

**An Enriched Finite State Machine Model-based Formalism for Layer-
5 Internet Protocols Modelling – An Investigation on Protocol
Performance**

Alexandru Catalin Ionescu

University of Tampere
Department of Computer Sciences
Computer Science / Int. Technology
M.Sc. thesis
Supervisor: Eleni Berki
June 2008

Internet level-5 protocols are defined by the Internet Engineering Task Force (IETF). Some of these specifications were developed long before the need for mobile support. As a consequence they are extremely solid but not flexible enough when used in the desktop environment and fail to deliver when ported onto mobile handsets.

In this thesis I investigate the level-5 protocols in particular, in order to analyze, understand better and enhance their performance. First we take a look at Mobile Services and Mobile Networks. I study how Packet Data services are handled and how a communication protocol can affect their behaviour and performance. Then I discuss the Internet Level-5 protocols and focus on their main characteristics. Finally I develop a model for Protocol Performance Measurements. The model addresses the level-5 protocols but it can also be used for lower layers as well. Ultimately I used the model in order to analyze one of the most important use cases on IETF's agenda – Presence. This was done by conducting observations in the real, mobile environment, upon the developed model's application. I showed how this model can be used in order to measure the Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) protocol suite performance. In the thesis it is also shown how the same model can be used in order to prove the benefits of a new IETF proposal.

The theoretical concepts utilized in this thesis belong to the classical knowledge of computation science. The basic automaton, the Finite State Machine, was semantically extended and used to model the dynamic behaviour of communication protocols. Furthermore, its enhanced version, the Finite State Protocol (FSP), provides metrics that can be used as indicators for the system's dynamic/evolutionary behaviour and for the communication protocols' performance.

This thesis work has been based on ISO definitions on quality concepts, and it, in particular, creates new knowledge by associating protocols' effectiveness with design quality, and by proving that other quality attributes - such as reliability and resilience – can be formally enhanced. This can start from the very early stages of the mobile software development as preventive maintenance principles indicate.

Key words and terms: Finite State Machine (FSM), Communication Protocol, International Organization for Standardization (ISO), Quality Standards, Performance, Mobile Technology, Metris (and Measurement)

Contents

1. Introduction	1
1.1. Problems and Methods.....	3
2. Background – Mobile Services and Applications	5
3. Mobile Internet Communications.....	9
3.1. Performance in Mobile Networks	9
3.2. Resilient Level-5 Protocols in Mobile Networks	11
3.3. Reliable Level-5 Protocols in Mobile Networks	12
3.4. Effective Level-5 Protocols in Mobile Networks	12
4. Mobile Packet Access	13
4.1. Packet Data Mobile Communications	13
4.2. Wideband Code Division Multiple Access Packet Data (WCDMA)	15
4.2.1. Transport Channels for Packet Data	15
4.3. Selection of the Channel Type.....	17
5. Layered Networks Architecture	18
5.1. Data Networks and Layered Architectures	18
5.2. OSI Standard Architecture and Protocols	20
5.3. OSI Protocols	21
5.4. Internet Reference Model	25
5.5. Internet Reference Model in Mobile Networks	26
6. Protocol Modelling.....	28
6.1. Models for Protocol Specification	29
6.2. High Level Programming languages	30
6.3. Finite State Machines.....	31
6.4. Petri Nets.....	33
7. Modelling Protocol for Performance Measurement.....	38
7.1. Measuring Effectiveness	38
7.2. A User – Client – Server Model	41
7.3. Modelling Protocols with Finite State Machines.....	46
7.4. Modelling Use Case with Finite State Machines.....	46
7.5. Modelling Protocols with Petri Nets and other Related Work	49
7.5.1. Coloured Petri Nets	49
7.6. A Graphical Representation for Finite State Protocols.....	51
8. Internet Presence – A Case Study	53
8.1. A Presence Data Model	53
8.2. Presence Service Architecture	54
8.3. Standard Definitions	55

8.4. Presence Deployment - Example	56
8.5. Presence Traffic Management	57
8.6. A FSP Time Line Model for Presence	57
8.7. The SIMPLE Model for Presence	58
8.8. A Presence Use Case	61
8.9. Improving the SIMPLE protocol	63
9. Conclusion and Future Development Plans	65
9.1. Future Work	66
Appendix A - IP Multimedia Presence Service	73

List of Figures

Figure 1 - IETF Multimedia Architecture	8
Figure 2 - Protocol Performance	10
Figure 3 - ETSI Model for Packet Service Session.....	14
Figure 4 - Layered Model and Peer Protocols	19
Figure 5 - Relay Open System	20
Figure 6 - A 3-phased communication at a layer	22
Figure 7 - (N+1)-Entities and N-Services	23
Figure 8 - Data Units according to OSI Architecture	24
Figure 9 - Basic OSI primitives	25
Figure 10 - IP Suite Stack Host to Host communication over Internet.....	26
Figure 11 - Internet Reference Model in Mobile Networks.....	27
Figure 12 – Architectural Layer	30
Figure 13 - Client Server Protocol modelled with Finite State Machines	31
Figure 14 - Marked Petri Net	34
Figure 15 - Petri Net at work.....	35
Figure 16 - Modelling communication Protocols with Petri Nets	36
Figure 17 - Protocol Measurement System.....	39
Figure 18 - Handling an external event in the measuring environment.....	40
Figure 19 - A User-Client-Server Model	41
Figure 20 - Connect – Subscribe – Notify – Disconnect Paradigm	42
Figure 21 - Complete Internet Service Usage Paradigm.....	43
Figure 22 - System State View.....	44
Figure 23 - Use Case Execution.....	45
Figure 24 - Simple example of Coloured Petri Nets	50
Figure 25 - Time Line Graphical Model for FSPs	52
Figure 26 - A Data Model for Presence	54
Figure 27 - Presence Service System Architecture	55
Figure 28 - Presence Service Deployment	56
Figure 29 - Abstract Time Line Model for Presence	58
Figure 30 – SIMPLE Connection to Presence Service	59
Figure 31 – SIMPLE Presence Publication.....	59
Figure 32 – SIMPLE Presence Subscription.....	60
Figure 33 – IMS Registration Procedure	73
Figure 34 – Notifying Presence Information Updates in IMS	76
Figure 35 – Publishing Presence Information in IMS.....	81

1. Introduction

Over the last few years a steady stream of innovations has been brought into the mobile communications market. Services that were known to be working only in the fixed Internet environment are emerging also in mobile networks. We see Service Providers deploying the experience available on desktop computers connected to Internet over wired networks into mobile devices connected over the radio networks. It is however a problem to address when dealing with the *wired to mobile* migration. This constitutes the main difference between desktop devices and mobile devices. A mobile device is obviously weaker than a desktop computer when it comes to processing power regarding for instance: Input / Output resources, battery life and the list can continue. Another problem to address is the connectivity. In the context of the fixed Internet environment the applications are running on powerful computers connected over wired networks. The amount of data that is sent or received is not considered a problem anymore. However, in mobile networks the radio resources are at premium. The network traffic – amount of data and number of messages – needs careful consideration before a service is deployed. From this point of view the protocols used in the fixed Internet are not always suitable for mobile use.

Second generation (2G) telecommunication systems brought voice into a mobile environment. However, these networks are not successful in handling data communications. Their capabilities are somewhat limited by the low bit rates. Services such as high quality image transfer or video transmissions are not supported. Third Generation (3G) networks are emerging at the moment. The bit rates offered in this new environment are high and a variety of new services can be deployed.

Higher bit rates open new opportunities for new services in mobile environment. In particular, the services that are currently available in the Internet environment are increasingly becoming mobile. This calls for effective handling of TCP / UDP / IP traffic. The development of new standards in the telecommunication needs to take these requirements into consideration. One needs to be sure that protocols below layer-4 in the OSI and Internet reference models will be handled properly. However, the real service implementation will be based on application layer protocols – also known as Internet level-5 protocols.

The Internet Engineering Task Force (IETF) is the standardization body developing the majority of the protocols used by Internet applications. Their unwritten motto is “we believe in code that works”. In consequence the services built on top of the specifications released by IETF proved to be extremely solid from the technical point of view. However, the initial specifications of IETF have been released long time before the need of using the same protocols in the radio networks. These specifications are not necessarily suitable for mobile environments and modifying them proves to be a tedious

work. Consequently mobile service providers find it difficult to launch services based on specifications that are not suitable for their use. The need for IETF specifications tailored for mobile use is obvious.

Open Mobile Alliance (OMA) is a standardization organization that was formed by the major players in the mobile services market. Its mission is to facilitate global user adoption of mobile data services. One of the problems addressed by this standardization body is the connectivity in mobile networks. In fact OMA takes two approaches to solve this problem. First, new protocols are defined in order to address the known limitations of radio networks. Second, well known protocols developed by IETF are tailored for mobile use.

Defining a new protocol that addresses the known limitations of radio technologies might be easy. However, OMA's mission becomes difficult when already existing protocols need to be adjusted to mobile environments. In most of the cases there is more than one technology for solving a certain use case. In such cases the candidate technologies need to be compared against a set of criteria. Thus, a model to measure the performance of a given technology is needed. In case of mobile networks we are interested in measuring how a technology manages the radio resources. The amount of data and the amount of messages sent over the network is vital for the success of a service deployment.

Applications developed based on OMA specifications are deployed in live environments. At this stage the business takes priority over technology. Customers expect the service to work flawlessly. Errors can heavily impact the business of the service operator hence network planning is crucial. Based on a business case the network planner needs to estimate the generated network traffic. These metrics can be used in order to deploy the right amount of resources. Therefore, a model for estimating the network traffic generated by communication protocols is a must.

Some work in this area has already been done in IETF – [SAINTANDRE, 2007]. However, this work does not address the problem from a general point of view. Individual protocols have been analyzed on specific use cases without any theoretical consideration. A common theoretical model is needed in order to make a comparison between two technologies. This thesis develops such a theoretical model based on a concrete case – Mobile Presence Service for Mobile Operator Use. At the first stage the model allows us to find problematic areas for the communication protocols defined by the IETF's SIMPLE working group. We also define solutions to solve these problems. These solutions can then be considered by OMA or IETF in order to improve their specifications. The thesis does not compare any technologies. In the future, however, the same model presented in this thesis can be used in order to analyze two candidate technologies for the same use case.

1.1. Problems and Methods

The end-goal of this thesis is to estimate the generated network traffic while using a specific communication protocol. This work, being originally a constructive nature research project, is based on a real case study from real life telecommunications company. In proceeding towards this goal we need to answer the following research questions.

Question 1: How could one formally model the communication protocols according to a specific use case?

In this thesis I am searching to find a way for estimating the value of the traffic generated by communication protocols. Before we are able to asses the performance of such a communication protocol we need to model its behaviour in certain situations. We know that a protocol is just a set of rules that describe and govern the communication between two computing end points inside a system. These rules define the synchronization, semantics and syntax of the communication. It does not define at all how the actual end points use the protocol itself. Moreover, a protocol cannot define the behaviour of the entities involved in the communication - this behaviour depends on the context / environment where the communication entities operate. In practice, this behaviour is affected by various different internal or external events.

Based on the protocol description files I define a model to describe the endpoint behaviour. This model should cover the real life needs. In order to achieve that there is, first, the need to model how the particular system is used. There is a need to describe a way of using part of the system's functionality – define the *use case*.

Question 2: How to estimate the generated network traffic for a specific use case when using a certain communication protocol?

After we define the use case for which we measure the performance of the communication protocol we need to do the actual measurement. This is done according to a formula that allows us to calculate / estimate the network traffic.

Question 3: How to improve a protocol in order to decrease the generated network traffic?

Based on given measurements one could decide on improvements. One option is to improve the actual protocol in order to decrease the generated traffic. Another option is to find a new way of using the same protocol while still fulfilling the use case. The third option is to look for another technology that when used together with the protocol in question decreases the value of the generated traffic. In this thesis I deal with and analyze the second and third option. Thus, I suggest ways to improve the way we use the protocol and I show how the traffic can be decreased while applying compression.

Chapter 2 –continuous improvements of the radio networks call for improving the communication protocols being used by mobile services and applications. This section is the background for my thesis.

Chapter 3 – the increased performance offered by radio networks makes us demand more and more complete connectivity. In this section I discuss the performance expectations and challenges in future mobile networks.

Chapter 4 – new radio technologies offer better packet data access. In this section I describe the mobile environment and in particular the way that data communication is handled.

Chapter 5 – two different models are used by experts involved in communication protocols design: (1) Open System Interconnect (OSI) reference model and (2) Internet reference model. In this section I focus on the differences between them and justify the reason for choosing the Internet reference model as the base for my studies.

Chapter 6 – Informal models have been used in communication protocols development. However, formal models are more and more needed due to the ever increasing complexity of communication methods. In this section I discuss various different formal models that have been used by experts for communication protocols modelling.

Chapter 7 – one aspect to consider during communication protocol design is performance. In this section describe a new formal model for performance measurements based on Finite State Machines model.

Chapter 8 – in this section I discuss a case study on a concrete example – Internet Presence

Chapter 9 - Conclusions

2. Background – Mobile Services and Applications

One of the most important features of the new type of mobile networks are currently deployed is the high user bit rate. For example, in the Universal Mobile Telecommunications System (UMTS) the connections offer up to 384 kbps on Circuit-Switch and up to 2mbps on Packet-Switched. In this case it is natural that services, which could not be available in early mobile environments due to low data rates, are now being considered. Video telephony, voice and quick data download are only a few of those services. It is yet to be seen what the “killer” application is. Most likely it will be an application that offers almost instant access to information based on the user context and content. One good example is offering access to information based on the location of the user. Another example is the so called *Presence* considered when a decision on how to communicate is based on the information about the users of the system (The Presence case will analytically be exposed later on in chapter 8).

Compared to old-type mobile networks, such as GSM, the new technologies offer a very important feature: The clients involved in communication are able to negotiate the properties of the bearer – one client has the ability to find out the capabilities of the communication peer. In practice, this means that depending on the application needs the chosen bearer offers a minimum of quality – Quality of Service (QoS) – in order for the application to run properly. This really means that the mobile environment cannot be optimized for a single set of applications. It is mandatory to support different levels of quality of service. At the same time, this means that not all the applications will be offered the best QoS. Depending on the use case, some are offered the best quality available but some need to cope with fewer resources. However, no matter how many resources the network is able to give to an application; one could be sure that in some cases this amount is not enough. This leads us to the subject of this thesis. There is a need to provide a model that allows a developer to first analyze and eventually optimize the application protocol.

Generally speaking, applications and services are divided into various different groups. The criteria vary but the main objective is to satisfy the quality expectations of the user of the application. For example, the UMTS standardization has defined four classes. This classification has been done according to the quality of service needed by the applications and services considered during the UMTS standardization work. In fact the division takes into consideration how sensitive the applications are to delays. In Table 1 one can see QoS classes defined by UTMS [3GPP23907, 1999].

Table 1 - UMTS QoS Classes

Traffic Class	Conversational	Streaming	Interactive	Background
Main characteristics	<p>Conversational pattern</p> <p>Preserves the time relation between the informational elements of the stream</p>	<p>Preserves the time relation between the informational elements of the stream</p>	<p>Request - Response Pattern</p> <p>Data Integrity</p>	<p>Destination is not expecting the data within certain time limits</p> <p>Data Integrity</p>
Example	Voice call, Video Telephony	Streaming multimedia content, Internet TV	Web Browsing, Instant Messaging	Email

The *Conversational Class* is probably the best known of them all. Applications that fall into this category are those applications that the users are most familiar with – Voice also known as speech service over circuit switched. In the new Internet environments the voice service evolves towards a richer set of multimedia communication – voice over IP, video call, and so on. I am talking here by considering the real-time communications, where the traffic is nearly symmetric and the end-to-end delay is required to be low.

Streaming class is again something that we are already used to. Any user of a desktop computer has visited www.youtube.com or a similar service in order to watch video clips or listen to an internet radio service. The streaming technique is about transferring data in a steady flow that allows a receiving end-point to process and render it as a continuous flow. This helps two main use cases. The first and apparently most important for the mobile users is the ability to *consume* large multimedia content without the need to download it locally. This is needed because most of the cases downloading (locally) are not possible due to the memory limitations of mobile devices. The second aspect, that involves monetary aspects as well, is the ability of a service provider to allow the users to *consume* multimedia content with the possibility to record for future use.

The *Interactive Class* deals with those use cases where a user requests data from a service. The service is responding based on certain rules such as authentication or authorization. The most known application falling into this category is the WEB

browsing. Other applications start to emerge. One of them is Mobile Presence that will be discussed in Chapter 8.

Background Class is again something that we are familiar with. It is probably not acknowledged as much as the previous three classes but applications falling into this category are extensively used. Short Messaging Service (SMS) and Email are probably the most familiar ones.

IETF Multimedia Architecture

Current mobile applications are built on protocols defined by standardization bodies that did not consider the Internet as their main target environment. For example the GSM standardization body did not develop only the communication protocols but also the communication environment. As a consequence the related applications do not perform well in the new environment that is – the Internet. The complex signalling is not efficient on the new type of wireless links. Instead, the specifications defined by the main standardization body for Internet – IETF – are considered more and more. They became over the past twelve years the de facto standards hence the new vision called IETF Multimedia Architecture. This architecture covers several areas and can be seen in Figure 1. That means that text-based level-5 signalling protocols like the ones enumerated in the list below are used for multimedia communications:

- Session Initiation Protocol (SIP) for setting up and tearing down communication sessions [SIP, 1999]
- Session Announcement Protocols (SAP) for advertising Audio / Visual sessions being broadcasted [SAP, 2000]
- Session Description Protocol (SDP) for a text-based description of the communication sessions [SDP, 1998]
- Real-time Streaming Protocol (RTSP) for controlling remote servers [RTSP, 1998]
- Real-time Transport Protocol (RTP) for media encapsulation [RTP, 1996]

The list above only refers to a few protocols – probably the most important – defined by IETF and used for communication in the Internet.

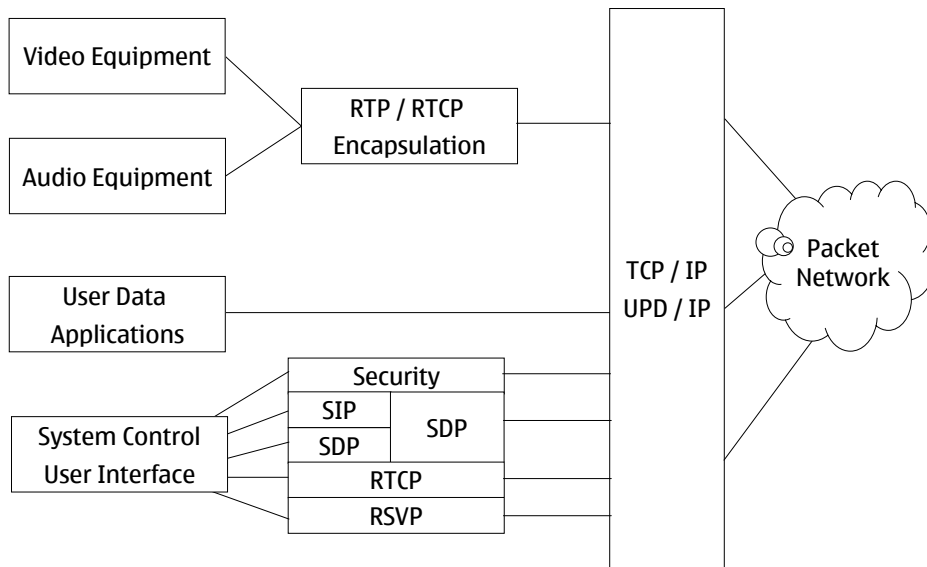


Figure 1 - IETF Multimedia Architecture

The protocols mentioned above have already proved their efficiency in mobile environments. However, they are only a few of the protocols defined by IETF. Others that have been used over time in fixed networks are gradually being introduced, emerging from the user needs. One example is the SIMPLE protocol suite defined by IETF working group with the same name. I have personally been part of IETF debates where it has been argued the fact that SIMPLE Specifications are a “good” example of a non-efficient protocol for mobile use. We discuss more about this in Chapter 8.

3. Mobile Internet Communications

Mobile telecommunications changed the way we see the world. Since the introduction of mobile services we demand complete connectivity at any point in time and no matter the place. Everything started with the familiar and valued voice transmission. However, over the years people started to use some other services as well. First it was the very well known Short Message Service (SMS). It was the beginning of sending and receiving data. FAX, Multimedia Messaging and other types of Internet Communications followed. These new types of communication are not very well known; hence they are seldom used. Initially, the reason for not taking these new data services into use was the “cost” set by the GSM network. Slowness of transmission and high monetary costs kept the users away.

For example, a single, non-compressed picture with a resolution of 800X600 pixels (that is the average size of a picture taken with a regular mobile phone camera) would take up to 3 minutes for its complete transmission in a GSM network. In addition to the time we can add the cost of the 3 minutes of usage. One can ask if there is any value in sending a picture, when compared with the amount of information that could be conveyed in three minutes of voice communication. The amount of time and high costs are just not acceptable by the mobile phone users.

3.1. Performance in Mobile Networks

Communication protocols are designed according to certain principles. *Reliability*, *effectiveness* and *resiliency* are the most important quality features we are looking for. When all these are satisfied we consider a protocol to be performing well. The problem that everybody faces is that it is not easy to analyze these features. Performance - otherwise called efficiency by Quality Standards - is also difficult to define formally, analyse and finally accept. Next, this thesis attempts a closer look at the particular meanings and the significance of the quality features of reliability, effectiveness and resiliency, in the context of communication protocol performance. Depicted initially in Figure 2 – Protocol Performance, these stakeholders’ expectations [Berki and Siakas, 2007] form the must that would guarantee the efficiency of a quality mobile service. Assuring these early in the design level, could provide enhanced service efficiency, and therefore, increase in service usability. The latter is another quality factor widely acknowledged by the international quality Standards for both process and product.

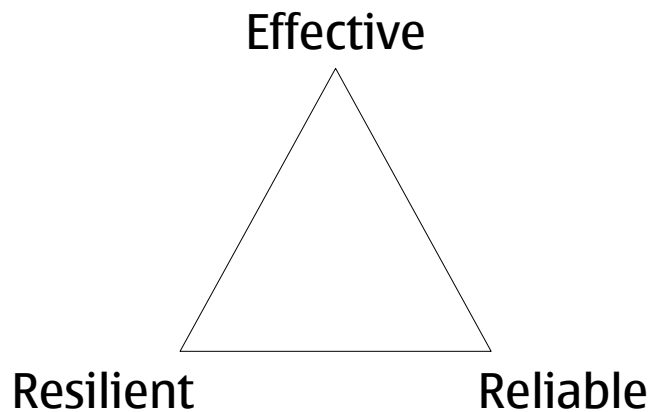


Figure 2 - Protocol Performance

Reliability

One thing that I can be sure after 10 years of working in mobile communication is that transmission media are faulty. Along the years I have personally been involved in many experiments on real-life systems that proved this. This is even more obvious in networks where the communication takes place over radio. Adding the mobility factor to the transmission equation could even make things worse. In GSM / UMTS networks the data is transmitted between mobile devices roaming a radio network and network-based servers. Assuring the *Reliability* of the data transmission is a must. *Error detection* and *correction* is the most used technique. While *error detection* in radio networks is possible the *correction* is not always an option. The actual data can easily be corrupted beyond repair hence the entities involved in the communication need the ability to request retransmission. The search for suitable software design architectures as part of software quality assurance (see e.g. Ince, 1995) seems to be a must for the redesign of transmission media.

