Delft University of Technology

Wireless and Mobile Communication (WMC) Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Flexible and Adaptable Service Provisioning for Federated Personal Networks

Assignment: Master Thesis
Instructors: Jinglong Zhou
Malohat Ibrohimovna
Supervisor: Prof. Dr.Sonia Heemstra de Groot
Student: Bing Liu (1532502)

Acknowledgement

I would like to express my deepest appreciation to my instructors, Jinglong and Malohat, who gave me many invaluable guidance and supports over the whole process of my project. I would also like to thank them for all the time we spent together discussing and validating results, and introducing me to write the technical thesis.

Next, I would like to thank Prof. Sonia Heemstra de Groot, as my wonderful supervisor and lecturer at Delft University of Technology. I would express my gratitude to her for the supervision of this thesis.

Furthermore, I would like to thank the WMC group for giving me this opportunity to work on this interesting project and allowing me to work with all the wonderful people here. I would like to thank Martin Jacobsson who offers me precious suggestions and helps about learning the Ppand software.

Last but not least, I would like to thank my family for their support and encouragement. Also I want to say thank you to my all friends who give me supports during the whole project.

Abstract

Personal network and their federations (called Fednet) are considered as one of the promising future concepts regarding the personal communication. In this thesis, we first studies the state of the art of Fednet, from which we know there are two approaches for providing the services in Fednet, one is overlay and the other is proxy-based. Each of the approaches has the advantages and drawbacks. To trade off between these two approaches, we propose a new scheme, which can make the way of service provisioning in Fednet flexible and adaptable to the changing environment and the user's preference.

Some modules are designed in the Fednet agent (FA), Fednet manager (FM) and the service proxy to support our new mechanism. However, due to the time limitation, it is not possible for us to address all of them. Here we just focus ourselves on discussing the modules proposed in the FA, especially the policy engine, which is directly related to the service provisioning. To fully illustrate the concept of Fednet and the service provisioning in Fednet, we implement a Fednet prototype. It shows us how the Fednet is formed and how messages are exchanged between the FA and the FM. The context collector, context interpreter and the policy engine in the FA are also executed in the prototype. With the aim of proving our decision making algorithms can make decisions according to the user's requirements and the changing context, we carry out some simulations and a real test bed experiment based on one or two parameters. The simulations results show that our new service provisioning mechanism could make the way of service provisioning flexible and adaptable as we have expected.

Content

Acknowledgement **2**

Abstract **3**

1. Introduction **9**

1.1. What is New 9
1.1. Background 9
1.1.1. Personal Network 9
1.1.2. Federated Personal Network 10
1.1.3. Existing Service Provisioning Technology in Fednet 12
1.2. Problem Statements 14
1.3. Motivations, Scopes and Goals 15
1.3.1. Motivations 15
1.3.2. Scopes 16
1.3.3. Goals 16
1.4. Research Challenges 17
1.5. Methodology 18
1.6. Contributions 18
1.7. Thesis Outline 19

2. Requirements of Service Provisioning in Fednet **21**

2.1. Basic Requirements 21
2.1.1. What environments 21
2.1.2. What kinds of services 22
2.1.3. Who are the users 23
2.2. Context-aware 23
2.3. Flexible and Adaptable 23
2.4. Scalable 24
2.5. QoS 25
2.6. Summary 25

3. Survey on Service Provisioning **27**

3.1. Introduction 27
3.2. Service Provisioning in Different Networks 27
3.2.1. In Ad-hoc Network 27
3.2.2. In Peer-to-Peer Network 28
3.2.3. In Grid Network 30
3.2.4. Conclusion 31
3.4. Service Provisioning for Mobile System 32
3.4.1. Conclusion 34

3.5. Service Provisioning with Special Requirements 34
3.5.1. User-centric 34
3.5.2. Context-aware 35
3.5.3. Federated Service Provisioning 36
3.5.4. Conclusion 38
3.6. Summary 38

4. The Service Provisioning Architecture in Fednet 42

4.1. Introduction 42
4.2. The Proposed Architecture 42
4.3. Modules in Fednet Manager 43
4.3.1. Service Discovery 44
4.3.2. Service Manager 44
4.4. Modules in Fednet Agent 44
4.4.1. Service Parser 45
4.4.2. Context Collector 46
4.4.3. Context Interpreter 50
4.4.4. Policy Engine 52
4.5. Module in Service Proxy 52
4.6. Pros and Cons of Our Architecture 53
4.7. Summary 53

5. Context-aware, Flexible and Adaptable Service Provisioning 56

5.1. Introduction 56
5.2. General Scenario 57
5.3. Specific Scenario 58
5.3.1. Unfriendly Environment 58
5.3.2. Confidential Application 59
5.3.3. Short-range Scenario 60
5.4. Examples 61
5.4.1. Overlay 61
5.4.2. Proxy-based 62
5.4.3. Hybrid 63
5.5. Algorithms' Discussion 64
5.6. Architecture's Discussion 66
5.7. Summary 67

6. Fednet Prototype Implementation 69

6.1. Introduction 69
6.2. Platform 69
6.2.1. Hardware 69
6.2.2. Software 70
6.3. Implementation Prototype 71
6.3.1. Creating Application Socket 71

6.3.2. Sending and Receiving Packets 72
6.3.3. Prototype Architecture 73
6.4. Summary 79

7. Simulation and Experiment 82

7.1. Introduction 82
7.2. Simulation 82
7.2.1. Scenario 1 83
7.2.2. Scenario 2 84
7.2.3. Scenario 3 85
7.3. Real Test Experiment 86
7.3.1. LQA 87
7.3.2. Setup 88
7.3.3. Results and Analysis 89
7.4. Summary 91

8. Conclusion and Future Work 94

8.1. Conclusion 94
8.2. Future Work 95

Reference 98

Appendix A 102

Appendix B 103

Appendix C 105

Appendix D 106

1

Introduction

1.1. What is New

For the PN owners involved in forming a Fednet, there are two approaches for them to provide the services. One is overlay and the other one is proxy-based. Each of the approach has the merits and shortcomings. Then which one shall the PN owner choose? Under what situation? When the environment is changing, can the PN owner sense it? Is that smart to maintain the way of service provisioning still even after the surrounding is changed? Can the user's preference be taken into consideration when providing or consuming the services? In this thesis, we will propose a new service provisioning mechanism for Fednet to solve the problems mentioned above. The new scheme can make the way of service provisioning in Fednet flexible and adaptable to the changing environment and the PN owner's requirements, which has been verified by the implementation results and the real test bed experiment.

1.2. Background

1.2.1. Personal Network

With the rapid development of the wireless technology, people are having more and more wireless devices. To connect these digital devices for better serving the people, it leads to a new network, which can be dynamically created around the user and centered on his or her needs. Personal Area Network (PAN) is one of the examples, which maybe wired or through wireless technologies, such as Bluetooth, UWB and Zigbee etc. Typical wireless PAN communication can take place when a user requests time and location information from his Global Positioning System (GPS) receiver or transfers files from the laptop to his smart phone, as shown in Figure 1.1.

Figure 1- 1 An Example of WPAN

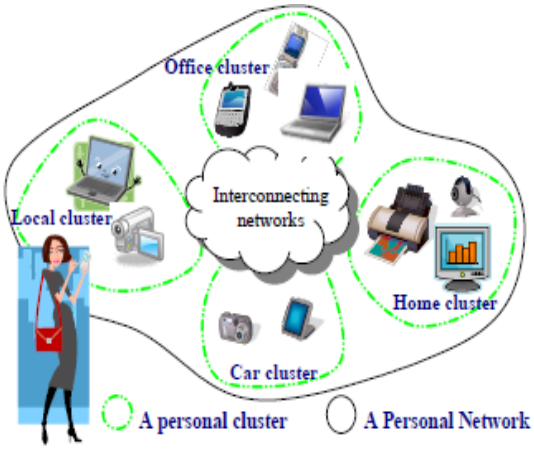

Figure 1- 2 An Example of Personal Network [1]

PAN connects a person's devices around him. But in the real scenario, the physical environment of a person could be his or her home, office, car, public places he or she visits and public transportation etc [2]. To interact the user with his environment, Personal Network (PN) is proposed to extend PAN with other devices and services farther away [3]. This extension will be made via interconnecting structure (Internet, UMTS, WLAN and Ad-hoc etc). Personal area network can be considered as one of the clusters in the personal network. A PN is self-configured, context-aware, and adaptable to the change in the surroundings. The clusters and devices communicate with each cluster regardless of their geographical locations. Figure 1.2 gives an example of a personal network.

1.2.2. Federated Personal Network

As the rapid development of the PN system and the purpose driven by group applications, personal networks will go beyond person-centric networks and federate into group-centric networks. Then the federated personal network, namely the so-called Fednet is proposed [2]. Fednet defines as a temporal, ad-hoc, opportunity or purpose-driven secure cooperation of independent PNs. The goal of the Fednet is to share and collaborate resources to achieve a specific objective, such as colleagues share internet access and documents, friends share pictures and firemen share sensor information in a disaster relief situation etc. An example of simple Fednet is illustrated in Figure 1.3, in which four friends are sharing photos.

Figure 1- 3 An Example of Fednet [4]

Fednet can be classified into purpose driven PN federations and opportunity driven PN federations, depending on the way that the Fednets are initialized. When the need for achieving a common task arises, purpose driven Fednet is formed. When the PNs of people with strong common interests detect each other and see opportunities, opportunity driven Fednet is formed.

PNs provide and consume the services with each other in Fednet by peer to peer manner. Each PN requires the following components to be able to federate with other PNs: Fednet Manager (FM, one per Fednet); Fednet Agent (FA, one per PN); Gateway (GW, one or more per PN); Fednet services (a set of PN services). Figure 1.4 shows the basic proxy-based architecture of Fednet [4].

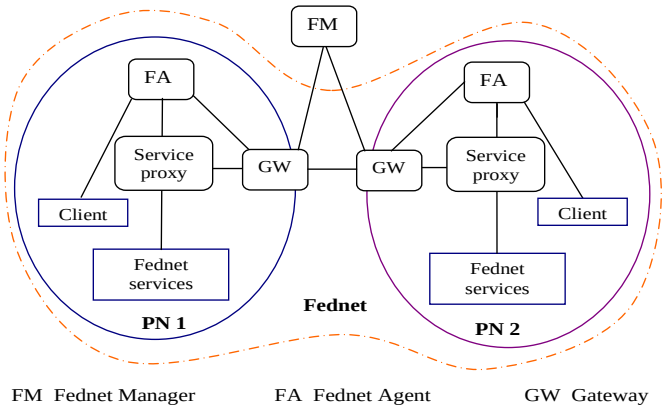

Figure 1- 4 Basic Proxy-based Architecture of Fednet [4]

The Fednet manager is the main component in the Fednet, which is responsible for creating Fednet advertisement, building service directory, providing the service lookup to Fednet members and producing decisions on the access control to the Fednet. The Fednet manager can be located either within the Fednet or outside the Fednet. In each participating PN in Fednet, there is a Fednet agent, with the main functions of managing the participation of a PN in a Fednet, creating participation profile, configuring service proxy for the members and producing access control decisions to PN services. The gateway is a device with multiple network interfaces. A PN communicates with other PNs of the Fednet through this gateway, by making one of its interfaces publicly addressable [5].

A two-level approach for the access control is proposed in Fednet. The first-level access control, also called Fednet access control, takes place when a new member joins a Fednet, carried out by the Fednet manager, which makes a centralized management of the joining and leaving Fednet members. The second-level access control takes place when a Fednet member requests a Fednet service, carried out by the Fednet agent of a PN, which is also called service access control. This allows a PN to keep the control over its personal resources and services.

1.2.3. Existing Service Provisioning Technology in Fednet

Based on the current researches, services in Fednet can be provided in two ways [1]:

(1) overlay approach, providing services directly through the personal devices, as shown in Figure 1.5

(2) proxy-based approach, services going through a specific entry point of a PN, called service proxy, as depicted in Figure 1.6.

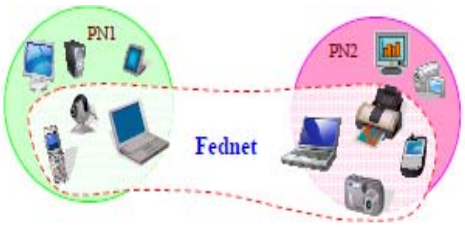

Figure 1- 5 Network Overlay between PNs

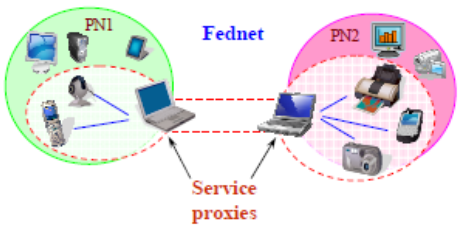

Figure 1- 6 Service Proxies between PNs [1]

In the overlay approach, all participating devices in a Fednet are directly visible and

accessible to each other, which can establish a fast access to the service. However, this approach has the following drawbacks: (1) The fast service access requires additional traffic and functionality on the network layer at all the service providing devices. (2) When new services are added to Fednet, the complexity of the service management will be raised. (3) PNs' internal structures are revealed to each other due to the engaged devices' visibility and accessibility [1].

Sometimes PN owners want to share services without exposing their personal devices. To solve this problem, proxy-based approach was proposed, in which the services are not provided directly by the devices, but through a specific entry point, called service proxy. The service proxy acts as an intermediary between the service offering and service consuming PN. This approach brings a number of advantages: (1) Each PN has a centralized control over its resources and keeps its autonomy by having a separate security domain. (2) Since the access is controlled at the borders of a PN, even a simple device, like webcam can export its services with the support of the service proxy. (3) By using the existing network functionality, no additional networking functionality is required. However, this approach also shows some drawbacks, such as delay due to the service proxies, performance degradation and bottleneck at the gateways and additional processing capacity is required at the gateways to handle the service proxy function [1].

In some cases, at the client or server side, the PN owner may prefer to use service proxy to protect his PN, while at the other side (server or client), overlay is chosen for getting first and direct service access. It is crucial to adjust the service provisioning pattern to the application, the user's context and privacy issues. To solve this problem, a hybrid approach for service provisioning in Fednet is proposed in [1]. It combines the two approaches to benefit from their advantages and to add flexibility to service provisioning process. A high-level view of the hybrid approach is shown in Figure 1.7. Actually, from each PN's point of view, it still can be considered as a single approach of just using overlay or proxy-based. Therefore, in the rest of this thesis, when talking about the service provisioning approach, we only mean overlay or proxy-based.

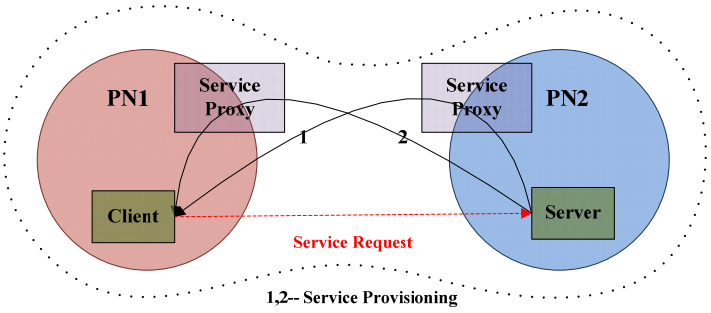

Figure 1- 7 Hybrid Approach for Service Provisioning in Fednet

1.3. Problem Statements

As we discussed above, both overlay and proxy-based approaches for service provisioning in Fednet have their advantages and drawbacks. The PN owner may ask which approach I shall choose when I am involved in the Fednet for providing some services? Furthermore, as an extension of personal network, Fednet inherits most of the PN characteristics, such as being self-organized, user-centric and context-aware. It would not be smart to provide the services always by one approach in Fednet because the context information are changing, and service provisioning should be adaptable to the changes in the surroundings. Thus, we are wondering whether there is a mechanism that can make the service provisioning in Fednet context-aware, adaptable and flexible.

[1] seems to propose a solution to our problem, in which a policy-based service provisioning architecture is designed to implement the hybrid approach, as shown in Figure 1.8. The Fednet services are classified into overlay services and proxy-based services according to a number of policies, such as device capability, service type, social context of the user and privacy. Based on the classification, the controller in the Fednet agent determines the way of service provisioning in Fednet.

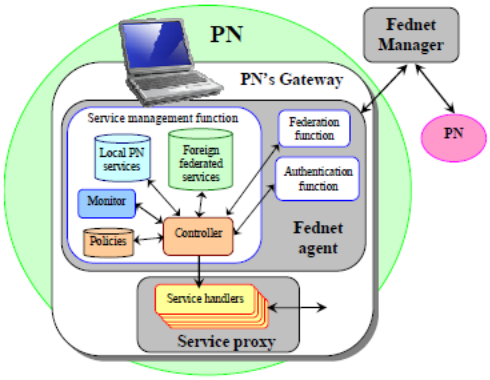

Figure 1- 8 Policy-based Service Provisioning Architecture [1]

This architecture gives us a basic idea that the policies could influence the way of service provisioning. However, it doesn't show how the policies interact with each other for determining the way. Besides that, this architecture also has some drawbacks with the following aspects. (1) There are too many modules in FA, and it is not clear when a service request takes place, which module will handle the request first, which comes next and then what will happen. (2) Where the FA gets the policies' information is not illustrated. (3) How to make the service provisioning context-aware and adaptable are not addressed.

Since the existing architecture cannot totally solve our problem, we decide to propose a new mechanism, which can allow the service provisioning in Fednet be context- aware, adaptable and trade off between the overlay and proxy-based approach.

1.4. Motivations, Scopes and Goals

In this section, we will explain why we want to propose a new mechanism for service provisioning in Fednet, what the scope of the thesis is and what targets we plan to achieve.

1.4.1. Motivations

It is predicted by the Wireless World Research Forum that 7 years from now, there will be 7 trillion wireless devices serving 7 billion people, that is, on the average 1000 wireless devices per person. The rapid increase of wireless and portable devices will result in new paradigms for service oriented computing. Personal network and their federations are examples of this, which are also considered as one of the next generation networks. Recently, some papers have

been published about relevant technologies in personal network and Fednet. But there are still some issues immature and under on-going work, service provisioning in Fednet is one of them.

Service provisioning obtains increasing attention due to the fast growth of telecom technologies and high-quality requirements of the users. Overlay and proxy-based have already been proposed for service provisioning in Fednet. However, there still needs some improvements or integrations towards the two approaches to make them fulfill the requirements of the service provisioning in Fednet. Therefore, the state of art in Fednet drives us to create a new service provisioning architecture.

1.4.2. Scopes

As can be seen from the previous sections, a lot of research and design are needed before we reach the goal. During the master project, it is hard for us to address all the issues in Fednet and service provisioning. In a consequence, we have to focus on some specific aspects.

First of all, in this thesis, the service provisioning mainly means the way of providing services. In most of the cases, when we talk about the service, the server and the client, they are all in an abstract level, not specific to any certain service or providers. Apart from that, we concentrate our design on two PNs; one is the client PN and the other is the server PN. For the Fednet formed by more PNs, we believe that it may work as the same way with the two PNs. But some scalable schemes might be needed. Some modules and parameters are proposed for designing the schemes in our new service provisioning architecture. However, only part of them are implemented or verified in the prototype because we do not have enough time to realize all, which we will leave for the future work.

1.4.3. Goals

As can be seen in the problem statement and the motivations, the goal of this thesis is to propose a new service provisioning architecture for Fednet based on the existing work, which can benefit from overlay and proxy-based approaches. Apart from that, we plan to implement a prototype of Fednet, the proposed architecture and simple service provisioning scenarios.

Then analysis and comparison of the two service provisioning approaches is required. However, before doing the design, we should ask ourselves the following questions.

1) What are the requirements for service provisioning in Fednet?
2) How do we exactly design the architecture?
3) Can the proposed architecture be implemented in reality?
4) Can our mechanism satisfy the requirements?

In the following chapters, we will figure out the above questions one by one.

1.5. Research Challenges

During the more than one year work, we have been facing a lot of troubles and challenges. The first problem we met was to collect the relevant materials. Service provisioning in Fednet is a new topic, not so many papers or schemes have been reported about it. To learn about the concept of Fednet and service provisioning, we first read lots of papers about personal network and Fednet. Then we spent almost one month on conducting a survey on service provisioning mechanisms. Defining the requirements of service provisioning in Fednet was also not very easy, because there was no reference about such an issue in Fednet, we had to refer to some related work in PN and analyze the characteristics of Fednet by ourselves.

Another challenge of the thesis was the architecture and algorithms design. After a deep research on the related work, we found out that the existing technologies are not capable of solving our problem. Therefore, we had to propose a new solution to fulfill the requirements that we listed. Since we want the service provisioning to be context-aware, what kinds of context data to collect and how to collect these data became a question for us. Besides that, the way of considering these factors was another problem, because the factors depend upon the scenarios, their weights and how they interact with each other could affect the way of service provisioning.

The biggest difficulty for us in this project was the implementation, which was based on the PN software named Ppand [3]. Reading the original codes of Ppand nearly took us two

months. After knowing how the software works, we wrote some C codes to implement the Fednet prototype and the new service provisioning architecture. Coding was a tedious job and the prototype, even the Ppand software needed to be tested to find the bugs. To verify our service provisioning can be flexible and adaptable to the changing environment, we simulated some Fednet scenarios and carried out a real test bed experiment, the setup of which required careful measurement.

