An Investigation into the Efficacy of Resource List Servers in IMS Presence Service Applications

Prepared by:
Michael James Pitman

Supervised by:
Neco Ventura

Department of Electrical Engineering
University of Cape Town
2009

This dissertation is submitted to the University of Cape Town
in fulfillment of the academic requirements
for the Degree of Master of Science in Engineering

22 July 2009

Declaration

I declare that this thesis is my own work. Where material generated by other researchers is included, the parties and/or material are indicated in the acknowledgements or references as appropriate.

This work is being submitted for examination for the Degree of Master of Science in Electrical Engineering at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

Signed by candidate

Michael James Pitman Date

Acknowledgments

I would like to thank a number of people for their assistance and guidance during this thesis.

Firstly, many thanks to my supervisor, Mr Neco Ventura, for his help, guidance and funding through the course of this thesis. Also, to Dr David Waiting, Vitalis Ozianyi, and Richard Good for their help, advice and guidance regarding the IMS and postgraduate studies, and Eugene Golovins for his highly appreciated and valued reviewing.

My parents, for their support of my extended studies.

Dr Bronwyn Weber for her support and encouragement during the development and writing of this thesis.

Jean Deruelle of Mobicents, for his work on Mobicents Tomcat, and technical help during development.

Rob, Richard S, Gabriel, Armstrong and Phillipa in the lab, and Barry, Fabio, William, Gunther, Ben, Sam and others during breaks.

Synopsis

The Internet Protocol (IP) Multimedia Subsystem (IMS) is a service-provisioning framework specified by the Third Generation Partnership Project (3GPP) to enable the development and delivery of new and innovative services to a subscriber base quickly and efficiently, through the convergence of disparate access network technologies. The service delivery and network management is managed through a bandwidth-rich core network, while service access for subscribers occurs over wire-line or wireless access network technologies, with vastly different bandwidth resources. With a myriad of services likely to be available, the effective utilisation of the access network resource becomes important, especially when running services simultaneously.

Presence is one of the first services being implemented in the IMS, and is likely to become a standard service to almost all subscribers. The service is relatively simple itself, and likely to be "always on". A subscriber publishes their presence information (presentity) to their Presence Server (PS), and subscribes to "watch" other presentities. Any one subscriber is likely to subscribe to numerous other presentities, and then receives all updates made to the subscribed presentities. The volume of data over the access network caused by these updates can become quite large, especially if many active presentities are subscribed to. This is affected by design decisions made during the specification of both the presence and the IMS, which yields a poor encoding efficiency for presence information, and a particularly data-intensive signalling schema.

Attempts to mitigate the effect this data has on the access network resources has led to the introduction of the presence Resource List Server (RLS). The RLS subscribes to presentities contained in an addressable Resource List on the subscribers' behalf, and remains on the signalling path. It aggregates the subscribed presence information it receives by combining a number of received presentities into a single periodic notification transmitted to the subscribers' Presence User Agent (PUA), reducing the SIP overhead cost per presentity.

This study examines the operation of the RLS in the IMS presence framework, and the two primary data reductions performed: transferring most of the presence signalling from the access to the core network, and reducing the Session Initiation Protocol (SIP) packet overhead by aggregating the presence information transmitted to the PUA. The effectiveness and efficacy of the aggregation methods available to the RLS form the primary examination in this thesis.

An evaluation framework is designed and implemented in order to examine the performance of the different aggregation methods available. The framework utilises SIP traffic generators and sinks, interacting with the developed RLS under different scenarios, based on anticipated operating conditions for an RLS. Each aggregation method is tested under similar conditions for comparative purposes. The RLS logs the incoming traffic from the respective PSs, as well as the outgoing traffic to the subscriber. The logs are recorded and analysed, the findings examined, and compared to the other methods used in the study. The findings illustrate that the respective methods aggregate the presence data to the PUA with varying effectiveness under most conditions. It is noted however that with certain parameter conditions, the methods investigated become increasingly inefficient, and can render the RLS totally ineffective.

Contents

List of Figures

List of Tables

Nomenclature

3GPP Third Generation Partnership Project - standardisation body driving specifications for the development of the NGN.

AIM America Online (AOL) Instant Messenger - An instant message protocol and chat client, released by AOL.

B2BUA Back-to-back User Agent - An entity implementing two basic user agents, one a client, and the other a server.

BICC Bearer Independent Call Control - a telecommunications protocol considered as a candidate for the signalling protocol for the IMS.

CPP Common Profile for Presence - An IETF specification defining a presence profile to allow interoperation between different presence implementations.

EDGE Enhanced Data-Rate for GPRS Evolution - An evolutionary enhancement made to GPRS, allowing for higher data transmission rates than GPRS.

ETSI-TISPAN European Telecommunications Standards Institutes' Telecommunications and Internet converged Services and Protocols for Advanced Networking - European standardisation body working in conjunction with 3GPP to develop NGN specifications.

EXI Efficient XML Interchange - An implementation of XML using binary encoding to reduce the encoding overhead.

EXIWG Efficient XML Interchange Working Group - The W3C Working Group responsible for the development of the EXI standard.

GGSN Gateway GPRS Support Node - A GPRS network element acting as a gateway between the GPRS network and the Internet.

GPRS General Packet Radio Service - a data-transmission enhancement made to the GSM standard.

GSM Global System for Mobile Communications - second generation mobile telephone standard, used worldwide.

HTTP HyperText Transfer Protocol - a World Wide Web Consortium communication protocol for requesting and receiving HTML documents from a server.

ICQ ICQ - a homophone for I Seek You, an instant messenger protocol and chat client.

IEEE Institue for Electrical and Electronics Engineers - professional organisation for electrical and electronic engineers, involved in the standardisation of various technologies.

IETF Internet Engineering Task Force - An international community involved in the evolution of Internet architecture, who develop relevant standards and protocols.

IM Instant Messaging - IP applications allowing two or more users to exchange ext messages, files, URLs and more, in real time conversations.

IMS Internet Protocol (IP) Multimedia Subsystem - a service provisioning framework for NGN.

IP Internet Protocol - an Internet protocol for delivering datagrams from source to destination based on the destinations address.

IPTV Internet Protocol Television - television services provided over IP networks

ISBN International Standardized Book Number - A unique identity number assigned to reference published books.

ITU-T International Telecommunications Union-Telecommunications - A sector of the International Telecommunications Union managing telecommunications standardisation for its parent body.

LTE Long Term Evolution - The fourth generation successor to the current third generation UMTS network, with higher available data rates and the migration to an entirely IP-based architecture

MPLS Multi-Protocol Label Switching - A mechanism for carrying data over networks, originally developed by Cisco.

OSS Open Source Software - Software that meets the Open Source Definition. Such software must allow users to use, modify and improve the software, and then redistribute it in modified or unmodified form.

PIDF Presence Information Data Format - An IETF specification for encoding presence information into XML.

PIDF Presence Information Data Format - An IETF-defined specification for the encoding of presence information using ML

PNA Presence Network Agent - IMS Core Network entity that updates a Presentity's presence on it's behalf, based on information gleaned from IMS signalling

PS Presence Server - IMS Application Server that acts as a repository for Presence Information uploaded by Presentities, and diseminated to Watchers

PUA Presence User Agent - A User Agent, used in the context of the presence service

QoS Quality of Service - a measure of the level of service being provided by a network or application.

RFC Request For Comments - Standards released by the IETF, governing the operation of protocols in the Internet.

RL Resource List - An addressable list of resources that can be subscribed to

RLS Resource List Server - IMS Application Server that holds a Resource List on behalf of a Watcher, and manages a Watchers' subscription to the Resource List

ROHC Robust Header Compression - IETF Working Group developing compression schemes for header and signalling compression of various application protocols.

RTP Real-time Transport Protocol - A protocol for transmitting real-time media across the Internet

RTSP Real-Time Streaming Protocol - A network control protocol used primarily for streaming media.

RTT Round Trip Time - A measure of the average time required for a response to a packet, incorporating outgoing and return transmission time .

SIP Session Initiation Protocol - An IETF-defined protocol for setting up, modifying and tearing down multimedia or other sessions between hosts

TCP Transmission Control Protocol - a transport layer protocol controlling transmission of data over the Internet.

UCT University of Cape Town

UDP User Datagram Protocol - A connectionless data transmission control protocol for the Internet.

UE User Equipment - The access network device used by a network subscriber to access core network services through the access network.

UML Unified Modelling Language - A modelling language for describing and visualising object-oriented systems.

UMTS Universal Mobile Telephone System - third generation mobile telephone standard.

W3C World Wide Web Consortium - An international consortium of organisations and individuals that develop standards and technologies for internet applications.

WCDMA Wideband Code Division Multiple Access - A standardised commercial implementation of CDMA, used in UMTS as an air interface.

WLAN Wireless Local Area Network - A generic name for short range, high bandwith wireless LAN technologies, such as IEEE 802.11.

XCAP XML Configuration Access Protocol - HTTP-based protocol for accessing XML information stored in repositories.

XML eXtensible Markup Language - A general-purpose specification to create customised markup languages, in which persons may define their own mark-up elements

Chapter 1

Introduction

1.1 Background Information

The IP Multimedia Subsystem (IMS) is a service-provisioning framework designed by the Third Generation Partnership Project (3GPP) [4], and the European Telecommunications Standards Institutes' Telecommunications and Internet converged Services and Protocols for Advanced Networking (ETSI-TISPAN) [5] group, to counteract the marginalisation of traditional telecommunications providers in the current converging environment. The convergence of mobile and fixed-line access technologies towards an all-IP based network has opened the infrastructure of traditional telecommunications providers to services and applications operated and owned by third parties, and hasten the commodification of their access networks [6]. The IMS caters for the traditional operator services such as user-to-user communications, as well as serving as a platform for the operator (and third parties) providing innovative and value-added services to subscribers [1]. The operator, however, maintains control of the business aspects of the network, and acts as an intermediary between subscribers and third party service providers, while leveraging features such as single sign-on and the unified billing of utilised services [6].

The IMS abstracts functionality to the IMS core network, where SIP proxy servers route signalling traffic either to other IMS subscribers, or to application servers providing services and applications. By controlling IMS service delivery through the core network, whether in-house or from third parties, operators are able to manage and monetise their networks more efficiently. As the network functionality and service delivery is managed from a core network, subscriber access to services can be achieved through any IP-based access network technology, such as wire-line, cellular (GSM, UMTS), or WiMax (IEEE 802.16) [7]. This results in better and ubiquitous connectivity, with the migration of services and active sessions between devices possible, as required.

The access-technology-agnostic approach to service delivery does however present a challenge, particularly to Quality of Service (QoS) provisioning. The access networks on hand may offer vastly different available bandwidths and service guarantees, which need to be carefully mana-

ged. This is particularly pertinent for applications with stringent QoS or bandwidth demands. Wire-line networks such as cable or Ethernet may offer high available bandwidths, but are geographically fixed, offering no mobility. WiFi (IEEE 802.11) can also offer relatively high bandwidths to mobile subscribers, but is restricted in range to a relatively small geographic area. Mobile networks with considerably longer range, such as WiMax, UMTS and GSM, are characterised by substantially lower bandwidths. The 3GPP's Long Term Evolution (LTE) standardisation effort aims to increase the cellular network bandwidths, but is currently still far from completion. The IMS core network entities, and particularly the application server in question, will need to be aware of such constraints, and adjust their service as necessary. This could mean reducing the speed of FTP transfers, and encoding audio or video to a lower quality stream, in order to satisfy the bit rate constraints [6].

This situation can be further complicated by a subscriber running a number of background applications on their User Equipment (UE) simultaneously. The applications would have to be managed carefully by the controlling IMS core, taking into account their respective data requirements. Real-time traffic, such as voice, video calls or conferencing, are sensitive to delay and jitter, and so would need to be given priority over other time-insensitive applications such as email, Instant Messaging (IM) or presence. However these time-insensitive applications are still likely to be of importance to the subscriber, and should not be starved by other bandwidth-heavy applications.

Most currently used implementations of presence form an integral part of IM applications. In this role, presence indicates the availability of an IM "buddy" for communication, with information regarding the other parties mood or activity supplied by the "buddy". These implementations are highly fragmented, and can not inter-operate with other applications. This has led to many IM users maintaining subscriptions with multiple IM clients (for example, .NET Messenger, AOL Instant Messenger (AIM), Excite Pal, Google Talk, ICQ, Jabber, Skype and Yahoo!), replicating the same functionality and presence information between each client. An integrated, converged presence service is seen by industry commentators and bloggers as the likely "Killer App" application that will drive initial adoption of the IMS [8], [9]. Presence can be used as a service enabler, in a similar manner to its use in IM, by plugging into almost any application to provide presence-enriched services and information. IM is something of a "Killer App" itself, the use of which has exploded over the last few years. Table 1.1 illustrates the subscriber bases of some major IM services. (IM services such as QQ and Google Talk are omitted due to lack of available information).

Service	User Base	Max Simultaneous Users	Total IMs / Day
Windows Live	204m [10]	30m [10]	5.7 billion [11]
Yahoo!	60m [12]	Unknown	+1 billion [13]
Skype	196m [14]	14m [15]	Unknown

Table 1.1: Usage Statistics of Major IM Networks

Presence information can also be included in applications such as presence-enabled address books or Emergency Service call-centres, where the information is used to geolocate emergency callers.

Microsoft has recognised the need for presence-enabled systems, and released Microsoft Office Communications Server 2007 [16], which provides integrated presence, IM and web conferencing capabilities in an Office environment, using SIP for signalling [17]. All these applications are likely to receive presence information, and generate it as well, increasing the amount of presence-related traffic in all networks.

Presence is an application that is likely to be running continuously on almost all subscribers' UEs, including lower bandwidth connections such as GSM or UMTS. Although not subject to the time-constraints of audio or video calls, presence data is dynamic, and can become stale and essentially useless if not delivered to the subscriber timeously. Because of design decisions made with regard to IMS operation, and with a view to future use and extensibility of applications such as presence, the signalling used in applications, and the presence data encoding in particular lacks efficiency. For this reason, it is desirable to examine the operation of such services and implement measures to reduce both the frequency and volume of access network traffic, while maintaining timely delivery of the data. This research examines the primary solution proposed for this role in the presence application, the Resource List Server (RLS).

1.2 Problem Definition

The commercial promise of the IMS is the rapid and easy introduction of innovative and profitable services to operators' subscriber bases. Numerous design decisions during the early development of the IMS were made to allow for the easy extensibility and the introduction of new services to the architectural framework, through the use of common service enablers [18].

These decisions, however, have not always taken into account such aspects as access network and subscriber UE constraints. The presence application in the IMS is a service that has been affected by some of these overall IMS decisions, as well as by some decisions related specifically to presence. These decisions affect the presence application in that considerable amounts of data may be transmitted over the access network to the UE, while carrying an effectively small amount of actual presence data. It is desirable to minimise the flow of this bloated information, while still providing an effective and timely transfer of presence data to the UE.

A proposed solution in the literature of the standards bodies is the use of a Resource List Server (RLS) in the presence framework. The RLS serves as a data aggregator between the Presence Server, which is the source of presence information for the UE, and the UE itself.

Some of the primary IMS design decisions which act as motivation for the development of the RLS, and how they affect the presence application, are:

- The Session Initiation Protocol (SIP) [19], developed by the Internet Engineering Task Force (IETF), was chosen by the 3GPP as the signalling protocol for use in the IMS. SIP was chosen over the other IP-compliant protocols under consideration, Bearer Independent Call Control (BICC) and H.323, both International Telecommunication Union-Telecommunications (ITU-T) -specified protocols for signalling and session establishment

[1]. The signalling structure of SIP is borrowed from the HTTP Request/Response model. Each SIP transaction is characterised by a request message that invokes a procedure or function on the receiving server, and one or more response messages [19]. SIP itself does not provide any services, but acts as a carrier for additional protocols such as the Session Description Protocol (SDP) [20], which describe the content and parameters of the session media. SIP headers are human-readable (like HTTP), allowing faster and easier development and debugging of SIP applications. However, this has a significant drawback in transmission as the size of the SIP message header can become very large.

- The IETF specified the use of eXtensible Markup Language (XML) for encoding presence information (Presence Information Data Format (PIDF) [2]). XML was considered to be the most suitable framework for this, as it met the requirements for a hierarchical, fully extensible format. The 3GPP acknowledged this by specifying the PIDF as the format for presence information uploaded by the subscribers' UE [21], as well as specifying the IETF-defined ability to publish partial presence information as required [22], [23]. While XML offers a number of highly desirable features, it does incur a high information encoding overhead. This results in a presence entity (presentity) becoming disproportionately large, in comparison with the amount of raw presence information being encoded.

The aforementioned solutions act to increase considerably the amount of data transmitted across the access network, especially when only small updates of presence information are carried. While the UE would only periodically publish this information to the subscribers' allocated Presence Server (PS), any changes to, or the addition of new presentities would generate a notification from the PS to the subscriber. A watcher monitoring only three or four presentities would be relatively unaffected by this. It is likely however, that most subscribers would be watching a large number of presentities at any given time, especially by utilising converged applications, such as an online presence-enabled address book, which subscribes to all the presentities in the address book.

In addition to this, it is likely that operators would implement IMS core network entities such as Presence Network Agents (PNA) [24] to enable active updates of a subscribers' presentity on their behalf, from within the core network. The combination of subscribers watching large numbers of presentities, and the actions of an entity such as a PNA, are likely to generate a significant amount of presence data, encoded in XML and encapsulated in SIP messages. In addition, presence is likely to be an application that will be "always on", continually publishing updates, and receiving notifications regarding its watched presentities.

The mechanism for reducing the presence traffic across the access network is comprised of two parts. Firstly, the RLS acts as an agent in the IMS core-network operating on behalf of the presence service subscriber. The subscriber uploads their presence information in the form of a "Presentity" (presence entity) to their corresponding Presence Server (PS), and subscribes to their Resource List (RL) held by the RLS. The RLS then subscribes to each presentity contained in the Resource List on behalf of the subscriber, who becomes a "Watcher" of each of those presentities. Without the actions of the RLS in the core network, the subscriber would have to

subscribe individually to each presentity they wish to watch. The signalling required for a single subscription to a presentity is significant, with subscription requests, acknowledgements, and all subsequent presence information messages. By subscribing to an RL from the access network, the repetitive subscription signalling is moved to the IMS core. This leads to a considerable reduction in traffic over the access network to and from the Presence User Agent (PUA).

The second aspect in the RLS traffic reduction is the aggregation of the presence information to the PUA from the PSs of the respective watched presentities. Every change in the presence status of a watched presentity uploaded to its PS, results in a SIP NOTIFY message being sent to all subscribed watchers of that presentity. The RLS acts as a proxy between the PS and PUA, and intercepts these NOTIFYs. The normal proxy behaviour of immediately forwarding the message will effectively minimise any traffic reduction effect the RLS may provide. Instead, the RLS can be used to strip the presence information contained in the body from the message, and buffer it. Thus, a collection of the information from different presentities can be stored, and then forwarded to the PUA when a threshold is met, or other conditions are satisfied. This aggregation removes the SIP header overhead that would have accompanied every NOTIFY message, and replaces it with only one SIP header.

The first aspect, the removal of presence signalling from the access network, can be examined and quantified with relative ease. The second aspect is subject to a practical investigation, and this forms the primary focus of our research.

1.3 Thesis Objectives

This research aims to investigate the role a Resource List Server can play in an IMS presence application framework; in reducing the overall volume and frequency of presence service traffic to the PUA over the shared-medium, bandwidth-constrained access network.

The first objective is the development of a test RLS that enables the manipulation of the aggregation method and the relevant parameters, on the fly. There are three parameters affecting the RLS examined in this study. The first two, which directly affect the incoming stream of presence information are the Resource List (RL) size, and the period between presence updates from a presentity. The latter determines the volume of data per presentity over time, and is referred to as the "update period" in this study. The third parameter in the study is the aggregation parameter value, of which a range are tested for each combination of the RL size and update period, which determines the RLS effectiveness at reducing the volume of data transmitted.

One also needs a means to record characteristics of the incoming packet stream from the respective PSs, and of the aggregated packet stream from the RLS to the PUA. For this purpose a testbed is to be developed. The characteristics of interest for the study are the sizes of the presence data stream received at the RLS and of the aggregated data stream transmitted, and the time between successive NOTIFY messages to the PUA. These parameters are used to de-

termine a relative performance for the RLS, based on how the RLS reduces the received presence information into the aggregated data stream, and how regular presence updates to the PUA are.

The testbed will consist of a number of network entities interacting with the RLS. An entity to subscribe to the RL held by the RLS is required, and a number of entities are required to represent Presence Servers. The RLS subscribes to presentities "served" by the presence servers, which return updated presence information over time. In order to change the aggregation method and parameters in the RLS, an entity is required to send a formulated message to the RLS, containing the new parameters.

Three primary methods for performing the RLS aggregation on the received NOTIFY messages from the PS are examined. Various parameters governing the operation of the RLS aggregation method are considered, and the operation of the RLS under different parameter values is examined. The three methods examined are based on the following criteria: elapsed time, the size of the presence information content gathered, and the number of presence information updates received.

Two secondary methods are also examined and compared in a similar manner. These secondary methods combine the "content size" and "number of presentity updates received" methods respectively, with the "elapsed time" method. These additional methods aim to reconcile potential performance benefits of the former methods, with the timeliness guarantees of the latter.

The examination of each of these methods individually will provide data obtained under a range of similar conditions, to enable comparisons of the different methods. The data obtained should provide suitable guidance for the further design and configuration of RLS implementations in presence frameworks.

1.4 Scope and Limitations

The aim of this work is an experimental investigation of the performance of different aggregation methods implemented in the RLS. A number of different parameters are considered, such as the size of the subscribed resource list, and the rate of activity of the presentities being watched. A set of suitable parameters for each aggregation method is examined for each combination of external factors.

Only the input traffic versus the output traffic of the RLS is under consideration. The incoming and outgoing route of the presence traffic through the IMS core network is not considered to be relevant to this study. As the presence traffic entering the RLS is logically sourced from presence servers located in the home network and possibly presence servers in foreign networks, simulating these entities with SIP traffic generators is considered to be sufficient.

An effective, fully integrated presence service would utilise numerous IMS core network elements and various application servers, such as the Presence Network Agent [24], to ensure that an up-to-date and accurate indication of a subscribers' communication availability, and the mode of communication, is available to watchers at all times. The numerous means and methods of

generating and pushing this information through SIP PUBLISH messages to the respective presence servers is considered to be outside the scope of this investigation.

We examine the mechanisms incorporated into the presence service, as defined by both the IETF and 3GPP, to reduce the amount of encoded presence information pushed to the PS (and then disseminated to the relevant watchers). The presented analysis deals with the impact of such mechanisms on the performance of the RLS.

Additional methods of reducing both the amount of traffic sent from the PS to the PUA, and reducing the effective size of the information transmitted across the access network, are considered to be outside the scope of this project. Although some of them are included in Chapter 2 (Literature Review and Related Work) for completeness, their implementation is separate from that of the RLS, and may or may not be desirable, depending on the access network being used, and the nature of the PUA.

1.5 Thesis Outline

Chapter 2 examines three topics relevant to presence aggregation and the IMS presence framework. Techniques for improving the performance of transport protocols such as TCP over GPRS networks, and the routing of SIP over MPLS through aggregation and resource reservation are examined. An overview of the Session Initiation Protocol (SIP) is then provided, with a focus on the use of SIP in presence standards, and methods available for reducing the SIP header and message through the use of compression. Last is an examination of XML in web services, with a focus on the data coding efficiency and compression of XML-encoded data. Completing Chapter 2 is an examination of the use of XML in IETF presence standards.

Chapter 3 presents the design considerations for the work performed. Functional aspects of the RLS are examined covering the management of presence information and aggregation methods. The testing and analytical methodologies employed are also discussed.

Chapter 4 details the architecture and objectives of the emulation framework. The various entities developed for and employed in the implementation are examined, with some limitations in the implementation of the entities discussed. The approach taken in evaluating the system is presented, detailing the tests performed and the parameters used in each.

Chapter 5 presents the results and findings from the various tests performed using the developed RLS framework. Results obtained from testing are analysed and presented, with a discussion regarding the findings.

Chapter 6 presents conclusions drawn from the material presented in Chapter 5 and the rest of this body of work. Some recommendations for future work are provided, which are based both on the limitations inherent in the current implementation, and the novel models for presence behaviour observed amongst large groups of subscribed presentities.

Chapter 2

Literature Review

We first examine some methods of IP traffic aggregation, and existing implementations to aggregate data, in order to obtain better utilisation and performance of the available network resources. An overview of SIP follows, looking at the standardisation of the protocol and subsequent extensions made to the base protocol to enable development of applications and frameworks, such as presence and the IMS. The use of SIP is the first of two factors that motivate for the development and implementation of RLS services in the presence framework, and is covered in Section 2.2. The second aspect is the use of XML for the encoding of data such as basic and enriched presence information, subscribers presence resource lists, watcher authorisation lists, and event notification filters. Various methods to compress or reduce the size of the XML encoded data are examined in Section 2.3. Lastly, some additional methods proposed for the optimization of presence traffic are examined in Section 2.4.

2.1 IP Traffic Aggregation

The process of traffic aggregation over IP networks uses similar traffic classes grouped together into IP trunks, in order to facilitate common treatment for the traffic classes between points on the network. The use of aggregated traffic flows simplifies network management, allowing the implementation of congestion control or Quality of Service (QoS) provisioning. There are numerous methods available for the management of aggregated IP traffic streams.

This section examines two implementations of aggregation that are potentially applicable to our research problem.

An examination of the operation of the Transmission Control Protocol (TCP) over the General Packet Radio Service (GPRS), and a proposed solution to allow the effective use of TCP over GPRS, is presented. GPRS is a wireless data network extension to the GSM cellular network, that allows data access to cellular subscribers. The operational characteristics of GPRS are significantly different to those of typical wired networks, which can lead to quite different network performance in terms of throughput and delay, when compared to typical wired networks.

We also examine a proposed interface between SIP and Multi-Protocol Label Switching (MPLS) networks, and discuss how the interface improves the performance of SIP traffic across such MPLS-enabled networks by aggregating the data streams across the network. MPLS is a widely used technique that utilises IP trunks to route traffic streams through the Internet [25]. The use of MPLS in the current networking environment is beneficial because of the traffic engineering capabilities it offers; such as QoS guarantees, the efficient use of network resources, and resilience.

2.1.1 Aggregation of TCP over GPRS

GPRS is an extension to the GSM standard, providing an always-on data service for cellular terminals, which utilises network resources only when there is data to send or receive. GPRS traffic is typically considered to be of lower priority in the network than voice traffic, and most implementations only operate on a "best-effort" service. The available data rates for GPRS depend on network utilisation and the operators implementation, but typically provide 40.2kbps on the down link, and 13.4kbps on the up link [26]. Functions such as ordered packet arrival and packet loss recovery are managed by the Radio Link Control (RLC), in conjunction with Automatic Repeat Request (ARQ) and Forward Error Correction (FEC) tools.

Investigations carried out by Chakravorty et al [26] using User Datagram Packet (UDP) streams over GPRS show that Round-Trip-Times (RTT) of about a second can be commonplace, with the latency being both large and highly variable. Incidences of bursty data result in large jitter on the first packet transmitted, as the link transits from idle to active, while subsequent packets incur minimal jitter. Packet loss and reordering are found to be rare, due to the RLC and FEC tools, but link outages are not uncommon, particularly for actively mobile units.

The Transmission Control Protocol is one of the most important protocols on the Internet, providing reliable transport flow and congestion control to applications requiring a connection-oriented service [27]. TCP employs numerous control mechanisms in order to provide these services, and utilises information such as RTTs, congestion windows and packet loss to do so. These mechanisms are considerably less effective however, in a GPRS environment.

The performance of TCP over GPRS is affected by the high RTTs experienced. The investigations show that the TCP Slow-Start (SS) phase transits into the Congestion Avoidance phase during steady-state testing, because of receiver window limitations. Also, because GPRS devices can operate in a half-duplex mode, the packet ACKs for multiple received packets to be compressed into a single transmission from the device. This results in subsequent data being transmitted to the device in a bursty manner, as numerous ACKS are received in intervals.