Resilience

Resilience is, in essence, the speedy recovery from problems and the ability to recover quickly from setbacks.

Resilience addresses a form of network failure also known as *topological failure*. The communication link is cut completely; or, the quality drops below the usable

levels. In mobile networks this is a widely known problem. A common solution is to enable the entities to be able to test the communication link.

Testing the latter should be facilitated with a suitable dynamic formalism. This would improve the trust on network efficiency and this directly influences the usage of the services [Berki et al., 2007]. Hence, a communication protocol that would cater for testing could increase the ability and quality of the overall performance.

Effectiveness

Communication protocols are specified in such way that they can easily be implemented and used. This is the high level understanding on what *Effectiveness* means [Berki and Siakas, 2007]. Going deeper into the definition one can ask what it means *Easy* to implement and *Use*. That is depending on the context that the protocol is used. For example an eXtensible Markup Language (XML) based protocol is easy to implement in a desktop environment. The amount of tools makes that work easy and it also allows the protocol to be widely used. But what about utilizing the same protocol in a mobile environment? Well, things are not the same. The verbosity of XML makes it hard to be used due to the amount of data that needs to be sent. At the moment it is also a fact that XML processing tools on mobile devices are rare. That means, in practice, even if a protocol is considered effective in certain context is not necessarily effective in another context.

3.2. Resilient Level-5 Protocols in Mobile Networks

In mobile networks the communication between two computing endpoints using a Level-5 protocol goes according to the model described later on in Figure 4, chapter 5. The data sent by the mobile device goes over the various different physical media of which the first leg is the radio network. Similarly, when the device receives the data, the last leg of physical communication is again the radio wave. In a mobile context the topology of the network from the radio point of view never changes assuming that mobile device offers one connectivity solution only. A different scenario is when the mobile device is capable of connecting using different radio technologies like for example GSM / UMTS and Wireless LAN (WLAN) - Unified Mobile Access (UMA) allows a mobile handset to connect on both GSM and WLAN[UMAOVERVIEW].

In this thesis I consider the case when a mobile device is capable of handling GSM / UMTS connectivity only. The protocol itself cannot solve link loss since there is no possibility to create another link. In this case the *Resilience* of Level-5 protocols translates to the ability of the protocol to test the availability of the link. For Example, a Voice over IP application uses SIP in order to perform the so called Registration. The Client registers its “location” to a Registrar Server. The server then knows how to contact the Client when there is an incoming call. In a GSM / UMTS environment the

SIP protocol offers the possibility to both Client and Server to query the availability of each other. In case the Server notices a connection break it can reject all the incoming calls. At the same time a Client can try to re-Register.

3.3. Reliable Level-5 Protocols in Mobile Networks

In mobile networks the radio resources are the most likely to create connectivity problems. Signal losses or bad-quality signal are the most common problems that we experience when roaming in such networks.

When the quality of the signal drops errors start to occur. Low level protocols - Level-4 downwards – are capable of detecting and correcting errors. When errors cannot be corrected these protocols take care of the retransmissions. However, there is one more extreme case that we need to consider – signal loss. The actual radio connectivity can be lost due to lack of power in the mobile device or lack of coverage. In this case the low level protocols cannot handle the retransmission. The burden is now on the Level-5 protocols. Similarly to the Resilience case, a Level-5 protocol needs to be able to do retransmissions. To some extent one could observe that Resilience and Reliability have the similar meaning in case of Level-5 protocols in mobile networks.

3.4. Effective Level-5 Protocols in Mobile Networks

Effectiveness in mobile networks relates to two aspects – monetary and non-monetary. An effective Level-5 protocol is network friendly. In a mobile network where the radio resources are at premium the protocol needs to be light and not cause much traffic.

The network resources need to be considered mainly due to the monetary costs that are generated. These monetary costs can be direct – the amount of data sent and received is too expensive – or indirect – the amount of data is too high and consumes resources that will otherwise be useful to other services. The non-monetary aspects need to be considered as well. The mobile device resources are not costly but limited. The Processing power, In fact, in most of the usability tests I have conducted as part of my work showed that User Interface and Battery Capacity are low compared to other computing environments. In the next chapter we take a closer look at how the data communication takes place in a mobile network. This helps us understand what protocol efficiency means in a mobile environment.

4. Mobile Packet Access

In chapter 2 and chapter 3 we discussed two important aspects of Mobile Communications – QoS classes and protocol performance. When combining these two aspects we start talking about protocol performance within a QoS class. That means that the performance of a protocol needs to be considered in the context of the QoS class that it is used. A protocol can offer good performance when used by applications running under the *Conversational Class* but could be the wrong tool for the job within the *Interactive Class*. This example has been used here on purpose. One can argue that a protocol that is able to perform well in the *Conversational Class* where the need for resource is high, it performs even better in the *Interactive Class* where the resource need is not as high as in the *Conversational Class*. However, in this chapter we will see that the mobile network allocates the communication resources differently, depending on the QoS class. This, in practice, means that an application running within the *Conversational Class* has more resources to spare compared to an application running within the *Interactive Class*. This is the reason for analyzing the protocol performance. Protocols that have already been defined and perform well are not necessarily the best choice for the new applications being deployed.

4.1. Packet Data Mobile Communications

At the moment the four classes split the mobile network resources between each other in such manner that the *Interactive* and *Background Classes* used the Packet-Switch services and the *Conversational* and *Streaming Classes* use the Circuit-Switch services. This is not how the future looks like. The target architecture for the Next Generation mobile networks talks about running all the services on IP. Thus all of them will be using the Packet-Switch service.

The Packet-Switch data traffic for *Interactive* and *Background Classes* has been modelled by European Telecommunication Standards Institute (ETSI) and it is presented in Figure 3. One or more data packets make up a data call. This number of packets depends on the application. In most of the cases it is a bursty sequence, which is, in fact, a characteristic feature of the packet call. For instance, when browsing the WEB, the application receives a burst of packets that corresponds to the downloading process. After the WEB page is locally available the user will take the so called *Reading Time* in order to *consume* the content.

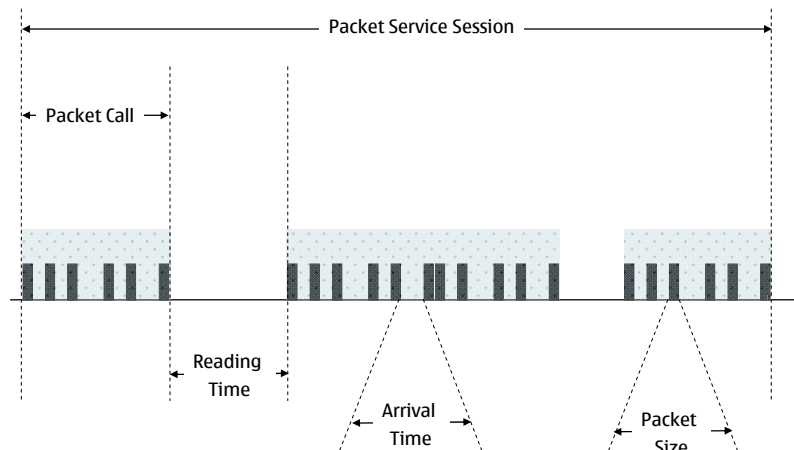


Figure 3 - ETSI Model for Packet Service Session

The model presented in Figure 3 is just an example. It is based on a WEB browsing session. Generally, a data traffic session is characterized by the following parameters:

- session arrival process
- number of Packet Calls per session
- reading time between packet calls
- number of packets with a Packet call
- packet size
- time between the transmissions of two packets within the same Packet Call

There are major differences between the applications running with the *Interactive* and *Background Classes* and those running within the *Conversational* and *Streaming Classes*. Nevertheless, all of them can run on packet networks. The main differences are stated in Table 2, next.

Table 2 - Differences between QoS classes

Conversational and Streaming	Interactive and Background
Packet Data is constant. The required bit rate is not expected to change during the session. The Packet Service Session is expected to be in one continuous Packet Call.	Packet Data is bursty. The required bit rate is expected to change rapidly from zero – reading time – to high bit rates – packet Call.
Real-time applications that do not tolerate delays. In case those delays	Non-real-time services are not affected by delays. Reasonable time intervals

occur the user experience is not affected.	between the Packet Calls do not affect the user experience. Especially in case of <i>Background</i> applications the user is not even aware of these delays.
Errors on the transmission media force the applications to retransmit packets. This is not always the way that real-time applications are implemented. In some cases in order to keep the user experience at reasonable expectation levels the packets are just dropped and the transmission continues from where it has been left. In any case, should the application retransmit the lost / corrupted packets there will be a delay, which affects the user experience anyway.	Packets can be retransmitted over the radio link and the experience is not affected. Of course the retransmission introduces delays but as long as they are kept within reasonable limits the user is not aware of them.

4.2. Wideband Code Division Multiple Access Packet Data (WCDMA)

The WCDMA networks seem to be the future of Mobile communications. All the considerations and assumptions being made at the moment, when developing new protocols, are based on the fact that applications using them will be running over radio networks governed by these technologies. In the following paragraphs we discuss how the data communication is handled in WCDMA networks.

There are three important aspects to be considered in the radio access. First is how to divide the available radio resource between the users so their transmission and reception needs are fulfilled. That means that the capacity of the air interface is shared. Another aspect to consider is what kind of transport channel is allocated to a user. Last, but not least, the network needs to monitor the packet allocation in order to keep the network load under control.

4.2.1. Transport Channels for Packet Data

There are three types of data channels to be used in order to transmit or receive a packet: *common*, *dedicated* and *shared* [Ghosh et. al., 1999]. How a client is allocated the data channel to communicate is decided in the network and it is based on the so called *scheduling algorithm*.

Common Channels

These channels are used mostly for carrying the signalling within the network. However, in some cases they are also used for user data. This is not the case in all the mobile networks. This is the way the communication channels are handled in WCDMA networks. Their main characteristic is the low setup time. This is needed since they are used in order to set up the communication itself.

There are many advantages in using these channels; however, they are not suitable in all the cases. One disadvantage of using them is the fact that they cannot handle the so called *soft handover*. This means, in mobility terms, that when the mobile device is roaming within the mobile network and there is a need to *travel* to another cell, the channel will break.

In conclusion, these channels are fast to establish in order to send and receive data and then tear-down, which will free them for other use. The network will not allocate them when the data amounts to be sent or received is high. Thus, if a protocol needs to make use of these channels it needs to work with small individual packets. We see here one reason why a protocol defined for fixed networks is not necessarily suitable for mobile use. Having a verbose protocol will prevent from the start the usage of the common channels for communication.

Dedicated Channels

The Dedicated Channels can be considered the exact opposite of the Common Channels. They take a lot more time to set up. However, there are advantages. The bit rates that can be achieved on these channels go as high as 2 megabits per second and the bit rate can be changed during the transmission. Also the radio performance is improved.

Any protocol can be used on these channels. Issues can arise only when the entities involved in the communication expect responses to their request within a certain time interval. The nature of the level-5 protocols requires this feature. This means, in practice, that if a protocol is verbose and will always be scheduled on dedicated channels, then it needs to be able to cope with certain delays due to the time needed for communication channel allocation.

Shared Channels

The basic idea is to share a channel in time between different users. The same codes are used among users. The bit rate is lower in comparison with the achieved rates on the

dedicated channels. This is not necessarily bad in case of those applications that generate bursty traffic. The advantage is that the capacity of the air interface is shared among many users at the same time – time as the user perceives it.

4.3. Selection of the Channel Type

We saw that the actual data communication in radio network takes place on Transport Channels. The type of channel being allocated for communication can affect the user experience – to better or worse – and can be friendly towards the air interface. We now take a look at how the channels are allocated. This gives a better understanding on how a protocol can consume or save resources.

The transport channel to be used for communication is chosen at the Radio Network Controller (RNC) level in the network [UMTS3003, 1997]. The decision the RNS make is based on:

- Service Type or bearer requirements (for example the delay parameters)
- Data amount
- Actual load of the common channels and shared channels
- Interference levels in the air interface
- Radio performance of different transport channels

Table 3 shows a summary of what kind of data can be transmitted on different transport channels.

Table 3 - Channel types and their properties

	Dedicated Channels	Common Channels		Shared Channels	
Uplink / Downlink	Both	Uplink	Downlink	Uplink	Downlink
Suited For	Medium or large data amounts	Small data amounts	Small or medium data amounts	Medium or large data amounts	Medium or large data amounts
Bursty traffic	No	Yes	Yes	Yes	Yes

We see that the communication protocol affects which transport channel is allocated for communication. In turn this affects how the mobile device reacts – for example when using the dedicated channel the fast power control is used, which, in turn, affects the battery consumption. On the other hand it affects the network capacity. For example, verbose protocols force the network to allocate dedicated channels, hence consuming the air interface.

5. Layered Networks Architecture

In the previous chapters we went through a short introduction on mobile networks and the services that are expected to evolve in mobile environments. One important aspect to consider here is the difference on how mobile services have been defined in the past decades compared to how they are currently being developed at the moment. In particular we need to take a look at how the communication protocols have been recently developed.

In cellular networks the communication protocols are the result of the standardization work of ETSI or other standardization bodies that are more or less closed. At the same time, the new services and applications work based on the support of the protocols defined by IETF. We saw in Chapter 2, that the so called new IETF Multimedia Architecture is becoming more and more a de facto standard. As a result, protocols defined by IETF are used by applications running in mobile environment. Due to the nature of IETF – it develops protocols for Internet use – some of these protocols are not suitable for mobile environments as they are defined. Optimizations are needed. In order to do these optimizations one needs to analyze the initial version of the protocol and then decide on which parts need to be changed.

In this chapter we take a look at the basic structure of the IETF's protocols. They are based on a layered structure defined by Opens Systems Interconnection (OSI) [ISO 1983]. This layered structure is not entirely adopted by IETF. However, the basic ideas are the same in both OSI and IETF views.

5.1. Data Networks and Layered Architectures

There are two modes in which the data transmission of data happens between two end-points. A connection oriented mode assumes that the packets are sent in a sequence that arrives at the receiving end-point in the same order as they were sent. This transmission sequence is constrained to happen as specified above. The alternative to connection oriented communication is the connectionless oriented or otherwise known as datagram mode. As the name suggests, the packets travel between end-points in an unorganized order. Packets can be received in a different order that they have been originally sent. The difference between the two communication methods is that in the first case a connection needs to be established between the end-points before the data can be exchanged. In practice a route needs to be established. In some cases this connection phase slows down the transfer rate [Chapin, 1983].

When data is transmitted over the network, no matter if it is a connection or a connectionless transmission it must be carried out in a timely and cost effective manner. Both concepts refer to the user. The data must reach its final destination uncorrupted and recognizable. The meaning of 'uncorrupted' and 'recognizable' are not in the scope

of the transmission technology. It is rather the interest of the consumer that defines those terms. By ‘user’ we do not necessarily identify a person. It can be any entity that is involved in a communication using a certain technology. For instance, a WEB Browser is using TCP / IP in order to communicate with a WEB server. It is not for the TCP / IP specification to define what ‘correct’ data mean.

We are now left with two problems in our hand – the two major problems of communications. On one hand we need to make sure that the data is correctly transported in a timely and cost effective manner. On the other hand we need to ensure that the data is delivered to its user in a recognizable form. The concept of layered architecture is next introduced in order to tackle these two issues. This concept distinguishes between two sets of layers – the lower layers and the upper layers. In the lower layers the data is sent across network nodes between devices. The upper layers need to process the raw data and provide it to its user in a recognizable form. Figure 4 and Figure 5 show the layered architecture model as defined by OSI. They are both defined as part of the work done inside the International Organization for Standardization [ISO 1983].

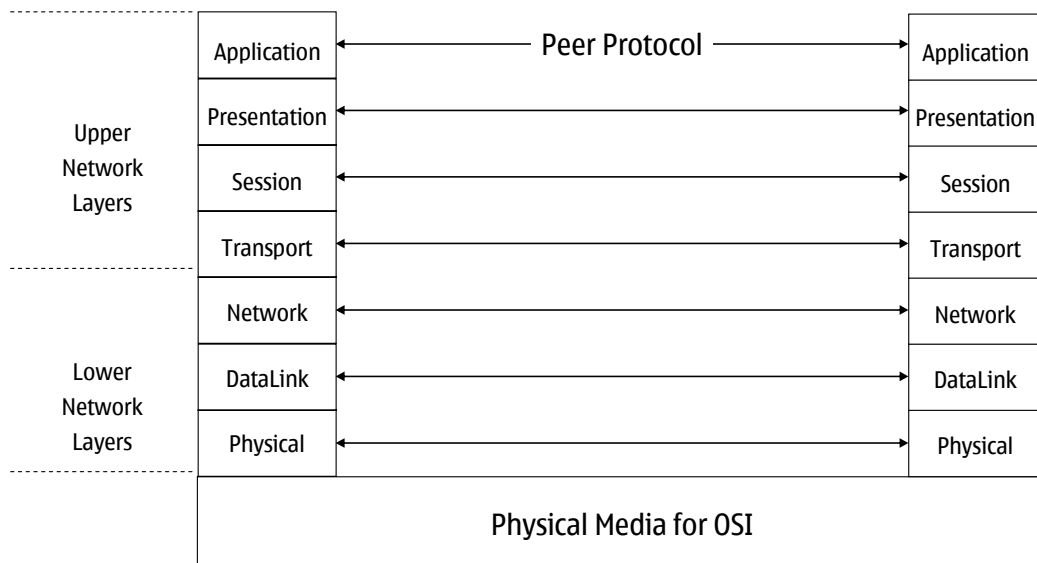


Figure 4 - Layered Model and Peer Protocols

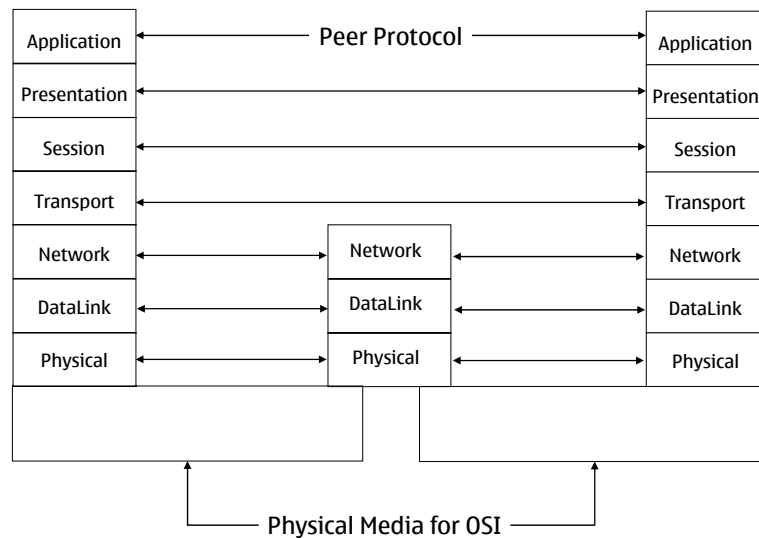


Figure 5 - Relay Open System

This seven-layered architecture (see Figure 5) assumes that the bottom three layers are taking care of the networking, while the upper four layers are taking care of the data processing and presentation towards its intended user.

5.2. OSI Standard Architecture and Protocols

The OSI Standard architecture introduced in Figure 4 and Figure 5 has been defined in order to allow its users to conduct an open network communication. This seven-layered model allows the definition of networking protocols. One can wonder why a *seven-layered* architecture. The answer has been attempted many times. OSI supporters think that this is the best way to ensure good and viable products [Zimmerman, 1980]. As can be seen later in this chapter, when we discuss the Internet reference model, this *seven-layers* is not a magic number in all the cases.

The idea of having a layered architecture however is due to the needs of having just enough processing levels that (1) are not too complex to define and implement; (2) in order not to have too many integration points; and (3) allows the selection of boundaries that group similar functions into one layer and different functions into different layers. This layered architecture results in a minimal interaction between layers.

Application Layer

The top-most layer in the OSI architecture is the application layer. Its task is to ensure that two or more applications carrying out the communication over the network

and reside in different nodes, understand each other. This means that the semantics of the communication are taken care of within this layer.

Presentation Layer

As shown in Figure 5 the presentation layer is right under the application layer and uses the services provided by the session layer. This means that its task is to ensure the correct syntax of the communication. It isolates the applications layer from the differences found in the representation and syntax of the data being transmitted. furthermore, this layer provides the means to the upper layers in choosing the syntax to be used when data is transmitted between the entities.

Session Layer

The third layer downwards in the OSI architecture (Figure 5) is the session layer. This provides services to the presentation layers. Its meaning is to manage the dialogue between the presentation layers – directly – and application layer – indirectly. A connection must be first set up before any communication can occur. In consequence, this layer allows its users to conduct an orderly dialogue.

Transport Layer

Moving down in the layer architecture but staying still in the Upper part one views the transport layer. Its services are used by the session layer. In here we distinguish between two types of data transmission that have been briefly discussed in the beginning of this chapter – connection oriented and connectionless transmissions.

The four layers described above constitute the Upper layers on the OSI hierarchy. The protocols that are corresponding to these layers reside in hosts (end-points) involved in the communication. These upper layers use services offered by the lower levels of the OSI architecture. The protocols that correspond to these layers reside in the network nodes and their task is to route the messages from the source to origin.

The lower layers highly depend on the actual network being used for communication hence it is hard to describe each one individually. A few characteristics of these layers in mobile networks have been already discussed in Chapter 3. More will be discussed later in this chapter when we talk about the Internet reference in mobile networks.

5.3. OSI Protocols

One thing that probably kept the reader's attention was the fact that all the services mentioned in the OSI architecture provide services or consume services. This is

probably the only similarity between them. That leads us to define two concepts; a *service user* is a layer that uses services from an immediate lower service. In a similar manner we define the *service provider* as the lower layer providing services to its immediate upper layer. When looking at the interaction between a *service provider* and a *service user* we notice three phases of operation. In Figure 6 one can see the behaviour of two systems that desire to communicate. Here the three phases are in order: (1) a connection establishment, (2) a data transfer and, finally, (3) the connection release.