1.6. Methodology

To get our research approach started, we first formulated the requirements of service provisioning in Fednet and study on the existing service provision mechanisms to get some design ideas. Then, we defined a new architecture for a potential solution. At the early stage, we made the architecture concrete by discussing about some certain components, especially the policy engine. Even though, we knew that there are lots of occasions to form a Fednet for sharing some services, we only concentrate ourselves on designing the policy engine for some specific scenarios due to the time limitation. Subsequently, we did our utmost to validate the architecture's usefulness and feasibility. After theoretical research, we were eager to implement our ideas in the real test bed. Then, the Fednet formation and some modules in our service provisioning architecture were implemented. We also carried out some simulations and experiments to illustrate that our architecture can satisfy the requirements. There are also some other issues we observed during the theory study and implementation, for them, we would take as the future work.

1.7. Contributions

The purpose of this section is to highlight our contributions of this thesis.

- We are the first one to analyze and figure out the requirements of service provisioning in Fednet, as discussed in chapter 2.
- The related work about the existing service provisioning technologies in Chapter 3 has been compiled in its entirety by us.
- The new service provisioning architecture in chapter 4 is proposed by us based on some existing modules, but the main ideas are still from us. And the architectural discussions is

our work

- The decision making algorithms for different scenarios introduced in chapter 5 are totally designed by us.
- We are the first one to implement the Fednet prototype, as shown in chapter 6.
- For chapter 7, the simulations are completely carried out by us. The link quality experiment is also tested by us, based on the existing LQA method.

1.8. Thesis Outline

The rest of the thesis is organized as follows. In chapter 2, we present the requirements of service provisioning in Fednet. Chapter 3 introduces the survey we conduct about the service provisioning technologies. By inspired by some existing modules and some related work, we propose a new service provisioning architecture, which is depicted in chapter 4. In chapter 5, we focus ourselves on designing the policy engine for some specific scenarios. To demonstrate the Fednet formation and our new service provisioning architecture, we implement a Fednet prototype in chapter 6. In chapter 7, we simulate some scenarios in the policy engine and carry out a real test experiment on the link quality parameter. Finally, in chapter 8, we summarize the whole thesis and show some directions about the future work.

2

Requirements of Service Provisioning in Fednet

Before coming up with a new solution for service provisioning in Fednet, it is important to understand what has been solved and what exactly still needs to be solved. In this chapter, we will discuss what are the requirements of providing services in Fednet and then argue why they are important. Although these requirements are described at a very high level, it is important to try to formulate them, so later on we can validate if our solution could address all or most of the requirements to a satisfactory level.

2.1. Basic Requirements

Fednet extends the boundary of personal network to involve multiple users having common interest to share resources. The interactions between the Fednet members should be secure, self-organized and confined within the subset of collaborating devices. These are the basic requirements of forming a Fednet [6].

2.1.1. Which environments

The environments of forming a Fednet and then providing services in it could be: collaborative working, virtual meetings, family networks, virtual classrooms, office network, distant learning, vehicle networks and emergency networks etc [6]. For example, in the highway, when cars are traveling sufficiently close to establish radio links between them, sensors of each individual vehicle network could be federated to share the real-time data about the hazardous road conditions. This is depicted in Figure 2.1. In this case, Fednet is used to increase the road safety.

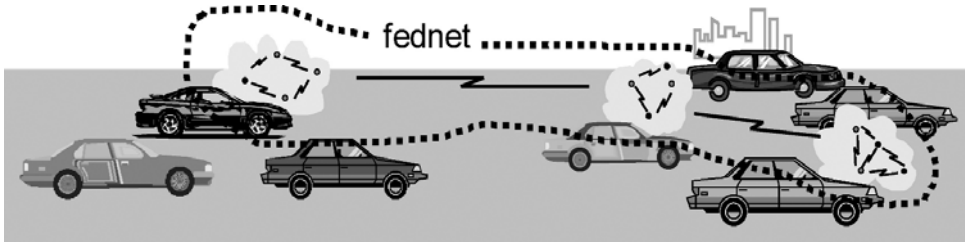

Figure 2- 1 Fednet for Vehicle Network [2]

Another example is shown in Figure 2.2, in which three people are sitting together in the restaurant to form a Fednet and then sharing the documents from one people's home server through internet. Note that the federation of PNs does not require the user are present at the same location. They could be anywhere as long as their PNs can be communicated by using the interconnecting structure (e.g. Internet, UMTS, WLAN and ad-hoc etc).

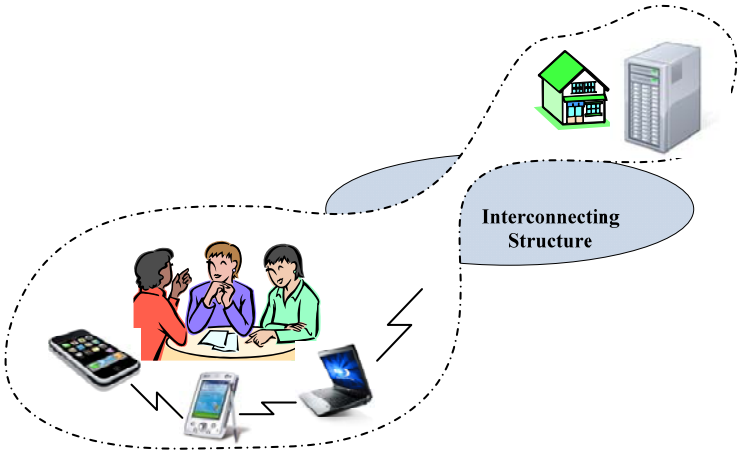

Figure 2- 2 Fednet for Sharing Personal Resources

2.1.2. What kinds of services

The main purpose of Fednet is federating different PN owners for sharing the services in their PNs for a common purpose. The types of services can be like sharing resources, such as photos, videos, personal files, project schedules, storage, internet access, e-mails, software and agenda. And the services can also be like photo editing, music editing, printing and playing games. In the Fednet sharing information by sensors (e.g. vehicle network, disaster relief situation), the information could be location, temperature and images. Fednet only contains the resources, applications and services needed to achieve the common goal.

2.1.3. Who are the users

As mentioned above, Fednet is formed by the PN owners with common interest. The common interest groups could be: family members, colleagues, friends, kids at school, public servants, car drivers (in vehicle network) and emergency teams (e.g. firemen, police, environment specialist and medical patient etc). People in Fednet can be both the providers and consumers, because they share resources in P2P manner.

2.2. Context-aware

Fednet is a context-aware network, in which context can trigger a purpose-driven or opportunity-driven federations [2]. The context is seen such as location information, energy levels, application type, environmental information and available resources etc. These kinds of information can be relevant for determining whether or not and how to provide the services in Fednet.

Today, users expect their services to be intelligent and adaptable to the changing environment. Therefore, for achieving the user's high expectation, it is necessary to create a context-aware framework, which can discover, collect and process the relevant context information for service provisioning in Fednet. It is crucial to adjust the service provisioning pattern to the type of service, context and some other issues such as privacy and security required by the PN owners. The more context information available, the better that the service provisioning system can respond to the user and the situation.

2.3. Flexible and Adaptable

The above section illustrates that service provisioning should be context-aware. When the context changes occur, it might influence the way of providing services in Fednet. In section 1.3, we depicted that there are two approaches for service provisioning in Fednet, namely overlay and proxy-based. Figure 2.3 shows the overlay approach for providing services, in which Kate and Bob are neighbors, they decide to directly communicate with each other to form a Fednet for sharing photos from Bob's PDA.

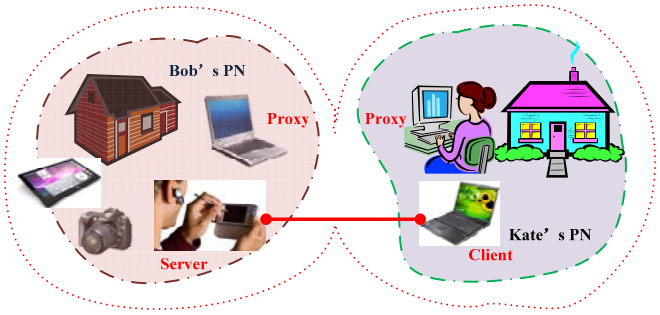

Figure 2- 3 Overlay Approach for Service Provisioning in Fednet

Now Bob is in his office, which is far away from Kate, as shown in Figure 2.4. Suppose Bob has a laptop used as the service proxy in his house and he has uploaded all the photos in the proxy. In this case, it is better for Bob to change the way of service provisioning into proxy-based. Then the services are being provided by the hybrid approach.

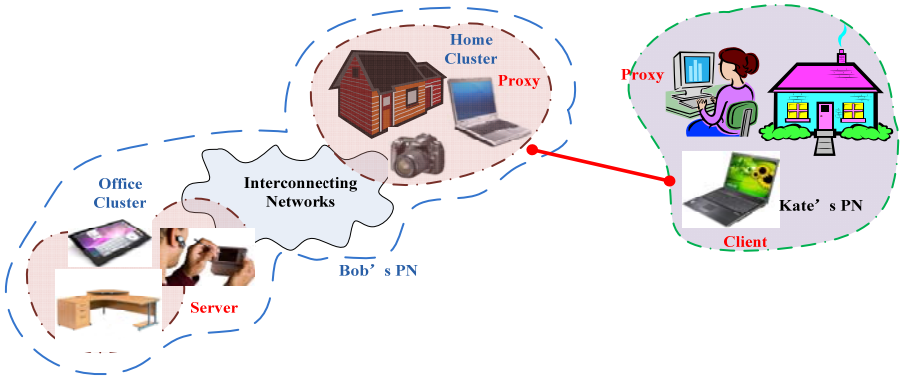

Figure 2- 4 Hybrid Approach for Service Provisioning in Fednet

The two scenarios show that sometimes using a fixed approach for providing services may not be able to maintain the quality of services in the whole process. There requires a service provisioning system, which could be flexible and adaptable to the changes of the environment.

2.4. Scalable

PN federations enable a lot of potential application scenarios and address a large user group. One personal network may participate in multiple federations and the services in the PN will

be involved in several different Fednet too. As a result, the number of federations could become huge. Solutions are needed to assure that the services in the PNs are still available and accessible under the multiple federations' situation [6]. And the management of the services also requires to be scalable.

2.5. QoS

Some real time applications in the personal network have high demands for end to end quality of service (QoS), such as voice over IP, online games and IP-TV etc. The service provisioning system of the Fednet should meet the demands of these applications with respect to QoS. The parameters, such as bit rate, delay, bandwidth, and capacity need to be taken into consideration when we design the service provisioning system. It is necessary to propose different mechanisms to fulfill the end to end demand of the different applications.

2.6. Summary

In this chapter, we analyzed the requirements of service provisioning in Fednet that need to be fulfilled to the current state of art. We first described the basic requirements of providing services in Fednet, such as the environment, what kinds of services and who are the users. Then some high-level requirements were defined based on the features of personal network and Fednet, including context-aware, flexible, adaptable , scalable and guarantee QoS. This set of requirements may be not complete. We believe that with the rapid development of the Fednet, there might be more requirements and demands from the users. However, we only focus ourselves on designing a service provisioning architecture which can fulfill the requirements listed in the above.

3

Survey on Service Provisioning

3.1. Introduction

Service provisioning is the process that servers provide services to the client or users share services and resources with each other, based on the designed systems and infrastructures. Generally speaking, the service provisioning process includes: service discovery, resource management and service delivery. Service provisioning has obtained increasing attention of the researchers due to the fast growth of telecommunication technologies and high-quality requirements of the users. A large number of mechanisms and paradigms have been reported in this popular field [5-43].

In this chapter, we list a survey on some related service provisioning technologies. These technologies have similar points with our thesis in some aspects. Examples are service provisioning in peer-to-peer network, ad-hoc network and context-aware service provisioning etc. The survey provides us an overview of what the scope and the challenges of these service provisioning mechanisms are and how they provide solutions. We hope that from the survey, we can get some design ideas and see if some of the existing modules could be useful in our work. Finally we summarize our survey in a table.

3.2. Service Provisioning in Different Networks

3.2.1. In Ad-hoc Network

Mobile ad-hoc Network (MANET) has been obtaining much attention due to the increased growth of laptops and the 802.11 wireless networking technologies. MANET can not only be data carriers for mobile devices but also can provide services to ubiquitous computing

environments. A node in MANET could potentially be the service provider as long as it has the service that other nodes are requesting. However, service provisioning in MANET is a challenge, where a node host has a certain service may become unavailable any longer due to the rapidly changing topology. Thus, most of the frameworks about service provisioning in MANET focus on designing routing protocols and scalable membership management [7-9]. In the following, we will discuss some related frameworks.

In [7], it designs a scalable and efficient geographic routing and service provision framework for MANETs. The framework consists of the following components: (1) a self-configuring, distributed and hierarchical structure, with which scalable membership and service management as well as efficient service provision can be implemented. (2) scalable membership management, being able to handle the frequent joining and leaving of the service nodes and their changing service states. (3) adaptive routing protocols to meet different routing requirements in service provision scenarios. (4) service provision scheme to realize the service provision functions, such as service discovery, delivery and coordination.

[8] gives us a framework for service provisioning in intermittently connected MANETs, relying on a flexible addressing scheme, content-based management of messages and asynchronous communications. The proposed framework is able to describe local and remote services, to characterize service providers, to define messages exchanged in the service provision process, to support a proactive/reactive service discovery and to achieve asynchronous service invocation. In [9], it presents a context-aware service provisioning model based on the concept of migratory service. Unlike a regular service that always executes on the same node, the migratory service is capable of migrating to different nodes in the network to guarantee the service continuity. The service migration is triggered by the context changes of the nodes. The model allows context-aware and adaptable service provisioning in ad-hoc network.

3.2.2. In Peer-to-Peer Network

In contrast to the traditional client-server model where only servers supply and clients consume, a peer-to-peer (P2P) network is a distributed network, in which participants are both

the suppliers and consumers of the resources. Each participant is called “peer”, making a portion of their resources directly available to other peers. There are numerous applications in P2P networks, such as file sharing, cloud computing, P2P TV, Skype and VoIP. The decentralized requirement and dynamic peer arrivals /departures makes the service provisioning in P2P network challenging. In this section, we will show some related systems/middlewares regarding peer interconnections and sharing contents in P2P network.

In [10], a peer service network middleware is proposed, which is used to initiate the peer interconnections in order to develop and to deploy P2P network services. After initiating the connection with other peers, services will be formed from a primitive execution service and bridges between service networks. The peer service network middleware enables to maximize the transparency to the operating system and user level command execution, by providing transport of standard I/O streams via the P2P network. In addition, security problems, about secure connections and secure service provision are emphasized by applying secure socket layer and simple public key infrastructure.

In [11], it presents the design and the implementation of a platform based on JXTA [12], which allows users to share contents in a P2P network while discovering others. As we can see from Figure.2.1, the platform is the core part of the basic node infrastructure, composed by four modules: (1) Repository access, a data management system manipulates local contents stored at the node. (2) P2P access, includes high-level methods in order to get access to P2P network, share contents with other nodes and search contents on the peer group. (3) Requester and Provider handle all communication aspects within the P2P network and retrieve content descriptors and related contents. The middleware platform also provides independent context-used APIs to build advanced P2P applications.

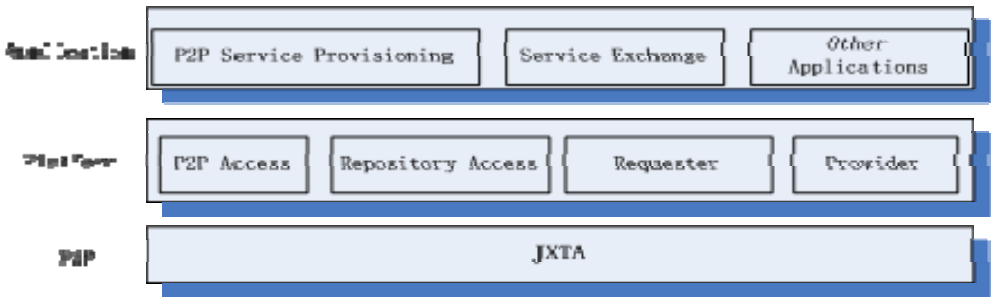

Figure 3- 1 Node Architecture [11]

[13] describes an integrated P2P service composition framework called SpiderNet. SpiderNet performs a novel bounded composition probing protocol to provide fully decentralized QoS-aware and resource-efficient service composition. [14] and [15] both address service provisioning problems in a P2P streaming environment. [16] and [17] propose billing and charging service provisioning schemes in federated PNs from the business point of view.

3.2.3. In Grid Network

Grid network combines the distributed and heterogeneous resources from multiple administrative domains to solve complex problems. Grids are often constructed with the aid of general-purpose grid software libraries known as middleware. The ultimate goal of grid computing is to share resources among grid participants and to provide mechanisms for users to use grid resources. Providing efficient, scalable and robust grid services is one of the challenges of the next generation grid systems. [18] gives an overview of architectures enabling grid based application service provisioning. [19] shows a business model of service provision through the grid network. Apart from those, [20-23] all concerns service provisioning issues in the grid network. In this section, we will discuss these approaches in details.

[22] and [23] address the problem of on-demand service provisioning in gird network. [22] proposes an architecture, which is constituted of Universal Factory Service (UFS) in order to provide a dynamic grid service deployment mechanism and a resource broker called door service to relay service requests to proper resources, and to deploy actual services to the resources. [23] illustrates an approach based on MAPE (monitor, analyze, plan and execute

functions) loop to support automatic management of grid service migration. Furthermore, it enables service provisioning on demand and achieves desired QoS. [21] depicts a framework called Harmony II, a platform for service execution on a grid infrastructure using standardized service management practices. In the context of the grid resource sharing model, Harmony II can be considered as an extension of the grid framework optimized for deploying and providing customized services in a high level.

[20] describes a flexible grid service provisioning framework built on two layers: a low-level network communication layer and a higher-level service-provisioning layer, as show in Figure 2.2. (1) The network communication layer, composed by an overlay management logic and a communication service provider, is responsible for low level connectivity between grid nodes, as well as basic information delivery mechanisms. It also maintains an optimized topology rules to create additional links and to rearrange existing ones, aiming at bounding the distance between each pair of nodes. (2) The service provisioning layer provides an interface between grid applications and the resources shared in the grid. Interaction between the service provisioning layer and the overlay management is asynchronous, and based upon a custom message passing protocol and callback notifications. This framework provides an adaptive, reliable, robust and scalable communication platform for service provisioning in grid network.

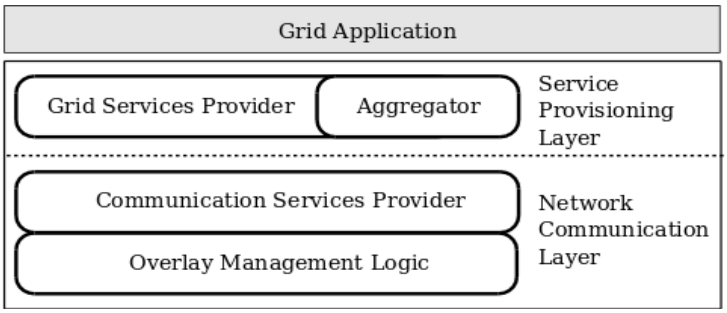

Figure 3- 2 Framework Architecture [20]

3.2.4. Conclusion

Service provisioning in ad-hoc, peer to peer and grid network all face the problem of the network‘s dynamic nature. However, the levels of solutions for these three networks are

different. In ad-hoc network, most of the service provisioning framework focus on improving the network layer issues, such as routing protocols and addressing schemes. For P2P network, service provisioning platforms are designed between network and application layer, namely middleware. Grid network solves service provisioning not only from the network layer, but also concerns about the upper layer.

The dynamic nature of the network is also a challenge for service provisioning in Fednet. As discussed in chapter 1.4.2, we want to propose an architecture working on the application layer. Thus, we will not address the routing issues. We think that the component of scalable membership management and the concept of migratory service in [9] would be useful for our design.

3.4. Service Provisioning for Mobile System

The evolution of 3G/4G mobile systems introduces a new era in advanced multimedia service provisioning to mobile users. To provide flexible, adaptive, reconfigurable, context-aware and dynamic services are the goals of the future mobile systems and networks. As the service provider, telecoms are concentrating on the development of innovative approaches to fulfill the requirements of the mobile clients, so they can still increase profit and remain competitive on the telecommunication market. In the following, some existing frameworks and architectures for service provisioning in mobile network will be discussed.

[24] and [25] all illustrate a generic framework for reconfigurable service provisioning model (RCSPM) and value-added service to mobile users, which supports for adaptability, flexibility and reconfigurability. The brilliant part of RCSPM is the reconfiguration control/service provision manager, which consists of the following aspects: service discovery, service deployment, service data manager, user access session, user profiling and reconfiguration manager. This part interacts with technology independent interfaces (standard open network APIs) and underlying 3G core network and enables new approaches to service provisioning, customization, and personalization.

In [26] and [27], the service provisioning in Next Generation Networks (NGN) is discussed. As the increasing development of NGN, there is a need for a next-generation service delivery platform, from which seamless, adaptive, dynamic provisioning of multimedia information and communication service to mobile users can be realized. [26] shows a self-adaptive service provisioning middleware framework, which can seamlessly provide services to mobile users anywhere, anytime and in any context, by interoperating with existing service delivery platforms. This allows it to offer a unified, ambient-aware, adaptive and personalized service provisioning environment. In [27], the authors propose a solution for semantic-aware service provisioning in agent-support NGN. It enables network operators to enhance the provisioning process with features of reactivity, semantic-awareness and consumer context-awareness.

Some mobile network models/platforms are also focusing on investigating a multi-functional for dynamic, reconfigurable, context-aware and flexible service provisioning to satisfy the various requirements of the mobile users. Examples are service provisioning based on mobile agent in [28], as shown in Figure 3.3, dynamic context aware service provisioning in [29], dynamic configuration of semantic-based service provisioning in [30] and adaptive service provisioning platform in [31].

