vidal, “Wireless Network Architecture for Digital Inclusion in Rural Environments”, in *Proceedings of the 1st International Global Information Infrastructure Symposium*, Jul. 2007, pp. 20–26.
- [137] R. Stewart, *Stream Control Transmission Protocol*, RFC 4960 (Proposed Standard), Updated by RFC 6096, Internet Engineering Task Force, Sep. 2007.
- [138] *Witmate*, *The Only one pure Logic/Rule Engine executable form J2ME to J2SE/J2EE*, <http://www.witmate.com/>, [Online; accessed on December 2012], 2012.
- [139] *JSR-000094 Java Rule Engine API*, <http://www.jcp.org/en/jsr/detail?id=94>, [Online; accessed on December 2012], 2012.
- [140] 3GPP, “General Packet Radio Service (GPRS); Service description; Stage 2”, 3rd Generation Partnership Project (3GPP), TS 23.060, Sep. 2008. [Online]. Available: <http://www.3gpp.org/ftp/Specs/html-info/23060.htm>.
- [141] *IST WIP - An All-Wireless Mobile Network Architecture*, <http://www.ist-wip.org>, [Online; accessed on December 2012], 2012.

- [142] N. Negroponte, "Being Wireless", *Wired Magazine*, vol. 10, 2002.
- [143] P. S. Mogre, M. Hollick, C. Schwingenschlögl, and R. Steinmetz, "Qos architecture for efficient bandwidth management in the ieeee 802.16 mesh mode", in *Review Literature And Arts Of The Americas*, Y. Xiao, Ed. Auerbach Publications, CRC Press, 2006.
- [144] A. Satsiou and L. Tassiulas, "A trust-based exchange framework for multiple services in p2p systems", in *Proceedings of the 7th IEEE International Conference on Peer-to-Peer Computing*, ser. P2P '07, Washington, DC, USA: IEEE Computer Society, 2007, pp. 45–52, ISBN: 0-7695-2986-0.
- [145] F. Rousseau, Y. Grunenberger, V. Untz, E. Schiller, P. Starzetz, F. Theoleyre, M. Heusse, O. Alphand, and A. Duda, "An Architecture for Seamless Mobility in Spontaneous Wireless Mesh Networks", in *Proceedings of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*, ser. MobiArch '07, Kyoto, Japan: ACM, 2007, 2:1–2:8, ISBN: 978-1-59593-784-1.
- [146] L. Buttyán and J.-P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks", *Mob. Netw. Appl.*, vol. 8, no. 5, pp. 579–592, Oct. 2003, ISSN: 1383-469X.
- [147] J. P. Barraca, S. Sargento, and R. L. Aguiar, "The Polynomial-Assisted Ad Hoc Charging Protocol", in *Proceedings of the 10th IEEE Symposium on Computers and Communications*, ser. ISCC '05, Washington, DC, USA: IEEE Computer Society, 2005, pp. 945–952, ISBN: 0-7695-2373-0.
- [148] Z. Han and H. V. Poor, "Coalition games with cooperative transmission: a cure for the curse of boundary nodes in selfish packet-forwarding wireless networks", *Trans. Comm.*, vol. 57, no. 1, pp. 203–213, Jan. 2009, ISSN: 0090-6778.
- [149] IST-WIP Consortium, "Radio Internet: Global architecture for an allwireless global Internet", Tech. Rep. WIP D1.4, 2008.
- [150] —, "Solutions - Mesh Networking, Multi-hop Relaying, Crosslayer Design, Communities, Operator/-cellular Assistance", IST-WIP Consortium, Tech. Rep. WIP D1.3, 2007.
- [151] *Detroit Digital Justice Coalition: Wireless Mesh*, <http://detroitdjc.org/wireless-mesh/>, [Online; accessed on December 2012].
- [152] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, RFC 2119 (Best Current Practice), Internet Engineering Task Force, Mar. 1997.
- [153] Q. Zheng, P. Zhu, Y. Wang, and M. Xu, "EPSP: Enhancing Network Protocol with Social-Aware Plane", in *Proceedings of the IEEE/ACM International Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec. 2010, pp. 578–583.