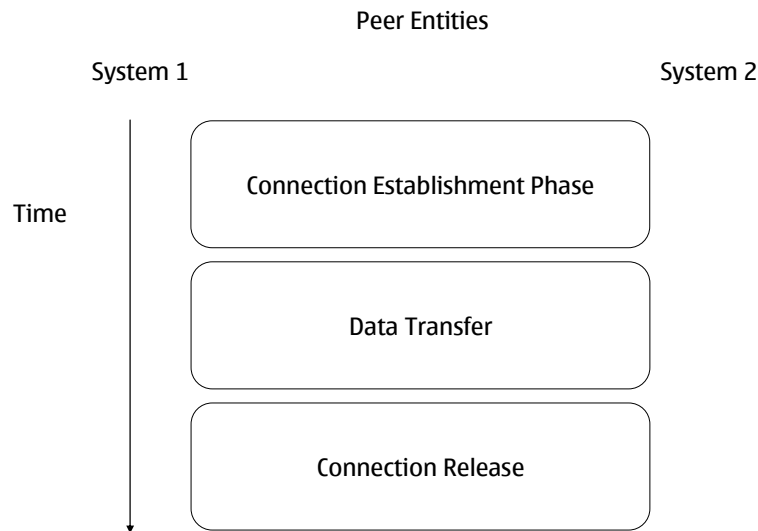


Figure 6 - A 3-phased communication at a layer

In the first phase of the communication, the two peer entities will open a connection and negotiate a set of parameters to be used during the data transmission. Once this step is carried out the communication goes to the actual data transfer. At this point in the communication sequence the data is exchanged between the two end-points. Error control is performed – this is one of the services that a *service provider* must offer. Other services can be offered as well.

Depending on the layer, we have different requirements. However, we have a few similar concepts. This allows us to define a unified concept where a Layer N offers services to a higher layer N+1. Figure 7 shows this concept schematically.

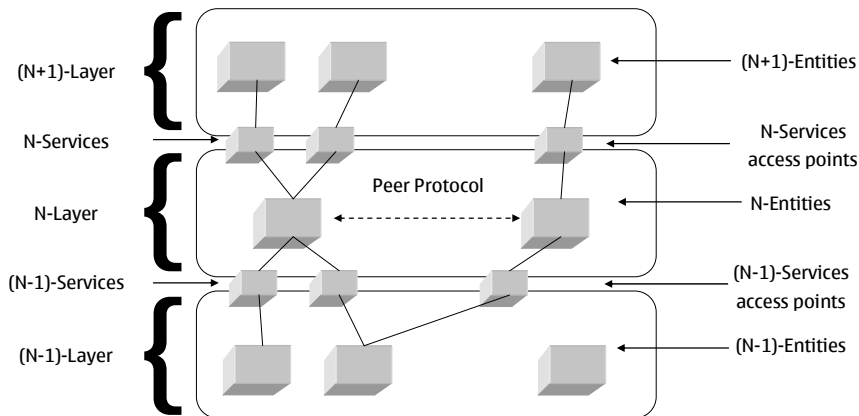


Figure 7 - (N+1)-Entities and N-Services

In an OSI layered Architecture, the N-entities in the N-layer provide services to the (N+1)-entities. This happens via the so called Service Access Points. In this case the N-entity is a *service provider* for the (N+1)-Entity.

The data transferred between the peer entities contains: (i) user data, passed from the (N+1)-layer towards its *service provider*; and (ii) protocol control information added in the N-Layer. Figure 8 shows how a *service provider* layer adds its needed control information to the Protocol Data Unit (PDU) received from the *service user* layer. A PDU generated by one layer contains both the Protocol Control Information (PCI) added within the layer and the user data originated in the layer above. In Figure 8 the data crossing the boundary between the (N+1)-layer and N-layer is mapped as N-Service Data Unit (N-SDU). The way that this N-SDU is sent forward to the (N-1)-layer depends on the size of N-SDU and the capability of the protocols running on N-layer.

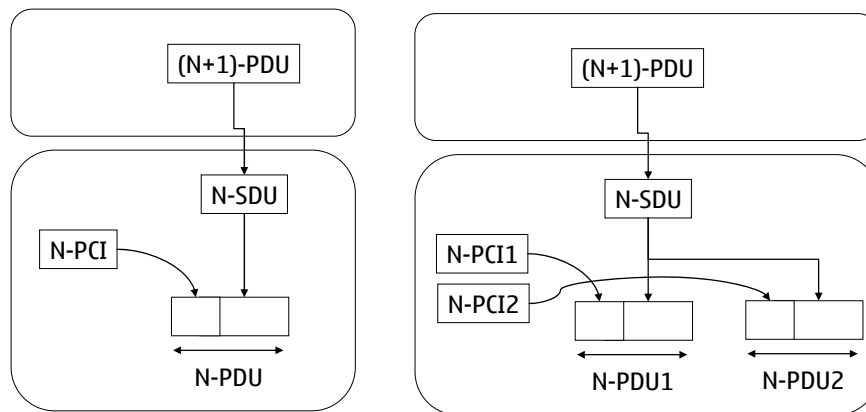


Figure 8 - Data Units according to OSI Architecture

In mobile networks, the concept introduced above is very important. Especially when applications use the Internet Protocols defined by IETF the Transport Control Protocol / Internet Protocol (TCP / IP) or User Datagram Protocol / Internet Protocol (UDP / IP) are used as transport. In practice, the size of the PDU as the transport layer receives is too big to be sent in one PDU by the network layer or the layers below. This results in many low level PDUs sent over the air interface. We saw in chapter 3 that the scheduling algorithms choose a transport channel depending on the amount of data to be sent. Now we understand better why that happens. In case that the PDU is too big it needs to be split in many low level PDUs. That results in a somewhat large number of messages. That, in turn, results in the need for a dedicated channel being chosen in order to be able to cope with the mobility.

Service Primitives

Following the model above, the OSI standardization body defined four basic service primitives at each level of the architecture. These primitives will provide the interaction between the *service provider* and its *service user*. The four types are: (1) *request*, (2) *indication*, (3) *response* and (4) *confirm*. These primitives are represented schematically in Figure 9. In System 1 the (N+1)-layer issues a *request* in order to invoke a procedure at the N-layer. As a result an N-PDU is sent to System 2 at its N-layer. Depending on the actual system environment we are discussing here, it will generate an *indication* being sent to (N+1)-layer in System 2. A *response* is always

generated and as a result an N-PDU is sent back to System 1. At this point the N-PDU received in the N-layer is sent upwards to (N+1)-layer and a *confirmation*.

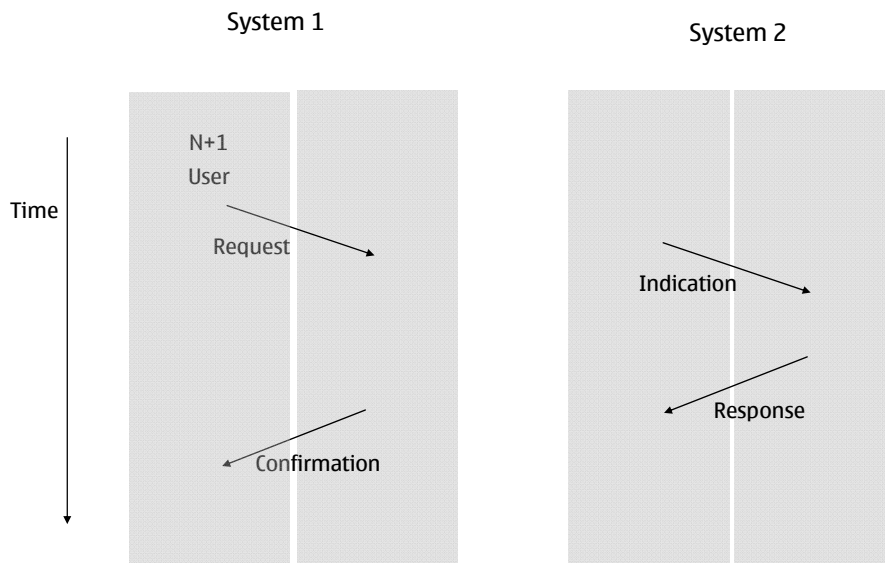


Figure 9 - Basic OSI primitives

Again it is important to remember that depending on the capabilities of the lower levels the N-PDU can be one message or more. Especially in mobile networks it is very likely that we are talking about more messages on the air interface when the size of the (N+1)-PDU is large. For instance, the Internet datagram size is 1500 bytes. Thus, when messages larger than 1500 bytes are sent over Internet they are split at the IP level into many shorter datagrams (shorter than 1500 bytes). This is the only way that long messages can be sent. In case an application sends large messages over UDP / IP there is a high risk to lose data. This happens because the connectionless nature of the UDP protocol. If the same application is sending the messages over TCP / IP the data is communication is safer. The connection-oriented nature of TCP protocol ensures that the IP datagrams are delivered to their destination.

5.4. Internet Reference Model

OSI Model has been traditionally and widely used for years in the development of new protocols. However, the mistake that many tend to do when studying or designing a new Internet Protocol is to try to fit it into one of the seven basic layers. The main issue considered here is that in the nowadays world, the Internet protocols are designed and developed according to the TCP/IP model also known as Internet Reference Model.

In an IETF document - Some Internet Architectural Guidelines and Philosophy [RFC3439, 2002] - the authors state the philosophical guidelines and principles that architects and designers of Internet backbone networks should adhere to. In this

document a section entitled “Layering Considered Harmful” emphasizes the fact that the layering, as a key driver, is not a feature of the Internet Reference Model. It is, in fact, an added feature of the OSI Model, and as a consequence it is not a good idea to force this layering onto an Internet Architecture.

TCP/IP or Internet Reference Model was created in 1970s by the Defense Advanced Research Projects Agency (DARPA). Its intended use was to assist Internet Protocol development. It is a layered abstract description for communications and computer network protocols and its original form described a four-layered architecture. In fact, Internet Engineering Task Force (IETF) has never agreed with the idea of a five-layer model, since the lower transmission layers have never been a part of IETF’s agenda. One other reason to consider is the fact that the Internet Reference Model has been defined before the OSI Model. In this context IETF has never felt the need or obligation to adhere to it. The seven-layer model does not reflect the real-world protocol architecture as used in Internet [RFC1122, 1989].

In the Internet Reference Model the layers close to the top are closer to the applications as the lower layers are closer to the actual transmission of the. Figure 10 depicts the IP Suite stack showing two hosts connected via a number of routers. The picture also shows the corresponding layers at each hop.

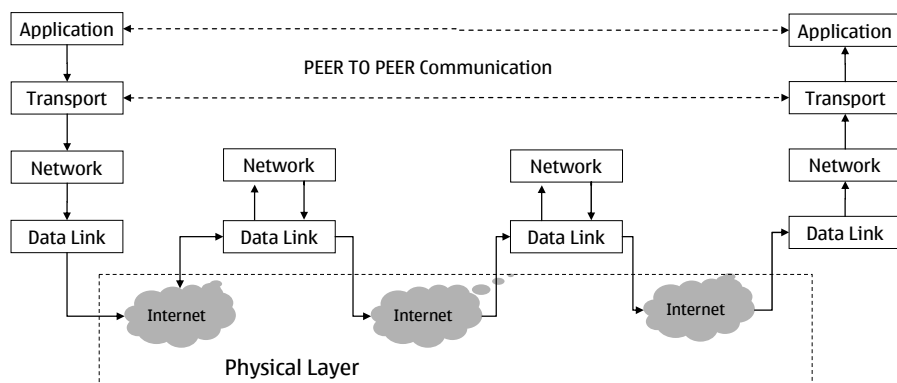


Figure 10 - IP Suite Stack Host to Host communication over Internet

5.5. Internet Reference Model in Mobile Networks

In a Mobile Network it is obvious that a Mobile device does not connect directly to an Application Server via the same physical media. In this case the communication

between the two entities goes according to the Model described in Figure 10 – Host to Host Communication over Internet. The mobile host is, in practice, a mobile handset. An application running in the device communicates with a network-based host known as Application Server. The data between the two hosts is transmitted over a number of routers and network elements. In practice the information travels over the radio interface between the mobile station and the so called Base Transmission Station. From here onwards the data is sent over various different types of media. It can be fibre optics, wired networks, etc...

At the application level the communication between the two entities goes over a network protocol. In this thesis we discuss the case when the mobile handset and the Application Server communicate over a protocol that fits into the fifth level of the Internet Reference Model – see Figure 10. We study how the protocol uses the resources of the underlying layers that provide data transmissions.

TCP and / or UDP are the transport protocols most commonly used. IP Connectivity is provided between the Mobile Device and Gateway General Packet Radio Service (GPRS) Support Node over the Radio Interface and a series of Network Elements. This one in turn relays the IP information to the Application Server over wired Ethernet. Figure 11 shows the Internet Reference Model in a Mobile Network.

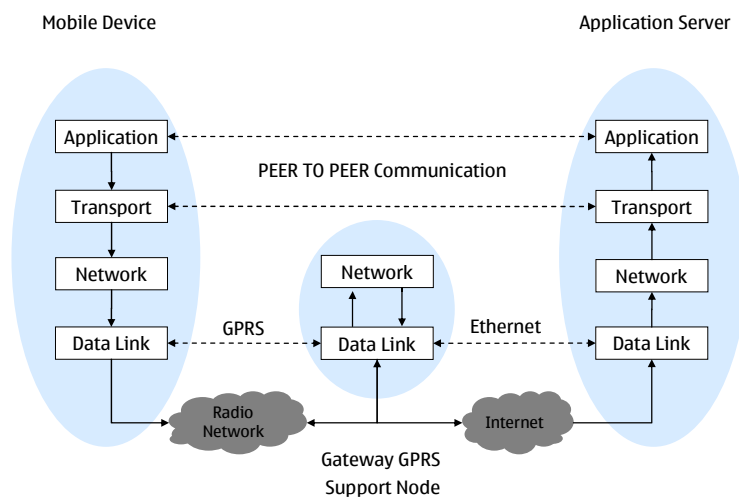


Figure 11 - Internet Reference Model in Mobile Networks

6. Protocol Modelling

Informal methods for protocol development have been successfully used in the communication protocols development area. However, with their ever increasing complexity formal models have also been defined and used in order to specify and analyze more rigorously and more efficiently the communication protocols. In fact there are several excellent works on the subject [Brand and Zafiropulo, 1981] proposing various different formal methods that provide state-of-the-art tools for validating and verifying protocols.

It must be noted that the methods mentioned above are valuable only for the verification and validation. Performance analysis of network protocols has not been addressed much. In fact, when addressed it has been done in an informal manner. Several surveys conducted as part of IETF work measure the performance of Internet protocols but none of them proposes a general model to start from. This is the root cause for many debates on which protocol offers the best performance. For example, Singh et al. in Presence Optimization Techniques [SINGH, 2006] acknowledges the verboseness of the SIMPLE protocol for Presence and proposes few optimizations; but the study does not offer clear figures on the value of the traffic both before and after the optimization.

In a similar manner, Saint-Andre in his Interdomain Presence Scaling Analysis for the Extensible Messaging and Presence Protocol (XMPP) [SAINTANDRE, 2007] uses an informal method to calculate the generated traffic while using the XMPP protocol. The work is valuable since it gives a good overview on how much traffic an XMPP client generates in this particular case. However, the lack of formalism makes it hard to estimate how the results can be compared with similar figures computed for another protocol fulfilling the same use case.

The work on IETF protocols performance generates a lot of debate. In most of the cases two similar technologies compete for a place in the technology landscape. Which one is better? Which one is more suitable for a certain case? These questions are hard to answer if a common model for analyzing the performance of the “competing” technologies is not in place. These are, on one side, interesting studies discussing the optimization of the SIMPLE protocol [Singh et al., 2006]. On the other hand we have another paper discussing the traffic value for particular use case using XMPP. However, during the IETF meeting I have participated I observed that there are also experts advocating for both of them. Without a common ground for comparison it is not easy to take sides. Furthermore, it is difficult to evaluate these research results due to lack of comparison standards.

Using Models in Protocol Development is not a new concept in IETF. The need for such models comes from the fact that IETF Process itself is based on peer review. RFC

4101 [RFC4101, 2005] proposes an approach to allow reviewers to quickly understand the essence of the system. If a Model is proposed for the development process of an IETF protocol, why wouldn't there be a model that proposes a common way of analyzing the protocol performance when it comes to generated traffic? In the following chapters we take a look at a few models used for protocol specification. We try to find a model that suits our purpose to calculate the traffic generated by the protocol.

6.1. Models for Protocol Specification

A protocol specification is irrelevant to its user. The machine providing the service is a so called black box. The internal structure is not to be shown to the consumer users. However, we saw in the Chapter 2 that a network protocol designer must be concerned with the internal structure of the protocol. A protocol must be defined considering its context. This context is in fact given by the architectural layer where the protocol is used (see Figure 12). In this thesis we address the Level 5 protocols as defined by IETF [RFC1122, 1989]. Apart from the layer where the protocol operates we need to describe the protocol used between the entities involved in the communication. This includes:

- Informal operation of the entities
- Actual protocol specification:
 - Types of data exchanged between entities
 - Messages exchanged between entities
 - How an entity reacts to external events including - but not only user - commands
 - How an entity reacts to receiving messages from other entities
 - How an entity reacts to internal events
- Additional details not included above, such as:
 - Efficiency consideration
 - Implementation guidelines

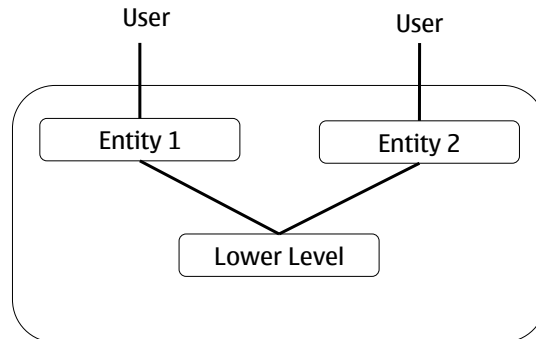


Figure 12 – Architectural Layer

The description stated above needs to be concise, precise and easy to understand. This is often hard to achieve since these goals usually conflict. An easy to understand protocol definition turns easily into an informal definition. It becomes ambiguous hence, it is not precise anymore. On the other hand a more formal approach, though precise, tends to be hard to understand. It seems that it is always a trade-off between using one of the two approaches. For our goal, however, a formal model is needed. Next we take a quick look at three most used protocol models.

6.2. High Level Programming languages

Parallel Programs are probably the most general model to describe protocols [Brand and Zafiropulo, 1981]. A party involved in the communication is modelled using a formal description - a high level program. The languages used are universal hence one can represent any characteristic of the interested party.

In practice, these high level programming languages are a convenient tool. They can be used in order to represent numbers, data, variable, counters, etc... However, they are not that useful when it comes to complex structures. In this case these models are used mainly for representing the data transfer aspects of the protocols. In order to address all the aspects of a protocol other methods are used – mostly graphs and Petri Nets.

The power of the high level programming languages in representing the data transfer can help us. However, they lack features when it comes to dealing with the other aspects of the protocol definition such distinguishing between inputs and outputs

[Sunshine, 1978; Sunshine, 1981]. In this case they do not fit to our purpose of calculating the traffic generated by a protocol according to a specific use case.

6.3. Finite State Machines

Generally, Finite State Machines have been used at quite early stages in protocol development [Merlin, 1979]. Depending on the protocol a single machine can be used to describe the whole protocol. Another alternative is to use a machine for each party involved in the protocol. If a multi-machine approach is used we talk about a pair of transactions – SEND / RECV. A SEND fires simultaneously with a RECV. They both have the same parameter - for example the same messages are sent and receive respectively. The model is applicable to any protocol having a finite number parties involved in communication.

Brand and Zafiropulo work [Brand and Zafiropulo, 1981] is one excellent paper that describes the Finite State machine Model. In this model each process is represented as a finite state machine. These processes are then “connected” using a First In First Out (FIFO) channel. The channel is in this so called Ideal – communication is smooth and messages are not lost or corrupted. In case we need to model a non-ideal channel we can introduce a new finite state machine that behaves according to our specifications and introduces errors in the communication (drops messages, corrupts messages, etc...)

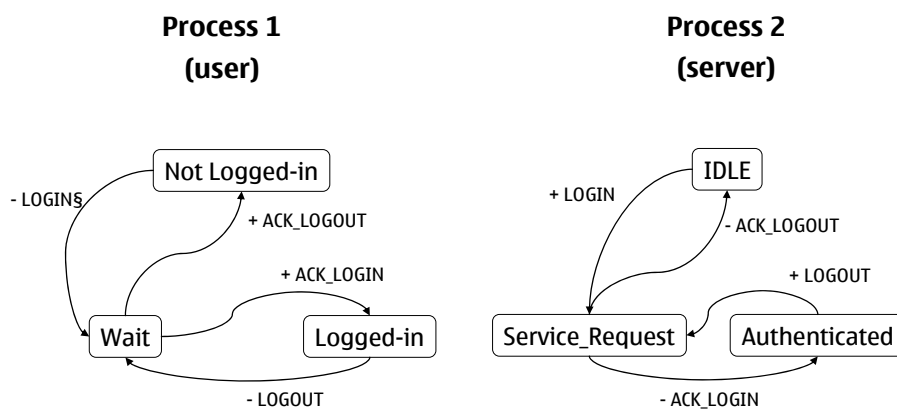


Figure 13 - Client Server Protocol modelled with Finite State Machines

A protocol is modelled using finite state machines (see also Wood, 1987) and using a specific notation for expressing the transmissions. An example is shown in Figure 13. A Client Server Protocol is modelled so that a “minus” sign is used for sent messages

and a “plus” sign is used for the reception. Protocol states are modelled using FSM states. A transition from one state to another happens when a message is sent or received – similarly to graphs an arc is traversed.

In our example a Client is in a Not Logged-in state. When a login (- LOGIN) is sent the server receives the message (+ LOGIN) and enters a Service_Request state where it authenticates the Client. While the server-process the login request the client is in Wait state. On successful authentication a response is sent (- ACK_LOGIN) and the server state changes to Authenticated. The client receives the response to the login request (+ ACK_LOGIN) and enters the Logged-in state. When the Client needs to disconnect a new request is sent (- LOGOUT) and again the Wait state is activated. The server receives the message (+LOGOUT) and moves to Service_Request state to process. An acknowledgement is sent back to client (-ACK_LOGOUT) and server moves to IDLE state – the session is closed.