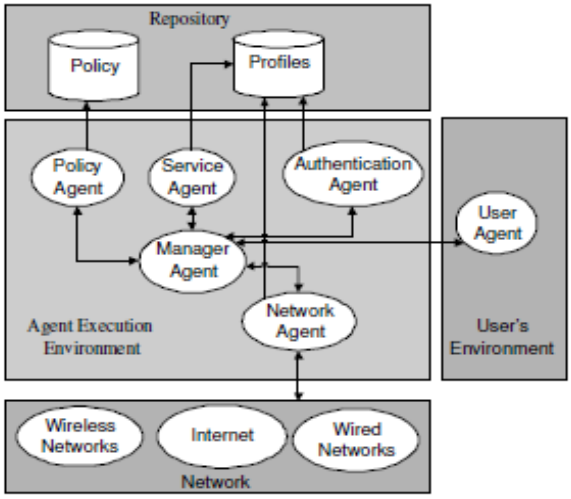

Figure 3- 3 Mobile Service Provisioning Framework [28]

3.4.1. Conclusion

Fednet could be formed by mobile users for achieving a common interest, such as sharing photos, videos and internet access etc. In other word, providing services in mobile system could be one of the application scenarios in Fednet. Telecoms are concentrating on the development of making the service provisioning in mobile system flexible, adaptive, context-aware and reconfigurable, which is also the target of service provisioning in Fednet. This section gives us much insight into how the designed modules interact with each other, and how they interact with the underlying core network, as discussed in [24, 25] and in Figure 3.3. As a consequence, the technologies of service provisioning in mobile system will be an important reference for us when design the service provisioning architecture in Fednet.

3.5. Service Provisioning with Special Requirements

3.5.1. User-centric

The main idea behind the user-centric service provisioning models is the service provider concerns about the quality of service (QoS) or service performance or preference with the user's perception [32-35]. The user should express his intention when requesting a service with specific quality parameters to the provider. But this will arise some issues, such as how to get necessary service-related information, how to configure and control resources (e.g. network devices, end systems, service level agreements, etc) [33]. In the following, we will discuss some models regarding user-centric service provisioning.

In [32], a user-centric service provision model (UCSPM) is illustrated, based on unified context model, a user-oriented QoS model and an adaptive decision model. This model is aiming at facilitating the development of ubiquitous computing applications and supporting for user-oriented quality of service evaluation and dynamic service delivery. Besides, it also provides effective support in collecting and processing contexts, the interaction between mobile clients and service manager. Figure 3.4 shows the architecture of UCSPM. Mobile client is a service consumer, holding service lookup request and also acting as a context collector that gathers basic context information, such as its location, battery life and network bandwidth. The context interpreter in the service manager is an important component that

translates low-level context into high-level context. The value-added service provider (VASP) can add, delete and update service components. The proposed QoS model can be described as QoS (q, f), where q is the set of quality criteria and f is the set of QoS operators. The factors in quality criteria are device capability, user preference, user status and network parameter. A decision model is used to evaluate and select the services with high QoS values.

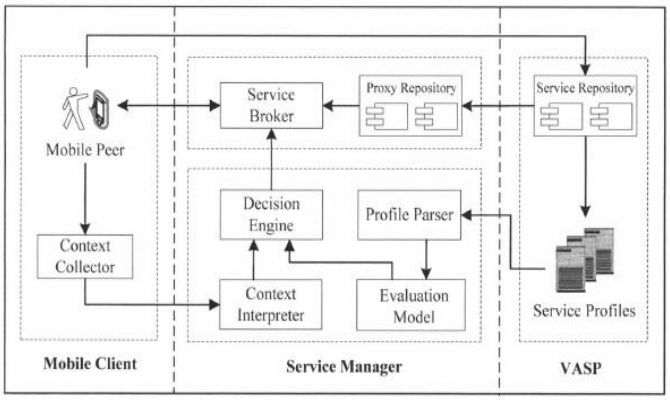

Figure 3- 4 User-centric Service Provision Model [32]

3.5.2. Context-aware

In mobile and ubiquitous computing, it is a challenge for the system to exploit the dynamic context environment of the user, such as the users' location, the network status and the terminal devices etc. Context-aware service provisioning considers user's current context, including personal preferences and environment's capabilities. Based on this information, the service first adapts its behavior to fulfill the user's needs and then automatically executes the services and the commands. The contextual information can be integrated into various related profiles, such as user preferences profile, network profile and service profile etc. The combination of all these profiles constitutes the user profile [29]. It has been widely concerned about how to continuously sense the dynamic context and to transmit the changes to the service providers for real-time service re-adaptation. Some solutions will be discussed in the following.

Service provisioning in ubiquitous environments requires full visible and flexible management of the context information. Ubiquitous context-based security middleware

(UbiCOSM) proposes and implements a novel security model for context-centric access control in ubiquitous service provisioning scenarios [36, 37], as depicts in Figure 3.5. [38] shows a hybrid service provisioning model (HSPM), which aims to offer the mobile users a way to: (1) constantly receive information regarding services of interest in the present situation. (2) share information about services in a small/medium–sized community of users. Service provider offers the users available services in a context-aware manner. User can attach its observations and personal opinions to the service descriptions, and share it with others within a certain community.

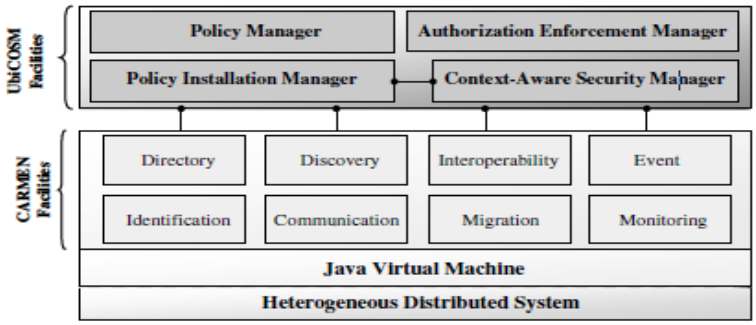

Figure 3- 5 UbiCOSM Middleware Facilities [36]

[29] illustrates a framework that enables efficient management of contextual profiles, with the focus on location and context awareness in mobile service provisioning. In the proposed network, location sensor is identified as location manager, which is responsible for retrieving and managing the information related to the location and mobility of the user. Personalization and customization during service provision is achieved by distinguishing the user preferences according to the location of the user and by maintaining different user preferences profiles. The location-sensitive user profiles are managed by the user profile manager. Then a flexible and innovative model for user profiling is introduced, in which enhanced adaptability and personalization can be realized.

3.5.3. Federated Service Provisioning

In recent years, inspired by the emerging web services standard and peer to peer technology, a new federated service providing (FSP) system paradigm has aroused increasing research

interests. Peers in the FSP system comprise multiple service components distributed across numerous network nodes in order to offer domain specific services. The FSP system has been widely investigated in many distributed applications, such as steaming processing [39], web service [40, 41] and universal description discovery and integration [42].

Web services in the context of a federated environment have been discussed in [40] and [41]. [40] focuses on providing web services security from business federations. Applications must be able to determine the levels of authentication and authorization before accessing web service. Federations are responsible for creating an infrastructure, collecting the necessary security attributes from the requestors and then letting them access the data of one or more service providers. [41] develops a framework for service level management in web service federated context.

[39] first shows us a federated service provisioning system architecture, in which peers form service groups depending on the types of services they provide, as depicts in Figure 3.6. Then it presents a coordination mechanism, including a labor-market model, a recruiting protocol and a policy-driven decision architecture, as a solution for how to self-organize the service group structures in response to the varying service demands. The labor-market model establishes an analogy between FSP systems and a simplified social recruiting structure. On top of the model, recruiting protocol is used to regulate the peers' interaction through a recruiting process. Peers can make their service provisioning decision according to the policy-based decision criterion.

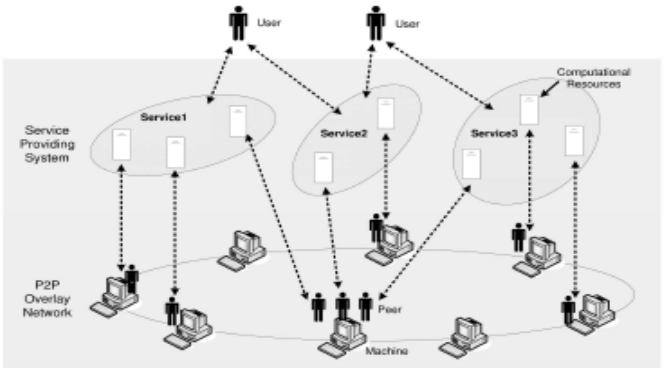

Figure 3- 6 Federated Service Provisioning System Architecture [39]

3.5.4. Conclusion

The technologies about user-centric, context-aware and federated service provisioning cover the requirements of service provisioning in Fednet. User-centric service provisioning mainly addresses the issue of quality of service. The modules in the user-centric service provision model in [32] give us some design ideas. The components such as context collector, context interpreter, evaluation model and decision engine, show us how to collect the context data, the interactions between the client and the service manager and the service selection with high QoS values. For context-aware service provisioning, the detected context information could be used to analyze the quality of the service. The directory and discovery modules in Figure 3.5 reminds us that discovering and managing services are very important issues in the service provisioning process. The federated service provisioning system discussed in this section also aims to offer domain specific services. But it focuses on how to collect the federated peers.

3.6. Summary

In this chapter, we list a service provisioning survey. Admittedly, the scopes of service provisioning mechanisms are much broader than what we discussed here. Other technologies, like service provisioning in ATM and service provisioning in optical network are also well developed. But since that is not possible for us to obtain all the service provisioning methods, we just emphasize our survey work on ad-hoc, gird, P2P, mobile service, user-centric and context-aware and federated service provisioning, as shown in Figure 3.7. We believe that these technologies have much common with the service provisioning in Fednet.

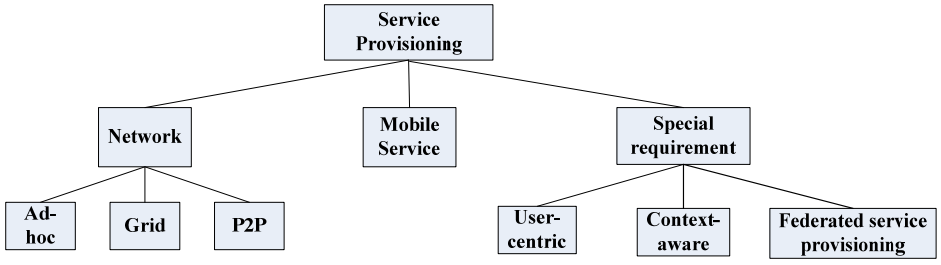

Figure 3- 7 Taxonomy of Our Service Provisioning Survey

Some of the models cover the requirements that we listed in chapter 2 for the service

provisioning in Fednet, but no one covers all. Thus, we summarize our survey on the existing service provisioning technologies and paradigms and list the requirements that the models might fulfill, as shown in Table 3.1.

Table 3- 1 Summary of the Service Provisioning Mechanisms

Aspect of View	Type of Identities	Scope	Challenges	Solutions	Requirements Fulfilled[1)]
Network	**Ad-Hoc**	Providing specific services to ubiquitous computing environment	Node hosts a certain service may become unavailable due to the rapid changing operating contexts	Flexible addressing scheme [8], scalable membership management and adaptive routing protocols [7].	Flexible, adaptable and scalable.
	P2P	Each peer shares its resources/services with other peers and discovers other available services on the network	Decentralized requirement and dynamic peer arrivals/departures	Peer service network middleware [10], platform based on JXTA [12], Spidernet [13], admission control mechanism [14].	QoS
	Grid	Provide services in large scale computing environment, matching demands for services with resources	Dynamic nature, resources and demands are fluctuating	Dynamic grid service deployment mechanism [22], optimized topology rules [20], automatic management of grid service migration [23].	QoS and adaptable
Application	**Mobile system**	Flexible, adaptive, and context-aware service provisioning	Seamless service provisioning for mobile users anywhere, anytime and in any context	Middleware, service delivery platform [26, 27], service provision manager [24].	Flexible, adaptable and context-aware
Special Requirement	**User-centric**	Service provider concerns about QoS and preference with the user's perception	How to configure and control resource(e.g. network devices, end systems, SLA etc)	Cognitive resource management for QoS [35], user centric service model [32], automatic service provisioning system [33].	QoS

	Context-aware	Taking user's current context into account, including personal preferences and environment's capabilities for real time re-adaption	It is a challenge for the system to exploit user's dynamic context environment , requiring full visible and flexible management of context information	Ubiquitous context-based security middleware [36, 37], hybrid service provisioning model [38], location and user profile manager [29].	Context-aware
	Federated service provisioning	Comprising multiple service components distributed across numerous network nodes in order to offer domain specific services	How to self-organize the service group structures in response to the varying service demands	Federated service provisioning system architecture [39].	Adaptable

1) The requirements listed in chapter 2 that the technologies can fulfill.

4

The Service Provisioning Architecture in Fednet

4.1. Introduction

In the previous chapters, we have introduced the basic concept of Fednet and the two existing service provisioning approaches in the Fednet, namely overlay and proxy-based. Each approach has the advantage and disadvantage. For satisfying the user's preference and always providing the services in a high quality, we aim to design a new service provisioning architecture that can trade off between the two approaches. We hope that our architecture can fulfill the requirements listed in chapter 2, such as context-awareness, adaptability and flexibility. Learning the related work in chapter 3 shows that there is no integrated solution to meet all the requirements. However, there are some modules that could be useful for our design, such as the modules of context collector and context interpreter and policy engine in [32]. Our architecture is proposed based on some of the existing modules.

In this chapter, we first give an overview of the new service provisioning architecture and then describe each module. The modules in Fednet agent (FA) are the most important part of our design, especially the policy engine. The FA is responsible to collect the context data and to choose the service provisioning approach based on the context information. For the modules in the Fednet manager (FM) and the service proxy, we briefly introduce them but not discuss in details, because we focus our design on the Fednet agent, which is directly involved in the service provisioning process.

4.2. The Proposed Architecture

Figure 4.1 shows the architecture we propose for service provisioning in Fednet. The architecture consists of the following parts: (1) service discovery and service management in

Fednet manager, being able to discover the resources and services needed in the Fednet and to manage the joining and leaving of the services. (2) service parser, context collector, context interpreter and policy engine in Fednet agent, aiming to collect the context information of the selected service and to decide which service provisioning approach (overlay or proxy-based) is used. (3) service handler in service proxy, taking charge of handling the service approach switching process, such as triggering the service proxy, holding data in the buffer to avoid severe switching delay and interruption.

The Fednet is formed of two PNs in Figure 4.1. The FM is the main component for federating the participating PNs. In PN1, there is a Fednet agent and a service proxy consisting of our designed modules. In PN2, there are FA and service proxy, having the same modules with those in PN1. For easy discussion, we just draw a full structure of PN1 and use a simple figure to represent PN2.

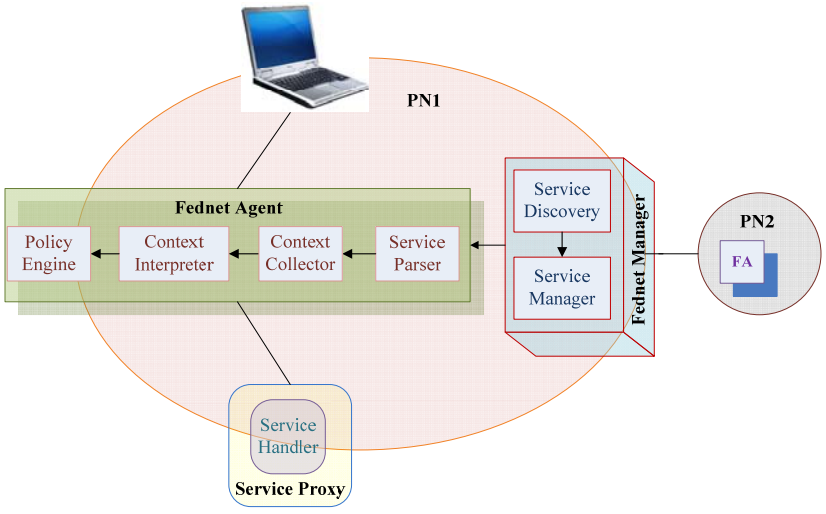

Figure 4- 1 New Service Provisioning Architecture for Fednet

4.3. Modules in Fednet Manager

The Fednet manager is responsible for creating Fednet advertisement, inviting PNs who wants to join in and building service directory. To easily manage the services of the participated PNs, there are service discovery and service manager functionalities in FM.

4.3.1. Service Discovery

Fednet is formed by the PN owners who want to share resources for a specific objective. Service discovery functionality is in charge of discovering the resources, applications and services needed to achieve the common goal. Here we distinguish the two possible types of service discovery in Fednet: proactive and reactive. (1) Proactive discovery takes place when there is a need for achieving a certain goal. The Fednet manager will discover where the needed services are and invite the relevant PN members to join in. (2) Reactive service discovery takes place when some services are needed by PN owners. If these services are not contained in the Fednet, FM will search for them and then involve them in the Fednet. All the participating services will be stored in the FM to build a service directory, which will be used for the service lookup later on.

4.3.2. Service Manager

The service manager is designed to manage the services listed in the service directory in the FM. Since Fednet may be a purpose-driven or opportunity driven ad hoc network, people may join or leave the Fednet frequently. A service might become unreachable due to the leave of the PN owner. The service manager is used to handle the joining and leaving of the services and timely update the service directory. In addition, Fednet enables large number of PNs to participate, and the services offered by the PNs could be geographically distributed. Therefore, the membership managements should be geographically scalable.

4.4. Modules in Fednet Agent

In each participated PN in Fednet, there is a Fednet agent. It generates service lookup request to the FM and produces access control decisions for PN services. FA is the most important part of our design because it is directly involved in the service provisioning process. Our proposed modules in FA mainly solves the following problems: (1) service selection (2) how to collect context data (3) how to choose service provisioning approach according to these context information. In the following, we will discuss them in details.

4.4.1. Service Parser

When a client asks for a service (e.g. printing) from the Fednet, the Fednet agent of client's PN first initiates a service lookup request to the Fednet manager to check where the service is. After receiving the lookup request, the FM will search in the service directory for the service information and then give a lookup response, as shown in Figure 4.2.

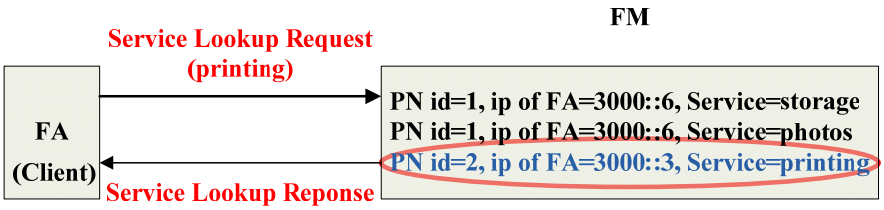

Figure 4- 2 One server is available

Sometimes there might be more than one serving PN that can provide the service. Service parser takes charge of assigning a service to the client by comparing and analyzing the candidate servers' states, such as the battery lifetime, location and whether it is busy etc (we assume these information are available). As shown in Figure 4.3, PN2 and PN3 both could provide the printing service. We assume that the printer in PN2 is free right now and near to the client, while the printer in PN3 is having a lot of printing tasks. After comparison, service parser chooses PN2 to provide the service. The FM might return an empty list if there is no such service in the Fednet. Actually, service parser is dealing with the service selection, which is also an important issue in Fednet. Here, we just give a brief idea and we would like to investigate more parameters to make the selection decision in the future work.

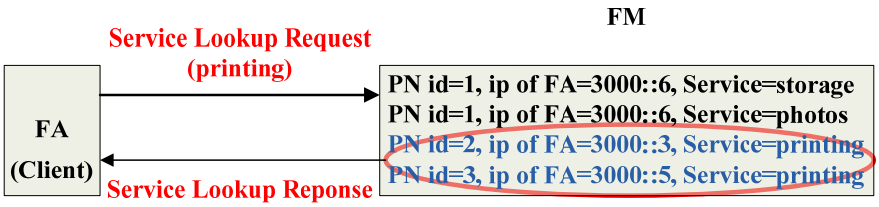

Figure 4- 3 Two servers are available

4.4.2. Context Collector

Since Fednet is a context-aware network, it requires the service provisioning in such a network to enable to sense the environment and adapt to the changing context. Context collector is proposed to gather the context data. In our design, when talking about the context information, we mean the following parameters: device capability, service type, battery lifetime, link quality, security, privacy, location, reputation and network dynamics. Some of the policies are proposed based on the current research [1]. Some of the them are came up from the related projects [43][52]. We classify the Fednet services for the service provisioning purposes into overlay services and proxy-required services [1]. All the criteria mentioned above play a role in classifying the services. In this section, we will explain each criteria and describe how to collect them.

a) Device capability [1]

Some devices in PN have strong capabilities and can provide services directly, such as laptop, desktop and PDA etc. We classified such devices as ‘strong’ device and their services could be provided by overlay approach. Some devices are not able to provide their services independently, because they do not have networking capabilities (e.g. loudspeaker, webcam, local printer, sensor node, earphone etc). These devices need to be attached to other devices, such as a laptop or a computer, which act as the proxies for them to export their services. These devices are considered as ‘weak’ devices and service proxy is required for their services. Device capability is given by the PN owner when his device/service is added in the Fednet.

b) Battery lifetime

The battery lifetime of the device is taken account into consideration, especially for the portable device (e.g. PDA, smart phone and camera etc). Because portable devices are easily out of power compared to the regular devices, such as desktop and printer. When the power of a device is lower than some degree (like 30% or 20%), we would suggest to use the proxy-based approach (assume the proxy has more power) for providing the services offered by this device. Thus, it could prevent that the service is suddenly interrupted due to the power

missing. If the proxy does not have too much power, probably going through the device itself is a better way.