TCP operating over GPRS also experiences excessive queueing, because of GPRS buffering before the bottle-neck of the air interface. The buffering prevents packet loss, which TCP uses as an indicator to reduce the transmission rate (ie packets are being dropped because links are full), but also delays packet delivery, so increasing the RTT. The excessive queueing leads to other problems, such as the RTT inflation blocking new connections because of connection requests timing out, buffered data becoming stale but still clogging the buffer, and inflated

TCP Retransmit Timer values. The occasional link outages also cause problems by freezing transmission across the link during the break, which may lead to TCP Time-Out retransmissions, which are subsequently rendered useless when the buffered data is finally delivered [26].

The performance problems experienced by TCP operating over GPRS, through high RTT times and excess queueing, are significant. Chakravorty et al propose a solution using a transparent proxy server, which removes the need for modification to either party. The proxy is ideally co-located with the gateway between the GPRS network and the Internet, the Gateway GPRS Support Node (GGSN).

By splitting the TCP connection into "wired" and "wireless" sections, the proxy is able to manage the two TCP sessions, and interact with each session to match their requirements. It does this using a fixed-size congestion window on the GPRS-side, to keep the incoming data rate in line with the GPRS transmission capabilities. By doing this, the TCP Slow-Start is eliminated, and the data from the wired link, queued at the proxy, can be controlled by judiciously rewriting the Receiver Window in the ACKs from the wireless link. Thus, the proxy controls the TCP-leg to conform to the limits of the GPRS network, by manipulating some of the TCP features involved in the initial problem.

This proxying of data at the ingress to the GPRS network is similar to the role the RLS plays, by managing traffic in the network before forwarding it to the client. The result is a data stream better suited to the access network (in this case GPRS), and the reduction of possible excessive loads. It also provides a second stage of data management for RLS-aggregated data passing from the IMS core to a GPRS access network. The benefits of this are discussed in Section 3.1.3.1.

Experimental evaluations of the solution show that it succeeds in reducing the excessive queueing, as well as RTT inflation and the Retransmit Timer values. The TCP SS is avoided, improving performance - particularly for small amounts of data. Importantly, the limiting of the data rate keeps the buffering of data for transmission over the air interface to a minimum, leading to faster buffer drain times, and hence fewer retransmissions due to transmission time-outs. The use of the transparent proxy leads to a significant improvement in the overall throughput over the GPRS network [26].

2.2 The Session Initiation Protocol (SIP)

2.2.1 SIP in Presence Services

The Session Initiation Protocol was chosen by the IETF as the protocol for encapsulating and transporting presence information on the Internet in RFC 3856 [28]. The IETF utilised the SUBSCRIBE and NOTIFY request messages added to the general event notification framework described in RFC 3265 [29] ("Session Initiation Protocol (SIP)-Specific Event Notification") to enable Presence Agents (PAs) to upload and receive presence information to and from PSs or

RLSs. The presence information is transported across the network as the body of the SIP request.

Because the initial specifications for presence utilised SIP, it was natural that other aspects of the service would also be based on the protocol. The "Functional Description of Event Notification Filtering" in RFC 4660 [30] describes the operations required to put into place desired filtering rules for an event notification subscription, using SIP to transport the XML-encoded filter rules. The "Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists" in RFC 4662 [31] extends the Specific Event Notification framework, to enable the subscription to homogeneous lists of resources. A template event-package for watcher information is defined in RFC 3857 [32] ("A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)"), and facilitates the monitoring of events in the watcher information event-package. The specification defines a template event-package, and as such can be applied to any event package, including itself.

SIP was chosen as the signalling protocol for the IMS in its early development stages. It is then natural that SIP continues to be used in the definition of IMS presence standards. The functionality of the IETF presence standards is introduced into the IMS in the 3GPP Technical Specification Group Services and System Aspects' Presence Server, Stage 1 (Release 7) [33], and extended in the 3GPP Technical Specification Group Services and System Aspects' Presence Service - Architectural and Functional Description (Release 7) [34] and 3GPP Technical Specification Group Services and System Aspects' Presence Service - Using the IP Multimedia (IM) Core Network (CN) Subsystem, Stage 3 (Release 7) [21].

The constraints in cellular devices and wireless networks have forced IETF and 3GPP engineers to optimise the amount of presence information sent to PUAs. Solutions such as the use of RLSs, event notification throttling, control and event filtering, while applicable to any event notification framework, are particularly useful in presence implementations [1]. Other optimisations such as Partial Notification are described in Section 2.3.3. These extensions which optimise presence are implemented by all IMS networks and terminals [3].

2.2.2 SIP Header and Message Compression

The Session Initiation Protocol is based in part on the extremely successful and popular HTTP 1.1 Request-Response model. HTTP has been largely responsible for popularising the World Wide Web, and so it made sense to adopt the aspects thereof which could be reused. The use of HTTP as a base for SIP implies that much of the SIP header and message syntax is identical to that of HTTP 1.1. Like HTTP, SIP is text-based, allowing the headers to be human-readable, and making the use and debugging of SIP signalling simple [19]. The text-based nature of the messages does however mean that SIP headers are large, particularly when the amount of routing and authentication information that may be included is considered, as shown in Figure 2.1. SIP signalling exchanges may also feature large amounts of repeated or predictable information, which waste scarce bandwidth resources.

```
REGISTER sip:homel.net SIP/2.0
Via:  SIP/2.0/UDP [1080::8:800:200C:417A];comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards:  70
P-Access-Network-Info:  3GPP-UTRAN-TDD;
                        utran-cell-id-3gpp=C359A3913B20E
From:  <sip:alice@homel.net>;tag=s8732n
To:  <sip:alice@homel.net>
Contact:  <sip:[1080::8:800:200C:417A];comp=sigcomp>
          ;expires=600000
Call-ID: 23fi571ju
Authorization:  Digest username="alice_privateShomel.net",
                realm="homel.net", nonce="",
                uri="sip:homel.net", response=""
Security-Client:  ipsec-3gpp; alg=hmac-sha-l-96;
                  spi-c=3929102; spi-s=0293020;
                  port-c:3333; port-s=5059
Require:  sec-agree
Proxy-Require:  sec-agree
Cseq:  1 REGISTER
Supported:  path
Content-Length:  0
```

Figure 2.1: SIP REGISTER Message with Additional IMS Headers [1]

The SIP standard provides a mechanism which allows the representation of common header field names in an abbreviated form [19]. For example, the "Contact:" header field may be replaced with "m" (for "moved"), and the Call-ID header field replaced with "i". This does reduce the size of the SIP header to some degree, but most information in the header is still present in the form of routing information, session and dialogue data for the protocol.

Another solution to the problem of very large SIP messages is to use compression, which is applied to the entire message. The IETF addresses this problem through the Robust Header Compression (ROHC) Working Group, whose goal is to develop generic header- and signalling-compression schemes that perform well over wireless and cellular links such as WCDMA and EDGE, or links with high error rates and long round-trip times [35].

The ROHC has since its inception developed a large number of RFC standards that deal with signalling compression; with RFC 3095 specifying a highly robust and efficient header compression scheme for RTP, UDP and IP [36], and updated with corrections and clarifications in February 2007, by RFC 4815. However, RFC 3095 deals only with the compression of message headers, leaving the compression of signalling to RFC 3320, Signalling Compression (SigComp) [37]. SigComp provides a solution for compressing the messages generated by application protocols like SIP and the Real Time Streaming Protocol (RTSP). SigComp details the architecture and prerequisites for use, as well as the message format required. SIP entities must indicate to their corresponding party when SigComp compression is desired for messages, and when it may be appropriate to send compressed SIP messages. The entities can request SIP compression using the mechanism specified by RFC 3486, which defines how to request SIP message compression, as well as how and when to implement it [38].

2.3 XML in Web Services

The Extensible Markup Language (XML) is a meta language defined as a subset of Standard Generalised Markup Language (SGML), used to describe documents containing structured information. XML does not specify semantics or a tag set, but instead allows a user to uniquely define the grammar of a document by specifying new tags and the structural relationships between them, within the context of a structured document [39]. This extensibility means XML can be fitted to almost any structured information, making it very useful for sharing structured data over the Internet [40].

The specification of XML is being driven by the World Wide Web Consortium (W3C), an international union of member organisations, through numerous different specialised working groups. A list of ten design goals were drawn up for the standardisation of XML, in order to keep the standard widely applicable and suitable for as many applications as possible. The goals mandate that the standard is easy to use, straightforward, and importantly, human-legible and reasonably clear. The last goal states that terseness in XML markup is of minimal importance [40], [41], encouraging descriptive tag specification.

It is a combination of these design goals that cause XML to become an inefficient scheme for data encoding.

2.3.1 Data Coding Efficiency of XML

By aiming to support a wide variety of applications, XML encoding efficiency cannot be optimised, compared to when designed specifically for a small set of desired applications. The requirement for human legibility and reasonable clarity in XML document encoding means that tags are written in clear text. While this allows high readability, it also incurs high levels of redundancy. Additionally, the final design goal, "Terseness in XML markup is of minimal importance" [41], encourages the use of descriptive and accurate markup, adding to the encoding overhead through highly detailed or verbose markup. The descriptiveness and redundancy of the added XML can become a substantial overhead to the data to be transmitted [39].

Having recognised the inefficiencies of the normal text-based encoding scheme, the W3C has specified an additional XML standard that uses binary encoding to reduce the encoding overhead. The standard is being driven by the Efficient XML Interchange (EXI) Working Group (EXIWG) [42], which has identified Efficient XML [43] as the basis for the new encoding specification. The specification aims to provide a very compact representation of XML, for use over low bandwidth links. The trade off however, is that the new format sacrifices some of the interoperability of the main XML specification. At the time of writing, the EXIWG has released a Last Call Working Draft.

To illustrate the point of XML encoding redundancy, Tian et. al. [44] provide a comparison between an application implemented as a Web Service that uses XML, and the same application implemented using Active Server Pages (ASP). The server application returns stored information

for a book, when queried by a client, using the books International Standard Book Number (ISBN). Information for a book, which comprises 589 bytes, is requested from both applications. The ASP application adds an overhead of 593 bytes to the requested information, while the Web Services implementation, using XML, adds 3363 bytes overhead. Therefore, the ASP application increases the transmitted data size by 100%, while the increase in transmitted data with the Web Service is 570%. In another comparison of XML and other schema for data encoding, data represented in XML is on average 400% larger than the same data represented in binary format [45].

The overhead added by the use of XML is significant in the context of encoding presence information. The amount of real data in a presentity can often be quite small, but highly structured. Declaring the structure of presence data in XML can add a significant overhead to the information to be transmitted, and as shown previously, can be many times greater in size than the un-encoded information.

2.3.2 XML Compression

The encoding overhead incurred by XML is a drawback of using such a multifunctional, extensible data encoding format. A simple way to mitigate the impact of the overhead on encoded data size and transmission times, is to compress the XML data. Numerous general-purpose data compressor implementations are available, and fall into one of two classes; arithmetic- and dictionary-based [46]. The respective classes are characterised by different benefits and drawbacks, such as memory requirements during operation, and execution times.

In order to decrease XML file size, and the processing required at remote nodes, XML can be encoded into binary formats (as binary data). While this reduces the file size, it also reduces the scope for interoperability between different platforms, contrary to the specifications design goals [41]. The driver for binary XML formats is the use of XML in wireless devices, which typically have restricted bandwidths, and the use of XML in large databases, where encoding redundancy affects data storage efficiency [46]. While a number of different binary XML-encoding schemes exist, an investigation of these formats is beyond the scope of this overview.

Binary-format XML is not compressed, allowing possible random access or queries to the data to be made. Compressed XML, while also in a binary format, can not offer these functions directly, instead requiring decompression first. XML-specific compression algorithms take advantage of structure in the XML, and assumptions and likelihoods of similar entropy between similarly named elements [46].

Augeri et. al. [46] provide an analysis of the compression efficiency of a number of both general-purpose and XML-specific data compressors. The analysis examines the efficiency of each compressor utilising a metric, which combines the compression ratio achieved and the execution speed of the method. Fourteen different compressors are considered in the examination, operating on 44 test files, ranging from 1kB to 4MB, with one large 40MB file. The thorough statistical analysis performed on the collected data showed that an XML compressor

offers the best combined efficiency. However, its performance is statistically similar to three of the general-purpose compressors.

The authors findings from the investigation are that in most instances, a general-purpose compressor is suitable and offers similar performance to XML-specific compressors. They also found that the binary formats worked particularly well on small XML files. The analysis did not assess decompression performance, nor the memory requirements of the compression algorithms. The study was not able to examine Efficient XML - the binary XML format specified for use by the EXIWG - because a publicly accessible version was not available at the time of the analysis.

Even with the EXI standards being released and implemented, it is likely that the compression of XML for transmission and data storage will continue to be researched and monetised in various forms [39].

While there are considerable efforts underway in developing and evaluating XML compression algorithms, the computational cost of the compression cannot be escaped. Compression may incur a significant computing cost, particularly on devices with constrained computational resources such as mobile telephones. Such devices are quite likely to be limited in terms of processing speed, available memory, and battery power.

M. Tian, et. al. [44] present a study of the effect of XML compression in applications implemented as Web Services. Compression is examined with regard to both the amount of data transmitted across a wireless network, and the additional computational load it imposes on the server and client. The authors propose a system with tiers of compression service available to clients accessing a server application, in order to reduce the computational load and maintain server performance. Clients have the option to refuse compression, receive compression if available, or always receive compression. This enables the server to provide XML compression to those who specifically request it, or request it if available. If server utilisation goes above a given computational load threshold, compression is provided only to those clients who explicitly request it, while below the threshold the server may provide it to clients who request optional compression.

Testing showed that the compression of all the messages reduced the throughput of the server significantly as the load increased, compared to no compression being used. The proposed framework to provide compression dynamically to those who requested it or expressed no preference, approached the performance for when no compression was used at all. The framework was able to increase throughput, while maintaining the requested service. The compression was however costly on the client-side, with an increase in the computational delay, which may invalidate the transmission gains attained through compression.

The transmission times for compressed and uncompressed data over an IEEE 802.11b WLAN, Bluetooth and emulated GPRS network are also examined in the study. Results obtained indicate there is little difference between the transmission of compressed and uncompressed data over the WLAN and Bluetooth network. Across the emulated GPRS network however, compressed data transmission outperformed the transmission of uncompressed data. Furthermore, the

performance of compressed data transmission over the GPRS network with poor connectivity actually outweighed the computational costs incurred by the compression.

2.3.3 XML in Presence Standards

The IETF was the first standardisation body to begin releasing widely used specifications for presence on the Internet. Other standardisation bodies, such as the 3GPP, have to a large degree followed the IETF standards in the development of their own specifications.

The IETF set a foundation by defining abstract models and requirements for presence and IM, which at the time were considered interlinked [47], [48]. The specifications described entities, terminology and a standard protocol which enabled inter-operation between different implementations of IM and presence services. The foundation-laying protocols were refined in August 2004, with the Common Profile for Presence (CPP) [49] defining common semantics and data formats, to facilitate gateways between different presence implementations. The CPP identified the Presence Information Data Format (PIDF) [2] as the basic presence format to be supported. Any presence application wishing to send SIP NOTIFY messages through a CPP gateway must support the PIDF format, while optionally supporting other content formats.

The PIDF specification provides a common presence data format for use with CPP-compliant presence protocols. The data format chosen for use in the PIDF is XML, because its features, such as hierarchical structure and easy extensibility, make it the most suitable framework for encoding presence information. The specification defines a number of required XML elements and their structural relationships, in order to interact with CPP-compliant systems. Extensions are also possible, allowing for additional elements to be added to the presence information structure later [2].

The presence information structure specified by the PIDF is however, relatively rigid. The Rich Presence Extensions to the Presence Information Data Format (RPID) [50] seek to rectify that, by introducing a number of additional XML elements. Information contained in a presentity for human consumption can be conveyed in the XML `<note>` element, with the contextual meaning easily conveyed to the user. The receiving client application however, is not capable of interpreting this, and so the additional elements are introduced. The additional elements extend the information capabilities of the presence data format to match those of contemporary applications at the time of RPID publication (July 2006). The task of populating these additional fields is envisaged to be handled automatically by either the users communication devices, or integrated network service provider. The RPID equips the presence application with the tools required to create a more comprehensive and active presence service to subscribers, but the cost of this is the addition of considerable encoding redundancy to the presence payload.

The full set of presence information for a presentity can be considerable, particularly if multiple, rich presence tuples are included. The Publication of Partial Presence Information [22], [23] specification leverages the hierarchical structure of XML to reduce the data that must be transmitted to the PS each time an aspect of a presentity is updated. Transmitting the entire set of

presence information from the presentity when only a small change occurs is highly inefficient and costly, particularly over low-bandwidth and high-latency networks.

Presence information is uploaded by the presentity to a PS, and watched by one or more watchers. While serving presentities to watchers, the PS also provides information to the presentity regarding the watchers of its presence information. The Extensible Markup Language (XML)-Based Format for Watcher Information [51] specifies the use of XML for representing this information, as it is also typically highly structured. The format allows watcher lists for multiple resources to be included in the information. The standard specifies mandatory elements in the watcher list, as well as some optional elements, for the sake of completeness. The watcher data transmitted to the presentity typically only notifies the presentity of changes to the list of watchers and watcher information, reducing the data overhead. This means that the watcher information updates must be combined over time in order to get a comprehensive view of the resources watchers.

The instances of XML used in the presence service outlined above, form a subsection of the web application space in which XML is used to transmit data between entities in the network. While XML encoding adds redundancy and increases the data transmitted, its hierarchical nature allows subsets of information to be transmitted, in particular when updates are made to presence or watcher information. XML is also used internally in applications for representing data in certain other presence applications.

Data such as resource lists, and their presence authorisation rules, are documents containing considerable amounts of structured information. The IETF acknowledged this by specifying the use of XML for representing this data [52], [53]. These documents are typically not exchanged across the network by the network entities that manage the subscriptions to the RL (such as an RLS), or enforce the authorisation rules (typically a PS), but may be updated or built up over time by the subscriber.

Another example of stored XML documents in the presence framework is the encoded data used in managing event notification filtering [54]. By using event notification filtering, a watcher is able to specify which changes to a presentity, or presentities, cause an update to be sent from the repository, such as an RLS or PS. Because a watcher may specify different update parameters for individual presentities while watching a considerable number of presentities, the notification filtering information requires an hierarchical structure. It is because of this that a watcher can configure specific filtering elements or attributes in the XML differently for different presentities, or groups of presentities.

The above examples of server-side XML-encoded documents do not present the challenge of information with high encoding redundancy being transmitted across the network. While there may be concerns regarding efficient storage of the information, that is beyond the scope of this discussion.

In order for service subscribers to upload and incrementally build the stored documents, the XML Configuration Access Protocol (XCAP) [55] is specified. XCAP allows clients to read, write and modify their information using HTTP. XCAP specifies a set of conventions that enable the

mapping of XML documents and document components to HTTP URIs, as well as rules for how modifications affect other elements and resources. This allows modifications to the information to be made through a web browser, or through other means.

2.4 Techniques for Presence Traffic Optimization

This final section of the literature review examines techniques specifically put forward for the optimisation of presence traffic in networks. The principle and intended use of the RLS for this has been outlined and described in both relevant standards [21, 34] and research [3, 56]. While the intended use of the RLS is clear in the literature, no body of work dealing specifically with the manner in which the RLS carries out the aggregation was found.

Singh et al [56] provide a number of different possible solutions in their examination of traffic optimization techniques. Among their proposals are the Partial Publication capability of XML (Section 2.3.3) and the use of an RLS, although the RLS is only considered as a subscription manager, rather than an entity for performing aggregation from within the network to reduce the volume of NOTIFY updates to the PUA.

Many of the proposed actions for traffic optimization place the onus on the subscriber to manage or control aspects of the service operation. This is the case for both Conditional Subscriptions and Watcher Filtering. These techniques allow the subscriber to specify for instance whether they should receive updates when subscriptions to presentities are renewed, or to specify which particular presentity fields will trigger a NOTIFY to be forwarded to the PUA. These functions can be implemented at the subscribed presentity's PS, but just as easily at an RLS.

The additional techniques examined by Singh et al [56] focus on the management of the presence information within the core network between domain entities, rather than between a core network entity and the PUA. These methods include the use of Common NOTIFYs for Multiple Watchers, whereby a single NOTIFY is sent from Domain A to Domain B, at which point it is disseminated to each watcher of the presentity in Domain B. This mechanism effectively acts in manner opposite to that of the RLS, by concentrating the subscription of a number of watchers in Domain B into a single subscription for a presentity in Domain A, and disseminating the returning data to the numerous watchers at the network edge. Also examined is the use of aggregated NOTIFY messages between core network entities in different domains. The NOTIFY messages from Domain A are aggregated before being transmitted to Domain B, at which point they are either disseminated to multiple watchers, or delivered to a single PUA watching all of the presentities aggregated in the NOTIFY message.

Calculations highlighting the benefit the techniques provide for the management of presence traffic are provided, as well as traffic flow diagrams between the respective entities. However, there is no evidence of efficacy obtained through simulations or experimental deployments provided. While the techniques examined focus specifically on presence traffic management, the application is limited to intra-domain traffic management, rather than on reducing the volume of presence data passing through the access network.

Chapter 3

Design Considerations

As explained in Chapter 1, presence applications in the IMS use SIP messages to carry the XML-encoded presence information from the Presence Server to PUAs through the IMS core, across the myriad of available wired or wireless access networks, that are likely be bandwidth limited. The use of SIP and XML for encapsulating and transporting this information often leads to very large message sizes, compared to the raw presence information encoded.

The Literature Review in Chapter 2 discusses some methods for reducing the size of the message transmitted to the PUA, and proposes implementations to improve the effectiveness of higher layer network protocols. These solutions affect the size and routing of each individual message, but do not perform any management action on the stream of presence information which these messages collectively form.

This chapter examines the operation of the RLS in terms of the functional role it performs in the presence framework, as well as the particulars of its internal operation. This includes discussion of the various aggregations methods, analytical metrics and the testing methodology employed.

3.1 Functionality of the Resource List Server

The Resource List Server can implement some of the schemes described to reduce the size of SIP messages in presence applications, and can operate over MPLS and GPRS networks with the improvements and additional entities described in Chapter 2. However, the real benefit gained from the RLS in terms of traffic management is not obtained via the described methods, but rather through the aggregating action performed on the stream of presence messages for a PUA's Resource List (RL) subscription. The RLS manages the streams of presence information subscribed to by the PUAs through the restructuring of loads and reductions in the size and amount of traffic, using the aggregation methods outlined in Section 1.3. These methods form part of the thesis objectives, as the IETF and 3GPP specifications describing the presence framework in the IMS describe the functions of the RLS, while providing no details regarding the operation nor implementation of the server [31, 21].

By managing and aggregating presence streams, the RLS is able to reduce the impact of fluctuations in the packet stream transmitted to the PUA by PSs. This management can allow resource allocation algorithms at the ingress of access networks to provision resources more effectively. This in turn can reduce the frequency and magnitude of any limitations in the network or available bandwidth to other applications utilising the wireless interface simultaneously, many of which may require stringent QoS guarantees. The management and aggregation of the data stream from the PS to the PUA via the RLS is the primary focus of this work.

The role the RLS plays in managing presence subscriptions for PUAs from within the IMS Core Network (CN), is also very important. By storing RLs and managing subscriptions for PUAs, the RLS is able to remove significant amounts of SIP signalling from the access network, replacing the PUA's subscriptions to numerous presentities with a single subscription to the PUA's RL. While subscription signalling from a PUA to a PS typically does not carry any XML-encoded payload, it has to be repeated for each presentity subscribed to, hence the volume of data increases linearly with the number of subscribed presentities. Without the intervention of an RLS, the requested presence information from numerous presentities is returned to the PUA by the respective PSs individually. With no traffic management in place, this results in a large spike in data traffic to the PUA. It is preferable to manage such data before it reaches the access network, as solutions such as the management of TCP traffic over GPRS are designed for steady-state transmission, which is unlikely to occur under this scenario.

The RLS in IMS presence applications reduces the volume of presence traffic across the access network through two mechanisms, as outlined previously. The latter mechanism transfers the repetitive SIP subscription signalling between the PUA and PSs from occuring across the access network, to occuring between the RLS and PSs, through the IMS core network. This is achieved by using the RLS to store and manage RLs of presentity URIs on behalf of subscribing PUAs. A PUA, accessing the presence and RLS service, subscribes to their RL held by the RLS, which then subscribes to each presentity contained therein. As the subscriptions are accepted and processed by the receiving PSs, they transmit the subscribed presentities to the RLS, as it remains on the signalling path acting as an aggregating presence proxy. Presence stream management begins at this point, which is described further in Section 3.1.2.

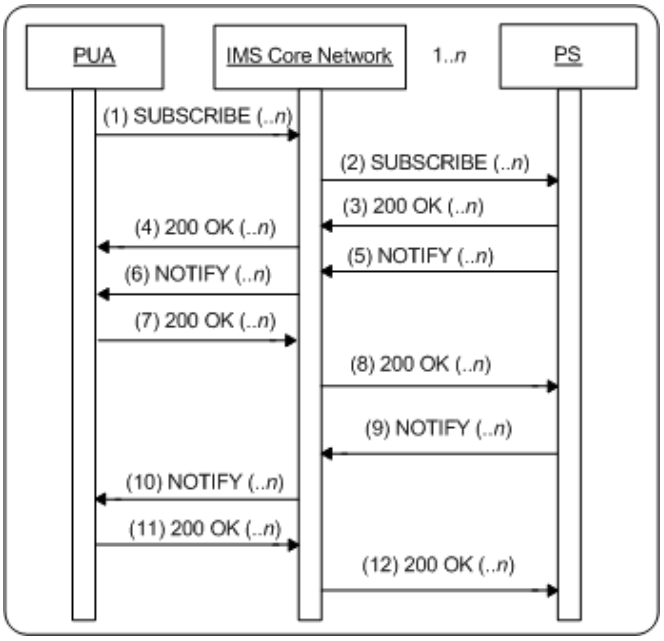

Figure 3.1: PUA Subscription to n Presentities, without the use of an RLS

3.1.1 Transfer of Signalling to IMS Core Network

In the case of a PUA wanting to subscribe to a number of presentities of interest (for instance, n presentities) without the services of an RLS, it will need to send n separate SIP SUBSCRIBE messages to PSs managing the presentities. The n resulting SIP dialogs must then be managed, with the PS providing the subscribed presentity information. This is shown in Figure 3.1. Furthermore, any updates made to any of the presentities necessitates the sending of updates to the PUA, which are transmitted individually across the access network. It is clear that there is a considerable amount of repetitive SIP signalling that must traverse the access network, as well as the transmission of NOTIFY messages carrying XML-encoded presence information.

The RLS mitigates this waste of resources by transferring the repetitive subscriptions to the IMS CN, which is expected to be amply provisioned with bandwidth. The PUA subscribes to their RL held by the RLS, which then performs all the required subscription signalling on the PUAs behalf. The RLS remains on the signalling path from the PS to the PUA, allowing for the aggregation of the presence information stream passing through it. This modification to the signalling sequence is shown in Figure 3.2.