A model is defined formally in [ZAFIROPULO]:

Definition: A protocol Finite State Protocol (FSP) is a quadruple

$$FSP = (S, M, \Sigma, O, \tau)$$

- (i) $S = (S_1, S_2)$, two disjoint finite sets representing the states of processes P_1 and P_2 respectively
- (ii) $M = (M_{12}, M_{21})$, two finite sets where M_{12} represent the messages that can be sent from P_1 to P_2 and M_{21} represent the messages that can be sent from P_2 to P_1
- (iii) $\Sigma = (\Sigma_1, \Sigma_2)$, two finite sets on M of the following kind
 - for every message $\chi \in M_{ij}$, the sending of message χ is denoted by $-\chi$. Every sent element $-\chi$ is an element of Σ_i
 - for every message $\chi \in M_{ji}$, the sending of message χ is denoted by $+\chi$. Every sent element $+\chi$ is an element of Σ_j
 - λ is an empty string of event or no event $\lambda = \{ \}$ or λ is an element of Σ_i .
- (iv) $O = (o_1, o_2)$, where $o_1 \in S_1$ and $o_2 \in S_2$. o_1 and o_2 are the initial states for processes P_1 and P_2 respectively
- (v) τ is a transition function: $S_i \times \Sigma_i \rightarrow S_i$, $i = 1, 2$. The transition function for an event $\sigma \in \Sigma_i$ at state s can be written as $\tau(s, \sigma)$. It represents the next state reached after triggering the transition σ at s

Definition: A channel C_{ij} is a FIFO queue connecting process i and process j . The contents of C_{ij} is marked c_{ij} . This is a string of symbols from M_{ij} ($c_{ij} \in M_{ij}$) and represents the queue of messages sent from i to j .

In this model there are no assumptions on time. For example, it is not specified how long a message spends in a transmission channel. Also, it is not specified how long a process takes to process a message – the server can take as much as needed to process the LOGIN request. These can be limitations depending on what properties of the protocol need to be analyzed. However, in our case we are not interested in how much time a party needs to process a message as we are not interested in how much time a message spends in the transmission media. In fact in the mobile network this is not trivial to model this. The needed time for message to be transmitted over the radio interface depends on many different variables such as signal strength, distance to the base station or available radio technology. For example, we know that a mobile network today offers at least GSM and UMTS as communication technology and in some cases Wireless LAN is available as well. The time a message spends in the channel varies a lot depending on which of these technologies are used.

One other limitation to consider is the inability to express a protocol where there are an arbitrary number of messages in transit. Also, it does not address the size of the messages. As a consequence, without improvements, this model is not enough for those situations where we study a protocol performance for a specific use case. FSMs, as a dynamic and computational formalism, with a natural testing procedure encapsulated in their structure, effectively support the accurate representation of a communication protocol used in mobile environments. Checking the performance of it, though, requires suitable metrics to define the system's states, especially those associated with the communication system evolution – due to the amount of messages - and those associated to the communication system's entropy – caused due to the amount and size of the sent or received messages [Berki, 2001, p232-233].

The Finite State Machines are, however, powerful and expressive enough to allow us to model a Level 5 Protocol. The number of messages exchanged over the network could also be calculated as well. For measuring the amount of data improvement is needed. Metrics, morphological and complexity metrics in particular, are significant indicators for estimation and evaluation on system's zoticality [Berki, 2001, p232] which is the ability of a system to respond to dynamic changes.

6.4. Petri Nets

Petri Nets are yet another technique that can be used to model a communication protocol [Peterson, 1977]. For example, using Petri Nets one can model a protocol that has an infinite number of states. More, we can develop a model that allows messages to have size using the so called Coloured Petri Nets extension [Jensen, 1994]. The main shortcoming of this modelling tool is, as in case of Finite State Machines, the size of the graph. Even in the case of simple protocols this can result in growing to complexities that are not easy to manage. In this chapter we discuss the Petri Nets and their Coloured Petri Nets extension.

In a nut-shell a Petri Net is a *formal* and *executable* technique that allows a *graphical* specification of concurrent, dynamic systems. Its *formalism* ensures that the technique is mathematically sound. The *graphical* technique, which in fact is part of the graph theory, allows a better understanding of the system being modelled. At the same time the complexity can grow beyond levels that can be easily managed. However, tools exist to allow construction and visualization with ease. Same tools can be used to *execute* and observe the dynamic behaviour of the model.

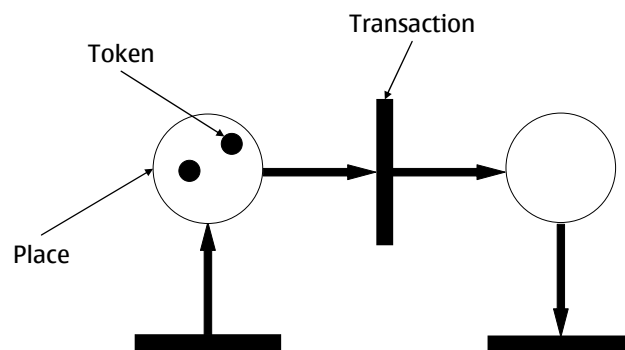


Figure 14 - Marked Petri Net

In Figure 14 we have a Graphical representation of a Petri Net. From the picture we can easily distinguish three main elements:

Places

Nodes of the Petri Nets containing *Tokens* and connect via arcs to *Transitions*.

Transitions

Nodes of the Petri Nets that can *fire* and move *Tokens* between *Places*.

Directed Arcs

Connects *Places* with *Transitions* and *Transitions* with *Places*. When a *Directed Arc* connects a *Place* with a *Transition* we call the *Place* an *Input Place*. When a *Directed Arc* connects a *Transition* to a *Place* we call the place an *Output Place*.

Tokens

Places contain *Tokens* and the distribution of the *Tokens* inside a Petri Net is called *marking* of the net. When a transition is *enabled*, it *fires* and takes away a *Token* from an *Input Place* and moves it to an *Output Place*. A *Transition* is *enabled* when there is at least one *Token* in an *Input Place* connected to the *Transition* with a *Directed Arc*.

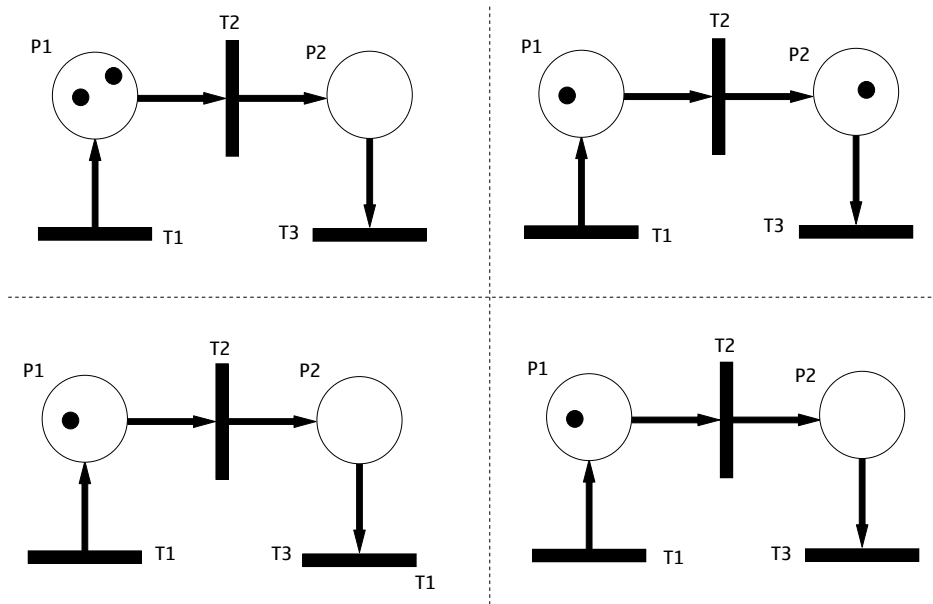


Figure 15 - Petri Net at work

In Figure 15 we see a simple Petri Net. P1 and P2 are *Places*. T1, T2, T3 are *Transitions*. P1 is an *Output Place* for T1 and an *Input Place* for T2. A *Token* is placed in P1 when T1. T2 *enabled* and can *fire*. A *Token* is taken from P1 and placed in P2. In a similar manner now T3 can *fire*. The *Token* from P2 is taken away. This is a simple example of a Petri Net. We can already notice a very important property. T1 has no input conditions, therefore, it can fire arbitrarily and produce an arbitrary number of *Tokens* in P1.

The Petri Net Model is formally defined in [Peterson, 1977]. This is not a complete list of references since the subject is widely debated. The following definition is not the only one that can be found in the literature.

Definition: A Petri Net is a Tuple

$$PN = (S, T, F, M_0) \text{ where}$$

- (i) S Is a set of *Places*
- (ii) T is a set of *Transitions*
- (iii) S and T are disjoint

- (iv) F is a set of *Direct Arcs* with the restriction that F cannot contain an arc that connects two *Places* or two *Transitions*
- $$F \subseteq (S \times T) \cup (T \times S)$$
- (v) $M_0 : S \rightarrow \mathbb{N}$ is an *initial marking*, $s \in S$ there are $n_s \in \mathbb{N}$ *Tokens*

We notice from the definition above and the definition of the Finite State Machine that the Petri Nets are a broader model. In fact a Finite State machine is Petri Net with the restriction that for each transition there is only one incoming arc and only one outgoing arc. In Petri Nets terminology a *Transition* has only one *Input Place*. It has been argued and further shown [Holcombe, 1998; Berki, 2001] that a FSM can represent a Petri Net.

How can a communication protocol be modelled using Petri Nets? Figure 16 shows how the same Login / Logout sequence between a Client and a Server that has been modelled with Finite State Machines is modelled with Petri Nets.

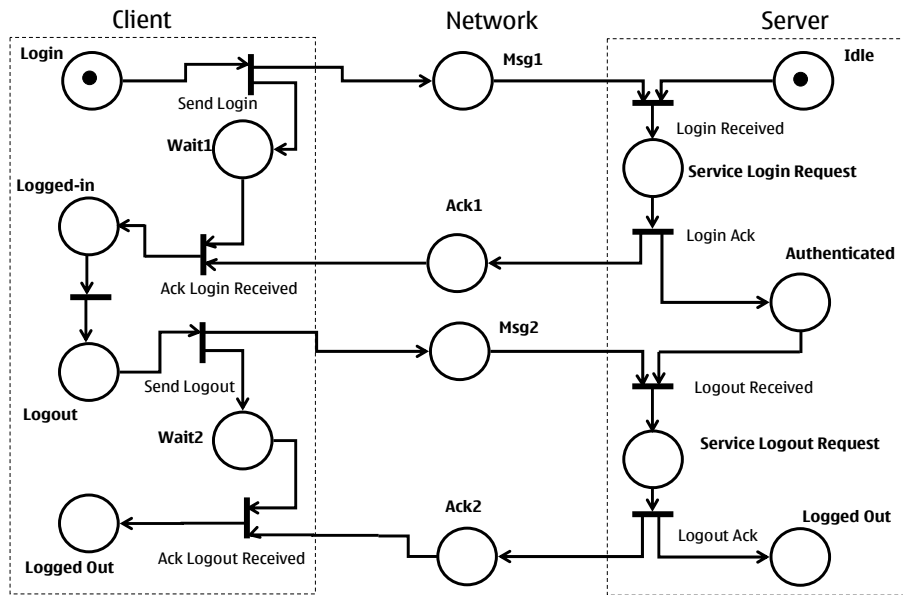


Figure 16 - Modelling communication Protocols with Petri Nets

In the model above the Client and the server are in an initial state – Client not logged in and Server in Idle state. This is also the initial marking of the Petri Net. In this initial state only one transaction can *fire* – Send Login. When this one *fires* a token is taken from the Login Place and put in Wait and on the Client Side and Msg1 on the Network Side. When the Msg1 Place contains a *Token* Login Received *Transaction* is *enabled* on the Server side. This *fires* putting a token in Service Login Request. In turn this enables Login Ack *Transaction* which fires putting a *Token* in Ack1 Place on the Network Side and another *Token* in the Authenticated Place on the Server. When Ack1 contains a *Token* Ack Login Received *Transaction* is *enabled* – remember that Wait1

had a *Token* since the Login messages was sent out. After Ack Login received *fires* a *Token* appears in Logout. At this stage the logout procedure can start in a similar manner.

We saw how Finite State machines and Petri Nets can be used to model simple communications protocols. Similar techniques can be used in order to model more complicated scenarios. However, both of these techniques fail to help us all the way in measuring the performance of the protocol when a certain use case needs to be fulfilled. First, we need to come up with a technique that allows us to model a Use Case. Second, while both models can measure the number of messages over sent between two entities involved in communication they both fail to measure the amount of data sent over the network. In fact the FSMs and Petri Nets have been used and praised for their strengths by a number of researchers, modellers and meta-modellers in traditional, classic research [Holcombe, 1998; Berki and Georgiadou, 1996; Berki, 2001; Saadia, 1999]. Their computational features and encapsulated abstraction server as a guarantee for formal specification, where rigour is required. FSMs, in particular, facilitate testing [Chow, 1978; Fujiwara et al., 1991; Berki 2001] which is a requirement for modelling communication protocols, since it enhances the trust on the communication procedure and increases the system usability (see chapter 3, page 9). The FSM formalism, however, can sometimes be too generic and general to model complex situations and systems rich in detail. A more generic, general and still computational and expressive pattern is needed in order to capture semantics and syntax of situations where the details can be proved critical in modelling [Berki, 2001] and metamodelling [Berki et al, 2004; Berki and Novakovic, 1995]. Later on Berki, Holcombe and Ipate [Holcombe and Ipate, 1998; Berki et al, 2004] have provided criticisms as well as suggestions regarding the extension of their modelling power. They have extended the FSM to the X-Machine model and subsequently used it to model a wide range of complex situations in diverse application domains. The X-Machine ideas were based on the theory of the general machines and general automata [Eilenberg, 1974] and they were further developed (see e.g. Balanescu et al., 2000) and applied widely in modelling and metamodelling, with specialized semantics and customized notations to the particular application domain. These new extended models always kept their computational rigour and ability to provide testing, in order to check the executions and communication paths and ultimately provide an indication for improving system's performance (see also Veijalainen et al. 2005).

The next chapter also proceeds to an extension and enhancement of FSM in order to provide its modelling power for the dynamics of communication protocols.

7. Modelling Protocol for Performance Measurement

We already discussed that in order for a protocol to perform well it needs to be *Effective*, *Resilient* and *Reliable* (see also Pfleeger, 1998; Sommerville, 1992). Depending on the context in which the protocol is used these terms have different meaning; *Resiliency* and *Reliability* are out of the scope of this thesis. Our research is not trying to prove that protocols are correct but rather considers them as correct and tries to build a model that allows us to measure how they behave in certain situations – we call those situation use cases. However, in the past, the models previously discussed in this thesis were widely used in order to verify and validate communication protocols. This proves that they are both sound tools for protocol modelling. This chapter suggests possible improvements for these traditional models that allow us to see how *effective* a protocol is.

The improvement is needed due to the fact that both Finite State Machines and Petri Nets fail in measuring the amount of data over the network due to the fact that none of them can quantify the primitives, which model the amount messages exchanged between entities involved in communication.

7.1. Measuring Effectiveness

Quality is an elusive term [Berki et al, 2004] by its very nature. System *effectiveness* is, like the concept of software quality, a loose term, dependent on the interested in it group of people (Berki and Siakas, 2007). Saying that a communication protocol is effective does not necessarily mean much if we do not describe also the context in which the protocol is used. Moreover, saying that one protocol is more effective than another is hard without having a common comparison ground to base this statement on. Usually, when reaching to such statements one needs to describe the input data set and the measured outcome. Thus, we need to define a model for the input data set and list that contain the required parameters to be measured. Similar FSM extended models are the so called Stream X-machines [Holcombe and Ipate, 1998], communicating X-machines [Barnard and Woodward, 1996; Barnard, 1998] and others.

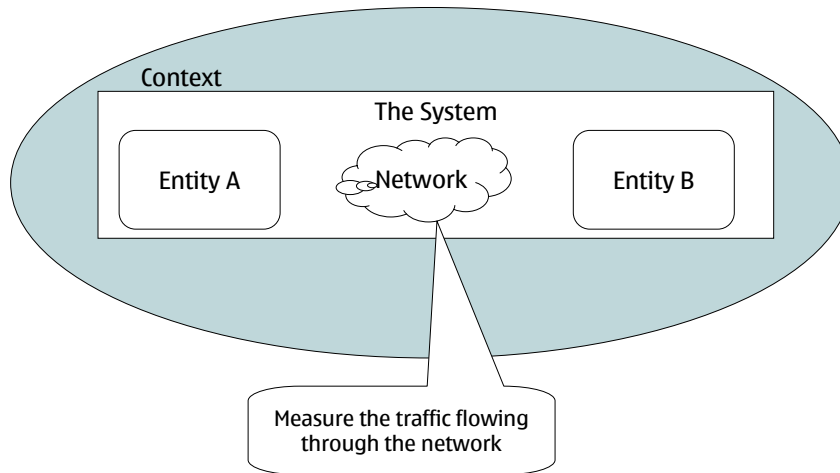


Figure 17 - Protocol Measurement System

Input Data Set

Different protocols are developed for the same purpose. This is not news since we can find many examples in IETF's work. Maybe the first that come to someone's mind is the classic Electronic Mail. Two famous protocols – Post Office Protocol (POP3) and Internet mail Access Protocol (IMAP4) – have been used in parallel for decades. Some favoured POP3 and some favoured IMAP4. It is many times supported that one is better than the other. POP3 has always been favoured by those “light” email clients that are not always connected, while IMAP4 is the de-facto standard for Email clients accessing complex mailboxes containing large amount of Email messages distributed in various different folders. The point is that if one was to test a POP3 on such complex mailbox the outcome would not be desirable. However, this does not make POP3 an undesirable protocol; it is just not suitable for that particular job.

Testing two different protocols that are developed for the same purpose is not easy especially when we need to decide which one performs better. The best starting point is to make sure that we feed the same information from outside the system and have a common understanding on how the outcome is measured. In our case, the outcome that is measured is the amount of traffic over the network. Figure 17 shows a high level picture of our measurement environment.

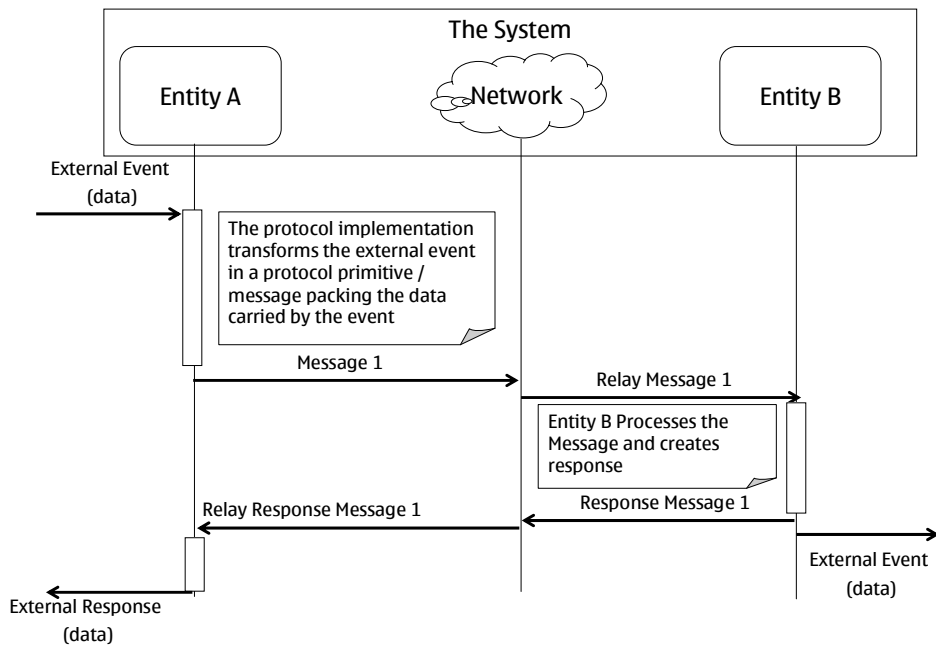


Figure 18 - Handling an external event in the measuring environment

When testing a communication protocol we need to first define and model the context where the measurement takes place. This model needs to define what are the external events that the System undergoing testing need to handle. Eventually the model specifies what Events the context expects from the System. Figure 18 describes how the Context gives an external Event to one of the Entities that are part of the communication – Entity A. We saw in chapter 5.3 that the Event is handled so that the data received is packed into Message (PDU) according to the rules laid down by the protocol specification and then sent over the network. Entity B receives the message and after processing a response is sent back. At this stage an optional external event is generated back to the outside environment. The way the two entities that are part of the communication behave depends on the protocol and the nature of the External Event that started the flow of messages. One thing that needs to be noted is that a certain External Event changes the flow and therefore the state of the system. In this new state Internal Events can generate traffic over the network.

For example let's consider the classic client / server situation when the client needs to connect and then maintain a session until it disconnects. An External Event – let's say turn on the computer – will make the client connect. Once connected the connection is maintained by the client by sending the so called keep-alive messages every 10 seconds. In this case these Internal Events generated every 10 seconds have an impact on the generated traffic.

7.2. A User – Client – Server Model

We discussed so far about protocols and ways to model the protocols. However, one thing to keep in mind is that the protocol itself is nothing but a set of rules that governs the communication between two entities. We usually refer to these two entities as *Client* and *Server*. The *client* is the application that is used in order to access remotely available information. The information resides on another computer system known as *server*. The client-server model is used today on the Internet, where a *user* may connect to a service operating on a remote system through the internet protocol suite.

The *user*, sometimes referred as *end-user*, is a popular concept. In software engineering we use this term in order to abstract a person or a group of people that use or operate on a software application. The concept is important since the person(s) operating the software decide and dictate the behaviour of the system. The user or the user behaviour ultimately “decides” the amount of traffic sent over the network when the application in question is a *client* application interacting with a remote *server* application. Figure 19 depicts a *user-client-server* model.