- [154] A. Tasch and O. Brakel, "Location based community services: new services for a new type of Web communities.", in *Proceedings of the IADIS Conference on Web Based Communities*, 2004.
- [155] 3GPP, "Network architecture", 3rd Generation Partnership Project (3GPP), TS 23.002, Sep. 2008. [Online]. Available: <http://www.3gpp.org/ftp/Specs/html-info/23002.htm>.
- [156] —, "General UMTS Architecture", 3rd Generation Partnership Project (3GPP), TS 23.101, Jun. 2007. [Online]. Available: <http://www.3gpp.org/ftp/Specs/html-info/23101.htm>.
- [157] N. Prigent, C. Bidan, J. Andreaux, and O. Heen, "Secure long term communities in ad hoc networks", in *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, ser. SASN '03, Fairfax, Virginia: ACM, 2003, pp. 115–124, ISBN: 1-58113-783-4.
- [158] S. Kosta, A. Mei, and S. J., "Small World in Motion (SWIM): Modeling Communities in Ad-Hoc Mobile Networking", in *Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks*, Jun. 2010, pp. 1–9.

- [159] Filiposka, S.V. and Trajanov, D.R. and Grnarov A.L., "Performance analysis of scale-free communities in ad hoc networks", in *Proceedings of the 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology*, May 2009, pp. 399-403.
- [160] H. Pan, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks", *IEEE Transactions on Mobile Computing*, vol. 10, no. 11, pp. 1576 -1589, Nov. 2011, ISSN: 1536-1233.
- [161] *Apple Bonjour*, <http://www.apple.com/support/bonjour/>, [Online; accessed on December 2012].
- [162] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults", *Journal of ACM*, vol. 27, no. 2, pp. 228-234, Apr. 1980, ISSN: 0004-5411.
- [163] G. Vasilakis, G. Perantinos, I. Askoxylakis, N. Mechin, V. Spitadakis, and A. Traganitis, "Business Opportunities and Considerations on Wireless Mesh Networks", in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops*, Jun. 2009, pp. 1-8.
- [164] *Population Reference Bureau*, <http://www.prb.org>, [Online; accessed on December 2012], 2012.
- [165] J. Baliosian, J. Visca, E. Grampín, L. Vidal, and M. Giachino, "A Rule-based Distributed System for Self-optimization of Constrained Devices", in *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, ser. IM'09, New York, NY, USA: IEEE Press, 2009, pp. 41-48, ISBN: 978-1-4244-3486-2.
- [166] K. Jones and L. Liu, "What Where Wi: An Analysis of Millions of Wi-Fi Access Points", in *Proceedings of the IEEE International Conference on Portable Information Devices, PORTABLE07*, May 2007, pp. 1-4.
- [167] ITU, "Network and Customer Installation Interfaces- Asymmetric Digital Subscriber Line (ADSL) Metallic Interface, ANSI T1.413-1998", International Telecommunication Union, Tech. Rep., 1998.
- [168] S. Aoyagi, M. Takizawa, M. Saito, H. Aida, and H. Tokuda, "ELA: a fully distributed VPN system over peer-to-peer network", in *Proceedings of The 2005 Symposium on Applications and the Internet*, Jan. 2005, pp. 89 - 92.
- [169] *OpenID Authentication 2.0 - Final*, <http://openid.net/specs/openid-authentication-2.0.html>, [Online; accessed on December 2012], 2007.
- [170] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Onion Routing: Making a Traffic Analysis Resistant Network Robust and Fault-Tolerant", *US Patent Office*, 1998.
- [171] M. Lima, A. dos Santos, and G. Pujolle, "A survey of survivability in mobile ad hoc networks", *Communications Surveys Tutorials, IEEE*, vol. 11, no. 1, pp. 66-77, 2009, ISSN: 1553-877X.
- [172] P. Antoniadis and B. Le Grand, "Incentives for Resource Sharing in Self-Organized Communities: from Economics to Social Psychology", in *Proceedings of the Workshop on Dynamic Communities: from Connectivity to Information Society*, IEEE Computer Society, 2007.
- [173] E. Wenger, R. McDermott, and W. M. Snyder, *Cultivating Communities of Practice*, 1st ed. Boston, USA: Harvard Business School Press, Mar. 15, 2007.