Considering the real scenario, it does not mean that the service is inaccessible when current battery life is lower than 30%, but the service might be less satisfactory. We assume that there is an API, which allows the context collector to be aware of the battery status of the device.

c) Location

Here we distinguish the two types of location information: (1) the distance between the client and the server (2) the distance between the client/server and the service proxy. We suppose that there could be a device in the FA can measure or sense the distance. If the client and server are far away from each other, but the service proxy of the PNs are quite near, we suggest the proxy-based approach. In some cases, the service proxies in PNs may be far away from the client/server, which may highly degrade the overall service provisioning performance. Under this situation, overlay approach is the better choice.

d) Link quality [43]

The link quality of the wireless channel between the client-PN and the server-PN affects the quality of service, since higher layer application's performance is very sensitive to link dynamics and the changing topologies. Sometimes overlay can give a higher link quality than using a proxy. While sometimes going through proxies shows better performance than the direct communication between the client and the server. Therefore, the link quality will have impact on final decision making. A novel link quality assessment (LQA) method is proposed in [43], which can lead to a fast and smart routing decision. In our design, we adopt the LQA method to measure the link quality through the overlay link and the proxy-based link. Using this method, the link with higher quality (i.e., throughput) will be chosen for the service provisioning.

The link quality parameter is only suitable for the short-distance scenario (less than 50m according to the experiment). If the devices are quite far away from each other, it is difficult for us to measure the wireless link based on the LQA.

e) Privacy

Privacy sensitivity is important when choosing the service provisioning method. The proxy-based approach fits for the scenarios which a PN owner wants to export its service without revealing other existing devices and services. If the PN owner or some applications have the low privacy sensitivity, overlay approach can be applied. This kind of information is given by the PN owner when he enters the Fednet.

f) Security

Sometimes the PN owners in the Fednet want the services to be encrypted when sharing them, which requires a security communication channel. We suggest that the service could choose the approach which is able to provide the secure channel. For instance, if the service proxies in the PNs enables encrypted channel, then proxy-based approach is suitable. Similarly, if the client and the server establish encryption, overlay approach could be used. PN owner gives the security preference when the service is added in the Fednet.

g) Service Type

We classify Fednet services into common and specific services based on the access control method. A common service is accessible by all Fednet members upon presenting their membership credential. A common service can be for example, a forwarding service, a display service, a printing service, internet access, storage facilities, etc. Since the access to common services of a Fednet is granted to all members without additional access control and provided at specific entry points, so called proxies. Therefore, the common services are classified by default as ‘proxy-based’ services. The remaining services in the Fednet are termed as specific services, which require the second-level access control procedure, i.e. evaluating service access policies by the Fednet agents in each PN. An example of a specific service can be a file sharing service and webcam service. For the purposes of service provisioning, the specific services can be ‘proxy-based’ as well as ‘overlay’ services, depending on other criteria, such as reputation and device capability [1].

Apart from the access control method, service type also has some relationship with the user's preference. The preference we mentioned here could be like the following situations: (1) It is a new device, the user does not want it to be involved in so many applications. (2) To much access to his devices could lead to the power missing, which will influence the using time of the devices. The user tries to avoid that. If the services are defined as common services, then proxy-based approach will be used.

h) Reputation

We define reputation for a PN, for a PN's service and for a Fednet in the following ways: (1) Reputation of a PN is based on its contributions to this Fednet, its service providing and consuming behavior. Reputation of a PN will be used for service provisioning decision making. (2) Reputation of a PN's service is based on the service's quality, content, availability, performance and price etc. This kind of information will be used in the server selection in the service parser. (3) Reputation of a Fednet is the reputation that a PN concludes about the Fednet, based on the quality of cooperation experienced while being a member of this Fednet. It is a reputation that the Fednet has obtained from its members, which will be used for choosing a Fednet to join in.

Here we only use the reputation of a PN to be a criteria in the policy engine. We consider that the PNs have a positive reputation if they are known or familiar to each other through positive and successful interactions. Then the service can be provided in both ways, depending on other criteria. Otherwise, the service is provided via the proxy.

i) Network dynamics [52]

At the beginning of my master project, I was doing an internship in TNO for a few months, with the topic: *Characterizing the Dynamics of Dutch Home Network* [52]. During the internship, I found that the concept of the network dynamics could be useful for my master thesis.

Network dynamics is a characteristic of the network, which describes the uncertainty of the network system. The higher the dynamics, the more uncertainty of the network is. In other

words, high network dynamics means there are many activities occupied in the network. One of the consequences of the highly dynamic network is there might not be enough bandwidth left, which will influence the performance of the coming network applications. Thus, if the network is estimated in a high dynamic state or it is predicted that the home network will be more dynamic, the network applications (e.g. streaming video, VoIP and video chatting etc) will be rejected right now and could be accessed after waiting some time.

Since each PN cluster can be seen as a network, the criteria of network dynamics could also be involved in the policy engine to make decision. It might work in this way: Based on the current state of the cluster and the complicated transformation matrix [52], we can predict the next state of this cluster, such as stable, unstable, very dynamic etc. If the cluster is predicted into being a dynamic or unstable state, we suggest not providing or consuming any services from this cluster, for the consideration of the cluster's stability.

However, to get the dynamics information of the system is difficult. One method is proposed in [52]. Since it is totally out of the scope of this thesis, we will not discuss it in details. People who are interested in it, please refer to my internship report [52].

4.4.3. Context Interpreter

Context interpreter is responsible for translating the context data into numerical values. Table 4.1 shows the value of each parameter, which will be used in the policy engine for a decision making of choosing the service provisioning approach.

Table 4- 1 Values of the Parameters

Parameters	Binary Value	
Device capability	Weak $\in$ **0**	Strong $\in$ 1
Service type	Common $\in$ 0	Specific $\in$ 1
Reputation	Low $\in$ 0	High $\in$ 1
Privacy	High $\in$ 0	Low $\in$ 1
Approach	Proxy-based	Overlay

Location*	0	1
Battery lifetime*		
Link quality*		
Security*		
Network dynamics*		
Approach	It depends on the situation	

For simplification, we use only low and high values to represent the parameters. For instance, if a device is defined as a weak device by the user when this device enters the PN, the context interpreter translates it into 0. If it is a strong device, 1 is given. For other parameters, they are interpreted in the same way. We propose this module with the following reasons:

(1) The management of these parameters becomes easy. As we can see from Table 4.1, the parameters given by 0 are classified into using proxy-based approach. Otherwise, overlay approach is proposed. As to the parameters with star, namely battery lifetime, link quality, security, location and network dynamics, the service provisioning approach decided by these parameters depends on the actual situation. If proxy-based is chosen, we use 0 to represent it. On the other hand, when overlay is suitable for the case, 1 is assigned.

(2) Numerical values are convenient for being used in the policy engine. We will propose some decision making algorithms in the policy engine, which are responsible for choosing the way of service provisioning in Fednet. These algorithms are designed in a hierarchy way. It means that the policy engine will evaluate the policies in each layer and then get a decision of that layer. After collecting the context data, there needs a way to translate these parameters into the simple language rather than typing the English letters in each layer, such as “Low”, “Specific” and ”Weak”. In this way, it is easy to see which approach will be used. For instance, if the evaluation result of the parameters is 0, we know that proxy-based will be used.

Admittedly, not all the parameters could be completely classified into strong/weak or high/low. There is something in the middle. As the world is not only just black or white, there

is still gray zone. For the parameters in the gray zone, when user defines them, it is also better to refer to the real situation. For example, if a device has 29% battery left, it does not mean we must use a proxy for this device. Overlay can of course be used as long as the device is still working to provide the service.

4.4.4. Policy Engine

Policy engine is the core part of our design, in charge of making a final decision for choosing the service provisioning in Fednet, based on the context information. We focus the design on the concept of policy, in particular on the following questions:

(1) How to give a weight value to the policies? Since there are so many parameters involved in this process, it is useful and necessary to distinguish among the parameters according to level of their importance.
(2) How to provide service based on the policies? The evaluation of each parameter in the policy engine leads to the classification of the service into an overlay or a proxy-based. Actually, the policy engine in our design can be seen as a multi- criteria decision making process.

As to how the policy engine works, we will discuss it in chapter 5.

4.5. Module in Service Proxy

Our new mechanism for service provisioning in Fednet aims to be able to context-aware and trade off between the overlay and proxy-based approach. The service handler deals with the approach switching process with the following aspects. (1) When the service provisioning approach is switched from overlay to proxy-based, the service handler takes charge of triggering the service proxy and setting up the communication channel between the client/server and the proxy. (2) When changing the approach from proxy-based to overlay, service handler is responsible for holding the services for a certain time, while waiting for the channel’s setup between the client and the server. We assume that there are buffers in the service handler, which would be used to store an advance supply of data to compensate for delays or interruptions caused by the approach switching.

4.6. Pros and Cons of Our Architecture

In chapter 1.3, we discussed a policy-based service provisioning architecture for Fednet in [1] as shown in Figure 1.8, which aims to make service provisioning context-aware. However, this architecture just shows us a brief idea that we could use some policies for choosing the overlay or proxy-based approach. It does not describe how the proposed modules interact with each other and what kinds of roles they are playing when providing the services.

Compared to [1], we make the service provisioning architecture easier to understand and clearly show how each part is connected with others. Besides that, we add some new modules in the Fednet agent, such as service parser, context collector, context interpreter and policy engine. We propose more criteria in the policy engine for the decision making. They are location, battery lifetime, link quality, network dynamics and security. We hope that with these criteria we can make the decision making more suitable for the system and the applications. However, our new architecture might take time to collect these parameters and analyze them. And the added modules make the system complicated, which may decrease the system's efficiency.

4.7. Summary

In this chapter, we introduced the new service provisioning architecture for Fednet. The architecture shows us the process of the service provisioning, including service discovery, service management and service delivery, and how the proposed modules interact with each other. We also propose a service parser to be responsible for the service selection and a service handler in the service proxy to handle the interruption and delay caused by the approaches switching.

A context collector is proposed in the Fednet agent to collect the context information. In section 4.4.2, we illustrated the context data that we want to gather and described the meaning of each parameter in detail. All the parameters are involved in classifying the Fednet service and playing a role of choosing the service provisioning approach. Compared to the policies depicted in [1], we have more policies and make them more logical. Admittedly, there could

be more context data than what we listed here. However, we only confine our discussion on the 9 kinds of criteria, which are directly related to the service provisioning.

Fednet agent is the core part of our design, especially the policy engine. How to provide service based on the policies is the most important issue in our design. For details, we will discuss them in chapter 5.

5

Context-aware, Flexible and Adaptable Service Provisioning

5.1. Introduction

The last chapter has introduced our new architecture for the service provisioning in Fednet. As we mentioned above, the modules in the Fednet agent are the most important part in our design, especially the policy engine, which is in charge of choosing the approach (overlay or proxy-based) for the service provisioning in Fednet. We have 9 kinds of criteria designed for the policy engine. There are device capability, service type, battery lifetime, link quality, security, privacy, location, reputation and network dynamics. The policy engine actually solves the multi-criteria decision making problem.

Fednet enables many applications, such as file sharing, photo editing and playing games etc. And there are a lot of scenarios, in which Fednet, as discussed in chapter 2.1. Examples are friends sitting in the restaurant share documents, people in the office want to connect the computer in his apartment, cars in the highway share the real-time data about the hazardous road conditions etc. We believe that for the various scenarios and applications, the way of choosing the service provisioning approach might different from each other, because the parameters involved and their impact on the decision making might be different.

Here we concentrate ourselves on designing the decision making algorithms for the following scenarios: confidential application, short-range scenario and unfriendly environment. Besides that, we also propose an algorithm for being used in the general cases. As to the scope of policies, we skip the network dynamics, which is out of the scope of this thesis. We mainly focus on device capability, service type, privacy, location, reputation and battery lifetime. For

other parameters, they might be involved in a specific scenario.

In this chapter, we first discuss the following problems: (1) which policies will be used in the policy engine for each scenario, (2) what is the decision making algorithm for that scenario. Then we give some examples to show how the decision making algorithms make the service provisioning in Fednet flexible and adaptable to the changing environment. Finally, we summarize our algorithms and the whole service provisioning architecture.

5.2. General Scenario

The decision making algorithm in the this scenario is an algorithm for general and common Fednet cases, such as file sharing, photo editing and playing games etc. We consider the Fednet environment here as a friendly environment. For instance, home Fednet, which is formed within family members, office Fednet which is formed between the colleagues familiar to each other. The following parameters are involved in the policy engine for this scenario: device capability, service type, privacy, location, reputation and battery lifetime. The decision making algorithm is shown in Figure 5.1. 0 represents for the proxy-based approach, 1 means overlay approach, which we have discussed in the context interpreter in chapter 4.4.3.

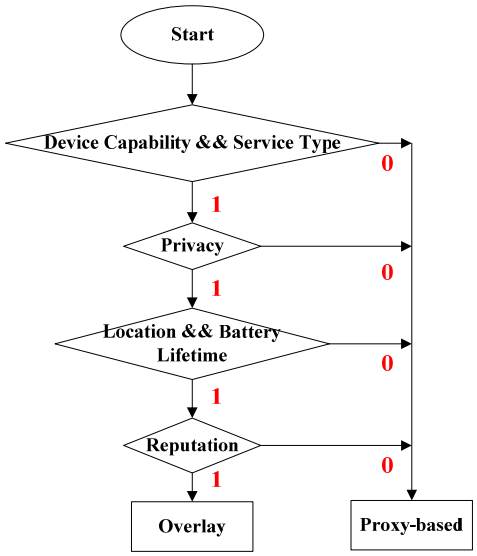

Figure 5- 1 Decision Making Algorithm for General Scenario

We give the device capability and the service type with the highest priorities in the algorithm. The device capability is classified by the networking capability. If a device is defined as a weak device, proxy-based approach is required. There is no need to consider other parameter because this device must be attached to some other computing equipment for exporting the services. The service type has the same situation as the device capability. If the PN owner defines his service as a common service, it has to be provided through the service proxy.

Fednet is a user-centric network, the user's privacy also needs to be highly concerned. The proxy-based approach is suitable for the scenarios which the PN owner does not want to reveal his PN internal structure. If the owner has a low privacy sensitivity, overlay approach can be applied. After considering the user's privacy sensitivity, we check the physical parameters, namely location and the device's battery lifetime. The reputation comes finally because we assume that the environment of the general scenario is friendly, which denotes there is always a sufficient reputation value between the client PN and the server PN. High reputation allows the overlay service provisioning. Changes in the conditions of the policies will affect the way of service provisioning.

5.3. Specific Scenario

In the general scenario, we assume the Fednet applications are going on between familiar people, namely, there is a friendly environment. However, in the real cases, there might be a need or an opportunity for the unfamiliar people to form a Fednet to achieve a common goal. Apart that, some applications require quality of service or secure transmission channel. Then what would the decision making algorithms be in the unfriendly environment or for the applications with special requirements? In this section, we will describe the specific Fednet scenarios and give a decision making algorithm especially for each case.

5.3.1. Unfriendly Environment

Unfriendly environment can be also called anonymous environment, which means a spontaneous Fednet, formed for example, in the train between unknown people. The reputation value plays an important role in this situation, since it is a parameter estimates the

unknown people's behavior and contribution, the change of which may influence the way of providing services. Thus, after the evaluation of device capability and service type, we continue with the consideration of user's privacy and its reputation value towards the server/client PN, as shown in Figure 5.2. Finally the location and battery lifetime are evaluated.

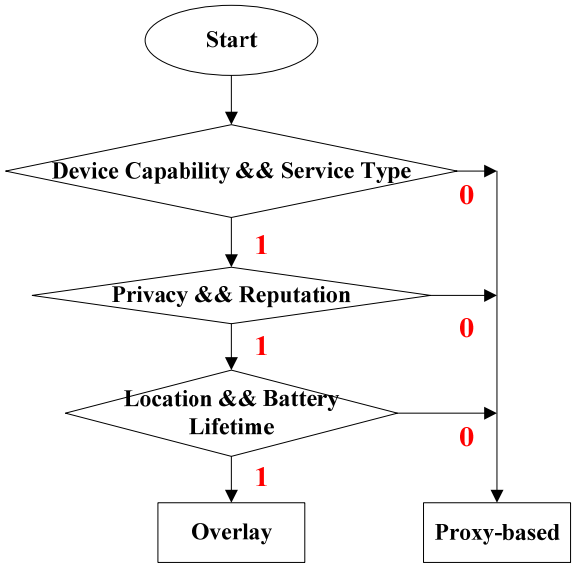

Figure 5- 2 Decision Making Algorithm for Unfriendly Environment

5.3.2. Confidential Application

When the PN owners require encryption when providing the services, we suggest that the user could choose the approach which is able to provide the secure channel. For instance, if the service proxies in the PNs enables encrypted channel, then proxy-based approach is suitable. Similarly, if the client and the server establish encryption, overlay approach could be used. We give the security parameter the same weight with the user's privacy sensitivity, as shown in Figure 5.3. Because security preference is also a part of the user's preference, which needs to be highly considered when making the decision. For the parameters, such as location, battery lifetime and reputation, there will not be taken into account. In this way, the approach which can provide privacy and security will always be the decision for the service provisioning.

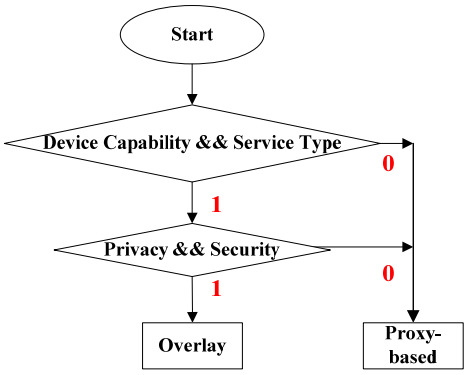

Figure 5- 3 Decision Making Algorithm for Confidential Application

5.3.3. Short-range Scenario

A novel link quality assessment (LQA) method is proposed in [43], based on which a smart and fast routing decision and high end to end throughput can be achieved. However, the measurement of the wireless link quality in this method is only available in a short distance, like less than 50 meters according to the experiment. Therefore, we take the link quality into the consideration especially for the Fednet formed by PN owners near each other. Sometimes overlay can give a higher link quality, while sometimes going through proxies shows better performance. Since we want to make sure that the link with higher quality (i.e., throughput) will always be chosen for the service provisioning, we skip other parameters and end up the decision making algorithm with the link quality, as depicted in Figure 5.4.

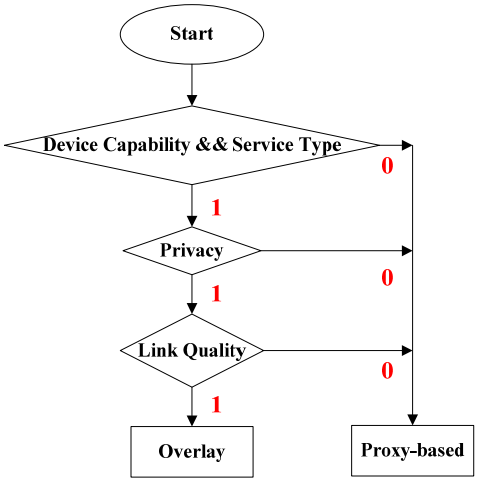

Figure 5- 4 Decision Making Algorithm for Short-range Scenario

5.4. Examples

The parameters in the policy engine play the roles in classifying the Fednet services into overlay services and proxy-required services [1]. Changes in the conditions of the policies will affect the way of service provisioning. Since the working method behind each decision algorithm is almost the same, in this section, we only take the algorithm of the general scenario as an example to illustrate how the policy engine works in the real cases and how the changing polices will make the service provisioning flexible and adaptable.

To understand the Fednet applications and environment, let us make up the story first. We still use the example described in chapter 2.3, as shown in Figure 5.5. Bob and Kate are neighbors. They are quite familiar with each other and always form Fednet to share some pictures or documents. Thus, the Fednet environment here is friendly, which means Bob has a high reputation towards Kate's PN and the vice versa. Bob takes his laptop as his PN's proxy, Kate use her desktop as the service proxy for her PN. One day, Kate wants to share the pictures from Bob's PDA by her laptop. Then they decide to form a Fednet.

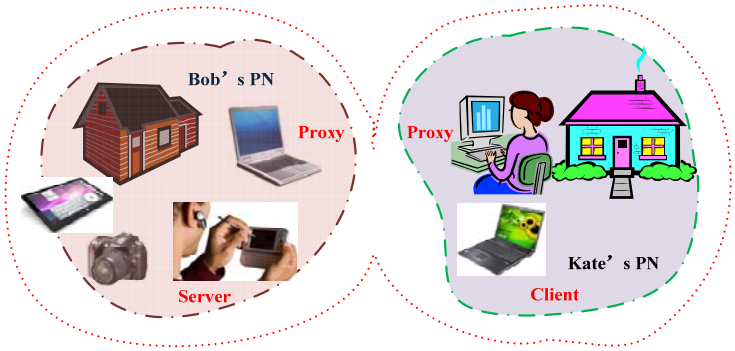

Figure 5- 5 Scenario Setup

In the following, we will discuss three different setting up conditions of the policies to show how the general decision making algorithm in Figure 5.1 has impact on the service provisioning.

5.4.1. Overlay

Set up: Bob defines his PDA as a strong device (1) and this sharing picture service as a

specific service type (1). There is no privacy preference from him (1). He is near to Kate, so service could be directly provides to Kate (1). The battery power of his PDA is high (1). The reputation value that Bob has towards Kate's PN is high (1). The parameters from Kate are the same with Bob's. The number (1 or 0) is the value that the context interpreter gives to the parameters.