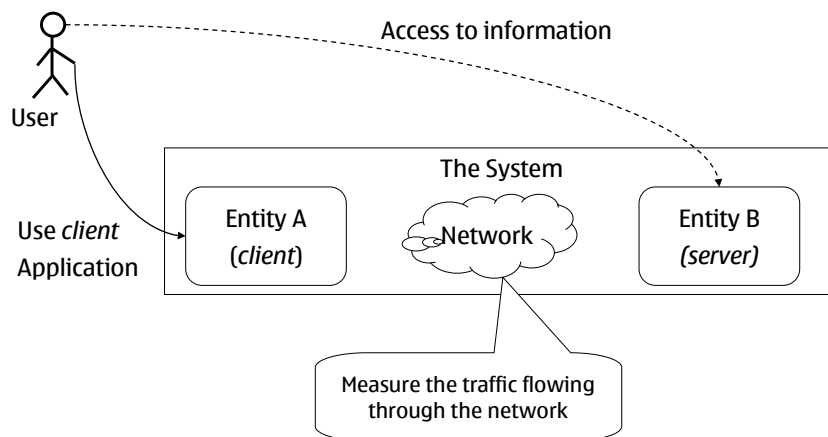


Figure 19 - A User-Client-Server Model

The model in Figure 19 is the simplest version of the User-Client-Sever model. In real life this gets more complicated depending on the type of the information residing on the server. For example, if the Entity B is a regular WEB server the model in the picture is accurate. The User will use a *client* application – WEB Browser – in order to access the WEB pages stored on the server. Generally, the server only responds to *user* requests and it does not care who is accessing the information. It is a simple procedure – Request Response. However, in some cases the access to information is controlled but this does not change much the procedure. The only difference is that the *client*

application needs to communicate more with the *server* in order to authenticate and authorize the *user*.

Figure 20 depicts a more general model. In this case the *client* and the *server* do not interact according to the *request – response* scenario only. The *server* itself can decide to send information to the *user*. The most common example of such service is Email. The Email server sends new Email messages to *user*'s Inbox and they might arrive at any time. In order for the server to be able to locate the *user* and be able to send these messages the *client* application needs to *connect* to the *server*. While this connection is available the *server* can send messages towards the *user*. In this case we talk about *connect - subscribe – notify – disconnect* scenario.

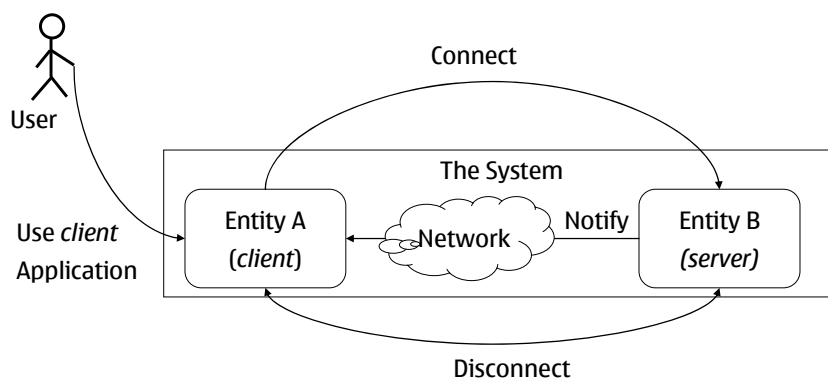


Figure 20 - Connect – Subscribe – Notify – Disconnect Paradigm

The *client* application connects to the *server* based on *user* needs. While connected the *server* pushes information to the *user*. The connection can be terminated by both the *client* at *user*'s request or *server* based on internal events.

The connect – subscribe – notify – disconnect scenario, however, does not cover the complete *user* needs. At a closer look one can notice that the user wants more than receiving information. In most of the cases the user needs to have control. That means that the system must allow the *client* application to *publish* information on the *server*. The complete list of actions that a *user* expects from a system can be seen in Figure 21. We can call this Complete Internet Service Usage Paradigm.

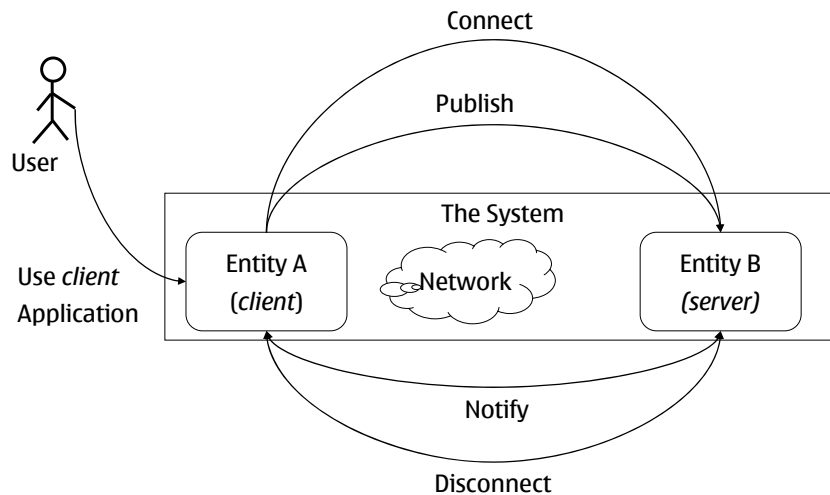


Figure 21 - Complete Internet Service Usage Paradigm

According to Figure 21 the *user* expects the following behaviour when using an Internet Service:

- Start using the *Client* application – connect to the server
- Use the *Client* User Interface to sent information to the server – publish information to the *server*
- Use the *Client* User Interface in order to read the up to date information available on the *server* – receive Notifications form the *server*
- Close the *Client* application – disconnect form the server

These are the basic expectations of a *user* of an Internet Service. One important thing to notice here is that the communication takes place over a network. The Person / *User* using this service does not care (know) what happens over this network. As a consequence, it is the *client* and *server* that need to take care of the connection while the application is running. In practice, external network events can break this connection and the *client-server* protocol needs to cope with this behaviour.

In Figure 22 we take a deeper look into the System. We see that the *client* and *server* applications are in a different state depending on the internal or external events that have occurred.

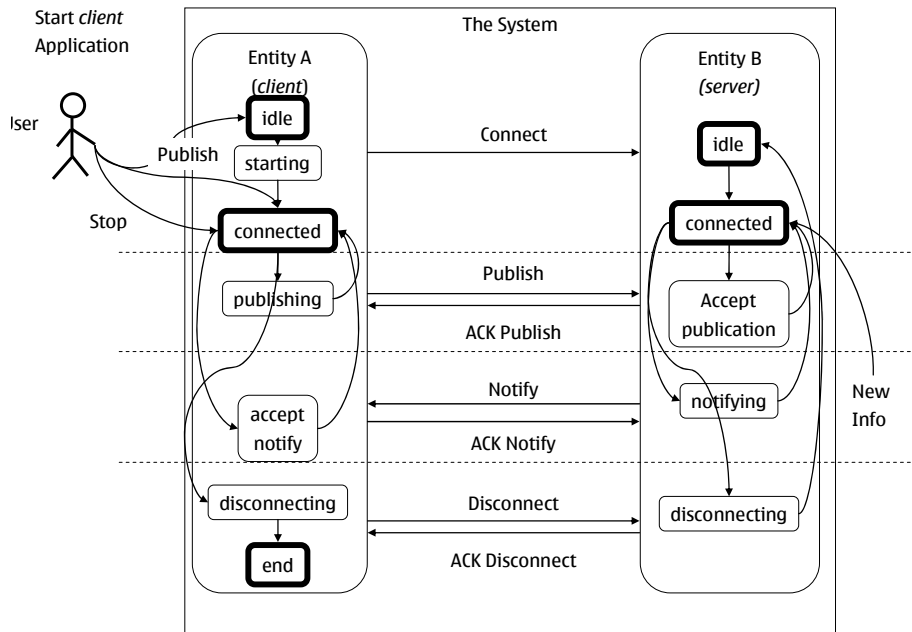


Figure 22 - System State View

In Figure 22 we notice that throughout its lifetime System reaches different states. Some of these states are more important than others when we talk about the use case. We distinguish two types of system states: (1) *stable state* and (2) *non-stable state*. The arrival of an external event is needed in order to change a *stable state*. As long as no external event arrives the system stays *stable* (Idle). An external even triggers the system. *Non-stable* states are reached. At this stage the *client* and the *server* applications communicate in order to reach again a *stable state*. *Stable* and *non-stable states* allow us to define a *use case* as the transition between two *stable states*.

When an external event arrives and triggers the system starting a *use case* it starts a chain of transitions at *client* and *server* side. This transition chain assumes a number of messages being sent over the network. The messages are sent back and forth in order to reach a (new) *stable state*. Figure 23 describes a *use case* execution.

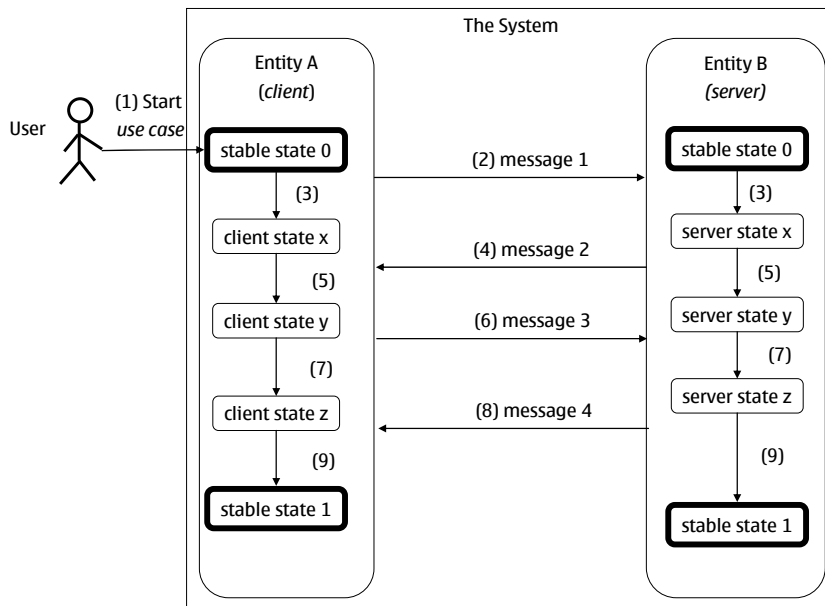


Figure 23 - Use Case Execution / Possible Example Scenario

The use case execution is started on the arrival of an external event – in Figure 23 this is a *user* request (the *user* acts on the application UI).

- 1) The *user* chooses to act on the UI of the application
- 2) *Client* applications sends message 1 to the *server*
- 3) *Client* moves to a new state and waits for a new message back from the server. The *server* moves to a new state and processes message 1.
- 4) *Server* sends message 2
- 5) *Client* moves to a new state and processes message 2. The *server* moves to a new state and waits.
- 6) *Client* sends message 3
- 7) *Client* moves to a new state and waits for message back from the server. The *server* moves to a new state and processes message 3.
- 8) *Server* sends message 4
- 9) *Client* moves to a new state and processes message 4. The *server* moves to a new state and waits.
- 10) Both *client* and *server* are now in a stable state. The *use case* is over.

A *use case* can start upon the arrival of one of the following events [Veijalainen et al., 2005]:

- The person / *user* triggers an external even – for example the user opens a WEB page in the browser
- An external event arrives at the server – for example a new email arrives
- A network event arrives – for example in a mobile network the coverage is lost, or the server notices a TCP / IP disconnection

- A time-triggered event arrives – for example the 5 o'clock event marks the end of a peak time for an office email system.

We notice that during the *use case* execution the *client* and the *server* exchange messages over the network. Our target is to measure this traffic. We need to find out the number of messages according to the *use case* and the amount of data exchanged over the network. One important thing to remember is that the arrival of an external message does not necessarily generate network traffic. These events can only generate a state change. For example, if an application running in a mobile device notices a coverage loss it will not send any messages to the server. In this case there will not be any connectivity. However, it will change its state to *disconnected*. On a subsequent event – for example coverage back in coverage – it will reconnect.

7.3. Modelling Protocols with Finite State Machines

Finite State Machines have been defined in Chapter 6.3. At that point we discussed how they are used to model protocols correctly, especially for verification and validation. We concluded that the model as it is defined now is not useful for traffic measurement. It lacked two important features.

In order to make a statement on protocol performance we need to measure two parameters. First, we need to find out the number of messages over the network. The second parameter to measure is the amount of data sent between the entities involved in communication. Both parameters are measured in the context of the so called *use case* execution.

In order to improve the Finite State Machine model to be suitable for our purpose and application domain we need to:

- Model the *use case* execution
- Define how the number of messages is measured
- Define how the amount of data is measured

Section 7.3.1 that follows attempts these new definitions and coins new term to accommodate these concepts, that of a Finite State Machine (FSM).

7.4. Modelling Use Case with Finite State Machines

In order to start this definition let us recall what a Finite State Machine / Protocol are. The definition can be found in chapter 6.3. There is, however, a need to customize and at the same time extend, the modelling power of this automata family for understanding the concepts of states and transitions as these exist with the domain constraints in the field of our application domain – mobile communications. Similar work in different and similar applications can also be found in (Rolland and Richard, 1983; Fencott, 1996)

Definition: A protocol Finite State Protocol (FSP) is a quadruple

$$FSP = (S, M, \Sigma, O, \tau)$$

- (i) $S = (S_1, S_2)$, two disjoint finite sets (alphabets of transitions) representing the states of processes P_1 and P_2 respectively
- (ii) $M = (M_{12}, M_{21})$, two finite sets where M_{12} represent the messages that can be sent from P_1 to P_2 and M_{21} represent the messages that can be sent from P_2 to P_1
- (iii) $\Sigma = (\Sigma_1, \Sigma_2)$, two finite sets on M that contain tuples of elements of the following forms and types
 - for every message $\forall \chi \in M_{ij}$, the sending of message χ is denoted by $-\chi$. Every sent element $-\chi$ is an element of Σ_i
 - for every message $\forall \chi \in M_{ji}$, the sending of message χ is denoted by $+\chi$. Every sent element $+\chi$ is an element of Σ_j
 - λ is an empty string of event or no event $\lambda = \{ \}$ or λ is an element of Σ_i .
- (iv) $O = (o_1, o_2)$, where $o_1 \in S_1$ and $o_2 \in S_2$. o_1 and o_2 are the initial states for processes P_1 and P_2 respectively
- (v) τ is a transition function defined as follows: $S_i \times \Sigma_i \rightarrow S_i$, $i = 1, 2$. The transition function for an event $\sigma \in \Sigma_1$ at state s can be written as $\tau(s, \sigma)$. It represents the next state reached after triggering the transition σ at s

Definition: A channel C_{ij} is a First-In-First-Out (FIFO) queue connecting process i and process j . The contents of C_{ij} is marked c_{ij} . This is a string of symbols form M_{ij} ($c_{ij} \in M_{ij}$) and represents the queue of messages sent from i to j .

As we discussed earlier the first step is to model the *use case* using FSP. For that, we start with defining a new concept called *stable state*.

Definition: A system state is a pair $\langle S, C \rangle$ where $S = (s_1, s_2)$ with $s_1 \in S_1$ and $s_2 \in S_2$. s_1 and s_2 represent the current state of processes P_1 and P_2 respectively. $C = (c_{12}, c_{21})$ represent the current content of the communication channels c_{12} and c_{21} .

The formal definition above describes a system state where the communication entities reached certain internal states and the communication channels contain (or not) messages. It is now obvious that if there are any messages in any of the channels one of the tow entities needs to process those messages. This is most likely to take the system to a forward state. New states will be reached and eventually new messages will be sent. Thus the system cannot be considered stable; hence the following definition.

Definition: A stable state is a pair (s_n, s_m) derived from a system state $\langle S, C \rangle$ where $S = (s_1, s_2)$ with $s_1 \in S_1$ and $s_2 \in S_2$. s_1 and s_2 represent the current state of processes P_1 and P_2 respectively. The communication channels $C = (c_{12}, c_{21})$ are empty.

In the above definition we define the stable state as that state that is reached when both the communication channels are empty. That means that there are no messages to be sent or received. At this point only an external event can tip the system again and trigger a sequence of messages being sent and / or received.

We define the following concepts.

Definition: We say that two states $s_n \in S_i, s_m \in S_i$ are consecutive in P_i if $\exists \chi \in \Sigma_i$ and $s_m = \tau(s_n, \chi)$.

The formal definition expresses formally that two states are consecutive if there is a message received or sent that moves the process from one state to the other.

Definition: We say that a state $s_m \in S_i$ is reachable from state $s_n \in S_i$ in P_i if:

there is $s_1, \dots, s_k, s_1, \dots, s_k \in S_i$ and s_i and $s_{i+1}, 1 \leq i < k$, are consecutive in P_i

- s_n and s_1 are consecutive in P_i
- s_k and s_m are consecutive in P_i

Definition: A system state $\langle S', C' \rangle$ is reachable from $\langle S, C \rangle$ if for $S = (s_1, s_2)$ with $s_1 \in S_1, s_2 \in S_2$ and $S' = (s'_1, s'_2)$ with $s'_1 \in S', s'_2 \in S'$ Then s'_1 is reachable from s_1 in P_1 and s'_2 is reachable from s_2 in P_2

Definition: Let a system state be $\langle S', C' \rangle, S' = (s'_n, s'_m)$ with $s'_n \in S', s'_m \in S'$ and let a system state be $\langle S, C \rangle, S = (s_n, s_m)$ with $s_n \in S, s_m \in S$. Then $\langle S', C' \rangle$ is reachable from $\langle S, C \rangle$ and both $\langle S', C' \rangle$ and $\langle S, C \rangle$ are stable states. We call *use case execution* in P_i the sequence $s_n, s_1, \dots, s_k, s_m, s_1, \dots, s_k \in S_i$ and:

s_n and s_1 are consecutive states

s_k and s_m are consecutive states

One note to keep in mind is that a *use case* can be executed from one stable state to the same stable state. For example if we consider the Publishing *use case* in Figure 22 the *execution* goes: Connected state – Publishing State – Connected State. This, in FSM terminology is known as a transition from a state to itself.

Above I modelled the *use case* concept. The next step is to define a way to calculate the number of messages sent and received in a *use case* execution. This can be easily if we consider the following definition and subsequent formula.

Definition: The number of messages sent and received by process P_i in the *use case executions* $s_1, \dots, s_k, s_1, \dots, s_k \in S_i$ is given by the following formula:

$$\sum_1^k \chi * \mathcal{G}; \chi \in \Sigma_i \text{ and } s_{i+1} = \tau(s_i, \chi)$$

$\mathcal{G} = 0$ if χ is an internal event
 $\mathcal{G} = 1$ if χ is an external event

The next problem to solve is calculating the data amount sent over the networks. Having already a formula for the number of messages the process of calculation becomes easy. The only thing we need to do is to quantify the amount of data carried over the network by each message. For that we define the following function.

Definition: θ is a size function on Σ_i . $\theta : \Sigma_i \rightarrow \mathbb{N}$ and $i = 1, 2, \forall \chi \in \Sigma_i, \theta(\chi) =$ the size of message. For internal events represented by λ the size is 0.

According to the definition above we can calculate the amount of data sent over the network with the following formula:

$$\sum_1^k \theta(\chi); \chi \in \Sigma_i \text{ and } s_{i+1} = \tau(s_i, \chi) \text{ and } \theta \text{ is a size function on } \Sigma_i$$

We are now able to use an enriched extension of FSMs, FSPs in order model and measure the amount of network traffic – number of messages and amount of data. However, there is one detail that makes this model hard to use. One needs to define the size function in order to be able to measure the amount of data being exchanged.

7.5. Modelling Protocols with Petri Nets and other Related Work

Another solution for protocol performance measurement could be based on Petri Nets. We saw in chapter 6.4 that protocols can be designed using this modelling tool. We found, however, the same limitation regarding FSPs. It is not straight forward to measure the number of messages and the amount of data over the network. In this chapter we discuss a particular type of Petri Net – Coloured Petri Nets. We only describe the model in the next section 7.4.1 in order to remind the reader similarities in modelling power and expressability to the modelling approach that we defined earlier, the FSP.

7.5.1. Coloured Petri Nets

Coloured Petri Nets (CP-nets) is an extension of the Petri Nets [Jensen, 1994]. It is an executable and formal specification method that provides graphical tools for specification and analysis of dynamic system. It is mostly used in areas such communication protocol modelling, work flow analysis and embedded systems.

The CP-nets combine the strong formalism of Petri nets with the power of programming languages. CP-nets as regular Petri nets allow the specification of system processes. On top of that the programming languages can be used to do data manipulation.

A small example of a CP-net is shown in Figure 24. It shows a simple transport protocol based on the model depicted in Figure 22. The Petri Nets *places* are marked as ellipses and are used to describe the state of the system. The *transitions* are marked as *rectangles* and are used in order to describe actions. The *arcs* are marked by arrows and the *arc expressions* describe how the state of the CP-net changes when the transitions occur.

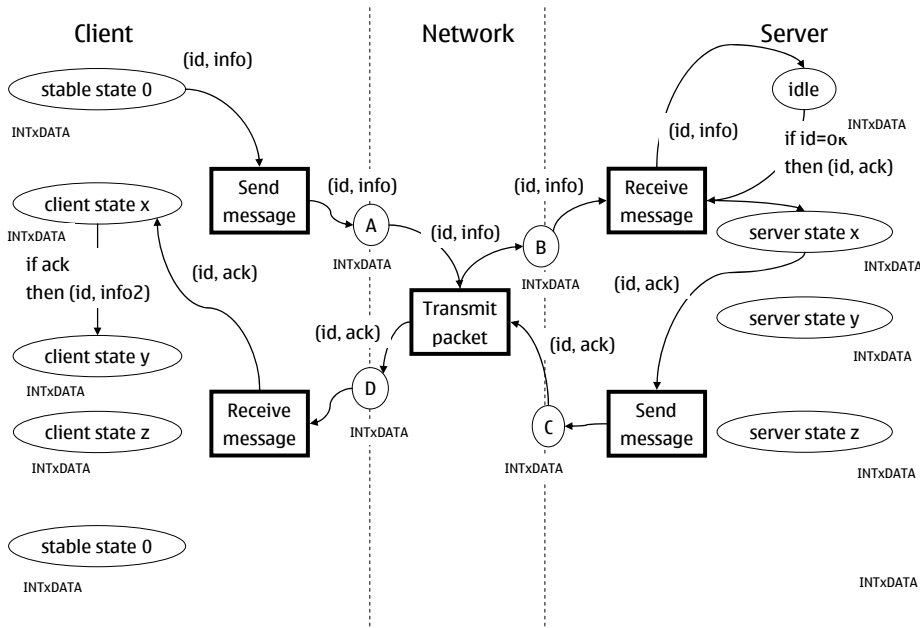


Figure 24 - Simple example of Coloured Petri Nets utilized in a network

Like in regular Petri Nets the *places* contain *tokens*. While in Petri Nets the *tokens* do not have any meaning in CP-nets they carry data values. The values belong to a given type. For example all the *places* may contain *tokens* belonging to integer and string and they form a Cartesian data (INTxDATA).

In the initial marking there is only one token – (id, info) – in the *stable state 0*. All the other states are empty (Figure 24). Because of the *token* in *stable state 0* the CP-net will start *firing transactions*. The following execution sequence (similar to configuration sequence in FSMs) takes place up until the CP-Net reaches *client state y*:

- *Send Message* fires and moves (id, info) to A
- *Transmit packet* fires and moves (id, info) to B – the message is sent across the network
- *Receive Message* fires and moves (id, info) to *idle* state on the server

- In *idle* state the server processes the messages, generates the answer (id, ack) and moves to *server state x*
- *Send message* fires and moves (id, ack) in *C*
- *Transmit packet* fires and moves (id, ack) in *D*
- *Receive Message* fires and moves (id, ack) to *client state x*

At this stage the client processes the message, generates an answer and moves to *client state y*.