3.1.2 Presence Stream Management

Having moved the subscription signalling to the RLS in the core network, the inefficient use of resources by the presence data stream returning from the PSs can be addressed. The volume and load profile of the presence information returned by PSs hosting subscribed presentities are relatively unpredictable. The response by a PS to a new subscription is to provide all the

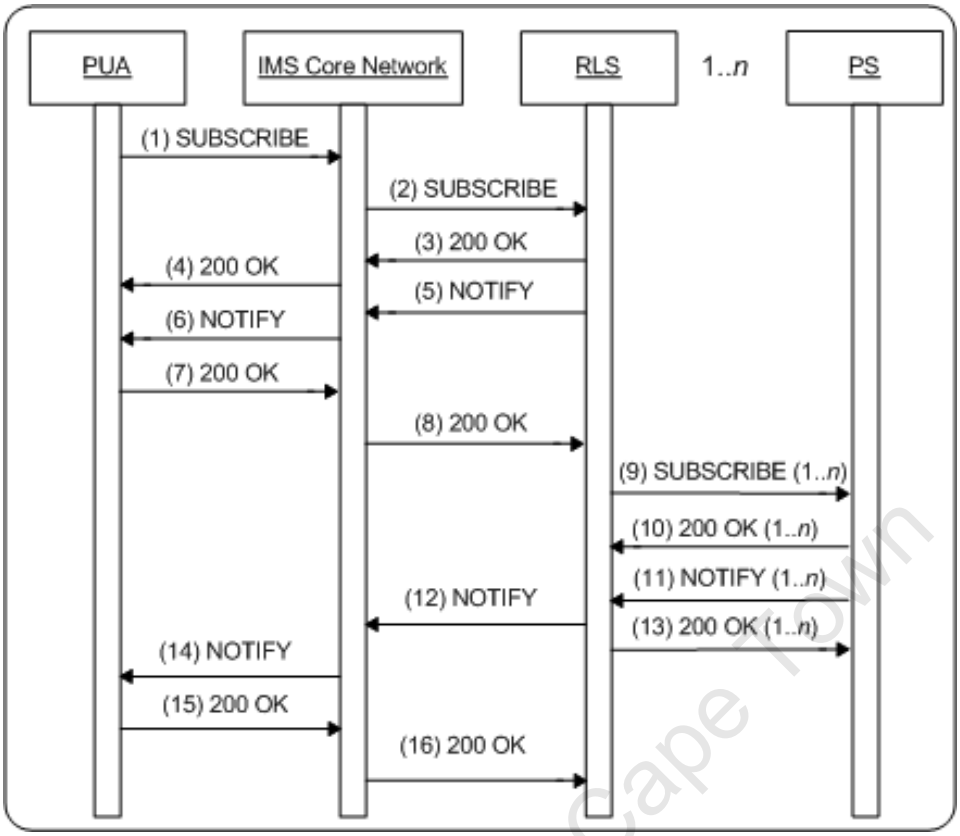

Figure 3.2: Subscription to a Presentity, Using a Resource List Server

information stored for the subscribed presentity, encoded in XML. Once the "full" presentity has been provided to the PUA in response to the subscription, updates are generated if and when aspects of the information for a presentity change. An example "full" presentity is provided in Figure 3.3. The presentity "someone@example.com" has two tuples. In the first, the IM status is set to "busy" and location to "home", while notes in French and English are included, asking not be bothered. The second tuple sets the email address "someone@example.com" as the primary address for contact, and includes personal itinerary information in a note.

Because an RL normally contains numerous presentities of interest, a significant amount of data may be received by the PUA over a very short period of time after the initial subscription. After this initial information has been delivered, updates to the presence information are generated relatively randomly by the subscribed presentities. The use of XML for the encoding of the presence information allows for only the specific information updates to be transmitted to the PUA, as described in Section 2.3.3, and provided as an example in Figure 3.4. In the partial update, a tuple is being set to "open" and updated with a new medium for communication (the telephone number "+09012345678"), and so only this information is included in the message.

It is this stream of presence packets from PSs through the RLS that is of primary interest for aggregation and management. The stream can be considered to be dynamic, with a variable message arrival rate, and fluctuations in the size of the updates. This variation is governed partly

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
               xmlns:im="urn:ietf:params:xml:ns:pidf:im"
               xmlns:myex="http://id.example.com/presence/"
               entity="pres:someone@example.com">
   <tuple id="bs35r9">
      <status>
         <basic>open</basic>
         <im:im>busy</im:im>
         <myex:location>home</myex:location>
      </status>
      <contact priority="0.8">im:someone@mobilecarrier.net</contact>
      <note xml:lang="en">Don't Disturb Please!</note>
      <note xml:lang="fr">Ne derangez pas, s'il vous plait</note>
      <timestamp>2001-10-27T16:49:29Z</timestamp>
   </tuple>
   <tuple id="eg92n8">
      <status>
         <basic>open</basic>
      </status>
      <contact priority="1.0">mailto:someone@example.com</contact>
   </tuple>
   <note>I'll be in Tokyo next week</note>
</presence>
```

Figure 3.3: Full Presentity, Encoded in XML with Status Extensions [2]

by how many presentities are being watched in the RL, and by the nature of the presentities being watched. The manner in which the RLS manages this information, while maintaining timely delivery, is particularly important. We investigate three primary methods, and two secondary methods which comprise combinations of the three.

The motivation behind the investigations into these methods is to gain an understanding as to their efficacy in the management of the data streams. The constraints and parameters affecting and categorising subscriptions may have a marked effect on how effectively a subscription is handled by the RLS. The results obtained show that aggregation parameters which are poorly matched or incorrect for the data stream under consideration yield aggregated data with little to no reduction in data volume, effectively negating the desired effect of the RLS. In the interests of completeness, a number of parameters for each aggregation method are examined.

```
<?xml version="1.0" encoding="UTF-8"?>
<impp:presence xmlns:impp="urn:ietf:params:xml:ns:pidf-diff"
                        entity="pres:someone@example.com">
   <impp:tuple id="sg89ae">
      <impp:status>
         <impp:basic>open</impp:basic>
      </impp:status>
      <impp:contact priority="0.8">tel:+09012345678</impp:contact>
   </impp:tuple>
</impp:presence>
```

Figure 3.4: Partial Presence Update in XML

3.1.3 Aggregation Methods

The aggregating methods under examination, as stated in Chapter 1, are time-based, quantity-based and size-based aggregation. Combinations of the methods; time-quantity-based, and time-size-based aggregation, are also examined. Although the implementation of each method is different, they all utilise the same basic mechanism to manage the data stream. When a NOTIFY message is received by the RLS, the XML payload is stripped from the message and stored by the RLS. As more NOTIFY messages are received, the process repeats, and a store of presence information is accumulated. This aggregated information is then forwarded to the PUA in a NOTIFY by the RLS, when the aggregation method in use determines that its conditions for execution have been met. The removal of the SIP headers by the RLS during the management of the presence stream significantly reduces the amount of data added by the headers of an unmanaged presence stream, for transmission over the access network. Figure 3.5 is an example SIP NOTIFY message, with no presence payload. It is noted that the header is considerably larger than the partial update given in Figure 3.4.

The parameters of interest for the aggregation methods described below are the relationship between the size of the RL, the relative activity of the subscribed presentities within the RL and the aggregation parameters examined. Of particular importance are the bounds at which the operation of the RLS becomes ineffective. Parameter values may be too small in comparison to the data stream to effectively reduce the data traffic across the access network, or be too large, such that long delays and large message sizes result, negatively effecting the delivery of the proxied presence information.

```
NOTIFY sip:192.168.1.100:5080 SIP/2.0
Method:  NOTIFY
Call-ID: ca27add4c7b43d1a2420cf49f12c8bed@192.168.1.100
CSeq:  2 NOTIFY
From:  <sip:A1@192.168.1.100:5071>;tag=1
To:  <sip:subs1@127.0.1.1:5060>;tag=9079905777982525742_cc02fe1c29479
Via:  SIP/2.0/UDP 127.0.1.1:5071;branch=z9hG4bK-14967-1-3_15281
Contact:  <sip:127.0.1.1:5071;transport=UDP>
Max-Forwards:  70
Authorization:  Digest username="alice_privateShomel.net",
                realm="homel.net", nonce="",
                uri="sip:homel.net", response=""
Security-Client:  ipsec-3gpp; alg=hmac-sha-1-96;
                  spi-c=3929102; spi-s=0293020;
                  port-c:3333; port-s=5059
Require:  sec-agree
Proxy-Require:  sec-agree
Expires:  1800
Subscription-State:  active
Event:  presence
Content-Type:  text/plain
Content-Length:  315
```

Figure 3.5: SIP NOTIFY Message

3.1.3.1 Time-based Aggregation

By using a timer in the RLS to determine when stored information is forwarded to the PUA, the variable arrival rate of NOTIFY messages is replaced by highly regular updates, containing the XML-encoded presence information of all presentities received during the time period. The timeliness of information delivery is easily maintained through well-chosen time parameters, although large time values will increase the likelihood of information at the PUA becoming stale. Because time is the only parameter considered, the size of the accumulated presence information may become significantly large, especially with a sizable RL, or if the subscribed presentities are particularly active. This effect is highlighted, with a proposed solution for the ingress of GPRS networks, in Section 2.1.1. The secondary methods, examined in Sections 3.1.3.4 and 3.1.3.5 are intended to circumvent this problem, and ensure the timely delivery of reasonably sized messages.

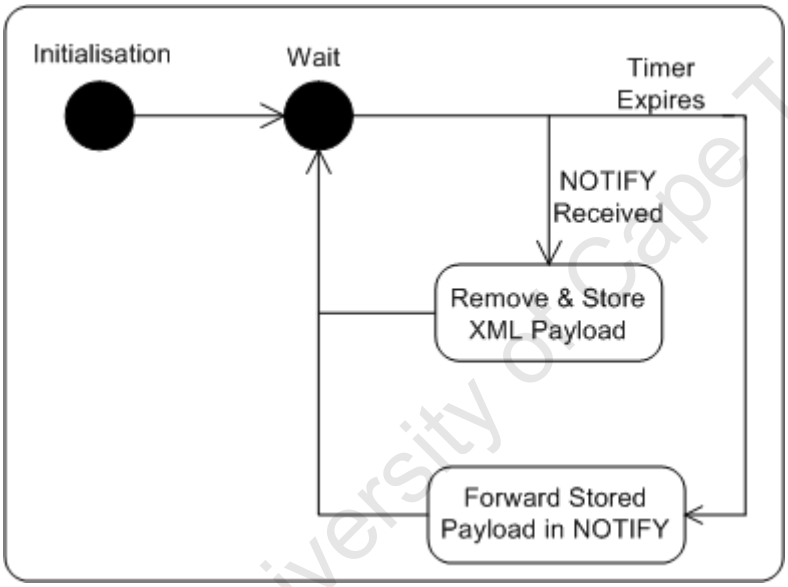

Figure 3.6: State Chart for Time Aggregation

The Unified Modelling Language (UML) statechart diagram in Figure 3.6 depicts the logical operation of the RLS operating using time-based aggregation. The RLS defaults to the Wait state, waiting either to receive NOTIFY messages for the PUA, or for the periodic timer to expire. When the timer expires, the RLS encapsulates the presence information accumulated in a SIP NOTIFY message, which is then forwarded to the receiving PUA. After doing this, the stored information is cleared, and the subscription again enters the Wait phase.

Time-based aggregation is best implemented with the timer being logically separate from the mechanism to receive, store and forward NOTIFY messages from the PS to the PUA. This means that the timer should interrupt the receiving of messages to facilitate the forwarding of the stored information, rather than the RLS checking a timer when a message is received. Such a mechanism of checking the timer status is susceptible to failure, through a lack of messages being received at the RLS, or causing the irregular forwarding of notifications. As it is likely

that only one, or a limited number of timers (in the form of a limited pool of periodic threads, for instance) would be used, all subscriptions held by an RLS will be serviced at the same time. Congestion is unlikely to be a significant problem as the core network is expected to be provisioned with ample bandwidth.

3.1.3.2 Quantity-based Aggregation

Quantity-based aggregation uses the number of presence information updates received from PSs destined for the PUA as the basis for triggering an outgoing NOTIFY message. Unlike the time-based aggregation method discussed, the timeliness of presence information delivered using the quantity-based aggregation is not guaranteed. The interval between NOTIFY messages sent from the RLS is dependent on the size of the RL, and on the level of activity of the watched presentities.

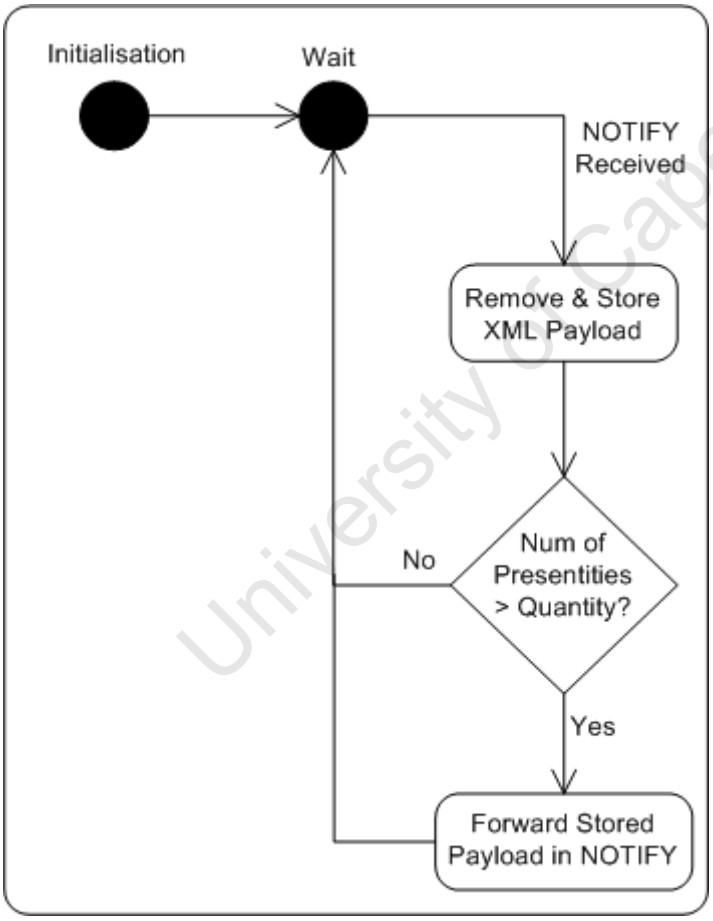

Figure 3.7: State Chart for Quantity Aggregation

Operationally, quantity-based aggregation differs from time-based aggregation in that the trigger to actively forward the accumulated information is checked every time a NOTIFY message is received, rather than treated as an interrupt causing the transition from the Wait state. Once a NOTIFY message is processed, the RLS checks to see how many presentities are stored for the

subscription, and either returns to the Wait state or forwards the stored information and clears the information store. This is shown in Figure 3.7.

The larger the RL is, the more likely it is that at any given time, a subscribed presentity will generate an update to its information. More active updaters (for instance, teenagers or travelling salesmen updating their location status) subscribed to by an RLS will help reduce the time between NOTIFY updates. On the other hand, an RL of presentities that seldom updates will lead to long periods between NOTIFY messages being sent to the PUA, and almost certainly result in stale information being held by the PUA.

Because the number of presentities carried in NOTIFY message from the RLS is limited, the size of the message is also relatively uniform. There will be a degree of variation in the size of the XML payloads received, which will carry through to the size of the payload forwarded to the PUA. The size of the payload is however unlikely to approach the worst-case scenarios of time-based aggregation payloads, but stays within reasonable upper and lower bounds.

Updates to the PUA by the RLS are generated when the number of NOTIFY messages received reaches a predetermined threshold. The implementation of this in the RLS can lead to significant differences in how the number of NOTIFY messages for a subscription is calculated. If, for instance, a second (or more) update is received from a presentity before the threshold value is reached, it should overwrite the currently stored information, if the second update refers to the same attribute as the first. Different implementations can at this point lead to a discrepancy in the "received message" count. The number of NOTIFY messages received can be counted as the number of units of XML presence information stored (counted once the XML payload is recorded), or as the number of actual NOTIFY messages received, irrespective of what presence information is contained therein (counted when NOTIFY messages arrive). By counting the units of XML stored, it is possible to receive many more NOTIFY messages at the RLS than there are presentities contained in the NOTIFY to the PUA, due to the stored XML payloads being overwritten, and hence not counted as separate updates. This will lead to information at the PUA becoming stale, as numerous NOTIFYs may be received before the data is forwarded to the PUA.

3.1.3.3 Size-based Aggregation

Size-based aggregation is similar in operation to quantity-based aggregation, with the size of the accumulated XML-encoded presence information acting as the trigger in the RLS, generating a NOTIFY containing the accumulated payload to the PUA. Again, the timely delivery of information is not guaranteed, as the generation of NOTIFY messages depends on the size of the accumulated presence information.

As can be seen from Figure 3.8, the operation of size-based aggregation is identical to that of quantity-based aggregation. The only difference is at the point where the RLS checks if the addition of the new NOTIFY message causes the accumulated information to exceed the aggregation parameter. In this case, the parameter compared is the size of the accumulated information, rather than the number of presentities stored.

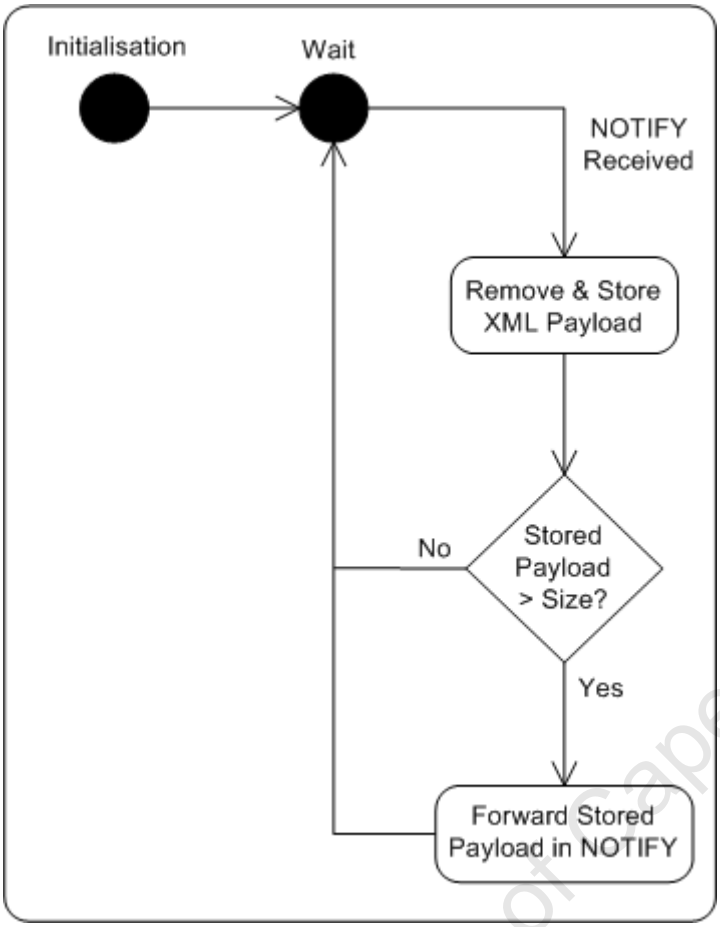

Figure 3.8: State Chart for Size Aggregation

The rate at which NOTIFY messages are sent again depends on the size of the RL, and the relative activity of subscribed presentities in the RL. Also relevant to the rate is the state of the subscription. In the period immediately after subscription to the RL, the messages returned to the PUA by the PSs contain all the presence information for the presentity, and are hence very large. After all the relevant complete presentities have been provided to the PUA, the operational state becomes steady, and the size of the presence payload shrinks to contain only the modifications made to presentities' information. The rate of updates emanating from the RLS at this point is likely to drop considerably.

Size-based aggregation also suffers from the problem described in Section 3.1.3.2, where issues of implementation may affect calculations of the size of the stored presence payload. In the case where a NOTIFY is received for a presentity which already has a stored update, the new update will replace the old one. This can cause extended delays before a NOTIFY is sent to the PUA, particularly if some presentities are particularly active updaters. It is possible that at times, a poor choice of the size threshold may lead to no notifications being sent from the RLS whatsoever, due to the overwriting of stored information or simply insufficient presence information being stored, which then never reaches the threshold value.

3.1.3.4 Size and Time Aggregation

The problems experienced by each of the individual aggregation methods discussed can become significant when parameters, such as the aggregation timer or presence update likelihood affecting the RLS are poorly matched. Mitigating these effects can be difficult, especially if the number of controlled parameters is limited. In an effort to increase the level of control over the RLS operation, time as a trigger for aggregation can be added to the numeric aggregation triggers of size and quantity.

Time and size aggregation methods operating together should provide considerably better performance, in terms of the timeliness and size of the NOTIFY update to the PUA, when the aggregation parameters are poorly suited to operational parameters outside the RLS control (such as the activity of subscribed presentities, size of RL, etc). The inclusion of time as an aggregation parameter ensures the timely delivery of information, while the size aggregation ensures that the size of the presence payload to PUAs does not become larger than a few small presentities combined. The two aggregation parameters do however need to be relatively well matched. If the parameters are badly matched, the RLS operation can essentially revert to using only one of the aggregation methods. The aggregation method the RLS reverts to is likely to offer better timeliness, but not necessarily better overall performance.

A state chart diagram for the combination of size and time aggregation is given in Figure 3.9. The default operational state for the RLS is to wait for incoming NOTIFY messages from a PS servicing a subscribed presentity. When a NOTIFY message is received, the RLS processes it by stripping the SIP header, and storing the XML presence payload for the receiving PUA. The RLS then compares the size of the accumulated XML payloads to the aggregation parameter threshold. If the accumulated payload falls below the threshold, the RLS continues to wait for NOTIFY messages, and for the timer to expire. If the accumulated payload is larger than the threshold, the RLS initiates the forwarding of the payload. The expiration of the timer also initiates the forwarding of the payload, separately from the size aggregation mechanism. The RLS encapsulates the stored payload in a SIP NOTIFY message, and forwards it to the PUA. The store for the PUA is then cleared, and the RLS waits again for further subscription messages.

The combination of aggregation methods offers subtle differences in implementation which may affect the operation of the RLS as a whole. The implementation of the timer(s) can lead to unwanted behaviour which may occur if their functioning is separate from the receiving of NOTIFY messages, and the confirmation of whether the accumulated payload exceeds the threshold for generating NOTIFYs to the PUA. In this instance, the two methods operate independently, which creates shorter intervals between NOTIFY messages, with smaller message payloads being sent. This occurs because the timer is not reset by size-aggregated messages being sent to the PUA, which may expire soon after a NOTIFY is sent. In such an instance, the data is effectively not size aggregated nor time aggregated, as the period after the last size-aggregated message is of random duration.

The alternative to this implementation is for the timer to be reset every time a size-aggregated

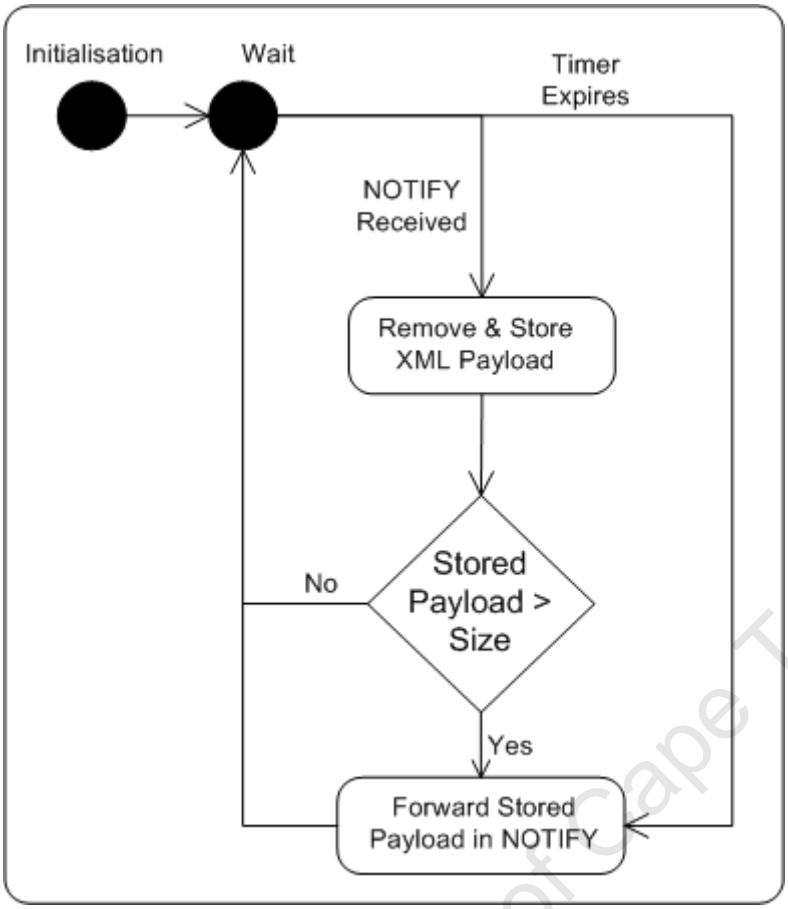

Figure 3.9: State Chart for Size-Time Aggregation

message is sent. This ensures that the period between NOTIFY messages has an upper bound of the timer time-out value, and that the accumulated information is only time-aggregated if the timer expires. This method, while desirable for the aggregation performance, is not a desirable solution for implementation. This is because a timer (for instance, a periodic thread) would most likely be required for every subscription to an RL held by the RLS, and would often be reset. In order to maintain system viability and performance, servers are implemented with thread pools, from which threads are taken when required and returned when no longer needed. The use of a thread pool limits the number of threads that may be executing at any time instant, and ensures that resource allocation in the system is equitable.

3.1.3.5 Quantity and Time Aggregation

The amalgamation of time and quantity aggregation is done for the same reasons as covered in Section 3.1.3.4. The addition of the aspect of time to quantity aggregation is expected to maintain a minimum level of throughput from the RLS, during periods in which the update activity from presentities in the subscribed RL is not sufficient to generate NOTIFY messages to PUAs at a normal rate.

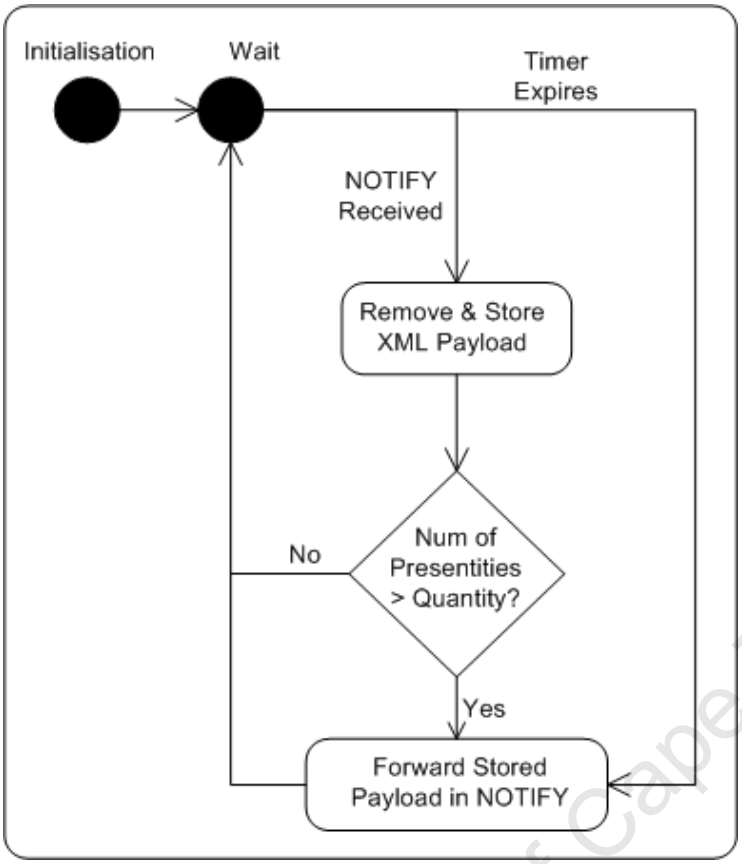

Figure 3.10: State Chart for Quantity-Time Aggregation

The operation of this combined method is given in Figure 3.10 in the form of a statechart. The operation is essentially identical to that described in Section 3.1.3.4. The RLS waits for incoming messages and processes received NOTIFY messages, by removing and storing the XML payload. If the number of presentities received surpasses the preset aggregation threshold, the accumulated information is forwarded to the PUA. The expiration of the timer thread can also force the RLS to move from waiting for a NOTIFY to arrive, to encapsulating the accumulated information and forwarding it to the PUA. Again, the subtly different implementations described previously can affect the overall operational efficacy of the methods in the RLS.

3.2 Analytical Metrics and Test Methodology

The testing of various RLS methods is to be conducted on an RLS implementation, operating in a pseudo-typical IMS execution environment. As only the functioning of the RLS itself is being considered, the standard IMS elements can be dispensed with for the purposes of the test. They are instead replaced with a series of data sources and sinks, which subscribe to presentities and provide presence information through the RLS. These external entities control parameters such as the size of the RL, the state of the subscription, and the activity level of the presentities subscribed to. As the focus of this work is on the aggregation of presence data

by the RLS on a per-subscription basis, aggregation is examined only for a single subscription, rather than a validation of the developed implementation for its ability to effectively manage numerous subscriptions.

Information regarding the performance of the aggregation method is collected by the RLS itself, through time-stamped simulation logs of events and RLS activity. Numerous relevant details, such as the time, sender, message size and payload size are stored. The information obtained is to be analysed after a series of tests is completed, in order to determine the timeliness of information delivery, effectiveness of the aggregation parameters and the general suitability of the method. This information can be obtained using data such as the number of presentities received per outgoing message, the size of the incoming message payload, the size of the entire incoming message, comparisons of the incoming data size to the outgoing message payload, the size of entire outgoing message and message timing information.

Analysis of this information is intended to determine the conditions under which an RLS is able to perform most effectively, as well as to identify conditions which may cause it to become less effective, or even totally ineffective. This will allow investigations into the reasons causing the system to be ineffective, and provide indications on how to reduce ineffectiveness at points in the future.

Chapter 4

Architecture and Implementation of an Evaluation Framework

4.1 Choice of Platform

The choice of platform for the Resource List Server in the evaluation framework was made between using a simulated RLS, or an emulation of the server. The implementation of an experimental platform is likely to yield a more accurate indication of the RLS operation, with design choices made during development influencing the overall server operation. Either of the choices for the platform (an emulation or simulation) would have to be examined while operating in a simulated environment. The simulated environment was necessary because of factors such as the number of presence servers likely to be required in order to yield realistic testbed implementations, with a home network PS and some visited-network PSs as well.

The data sources and sinks, such as the watcher subscribing to the RL and the various PSs serving presence information to the RLS, are simulated in the testbed. They are implemented using a SIP traffic generator, with statistical distributions used to simulate appropriate delays between receiving requests and sending responses, and to create pseudo-random delays between NOTIFY messages from the PS.

4.2 Objectives of the Implementation

The objective of the evaluation platform was to provide an opportunity to perform an experimental investigation into the performance of RLS aggregation methods, rather than making use of an entirely simulated environment to obtain results.

The implementation would have to provide an effective platform with which to evaluate the RLS aggregation methods under investigation. The implementation would be required to allow accurate measurements of presence traffic flows both into and out from the RLS to be made. The

measurements of data reductions between received and forwarded data and the time between successive updates to the PUA, once examined, would need to show whether the aggregation methods examined proved suitable to the task. Of particular concern would be whether the methods were able to perform efficiently under the different load conditions and aggregation parameters examined, and provide an indication of ideal aggregation parameter ranges for different load conditions.

Specific objectives of the implementation, with regard to the testing and validation of the aggregation methods, were considered to ensure a comprehensive examination took place. This included examining the effect of changing aggregation parameters on the performance (in terms of data-reduction and timeous delivery) of the RLS, and how factors such as the size of the presentity received by the RLS and the size of the RL subscribed to by the watcher affected the performance. These were considered to be important aspects in determining the efficacy of the RLS.

The implementation of the RLS carried out was also designed to serve as a basis for the later development of a comprehensive RLS service, using the developed RLS as a foundation. The implementation used in the research carried out, while functional, did make some concessions. The concessions were primarily made to enable faster and easier testing to be carried out, and to allow the focus on an RLS designed for examining traffic in a single-subscription environment.

4.3 Implementation Entities

4.3.1 The Subscriber

Any network subscriber accessing the RLS service becomes a subscriber to the RLS and a watcher of all presentities contained in their RL. In a deployed network environment it is likely that the RLS in an operator's network will handle the RL subscriptions for a portion, or possibly all of the RLS subscribers. Mechanisms to allocate service subscribers between the available RLSs are likely to ensure load balancing and operate the RLS cluster effectively. The use and operation of such mechanisms does not form a part of this investigation.

The simulation environment only caters for a single subscriber entity. This single subscriber (PUA) subscribes to the RL at the RLS, and receives presence updates in the form of SIP NOTIFY messages, as the RLS receives and aggregates the presence information being watched by the PUA. A single subscriber instance is used as it isolates the RLS operation for the RL subscription, simplifying the analysis of the resulting RLS logs after running the verification and performance tests. The use of multiple subscriber instances would require determining to which instance each of the NOTIFY messages received at the RLS is intended, and to which subscriber each update sent out from the RLS is intended.

The watcher subscribes to the RL with a SIP SUBSCRIBE message. For ease of implementation, the XML-encoded RL is included as the payload in the SUBSCRIBE message. The subscription to the RL is implemented in this way because of the initial simplification of the testing performed.

The subscriber entity is implemented using SIPp [57], a SIP traffic generator. The various subscriber entity configuration files are included in Appendix D.

4.3.2 The Resource List Server

The RLS functions as a normal RLS, and is capable of managing subscriptions by multiple watchers to their respective RLs. For the purposes of the investigation however, it manages only a single subscription during testing.

The developed RLS logs all relevant traffic into and out from the server to a log file for later analysis. All log entries are time-stamped and include information such as the aggregation method and relevant parameters being used, or the size of the message and presence data payload for all the NOTIFY messages sent or received. The logged data is analysed to determine statistics such as the frequency of updates, size of updates, number of presentities per update, or the effectiveness of the combination of time-based aggregation with either of the other two aggregation methods. The single subscription is used for testing, as it considerably simplifies analysis of the generated log file.

The RLS is implemented with five different aggregation methods available for use, and can configure the aggregation methods using provided parameters. The server can be reconfigured with a new aggregation method and parameters without having to be restarted. Reconfiguring the server flushes the current stored information and subscriptions, and kills any periodic threads that may be executing, leaving it ready to start a new series of tests.

The server is reset when a SIP MESSAGE message is received. The mechanism for delivering the MESSAGE to the RLS is described in Section 4.3.2.1.

The RLS is developed using Java [58] and the SIP Servlet [59] libraries, which port the Servlet paradigm for web application development to SIP-based applications. This allows the rapid development of SIP-based applications by experienced Java programmers, while removing the need to understand the intricacies of SIP signalling. The developed server is run in Mobicents' Tomcat [60], a web server container modified by Mobicents to incorporate SIP Servlets.

4.3.2.1 The Server-Reset Entity

In order for the RLS to be reconfigured while still running, it must receive a SIP MESSAGE message with an XML-encoded message payload. The new parameters to be used by the server are encoded in XML, as XML is easy to parse in the server, and because it allows a user to easily read what the new parameters to be loaded into the RLS are. An example of the message payload is given in Figure 4.1.

A SIPp entity is used to send this SIP MESSAGE to the RLS in order to reset it. The configuration files for the server-reset entity is included in Appendix D. The server is reset before the start of each new test performed.