- [174] N. Hara, P. Shachaf, and S. Stoerger, "Online Communities of Practice Typology Revisited", *Journal of Information Science*, vol. 35, no. 6, pp. 740-757, Dec. 2009, ISSN: 0165-5515.
- [175] L. Lamport, "The Part-time Parliament", *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 133-169, May 1998, ISSN: 0734-2071.
- [176] D. Meyer, L. Zhang, and K. Fall, *Report from the IAB Workshop on Routing and Addressing*, RFC 4984 (Informational), Internet Engineering Task Force, Sep. 2007.
- [177] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, *Host Identity Protocol*, RFC 5201 (Experimental), Updated by RFC 6253, Internet Engineering Task Force, Apr. 2008.

- [178] C. Perkins, *IP Mobility Support for IPv4, Revised*, RFC 5944 (Proposed Standard), Internet Engineering Task Force, Nov. 2010.
- [179] C. Perkins, D. Johnson, and J. Arkko, *Mobility Support in IPv6*, RFC 6275 (Proposed Standard), Internet Engineering Task Force, Jul. 2011.
- [180] P. Leach, M. Mealling, and R. Salz, *A Universally Unique Identifier (UUID) URN Namespace*, RFC 4122 (Proposed Standard), Internet Engineering Task Force, Jul. 2005.
- [181] S. Cheshire, B. Aboba, and E. Guttman, *Dynamic Configuration of IPv4 Link-Local Addresses*, RFC 3927 (Proposed Standard), Internet Engineering Task Force, May 2005.
- [182] S. Thomson, T. Narten, and T. Jinmei, *IPv6 Stateless Address Autoconfiguration*, RFC 4862 (Draft Standard), Internet Engineering Task Force, Sep. 2007.
- [183] Z. Fan and S. Subramani, "An address autoconfiguration protocol for ipv6 hosts in a mobile ad hoc network", *Comput. Commun.*, vol. 28, no. 4, pp. 339–350, Mar. 2005, ISSN: 0140-3664.
- [184] M. Patrick, *DHCP Relay Agent Information Option*, RFC 3046 (Proposed Standard), Internet Engineering Task Force, Jan. 2001.
- [185] A. Ganguly, D. Wolinsky, P. O. Boykin, and R. Figueiredo, "Decentralized dynamic host configuration in wide-area overlays of virtual workstations", in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2007, pp. 1–8.
- [186] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, RFC 3261 (Proposed Standard), Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, Internet Engineering Task Force, Jun. 2002.
- [187] P. Mockapetris, *Domain names - implementation and specification*, RFC 1035 (Standard), Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, Internet Engineering Task Force, Nov. 1987.
- [188] J. Postel, *Domain Name System Structure and Delegation*, RFC 1591 (Informational), Internet Engineering Task Force, Mar. 1994.
- [189] S. Cheshire and M. Krochmal, "Multicast DNS", Internet Engineering Task Force, Internet-Draft draft-cheshire-dnsext-multicastdns-15, Dec. 2011.
- [190] R. Herriot, S. Butler, P. Moore, and R. Turner, *Internet Printing Protocol/1.0: Encoding and Transport*, RFC 2565 (Experimental), Obsoleted by RFC 2910, Internet Engineering Task Force, Apr. 1999.
- [191] A. Gulbrandsen, P. Vixie, and L. Esibov, *A DNS RR for specifying the location of services (DNS SRV)*, RFC 2782 (Proposed Standard), Internet Engineering Task Force, Feb. 2000.
- [192] F. Proto and C. Pisa, "The OLSR mDNS Extension for Service Discovery", in *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009*, Jun. 2009, pp. 1–3.
- [193] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi, *DNS Extensions to Support IP Version 6*, RFC 3596 (Draft Standard), Internet Engineering Task Force, Oct. 2003.
- [194] C. Everhart, L. Mamakos, R. Ullmann, and P. Mockapetris, *New DNS RR Definitions*, RFC 1183 (Experimental), Updated by RFCs 5395, 5864, 6195, Internet Engineering Task Force, Oct. 1990.