From this point onwards the communication continues according to the same rules. When a *token* is available the enabled *transactions* fire and move it to a new *place*.

Petri Nets can be used to model communication protocol. The CP-nets are an even more expressive tool for this. The possibility of manipulation data and control gives them a strength that no other model gives except possibly some forms of FSM [Bavan et al., 1997; Berki 2001]. More information on how CP-Nets can be used in order to model communication protocols can be found in [Kristensen et. al., 1998].

The level-5 protocols we are studying in this paper are one of these cases of not too complex protocols. In Figure 9 it can be seen that the Internet protocols work according to a very simple pattern – request – indication – response –confirmation. In this case the FSPs and especially the representation that we introduce in the next sub-chapter 8 are more suitable tools for our domain needs and design purposes.

7.6. A Graphical Representation for Finite State Protocols

We agreed to use the FSP Model in order to model a Client Server environment. The model is used for protocol performance measurements. However, the graphical representation we used in Figure 13 is not suitable anymore for those cases when we need to represent real life protocols. An alternative to that representation is based on the so called *Time Line* representation of a system. Figure 27 shows an alternative representation of Figure 13 using *Time Line* graphical model, notations and semantics.

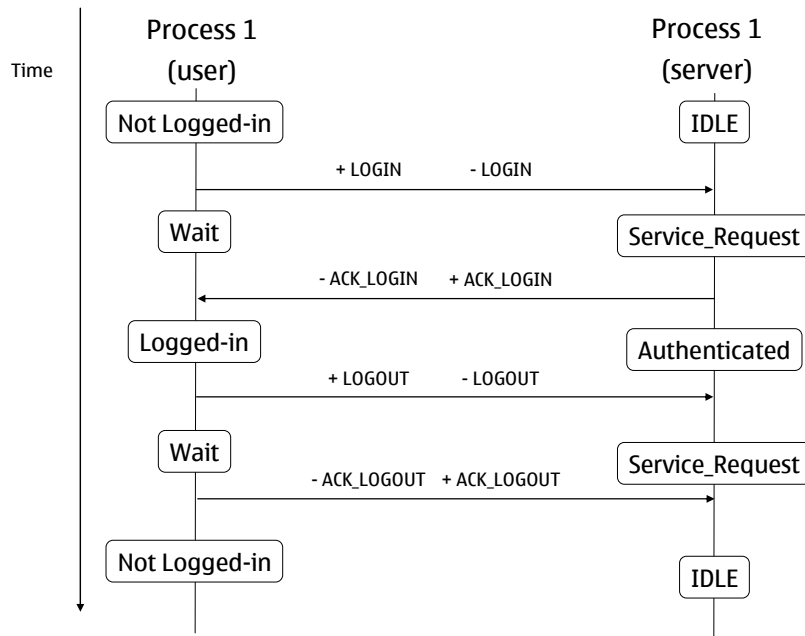


Figure 25 - Time Line Graphical Model for FSPs

The model in Figure 25 is more suitable for protocol representation. It allows us to put the execution of the *use cases* into time perspective. If needed the *Time Line* model can be used to revert back to the Finite State machine representation. This can happen for example when there is a need for testing. The FSM syntax and semantics assist the natural FSM-based testing procedure [Berki 2001]. However, the Time Line Graphical Model assists understanding and flow readability, providing a holistic, simple, but not accurate picture of the system and the communication that is generated with it.

The *Time Line* graphical representation is used in chapter 8 in order to model the Presence Protocols.

8. Internet Presence – A Case Study

In this chapter we put the Finite State Protocol model at work. We measure the performance of one of the so called *Presence* protocols. In computer and telecommunications networks, *presence information* conveys user's availability and willingness to communicate. *Presence information* is published to other systems' users—sometimes called watchers or subscribers—to convey its communication state.

Presence standards are defined by many different working groups in the internet industry. Some of them are more advanced than the others. Probably the most famous of them all is the IETF specification – RFC 2778 [RFC2778, 2000]. The document describes the abstract model for Presence and Instant Messaging. It constitutes also the base work to be considered for future *Presence* protocol developments. It provides basic definitions of entities, terminology to be used for mark-up for presence and instant messaging. For enhancing the thesis readability in the next paragraph, section 8.1, we take a look at some definitions present in this specification.

8.1. A Presence Data Model

As a one-line definition Presence is the ability and willingness of a user to communicate across a various set of services and devices. The following paragraphs will bring into the picture a few concepts that will be further used in this thesis.

SIMPLE working group defined a Presence Data Model. This model is defined so that it fulfils the requirements for Presence outlined in RFC 2778.

A communication system for interaction in-between users will be called **SERVICE**. It will provide certain modalities and content. Instant Messaging is one common service that we will refer a lot.

A **DEVICE** is a physical component the user interacts with in order to make and receive communications. The state of the device might enable or prevent a user to communicate, hence it affects the availability and willingness we discussed previously.

The central piece we are mostly interested in is the **PERSON**. This is the end-user with purpose for Presence and it is characterized by states that impact on her/his capabilities to communicate.

Figure 26 depicts the abstraction of the presence service. The main message is that a PERSON subscribes to services that run in devices. For example, a user can own a mobile phone and a personal computer. The mobile phone runs a Mobile Operator Push to Talk Client and an Instant Messaging client. The personal computer runs an Internet Instant Messaging (IM) client and a Mobile Operator Instant Messaging client.

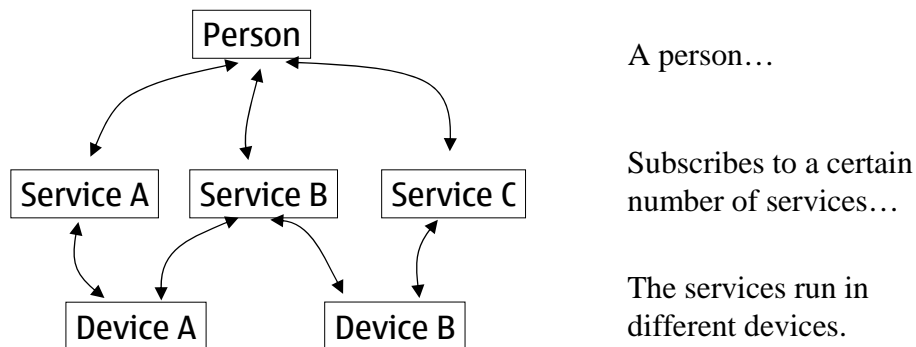


Figure 26 - A Data Model for Presence

8.2. Presence Service Architecture

Presence Service accepts presence information, stores it and then distributes it. However, it is extremely important to understand how the Presence Information is created and how it is published / made available for others. Another important aspect is how to give access to the Presence Information already available on the server.

The following definitions are given by RFC 2778.

Presentity: is the client that is creating the presence information on behalf of the user and provides it to the presence service for storage and distribution.

Watcher: is the client receiving and then consuming the presence information stored and distributed by the presence service.

Both presentities and watchers will interact with the presence service via User Agents (UA); hence, there will be Presentity UA and Watcher UA.

The Presentity will **publish** the presence information and the watchers will **subscribe** to the presence service for receiving it whenever this changes.

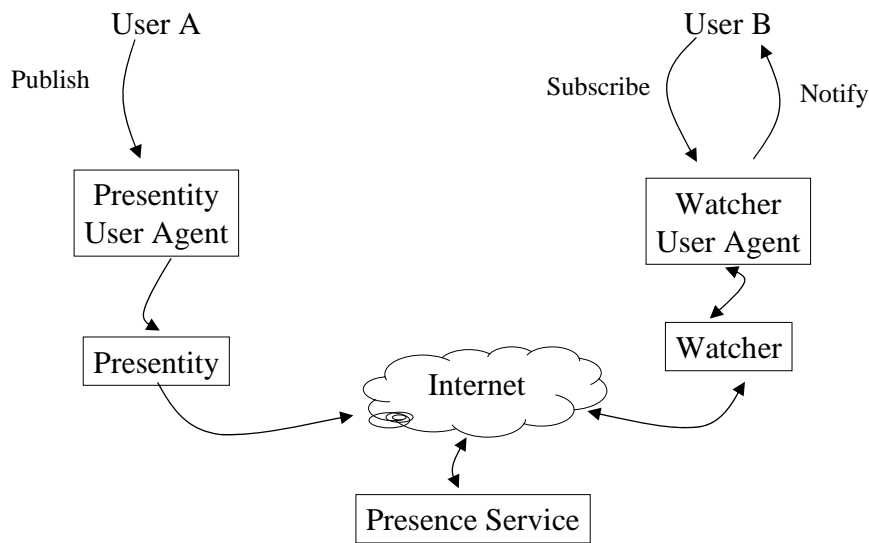


Figure 27 - Presence Service System Architecture

8.3. Standard Definitions

The following terms are introduced as they are defined in the standards documents. The reader must be familiar with them in order to have a better understanding of the standard specifications and the common language used by SIMPLE technologies. Further information can be found in [RFC4479, 2006].

Device: A device models the physical environment in which services manifest themselves for users. Devices have characteristics that are useful in allowing a user to make a choice about which communications service to use.

Service: A service models a form of communications that can be used to interact with the user.

Person: A person models the human user and their states that are relevant to presence systems.

Occurrence: A single description of a particular service, a particular device or a person. There may be multiple occurrences for a particular service or device, or multiple person occurrences in a document, in cases where there is ambiguity that is best resolved by the watcher.

Presentity: A Presentity combines devices, services and person information for a complete picture of a user's presence status on the network.

Presentity URI: A Uniform Resource Identifier (URI) that acts as a unique identifier for a Presentity, and provides a handle for obtaining presence information about that Presentity.

Data Component: One of the device, service, or person parts of a presence document.

Status: Generally dynamic information about a service, person or device.

Characteristics: Generally static information about a service, person or device. Useful in providing context that identifies the service or device as different from another service or device.

Attribute: A status or characteristic. It represents a single piece of presence information.

Presence Attribute: A synonym for attribute.

Composition: The act of combining a set of presence and event data about a Presentity into a coherent picture of the state of that Presentity.

Presence Document: The collection of presence and event data about a Presentity. This collection is represented in XML format. Different standard specifications use different XML formats.

8.4. Presence Deployment - Example

The previous section exposed an abstraction of the Presence Service. Figure 28 shows a practical deployment. This is just an example.

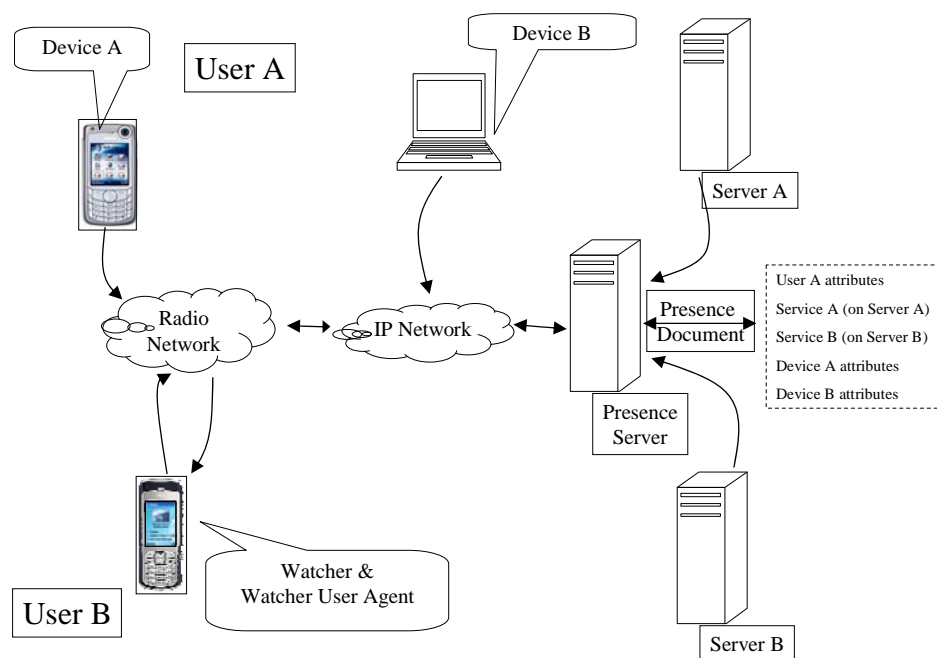


Figure 28 - Presence Service Deployment

Figure 28 above illustrates an example of a User A that owns a mobile phone and a laptop using three different services: the Presence Service, Service A and Service B. The Presence Server manages a Presence Document containing presence information

describing User A availability and willingness to communicate over Service A and Service B. In practice, the presence information is managed directly by the user itself; the user presence attributes are modified as a result of user's action. The device and service dependent attributes are modified indirectly by the device or the service as the reactions to certain user actions (e.g. the device becomes unavailable when the user switches it off or the service become available when the user logs in)

8.5. Presence Traffic Management

For each user (Person) a Presence Document (generally XML document) is stored on the Presence Server. For each such Presence Document a set of access rules is also defined and stored as well. Both, the Presence Document and access rule are managed by the user or by other entities on behalf of the user (e.g. the device will publish its network connectivity parameters or the IM server will publish the availability for IM communication). Other users have the possibility to ask for the presence information (Subscribe) and according to the access rules mentioned above they will receive the information that notifies Presence or not. In case that they are entitled to receive it the presence information will be transferred from the server to the clients in the form of notification (Notify).

Presence information management, access rules management, subscriptions and notifications are done by sending messages to and receiving message from the server respectively. Messages are sent to the server in order to PUBLISH, SUBSCRIBE or manage access rules. The server sends messages in order to convey presence information changes and inform on access rules management. The actual size of these messages is a major concern at the moment. Most of these messages “eat” resources hence they are expensive. Still, it is hard for the operator of the Presence Service to charge them separately. While charging models are yet to be defined in order to deal with this concern there is a need to address this in other ways too. Standard specifications deal with this to some extent. Implementation details also provide a suitable opportunity to resolve this issue for the particular group of interest.

8.6. A FSP Time Line Model for Presence

An abstract Time Line Model for a Presence Service is introduced in Figure 29. The ideal case shows us a *Presence Client* connected to a *Presence Service*. The Presence Information is published. Subscription for updates on other entities Presence Information is started on the server. Starting from the Idle State the system is trying to reach its ideal state – Connected & Published & Subscribed. An internal or external event can change its state to anything than the ideal state. In this case the system is again aiming at the ideal state.

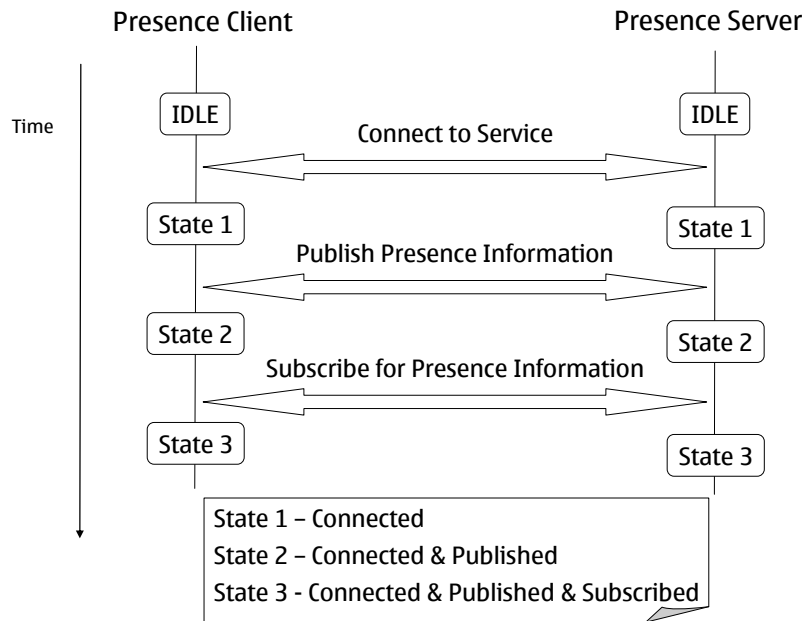


Figure 29 - Abstract Time Line Model for Presence

In the *Connected* state the user is authenticated and authorized to use Presence Services. The next step is to *Publish* its own information. This information is made available upon request for other users. The final step is to *Subscribe* for Presence Information Updates. We distinguish already two different *use cases*. First is the Publication of Presence Information. Second is the Subscription for Presence Information updates. One can consider a third *use case* – Connection to Presence Service. This third use case is considered separately since none of the first two can actually happen without a connection to Presence Service. In the following chapter we will base all our protocol performance measurements on these 2 main *use cases* – Publish and Subscribe.

In the Following chapter we take a look at the protocols for Presence defined by SIMPLE Working Group in IETF.

8.7. The SIMPLE Model for Presence

The Simple Working Group aims at developing a standard for Presence and Instant Messaging based on SIP. The requirements driving this work are stated in another IETF specification – RFC 2778 [RFC2778, 2002]. At the same time, other working groups develop specifications for Presence and Instant Messaging use. Our aim here is to develop a model that allows us to measure the performance of the SIMPLE protocols.

In Figures 30, 31, 32 we show how the abstract model presented in Figure 29 is implemented based on SIMPLE specifications.

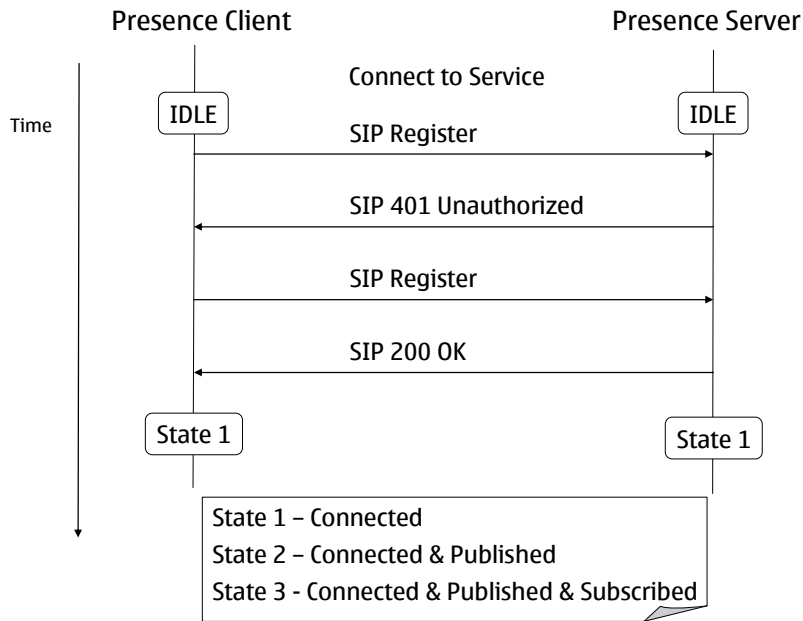


Figure 30 – SIMPLE Connection to Presence Service

The connection to Presence Service for the SIMPLE case is based on the so called SIP Registration Procedure. The client sends an initial SIP Register Request to the server. In order to authenticate the request the Server generates a *challenge* back to the client in a SIP 401 Unauthorized response. The client calculates the response to the *challenge* and sends a new SIP Register request containing the information requested by the server at the previous step. The server matches the client credentials to the internal database and authenticates the client (Figure 30).

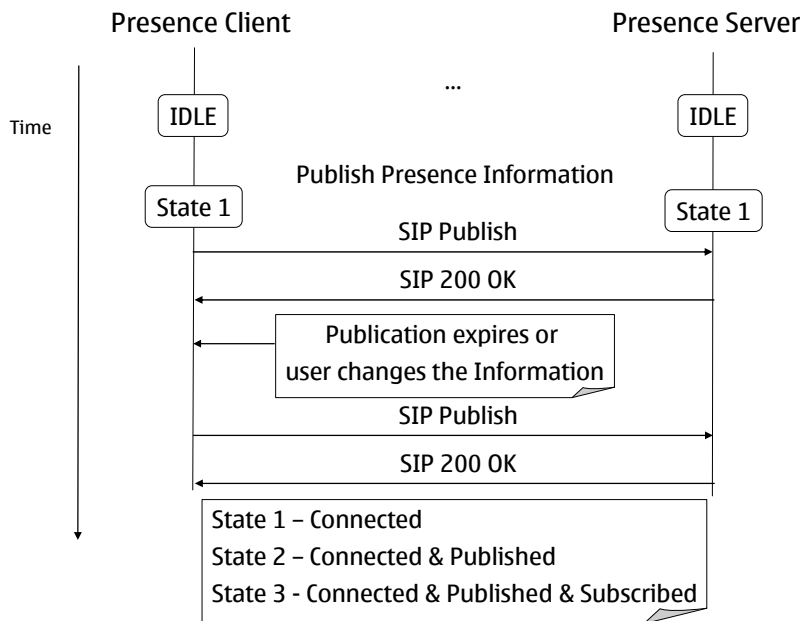


Figure 31 – SIMPLE Presence Publication

The Presence Information Publication is handled using SIP Publish [RFC3903, 2004]. The User Interface allows the input of human readable information that is subsequently carried over the Internet to the Presence Server wrapped in SIP PUBLISH requests (Figure 31). The actual payload is XML and it conforms to formats specified in [RFC3863, 2004]. The information available on the server changes as a result of two external events. One, and the most obvious, is the need to change due to user interaction – the user decides to change the information. The other event is the so called expiry timeout. According to SIMPLE specifications a set of information is Published as the so called Soft-State. That means that after a certain timeout the information expires. In order to keep the information on the server the client needs to perform the so called *Refresh* function. In practice, we are talking about a regular SIP PUBLISH Request similar to the initial publication. The difference is that the latter one does not include the full content but only enough information (ETag information) to identify the published information that needs to be refreshed (Figure 31).