Now let us apply those parameters into the algorithm for making the final decision. Figure 5.6 shows the overlay is chosen for Bob, which is also for Kate. Therefore, the approach of service provisioning between Bob and Kate will be overlay, as shown in Figure 5.7.

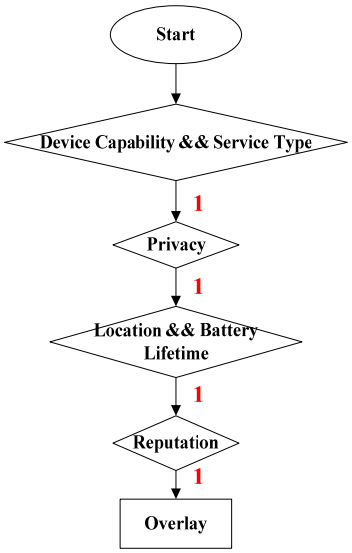

Figure 5- 6 Final Decision for Bob

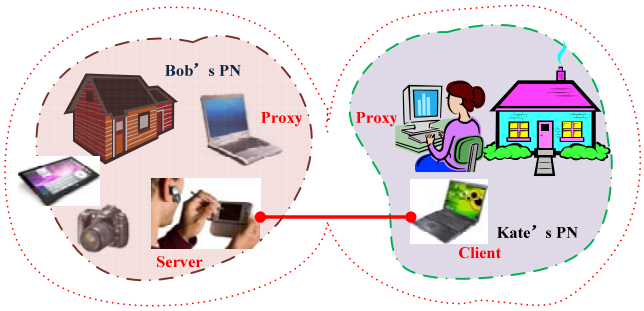

Figure 5- 7 Overlay Approach for Service Provisioning

5.4.2. Proxy-based

Set up: Bob defines his PDA as a strong device (1). Since it is a new PDA, Bob does not want to involve it into many applications. In other word, he prefers the services offered by the PDA to be provided through the proxy. Then he defines the service type as common (0). For other parameters from Bob, they are the same as the set up in 5.4.1. Kate still defines her device and service as strong device (1) and specific service (1). However, she wants to protect her PN, which means she has a high privacy sensitivity (0). The final decisions for Bob and Kate are shown in Figure 5.8 and 5.9.

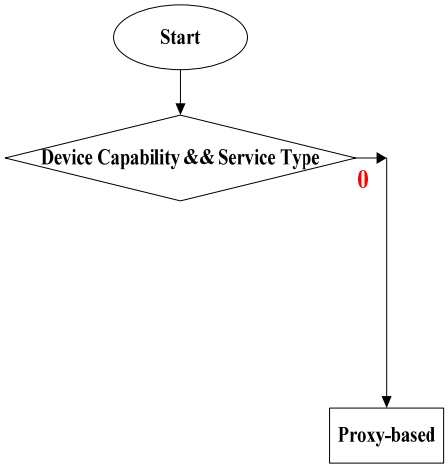

Figure 5- 8 Final Decision for Bob

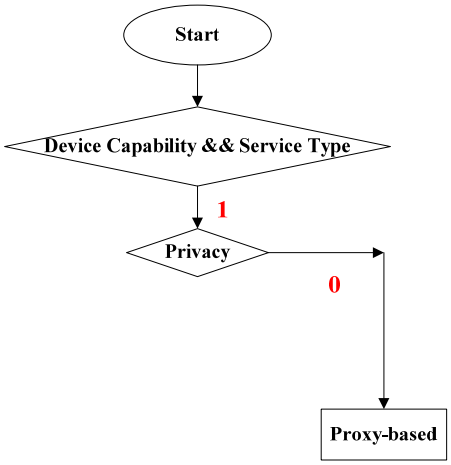

Figure 5- 9 Final Decision for Kate

The final decisions for Bob and Kate are both proxy-based. Thus, the services will be provided by proxy-based approach, as depicted in Figure 5.10.

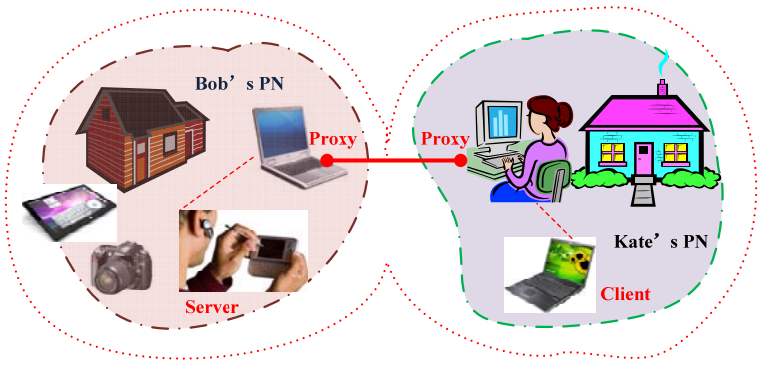

Figure 5- 10 Proxy-based Approach for Service Provisioning

5.4.3. Hybrid

Set up: Bob defines his PDA as a strong device (1) and the picture sharing services as specific service (1). He does not have special requirement about the privacy (1). He is in the office, which is a little far away from Kate. But his service proxy is near to Kate (0). Kate has the same setup as 5.4.1: strong device (1), specific service (1), low privacy (1), near to Bob's proxy (1), high battery lifetime (1) and high reputation towards Bob's PN (1). Figure 5.11 and 5.12 give the final decision for Bob and Kate.

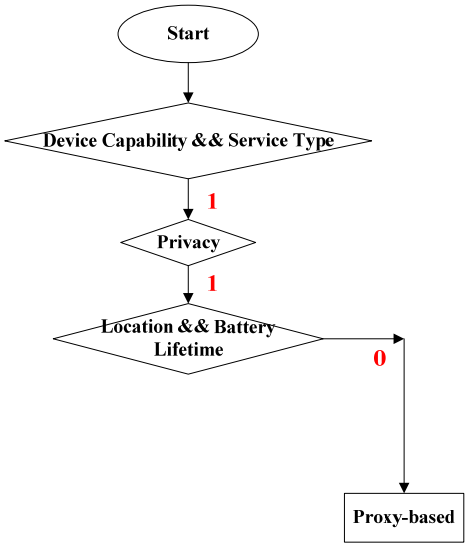

Figure 5- 11 Final Decision for Bob

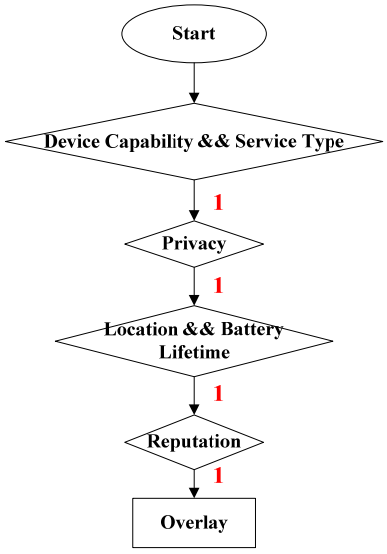

Figure 5- 12 Final Decision for Kate

As we can see from the figures, proxy-based approach is chosen for Bob, while Kate can provide her service directly with her device. The combination of overlay and proxy-based come into the hybrid approach, as shown in Figure 5.13, which we have discussed in chapter 1.2.3.

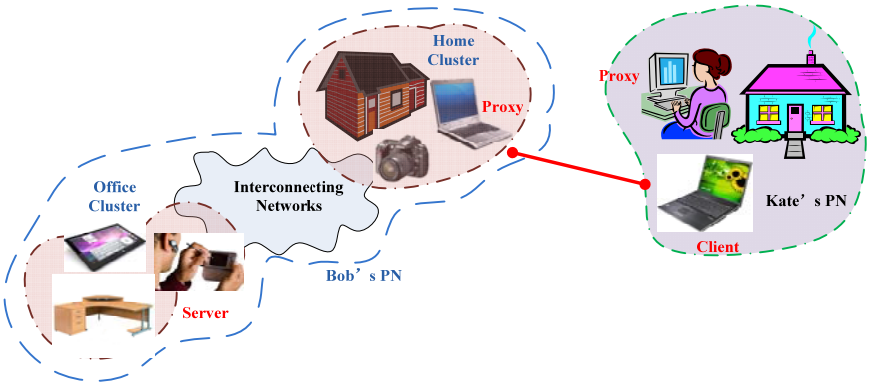

Figure 5- 13 Hybrid Approach for Service Provisioning

5.5. Algorithms' Discussion

As described above, we propose decision making algorithms for different scenarios. Here we would discuss why we motivate these four scenarios and what they are using for. The general scenario is designed for the common Fednet cases, in which the PN owners are familiar with

each other. However, in some occasions, there might be a command for forming a Fednet among unknown people. As to this situation, the general scenario might be not suitable, because we think the reputation criteria probably have more influence on the decision making. To solve this problem, we propose the unfriendly scenario. By taking advantage of the existing LQA method, we set up the short-range scenario, in which the link quality criteria has an important role to choose the service provisioning approach. The confidential scenario is designed specifically for the applications that require security channel.

We realized that there could be more scenarios than we what listed here. But for right now, we only have these scenarios in mind, which are the quite common scenarios in Fednet. In the future work, we plan to propose an algorithm that can suit all the Fednet scenarios.

Actually, in the real applications, there might be some cases that the parameters' values fluctuate between 0 and 1. For instance, the battery power of a device may highly decrease with the time going by (from 1 to 0). When our architecture senses the change of the criteria, the way of service provisioning might be changed according to the decision making algorithms. It probably switches the approach from using the device directly into going through proxy, in order to avoid interruption. However, we do not actively suggest to switch the service provisioning approach until the service is completely delivered. Because it will lead to severe delay and interruption. As the example in the above, Bob and Kate have couples of pictures to share. When the system decides that the service provisioning approach should be changed, we recommend to switch the approach after the current services' transmission are finished.

Apart from that, there might be real time applications in Fednet, such as video streaming, video conference, VoIP, online gaming, e-commerce transactions, chatting and IM (instant messaging) etc. The latency of these applications must be less than a defined value. In other word, we can call them delay sensitive services. For these services, we also recommend to provide the services by one approach in the whole service delivery process to prevent the delay and interruption. The service handler in the service proxy is responsible for triggering the switching process. Regarding how to trigger it, we will not discuss in this thesis.

5.6. Architecture's Discussion

In chapter 4 and chapter 5, we have introduced our new service provisioning architecture for Fednet, and the decision making algorithms designed for some special Fednet application scenarios. Now let us answer the questions that we asked ourselves in chapter 1.3.

1. Can the proposed architecture be implemented in reality?
2. Does our architecture fulfill the requirements listed in chapter 2?

We will answer the first question in the next chapter. Actually, answering the second question is more important, since if the architecture and the decision making algorithms in the policy engine cannot fulfill the requirements, then all the related work and design are meaningless.

To make the service provisioning be context-awareness, we propose a context collector in the Fednet agent to collect the context data. Some of the context data are given by the PN owner when he adds the services into the Fednet, such as device capability, service type and reputation etc. While some of the context are supposed to be sensed by hardware (like sensor or API) to get the information of battery and location. Context collector is responsible for collecting the context information, through which we make the service provisioning context-aware. We assume that there is a user interface in the Fednet agent, through which the PN owner could type his definition towards the parameters. When there is a change on the interface, the context collector should be aware of it. The changing policies will change the result of decision making, like the three examples showed in chapter 5.4. Different parameters' setup might lead to different ways of service provisioning according to the decision making algorithms. It makes the service provisioning in the Fednet flexible and adaptable to the environment.

For QoS and scalability, we think they depend on the used routing protocols and mobility mechanisms. Since our service provisioning architecture is designed on the application layer, the QoS and scalable issues are not dictated by our mechanisms. Therefore, we can say that some of the requirements in chapter 2 have been met, which are context-awareness, flexibility and adaptability.

5.7. Summary

In this chapter, we focused on designing the decision making algorithms in the policy engine for some special Fednet application scenarios. We first introduced what kinds of parameters could be adopted in each scenario. Then we explained what the algorithm is and why the policies are ranked in that way. In the example section, three setup cases were given to better understand how the algorithms work. In the last part of this chapter, we discussed the suitability of our new service provisioning architecture and the proposed decision making algorithms. We concentrated on the requirements listed in chapter 2 and verified whether the architecture can fulfill all the requirements. Some areas, such as context-aware, flexible and adaptable could be achieved. However, duo to the simplicity, the architecture and the decision making algorithms are not fully sufficient, some extensions are needed. For instance, we need to consider more Fednet application scenarios and design algorithms for them. The rest of this thesis will focus on implementing the Fednet prototype, some proposed modules of the service provisioning architecture and the decision making algorithms. For other issues, we will leave them for future work.

6

Fednet Prototype Implementation

6.1. Introduction

Since the overall service provisioning architecture in Fednet has been determined, it is time to look at how every module works in the real test bed. However, before going into the proposed modules, we need to build the Fednet first.

In this chapter, we start by implementing the Fednet prototype based on the existing personal network software named Ppand. The implementation focuses on building a Fednet, including two parts: (1) The Fednet access control, takes place when a new member joins in the Fednet. (2) The service access control, takes place when a Fednet member requests a Fednet. Then some modules in the Fednet agent of the new service provisioning architecture are executed, including context collector, context interpreter and policy engine. We hope that this prototype can show us how the Fednet is formed and how services are provided by the approach chosen through the decision making algorithms in the policy engine.

6.2. Platform

To better understand the Fednet formation and some of the modules in our proposed service provisioning architecture, we implement a Fednet formation prototype, which is based on the Linux operating system and the Personal Network software named Ppand [3].

6.2.1. Hardware

The Fednet prototype is implemented on three standard laptops (HP). All of them have an Intel Core 2 Duo 1.66 GHz processor. The laptops run the Linux 2.6.20 kernel and Madwifi driver version 0.9.2.1 [45]. They are all equipped with WLAN based on the IEEE 802.11b/g

card (3Com's Office Connect 108Mbps11g XJACK PCCard [45]), which has a wireless chip supported by the open source Madwifi driver.

6.2.2. Software

Ppand is the PN software designed by TU Delft WMC group in C programming under Linux, to demonstrate the intra-cluster communication in personal network and to combine all the cross-layer information [3]. The intra-cluster formation and communication program can receive and send packets in the normal way. Node discovery is implemented at the network layer to discover the PN neighbors and maintain the neighbor list. The data packets are encapsulated in UDP and sent with IPv6 packets using link local addresses.

Figure 6.1 shows the schematic view of the prototype implementation of a Node without Gateway Node functionality [3]. The thick arrows denote data traffic and the thin arrows represents the control, routing, and device discovery traffic. Linux provides a virtual network interface in software called Ethertap [46]. The virtual interface implemented by Ethertap is called ppan1, which is used for intra-cluster communication. Intra-cluster data that comes from a user application has to be sent through this virtual interface, which is done by configuring the kernel routing table (step 1 and 2 in Figure 6.1). The program connected to ppan1 is called ppand, implementing the Personal Node discovery and authentication process (step 3). Ppand also maintains the Personal Node neighbor table (PNNT) and makes sure packets on the virtual intra-Cluster interface (ppan1) are encrypted before being sent to a neighbor. Before sending the data packet to the next hop (step 5), ppand will add a second IPv6 header with a link local address of a directly neighboring Personal Node, which makes the packet pass the kernel routing module once more (step 4).

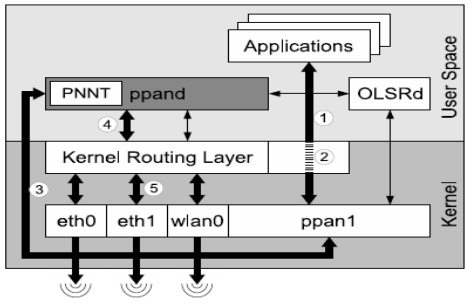

Figure 6- 1 Prototype of Intra-Cluster Communication [3]

Optimized Link State Routing Daemon OLSRD [47,48] (version 4.10.0) is used for routing and addressing in the software. It is configured to send and receive its routing packets only over the virtual intra-cluster interface. And it directly updates the kernel routing table. However, since ppand implements a Personal Node discovery protocol, it is unnecessary to have OLSRd . It is better if ppand informs the routing daemon directly when a Node is discovered or disappears [3]. But in the real test experiment, which is about the link quality assessment later on in this chapter, it will work on the OLSRd.

6.3. Implementation Prototype

After learning about how the Ppand works, we start the implementation of the Fednet formation and some designed modules in the new service provisioning architecture, including context collector, context interpreter and policy engine. The basic task of the implementation is we should first be able to send and receive packet through the socket on the application layer of the PN software. Because from the low layers of view, we can consider that the Fednet formation and the service provisioning processes are actually about exchanging packets among the FM, FA and the server and the client. To achieve that, we first create a socket on the application layer. Then we realize the sending and receiving packets functions on that socket. Finally, we show the prototype architecture and execute the functions in FM and FA. In the following, we will briefly explain the first two steps and discuss the prototype in details.

6.3.1. Creating Application Socket

Before setting up everything, first of all, we define the packet format that we want to send and receive, which we call Fednet packet. All the types of parameters mentioned in the context collector (chapter 4.4.2) are contained in the Fednet packet structure except network dynamics, which will not be used in the policy engine in any scenario. The Fednet packet is shown in Figure 6.2.

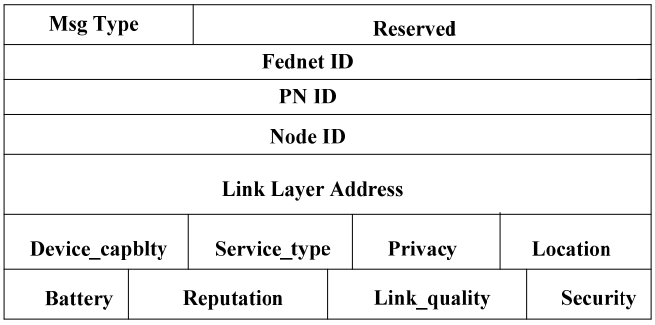

Figure 6- 2 Fednet Packet Format

Then we initialize a socket and its interface for exchanging the Fednet packets, which is called Fednet socket.

```
int    fd_init(__u32 pnid, __u32 nodeid, int bindsocket)
{
       struct sockaddr_in6    fd_addr;
                      :
       fd_fd = socket(PF_INET6, SOCK_DGRAM, 0);
       if (fd_fd < 0) {
       perror("socket");
       return -1;

       if (bindsocket) {
       memset(&fd_addr, 0, sizeof(fd_addr));
       fd_addr.sin6_family        = AF_INET6;
       fd_addr.sin6_port           = htons(PN_PORT_FD);
       res = bind(fd_fd, (struct sockaddr *)&fd_addr, sizeof(fd_addr));
       if (res < 0) {
              perror("bind");
              return -1;
       }
        }
                      :
                      :
}
```

6.3.2. Sending and Receiving Packets

As we mentioned before, realizing sending and receiving Fednet packets functions on the *fd_fd* socket is the crucial step of implementing our prototype. Several functions are involved to realize that, including int wif_recv_fd_packet, int if_handle_fd_packet, int fd_send_msghdr

and int fd_read_packet. Here we only show couples of lines of the codes. For details, please refer to the Appendix.

```
int    fd_send_msghdr(int if_idx, struct pn_fd_msghdr *msg)
{
                    :
       struct sockaddr_in6    toall;

       l = sendto(fd_fd, (void *)msg, sizeof(*msg), 0, (struct sockaddr *)&toall,
                sizeof(toall));
                    :
                    :
}

int    fd_read_packet(struct sockaddr_in6 *ofrom6, struct pn_fd_msghdr *omsg)
{
                    :
       struct sockaddr_storage    from;
       socklen_t    fromlen;

       fromlen = sizeof(from);
       l = recvfrom(fd_fd, buf, sizeof(buf), 0,
              (struct sockaddr *)&from, &fromlen);
                    :
                    :
}
```

6.3.3. Prototype Architecture

We have three laptops to do the implementation. To form a Fednet, we setup two PNs in the following way: suppose PN1 consists of two laptops, one is working as the Fednet agent of PN1 (FA1) and the other will act as the Fednet manager (FM) and the service proxy. PN2 has one laptop, which is also used as the FA of the PN (FA2). After the Fednet formed, FA1 (client) will request a service from FA2 (server), as shown in Figure 6.3.

Figure 6.3 also illustrates the interactions between FM and FA when two PNs are federating. The upper part is about the Fednet access control, the lower part shows the service access control. The positions with red points are where our new service provisioning architecture work. To clearly demonstrate this process, we will describe the details of the message exchanging in the two level access controls step by step with some programming codes.