```
<?xml version="1.0" encoding="UTF-8"?>
  <resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <list>
      <entry aggregationMethodIndicator="quantity" />
      <entry maxPresenceSize="12000" />
      <entry maxPresentitiesReceived="5" />
      <entry threadDelay="10000" />
      <entry threadPeriod="5000" />
    </list>
  </resource-lists>
```

Figure 4.1: XML-encoded Payload for RLS Reset Message

4.3.3 The Presence Servers

During an RLS subscription, the RLS is likely to interact with PSs located in both its own (home) network, as well as other foreign networks through SUBSCRIBE and NOTIFY messages. Because the RLS only interacts with the interface between the PS and the Watcher, it never receives any PUBLISH messages from a Presentity to its PS.

The PSs used in the implementation are instances of SIPp scenarios. The PSs receive subscriptions to watch Presentities from the RLS, which acts on behalf of the Watcher from this point on. The PS then sends the subscribed Presentity to the RLS, encapsulated in a NOTIFY message. Subsequent changes or updates made to the Presentity are also sent to the RLS. These updates continue for the duration of the Watchers' subscription to the RLS service.

The PSs are able to start operation in the Initialisation State, where the comprehensive Presentity is initially sent to the Watcher by the PS, followed over time by updates containing only the portion of the Presentity that has been updated. The PS can also be configured to start operation in a Steady-State mode, in which only the aspects of the Presentity that are updated over time are forwarded to the Watcher. The simulations of the PSs used in the implementation are essentially dumb, as they do not vary the Presentity data served to the Watcher intelligently, but instead forward a standardised response. While this deviates from what may be considered to be the normal operation of a PS, it does allow the tests to achieve a uniform average Presentity size across the numerous tests performed.

Four separate PS instances are created for the implementation. This choice was made for a number of reasons. Firstly, an RLS interacting with a number of PSs follows more closely the expected operation of a deployed RLS service, with the RLS interacting with PSs located in competitor operators' networks, as well as its' own. Secondly, by spreading the subscriptions to Presentities across a number of PSs, potential load problems and other problems encountered during the testing are easily identified and can be corrected.

The envisaged implementation of presence in the IMS, with its highly integrated architecture incorporating active presence-updating network entities such as the PNA, has not yet had any highly publicised implementations carried out. This, coupled with the relative randomness of the activities likely to cause network-generated updates and user-generated presence updates, makes a real-world model for the likely behaviour of presence updating intervals difficult to

obtain. The modelling of such activity is further complicated by the vast disparities in presence-updating behaviour of different subscriber groups such as teenage girls, in comparison to their grandparents. Because such a comprehensive model was not available, a vastly simplified model with statistically random delays were used to control the rate at which the PS generated a notification to the RLS.

4.4 Limitations of the Implementation

The implementation carried out is constrained by some restrictions to its operation. While the restrictions limit some of the functionality, and substitute certain entities in the presence framework with modelled entities rather than actual implementations, the results are considered to be a true reflection of the efficacy and effectiveness of the aggregation methods examined.

The primary limitations imposed on the framework are a result of decisions made during the design and development of the RLS. The main limitation within this context is the design choice made to focus primarily on developing the RLS to manage a single subscription to an RL, rather than multiple unique subscriptions to the individual subscribers' RLs. The result of this design decision is that for multiple unique subscriptions existing at the RLS, there is a failure to maintain differentiated levels of privacy for the Presentity. This is because the content of a NOTIFY received by the RLS is saved into the accumulating presence payload of all the relevant Watchers, rather than a specific Watcher to whom the NOTIFY may have been addressed.

A second limitation arising from the design decisions made, is the need for slight delays to be introduced between the sending of a SIP 200 OK message and the subsequent NOTIFY message by the PS. The delay is required to allow the web application container to initialise and store all relevant variables and data structures required by the RLS implementation for a new SIP dialogue, before the next message in the dialogue is received. The delay can possibly be equated to the required database access and transmission jitter in a real-world implementation with physically dispersed PS and RLS entities.

As outlined in Section 4.3.3, there are currently no models for presence-updating behaviour based on data from real-world, highly-integrated IMS presence implementations available. The lack of the availability of such models has required the use of simplified models instead. These models do not take into account possible differences between the update behaviour of different social groupings based on age, gender or occupation. The models used in the implementation, while not as detailed, do offer standard, averaged parameters for governing the update behaviour.

Subscription duration and subscription expiration at the RLS are not yet fully implemented. The mechanisms to monitor the duration and expiration times of subscriptions by the RLS exist, but because of ongoing development of the SIP-stack enabled Apache Tomcat web application container used in the implementation, the container functionality to monitor and control subscriptions only became available after RLS development on this aspect had been completed. In addition, subscription-state monitoring in the PS and Subscriber entities was also not

implemented. However, because the validation and investigative testing carried out using the implementation utilised finite time periods under one hour (the presence event package subscription default duration [28]) this did not present a problem during testing as the duration was simply set to be larger than the test duration.

There were however some positive outcomes obtained in addition to an implementation that allowed the full suite of tests to be carried out. The first of these is the operating system portability of the developed RLS. The RLS is developed entirely in Java [58], allowing it to be run on any platform for which a Java Virtual Machine (JVM) is available. The RLS also leverages the advantages provided by the use of the Apache Tomcat web application container, which has been adapted by Mobicents [61] to provide support for the SIP Servlet Java language classes [59] across multiple platforms [60]. SIP Servlets provide a suite of SIP-signalling and session management capabilities to servers hosting SIP and web applications, enabling the easy extension of server capabilities, and are becoming widely used by developers in SIP and IMS applications. Apache Tomcat also manages aspects such as multi-threading and the ability to utilise multiple network interfaces, with minimal effort on the application developers' part. These tools and capabilities enable the development of fully featured, high-performance SIP applications.

4.5 Evaluation Tests Performed

The design of the presence framework in the IMS has led to the creation of a number of configuration parameters that can be leveraged to achieve an optimal implementation performance. The parameters of primary concern for this research are the aggregation methods available for use. Each aggregation method offers particular performance benefits as well as drawbacks. Ideal aggregation method performance should balance the aggregation of presence data in a subscription with guarantees of the timely delivery of the aggregated presence information to the Watcher. The aggregation methods examined in the evaluation are described below. Additional parameters likely to affect the performance of the RLS are described following the aggregation methods. These parameters include the number of Presentities in the RL, the presence update activity of the subscribed Presentities and the subscription state.

4.5.1 Aggregation Methods

The three aggregation methods examined use elapsed time, the number of presentities received, and the size of the presence payload as triggers to determine when a NOTIFY should be sent to the Watcher.

4.5.1.1 Time Elapsed-based Tests

This method uses the time elapsed between successive presence updates - in the form of NOTIFY messages - as a trigger. The period between updates is uploaded to the RLS using the entity

in Section 4.3.2.1. The RLS creates a periodic thread using the uploaded period parameter, which is destroyed when the RLS parameters are reset. The periodic thread then determines whether an update is required each time it runs, and if so, it prepares and sends the NOTIFY message to the subscribed Watcher. The examination performed used six different time periods: 30 seconds, 1 minute, 2 minutes, 5 minutes, 7 minutes and 30 seconds, and 10 minutes.

For each time parameter used the set of tests was repeated, with other parameters under consideration such as the presence update behaviour and the number of subscribed presentities being changed.

Time-elapsed aggregation may cause stale information to be provided to the Watcher if the time intervals used are too large. It is also susceptible to bursts of information, which are not dealt with satisfactorily. The NOTIFY message from the RLS may become exceedingly large or very small, neither of which are desirable cases.

4.5.1.2 Quantity Received-based Tests

A NOTIFY message to the watcher is triggered when the number of presentities received by the RLS for a particular subscription reaches a predefined threshold value. The threshold parameter is uploaded to the RLS through the entity in Section 4.3.2.1. In this method the number of presentities carried in the NOTIFY remains constant, while the size of the message will vary between successive transmissions.

This effectiveness of this method is partly dependent on the arrival rate of NOTIFY messages from the respective PSs. Because of this, a low NOTIFY arrival rate or small RL may result in long delays between successive updates from the RLS. This is of particular concern when the timely delivery of data is important. A presentity can be delayed by up to a few minutes, but longer delays are likely to render the service ineffective. Alternatively, a very high message arrival rate is likely to cause many NOTIFY messages from the RLS to the UE. The effectiveness is also dependent on a well chosen threshold parameter. Large threshold parameter values are likely to cause the delivery of stale presentities, while very small values can reduce the effectiveness of the RLS entirely. The parameter values used in test are 1, 3, 5, 7, 10, 15, 20, 25 and 30 presentities received.

4.5.1.3 Presence Payload Size-based Tests

The third primary method examined uses the data size of the presentity payload to determine whether to send a NOTIFY to the UE. The payload size is calculated each time a new or updated presentity is received by the subscription. If the payload size equals or exceeds the uploaded aggregation parameter, a NOTIFY to the watcher is constructed and the presence payload attached.

The period of the NOTIFY messages from the RLS is variable and depends on the rate at which NOTIFY messages are received from the respective PSs. This may cause the problem of

the delivery of stale information, if few NOTIFY messages are received and they contain small amounts of information. The number of presentities contained in the RLS update is dependent on the size of the presentities received, and the size of the aggregation parameter. A poorly chosen parameter value can nullify the effect of the RLS by being too small, turning the RLS into a simple proxy. Alternatively, too large a parameter value will starve the UE of up-to-date presence information. The parameter values used in the test are 500, 750, 1000, 1500, 2000, 2500, 3000, 5000, 7500 and 10 000 bytes.

4.5.2 Resource List Size

The Resource List is the parameter in the implementation over which the watcher has the most control. It is likely to remain relatively stable, with only small variations in the size of the list and a low turnover of watched presentities. This is most likely due to a "core" group of watched presentities, comprising of friends', family and work colleagues' presentities, who are watched all the time. Additional presentities added to the list for short periods are likely to be the cause of most of the turnover. The size of the RL can vary from plausibly containing only a few presentities, to RLs based on a watchers' entire on-line phone book. The size of such an RL could easily be in the region of 100 to 150 different presentities. The analysis of IM data logs from a large corporate network found an average presentity list size of 22 [62], while a study examining a users' age and the management of their social networks found the subject group of 50 to 60 year-olds had 21 presentities per list, while 16 to 18 year-olds had on average 59 [63].

In order to account for the large range of possible RL sizes, the tests were performed with six different RL sizes: 5, 10, 15, 20, 30 and 50 presentities per RL.

4.5.3 Presentity Update Activity

One aspect of the evaluation framework that is particularly difficult to implement in a real-world manner is the updating behaviour of the watched presentities. As highlighted in Sections 4.4 and 4.3.3, comprehensive, accurate models for the updating behaviour of a group of presentities are not readily available. Because of this, the activity of presentities are modelled with "update periods". The "update periods" are implemented as SIPp delays, which implement a random delay using a uniform distribution, between a specified minimum and maximum value [64]. The properties of the uniform distribution ensure the sending of updates is random [65], which is an appropriate approximation for the purposes.

Four different update time periods are used, with maximum values of 2, 5, 7, and 10 minutes. Minimum values for the time periods (producing a guaranteed delay before an update may occur) are set to 500 milliseconds. The minimum delay is included to guarantee the RLS the opportunity to fully process the previous update sent. All tests in the evaluation were performed using each of the update time periods.

4.5.4 Subscription State

The lifecycle of a subscription to an RLS exists in one of two main lifecycle states. The first state is of short duration, and begins when subscriptions to listed presenties are made by the RLS. It continues to the point when the RLS subscriptions to the watched presentities begin to transmit updates made to the presentity, rather than transmit all the presence information held by the PS (the 'full' presentity) for the presentity. Because the full presentity is transmitted for each subscribed presentity, there is a large amount of data arriving at the RLS, which must be forwarded to the watcher. Furthermore, because the full presentity is held by the PS, there is no significant delay in transmitting it to the RLS, leading to a large amount of information arriving in a short period of time. This state is termed the "Initialisation State".

After the initialisation state ends, the subscription enters the stable state, which lasts until the watchers' subscription to the RL ends. During this time the updates received by the RLS are only of updates made to the presentities, and so contain only relatively small amounts of data. This state is termed "Steady state", as the rate at which updates are received is expected to remain relatively stable, with possible minor fluctuations.

The examination of the RLS aggregation methods in these two states is important. The two states place very different load characteristics on the RLS, and as such may require the aggregation methods to be specifically tailored to handle the load, in order to maintain effectiveness. Both subscription states are evaluated in the tests carried out. The duration of the tests for the Initialisation State are kept relatively short, at three minutes. The Steady State tests are carried out over longer periods, mainly 15 minutes. This is done because of the considerably slower arrival rate of updates in this state, compared to the Initialisation State. Some time-based Steady State tests were carried out over longer periods, in order to accommodate more RLS updates to the watcher during the testing period.

Chapter 5

System Validation and Performance Evaluation

This chapter presents the results of the developed system validation, and an evaluation of the of the Resource List Server performance under sets of differing parameters. The parameters of both the aggregation method and the stream of presence data supplied to the RLS are examined. Experimental variations of the parameters are performed, in order to gain an understanding of how such changes affect the output of the system. Sets of different parameters are tested under similar conditions to provide a collection of data which can be analysed to provide the desired insights.

The chapter is organised into three sections, examining the effect the RLS has on traffic across the access network through the migration of repetitive presence signalling from the PUA to the IMS Core Network in Section 5.1. Section 5.2 examines the operation of various aggregation methods. The section is split into two parts, providing an analysis of aggregation under Steady State operation, and a discussion of the testing performed for Initialisation State operation. In Section 5.3 comparisons are made between the three individual aggregation methods operating under a steady subscription state, using data obtained from the tests examined.

5.1 Signalling Transfer from Access Network to IMS Core Network

The RLS is able to reduce presence-related traffic across an access network to the PUA through two mechanisms: through aggregation of the received data, examined in this chapter; and by the transfer of presence signalling from the access network to the bandwidth-rich IMS core network, as explained in Section 3.1.1.