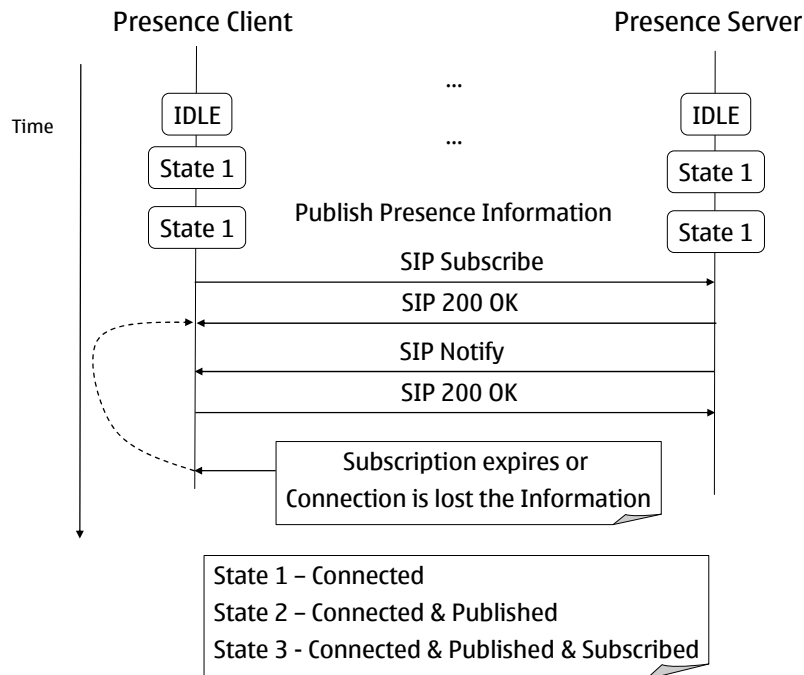


Figure 32 – SIMPLE Presence Subscription

The SIMPLE Subscription for Presence Information happens in two steps [RFC4840, 2006]. First the client sends SIP SUBSCRIBE Requests which are acknowledged by the server. Then, in a separate SIP NOTIFY the server sends the requested Presence Information. The Client is acknowledging it. The same expiring concept applies in here. The client needs to *refresh* the subscription whenever this expires (Figure 32).

The SIP traffic is going generally over UDP. That means that the client and the server do not have an ongoing connection. They both know each other's IP address and they also know the IP ports where datagrams should be sent. The Client knows the

server IP address from a local settings database. The server obtains the Client IP in the Registration procedure (see Figure 30). In some cases, mobile networks in particular, the client loses the IP address for some of time. In such cases a subscription refresh is also required in order to ensure that the most recent presence information is available.

8.8. A Presence Use Case

We discussed in the previous chapter about the two main Presence *use cases* – Publication and Subscription. In order to measure the performance of the SIMPLE protocol we need to define the following parameters:

- Presence Information – what kind of information is published / subscribed
- Subscription Life Time (SLT) –the time (expressed in seconds) of the session that a user remains connected
- Presence Updates per Hour (PUH) – the number of times the user will change the presence information in an hour
- Presentities (PRES) - Number of entities (presentities) the user subscribes for
- Refresh Interval (RI) – the time interval (expressed in seconds) that the client needs to refresh its subscriptions

In addition to the parameters above we also define the following variables. They express the size in bytes of the messages exchanges between the client and the server and they are calculated based on the type of Presence Information being exchanged. In practice they are calculated based on the size of the so called presence document published by a Presence client. An example is introduced in Appendix A.

- Publish Message Size – the size of the Presence document
- Notify Message Size – the size of the Presence document multiplied by the number of entities subscribed for
- Other SIP messages sizes are according to the SIP specification

Based on the parameters above we can calculate the number of messages based on the following formula

$$(Initial_publish + initial_publish_response) + (PUH * SLT) * (publish + publish_response) + (terminate_publish + terminate_publish_response);$$

For calculating the amount of data we need to quantify each and every one of the messages that are part of the formula. The *initial_publish* is the messages containing the initial Presence Document to be published. Its size can be considered the same as the *publish* message sent every time the information needs to be changed. The size of *initial_publish_response*, *publish_response*, *terminate_publish* and *terminate_publish_response* are according to SIP specifications. Examples can be found in Appendix A.

The same approach can be used in order to calculate the number of messages for the Subscription *use case*. The following formula is computed based on the model shown in Figure 32.

$$\begin{aligned}
 & (initial_subscribe + initial_subscribe_resp + initial_notify + initial_notify_resp) \\
 & + \\
 & SLT * PUH * PRES * (notify + notify_resp) + \\
 & (SLT / RI) * (refresh_subscribe + refresh_subscribe_resp + refresh_notify + \\
 & refresh_notify_resp) + \\
 & (terminate_subscribe + terminate_subscribe_resp + terminate_notify + \\
 & terminate_notify_resp)
 \end{aligned}$$

The amount of data is computed based on the size of the messages sent between the client and server. For example the *notify* messages contain the presence document, hence they depend on the type of the information published. All the other messages are according to the SIP specifications. Examples can be found in Appendix A.

Table 4 shows how the amount of messages is affected by different *use case* parameters. Use Case 1 shows a calculation over 1 day. Use Case 2 shows a more realistic case – 1 month. However, the refresh interval is not realistic at all and in Use Case 3 we can see the effects of increasing the Refresh Interval to 12 hours. Use Case – UC 4 - shows the impact of the number of presentities on the number of messages. The last Use Case – UC 5 – shows the impact of a higher rate of Presence Information updates.

Table 4 - Number of message and amount of data over the network

	UC 1	UC2	UC 3	UC 4	UC 5
Subscription LifeTime	86400 (1 day)	62208000 (1 month)	62208000	62208000	62208000
Refresh Interval (hours)	7200	7200	43200	43200	43200
Presence Changes (per hour)	1	1	1	1	2
Presences per watcher	10	10	10	20	20
Number of Messages	588	17292	16092	30492	60732
Amount of data (Mb)	1	31.65	30.11	56.87	113.43

8.9. Improving the SIMPLE protocol

One can only start thinking about improvements based on the results computed and presented in section 8.8. The model presented in Figures 30, 31, 32 can be analyzed. In fact, Niemi in his Internet draft - An Extension to Session Initiation Protocol (SIP) Events for Issuing Conditional Subscriptions [NIEMI. 2006] – describes such improvement. The idea is to limit the number of notifications. We notice by looking at the Subscribe model that in case of a timeout expiry a full notification is sent back to the client. This is, sometimes, useless since the information on the server is the same as what the client has stored locally. This refresh happens only due to the timeout and not due to a real need.

A real need to refresh appears when the client loses the connectivity for a short amount of time. When the connectivity is back a subscription refresh is needed. However, not in all cases the server has new information to send. In these cases a full notification is again not needed. The IETF draft proposed the suppression of some unneeded notifications. The formula for calculating the messages for the Publication Use Case does not change. However the Subscription use case is affected. Table 5 shows the new values according to the following formula for the Subscription *use case*.

$$\begin{aligned}
& (initial_subscribe + initial_subscribe_resp + initial_notify + initial_notify_resp) \\
& + \\
& SLT * PUH * PRES * (notify + notify_resp) + \\
& (SLT / RI) * (refresh_subscribe + refresh_subscribe_resp) + \\
& (terminate_subscribe + terminate_subscribe_resp)
\end{aligned}$$

Table 5 - Number of message and amount of data over the network

	UC 1	UC2	UC 3	UC 4	UC 5
Subscription LifeTime	86400 (1 day)	62208000 (1 month)	62208000	62208000	62208000
Refresh Interval (hours)	7200	7200	43200	43200	43200
Presence Changes (per hour)	1	1	1	1	2
Presentities per watcher	10	10	10	20	20
Number of Messages	564	16572	15792	39372	60612
Amount of data (Mb)	0.41	11.9	11.5	21.64	43.19

At the first glance and by comparing these results with the ones in table 4 we notice that the improvement on the number of messages is not impressive. However, the amount of data over the network is cut in half by eliminating these “not really needed” messages.

9. Conclusion and Future Development Plans

Second Generation (2G) mobile networks and the GSM networks in particular, have been very successful over the past years. The panorama is, however, changing rapidly because of new needs and challenges. New directions are shaped by the business requirements optimization needs for profitable investments. At the same time there are new trends and services that need to be standardized as major technical opportunities. Things become more complex when some of the already standardized technologies are becoming available for Mobile Telecommunications. One such example is the IETF, which for years focused on the fixed Internet and now it comes into the mobile environment. These trends and expectations make it hard for the operators to choose a suitable to their needs evolving path. It seems that, with respect to mobile network environments, there exists a past, rigid and not suitable to all stakeholders needs, mobile design technology. This thesis focuses on identifying proven design strengths that can successfully be used in communication protocol design for mobile use.

IETF protocols and especially the level-5 ones are becoming the *de facto* protocols for mobile services. While some of them seem to be the driving force for the service evolution already some still have a long way to prove their usefulness until they are ready for mobile use. A significant issue that most of the new technologies need to solve is the performance or efficiency evaluation. In the mobile networks the way a protocol behaves affects the behaviour of the network itself. Moreover, the way that the communication channels are allocated depends to some extent on the characteristics of the data. For example, a client sending its requests as part of very short data packages will benefit from using the so called signalling channel saving bandwidth and power resources.

One important aspect to be taken into consideration by the mobile operators is the ability to plan and manage the behaviour of their network. Consequently they need to understand how a certain technology works before choosing it for running one of their services. This potentially improves the performance and quality of their network and has in impact on the operational costs. Careful planning for services dissemination platform could potentially bring improvements in customer relationship management due to increase of user satisfaction and service usability.

IETF level-5 protocols invaded our lives during the past decades through various different applications. In the desktop environment for instance the applications use one of these protocols in order to communicate with a server. Protocols are now becoming more visible in the mobile communication. SIP especially, is a very good option for setting up communication sessions. Other protocols are still to prove themselves as suitable for mobile usability. The model proposed in this thesis assists in addressing the needs for dynamic modelling (see also Sol, 1992; Tardieu, 1992), analyzing and

understanding the level of performance one protocol offers. It also allows to bring about improvements through measurements and testing (see e.g. Phalp and Shepperd, 1994).

After looking at the available techniques for protocol modelling we chose two of them for deeper analysis – Finite State Machines and Petri Nets. In their original form they did not suit to all our needs. Although they are both suitable for protocol validation and verification, none of them was capable to deliver protocol measurements. For Petri Nets, an extension called Coloured Petri Nets is available. This can be used for protocol measurements. However, the technique is difficult to grasp and will probably alienate various stakeholders. An improvement of the Finite State Machines has been proposed and chosen as modelling tool for capturing performance measurements. This is in fact the main contribution of the thesis.

The FSP model for protocol performance measurement is an extension of the FSP model described in literature. It has been enhanced in order to handle use cases in mobile communications. This use case can then be measured. The measurement results can be collected and analyzed so two different protocols suitable for the same purpose can be compared using the same criteria. This improvement allows a smart choice of a communication technology. The case study in chapter 8 comes to support this statement. The thesis realized how the SIMPLE protocol can be analyzed and measured. The thesis also showed that an improvement proposal delivers enhanced performance. In consequence the new and improved model can be used for proving certain performance aspects of the level-5 protocols.

9.1. Future Work

The performance measurements need to address two aspects. First is the number of messages sent over the network. The second is the amount of data exchanged in the communication. The FSP model proposed by this work is able to handle well the first aspect, especially the graphical representation proposed in chapter 7.5. However, the amount of data aspect is not handled in an elegant way. The user of the model needs to read the protocol specifications and define the *size* function defined in chapter 7.3.1. In practice this means that it is not easy to analyze how the size of the content transported by the protocol affects its performance. An improved model to deal with this aspect is needed. Coloured Petri Nets are a good candidate for this particular aspect, and their modelling power will be investigated in further research.

Coloured Petri Nets are also a direction that needs more consideration. Their ability to use Programming languages in order to describe their behaviour makes them an ideal candidate for protocol simulations. At the same time the so called coloured tokens can be used in order to model the protocol message size – something the FSP model lacks. On the other hand extended versions of the general automata, such as X-machines, provide computability and, therefore direct implementability and testing, which are desirable properties in the dynamic application domain of communication protocols.

Last but not least it is for the future work to do the actual protocol analysis. One concrete work item can be to compare to the SIMPLE protocol already addresses in this work with its competitor technology – XMPP. They are both IETF technologies and they are both considered for the Presence use case. The actual outcome will probably not deliver a message like “X is better than Y” but it will, for sure, deliver the needed information to make an informed and knowledgeable choice.

References

- [Balanescu et al., 2000] Balanescu, T., Cowling, A. J., Georgescu, H., Gheorghe, M., Holcombe, M. and Vertan, C., Communicating stream x-machines systems are no more than x-machines, Internal Report, Dept. of Computer Science, University of Sheffield, 2000
- [Barnard, 1998] Barnard, J., COMX: a design methodology using communicating X-Machines, Information and Software Technology 40, 271-280, Elsevier 1998
- [Barnard and Woodward, 1996] Barnard, J. & Woodward, M., Communicating X-Machines, Information and Software Technology 38, 401-407, Elsevier, 1996
- [Bavan et al., 1997] Bavan, S., Berki, E., Georgiadou, E., Milankovic-Atkinson, M. & Walker, M. (1997). Towards a Formal Specification of an Object-Oriented Architecture for Parallel Computing". Arabnia, H. R. (Ed.) *PDPTA Proceedings of the International Conference for Parallel and Distributed Processing Techniques and Applications, PDPTA 1997*, Jun 30-Jul 3, 1997, Las Vegas, Nevada, USA. CSREA Press, pp. 1054-1060, Volume II, ISBN: 0-9648666-8-4.
- [Berki et al., 2007] Berki, E., Isomaki, H., Salminen, A. (2007). Quality and Trust Relationships in Software Process Improvement. Berki, E., Nummenmaa, J., Staples, G., Ross, M. (Eds). (2007). "*Software Quality meets Work-Life Quality*". Conference Proceedings of the Software Quality Management (SQM) International conference 2-4 April 2007 (Staffordshire, UK) and 1-2 August 2007 (Tampere, Finland). The British Computer Society Press for the 50th Anniversary of the British Computer Society.
- [Berki and Siakas, 2007] Berki, E., Siakas, K. and Georgiadou, E. (2007). Agile Quality or Depth of Reasoning? Applicability versus Suitability with Respect to Stakeholders' Needs. Book Chapter. Stamelos, I. & Sfetsos, P. (Eds) *Agile Software Development Quality Assurance*. IRM Press and Idea Group Publishing: Hershey, PA, USA. March 2007.
- [Berki et al, 2004] Berki, E., Georgiadou, E. & Holcombe, M. (2004). Requirements Engineering and Process Modelling in Software Quality Management – Towards a Generic Process Metamodel. *The Software Quality Journal*, 12, pp. 265-283, Apr. 2004. Kluwer Academic Publishers.
- [Berki, 2004] Berki, E. (2004), Formal Metamodelling and Agile Method Engineering in MetaCASE and CAME Tool Environments. Tigka, K. & Kefalas, P. (Eds) *The 1st South-East European Workshop on Formal Methods. Agile Formal Methods: Practical, Rigorous Methods for a changing world* (Satellite of the 1st Balkan Conference in Informatics, 21-23 Nov 2003, Thessaloniki). Pp. 170-188. South-Eastern European Research Center (SEERC): Thessaloniki.

- [Berki, 2001] Berki, E. (2001). *Establishing a scientific discipline for capturing the entropy of systems process models: CDM-FILTERS - A Computational and Dynamic Metamodel as a Flexible and Integrated Language for the Testing, Expression and Re-engineering of Systems*. Ph. D. thesis, Nov 2001. Faculty of Science, Computing & Engineering, University of North London, London, UK.
- [Berki and Georgiadou, 1996] Berki, E. & Georgiadou, E. (1996), Towards resolving Data Flow Diagramming Deficiencies by using Finite State Machines. I M Marshall, W B Samson, D G Edgar-Nevill (Eds) *Proceedings of the 5th International Software Quality Conference*. Universities of Abertay Dundee & Humberside, Dundee, Scotland, Jul 1996, ISBN: 1 899796 02 9.
- [Berki and Novakovic, 1995] Berki, E. & Novakovic, D. (1995), Towards an Integrated Specification Environment (ISE) Katsikas, S. (Ed.) *Proceedings of the 5th International Hellenic Conference of Informatics*. Athens, Greece, 7-9 Dec 1995. pp. 259-269, Greek Computer Society, EPY: Athens.
- [Bochman and Sunshine, 1980] George V. Bochman and Carl A. Sunshine, Formal methods in communication protocol Design. *IEEE Transactions on Communications*, Vol. COM-28, No. 4, April 1980, pp. 624-631
- [Brand and Zafiropulo, 1981] Daniel Brand and Pitro Zafiropulo, On communicating finite state machines *IBM Research Repor RZ 1053*, January 1981
- [Chapin, 1983] Lyman A. Chapin, Connection and connectionless data transmission, *Proceeding IEEE*, vol. 71, no. 12, December 1983, pp. 1365-1371
- [Chow, 1978] Chow, T.S., Testing Software Design Modeled by Finite-State Machines, *IEEE Transactions on Software Engineering*, vol. SE-4, no 3, May 1978
- [Eilenberg, 1974] Eilenberg, S., Automata, Languages and Machines, Vol. A, Academic Press, 1974
- [Fencott, 1996] Fencott, C., Formal Methods for Concurrency, International Thomson Computer Press, 1996
- [Fujiwara et al., 1991] Fujiwara, S., v. Bochmann, G., Khedek, F., Amalou, M. & Ghedamsi, A., Test Selection Based on Finite State Models, *IEEE Transactions on Software Engineering*, Vol. 17, No. 6, Jun. 1991
- [Georgiadou and Berki, 1996] Georgiadou, E. & Berki, E. (1996), Improving Systems Specification Understandability by Using a Hybrid Approach. M Bray; H-J Kugler; M Ross; G Staples (Eds) *INSPIRE I Process Improvement in Teaching and Training*. First International Conference on Software Process Improvement, Research, Education and Training. (INSPIRE '96), Sep 1996, Bilbao, Spain. Pp. 137-147, SGEC Publications, ISBN: 1899621113.
- [Ghosh et. al., 1999] A. Ghosh et. al., Shared channels for packet data transmission in W-CDMA, *Proceedings of VTC'99 Fall*, Amsterdam Netherlands, 19-22 September 1999, pp. 943-947

- [Holcombe, 1998] Holcombe, M., X-Machines as a Basis for Dynamic System Specification, *Software Engineering Journal*, March 1988.
- [Holcombe and Ipaté, 1998] Holcombe, M. and Ipaté, F., *Correct Systems - Building a Business Process Solution*, Springer-Verlag, 1998
- [Hopcroft and Ullman, 1979] Hopcroft, J.E. and Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Second Edition, Addison-Wesley, Reading, Massachusetts, 1979
- [Merlin, 1979] Philip M. Merlin, Specification and validation of protocols. *IEEE Transactions on Communications*, Vol. COM-27, November 1980, pp. 1761-1680
- [Ince, 1995] Ince, D., *Software Quality Assurance*, McGraw-Hill, 1995
- [NIEMI, 2006] Aki Niemi, An Extension to Session Initiation Protocol (SIP) Events for Issuing Conditional Subscriptions, draft niemi sip subnot etags 04, *IETF*, 2006
- [Jensen, 1994] Kurt Jensen: An introduction to the theoretical aspects of coloured Petri nets. In: *J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.): A Decade of Concurrency, Lecture Notes in Computer Science* Vol. 803, Springer-Verlag 1994, 230–272.
- [Kristensen et. al., 1998] L.M. Kristensen, S. Christensen, K. Jensen: *The Practitioner's Guide to Coloured Petri Nets*. *International Journal on Software Tools for Technology Transfer*, 2 (1998), Springer Verlag, 98-132.
- [Peterson, 1977] James L. Peterson, Petri nets, *ACM Computing Surveys (CSUR)*, v.9 n.3, p.223-252, September 1977
- [Pfleeger, 1998] Pfleeger, L. S., *Software Engineering, Theory and Practice*, Prentice Hall, 1998
- [Phalp and Shepperd, 1994] Phalp, K. and Shepperd, M., A Pragmatic Approach to Process Modelling”, in B. Warboys (Ed.), *Software Process Technology*, EWSPT’94, LNCS 772, pp. 65-68, Springer-Verlag, 1994
- [Rolland and Richard, 1983] Rolland, C. & Richard, C., A dynamic model of Information Processing System based on Finite Automata, *Systems Science*, Vol. 9, no 1-2, 1983
- [Saadia, 1999] Saadia, A., *An Investigation into the Formalisation of Software Design Schemata*, MPhil Thesis, Faculty of Science, Computing and Engineering, University of North London, May 1999
- [SAINTANDRE, 2007] P. Saint-Andre, Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, draft-saintandre-rfc3921bis-04, *IETF*, 2007
- [Singh et al., 2006] Vishal Kumar Singh, Henning Schulzrinne, Markus Isomaki, Piotr Boni, *Presence Traffic Optimization Techniques*, 28th Oct 2006

- [Sol, 1992] Sol, H.G., Dynamics in Information Systems, Dynamic Modelling of Information Systems II, H.G. Sol & R.L. Crosslin (eds.), North Holland/Elsevier Science Publishers, pp. 25-33, 1992
- [Sommerville, 1992] Sommerville, I., Software Engineering, Addison-Wesley, 1992
- [Sunshine, 1981] Carl A. Sunshine, Formal modelling of communication protocols. *University of Southern California, ISI RR-811-89*, March 5, 1981
- [Sunshine, 1978] Carl A. Sunshine, Survey of protocol definition and verification techniques. *Proc. Computer Network Protocol Symposium, Lizge, Belgium*, 1978
- [Tanenbaum, 2002] Andrew S. Tanenbaum, *Computer Networks 4th Edition*. Prentice Hall, 2002
- [Tardieu, 1992] Tardieu, H., Issues for Dynamic Modelling through Recent Developments in European Methods, Dynamic Modelling of Information Systems II, H.G. Sol & R.L. Crosslin (eds.), North Holland/Elsevier Science Publishers, pp. 3-23, 1992
- [Veijalainen et al., 2005] Veijalainen, J., Berki, E., Lehmonen, J. & Moisanen, P. (2006), Realising a New International Paper Mill Efficiency Standard - Using Computational Correctness Criteria to Model and Verify Timed Events. Eleftherakis, G. (Ed) *The 2nd South-East European Workshop on Formal Methods. Practical dimensions: Challenges in the business world*. 18-19 Nov 2005, Ohrid. Satellite of the 2nd Balkan Conference in Informatics, Ohrid, FYROM, 17-20 Nov 2005.
- [West, 1978] C.H. West, General technique for communications protocol validation. *IBM Journal for Research and Development*, vol 22, pp. 393404, July 1978
- [Wood, 1987] Wood, D., Theory of Computation, J. Wiley and Sons, 1987
- [Zafiropulo et. al., 1980] Pitor Zafiropulo et. al., Towards Analyzing and Synthesizing Protocols. *IEEE Transactions on Communications*, Vol. COM-28, No. 4, April 1980, pp. 651-661
- [Zimmerman, 1980] Hubert Zimmermann, OS1 reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, Vol. COM-28, No. 4, April 1980, pp. 425 – 432

References for technical issues and standardization aspects

- [3GPP23907, 1999] 3GPP, Technical specification group services system aspect, QoS Concept (3G TR 23.907 version 1.3.0), 1999
- [SIP, 1999] Jonathan Rosenberg et. al., SIP: session initiation protocol, RFC 2543, *IETF*, 1999
- [SAP, 2000] Charile Perkins et. al., SAP: session announcement protocol, RFC 2974, *IETF*, 2000

- [SDP, 1998] M. Handley et. al., SDP: session description protocol, RFC 2327, *IETF*, 1998
- [RTSP, 1998] Henning Shultzerinne et. al., RTSP: real time streaming protocol, RFC 2326, *IETF*, 1998
- [RTP, 1996] Henning Shultzerinne et. al., RTP: a transport protocol for real-time applications, RFC 1889, *IETF*, 1998
- [RFC3439, 2002] R. Bush and D. Meyer, Some Internet Architectural Guidelines and Philosophy, RFC 1889, *IETF*, 1998
- [RFC4101, 2005] E. Rescorla, Writing protocol models, RFC4101, *IETF*, 2005
- [RFC1122, 1989] Internet Engineering Task Force, Requirements for Internet hosts – Communication Layers, RFC1122, *IETF*, 1989
- [UMTS3003, 1997] Universal Mobile Telecommunication Systems (UMTS); Selection procedures for the choice of radios transmission technologies of the UMTS, TR 101 112 V3.1.0 (1997-11), UMTS 30.03
- [ISO 1983] ISO International Standard 7498, Information Processing Systems – Open System Interconnection – Basic Reference Model, Geneva, October 1983
- [RFC2778, 2000] M. Day. J. Rosenberg, A model for presence and instant messaging, RFC2778, *IETF*, 2000
- [RFC3265, 2002] A. B. Roach, Session initiation protocol, (SIP)-specific event notifications, RFC 3265, *IETF*, 2002
- [RFC3903, 2004] A. Niemi, Session initiation protocol (SIP) extension for event state publication, RFC 3903, *IETF*, 2004
- [RFC3863, 2004] H. Sugano et. al, Presence information data format, RFC 3863, *IETF*, 2004
- [RFC4840, 2006] H. Shultzerinne, RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF), RFC 4840, *IETF*, 2006
- [RFC4479, 2006] J. Rosenberg, A data model for presence, RFC 4479, *IETF*, 2006
- [RFC4840, 2006] J. Rosenberg, A Presence Event Package for the Session Initiation Protocol (SIP), RFC 4840, *IETF*, 2004

Appendix A - IP Multimedia Presence Service

The UE needs to register to IMS prior to using any of the services offered by the network. The Registration Scenario is depicted in Figure 3.