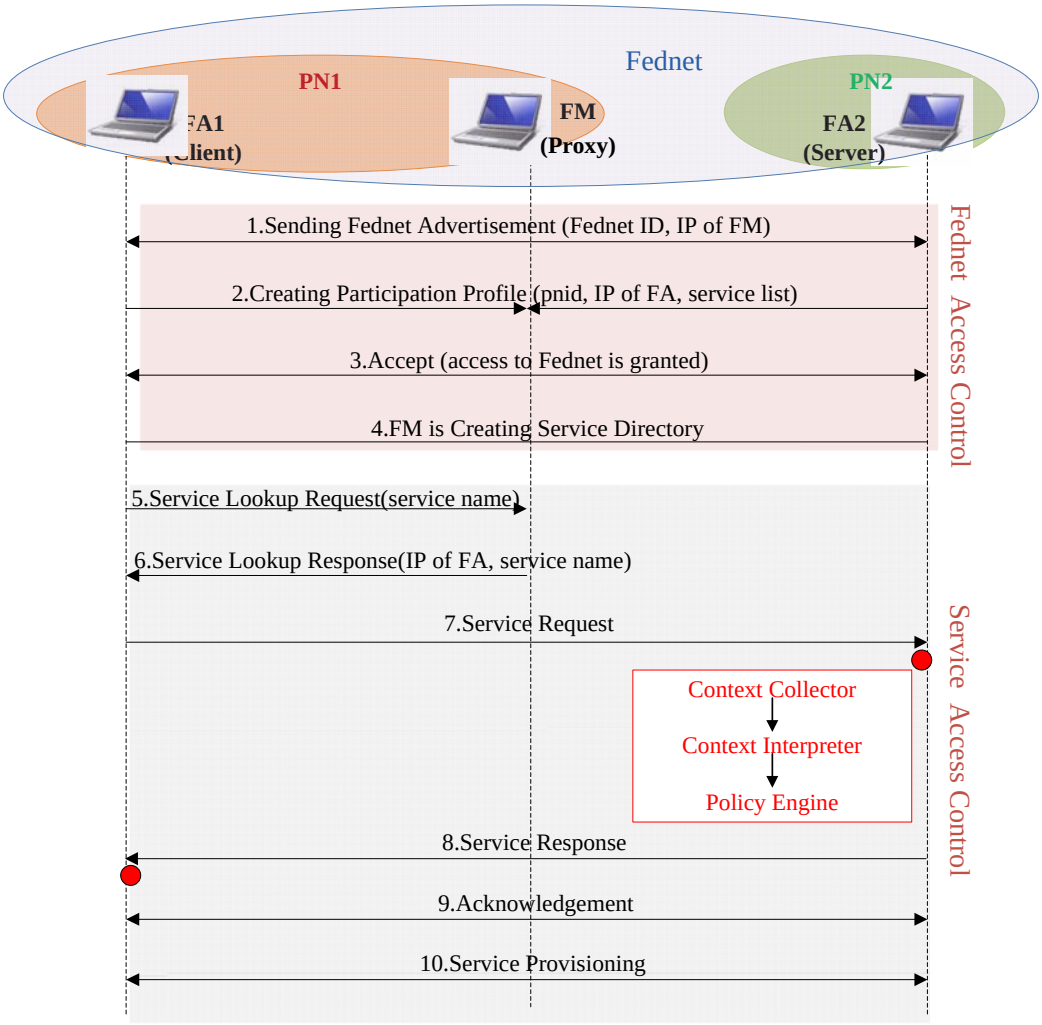

Figure 6- 3 The Implementation Architecture of the Prototype

a) Fednet Access Control

Step 1 to Step 4 show the Fednet access control, which takes place when a new member joins a Fednet, carried out by the FM.

Step 1. Sending Fednet Advertisement

When there is a request or purpose to form a Fednet, in the initial phase, the FM and the FA of the PNs need to discover each other. FM first sends an advertisement with the Fednet ID and its IP address to his neighboring devices. From the Fednet's point of view, it is discovering a new Fednet member. From the PN's point of view, it is discovering an e Fednet to join in.

```
int    fm_send_advertizement(struct interface *ifp, struct pn_fd_msghdr *msg)
{
       msg->fd_fdid=fd_fdid;
       strcpy(msg->ipfm,"3000::9");

       int t;
        t=fd_send_msghdr(ifp->if_index, msg);
                        :
                        :
}
```

Step 2. Creating Participation Profile

If a PN wants to participated in the Fednet, the FA of this PN should create its participation profile after it receives the Fednet advertisement from the FM, including the PN ID/name, IP address of FA and the list of services offered to the Fednet.

```
int    fa_particiption_profile(struct interface *ifp)
{
                        :
       struct pn_fd_msghdr    fdmsg;
       fdmsg=*fednet_service_three(ifp);

       strcpy(fdmsg.pnid, “2”);
       strcpy(fdmsg.ipfa, "3000::3");
       strcpy(fdmsg.service_name, “file3”);

       int t;
       t=fd_send_msghdr(ifp->if_index, &fdmsg);

                        :
                        :
}
```

Step 3. Accept

The FM informs the FAs by an accept message, if the accesses of the PNs are granted, namely the first access control.

```
int    fm_membership_credential(struct interface *ifp)
{
                        :
}
```

Step 4. Creating Service Directory

After FM gets the services needed to form the Fednet, it will create a service directory to store the information about the Fednet members and their services, including PN id, IP address of FA and list of services.

```
struct pn_fd_msghdr* fm_service_directory(struct interface *ifp, struct sockaddr_in6 *from6,
                        struct pn_fd_msghdr *fdp)
{
                        :
      checkfdp = servicetable;
      while(checkfdp)
      {
                           :
            if (!checkfdp->fd_valid)
            printf("INVALID");
            checkfdp=checkfdp->next_fm_service;
      }

      return servicetable;
}
```

b) Service Access Control

After the Fednet is formed, PN1 wants to request a service from PN2. Then there comes the service access control, which takes place when a Fednet member requests a Fednet service, carried out by the FA of the PN. It is shown from step 5 to step 10 in Figure 6.3.

Step 5. Service Lookup Request

When a client in PN1 wants to request a service in the Fednet, it sends a service request to the FA1 of his PN. The FA1 will first initialize a service lookup request to the FM to look up where the service is.

```
int    fa_service_lookup_request(struct interface *ifp)
{
                           :
      strcpy(fdmsg.service_name,"picture3");
                           :
}
```

Step 6. Service Lookup Response

The FM searches in the service directory for the service. If the service is found, FM returns the service information, including its PN id, node id, IP of the FA of its PN and the service name. If the service is not found, FM returns an empty list. In some cases, there might be more services which can match the request. However, since we do not implement the server selection module here, we just assume there is only one service matched.

```
struct pn_fd_msghdr * fm_service_lookup(struct interface *ifp, struct sockaddr_in6 *from6, struct pn_fd_msghdr *omsg)
{
	struct pn_fd_msghdr *fdp = servicetable;
			:
	if(fdp && (strcmp(omsg->service_name,fdp->service_name)==0)){
	printf("The service is found in pnid=%d,    nodeid=%d, ip of FA=%s, service=%s\n",
		fdp->fd_pnid, fdp->fd_nodeid, fdp->ipfa, fdp->service_name);
	}
			:
	return fdp;
}

int fm_service_lookup_response(struct interface *ifp, struct sockaddr_in6 *from6, struct pn_fd_msghdr *fdp)
{
			:
}
```

Step 7. Service Request

At the client side): After informing where the service is in PN2, the client forwards a service request to the server.

```
int fa_service_request(struct interface *ifp, struct sockaddr_in6 *from6, struct pn_fd_msghdr *fdp)
{
			:
}
```

At the server side): If the server agrees to provide the service, it will send a service response back to the client. But before that, the FA2 will first decide a way of service provisioning for the server. Our proposed service provisioning architecture for the Fednet is implemented here,

as highlighted by the red point under step 7 in Figure 6.3. It mainly includes the functions of the context collector, context interpreter and policy engine. As to the context collector, we assume there could be a user interface which allows the PN owner to define some context data, such as the service type and his privacy sensitivity etc. However, for easy discussion, in our prototype, we have encapsulated all the context data in the Fednet packet. The context data are transferred into 0 or 1 by the context interpreter and then sent to the policy engine. The policy engine makes a final decision to use overlay or proxy-based to do the service provisioning for this server, based on the context and the user preference.

```
int    fa_context_interpreter(struct interface *ifp)
{
                        :

      /* interpret context into binary value (0 or 1) */
      int cp;
      if(strcmp(fdmsg->device_cpblty,"strong")==0){
      cp=1;
      }
      else
      {cp=0;}
                        :
                       :
}

int    fa_policy_engine(struct interface *ifp)
{
      int fa_c_i;
      fa_c_i=fa_context_interpreter(ifp)
                        :
      switch(fdp->fd_scenario) {
      case PN_FD_SCENARIO_GENERAL:
                       :
                       :
   }
}
```

Step 8. Service Response

At the server side): The server returns the service response (i.e., service provisioning decision) to the client.

```
int fa_service_response(struct interface *ifp, struct sockaddr_in6 *from6,struct pn_fd_msghdr *fdp)
{
                        :
                        :

}
```

At the client side): Once the clients receives the response, similarly, the service provisioning modules in the FA1 will also choose an approach for consuming the service: via overlay or proxy.

Step 9. Acknowledgement

There is an acknowledgement between the client and the server to inform each other the service provisioning way and the correct IP addresses to contact.

```
int    acknowledgement (struct interface *ifp, struct sockaddr_in6 *from6, struct pn_fd_msghdr *fdp)
{
                        :
                        :
}
```

Step 10. Service Provisioning

Service is provided, using the approach chosen by the policy engine.

```
int    send_packet(struct interface *ifp, struct sockaddr_in6 *from6, struct pn_fd_msghdr *fdp)
{
                        :
                        :
}
```

6.4. Summary

In this chapter, we implemented a Fednet prototype mainly about the two-level Fednet access controls: the Fednet access control and the service access control. Figure 6.3 showed the every single step of our prototype architecture. Based on that, some modules in the Fednet agent in our new service provisioning were also implemented, including the context collector,

context interpreter and policy engine. The prototype gave us an overview of how the messages are exchanged between Fednet agent and Fednet manager when the Fednet is formed and when the client asks a service from the server.

7

Simulation and Experiment

7.1. Introduction

In this chapter, we simulate some scenarios mentioned in chapter 5. The simulations are running based on the Fednet prototype we built on the Ppand software. Since there are many criteria involved in the policy engine, we just concentrate on simulating some of the context data, such as service type, privacy and device capability etc. We want to verify whether our policy engine can make the service provisioning decision flexible and adaptable to the changing environment. Then some explanations are given about the simulation results. To take the advantage of the existing link quality assessment (LQA) method, we carry out a real test experiment based on the link quality parameter. The test bed also runs on Ppand. Throughput is used to compare the performance of overlay and proxy-based. Then we analyze the experiments results and conclude that the flexible service provisioning mechanism is better than always using a single approach. Finally, we summarize the whole chapter.

7.2. Simulation

In the Fednet prototype, we executed the function of policy engine, which is the core part of our service provisioning architecture. But how the policy engine will make the way of service provisioning be flexible and adaptable to the changing environment has been not shown yet. Therefore, in this section, we will set up some scenarios to simulate the decision making algorithms that we proposed in chapter 5. And in each scenario, we mainly focus on one or two certain parameters. Then we analyze the simulation results and draw the conclusions.

7.2.1. Scenario 1

In the following, we will introduce how we set up the parameters for each scenario in the prototype.

Setup)

It is an unfriendly environment; the algorithm in Figure 5.2 will be used. Since reputation plays an important role in this anonymous environment, we want to see how this parameter mainly affects the way of service provisioning. In the simulation, device capability and service type will always be set up to 1 and we ignore the influence from the parameter of location and battery. The value of privacy and reputation changes between 0 and 1. Then we have four cases, 0&&0, 0&&1, 1&&0 and 1&&1.

Result and Analysis)

The simulation result of scenario 1 is shown in the following figure. As we can see, if the privacy of the PN owner is high, proxy-based approach is chosen for service provisioning. Regarding the reputation, when it is low, going through proxy is required. The case of using overlay is lower than that of using proxy-based, which only comes when the reputations between the client and the server towards each other are high and the PN owners are not very sensitive about the privacy. That is because the PN owners' high sensitivity of protecting their PNs from the unfamiliar people.

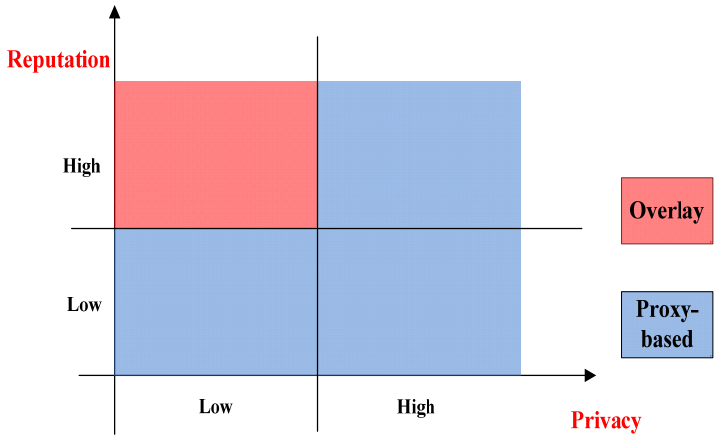

Figure 7- 1 Simulation Result of Unfriendly Scenario

7.2.2. Scenario 2

Setup)

It is a friendly environment; the algorithm in Figure 5.1 will be adopted. Our concentration is on the parameters of device capability, service type and privacy. Location, battery and reputation will be not taken into account.

Result and Analysis)

Figure 7.2 shows us the decision making results worked out by device capability, service type and privacy under the friendly environment. It is depicted that as long as the device capability is defined as weak, proxy-based is used anyhow. There is even no need to consider the other two parameters. The common service type and high privacy also have the same situation. The only case of choosing overlay is that the PN owner has strong device capability, specific service type and low privacy, which is the area highlighted by A. For other cases, they all go through the service proxy.

Why is the probability of using proxy-based so much higher than that of overlay? We think that it is because we have so many parameters involved in the policy engine to make the final decisions for service provisioning. Each policy plays a role in classifying the service into overlay and proxy-based. It is not like the situation with only one factor, in which the probability of using overlay and proxy is half to half. As we can see from Figure 5.1 to 5.4, the decision making algorithms are designed in a hierarchy way. Wherever there are two parameters working in one layer, the probability of choosing overlay is only 25%, which becomes less after going through other layers. In addition, Fednet is a user-centric network, user's preference is highly considered. If the user does not to reveal his PN structure or dislikes his device involved in Fednet, proxy-based is used anyhow.

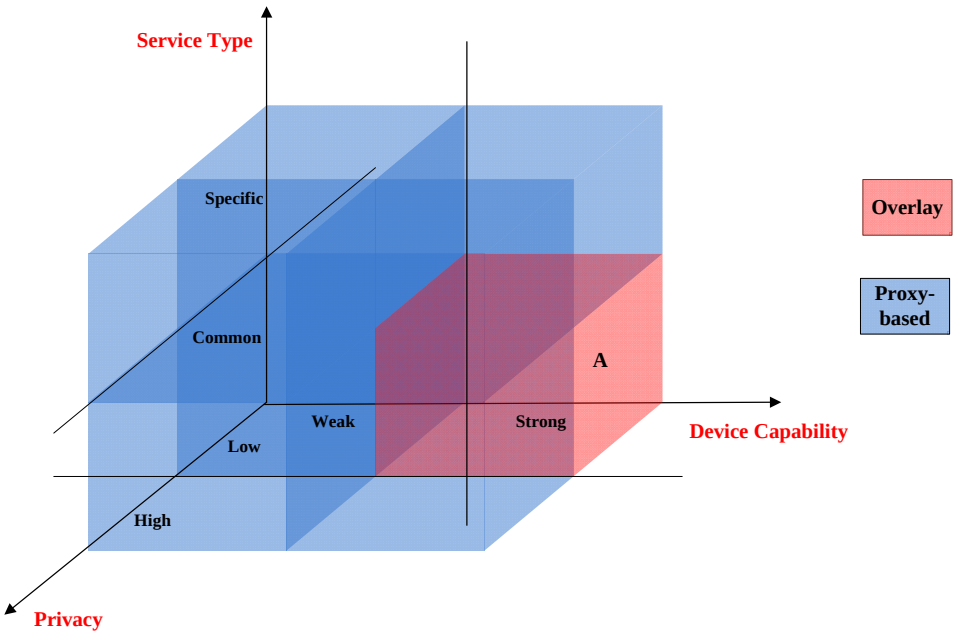

Figure 7- 2 Simulation Result of Friendly Scenario

7.2.3. Scenario 3

Setup)

It is a friendly environment; again we use the algorithm in Figure 5.1. But this time, we only consider the distance between server, client and service proxy. The approach with the shortest path will be chosen for service provisioning.

Result and Analysis)

The decision made by location is depicted in Figure 7.3. The red points on the X and Y axis represent where the server proxy and the client proxy are. The points in the pink shadow are marked to use overlay approach. Now let us take A,B,C and D to make examples. If the client is in A and the server is in C, then overlay is chosen, since it is the shortest path, as depicted by the dashed line. If the client is in B and the server is in D, the proxy-based is applied, because the path between the proxies is shorter than the direct communication between the client and the server, as shown by the solid line. If the client in B and the server is in C, then that is the hybrid approach, where the client is using proxy and the server is using overlay.

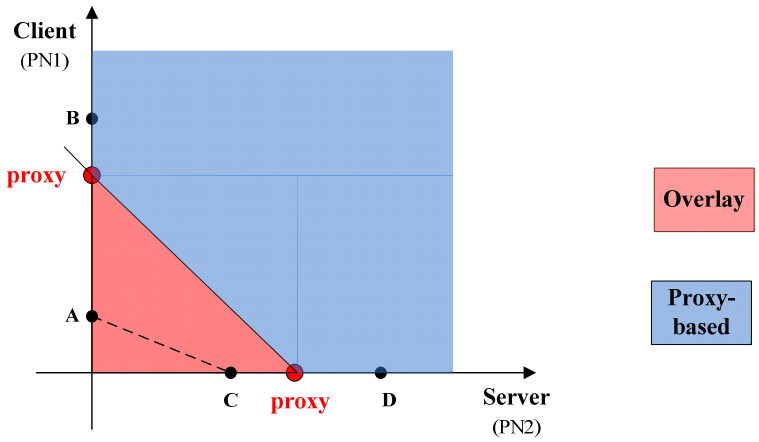

Figure 7- 3 Simulation Result of Location Parameter in Friendly Scenario

Admittedly, in the location simulation, we only compare the path length among the client and the server and the proxies, which is not enough to make the decisions for service provisioning. But here we just want to give a figure to show how the location affect the way of service provisioning.

7.3. Real Test Experiment

The initial motivation for us to propose the new service provisioning architecture is that we want to make the service provisioning in Fednet be flexible and adaptable to the changing environment. We believe that trading off between overlay and proxy-based could get higher service quality or fulfill the user's requirements in a better way than always using one approach. Is what we believe real? If it is not true, people may ask why not just use one approach and why we need to be adaptable and flexible? Then all the research, design and implementation are meaningless. Therefore, to prove the flexible mechanism is better than the single one, we carry out a real test experiment in this section.

Since the service provisioning in Fednet is a complex process, our work is not yet capable of verifying every parameter and every scenario that we proposed for the policy engine due to the time limitation. Here we only choose the link quality criteria and the short-range scenario to implement a simple test to validate our proposed scheme.

7.3.1. LQA

The test of the link quality parameter is based on the Link Quality Assessment (LQA) method proposed in [43]. The goal of LQA is to provide information for maximizing the quality of the end-to-end links and minimizing the packet loss and delay to achieve high throughput. The effectiveness of accurate and fast LQA is demonstrated by feeding it into the routing layer to enable the route decisions to adapt faster during changing situations, especially in mobile scenarios, by only using hello packets to determine the link quality. The test bed architecture is shown in Figure 7.4.

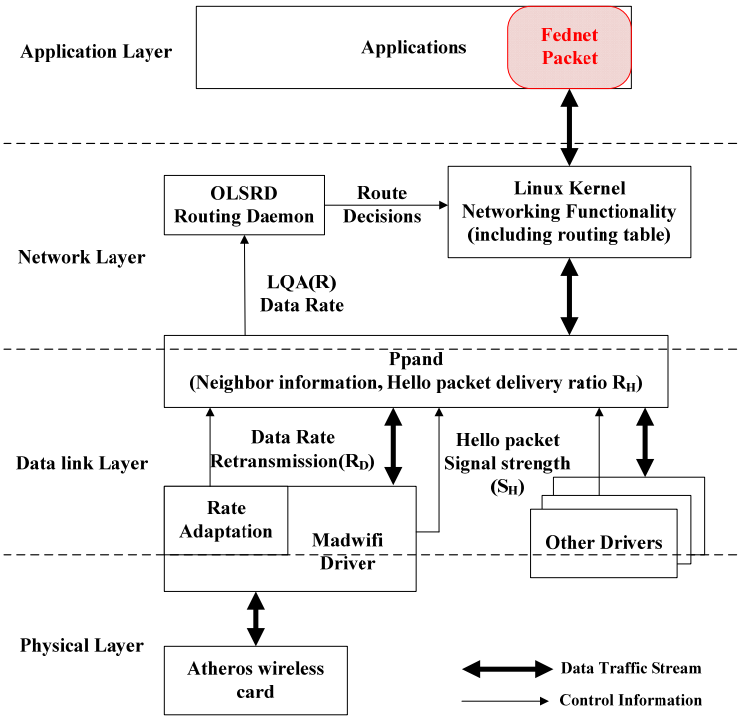

Figure 7- 4 Test Bed Architecture

As seen in Figure 7.4, Ppand sits in between the data link layer and the network layer, which generates and processes hello messages to discover neighbors. The Madwifi driver plays the role of physical and link layer, being responsible for forwarding the basic physical channel information such as signal strength and the link layer information including the available link types, number of transmitted and retransmitted packets with each neighbor to the Ppand. The LQA method in Ppand estimates the link quality and forwards it to the routing layer. OLSRD

is working on the routing layer, which uses the LQA calculated by Ppand to make route decisions and give those decisions to the Linux kernel routing table. The data packets sent from the application layer are sent to the kernel routing functionality, then, to the next hop based on the routing table. The experiment results in [43] show that the proposed LQA method can lead to faster and smarter routing decisions and higher end-to-end throughput compared to traditional methods.

7.3.2. Setup

Our link quality test experiment is carried out in a short-range and mobile environment by running OLSRD on top of Ppand. To compare the throughput, we set up the overlay and proxy-based scenarios respectively.