The potential for traffic reduction through the transfer of the signalling is clear, but has not yet been quantified. Before the primary objective of this work is examined, calculations are

provided in an attempt to quantify the effectiveness of the traffic reduction, as introduced in Section 3.1.1 with the aid of Figures 3.2 and 3.1.

A single subscription to a presentity or resource such as a resource list involves 4 initial SIP messages between the PUA and the IMS, two of which are SIP OK messages, and at least 2 subsequent SIP messages, when presence information is returned. Subsequent updates to the presence information are not considered in this section. In the case where an RLS is not available or is not being used, the same sequence of 6 SIP messages will have to be exchanged with presence servers for each presentity being subscribed to, giving a total of $6n$ messages for n presentities. However, if an RLS is available and being used by the PUA, it subscribes to its RL held by the RLS as per Figure 3.2. As the RLS subscribes to the presentities in the RL on the PUAs behalf, there are only 6 messages exchanged with the RLS in total, including the first message containing aggregated presentities received. Subsequent updates to presence information are stored and aggregated for the PUA, reducing the number of messages received further.

5.2 Examination of Individual Aggregation Methods

5.2.1 Steady State Operation

A subscription enters a steady state of operation when the initial high volume of data from the relevant PSs to the RLS diminishes to the point where the data stream comprises only the random presence updates of the subscribed presentities. The data stream in this state is relatively stable, and allows aggregation to be approached using a number of different methods.

The data is presented in the form of graphs, which are generated through analysis of the data files generated during testing. As each event during a test is referenced by the time of its occurence, the time elapsed time is plotted along the x-axis. The metric of interest is then plotted against the y-axis (with a second metric on the secondary y-axis, in some analyses). All plotted data is obtained through calculations using the data size and time of occurence recorded in the log for every SIP message.

Appendix A provides further information regarding the tests performed, and specifies the ranges of parameter values used in each series of tests. Only relevant subsets of this data are presented here, for clarity.

5.2.1.1 Size Aggregation

Three different aspects of the testing carried out for size aggregation are examined for each aggregation method. These effects are the changing of RL sizes, different presence update periods, and changing aggregation parameter values. A detailed discussion of size aggregation is found in Section 3.1.3.3.

Aggregation with Different Resource List Sizes The operation with a single size aggregation parameter is shown performing across a range of RL sizes in Figure 5.1. The time in minutes between successive transmitted messages is measured off the y-axis, and the duration of the test measured by the x-axis. It serves to illustrate how the delay in information delivery is affected by the size of the RL. As the aggregation method is independent of time, the period between messages serves as an indication of the effectiveness of the method in general, and the aggregation parameter, when considered for the specific conditions.

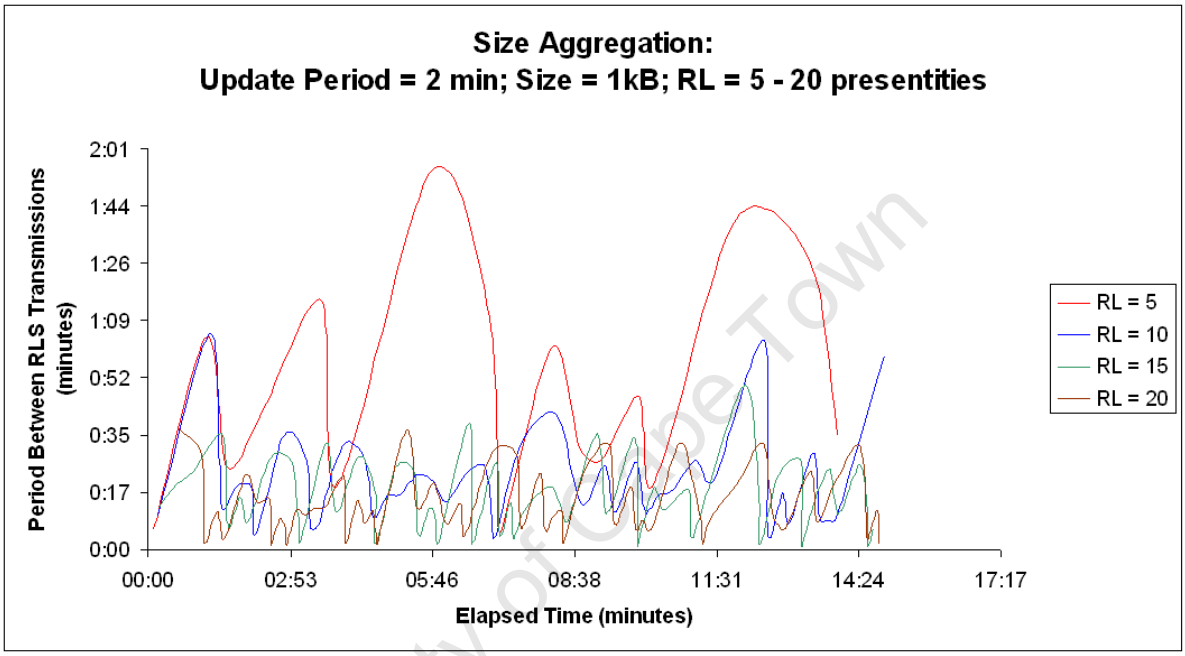

Figure 5.1: Size Aggregation: Presence Update Period = 2 min; Size Parameter = 1kB; RL Size = 5 - 20 presentities

Figure 5.1 shows that the time between successive NOTIFY messages from the RLS becomes shorter as the RL size increases. The jitter between messages also diminishes considerably with larger RL sizes, as the difference between the lines for an RL of five (red line) and an RL of twenty (purple line) show, as the larger number of presentities results in more regular, although still random, updates per unit time. With a size of 1000 bytes, the presence information of four unique presentities is required to pass the aggregation threshold, which accounts for the large jitter of the RL 5 test, as four fifths of the RL must have updated to cause a NOTIFY to be sent, compared to only one fifth of a RL containing twenty presentities.

Similar behaviour is displayed in Figure 5.2 as well, presenting a larger aggregation parameter and larger RL sizes. The axes are the same as for Figure 5.1.

Trends observed from the test results presented indicate the effectiveness of aggregation increases if the size parameter used is relatively small, in comparison to the anticipated overall resource list size.There is considerably less jitter, and shorter delays in the delivery of the stored presence information, than when the parameter value equates to a significant portion of the size of the effective data stream, as demonstrated by the transmitted data curves for RL 5 and RL 10 in

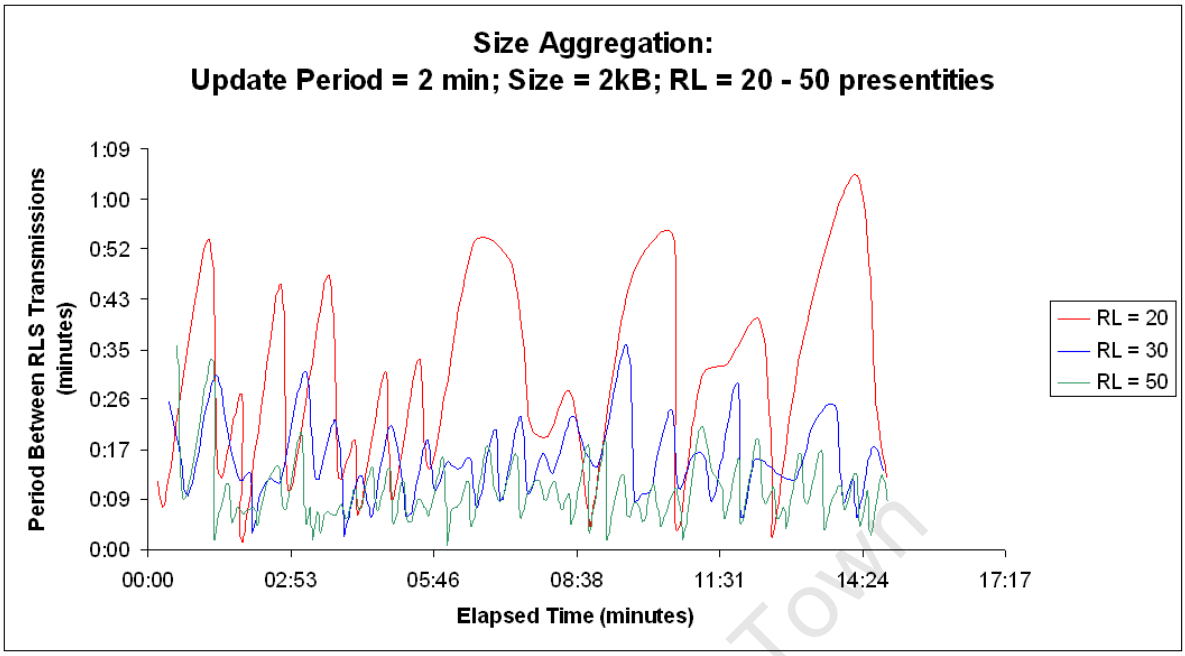

Figure 5.2: Size Aggregation: Presence Update Period = 2 min; Size Parameter = 2kB; RL Size = 20 - 50 presentities

Figure 5.1. The aggregation parameters being used by subscriptions should preferably be linked to the size of the RL, to limit the magnitude of the jitter between message. Care should also be taken to ensure the parameter in use is not too small, leading to a steady stream of messages that could be further aggregated, with little additional impact on performance.

Aggregation with Different Presence Update Periods The relative update activity of the subscribed presentities has a significant impact on the aggregation parameters' effectiveness. Figure 5.3 demonstrates the effect the update activity of subscribed presentities in an RL has for a fixed RL size and aggregation parameter.

The figure shows how size aggregation, with an aggregation parameter of 2kB and an RL of 10 presentities, performs against presentity update periods of 2, 5, 7 and 10 minutes. Because a different aggregation parameter is used for each stream examined, the aggregated streams are all identical in size, indicated by the dashed lines lying between 2000 and 4000 bytes. The data received by the RLS, however, shows a degree of variability in the volume received, which lies between upper and lower bounds of 6000 and 10 000 bytes, for all data streams.

The right-hand axis, in minutes and seconds, presents an indication of the time between aggregated data transmissions from the RLS using the straight-edged lines with point markers. The data stream providing the most regular updates unsurprisingly has the shortest time between messages, with some jitter. Both the time between NOTIFY messages from the RLS, and the jitter between messages in the aggregated data stream increase as the update period increases. With an update time of ten minutes the RLS operation is least effective, as over the fifteen

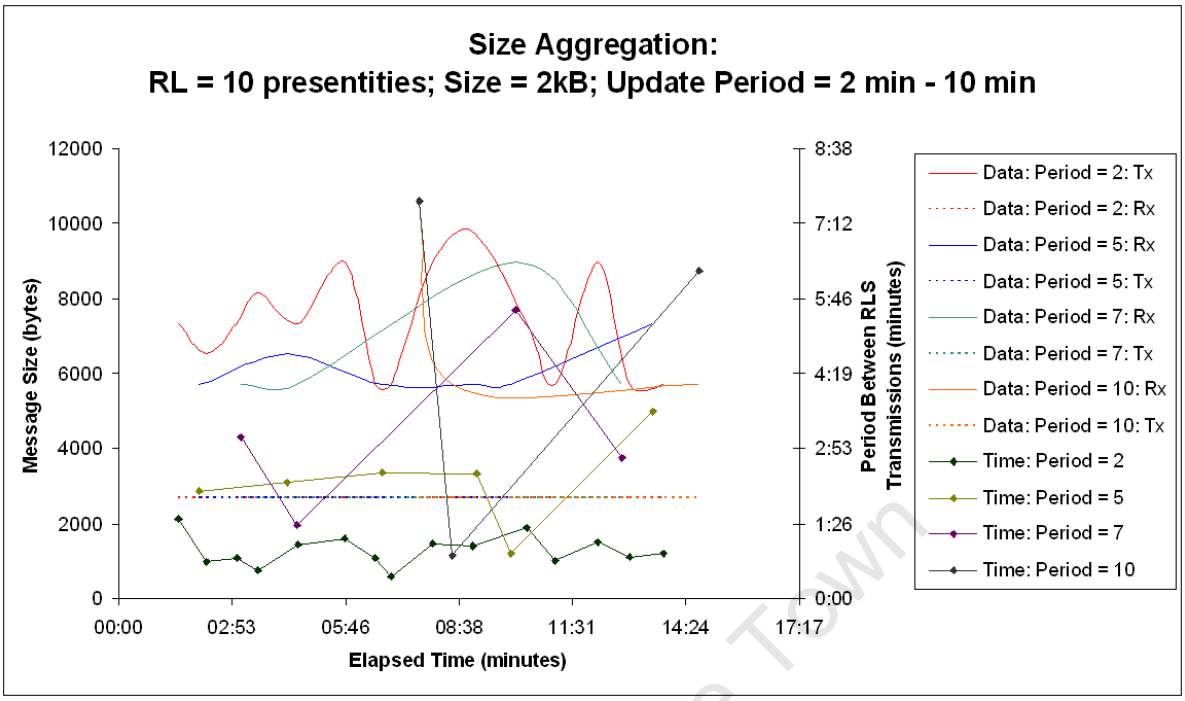

Figure 5.3: Size Aggregation: Size Parameter = 2kB; RL Size = 10 presentities; Presence Update Period = 2 - 10 min

minute test duration, only three NOTIFY messages are sent to the PUA, and the jitter and inter-message time approaches 7 minutes.

Performance of Different Aggregation Parameters Examined here is the performance of different size aggregation method parameters, operating with a simulated update period of 2 minutes. The seven smaller parameter values (0.5, 0.75, 1, 1.5, 2, 2.5 and 3kB) are presented in Figure 5.4 while the four largest (3, 5, 7.5 and 10kB) are presented in Figure 5.5. The parameter value of 3kB is used in both figures, as a reference between the two. Different resource list sizes (10 and 50) are used for clarity, and because at higher parameter values, smaller resource lists take significantly longer to reach the aggregation threshold, or simply do not generate sufficient data to trigger aggregation.

As can be seen, and as is expected, the size of the information aggregated by the RLS and forwarded to the PUA (the dotted "Tx" lines) is uniform and stable. The amount of information received, however (the solid "Rx" lines) is highly variable, and increasingly so as the parameter value increases. This is attributed to the variability of the updating by presentities - some of the presentities may update more than once before the aggregation threshold is reached, increasing the overall data received at the RLS, while the transmitted data size remains static. The out-of-step increase in the size of received data is also partly attributable to the SIP message header accompanying each update to the RLS, which is larger in than the XML-encoded presence information.

Very low values for size (0.5 and 0.75kB, plotted in red and blue) are shown to be effective at

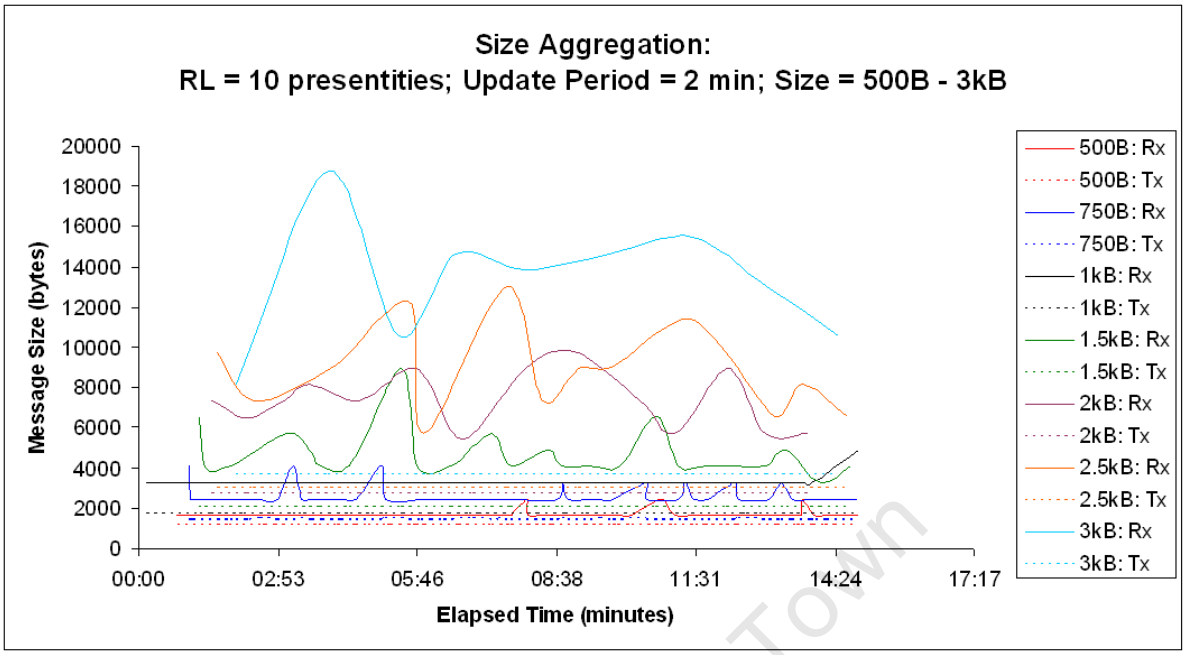

Figure 5.4: Size Aggregation: RL Size = 10 presentities; Presence Update Period = 2 min; Size Parameter = 0.5 - 3kB

reducing the volume of transmitted data, but not particularly efficient, as there is little reduction in the number of messages transmitted. The reduction in size between the data received at the RLS in comparison to the information transmitted to the PUA is not significant, particularly when compared to larger size aggregation values (2.5 and 3kB, plotted in orange and turquoise), in Figure 5.4. Figure 5.5 shows that the received information can become increasingly large as the aggregation parameter values increase, with the amount of received information increasing faster than the size of the aggregated information.

The number of presentities received per update sent to the PUA over time logically increases as the aggregation parameter gets larger. With a parameter value of 10kB, the number of presentities per update is approximately 40, while with a value of 1kB the average is 4, with considerably less variability around that average. The variability is due to some presentities updating their presence information multiple times before a NOTIFY is sent to the PUA, increasing the amount of received data per update.

5.2.1.2 Quantity Aggregation

Aggregation with Different Resource List Sizes The operation of quantity aggregation, when considering only a change in RL size, is naturally similar to size aggregation. The number of presentities received by the RLS per NOTIFY sent may vary, but the number of presentities included per NOTIFY is stable. The random nature of the updating by presentities results in large variation in the time between successive updates to the PUA. This effect can be isolated if parameters such as the aggregation parameter and presence update period are kept static for

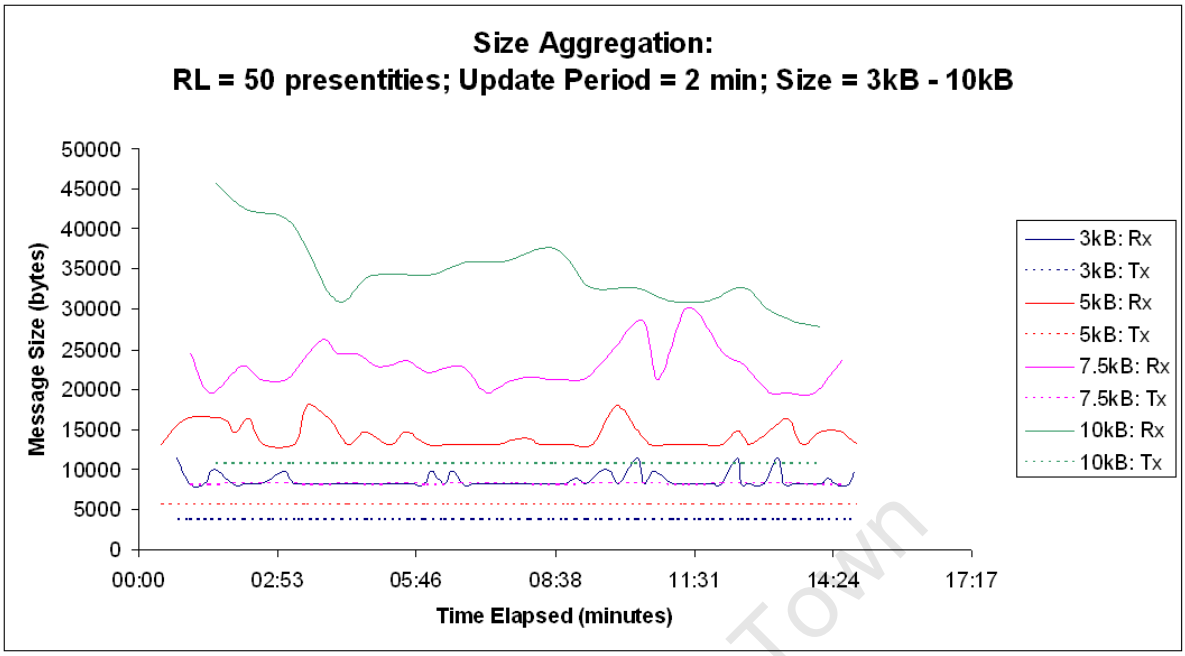

Figure 5.5: Size Aggregation: RL Size = 50 presentities; Presence Update Period = 2 min; Size Parameter = 3 - 10kB

a series of tests.

Figure 5.6 demonstrates the effect of RL size on the timeliness of the data transmitted by the RLS to the PUA. The aggregation method operation can be seen to be similar to that of size aggregation, in Figures 5.1 and 5.2. For relatively small RLs, such as that containing only 5 presentities, the jitter between messages is significant, with a range of almost 1 minute. As the RL size increases, the upper and lower bounds for the inter-message jitter decreases significantly. The increase in RL size does however also lead to more NOTIFY messages being sent, with the time period between these messages also decreasing. The average time between updates to the PUA for a RL of 50 presentities appears to be in the region of 9 seconds; considerably less than the average time between transmissions of 1 minute for an RL of only 5 presentities.

Aggregation with Different Presence Update Periods The RLS implementation used during testing calculates the number of presentities received by counting NOTIFY messages as they are received, rather than the number of stored presentities. Keeping a "running count" such as this eliminates the problem experienced in size aggregation, where the total data received may be significantly more than the data stored. This is highlighted in Figure 5.7, where the volume of received information for each of the incoming data streams is the same, between 5 and 6kB.

Because the volume of received data is uniform, the aggregated data streams are also relatively uniform, although they present a degree of variability over time. This variability is due to retransmissions by presentities, such that the presence information stored for the presentity is overwritten. However, each of the NOTIFY messages received is counted towards the total

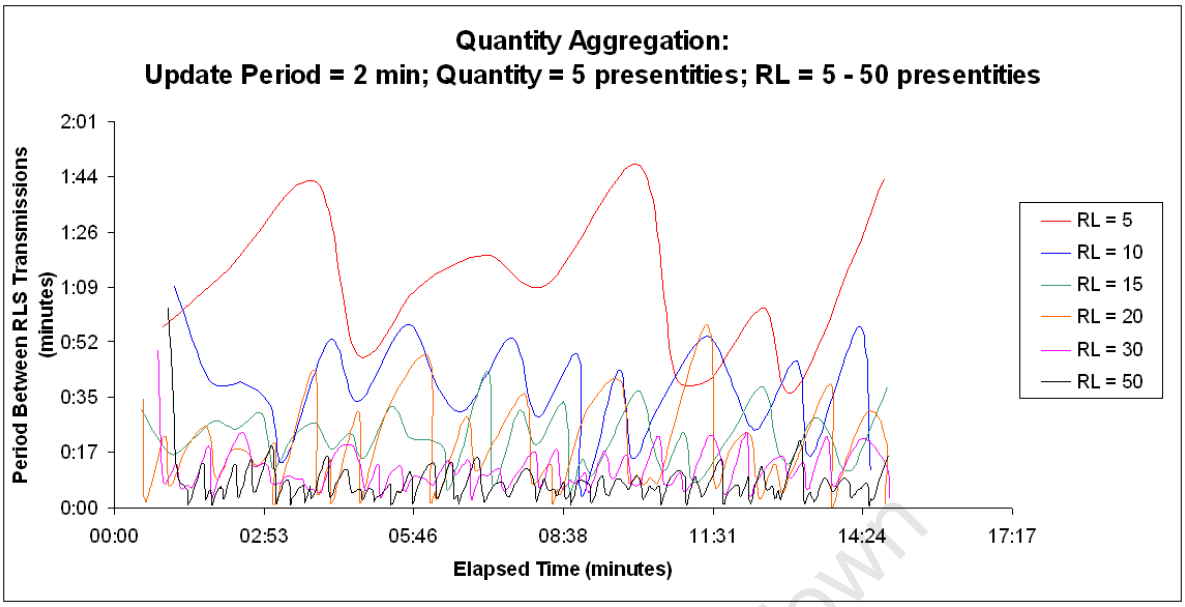

Figure 5.6: Quantity Aggregation: Quantity Parameter = 5 presentities; Presence Update Period = 2 min; RL Size = 5 - 50 presentities

number received, which is used for triggering aggregation. The low levels of activity for presentities with presence update periods of 7 and 10 unfortunately results in only a small data set being recorded.

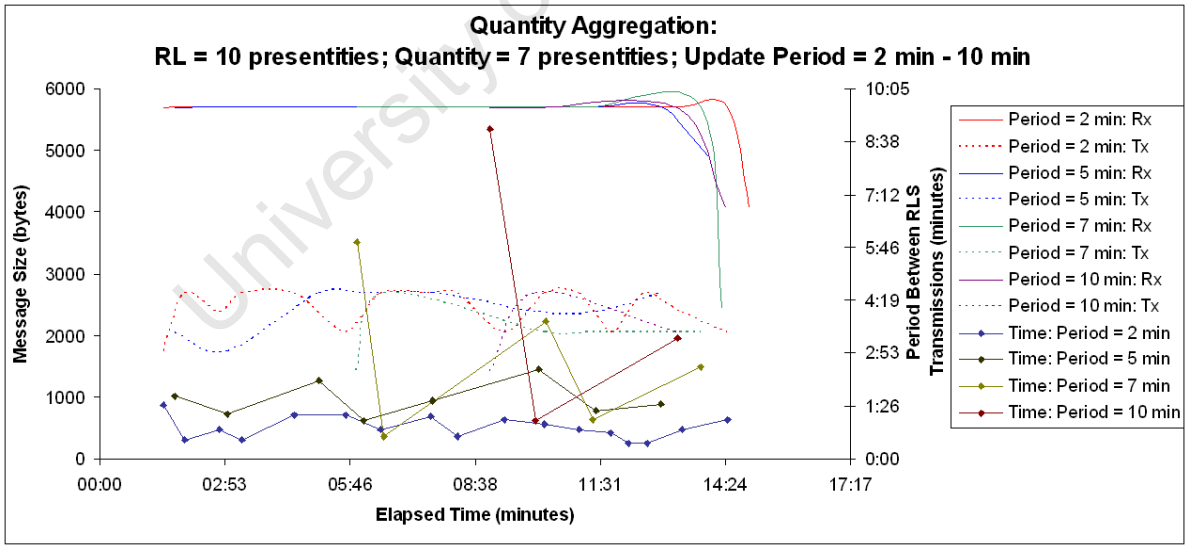

Figure 5.7: Quantity Aggregation: Quantity Parameter = 7 presentities; RL Size = 10 presentities; Presence Update Period = 2 - 10 min

The right-hand axis in Figure 5.7 indicates the time between consecutive NOTIFY messages from the RLS. It is clear that both the jitter and delay are minimised for a presence update period of 2 minutes, while the delay and jitter remain acceptable for an update period of 5

minutes. However for both 7 and 10 minutes, the first aggregated NOTIFY messages to the PUA occur at almost 6 and 9 minutes respectively, and then drop to inter-message times of 1 to 2 minutes. Delays and jitter of this magnitude are too large for the aggregation to be useful, causing erratic data streams and the delivery of stale presence information.

Performance of Different Aggregation Parameters This examines the performance of different aggregation parameters in the quantity aggregation method, operating with a presence update period of 2 minutes. Two different RL sizes of 10 and 50 presentities are shown in Figures 5.8 and 5.9, in part to allow a clear representation of the smaller aggregation values,and avoid being overshadowed when plotted together with data from the larger RL examined. The five smaller aggregation parameters (1, 3, 5, 7 and 10 presentities) are included in Figure 5.8, while the larger four (15, 20, 25 and 30) are in Figure 5.9. The aggregation parameter of 10 is included in Figure 5.9, so as to act as a reference between the two different RL sizes.

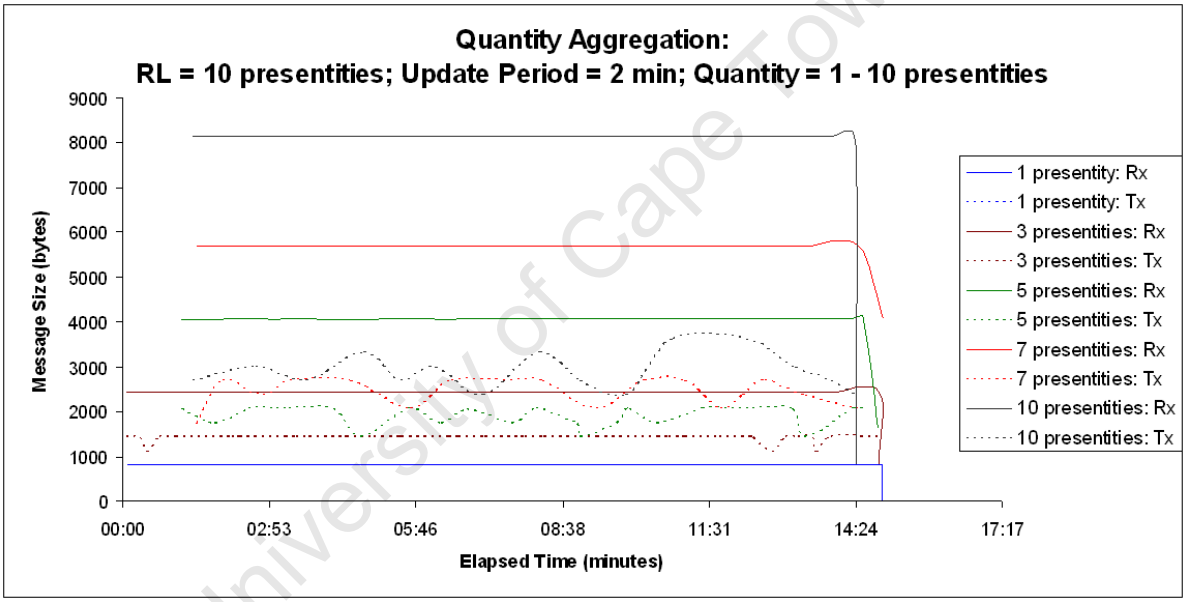

Figure 5.8: Quantity Aggregation: RL Size = 10 presentities; Presence Update Period = 2 min; Quantity Parameter = 1 - 10 presentities

The average presence payload used in the simulation shows in the relatively flat solid lines indicating received data. The transmitted data however, is considerably less uniform, with more variability in message size over time, particularly for the larger parameter values. This can be explained by a larger amount of repeated data being received from presentities, before the threshold value is reached. Figure 5.8 clearly demonstrates the ineffectiveness of a single presentity as an aggregation parameter value, as the outgoing, aggregated traffic at the RLS is matched by the data received, with no effective reduction. A parameter value of 3 presentities does provide some data reduction, with few messages being repeated, as the data transmitted to the PUA for this aggregation parameter is very uniform with regards to data size.

For different parameter values, the information transmitted by the RLS falls roughly between

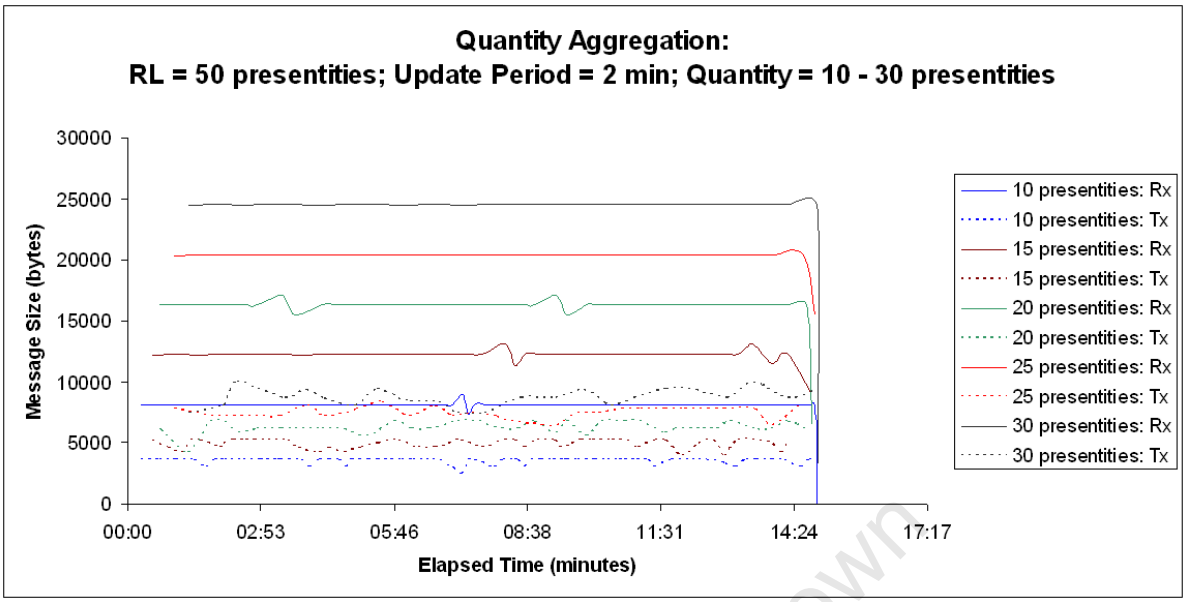

Figure 5.9: Quantity Aggregation: RL Size = 50 presentities; Presence Update Period = 2 min; Quantity Parameter = 10 - 30 presentities

1kB and 3kB, while the information received increases linearly, relative to the parameter value. This occurs because in this instance the RL has 10 subscribed presentities, and as the aggregation parameter increases, so do the number of repeated transmissions before the threshold is reached, hence increasing the volume of received information.

5.2.1.3 Time Aggregation

Aggregation with Different Resource List Sizes When using time as the aggregation parameter, the aggregated updates sent by the RLS are uniform with regards to time. However, the number of presentities received, and the number of presentities transmitted to the PUA, vary.

The number of presentities contained in a NOTIFY forwarded to the PUA is determined partly by the size of the RL, and by the level of update activity of the subscribed presentities. In this instance (Figure 5.10) however the update activity of the system is kept static for each figure, in order to gain insight into the effect of the RL size. The presence update period is set at 2 minutes for Figure 5.10, and 5 minutes for Figure . The aggregation parameter used is 2 minutes, and RL sizes from 5 to 50 presentities are examined in both figures.

The largest RL, with 50 presentities, yields the highest number of presentities per update, as expected. In Figure 5.10 the number of presentity updates received by the RLS per NOTIFY message sent to the PUA is approximately double the size of the RL. This occurs because most presentities have updated twice before the aggregation timer expires, which triggers the forwarding of the accumulated information.

The operation of time aggregation for smaller RLs such as those containing 20 and 30 presentities

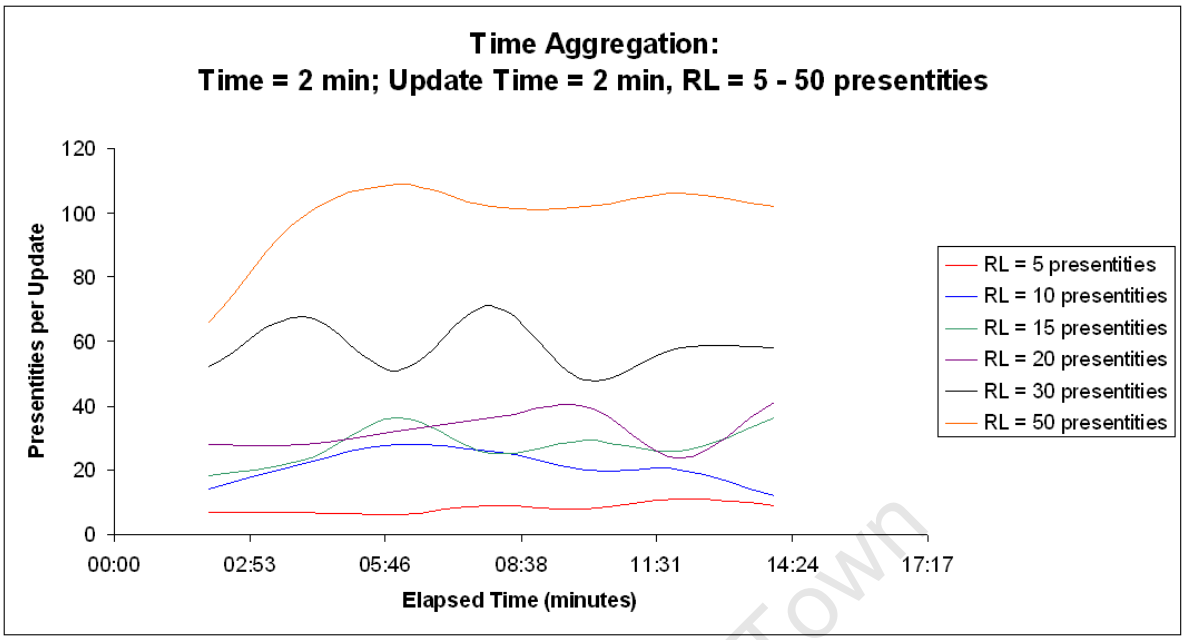

Figure 5.10: Time Aggregation: Time Parameter = 2 min; Presence Update Period = 2 min; RL Size = 5 - 50 presentities

shows some variation in the load curve over time, with upper and lower bounds of approximately 1.5 and 2 times the RL size. The variability in the load curve appears to diminish as the RL gets smaller, to the point that for an RL of 5, the number of presentities received by the RLS doesn't exceed an estimated upper bound of 8 to 10 presentities per update sent.

In Figure 5.11, the change in the update period causes a notable change in the relationship between the number of presentities received per update sent, and the RL size. Although in Figure 5.10 the number of presentities received per update approaches twice that of the RL size for large RLs, in Figure 5.11 the presentities per update is generally less than the RL size, due to the number of presentities likely to have updated being significantly lower.

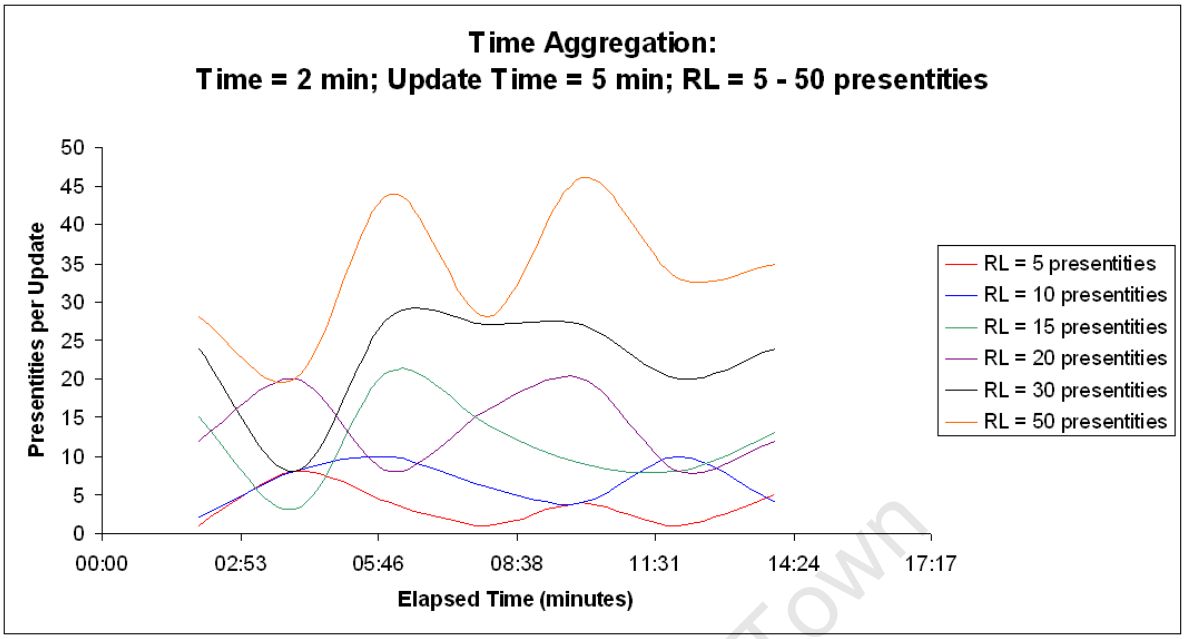

Figure 5.11: Time Aggregation: Time Parameter = 2 min, Presence Update Period = 5 min, RL Size = 5 - 50 presentities