- [195] C. Davis, P. Vixie, T. Goodwin, and I. Dickinson, *A Means for Expressing Location Information in the Domain Name System*, RFC 1876 (Experimental), Internet Engineering Task Force, Jan. 1996.
- [196] J. Hähner, D. Dudkowski, P. J. Marrón, and K. Rothermel, "A quantitative analysis of partitioning in mobile ad hoc networks", in *Proceedings of the joint international conference on Measurement and modeling of*

- computer systems*, ser. SIGMETRICS '04/Performance '04, New York, NY, USA: ACM, 2004, pp. 400–401, ISBN: 1-58113-873-3.
- [197] J. d. Bruijn, M. Ehrig, C. Feier, F. Martín-Recuerda, F. Scharffe, and M. Weiten, “Ontology Mediation, Merging and Aligning”, in *Semantic Web Technologies*, Wiley, 2006.
- [198] W. Nejdl, D. Olmedilla, M. Winslett, and C. C. Zhang, “Ontology-Based Policy Specification and Management”, in *Proceedings of the 2nd European Semantic Web Conference (ESWC)*, Springer, 2005, pp. 290–302.
- [199] R. Rivest, *The MD5 Message-Digest Algorithm*, RFC 1321 (Informational), Updated by RFC 6151, Internet Engineering Task Force, Apr. 1992.
- [200] D. Eastlake 3rd and T. Hansen, *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)*, RFC 6234 (Informational), Internet Engineering Task Force, May 2011.
- [201] J. Callas, L. Donnerhacker, H. Finney, D. Shaw, and R. Thayer, *OpenPGP Message Format*, RFC 4880 (Proposed Standard), Updated by RFC 5581, Internet Engineering Task Force, Nov. 2007.
- [202] International Telecommunication Union - ITU, “X.509 (10/12) Information technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks”, ITU Recommendation, Oct. 2012.
- [203] L. Erman, F. Hayes-Roth, V. Lesser, and D. Reddy, “The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty”, *Blackboard Systems*, T. M. R. Englemore, Ed., pp. 31–86, 1988.
- [204] M. Devera, *HTB*, <http://luxik.cdi.cz/~devik/qos/htb/>, [Online; accessed on December 2012].
- [205] R. Thayer, N. Doraswamy, and R. Glenn, *IP Security Document Roadmap*, RFC 2411 (Informational), Obsoleted by RFC 6071, Internet Engineering Task Force, Nov. 1998.
- [206] K. Zeilenga, *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map*, RFC 4510 (Proposed Standard), Internet Engineering Task Force, Jun. 2006.
- [207] K. Wehrle, M. Günes, and J. Gross, Eds., *Modeling and Tools for Network Simulation*. Springer, 2010, ISBN: 978-3-642-12330-6.
- [208] S. Mehta, N. Sulatan, and K. S. Kwak, “Network and System Simulation Tools for Next Generation Networks: a Case Study”, *Modelling, Simulation and Identification*, Azah Mohamed, Ed., 2010.
- [209] R. Bagrodia and M. Takai, “Position Paper on Validation of Network simulation models”, in *Proceedings of the DARPA/NIST Network Simulation Validation Workshop*, May 1999.
- [210] S. Kurkowski, T. Camp, and M. Colagrosso, “MANET simulation studies: the incredibles.”, *Mobile Computing and Communications Review*, vol. 9, no. 4, pp. 50-61, Mar. 17, 2005.
- [211] T. Nitsche and T. Fuhrmann, “A Tool for Raytracing Based Radio Channel Simulation”, in *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '11, Barcelona, Spain: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011, pp. 69–74, ISBN: 978-1-936968-00-8.
- [212] G. Judd and P. Steenkiste, “Repeatable and realistic wireless experimentation through physical emulation”, *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 63–68, Jan. 2004, ISSN: 0146-4833.
- [213] D. A. Maltz, J. Broch, and D. B. Johnson, “Experiences Designing and Building a Multi-hop Wireless Ad hoc Network Testbed”, CMU School of Computer Science, Tech. Rep. CMU-CS-99-116, Mar. 1999, pp. 99–116.