Figure 7.5 shows the overlay scenario for service provisioning in Fednet. The two laptops represent the two federated PNs. As we mentioned above, it is a mobile environment. One of the laptops (PN1) is stationary and the other one (PN2) is mobile (the mobile node's moving range is marked as the arrow). The path is designed so that the mobile node moves from the point where it starts to send packets until reaching the point where it cannot receive any data packets from the stationary node.

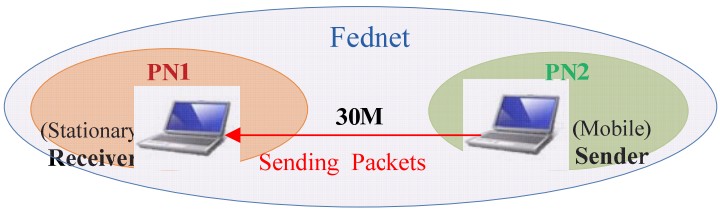

Figure 7- 5 Overlay Scenario

Proxy-based scenario is shown in Figure 7.6. There is one more stationary laptop, acting as the service proxy in PN1. The mobile node sends packets to the proxy too. The receiver will receive the packets via the intermediate node when the link quality by using proxy is better than the direct communication.

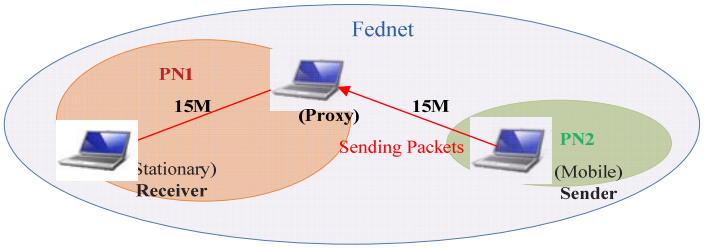

Figure 7- 6 Proxy-based Scenario

Each scenario is repeated three times to measure the throughput with almost exact time control. The speed of the mobile node is 1m/s. We can assume each movement of the mobile node is independent. The sender keeps sending UDP packets as fast as it can via the routing table provided by OLSRD. In the experiments, all the nodes use the fixed data rate of 54Mbps.

As mentioned above, we carried out the experiment in a short-range Fednet scenario. Thus, the decision making algorithm shown in Figure 5.4 will be used to choose a service provisioning approach. We suppose that the device capability, service type and privacy sensitivity of the participated PN members are both defined as strong, common and low. Therefore, according to the algorithm, we only need to consider the link quality for the final decision making of service provisioning.

7.3.3. Results and Analysis

Figure 7.7 shows the throughput between the two PNs by using overlay approach. As we can see, the performance tends to reduce with the increase of the distance, extremely after 15s. From 22s-30s, it seems that the service of overlay is unreachable, as there is almost no throughput generated in this interval.

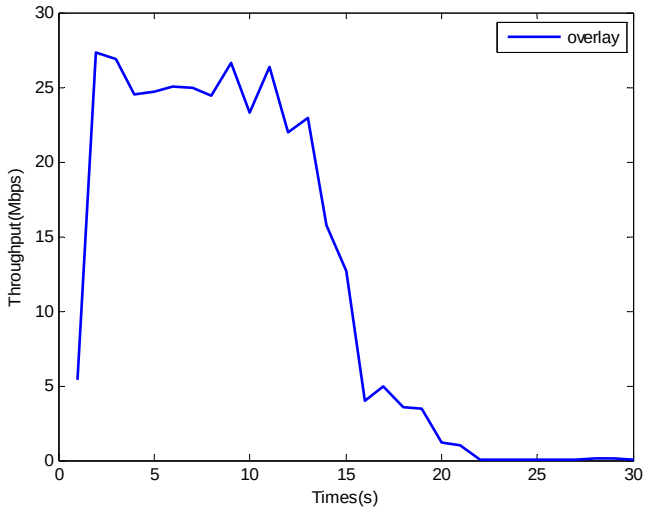

Figure 7- 7 Throughput of Using Overlay Approach

The throughput of using proxy-based approach also decreases with the distance, as depicted in Figure 7.8. But, not like the overlay approach, the service has still been provides after 20s, as the communication between the sender and the receiver is still maintained via the proxy.

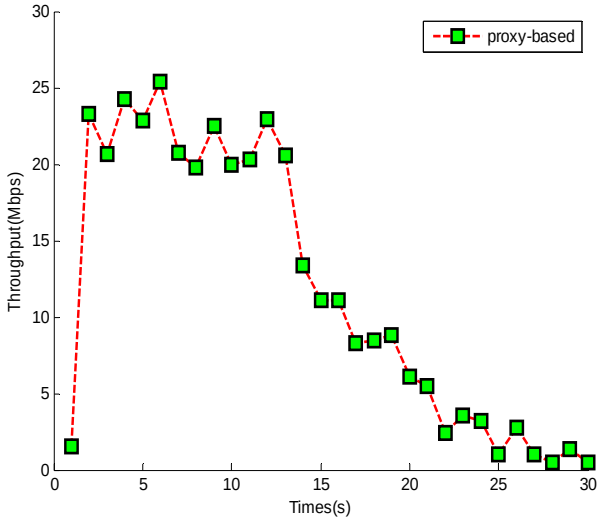

Figure 7- 8 Throughput of Using Proxy-based Approach

Generally speaking, the overlay and proxy-based approach both suffer from the high degradation of the performance after 15s. However, the two approaches result differently in different time interval, as compared in Figure 7.9. In the first half (0-15s), overlay performs

better than proxy-based does. While in the second half, the throughput of using proxy is higher, since the link quality via the two hops is higher than that of the direct communication as the distance increases. The time at 15s can be considered as the approach switching point. Before 15s, the service can be provided by overlay. After 15s, it is better to switch the way of service provisioning into proxy-based, which gives a higher link quality than still using the overlay. The LQA enables us to choose the link with the higher quality to adapt to the changing topology. Base on that, we can make our service provisioning in Fednet be flexible and adaptable between overlay and proxy-based approach, which provides better performance than always using the single approach. The service proxy here is used to maintain the link quality.

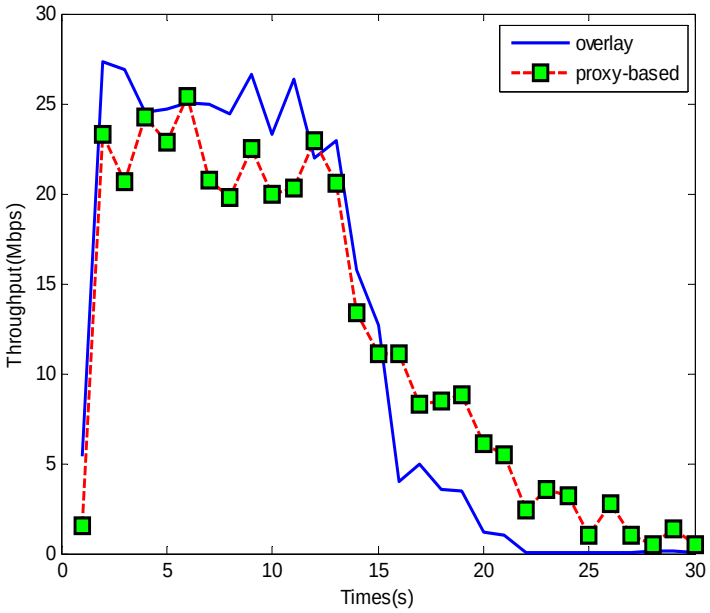

Figure 7- 9 Comparison of Overlay and Proxy-based Approach

7.4. Summary

To validate that our service provisioning architecture could be flexible and adaptable to the changing environment, in this chapter, we simulated some Fednet scenarios by changing one or two certain parameters involved in the corresponding decision making algorithms. The simulation results showed that our architecture could make the decision adaptable to the

context data it sensed, like the user's preference. We also found that the probability of using proxy-based is higher than that of using overlay, which is because the of hierarchy algorithms design.

The second part of this chapter introduced an experiment carried out on the link quality parameter based on LQA, with the aim of proving the flexible service provisioning scheme is better than always using a single approach. The set up of the test bed was not easy, because the position of the sending packets node and receive packet node needed to be measured carefully. Overlay and proxy-based scenarios were built respectively, and throughput was used to compare the performance of these two approaches. The result showed that LQA always enabled us to choose the way with higher link quality, according to which, we can make the service provisioning approach switch between the overlay and proxy-based, since it provided better performance than the single approach did.

8

Conclusion and Future Work

8.1. Conclusion

With the rapid development of the wireless technology, there is no doubt that more and more electronic devices will appear around one person and we can envisage that they will be connected via the communication networks in the future. Personal network and their federations are proposed as one of the promising future concepts. In this thesis, we have first introduced the background of personal network and the federated personal networks called Fednet.

We started out by showing the state of the art of the Fednet, from which we know there are two approaches for providing services in Fednet, one is overlay and the other is proxy-based. Each of the approaches has the advantages and disadvantages. Then there comes our question: which approach will the PN owner choose? According to what? Is that possible to make the service provisioning approach be flexible and adaptable to the changing environment? Based on these questions, we got our research started.

In chapter 2, we first gave the requirements we believe the service provisioning in Fednet should meet. Then, in chapter 3, we investigated whether the existing technologies that can be used to satisfy the requirements. Evidently, there are plenty of partial solutions that can fulfill individual requirement, but no overall solution.

Therefore, we decide to propose a new service provisioning architecture for Fednet, which was discussed in chapter 4. There were some modules designed, but we mainly focused ourselves on the functions in the Fednet agent, especially the policy engine. We used a whole

chapter, which was chapter 5 to explain some Fednet scenarios and the decision making algorithms we designed in the policy engine for every single scenario. Furthermore, we assessed the architecture's ability to meet the requirements we established as far as is possible for something that is not implemented, nor even fully specified.

After that, in chapter 6, we implemented a Fednet prototype based on the Ppand software. Since all the preconditions of our designed architecture were there was a Fednet, we wanted to show people how the Fednet is formed; what will happen between the Fednet agent and the Fednet manager when a member joins into the Fednet or asks a service. Further in this chapter, the modules in FA were executed in the prototype. To prove our decision making algorithms can make decisions according to the user's preference and the changing context, in chapter 7, we carried out some simulations and a real test bed experiment based on one or two parameters due to the time limitation. The simulations results showed that our new service provisioning mechanism could make the way of service provisioning flexible and adaptable to the environment and the user's requirements.

The proxy-based approach was originally proposed to protect the PN owner's privacy [1]. But in our research, we found that the service proxy could also be used to maintain the link quality, as tested in the LQA experiment. In addition, it can be as the intermediate node if the server and client are far away from each other where location is concerned; or be used to sustain the communication if the server or the client are out of power. To sum up, the service proxy might be used in more occasions than just privacy protection.

8.2. Future Work

Admittedly, during our research, we point out more issues and problems. However, it is not possible for us to address all of them. Here we would like to some ideas about the future work for the extension of this project. At first, in this thesis, we only use throughput to study the performance characteristics of the overlay and proxy-based approach. Actually more parameters would be proposed for the performance analysis of the two approaches, such as delay caused by service proxies and computational overhead by the policy engine.

Secondly, as we discussed in the chapter 4, some of the proposed modules in the service provisioning architecture need to be designed in details, such as the service handler in the service proxy, the service parser for server selection. And we could think more Fednet scenarios and design decision making algorithms for them. Or we would like to figure out if there could be an algorithm for generally being used in all Fednet scenarios? Thirdly, the quality of service requirement listed in chapter 2 has not been fulfilled, which still needs more investigations to address this issue.

Another future work will be assigning backups to the Fednet manager in Fednet. Since FM is the only service management point in the Fednet, a single failure to this point might break down the whole system. Therefore, it might be necessary to assign a backup of the FM.

I believe that there might be some feedbacks and comments from the people who read this thesis or are interested in the personal network topic. If there is a chance for me to continue with the further study, I would like to work more with this project and publish a paper about my findings and results.

Reference

[1]. M. Ibrohimovna and S.M. Heemstra de Groot, “Policy-based Hybrid Approach to Service Provisioning in Federated of Personal Networks”, The Netherlands.

[2]. I.G. Niemegeers and S.M. Heemstra de Groot, “FEDNETS: Context-Aware Ad-Hoc Network Federations”, Wireless Personal Communications, Vol. 33, pp. 305–318, 2005.

[3]. M. Jacobsson, “Personal Networks – An Architecture for Self-Organized Personal Wireless Communications”, PhD Thesis, TU Delft, The Netherlands, June 17, 2008.

[4]. M. Ibrohimovna and S.M. Heemstra de Groot, “Proxy-based Fednet for Sharing Personal services in Distributed Environments”, The Netherlands.

[5]. M. Ibrohimovna, S.M. Heemstra de Groot, J. Goseling, H. Benz, J. Stoter, “Federations of Personal Networks”, PNP2008 Deliverable DA.2.5, The Netherlands, 2008.

[6]. J. Hoebeke, G. Holderbeke, I. Moerman, M. Jacobsson, V. Prasad, N.I. Cempaka Wangli, I.G. Niemegeers and S.M. Heemstra de Groot, “Personal Network Federations”, The Netherlands.

[7]. X. Xiang and X. Wang, “A Scalable Geographic Service Provision Framework for Mobile Ad Hoc Networks”, Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications, 2007.

[8]. N. Le Sommer, “A Framework for Service Provision in Intermittently Connected Mobile Ad hoc Networks”, IEEE, 2007.

[9]. Riva, T. Nadeem, C, Borcea and L. Iftode, “Context-Aware Migratory Services in Ad Hoc Networks”, IEEE Transactions on Mobile Computing, Vol. 6, No. 12, December 2007.

[10]. A. Harwood and R. Balsys, “Peer Service Networks-Distributed P2P Middleware”, Australia, August 15, 2003.

[11]. E. Hammami, “Towards a Peer-to-Peer Content Discovery and Delivery Architecture for Service Provisioning”, Proceedings of the Fourth European Conference on Universal Multiservice Networks , 2007.

[12]. X. Xiang, Y. Shi, and L. Guo, “Rich Metadata Searches Using the JXTA Content Manager Service”, 18th International Conference on Advanced Information Networking and Applications (AINA’04), Vol. 1, pp.624, 2004.

[13]. X. Gu, K. Nahrstedt and B. Yu “SpiderNet: An Integrated Peer-to-Peer Service Composition Framework”, US.

[14]. M. Mushtaq and T. Ahmed, “QoS Provisioning for Video Streaming Over SP-Driven P2P Networks Using Admission Control”, IEEE, France, 2009.

[15]. X. Gui and K. Nahrstedt, “On Composing Stream Applications in Peer-to-Peer Environments”, IEEE Transactions on Parallel and Distributed Systems, Vol. 17, No. 8, August 2006.

[16]. V. Kaldanis, S. Tan and I. Windekilde, “Business Aspects of PN Service Provisioning in Federated Mobile P2P Networking Environments”, 15 May 2007.

[17]. V. Kaldanis S. Tan and R. Roswall, “Charging and Billing Schemes for PN Service Provisioning in Federated P2P Environment”, The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07), 2007.

[18]. T. Dimitrakos, D. Mac Randal, S. Wesner, B. Serhan, P. Ritrovato, and G. Laria, “Overview of an architecture enabling Grid based Application Service Provision”, UK.

[19]. B. Vassiliadis, K. Giotopoulos, K. Votis, S. Sioutas, N. Bogonikolos and S. Likothanassis, “Application Service Provision through the Grid: Business models and Architectures”, Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC’04), 2004.

[20]. A. Brocco and B. Hirsbrunner, “Service Provisioning Framework for a Self-Organized Grid”, 2009.

[21]. V. Naik, P. Garbacki and A. Mohindra, “Architecture for Service Oriented Solution Delivery Using Grid Systems”.

[22]. E. Byuny, J. Jang, W. Jung and J. Kim, “A Dynamic Grid Services Deployment Mechanism for On-Demand Resource Provisioning”, IEEE, 2005.

[23]. Y. LI, F. RAO, Y. Chen, D. Liu and T. Li, “Services Ecosystem: Towards a Resilient Infrastructure for On Demand Services Provisioning in Grid”, Proceedings of the IEEE International Conference on Web Services (ICWS’04).

[24]. S. Panagiotakis, N. Honssos and A. Alonistioti, “Integrated Genetic Architecture for Flexible Service Provision to Mobile Users”, IEEE, 2004.

[25]. N. Houssos, A. Alonistioti and L Merakos, “Advanced Adaptability and Profile Management Framework for The Support of Flexible Mobile Service Provision”, IEEE, 2003.

[26]. S. Karlich and Siemens, “A Self-Adaptive Service Provisioning Framework for 3G/4G Mobile Applications”, IEEE Wireless Communications, October 2004.

[27]. I. Ljubi, V. Podobnik and G. Jezic, “Cooperative Mobile Agents for Automation of Service Provisioning: A Telecom Innovation”, IEEE, 2007.

[28]. M. Ganna and E. Horlait, “On Using Policies for Managing Service Provisioning in Agent-Based Heterogenous Environments for Mobile Users”, Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY’05).

[29]. S. Panagiotakis, “Dynamic Context Aware Service Provision in Beyond 3G Mobile Networks”.

[30]. A. Corradi, R. Montanari and A. Toninelli, “Dynamic Configuration of Semantic-based Service Provisioning to Portable Devices”, Proceedings of the 2005 Symposium on Applications and the Internet (SAINT’06).

[31]. M. Chen and J. Hu, “The Study of Adaptive Service Provision Platform in Future Communication System”, Proceedings ofISCIT2005.

[32]. Y. Zhang, S. Zhang, S. Han, and H. Tong, “User-Centric Service Provision Model for Adaptive Ubiquitous Computing Applications”, IEEE International Conference on Systems, Man, and Cybernetics, 2006.

[33]. G. Dreo Rodosek and L. Lewis, “A User-Centric Approach to Automated Service Provisioning”, IEEE, 2001.

[34]. E. Babulak, “User’s Perception of Quality of Service Provision”, IEEE, 2003.

[35]. S. Tursunova and S. Trong, “Enhanced Cognitive Resource Management for QoS-guaranteed Service Provisioning in Home/Office Network”, IEEE, 2009.

[36]. A. Corradi, R. Montanari and D. Tibaldi, “Context-based Access Control for Ubiquitous

Service Provisioning", IEEE, 2004.
[37]. Antonio Corradi, Rebecca Montanari, Daniela Tibaldi, Alessandra Toninelli, "A Context-centric Security Middleware for Service Provisioning in Pervasive Computing", Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'05), 2005.
[38]. O. Riva and S. Toivonen, "A Hybrid Model of Context-aware Service Provisioning Implemented on Smart Phones", IEEE, 2006.
[39]. G. Chen, C. Ping and Z. Yang, "Coordinated Service Provision in Peer-to-Peer Environments", IEEE Transaction on Parallel and Distributed System, Vol. 19, No. 4, April 2008.
[40]. M. Lesk, M.R. Stytz and R.L. Trope, "Providing Web Service Security in a Federated Environment", IEEE Computer Security, 2007.
[41]. T. Nurmela, "Evaluation Framework for Service Level Management in Federated Service Management Context".
[42]. F.M. Cuenca-Acuna and Thu D. Nguyen, "Self-Managing Federated Services", Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS'04), 2004.
[43]. Jinglong Zhou, Martin Jacobsson, Ertan Onur and Ignas Niemegeers, "A Novel Link Quality Assessment Method for Mobile Multi-Rate Multi-Hop Wireless Networks", IEEE, 2009.
[44]. "Madwifi driver." [Online]. Available: http://www.madwifi.org.
[45]. 3Com OfficeConnect Wireless 108Mbps 11g XJACK PC Card, http://www.3com.com/prod/en_EU_EMEA/detail.jsp?tab=features\&sku=3CRXJK10075, Product Specification, Accessed in March 2008.
[46]. Ethertap, http://vtun.sourceforge.net/tun/.
[47]. A. Tonnesen, T. Lopatic, H. Gredler, B. Petrovitsch, A. Kaplan, and S.-O. Tcke, "Olsrd: An adhoc wireless mesh routing deamon," 2008. [Online]. Available: http://www.olsr.org.
[48]. T. Clausen and P. Jacquet, "RFC3626: Optimized Link State Routing Protocol (OLSR)," Internet RFCs, 2003.
[49]. IST 6FP MAGNET http://www.telecom.ece.ntua.gr/magnet/
[50]. Personal Network Pilot 2004 http://www.freeband.nl/project.cfm?id = 530&language = en
[51]. M. Beasley, et al., "Establishing Co-operation in Federated Systems", ICL Technical Journal, November 1994.
[52]. B. Liu, "Characterizing the Dynamics of Dutch Home Network ", Internship report, 2010.