Aggregation with Different Presence Update Periods Time aggregation, when examined against a series of different presence update periods, illustrates the difference between using a metric independent of the data stream, in comparison to size or quantity aggregation, where the metric for aggregation is dependent on it.

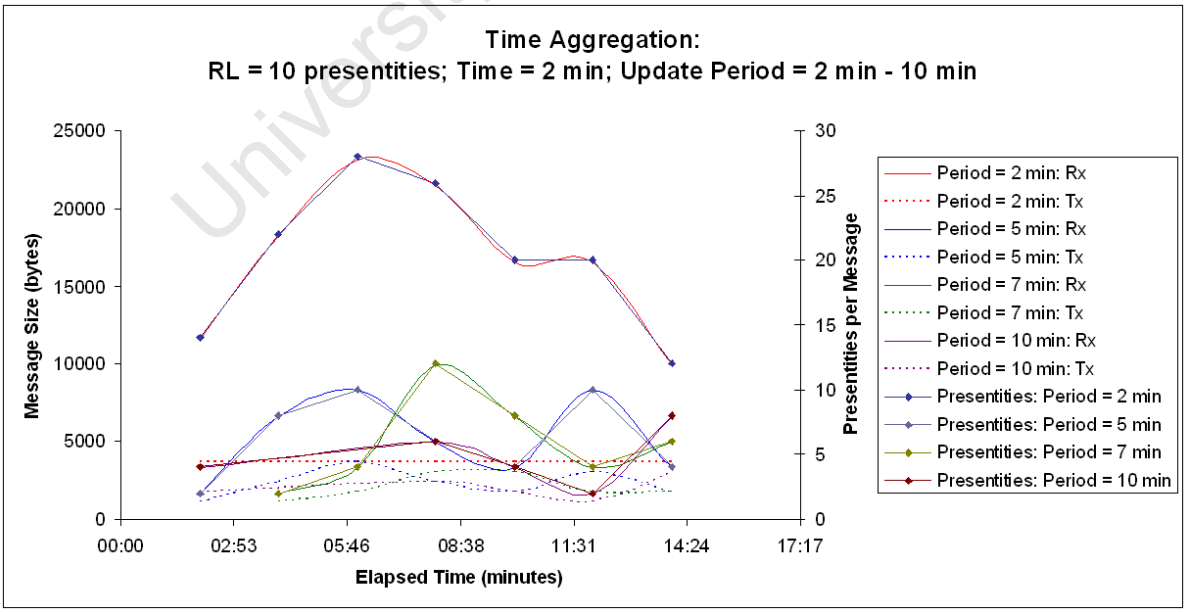

Figure 5.12: Time Aggregation: Time Parameter = 2 min; RL Size = 10 presentities; Presence Update Period = 2 - 10 min

Figure 5.12 demonstrates the effect of different presence update periods, as the RL size is fixed at 10 presentities and the time aggregation parameter is 2 minutes.

The aggregation parameter used in the test coincides with the minimum presence update time period examined, such that when the RLS forwards stored data, presence information for all the presentities in the RL has been received. This explains why the red dashed line, representing the transmitted (aggregated) data stream for the presence update period of 2 minutes, is flat, with no variation in data size. The amount of information received for this data stream is considerably larger than the aggregated stream, due to the number of secondary updates received from presentities, and the fact that only a single SIP header accompanies the forwarded message, instead of a SIP header for each individual update.

The other data streams in the test, for 5, 7 and 10 minute presence update periods, show the data received at the RLS to be considerably less than in the case of the update periods being 2 minutes. The degree of variability in the received data stream also decreases as the presence update period increases, as fewer updates are received at the RLS per NOTIFY sent, leading to smaller aggregated messages.

The right-hand axis in Figure 5.12 differs from similar examinations in quantity and size aggregation, because instead of measuring the time between successive aggregated transmissions (in this case, 2 minutes between each transmission), the value of interest is the number of presentities received. The number of presentities received per NOTIFY sent to the PUA naturally follows the same curve as the amount of data received, but is measured off the right-hand axis. As discussed regarding the data received at the RLS, the number of presentities received per NOTIFY sent is considerably higher for an update period of two minutes, when compared to the presentities per update for larger presentity update periods. The number of presentities received per NOTIFY message are indicated by the straight lines with the diamond point markers.

Performance of Different Aggregation Parameters Time aggregation shows subtly different characteristics from the two volume-based methods examined. The trigger for forwarding information from the RLS is not based on any metrics related to the data, but the expiration of a fixed-interval timer. Because the number of updates by the RLS per time-limited test can be calculated in advance, the test for an aggregation parameter of 5 minutes is run over 20 minutes instead of 15, in order to reduce the granularity of the data captured. This is shown in Figure 5.14.

Unlike size and quantity aggregation, time aggregation allows any number of presence updates to occur before forwarding the information. A limitation in the evaluation framework is however exposed by the time aggregation tests performed. The model for the updating of presence information by the numerous subscribed presentities uses random timeouts to trigger the sending of a presence update to the PS (in this case, from the PS directly to the RLS), in which the timeouts have an upper bound of 2, 5, 7 or 10 minutes. As shown in Figure 5.13 and 5.14, if the time value used for aggregation is the same or greater than the upper bound of the presence update period, all the subscribed presentities will have been updated when the timer expires.

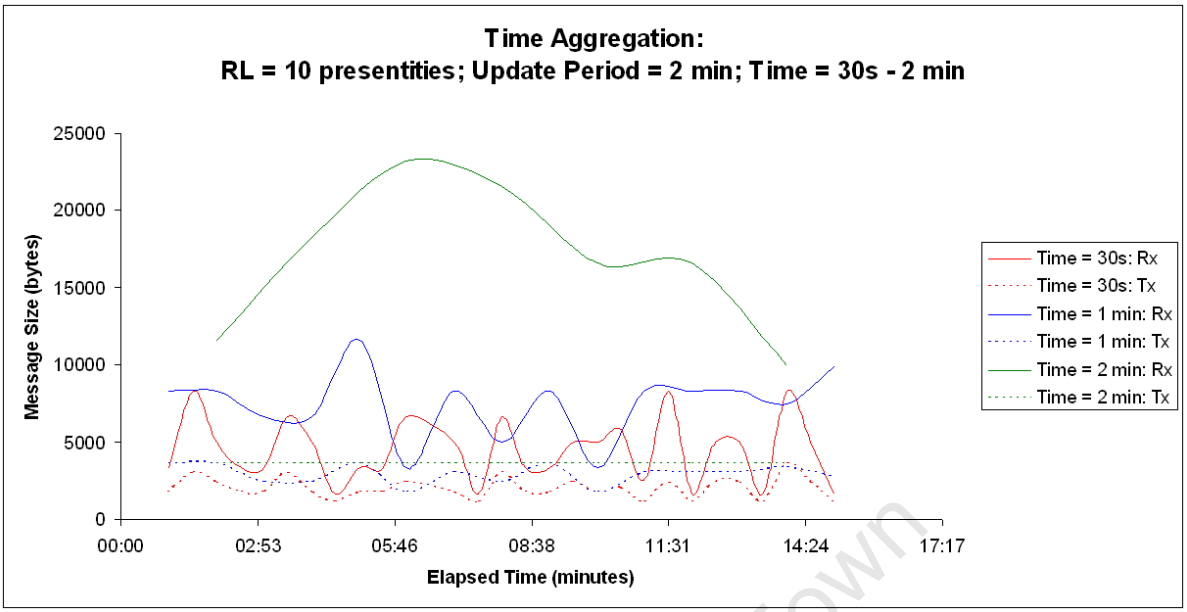

Figure 5.13: Time Aggregation: RL Size = 10 presentities; Presence Update Period = 2 min; Time Parameter = 30s - 2 min

This is illustrated by the transmitted data (Tx 2m) curve for a time parameter of 2 minutes in Figure 5.13, and by the transmitted data (Tx 5m) for a 5 minute parameter in Figure 5.14. While a real-world implementation is unlikely to be subject to such limited models, it does illustrate the importance of an applicable time aggregation parameter value. The use of an aggregation parameter larger than the upper bound used by presentities to update will lead to the delivery of stale information. While much of the received information may have been invalidated by subsequent updates, the objective of the RLS is to deliver as much of the received information as possible timeously, without it being overwritten.

The two figures show that the data streams from the RLS are relatively stable over time, with the data reduction increasing as the parameter values increase. The rate of increase in size of the aggregated data stream decreases with larger aggregation parameters, until the aggregation parameter is equal to the upper bound of the presence update period, as explained previously.

5.2.1.4 Time and Size Aggregation

The aggregation methods described above have their individual benefits and drawbacks. The use of combinations of these methods is hoped to provide aggregation performance that takes advantage of the benefits of each method, while eliminating the respective drawbacks encountered.

Aggregation with Different Resource List Sizes The combination of two aggregation methods offers the likelihood of better overall performance, but does also restrict the range

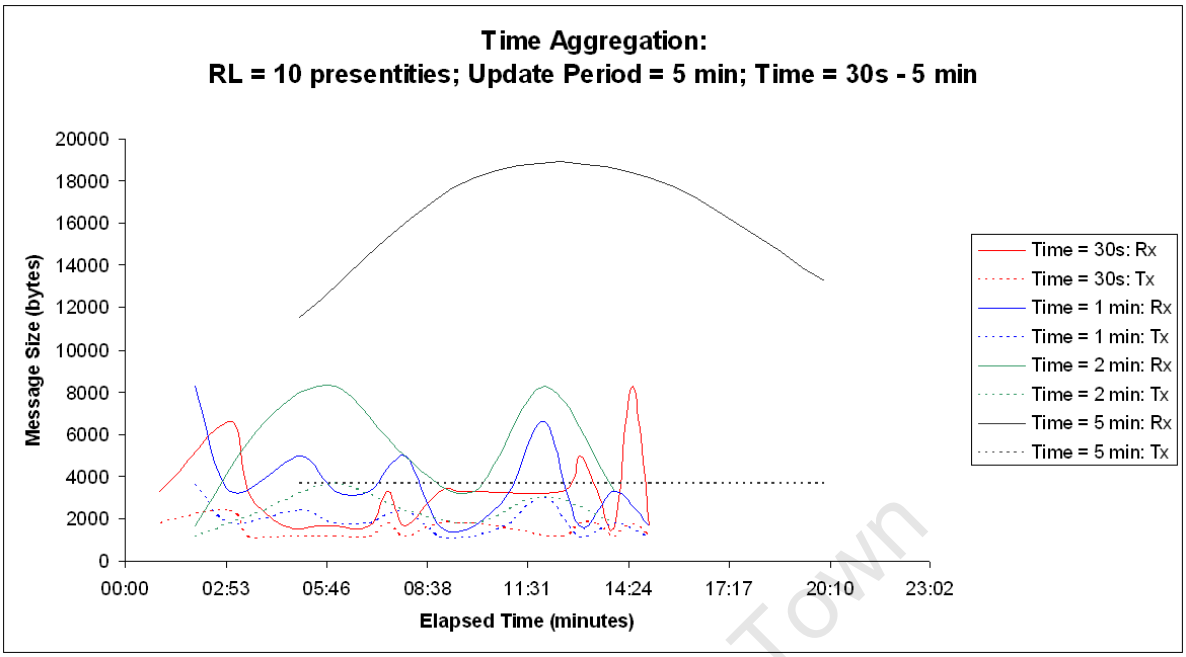

Figure 5.14: Time Aggregation: RL Size = 10 presentities; Presence Update Period = 5 min; Time Parameter = 30s - 5 min

over which these combinations are effective. The effective range for the combined methods is expected to change as the number of presentities contained in the RL increases.

The size of data transmitted to the PUA by the RLS appears to be relatively stable, with the dips in message size caused by the timers expiration. The size aggregation method appears to perform most of the aggregation, because of the relatively stable size of transmitted messages at just over 2 kB. There is considerable jitter between messages, causing a data stream that is not particularly stable with regards to inter-message timing and message size.

Because the implementation does not reset the timer every time an aggregated message is sent, the timer may expire soon after a size-aggregated message is sent. This is clearly not ideal as each method executes individually, rather than employing the metric-based method, where time aggregation ensures the timely delivery of information if no aggregation occurs before the timer expiration. This effect is clearly shown in Figure 5.15 by the straight lines with point markers. The steep drops in time between consecutive messages, as indicated for example by the orange "RL 30 time" line at the 2 minute mark on the x-axis illustrates this problem of independent aggregation method operation.

Aggregation with Different Presence Update Periods Examined in the following tests is the change in performance for different presence update periods, when examined under fixed aggregation parameters. As the presence update period changes, the workload distribution between time and size aggregation varies.

This is shown clearly in Figure 5.16, where aggregation performed on the data stream with a

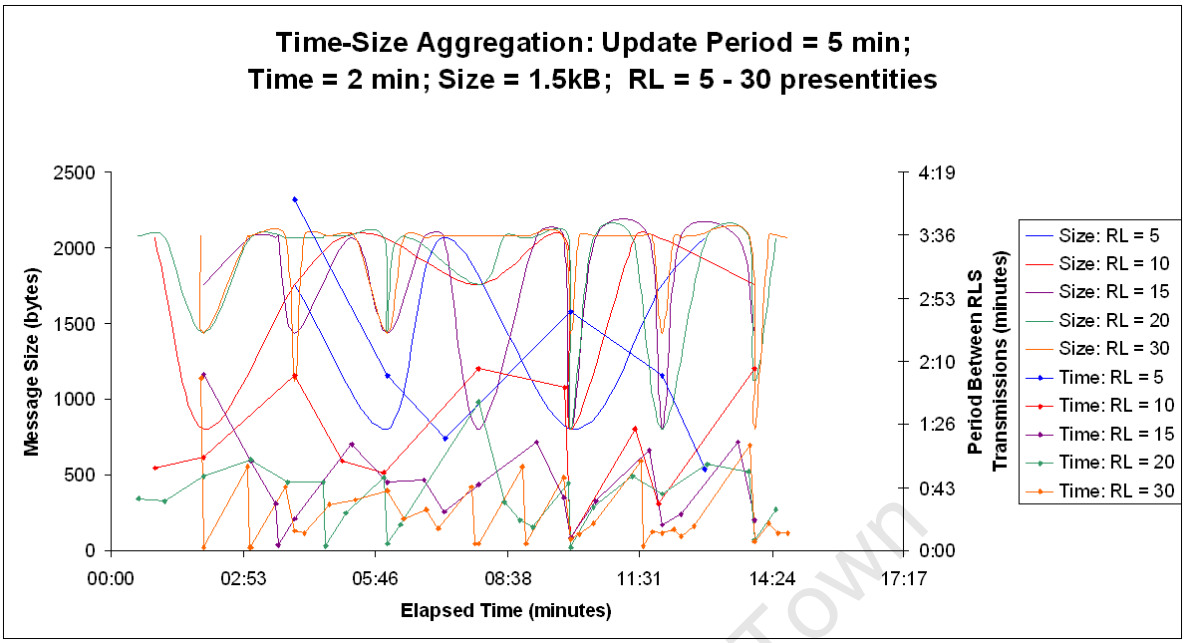

Figure 5.15: Time-Size Aggregation: Presence Update Period = 2 min; Time = 2 min, Size = 1.5kB; RL Size = 5 - 30 presentities

presence update period of 2 minutes is dominated by the size aggregation method, shown by the relatively flat line lying just above 2kB. Although the size aggregation parameter is set at 1.5kB, multiples of the average message size in the test first exceed the 1.5kB threshold with a value slightly larger than 2kB. The stability of the aggregated data stream for a presence update of 2 minutes is also affected by the choice of 2 minutes as the time parameter for aggregation. This is because when the aggregation timer thread expires, the presentities in the RL will have updated at least once.

The role played by size aggregation when applied to data streams with larger presence update periods is reduced, and the effect of time aggregation becomes more dominant. The change in dominant aggregation methods is anticipated because longer update times for presentities cause fewer updates to be forwarded to the PS and RLS. The reduced stream at the RLS does not surpass the size aggregation threshold as often, leading to smaller message sizes, and more periodic NOTIFYs being sent to the PUA.

Trend lines have been added to Figure 5.16, in order to better illustrate the overall trends exhibited. The trend lines indicate very similar performance characteristics for presence update periods of 7 minutes and 30 seconds, and 10 minutes. This similarity can probably be attribute to the smaller size of the time aggregation parameter, in comparison to the two update periods.

Performance of Different Aggregation Parameters Making accurate and well-grounded comparisons between the performance of different aggregation parameters when used in such combined-method aggregation is exceptionally difficult. This is because selecting parameter values that are well-matched to the characteristics of the examined data stream is difficult, and

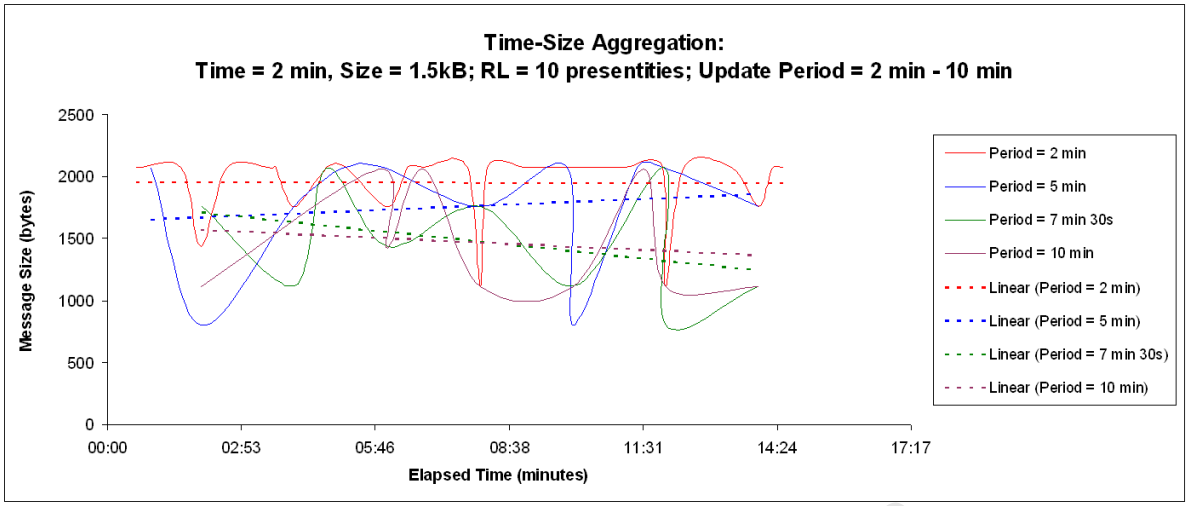

Figure 5.16: Time-Size Aggregation: RL Size = 10 presentities; Time = 2 min, Size = 1.5kB; Presence Update Period = 2 - 10 min

because the operation of the two aggregation methods are significantly different.

For this reason, the performance of a combined aggregation method is examined by keeping one aggregation parameter static, while a range of applicable values are examined for the other parameter.

Examined first is the effect of a changing time parameter, while the size aggregation parameter is held fixed at 1.5kB. The tests are performed using a RL containing 10 presentities, and a presence update time of 2 minutes. The data sets in Figure 5.2.1.4 display the size (in bytes) of the data stream from the RLS to the PUA.

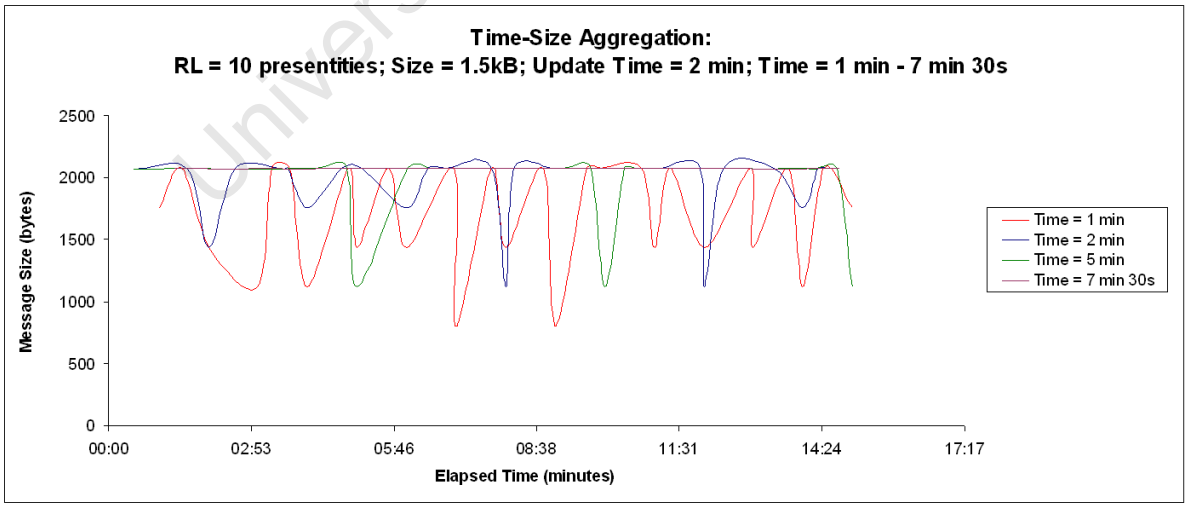

Figure 5.17: Time-Size Aggregation: RL Size = 10 presentities; Presence Update Period = 2 min; Size = 1.5kB; Time = 1 - 7min30s

The data sets shown indicate that for the smaller time parameters of 1 and 2 minutes, the

combined performance of the two aggregation methods is relatively well matched. The evidence for this is the number of dips in message size below the average for the size aggregation method, indicating that both methods are effectively operating. The effect of time aggregation in the aggregation of the data stream decreases as the value of the time parameter increases, to the point that size aggregation becomes the only aggregation method acting on the data stream across the RLS at a time value of 7 minutes and 30 seconds.

The investigation into the effect different parameter values exhibit for size aggregation differs from that in Figure , because inter-message timing is the metric most closely matching that of message size for time aggregation.

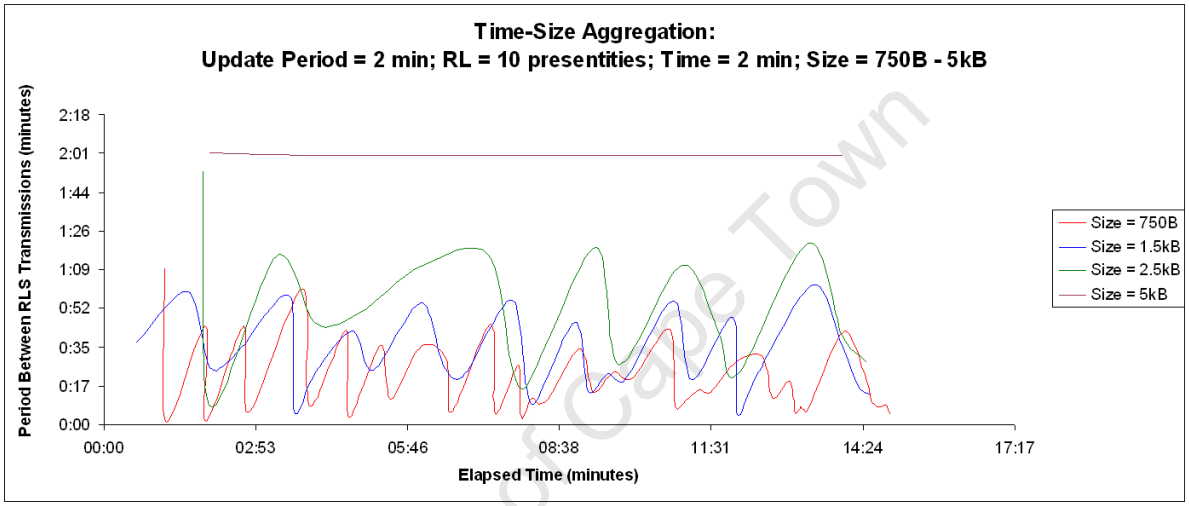

Figure 5.18: Time-Size Aggregation: RL Size = 10 presentities; Presence Update Period = 2 min; Time = 2 min; Size = 0.75 - 5kB

Relevant parameters from Figure 5.2.1.4 are reused such as the RL size of 10, and the presence update time of 2 minutes.

Figure 5.18 shows the results of varying the size aggregation parameter, with the time parameter static. The smallest value for the size parameter, 750b, unsurprisingly displays the shortest period of time between messages, although with jitter. The performance of a size parameter of 1.5kB is very similar to that of the 750b data stream, indicating that both cases are likely to have similar numbers of presentities per NOTIFY message sent to the PUA. The increase in parameter size to 2.5kB again displays similar performance, with relatively high jitter, but it is the data stream from a 5kB size aggregation parameter that is significantly different. The straight line occurring at two minutes is indicative of a poor size parameter choice for this data stream. There is no indication in the three other data streams examined of any time-based aggregation occurring. The choice of 5kB is clearly too large, and the number of presentities subscribed to in the RL cannot, under the conditions, supply enough unique information to the RLS to cause size aggregation to be performed.

5.2.1.5 Time and Quantity Aggregation

The second combination of aggregation methods examined substitutes the use of size as an aggregation method for quantity.

Aggregation with Different Resource List Sizes As discussed for Time-Size aggregation, the effectiveness of combined aggregation methods is likely to be significantly affected when the data stream is altered. Optimal performance in aggregating an incoming data stream by the combined methods is likely to occur in a relatively small region where the aggregation parameters match the parameters governing the data stream. Outside of this region, either one of the aggregation methods is likely to monopolise aggregation of the incoming data. This move to single-method aggregation can result in either improved or degraded performance, which depends on how the data stream changes in relation to the aggregation parameters employed.

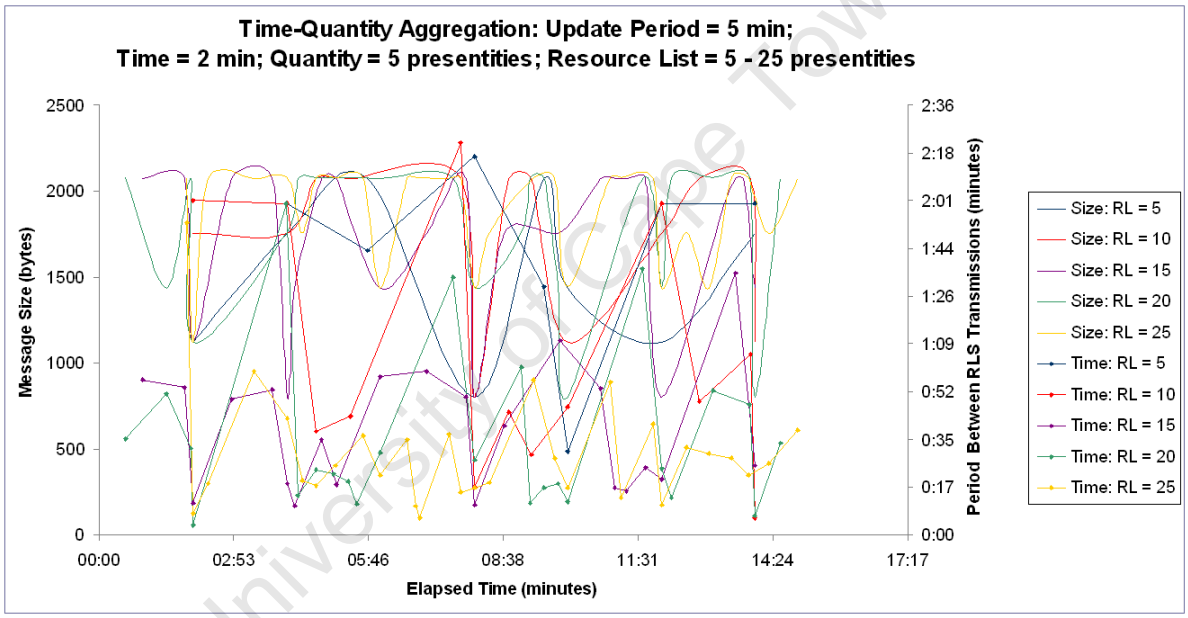

Figure 5.19: Time-Quantity Aggregation: Time = 2 min; Quantity = 5 presentities; Presence Update Period = 5 min; RL Size = 5 - 25 presentities

Figure 5.19 compares the aggregation of a series of different datastreams, each with a unique RL size. Each data stream is aggregated by the same set of time and quantity aggregation parameters, resulting in a series of data sets in which the effect of RL size can be examined. The smoothed data series are mapped to the left vertical axis, which provides a comparison of the message sizes transmitted to the PUA, per data stream. The datasets are measured for message size in bytes, as indicated in the legend by "RL 5 size", for example.

The upper bound for message size is restricted by the quantity parameter, which in the figure is limited to 5 presentities. While the quantity parameter creates a moderately stable upper bound for the message size, the independent operation of time aggregation succeeds in reducing the message size at regular intervals. The reduction in size is not uniform, as the amount of data

received since the last NOTIFY was sent is dependent on the activity of subscribed presentities in the RL.

As can be seen in Figure 5.19, data sets with smaller RL sizes, such as those containing 5 or 10 presentities (shown as blue and red smooth lines), exhibit a greater variation in message size than the data sets for larger RL sizes such as 25 presentities. The larger RL sizes show a lower limit for the message size that is considerably higher than for the smaller RL sizes.

Also of interest is the period of time between successive update messages for each data stream. The inter-message timing for each data set is plotted against time on the right-hand axis of Figure 5.19, and represented by the marked data-points joined by direct lines, indicated in the legend by the data sets marked as "time".

As expected the time between messages displays the greatest variation for lower RL sizes, as the accumulated presentities do not exceed the quantity parameter very quickly. The jitter is large, with the time between messages ranging from approximately 30s to over 2 minutes. In the case of large RL sizes, such as the RL containing 25 presentities, the time between messages is much less, ranging from approximately 10s to 50s.

The resultant aggregated message size and the inter-message timing yields a relatively stable data stream from the RLS to the PUA for larger RL sizes. For the smaller RL sizes however, the large message jitter and variation in message sizes makes the data stream relatively non-uniform. The result of this will make the effective allocation of resources at network management nodes outside the IMS core more difficult.

Aggregation with Different Presence Update Periods Changes in the update period of presentities in the subscribed RL influence the volume of presence information received by the RLS. Attempting to maintain a constant level of service and aggregation effectiveness across such changes is possible, but require that one or more of the aggregation parameters in use are adjusted to compensate for the change.

Examined in Figure 5.20 is the effect a variation in the update period has on a fixed set of aggregation parameters. For the test in question, a RL of 10 presentities is aggregated by a combined time parameter of 2 minutes and quantity parameter of 5 presentities. Presence update periods of 2, 5, 7 and 10 minutes are examined.

The figure presents similar results to the examination in Figure 5.16, where the fastest updating RLs are dominated by the size aggregation method. In the case of Figure 5.20, the size aggregation method is naturally replaced with quantity aggregation, and the overall effect is much the same. The transmitted message size remains very close to 2kB, with the linear trend line reinforcing the stable message size characteristic shown by the blue data set.

The larger update periods exhibit a more even distribution between time and quantity aggregation, with time aggregation becoming more pronounced as the update period moves towards 7 minutes. An update period of 10 minutes shows that very little quantity aggregation is performed on the data stream, as the message size only reaches 2kB at one point. This is due to

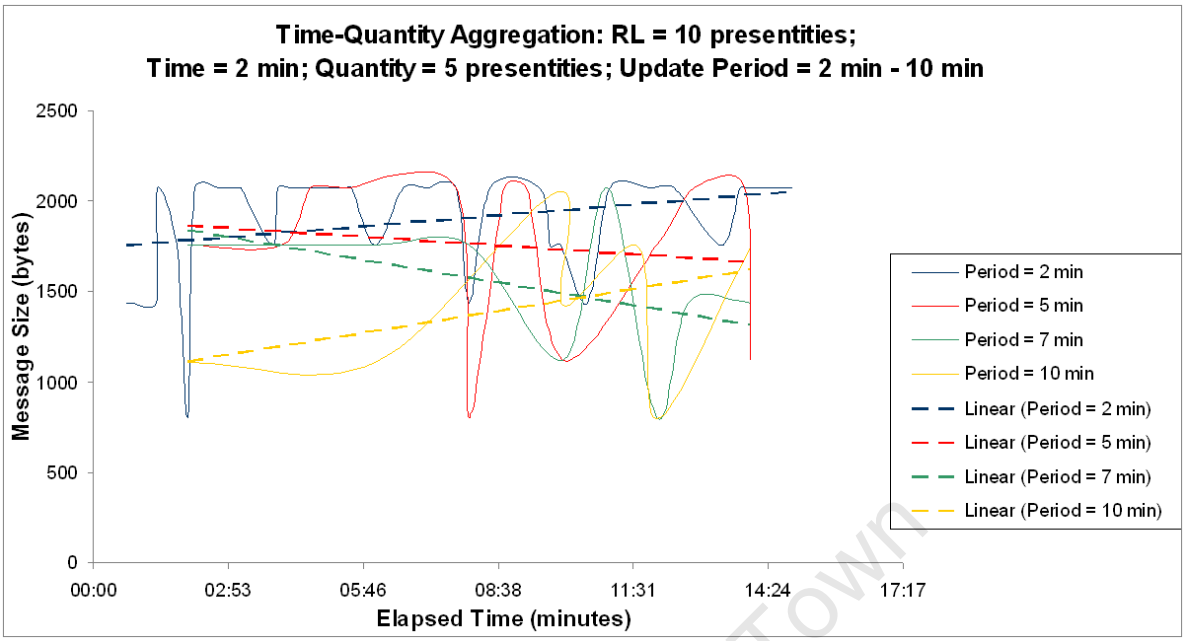

Figure 5.20: Time-Quantity Aggregation: Time = 2 min; Quantity = 5 presentities; RL Size = 10 presentities; Presence Update Period = 2 - 10 min

the rate at which presence information is accumulated at the RLS slowing with longer update times. The objective with regards to choosing ideal parameters for the two methods is to obtain an aggregated data stream that is both stable in terms of the size of messages to the PUA, and regular, without either very long or short time periods between successive messages.

The linear trend lines are included in the graph to provide indications of the overall effect of aggregation. The instances of non-zero trend line gradients for the projected message sizes are expected to tend to zero over longer test periods.

Performance of Different Aggregation Parameters As in Section 5.2.1.4, one aggregation method parameter is kept static, while a range of values for the other parameter is examined.