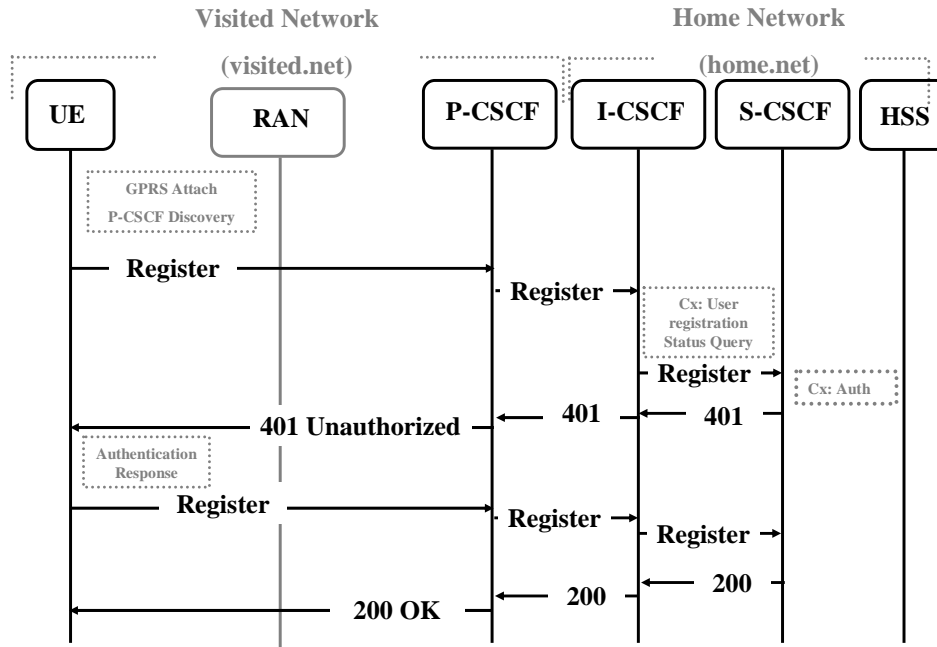


Figure 33 – IMS Registration Procedure

The UE performs GPRS attach and discovers the P-CSCF in the visited network. SIP REGISTER is sent to P-CSCF.

```

REGISTER sip:registrar.home.net SIP/2.0
Via SIP/2.0/udp
[5555::aaa::bbb::ccc::ddd];comp=sigcomp;branch=rfj345892y8r
Max-Forwards: 70
P-Access-Network-Info:3GPP-UTRAN-TDD; utran-cell-id-
3gpp=425252J053694R3
From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34
To: <sip:+358-40-4325555@home.net;user=phone>
Contact:
<sip:[5555::aaa::bbb::ccc::ddd];comp=sigcomp>;expires=600000
Call-ID: 485ucw34573w05dut92
Authorization: Digest username="user_x_private@home.net",
realm="registrar.home.net", nonce="",
uri="sip:registrar.home.net",

```

```

response=""

Security-Client: ipsec-3gpp; alg=hmac-sha-1-96; spi-c=233432;
spi-s=4234234;

port-c=1234; port-s=5466

Proxy-require: sec-agree

CSeq: 1 REGISTER

Supported: path

Content-length: 0

```

Based on DNS query the P-CSCF forwards the request the appropriate I-CSCF. The I-CSCF will contact the HSS in order to get S-CSCF capabilities. Based on the answer from HSS the I-CSCF will select a suitable S-CSCF and forward the SIP REGISTER.

The S-CSCF shall challenge the user and sends back a 401 Unauthorized containing the needed information for the UE to generate the response. The SIP Response travels back to the UE:

```

SIP/2.0 401 Unauthorized

Via SIP/2.0/udp

[5555::aaa::bbb::ccc::ddd];comp=sigcomp;branch=rfj345892y8r

From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34

To: <sip:user_x@home.net>; tag=45d1

WWW_Authenticate: Digest realm="registrar.home.net",
    nonce="base64(RAND + AUTN + Server specific data)",
    algorithm=AKAv1-MD5

Call-ID: 485ucw34573w05dut92

CSeq: 1 REGISTER

Security-Server: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; spi-
c=5457934252;

spi-s=4234234; port-c=4321; port-s=6645

Content-length: 0

```

The UE generates the Authentication Response and session keys and sends SIP REGISTER to the P-CSCF discovered in previously. The SIP REGISTER will travel from P-CSCF to S-CSCF via the I-CSCF.

```

REGISTER sip:registrar.home.net SIP/2.0

Via SIP/2.0/udp

[5555::aaa::bbb::ccc::ddd];comp=sigcomp;branch=rfj345892y8r

Max-Forwards: 70

P-Access-Network-Info:3GPP-UTRAN-TDD; utran-cell-id-
3gpp=425252J053694R3

```

From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34
 To: <sip:+358-40-4325555@home.net;user=phone>
 Contact:
 <sip:[5555::aaa::bbb::ccc::ddd];comp=sigcomp>;expires=600000
 Call-ID: 485ucw34573w05dut92
 Authorization: Digest username="user_x_private@home.net",
 realm="registrar.home.net",
 nonce="base64(RAND + AUTN + Server specific data)",
 algorithm=AKAv1-MD5, uri="sip:registrar.home.net",
 response="543759435fa863de348c3ba"
 Security-Client: ipsec-3gpp; q=0.1;alg=hmac-sha-1-96; spi-
 c=233432;
 spi-s=4234234; port-c=1234; port-s=5466
 Proxy-require: sec-agree
 CSeq: 2 REGISTER
 Supported: path
 Content-length: 0

Should the Response prove to be the right, one the S-CSCF will indicate a successful registration and a SIP 200 OK will be sent back to UE.

SIP/2.0 200 OK
 Via SIP/2.0/udp
 [5555::aaa::bbb::ccc::ddd];comp=sigcomp;branch=rfj345892y8r
 From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34
 To: <sip:+358-40-4325555@home.net;user=phone>; tag=45d1
 Call-ID: 485ucw34573w05dut92
 CSeq: 2 REGISTER
 P-Associated-URI: <sip:+358-40-4325555@home.net;user=phone>,
 <sip:user_x_public@home.net>,
 Content-length: 0

Presence Traffic over the Radio Interface

After a successful registration the UE is able to use other services offered by the network. As stated before, Presence is one of the most common services in an IMS network. It first and foremost enables the use of other services.

The UE needs to subscribe for presence information in order to start receiving it. The subscription scenario goes as shown in the following picture.

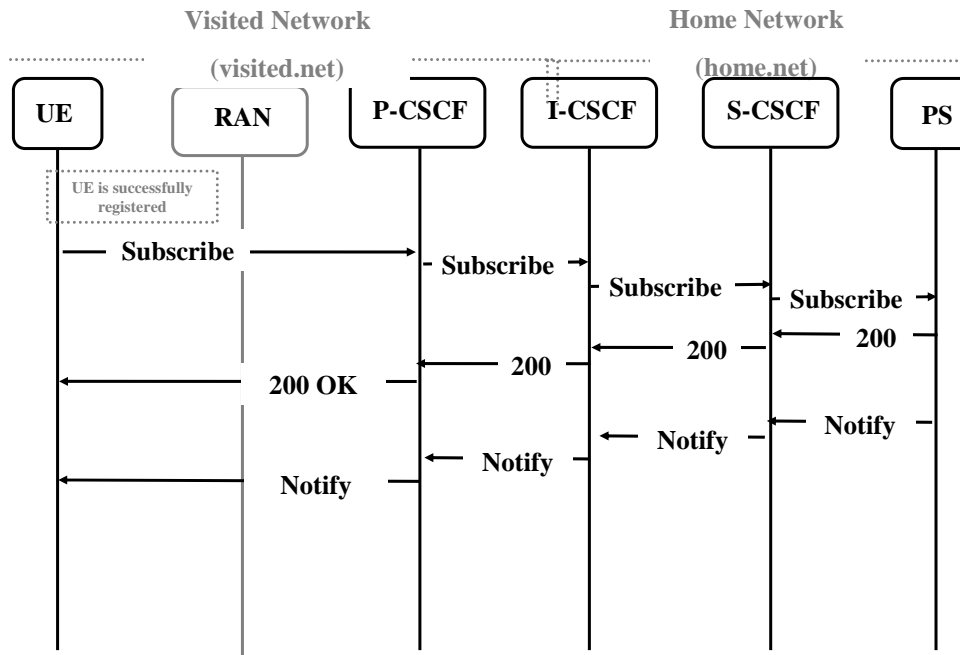


Figure 34 – Notifying Presence Information Updates in IMS

In this particular case we consider that a Resource List Server (RLS) is part of the Presence Server. The user already has a list on the RLS and how that list is created and managed is out of the scope of this research. The purpose of the list on the RLS is to enable the use case (scenario) when a single subscription is performed on the air interface instead of many individual subscriptions.

The UE send a SUBSCRIBE message to P-CSCF (the right P-CSCF was already discovered at registration phase).

```

SUBSCRIBE sip:+358-40-4325555_subscriptions@home.net SIP/2.0
Via SIP/2.0/udp
[5555::aaa::bbb::ccc::ddd];comp=sigcomp;branch=rfj345892y8r
Max-Forwards: 70
Route: <sip:pcscf1.visited.net:4324;Ir;comp=sigcomp>,
<sip:orig@scscf1.home.net;Ir>
P-Preferred-Identity, Joe Doe" <sip:+358-40-
4325555@home.net;user=phone>
  
```

P-Access-Network-Info:3GPP-UTRAN-TDD; utran-cell-id-
 3gpp=425252JO53694R3
 Privacy: none
 From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34423
 To: <sip:+358-40-4325555@home.net;user=phone>
 Call-ID: 658034vpert40584
 Require: sec-agree
 Proxy-Require: sec-agree
 CSeq: 321 SUBSCRIBE
 Event: presence
 Expires:60000
 Accept: application/pidf+xml, application/cpim-pidf+xml;q=0.2,
 application/xpidf+xml;q=0.1
 Allow: INVITE, ACK, BYE, CANCEL, OPTIONS,
 PRACK, INFO, MESSAGE, SUBSCRIBE, NOTIFY, REFER, UPDATE
 Security-verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; spi-
 c=5457934252;
 spi-s=4234234; port-c=4321; port-s=6645
 Contact: <sip:[5555::aaa::bbb::ccc::ddd];comp=sigcomp>
 Content-length: 0

The request is routed appropriately to the Presence server via the CSCFs. The Presence Server accepts or denies the subscription and the response is sent back to the UE.

SIP/2.0 200 OK
 Via SIP/2.0/udp
 [5555::aaa::bbb::ccc::ddd];comp=sigcomp;branch=rfj345892y8r
 Record-Route: <sip:pcscf1.visited.net:4324;Ir;comp=sigcomp>
 P-Asserted-Identity: <sip:scscf1.home.net>
 Privacy: none
 From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34423
 To: <sip:+358-40-4325555@home.net;user=phone>
 Call-ID: 658034vpert40584
 CSeq: 321 SUBSCRIBE
 Expires:60000
 Contact: <sip:sip:scscf1.home.net>
 Content-length: 0

Should the Presence Server accepts the subscriptions a notification containing the requested presence information will be sent back to the UE. The Notify message will travel via the CSCFs and it will be acknowledged by UE.

```
NOTIFY sip:[5555::aaa::bbb::ccc::ddd];comp=sigcomp SIP/2.0
Via SIP/2.0/UDP pcscf.home.net;branch=rfj345892y8r
Max-Forwards: 69
Route: <sip:pcscf.home.net;Ir>
From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34423
To: <sip:+358-40-4325555@home.net;user=phone>
Call-ID: 658034vpert40584
CSeq: 423 NOTIFY
Subscription-State: active; expires=60000
Event:presence
Content-type: application/pidf+xml
Contact: <sip:sip:scscf1.home.net>
Content-length: 2986
```

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
xmlns:pp="urn:ietf:params:xml:ns:pidf:person"
xmlns:pd="urn:ietf:params:xml:ns:pidf:device"
xmlns:rp="urn:ietf:params:xml:ns:pidf:rp-id-person"
xmlns:rt="urn:ietf:params:xml:ns:pidf:rp-id-tuple"
xmlns:rs="urn:ietf:params:xml:ns:pidf:rp-id-status"
xmlns:ot="urn:oma:params:xml:ns:pidf:oma-tuple"
xmlns:ots="urn:oma:params:xml:ns:pidf:oma-tuple-status"
xmlns:ops="urn:oma:params:xml:ns:pidf:oma-person-status"
xmlns:ods="urn:oma:params:xml:ns:pidf:oma-device-status"
entity="sip:someone@example.com">
<tuple id="a1231">
  <status>
    <basic>open</basic>
    <ots:willingness>
      <ots:basic>open</ots:basic>
    </ots:willingness>
    <ots:session-participation>
```

```

        <ots:basic>open</ots:basic>
    </ots:session-participation>
    <rs:status-icon> http://example.com/~my-icons/PoC-Session
</rs:status-icon>
</status>
<rt:class>forfriends</rt:class>
<ot:service-description>
    <ot:service-id>org.openmobilealliance:PoC-
Session</ot:service-id>
    <ot:version> 1.0 </ot:version>
    <ot:description>This is the OMA PoC-Session
service</ot:description>
</ot:service-description>
<rt:device-
id>urn:omai:be874b7a3a3fce7d0e91649a97762e64</rt:device-id>
    <contact>sip:my_name@example.com</contact>
    <timestamp>2005-02-22T20:07:07Z</timestamp>
</tuple>
<tuple id="a1232">
    <status>
        <ots:basic>closed</ots:basic>
        <ots:willingness>
            <ots:basic>closed</ots:basic>
        </ots:willingness>
        <rs:status-icon> http://example.com/~my-icons/PoC-Alert
</rs:status-icon>
    </status>
    <rt:class>forfriends</rt:class>
    <ot:service-description>
        <ot:service-id>org. openmobilealliance:IM</ot:service-id>
        <ot:version>1.0</ot:version>
        <ot:description>This is the OMA IM service</ot:description>
    </ot:service-description>
    <contact>sip:my_name@example.com</contact>
    <timestamp>2005-02-22T20:07:07Z</timestamp>
</tuple>

```

```

<pp:person id="a1233">
  <pp:status>
    <ops:overriding-willingness>
      <ops:basic>open</ops:basic>
    </ops:overriding-willingness>
    <rp:activities>
      <rp:activity> meeting </rp:activity>
    </rp:activities>
    <rp:place-type> office </rp:place-type>
    <rp:mood> <rp:happy/> </rp:mood>
    <rs:status-icon>http://example.com/~my-icons/busy
  </rs:status-icon>
    <rp:timeoffset>120</rp:timeoffset>
  </pp:status>
  <rt:class>forfriends</rt:class>
  <pp:note>I am in a boring meeting!!</pp:note>
  <pp:timestamp>2005-02-22T20:07:07Z</pp:timestamp>
</pp:person>
<pd:device id="a1234">
  <pd:status>
    <ods:network-availability>
      <ods:network id="UMTS"/>
      <ods:network id="GPRS"/>
    </ods:network-availability>
  </pd:status>

  <pd:deviceID>urn:omai:be874b7a3a3fce7d0e91649a97762e64</pd:device
  ID>
  <pd:timestamp>2005-02-22T20:07:07Z</pd:timestamp>
</pd:device>
</presence>

```

The UE acknowledges the notification:

SIP 200 OK

Via SIP/2.0/UDP pcscf.home.net;branch=rfj345892y8r

```

P-Access-Network-Info:3GPP-UTRAN-TDD; utran-cell-id-
3gpp=425252JO53694R3
From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34423
To: <sip:+358-40-4325555@home.net;user=phone>
Call-ID: 658034vpert40584
CSeq: 423 NOTIFY
Subscription-State: active; expires=60000
Event:presence
Content-type: application/pidf+xml
Contact: <sip:sip:scscf1.home.net>
Content-length: 0

```

The Presence Information stored on the Presence Server is updated by various different entities. One of the entities is the UE – the applications running in the UE. The publication of Presence Information goes according to the Figure 35.

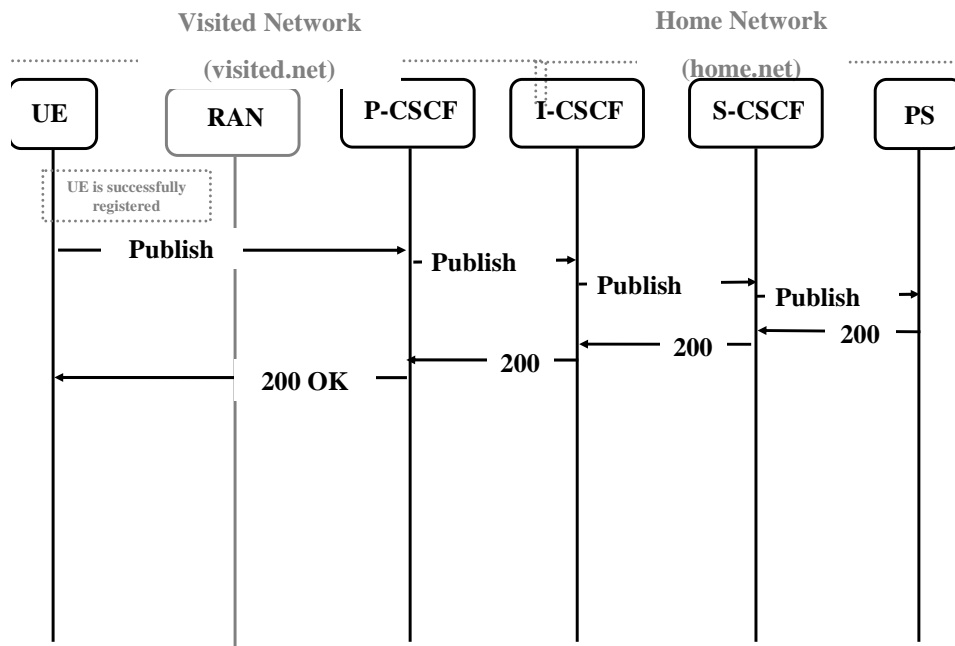


Figure 35 – Publishing Presence Information in IMS

The UE sends the following request to the PS via CSCFs. The route by which the SIP message goes to PS was agreed at registration phase.

```

PUBLISH sip:+358-40-4325555_subscriptions@home.net SIP/2.0
Via SIP/2.0/udp
[5555::aaa::bbb::ccc::ddd];comp=sigcomp;branch=rfj345892y8r
Max-Forwards: 70

```

Route: <sip:pcscf1.visited.net:4324;Ir;comp=sigcomp>,
 <sip:orig@scscf1.home.net;Ir>
 P-Preferred-Identity, Joe Doe" <sip:+358-40-
 4325555@home.net;user=phone>
 P-Access-Network-Info:3GPP-UTRAN-TDD; utran-cell-id-
 3gpp=425252J053694R3
 Privacy: none
 From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34423
 To: <sip:+358-40-4325555@home.net;user=phone>
 Call-ID: 658034vpert40584
 Require: sec-agree
 Proxy-Require: sec-agree
 CSeq: 321 PUBLISH
 Event: presence
 Expires:60000
 Accept: application/pidf+xml;q=0.1, application/cpim-
 pidf+xml;q=1,
 Security-verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; spi-
 c=5457934252;
 spi-s=4234234; port-c=4321; port-s=6645
 Contact: <sip:[5555::aaa::bbb::ccc::ddd];comp=sigcomp>
 Content-type: application/pidf-diff+xml
 Content-length: 2986

(The payload is the same as in the NOTIFY request sent to the UE.)

PS indicates the success or failure by sending the appropriate SIP Response back to UE. A successful publication generates the following message.

SIP/2.0 200 OK
 Via SIP/2.0/udp
 [5555::aaa::bbb::ccc::ddd];comp=sigcomp;branch=rfj345892y8r
 Record-Route: <sip:pcscf1.visited.net:4324;Ir;comp=sigcomp>
 P-Asserted-Identity: <sip:scscf1.home.net>
 Privacy: none
 From: <sip:+358-40-4325555@home.net;user=phone>; tag=fa34423

To: <sip:+358-40-4325555@home.net;user=phone>

Call-ID: 658034vpert40584

CSeq: 321 Publish

Expires:60000

Contact: <sip:sip:scscf1.home.net>

Content-length: 0