- [214] I. Broustis, J. Eriksson, S. Krishnamurthy, and M. Faloutsos, “A Blueprint for a Manageable and Affordable Wireless Testbed: Design, Pitfalls and Lessons Learned”, in *Proceedings of the 3rd International ICST*

- Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities*, May 2007, pp. 1–6.
- [215] B. A. Chambers, “The Grid Roofnet: a Rooftop Ad Hoc Wireless Network”, PhD thesis, Massachusetts Institute of Technology, 2002.
- [216] M. Ott, I. Seskar, R. Siraccusa, and M. Singh, “ORBIT Testbed Software Architecture: Supporting Experiments as a Service.”, in *Proceedings of the International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, IEEE Computer Society, Apr. 25, 2005, pp. 136-145, ISBN: 0-7695-2219-X.
- [217] *Onelab - Future Internet Testbeds*, <http://www.onelab.eu>, [Online; accessed on December 2012], 2012.
- [218] L. Peterson, T. Anderson, C. David, and R. Timothy, “A blueprint for introducing disruptive technology into the Internet”, *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, Jan. 2003, ISSN: 0146-4833.
- [219] M. Kropff, T. Krop, M. Hollick, P. Mogre, and R. Steinmetz, “A survey on real world and emulation testbeds for mobile ad hoc networks”, in *Proceedings of the 2nd International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2006.
- [220] *XiPi: Shining a Light on Infrastructures for the Benefit of the Future Internet Community*, <http://xipi.eu>, [Online; accessed on December 2012], 2012.
- [221] S. L. Shrestha, A. Lee, J. Lee, D.-W. Seo, K. Lee, J. Lee, S. Chong, and N. H. Myung, “A Group of People Acts like a Black Body in a Wireless Mesh Network”, in *Proceedings of the IEEE Global Telecommunications Conference*, Nov. 2007, pp. 4834 -4839.
- [222] *Universal Serial Bus Revision 3.0 specification*, 3.0, [Online; accessed on December 2012], Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, NEC Corporation, ST-NXP Wireless, and Texas Instruments, 2010.
- [223] *IEC 60529 Degrees of Protection Provided by Enclosures (IP Code)*, 2.1, ser. International Standard. International Electrotechnical Commission, Nov. 2001.
- [224] *Boost C++ Libraries*, <http://www.boost.org>, [Online; accessed on December 2012].
- [225] *Mono: Cross Platform, Open Source .NET Development Framework*, <http://www.boost.org>, [Online; accessed on December 2012].
- [226] N. Moore, *Optimistic Duplicate Address Detection (DAD) for IPv6*, RFC 4429 (Proposed Standard), Internet Engineering Task Force, Apr. 2006.
- [227] S. Deering and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 2460 (Draft Standard), Updated by RFCs 5095, 5722, 5871, Internet Engineering Task Force, Dec. 1998.
- [228] P. Deutsch, *DEFLATE Compressed Data Format Specification version 1.3*, RFC 1951 (Informational), Internet Engineering Task Force, May 1996.
- [229] M. Ye, W. Dong, and X. Sun, “Hardware ID Auto-assigning Mechanism for Embedded Wireless Network”, in *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology*, Aug. 2009, pp. 500-504.
- [230] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the xor metric”, *Peer-to-Peer Systems*, pp. 53–65, 2002.
- [231] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, *The Kerberos Network Authentication Service (V5)*, RFC 4120 (Proposed Standard), Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113, Internet Engineering Task Force, Jul. 2005.
- [232] U. Ashraf, S. Abdellatif, and G. Juanole, “Gateway Selection in Backbone Wireless Mesh Networks”, in *Proceedings of the IEEE Wireless Communications and Networking Conference*, Apr. 2009, pp. 1 -6.

- [233] *OLSRd - an Ad-hoc Wireless Mesh Routing Daemon*, <http://www.olsr.org>, [Online; accessed on December 2012].
- [234] A. Ulrich, R. Holz, P. Hauck, and G. Carle, "Investigating the openPGP Web of Trust", in *Proceedings of the 16th European Conference on Research in Computer Security*, ser. ESORICS'11, Leuven, Belgium: Springer-Verlag, 2011, pp. 489–507, ISBN: 978-3-642-23821-5.