First, we study the effect a varying time parameter has on the aggregated data stream, where the quantity parameter is held fixed at 5 presentities. Four time aggregation parameters, from 1 minute to 7.5 minutes, are illustrated in Figure 5.21. Smaller time periods for aggregation naturally lead to more regular updates, which are shown to have small message sizes. The smaller time periods do however also lead to a larger range of message sizes. This is most likely attributable to the accumulated messages occasionally reaching the threshold for quantity aggregation, and being aggregated. The subsequent time-based aggregation thread then collects the small quantity of presence information accumulated since that point, in a relatively small message. The level of variability in the data stream aggregated at 1 minute indicates that for this data stream parameter, time aggregation is the primary method.

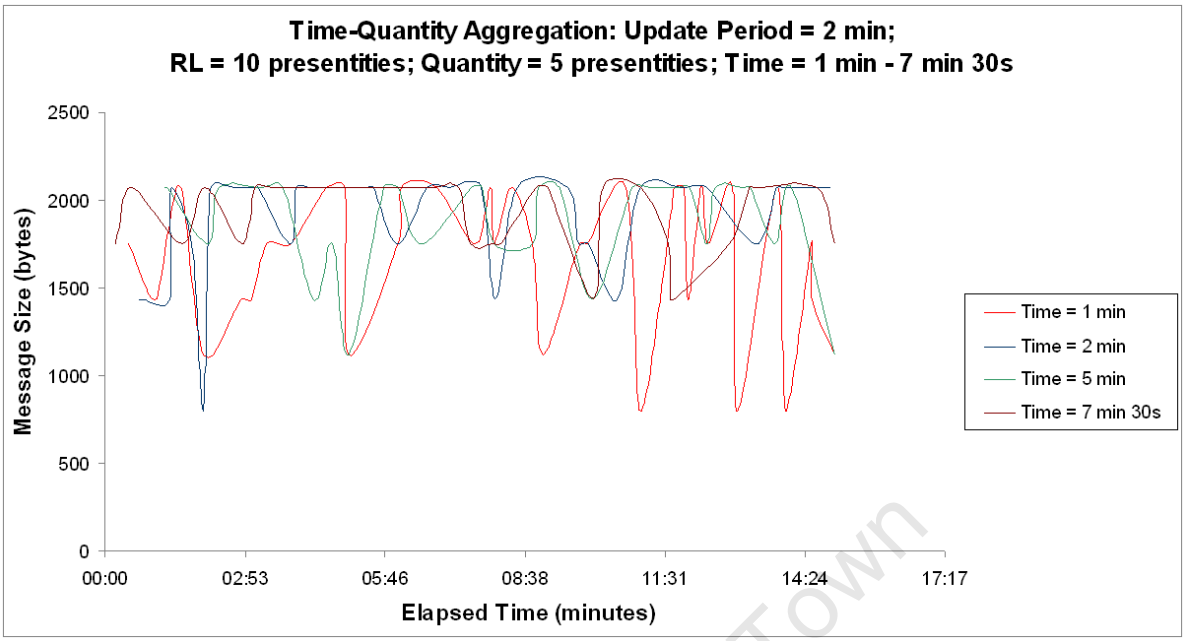

Figure 5.21: Time-Quantity Aggregation: RL Size = 10 presentities; Presence Update Period = 2 min; Quantity = 5 presentities; Time = 1 - 7min30s

At the other end of the scale, quantity aggregation is the dominant method, generating the majority of the NOTIFY updates. This is shown by the purple load curve for 7 minutes and 30 seconds, which is very flat at just above 2kB for most of the test, and shows very little variation in the message sizes sent to the PUA. This is an indication that for the aforementioned data stream parameters, the larger time values do not perform effective aggregation.

The effect of a varying quantity aggregation parameter is examined in Figure 5.22. The data for five different data sets is presented, with quantity parameter values of 5, 10, 15, 20 and 25 presentities. The time parameter used in the tests is 2 minutes, while the presence update period is 2 minutes and the RL size is 25 presentities.

In contrast to Figure 5.21, the message sizes in the aggregated data streams are more influenced by the varying aggregation parameter, rather than the fixed parameter. The fixed parameter of time also affects each data stream uniformly, which leads to the dips in the average message size across all the data streams at 2 minute intervals.

For the smaller quantity parameters such as 5 and 10 presentities, the action of quantity aggregation is dominant, as the message size remains very static, particularly for 5 presentities. The dips in message size not centered on multiples of 2 minutes are the result of a presentity updating its information twice between successive NOTIFY messages, and in so doing reducing the number of unique presentities transmitted.

The data streams for larger quantity parameter values such as 20 and 25 presentities also exhibit a significant influence from the quantity aggregation parameter, while the effect of time aggregation becomes more pronounced. While the average message size increases with the aggregation

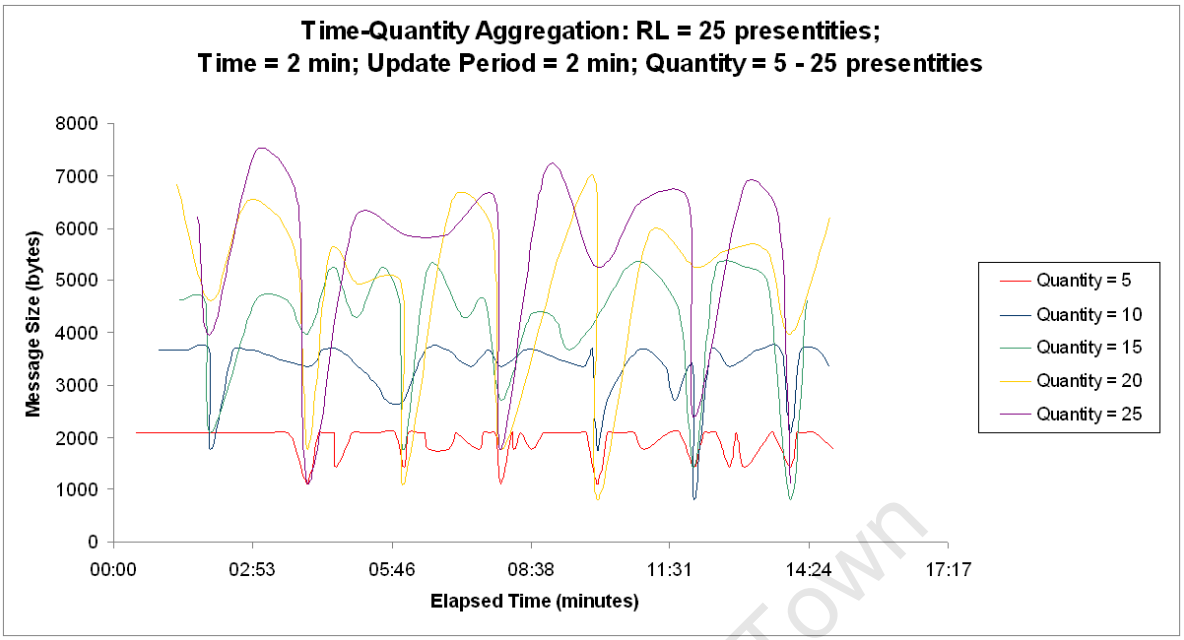

Figure 5.22: Time-Quantity Aggregation: RL Size = 25 presentities; Presence Update Period = 2 min; Time = 2 min; Quantity = 5 - 25 presentities

parameter, the effect of regular time aggregation results in significant drops in the message size, and leads to the message sizes in the data stream becoming rather unstable. The mismatch of parameters for the larger quantity values demonstrates the importance of well chosen aggregation parameters for effective RLS operation.

5.2.2 Initialisation State Operation

The discussion and examinations in Section 5.2.1 deal exclusively with the operation of RLS aggregation techniques under a steady-state condition, in which the subscription to the Resource List has been made, and the initial transfer of complete sets of presence information for the subscribed presentities has taken place. Under such conditions, the flow of presence update information from the presentities being watched can be considered to be relatively stable.

5.2.2.1 The Challenge of Initialisation State Aggregation

The start-up phase of an RLS subscription is considerably different however, as it is characterised by the full set of presence information of the subscribed presentities being transmitted by PSs to the RLS, and then aggregated by the RLS for the PUA. The resulting spike in data at the RLS must be stored and aggregated before being forwarded to the PUA, across an access network that may be bandwidth constrained, and shared between multiple other users. The large volume of data may also monopolise the network resources of the PUA, putting other applications that may utilise the resource at a disadvantage.

The challenge at the RLS under such conditions is to aggregate and forward the data received, without monopolising the network connection at the PUA. Also of importance is maintaining a steady data stream and ensuring the timely delivery of the presence information. The duration of the initial stage is not particularly long, and the incoming data stream is likely to stabilise within a few seconds to a minute or so. It is therefore also important for the RLS to recognise the change in the state of the subscription, and adapt or change the aggregation parameters currently in use to yield better performance.

5.2.2.2 Initialisation State Aggregation Testing

The tests performed for initialisation state aggregation are the same as in Section 5.2.1, for steady state subscriptions. The reaction of the simulated presence servers however was considerably different, with each subscription at the PS receiving the presence information of a full presentity as the initial response. After the initial full presentity is forwarded to the RLS, the subscription at the PS enters a steady state, and operates in the same manner as the steady state tests performed. Because the initial spike in data to the PUA is relatively short, each test is run for only 3 minutes, rather than the 15 to 30 minute durations used for the steady state tests.

The data generated in these tests is considerable, and because a rigorous examination of steady state RLS aggregation has already been performed, similar analysis has not been performed on this data. The results of these tests are included in Appendix D for future work.

5.3 Comparisons of Aggregation Methods

This section provides a brief comparison between the different aggregation methods examined when operating under similar conditions, with similar parameters for each method. Although in certain scenarios the methods may not be directly comparable, it serves to illustrate the differences in the operation of the different aggregation techniques, as well as the impact of design restrictions not considered at the development stage.

Due to analysis for Initialisation State aggregation operation not being performed (Section 5.2.2), there is data available to compare between aggregation methods under such a state. This is again left for future work.

5.3.1 RL Size Comparison

A change in the RL size has been shown previously to affect the performance and effectiveness of the different aggregation methods. This section examines how the performance of the three aggregation methods change in relation to one another as the RL size varies. The results for the tests of the two RL sizes examined are presented in separate figures (Figure 5.23 and Figure 5.24), in order for the plotted data to remain clear and comprehensible. The three aggregation parameters (size = 1.5kB, quantity = 5 and time = 1 minute), are kept static for both tests.

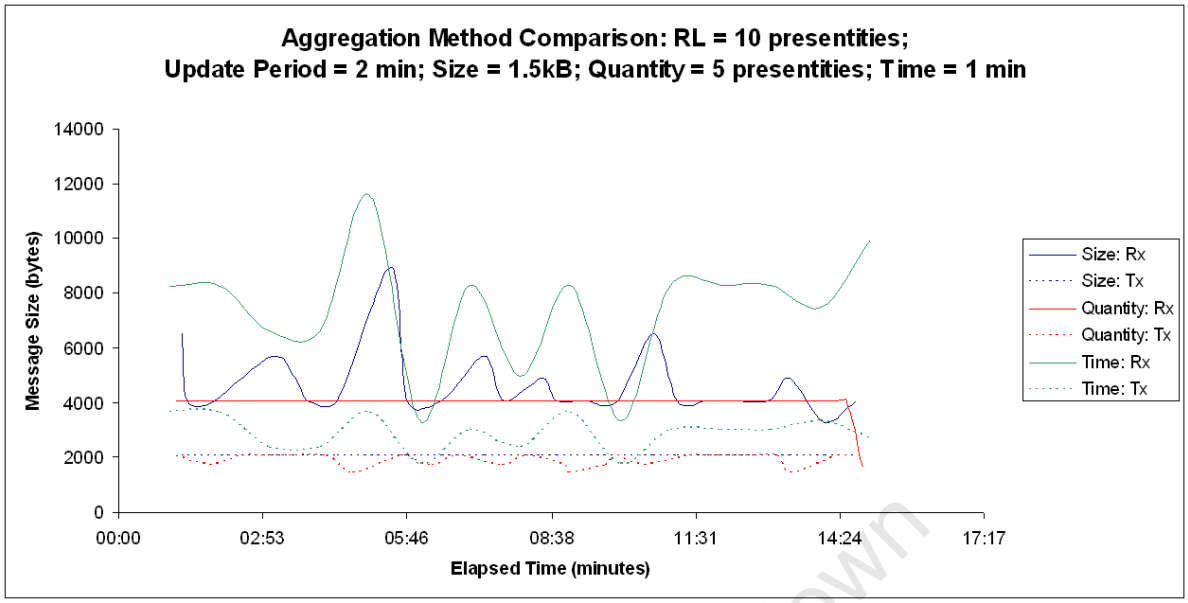

Figure 5.23: Aggregation Method Comparison: RL Size = 10 presentities; Presence Update Period = 2 min; Size = 1.5kB; Quantity = 5 presentities; Time = 1 min

Figure 5.23 summarises the lower RL size test scenario, with an RL of 10. The parameters for the three aggregation methods are chosen to provide aggregated data streams of approximately equal data size.

The subtle variations in implementation become clearly visible in the figure, as quantity aggregation keeps the received data per NOTIFY message constant with some variation on the data transmitted, while size aggregation results in a variable received data curve, and produces a constant transmitted data size. The operation of time aggregation being independent of the incoming data stream results in no such patterns emerging, but instead shows that the size curve of the transmitted data follows that of the received data.

The transmitted data from the RLS is of similar size across all three methods examined, with the most variation in size being from the size aggregation data stream. This mirrors the received information for the three methods, in which the received data for size and quantity aggregation is kept relatively stable and low, while the received time aggregation stream shows considerable size variation over time, in comparison.

Figure 5.24 examines a test with the same parameters as in Figure 5.23, but with an RL size of 20 presentities instead. The larger RL size does not affect the transmitted data from either size or quantity aggregation, and the received data size of the two aggregation methods is relatively unaffected as well. The major effect the change in RL size has is on time aggregation, where there is a significant increase in the volume of received data, which still exhibits considerable variation in message size over time. The data transmitted from the RLS for time aggregation also increases, to the point where it exhibits a similar size to that of the received data streams for size and quantity.

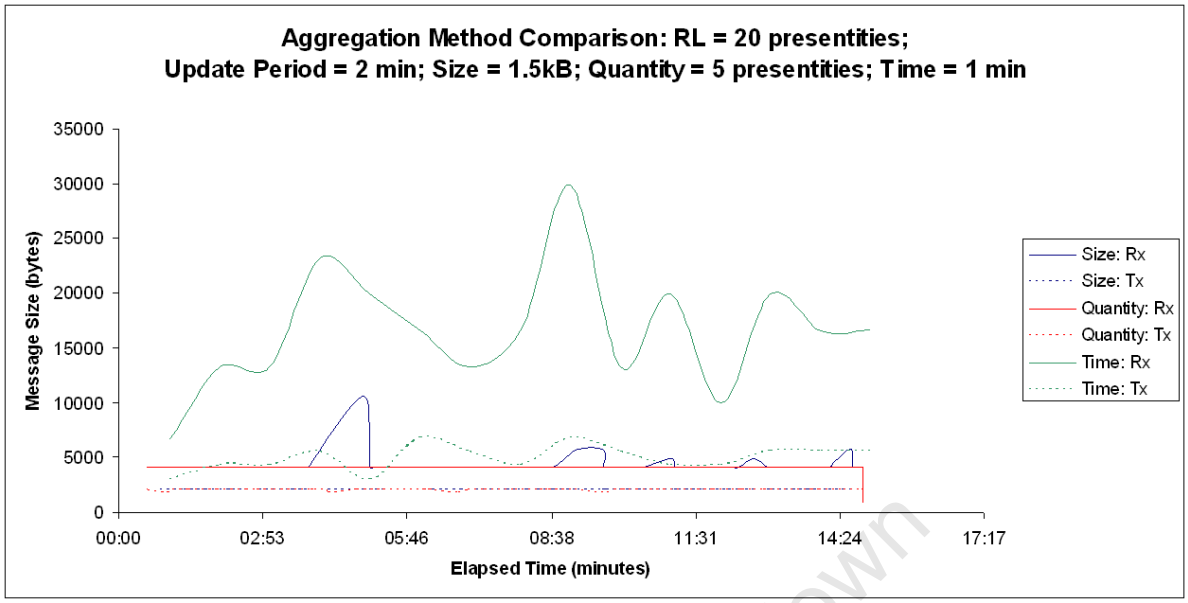

Figure 5.24: Aggregation Method Comparison: RL Size = 20 presentities; Presence Update Period = 2 min; Size = 1.5kB; Quantity = 5 presentities; Time = 1 min

5.3.2 Aggregation Parameter Comparison

Accurate comparisons of different aggregation parameters are very difficult to make, as the parameter values for all three methods are quite different. Size and quantity aggregation are also sufficiently different in operation, even though they are both dependent on the data received from subscribed presentities.

The question of what exactly to measure also becomes important, as different metrics are required to gain an insight into the operation and efficiency of the various methods. The combination of Figures 5.23 and 5.25 provide a cursory comparison of different aggregation parameters.

Although the aggregation parameters in the two figures differ, it is possible to show the differences in how the parameters affect aggregation, and their effectiveness. The first notable point is the consistency of quantity aggregation across different RL size and aggregation parameters. The difference in size between the received and transmitted data streams for quantity aggregation remains very stable, and in both cases the received data stream is relatively close in size to the transmitted data stream. This is notable when contrast to the received data for size aggregation across the two figures. While the transmitted data streams in both figures are similar, the received size-aggregated data stream in Figure 5.25 is considerably larger in size than that for quantity aggregation, suggesting that the data received is not aggregated and forwarded to the PUA as quickly and efficiently as is by quantity aggregation, for the update parameter in use. This may not be particularly visible at the lower parameter values, but does become significant at such higher parameters.

Demonstrating an effect that was not present in Figure 5.23, the transmitted data for the time aggregation method in Figure 5.25 is totally static. This is because the model for presence

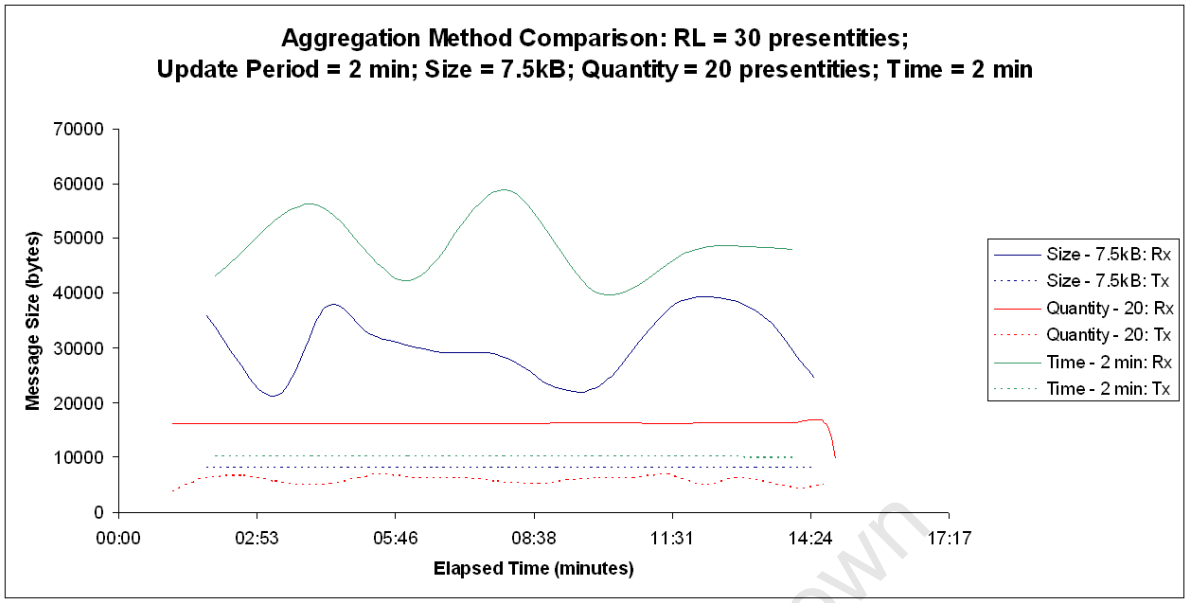

Figure 5.25: Aggregation Method Comparison: RL Size = 30 presentities; Presence Update Period = 2 min; Size = 7.5kB; Quantity = 20 presentities; Time = 2 min

updating activity has an upper limit of two minutes, matching the time aggregation parameter in use. This means that when the aggregation timer expires and causes the received information to be forwarded to the PUA, the entire RL in question has updated its information, so that presence information for all presentities has been received, and is forwarded. This is not an ideal situation, but can be seen as a limit of the model, rather than a failing of the aggregation method.

5.3.3 Presence Update Period Comparison

A change in the presence update period for a data stream passing through the RLS can affect the aggregation period quite significantly. The manner in which the aggregation method is affected by the change is not the same for all methods. For both size and quantity aggregation, an increase in the update period of the subscribed presentities results in longer delays between aggregated messages, as the number of presentities received, and hence the amount of data stored per unit of time decreases. As both of these methods rely on numerical thresholds being reached for aggregation to occur, the effect is very similar.

The effect of fewer NOTIFYs from PSs passing through the RLS per unit of time affects time aggregation quite differently to either size or quantity aggregation. Because the aggregation in this case is periodic, an increased presence period causes fewer presentities to be received and stored by the RLS per aggregation time period. This leads to a reduction in the transmitted message size from the RLS, while the period of successive updates remains unchanged.

Figure 5.26 compares the aggregated data streams of fixed-parameter size and quantity aggregation methods, over three presence update periods. The aggregation parameters for each method

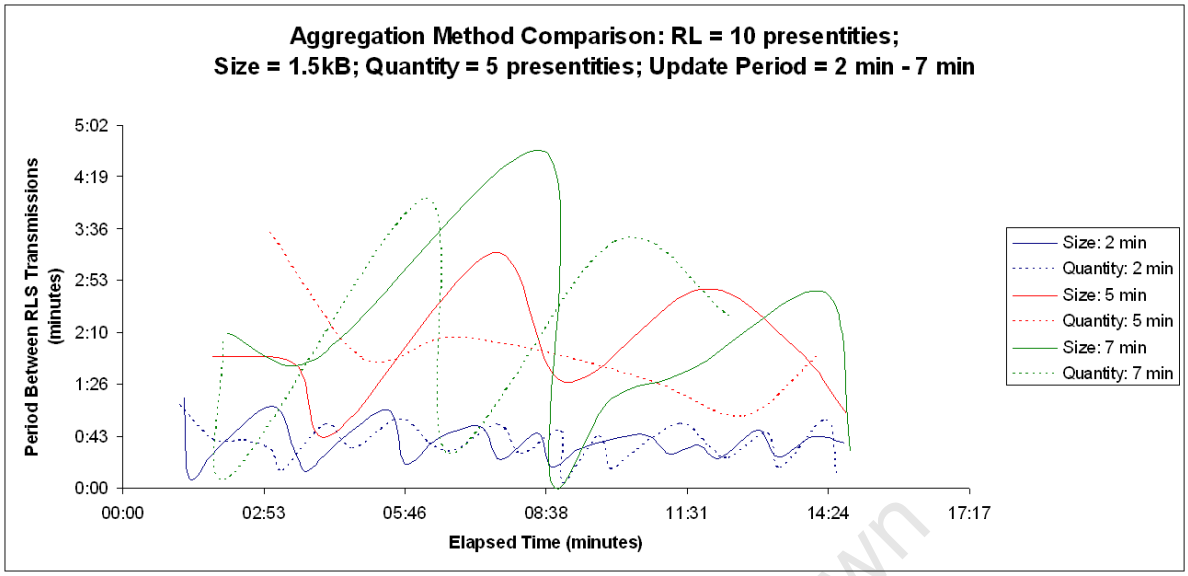

Figure 5.26: Aggregation Method Comparison: RL Size = 10 presentities; Size = 1.5kB; Quantity = 5 presentities; Presence Update Period = 2 - 7 min

(a size parameter of 1.5kB and quantity parameter of 5) are chosen to yield a similar aggregation performance for short presence update periods, in order to highlight any deviation between the methods that may occur for larger update periods.

As can be inferred from the figure, the two aggregation parameters are closely matched in average performance over the three update periods of 2, 5, and 7 minutes. Over a short update period such as 2 minutes, the aggregated data stream is stable with regard to inter-message timing, with little significant jitter. The data size of the aggregated stream is considered to be stable, as shown previously in the examination of the individual methods.

However, as the presence update period increases, the stability of the data stream decreases, particularly in the case of size aggregation. As the update period increases, the jitter becomes particularly serious for the size-aggregated streams, and clearly fail in the objective to provide stable data streams through aggregation. The result of quantity aggregation for the two larger presence update periods show the average inter-message time becomes larger as expected, but jitter in the transmission of the data becomes a serious concern for the largest update period.

The very different operation of time aggregation, when compared to size and quantity aggregation, makes its inclusion in this examination irrelevant. As shown in Figure 5.12 in Section 5.2.1.3, the size of the aggregated data decreases as the update period increases, and the inter-message timing remains constant, by virtue of the operation of the method.

Chapter 6

Conclusions and Recommendations

This chapter contains the conclusions drawn from the presented work. Some recommendations for possible future research are then discussed.

6.1 Conclusions

The objective of this thesis was to examine the operation of the IMS Resource List Server, and evaluate the proposed methods for effectively achieving the aggregation of presence information for subscribers to the presence service. This allows a better understanding of how aggregation at the RLS can be implemented, and provides a grounding for the selection of a particular aggregation method over another, in either research oriented or real-world RLS implementations.

An evaluation framework designed for testing various aggregation methods was developed. The methods examined were size, time and quantity aggregation. The analysis included hybrid methods, which combined time with size aggregation, and time with quantity aggregation. Three primary parameters governing the operation and performance of aggregation at the RLS were identified, and the effect of these parameters was examined for presence data streams for each of the 5 aggregation methods. The three parameters examined were the number of presentities in the resource list, the period between successive updates for a presentity (as a measure of presentity activity), and different aggregation parameters for the method. Useful results for the performance of each aggregation method were selected, presented and discussed.

The similarity between size and quantity aggregation followed through to the results obtained from the two methods, which corresponded with each other under similar operating conditions. Differences that emerged between the two methods under similar test conditions were attributed mainly to unintentional differences in implementation. The primary result of this was that for size aggregation, the volume of aggregated data (from the RLS) was very stable with no variation over time, while the volume of data received was unrestricted and could vary in size over the period of the tests. This effect was reversed in the implementation for quantity aggregation, as the aggregated data volume showed a degree of variation over time, while the received data

volume was fixed, matching the aggregation parameter value. For different aggregation parameters, effective aggregation was achieved by both methods, although individual results differed slightly. The deviation between the methods is particularly visible when examining the effect of varying aggregation parameters, as size aggregation yields stable aggregated data volumes but variable received data volumes, while the trends in the results for quantity aggregation are reversed.

The operation of time-based aggregation is considerably different to that of both size and quantity aggregation, and this was evident in the collected results. The use of time as the aggregator causes updates to the PUA to be perfectly uniform with regards to time, but does not limit the number of presence updates received, or the number of presence updates included in each NOTIFY to the PUA, and hence the size. The results obtained from the variation in data stream parameters are as expected, even though the metrics used in analysis had to be altered to yield useful data. The use of time aggregation, while effective under certain conditions, does not satisfy the requirements for a primary aggregation method particularly well, and should only be considered for use in the hybrid methods because of its inability to effectively control the volume of received or aggregated data.