Appendix A

List of Abbreviations

ATM	Asynchronous Transfer Mode
FA	Fednet Agent
FM	Fednet Manager
FSP	Federated Service Provisioning
GPS	Global Positioning System
GW	Gateway
IP	Internet Protocol
LQA	Link Quality Assessment
MANET	Mobile ad-hoc Network
NGN	Next Generation Network
OLSRD	Optimized Link State Routing Daemon
P2P	Peer to Peer
PAN	Personal Area Network
PDA	Personal Digital Assistant
PN	Personal Network
PNNT	Personal Node Neighbor Table
QoS	Quality of Service
RFID	Radio-frequency identification
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
UWB	Ultra-wideband
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

Appendix B

List of Figures

Figure 1- 1 An Example of WPAN 10
Figure 1- 2 An Example of Personal Network 10
Figure 1- 3 An Example of Fednet 11
Figure 1- 4 Basic Proxy-based Architecture of Fednet 11
Figure 1- 5 Network Overlay between PNs 12
Figure 1- 6 Service Proxies between PNs 12
Figure 1- 7 Hybrid Approach for Service Provisioning in Fednet 14
Figure 1- 8 Policy-based Service Provisioning Architecture 15
Figure 2- 1 Fednet for Vehicle Network 22
Figure 2- 2 Fednet for Sharing Personal Resources 22
Figure 2- 3 Overlay Approach for Service Provisioning in Fednet 24
Figure 2- 4 Hybrid Approach for Service Provisioning in Fednet 24
Figure 3- 1 Node Architecture 30
Figure 3- 2 Framework Architecture 31
Figure 3- 3 Mobile Service Provisioning Framework 33
Figure 3- 4 User-centric Service Provision Model 35
Figure 3- 5 UbiCOSM Middleware Facilities 36
Figure 3- 6 Federated Service Provisioning System Architecture 37
Figure 3- 7 Taxonomy of Our Service Provisioning Survey 38
Figure 4- 1 New Service Provisioning Architecture for Fednet 43
Figure 4- 2 One server is available 45
Figure 4- 3 Two servers are available 45
Figure 5- 1 Decision Making Algorithm for General Scenario 57
Figure 5- 2 Decision Making Algorithm for Unfriendly Environment 59
Figure 5- 3 Decision Making Algorithm for Confidential Application 60
Figure 5- 4 Decision Making Algorithm for Short-range Scenario 60
Figure 5- 5 Scenario Setup 61
Figure 5- 6 Final Decision for Bob 62
Figure 5- 7 Overlay Approach for Service Provisioning 62
Figure 5- 8 Final Decision for Bob 63
Figure 5- 9 Final Decision for Kate 63
Figure 5- 10 Proxy-based Approach for Service Provisioning 63
Figure 5- 11 Final Decision for Bob 64
Figure 5- 12 Final Decision for Kate 64
Figure 5- 13 Hybrid Approach for Service Provisioning 64
Figure 6- 1 Prototype of Intra-Cluster Communication 70
Figure 6- 2 Fednet Packet Format 72
Figure 6- 3 The Implementation Architecture of the Prototype 74

Figure 7- 1 Simulation Result of Unfriendly Scenario……83
Figure 7- 2 Simulation Result of Friendly Scenario 85
Figure 7- 3 Simulation Result of Location Parameter in Friendly Scenario.....86
Figure 7- 4 Test Bed Architecture 87
Figure 7- 5 Overlay Scenario.....88
Figure 7- 6 Proxy-based Scenario.....89
Figure 7- 7 Throughput of Using Overlay Approach.....90
Figure 7- 8 Throughput of Using Proxy-based Approach.....90
Figure 7- 9 Comparison of Overlay and Proxy-based Approach.....91

Appendix C

List of Tables

Table 3- 1 Summary of the Service Provisioning Mechanisms.....39
Table 4- 1 Values of the Parameters.....50

Appendix D

Some Implementation Codes

Part 1

```
/* ****************************************************************** */
/*    Initialize a socket and an interface for Fednet application */
/* ****************************************************************** */

/** Initialize the Fednet socket for application
  */
int
fd_init(__u32 pnid, __u32 nodeid, int bindsocket)
{
        struct sockaddr_in6 fd_addr;
        int loopback, mcastTTL;
        int res;

        fd_fdid=1;
        fd_pnid=get_pnid();
        fd_nodeid=get_nodeid();
        fd_next_packet= 0;
        servicetable = NULL;

        fd_fd = socket(PF_INET6, SOCK_DGRAM, 0);
        if (fd_fd < 0) {
        perror("socket");
        return -1;
        }

        if (bindsocket) {
        memset(&fd_addr, 0, sizeof(fd_addr));
        fd_addr.sin6_family        = AF_INET6;
        fd_addr.sin6_port          = htons(PN_PORT_FD);
        res = bind(fd_fd, (struct sockaddr *)&fd_addr, sizeof(fd_addr));
        if (res < 0) {
                perror("bind");
                return -1;
        }
        }

        loopback = 0;
```

```
    res = setsockopt(fd_fd, IPPROTO_IPV6, IPV6_MULTICAST_LOOP, &loopback,
                        sizeof(loopback));
    if (res < 0) {
        perror("setsockopt");
        return -1;
    }

    mcastTTL = 1;
    res = setsockopt(fd_fd, IPPROTO_IPV6, IPV6_MULTICAST_HOPS,  &mcastTTL,
                        sizeof(mcastTTL));
    if (res < 0) {
        perror("setsockopt");
        return -1;
    }

    LOG(LOG_ERROR, "Fednet socket initialized\n");

    return fd_fd;
}


/** Initialize the Fednet socket for an interface
*/
int
fd_add_interface(int ifindex)
{
    struct ipv6_mreq mreq;
    int res;
    memset(&mreq, 0, sizeof(mreq));
    inet_pton(AF_INET6, PN_MCAST_ADDR, &mreq.ipv6mr_multiaddr);

    mreq.ipv6mr_interface = ifindex;

    res = setsockopt(fd_fd, IPPROTO_IPV6, IPV6_ADD_MEMBERSHIP, &mreq,
            sizeof(mreq));
    if (res < 0) {
        perror("setsockopt");
    return -1;
    }
    return 0;
}
```

Part 2

```
/* ************************************************** */
/*    Send and receive packet through the Fednet socket            */
/* ************************************************** */

/**    Broadcast of the Fednet message header
 */
int
fd_send_msghdr(int if_idx, struct pn_fd_msghdr *msg)
{
      int       l;
      struct sockaddr_in6 toall;

      toall.sin6_family       = AF_INET6;
      toall.sin6_port         = htons(PN_PORT_FD);
      toall.sin6_flowinfo     = 0;
      inet_pton(AF_INET6, PN_MCAST_ADDR, &toall.sin6_addr);
      toall.sin6_scope_id     = if_idx;

      msg->fd_pnid=fd_pnid;
      msg->fd_nodeid=fd_nodeid;

      l = sendto(fd_fd, (void *)msg, sizeof(*msg), 0, (struct sockaddr *)&toall,
                  sizeof(toall));
      if (l < 0) {
      perror("sendto");
      return l;
      }

      return 0;
}


/** Read a Fednet packet on the Fednet listener file descriptor and verify it.
 *
 */
int
fd_read_packet(struct sockaddr_in6 *ofrom6, struct pn_fd_msghdr *omsg)
{
      char      buf[1600];
      int       l;
      struct sockaddr_storage from;
      socklen_t fromlen;
```

```
	fromlen = sizeof(from);
	l = recvfrom(fd_fd, buf, sizeof(buf), 0,
		(struct sockaddr *)&from, &fromlen);

	if (l < 0) {
	perror("recvfrom");
	return -1;
	}

	LOG(LOG_ND, "Received FD packet\n");

	if (fromlen != sizeof(struct sockaddr_in6) || from.ss_family != AF_INET6) {
	LOG(LOG_ND, " FD packet not an IPv6 packet\n");
	return 0;
	}

	/*if (l != sizeof(struct pn_fd_msghdr)) {
	LOG(LOG_ND, " FD packet has wrong size!\n");
	return 0;
	}*/

	memcpy(ofrom6, &from, sizeof(struct sockaddr_in6));
	memcpy(omsg, buf, sizeof(struct pn_fd_msghdr));


	/* This is actually not needed since we can avoid sending our own FD
	   messages to our selves. */
	if ((omsg->fd_pnid == fd_pnid) && (omsg->fd_nodeid == fd_nodeid)) {
	LOG(LOG_ND, " My own hello message. Drop!\n");
	return 0;
	}

	IFLOG(LOG_ND) {
	char	buf2[INET6_ADDRSTRLEN];
	LOG(LOG_ND, " Received FD packet from: pnid=%i nodeid=%i from=%s on
if=%s(%i)\n",
		omsg->fd_pnid, omsg->fd_nodeid,
		inet_ntop(AF_INET6, &ofrom6->sin6_addr, buf2, sizeof(buf2)),
		if_indextoname(ofrom6->sin6_scope_id, buf), ofrom6->sin6_scope_id);
	}

	return 1;
}
```

```
/** Handle a received Fednet message.
*/
int
if_handle_fd_packet(void)
{
	struct interface *ifp;
	struct timeval now;
	struct sockaddr_in6 from6;
	struct pn_fd_msghdr fdmsg;
	int res;

	if (gettimeofday(&now, NULL) < 0) {
	perror("gettimeofday");
	exit(1);
	return -1;
	}

	res = fd_read_packet(&from6, &fdmsg);
	if (res < 0) {
	LOG(LOG_ERROR, "fd_read_packet() failed receiving packet\n");
	exit(1);
	return -1;
	}
	if (res == 0)
	return 0;

	ifp = if_get_interface_from_index(from6.sin6_scope_id);
	if (!ifp) {
	LOG(LOG_ERROR, "IF(nd): No such interface (%d)!!!\n",
		from6.sin6_scope_id);
	return -1;
	}

	return ifp->if_recv_fd_packet(ifp, &from6, &fdmsg, now);
}
```

Part 3

```
/* ***************************** */
/*    Fednet Manager Functions        */
/* ***************************** */

/** Fednet manager (FM) is sending the Fednet advertisement
 *    including fednet ID, ip address of the FM
 */
int
fm_send_advertizement(struct interface *ifp, struct pn_fd_msghdr *msg)
{
      msg->fd_fdid=fd_fdid;
      strcpy(msg->ipfm,"3000::9");

      int t;
      t=fd_send_msghdr(ifp->if_index, msg);
      printf("FM is sending the fednet advertisement: Fednet ID=%d, IP of FM=%s\n",
                  msg->fd_fdid,msg->ipfm);

      return 0;
}


/** Fednet manager (FM) sends a membership credential message to the FA who
 *    wants to participate, including Fednet ID
 */
int
fm_membership_credential(struct interface *ifp)
{
      struct wlan_interface *wifp = (struct wlan_interface*)ifp;
      struct pn_fd_msghdr fdmsg;

      fdmsg.fd_type = PN_FD_TYPE_ACCEPT;
      fdmsg.fd_comm_method = PN_FD_COMM_METHOD_UDP;
      memcpy(fdmsg.fd_MAC, &wifp->wif_hwaddr, sizeof(fdmsg.fd_MAC));

      int t;
      t=fd_send_msghdr(ifp->if_index, &fdmsg);
      printf("FM is giving the membership credential(private key of Fednet) to the
PNs:AEM568\n");

      return 0;
}
```

```
/**An example of a Fednet service
*/
struct pn_fd_msghdr *
fednet_service_one(struct interface *ifp)
{
	struct pn_fd_msghdr *fdp;
	fdp = malloc(sizeof(struct pn_fd_msghdr));

	if (!fdp) {
	perror("malloc");
	return NULL;
	}
	memset(fdp, 0, sizeof(struct pn_fd_msghdr));

	fdp->fd_fdid=fd_fdid;
	fdp->fd_pnid=fd_pnid;
	fdp->fd_nodeid=fd_nodeid=9;

	fdp->fd_scenario=PN_FD_SCENARIO_GENERAL;

	strcpy(fdp->ipfa,"3000::6");
	strcpy(fdp->service_name, "printing1");

	strcpy(fdp->device_cpblty,"special");
	strcpy(fdp->service_type, "strong");
	strcpy(fdp->privacy, "low");
	strcpy(fdp->location, "overlay");
	strcpy(fdp->battery, "overlay");
	strcpy(fdp->reputation, "overlay");
	strcpy(fdp->link_quality, "overlay");
	strcpy(fdp->security, "overlay");

	return fdp;
}
```

```
/** FM creates the service directory for all the services he receives,
  *    including pnid, nodeid, IP address of the FA and service name
  */
struct pn_fd_msghdr*
fm_service_directory(struct interface *ifp, struct sockaddr_in6 *from6,
                        struct pn_fd_msghdr *fdp)
{
      struct pn_fd_msghdr *newfdp = NULL;

      struct pn_fd_msghdr *checkfdp = NULL;

      newfdp = malloc(sizeof(struct pn_fd_msghdr));
      if (!newfdp)
      {
            perror("malloc");
            return NULL;
      }

      memset(newfdp, 0, sizeof(struct pn_fd_msghdr));
      memcpy(newfdp,fdp,sizeof(struct pn_fd_msghdr));

      newfdp->fd_packet = fd_next_packet++;
      newfdp->fd_valid = 1;
      newfdp->next_fm_service= servicetable;
      servicetable = newfdp;

      printf("The following services are registered\n");

      checkfdp = servicetable;
      while(checkfdp)
      {
            fdp=fednet_service_one(ifp);
            printf("pnid=%d,    nodeid=%d, ip of FA=%s, service=%s\n",
                  fdp->fd_pnid, fdp->fd_nodeid, fdp->ipfa, fdp->service_name);
            printf("pnid=%d,    nodeid=%d, ip of FA=%s, service=%s\n",
                  checkfdp->fd_pnid,          checkfdp->fd_nodeid,          checkfdp->ipfa,
checkfdp->service_name);

            if (!checkfdp->fd_valid)
            printf("INVALID");
            checkfdp=checkfdp->next_fm_service;
      }

      return servicetable;}
```

```
/** FM lookups the service that the FA is requestingin the service directory
 */
struct pn_fd_msghdr *
fm_service_lookup(struct interface *ifp, struct sockaddr_in6 *from6,
                        struct pn_fd_msghdr *omsg)
{
      struct pn_fd_msghdr *fdp = servicetable;
      while (fdp && !(strcmp(omsg->service_name,fdp->service_name)==0))
      fdp = fdp->next_fm_service;

      if(fdp && (strcmp(omsg->service_name,fdp->service_name)==0)){
      printf("The service is found in pnid=%d,    nodeid=%d, ip of FA=%s, service=%s\n",
                  fdp->fd_pnid, fdp->fd_nodeid, fdp->ipfa, fdp->service_name);
      }

      else
      {printf("The service is not found\n");}

      return fdp;
}


/**    Service lookup response from FM with the service list
 */
int
fm_service_lookup_response(struct interface *ifp, struct sockaddr_in6 *from6, struct
pn_fd_msghdr *fdp)
{
      struct wlan_interface *wifp = (struct wlan_interface*)ifp;

      fdp->fd_type = PN_FD_TYPE_SERVICE_LOOKUP_RESPONSE;
      fdp->fd_comm_method = PN_FD_COMM_METHOD_UDP;
      memcpy(fdp->fd_MAC, &wifp->wif_hwaddr, sizeof(fdp->fd_MAC));

      int res;
      res=fd_send_msghdr(ifp->if_index,fdp);

      return 0;
}
```

Part 4

```
/* ***************************** */
/*    Fednet Agent Functions          */
/* ***************************** */

/** FA creates the participation profile*
  */
int
fa_participtionprofile(struct interface *ifp)
{
      printf("After receiving the advertisement from FM, sending participation profile to FM\n");

      struct wlan_interface *wifp = (struct wlan_interface*)ifp;
      struct pn_fd_msghdr fdmsg;

      memset(&fdmsg, 0, sizeof(fdmsg));

      strcpy(fdmsg.ipfa, "ip of FA");

      int t;
      t=fd_send_msghdr(ifp->if_index, &fdmsg);

      fdmsg.fd_type = PN_FD_TYPE_JOIN;
      fdmsg.fd_comm_method = PN_FD_COMM_METHOD_UDP;
      memcpy(fdmsg.fd_MAC, &wifp->wif_hwaddr, sizeof(fdmsg.fd_MAC));

      return 0;
}


/** FA sends a service lookup request, including the service name.
  */
int
fa_service_lookup_request(struct interface *ifp)
{
      struct wlan_interface *wifp = (struct wlan_interface*)ifp;
      struct pn_fd_msghdr fdmsg;

      strcpy(fdmsg.service_name,"picture3");

      fdmsg.fd_type = PN_FD_TYPE_SERVICE_LOOKUP_REQUEST ;
      fdmsg.fd_comm_method = PN_FD_COMM_METHOD_UDP;
      memcpy(fdmsg.fd_MAC, &wifp->wif_hwaddr, sizeof(fdmsg.fd_MAC));
```

```
	int res;
	res=fd_send_msghdr(ifp->if_index, &fdmsg);

	printf("I want to lookup this service: %s in FM servicetable\n", fdmsg.service_name);

	return 0;

}
```

/ FA sends service request to the the server, after it receives the service lookup response from the FM**

***/**

```
int
fa_service_request(struct interface *ifp, struct sockaddr_in6 *from6, struct pn_fd_msghdr *fdp)
{
	printf("I received the response:ipfa=%s,service_name=%s\n",
			fdp->ipfa, fdp->service_name);

	printf("Now I request the service=picture3\n");

	struct wlan_interface *wifp = (struct wlan_interface*)ifp;
	struct pn_fd_msghdr fdmsg;

	fdmsg.fd_type = PN_FD_TYPE_SERVICE_REQUEST;
	fdmsg.fd_comm_method = PN_FD_COMM_METHOD_UDP;
	memcpy(fdmsg.fd_MAC, &wifp->wif_hwaddr, sizeof(fdmsg.fd_MAC));

	int t;
	t=fd_send_msghdr(ifp->if_index, &fdmsg);

	return 0;
}
```

```
/**
 * Context Interpreter in FA
 */
int
fa_context_interpreter(struct interface *ifp)
{
	struct pn_fd_msghdr *fdmsg;
	fdmsg=fednet_service_one(ifp);

	/* interprect context into binary value (0 or 1) */
	int cp;
	if(strcmp(fdmsg->device_cpblty,"strong")==0){
	cp=1;
	}
	else
	{cp=0;}

	int st;
	if(strcmp(fdmsg->service_type,"common")==0){
	st=1;
	}
	else
	{st=0;}

	int pri;
	if(strcmp(fdmsg->privacy,"low")==0){
	pri=1;
	}
	else
	{pri=0;}

	int btry;
	if(strcmp(fdmsg->battery,"overlay")==0){
	btry=1;
	}
	else
	{btry=0;}

	int lc;
	if(strcmp(fdmsg->location,"overlay")==0){
	lc=1;
	}
	else
	{lc=0;}
```

```
	int rep;
	if(strcmp(fdmsg->reputation,"high")==0){
	rep=1;
	}
	else
	{rep=0;}

	int lq;
	if(strcmp(fdmsg->link_quality,"overlay")==0){
	lq=1;
	}
	else
	{lq=0;}

	int sec;
	if(strcmp(fdmsg->security,"overlay")==0){
	sec=1;
	}
	else
	{sec=0;}

	return 0;
}



/**
 * Policy engine function in FA, before the service sends request
*/
int
fa_policy_engine(struct interface *ifp)
{
	int fa_c_i;
	fa_c_i=fa_context_interpreter(ifp);

	struct pn_fd_msghdr *fdp;
	fdp=fednet_service_one(ifp);
	//fdp->fd_scenario;

	int cp,st,pri;
	int btry,lc, rep;
	int lq,sec;
```

```
/*The algorithms of policy engine*
  **********************************/

switch(fdp->fd_scenario) {

/* The general scenario algorithm
  * including device capability, service type, privacy
  * battery, location, reputation*/
case PN_FD_SCENARIO_GENERAL:
if(cp&&st==1){
   printf("both are ok,need to go down\n");

     if(pri==1){
        printf("need to go down\n");

        if(lc&&btry==1){
           printf("still go down\n");

           if(rep==1){
              printf("overlay\n");
           }
           else
           {printf("proxy 4\n");}
        }
        else
        {printf("proxy 3\n");}
     }
     else
     {printf("proxy 2\n");}
}
else
{printf("proxy 1\n");}

break;

/* The unfriendly scenario algorithm
  * including device capability, service type, privacy
  * repuation, location, battery*/
case PN_FD_SCENARIO_UNFRIENDLY:
if(cp&&st==1){
   printf("both are ok,need to go down\n");

     if(pri&&rep==1){
        printf("need to go down\n");
```

```
        if(lc&&btry==1){
           printf("overlay\n");
         }
           else
           {printf("proxy 3\n");}
        }
        else
        {printf("proxy 2\n");}
     }
else
{printf("proxy 1\n");}

break;

/* The security scenario algorithm
  * including device capability, service type, privacy, security */
case PN_FD_SCENARIO_SECURITY:
if(cp&&st==1){
   printf("both are ok,need to go down\n");

     if(pri&&sec==1){
        printf("overlay\n");
      }
        else
        {printf("proxy 2\n");}
  }
else
{printf("proxy 1\n");}

break;

/* The short range scenario algorithm
  * including device capability, service type, privacy, link_quality*/
case PN_FD_SCENARIO_SHORT_RANGE:
if(cp&&st==1){
   printf("both are ok,need to go down\n");

     if(pri==1){
        printf("need to go down\n");

        if(lq==1){
           printf("overlay\n");
        }
```

```
                else
                {printf("proxy 3\n");}
            }
            else
            {printf("proxy 2\n");}
        }
    else
    {printf("proxy 1\n");}

    break;

    default:
        printf("More algorithms will be done in future\n");
    }

    return 0;
}
```

/ FA sends service request to the the server, after it receives the service lookup response from the FM**

***/**

```
int
fa_service_response(struct interface *ifp, struct sockaddr_in6 *from6,struct pn_fd_msghdr *fdp)
{
    printf("The service is giving the response\n");

    struct wlan_interface *wifp = (struct wlan_interface*)ifp;
    struct pn_fd_msghdr fdmsg;

    fdmsg.fd_type = PN_FD_TYPE_SERVICE_RESPONSE;
    fdmsg.fd_comm_method = PN_FD_COMM_METHOD_UDP;
    memcpy(fdmsg.fd_MAC, &wifp->wif_hwaddr, sizeof(fdmsg.fd_MAC));

    int t;
    t=fd_send_msghdr(ifp->if_index, &fdmsg)
    return 0;
}
```