The combination of different aggregation methods was used to determine whether the drawbacks inherent in size and quantity aggregation could be minimised or negated by coupling the methods with time aggregation. Such a combination can be used to ensure a maximum aggregated message size, and limit the time between consecutive messages to the PUA.

It was however not possible to achieve the optimal implementation of the combined methods. Furthermore, an ideal implementation of the combined methods would require a periodic thread per subscription managed by the RLS, limiting either the number of subscriptions that could be managed by the RLS, or severely limiting the operational performance of an RLS managing many subscriptions. The proposed implementations were effective under testing, and performed aggregation satisfactorily. Investigations into whether the timeliness of data delivery is preserved showed the current implementation ensures the timeous delivery of data, although not in the manner initially envisaged. Instead of the assurance that time aggregation would only be performed if the period between two consecutive size- or quantity-aggregated messages exceeded the time aggregation parameter value, time aggregation may occur at any point. The implication of this is that time aggregation may be performed at any point after a size- or quantity-aggregated message is generated, within a guaranteed maximum time period. While maintaining timeliness, this does reduce the effectiveness of the other aggregation parameter to produce a stable data stream with at least a minimum reduction in volume between the received and transmitted data.

6.2 Recommendations

During the course of this work, a number of issues relating to the scope, limitations and conclusions presented in the study were raised, which are suggested as avenues of potential future

work.

The work presented is based on the results obtained through experimental interrogation of the developed RLS framework. While aspects of the framework, in particular the developed RLS, were developed for and should be suitable for use in real-world tests and IMS presence architecture implementations, certain other aspects are based on what were considered to be reasonable assumptions.

The model used to govern the behaviour of presentities updating their presence information was developed by the author for the purposes of this investigation, and can be considered quite limited. The re-examination of aspects of this research is encouraged if and when other models for presentity behaviour become available. Models with input from, or based on data obtained from real-world presence applications utilised by a large, heterogeneous group of users are considered ideal.

Additionally, models dealing with smaller homogeneous groups of users with special characteristics could also be considered. Such groups could be distinguished by characteristics such as:

- Socio-economic circumstances (of the presentities subscribed to)
- Age (with subcategories of teenagers, young adults, middle aged and the elderly)
- Gender
- Geographic location
- The level of access to IMS infrastructure and presence applications
- The primary access technology utilised (cellular telephone, computer, etc)

Testing for extended periods with control groups comprising of human users is likely to provide incrementally improved implementations of RLS aggregation, and associated data. The ability of such real-world groups to upload and modify their own RLS update rules, specifying priorities for particular presentities, information of primary interest and other such criteria for aggregation is an aspect of RLS aggregation which should also be examined.

Models, in which fully converged IMS implementations are simulated or implemented, should be investigated when such resources become available. The model adopted in this work assumed the use of a single presence device generating presence information through user intervention, and resource lists comprising specific presentities of interest. IMS implementations, in which multiple user-interface devices contribute data for a single presentity, and resource lists are based on the contents of a subscribers' online telephone directory, form part of the ideal for a fully converged network. Core network elements such as Presence Network Agents will also be able to update presence information for subscribers using proxied signalling in the IMS core, providing an environment with significantly more presence information and presence interaction than is considered in this work.

An "ideal" implementation of the two hybrid methods may be considered for academic purposes, in which an individual thread for each subscription is maintained by the RLS. This will allow the implementation to reset the time aggregation timer thread, such that a guaranteed maximum time between aggregated messages can be implemented. Such an implementation will provide what is considered by the author to be the ideal operation of the hybrid aggregation methods, even though the implementation is unlikely to be suitable for deployment, due to the poor use of processor and other system resources for large numbers of subscriptions.

Bibliography

[1] G. Camarillo and M. A. Garcia-Martin, *The 3G IP Multimedia Subsystem (IMS) - Merging the Internet and the Cellular Worlds*, 2nd ed. John Wiley and Sons, Ltd, 2006.

[2] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W.Carr, and J. Peterson, "RFC 3863, Presence Information Data Format (PIDF)," 2004.

[3] G. Camarillo and M. A. Garcia-Martin, *The 3G IP Multimedia Subsystem (IMS) - Merging the Internet and the Cellular Worlds*, 2nd ed. John Wiley and Sons, Ltd, 2006, ch. 17, pp. 323–328.

[4] "Third Generation Partnership Project," *http://www.3gpp.org/*.

[5] "European Telecommunications Standards Institute - Telecoms and Internet converged Services and Protocols for Advanced Network," *http://www.etsi.org/tispan/*.

[6] A. Cuervas, J. I. Moreno, P. Vidales, and H. Einsiedler, "The IMS Service Platform: A Solution for Next-Generation Network Operators to Be More Than Bit Pipes," *IEEE Communications Magazine*, Aug 2006.

[7] D. Geer, "Building Converged Networks with IMS Technology," *Computer, vol. 38, no. 11, pp. 14-16*, Nov 2005.

[8] T. Kontzer, "Reuters Delivers IM For Financial Services," *InformationWeek*, Oct 14, 2002, http://www.informationweek.com/news/software/showArticle.jhtml?articleID=6503810.

[9] "Headworx", "Presence: Mobile Killer App," http://headworx.slupik.com/2006/11/presence-mobile-killer-app.html, Nov 2006.

[10] "Chris", "Messenger By The Millions," http://messengersays.spaces.live.com/?_c11_BlogPart_BlogPart=blogview&_c=BlogPart& partqs=amonthJan 2007.

[11] P. Wolff, "Skype Journal - Microsoft Messenger Claims Twice As Many Active Users as Skype," http://www.skypejournal.com/blog/archives/2006/08/microsoft_messenger_claims_twice_as_many.php, Aug 2006.

[12] Yahoo! Media Relations (UK & Ireland), "Yahoo! Extends Low Cost VoIP Services Strategy to the UK," http://investor.shareholder.com/yahooeu/releasedetail.cfm?ReleaseID=213742, Jun 2006.

[13] Symantic Security Response, "Top Five Instant Messaging Security Risks for 2006," May 2006.

[14] O. Malik, "Is Skype Showing Signs of Maturity?" http://gigaom.com/2007/04/20/is-skype-showing-signs-of-maturity/, Apr 2007.

[15] J. Mercier, "Skype Numerology," http://skypenumerology.blogspot.com/, Oct 2008.

[16] "Microsoft Office Communications Server 2007," http://office.microsoft.com/en-gb/communicationsserver/default.aspx.

[17] H. Yan, "TechNet Webcast: Implementing Instant Messaging/Presence and Conferencing in Microsoft Office Communications Server 2007 (Level 200)," http://msevents.microsoft.com/cui/WebCastEventDetails.aspx?culture=en-US&EventID=1032334972&CountryCode=US, Apr 2007, Microsoft Webcast.

[18] R. Good and N. Ventura, "Linking Session Based Services and Transport Layer Resource in the IP Multimedia Subsystem," *South Africa Telecommunications and Network Applications Conference*, 2008.

[19] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "RFC 3261, SIP: Session Initiation Protocol," 2002.

[20] M. Handley, V. Jacobson, and C. Perkins, "RFC 4566, SDP: Session Description Protocol," Jul 2006.

[21] 3rd Generation Partnership Project, "Technical Specification Group Core Network and Terminals; Presence Service Using the IP Multimedia (IM) Core Network (CN) Subsystem; Stage 3," Dec 2006.

[22] A. Niemi, M. Lonnfors, and E. Leppanen, "RFC 5264, Publication of Partial Presence Information," Sep 2008.

[23] M. Lonnfors, E. Leppanen, H. Khartabil, and J. Urpalainen, "RFC 5262, Presence Information Data Format (PIDF) Extension for Partial Presence," Sep 2008.

[24] S. Loreto and G. A. Eriksson, "Presence Network Agent: A Simple Way to Improve the Presence Service," *IEEE Communications Magazine*, Aug 2008.

[25] A. Chapman and H. T. Kung, "Traffic Management for Aggregate IP Streams," *3rd Canadian Conference on Broadband Research (CCBR'99)*, 1999.

[26] R. Chakravorty, J. Cartwright, and I. Pratt, "Practical Experience with TCP over GPRS," in *IEEE GLOBECOM*, 2002.

[27] J. F. Kurose and K. W. Ross, *Computer Networking - A Top-Down Approach Featuring the Internet*, International ed. Addison-Wesley, 2005.

[28] J. Rosenberg, "RFC 3856, A Presence Event Package for the Session Initiation Protocol (SIP)," 2004.

[29] A. B. Roach, "RFC 3265, Session Initiation Protocol (SIP)-Specific Event Notification," 2002.

[30] H. K. et. al., "RFC 4660, Functional Description of Event Notification Filtering," 2006.

[31] A. B. Roach, B. Campbell, and J. Rosenberg, "RFC 4662, A Session Initiation Protocol (SIP) Event List Notification Extension for Resource Lists," 2006.

[32] J. Rosenberg, "RFC 3857, A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)," 2004.

[33] 3rd Generation Partnership Project, "Technical Specification Group Services and System Aspects; Presence Service Stage 1," 2005.

[34] ——, "Technical Specification Group Services and System Aspects; Presence Service; Architecture and Functional Description," 2006.

[35] "Robust Header Compression (rohc) Working Group," http://www.ietf.org/html.charters/rohc-charter.html.

[36] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng, "RFC 3095, RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," 2001.

[37] R. Price, C. Bormann, J. Christoffersson, H. Hannu, Z. Liu, and J. Rosenberg, "RFC 3320, Signaling Compression (SigComp)," 2003.

[38] G. Camarillo, "RFC 3486, Compressing the Session Initiation Protocol (SIP)," 2003.

[39] T. Imaura, "US Patent 7143397 - XML Data Encoding and Decoding," *http://www.patentstorm.us/patents/7143397.html*, Nov 2006.

[40] N. Walsh, "A Technical Introduction to XML," *http://www.xml.com/pub/a/98/10/guide0.html?page=1*, p. 2, Oct 1998.

[41] W3C, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," *http://www.w3.org/TR/xml/*, Nov 2008.

[42] World Wide Web Consortium (W3C), "Efficient XML Interchange Working Group," *http://www.w3.org/XML/EXI/*, 2008.

[43] "Efficient XML," http://www.agiledelta.com/product_efx.html.

[44] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller, "Performance Considerations for Mobile Web Services," *Computer Communications*, vol. 27, no. 11, pp. 1097–1105, Jul 2004.

[45] A. Mani and A. Nagarajan, "Understanding Quality of Service for Web Services," *IBM DeveloperWorks, SOA and Web Services*, 2002, http://www.ibm.com/developerworks/library/ws-quality.html.

[46] C. J. Augeri, B. E. Mullins, L. C. Baird, D. A. Bulutoglu, and R. O. Baldwin, "An Analysis of XML Compression Efficiency," *Proceedings of the 2007 Workshop on Experimental Computer Science*, 2007.

[47] M. Day, J. Rosenberg, and H. Sugano, "RFC 2778, A Model for Presence and Instant Messaging," 2000.

[48] M. Day, S. Aggarwal, G. Mohr, and J. Vincent, "RFC 2779, Instant Messaging / Presence Protocol Requirements," 2000.

[49] J. Peterson, "RFC 3859, Common Profile for Presence (CPP)," 2004.

[50] H. Schulzrinne, V. Gurbani, P. Kyzivat, and J. Rosenberg, "RFC 4480, RIPD: Rich Presence Extensions to the Presence Information Data Format (PIDF)," 2006.

[51] J. Rosenberg, "RFC 3858, An Extensible Markup Language (XML) Based Format for Watcher Information," 2004.

[52] ——, "RFC 4826, Extensible Markup Language (XML) Formats for Representing Resource Lists," 2007.

[53] ——, "RFC 5025, Presence Authorization Rules," 2007.

[54] H. Khartabil, E. Leppanen, M. Lonnfors, and J. Costa-Requena, "RFC 4661, An Extensible Markup Language (XML)-Based Format for Event Notification Filtering," Sep 2006.

[55] J. Rosenberg, "RFC 4825, The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)," 2007.

[56] Singh, Schulzrinne, Isomaki, Boni, "Presence Traffic Optimization Techniques," http://app.cul.columbia.edu:8080/ac/bitstream/10022/AC:P:29478/1/426.pdf.

[57] "SIPp," http://sipp.sourceforge.net.

[58] "Sun Microsystems, Java," http://java.sun.com/.

[59] JSR 289 Expert Group, "SIP Servlet Specification, version1.1," 2008.

[60] "Mobicents SIP Servlets," http://www.mobicents.org/products_sip_servlets.html.

[61] "Mobicents," http://www.mobicents.org.

[62] G. Yan, Z. Xiao, and S. Eidenbenz, "Catching Instant Message Worms with Change-Point Techniques," *Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[63] H. Smith, Y. Rogers, and M. Brady, "Managing One's Social Network: Does Age Make a Difference," *Proceedings of INTERACT*, pp. 551–558, 2003.

[64] O. Jacques, "SIPp Reference Documentation," *http://sipp.sourceforge.net/doc/reference.html*, 2008.

[65] "NIST/SEMATECH e-Handbook of Statistical Methods," *http://www.itl.nist.gov/div898/handbook/eda/section3/eda3662.htm*, 2008, Section 1.3.6.2.2.

[66] "Apache Tomcat," http://tomcat.apache.org/.

[67] "Open Mobile Alliance," http://www.openmobilealliance.org/.

[68] N. Blum and T. Magedanz, "PTT + IMS = PTM - Toward Community/Presence-based IMS Multimedia Services," in *International Symposium on Multimedia*. IEEE Computer Society, 2005.

[69] "Open Source Initiative," http://www.opensource.org/.

[70] Wikipedia, "Open Source Software," http://en.wikipedia.org/wiki/Open_source_software.

[71] C. Mulligan, "Embracing Open Source Methods for the Standardization of NGN Services and Enablers," *IEEE Workshop on Open Source and Next Generation Networking*, Sep 2008.

[72] R. S. Pressman, *Software Engineering - A Practitioner's Approach*, 5th ed. McGraw-Hill, 2001, ch. 2, pp. 19–51.

[73] "Open Source IMS Core," http://www.openimscore.org.

[74] Free Software Foundation, "GNU General Public License," http://www.gnu.org/copyleft/gpl.html, Jun 2007, version 3.

[75] "Fraunhofer Gesellschaft," http://www.fraunhofer.de/EN/index.jsp.

[76] "UCT IMS Client," http://uctimsclient.berlios.de.

[77] "Ubuntu Linux Operating System," http://www.ubuntu.com/.

[78] "Apache HTTP Server Project," http://httpd.apache.org/.

[79] "The Apache Software Foundation - Meritocracy in Action," http://www.apache.org/.

[80] "Tomcat Wiki - 'Powered By:'," http://wiki.apache.org/tomcat/PoweredBy.

[81] Mobicents, "JBoss Communications Platform."

[82] "Java Advanced Intelligent Network Service Logic Execution Environment," http://www.jainslee.org/.

[83] J. Deruelle, "JAIN-SLEE and Sip-Servlets Interoperability with Mobicents Communication Platform," *NGNOPS'08*, 2008.

[84] "JAIN-SLEE - About JAIN-SLEE," http://www.jainslee.org/slee/slee.html.

[85] J. Deruelle, "My New Life @ Home," http://jeanderuelle.blogspot.com/2008/09/mobicents-sip-servlets-is-jsr-289-sip.html.

Appendix A

Description of the Evaluation Framework

The roles played by the various entities in the evaluation framework are described functionally in Section 4.3. This Appendix does not repeat the descriptions provided, but details relevant practical aspects of the software setup and the tests performed. The functional implementation of the test framework is described first, followed by descriptions of the tests performed.

A.1 Functional Implementation

As outlined in Section 3.2, the full set of IMS network entities can be dispensed with for the RLS evaluation. The entities are replaced by a series of data sources and sinks, which subscribe to the RL at the RLS, manage RLS subscriptions to presentities, and received the aggregated presence information.

A.1.1 System Hardware

The evaluation framework was implemented on a single desktop personal computer with the following characteristics:

Operating System:	Ubuntu 7.10 "Gutsy Gibbon"
Processor:	Intel Celeron 2.8GHz
RAM:	1GB
Network Interface:	127.0.0.1 / loopback

Table A.1: System Resources of the Evaluation Testbed

The loopback interface was used as it allowed multiple network entities to reside on the same physical machine, yet still functionally interact through a complete IP stack. This allowed the

author to use a single PC to perform testing. This reduced the demand on shared laboratory hardware, which was particularly beneficial when considering the long duration of the tests performed.

A.1.2 System Software

A.1.2.1 The Resource List Server

The RLS was developed as a SIP stack-enabled web application, which executes in Mobicents Tomcat [60] web-application container. The Mobicents Tomcat container is a modification of the Apache Tomcat web application container, with support for the SIP Servlet libraries added. This allows the container to support SIP-enabled as well as traditional HTTP-based applications, and for applications to interact with subscribers over two different interfaces. The container abstracts much of the complexity of a multi-interfaced, multi-threaded server from the application developer, and manages such resources transparently.

The RLS reacts and processes received SIP SUBSCRIBE and NOTIFY messages, and aggregates the information according to the server settings. The server parameters are reset by a SIP MESSAGE message containing an XML message of predefined format, allowing new test parameters to be implemented without restarting or deploying different server versions.

The network interface and ports the container listens on are configured through the server.xml configuration file, in the installations' /conf/ directory, while the log files generated through the server operation are stored in the /logs/ directory. The particular logging configurations used for the RLS server are to be found in the installations' /lib/log4j.xml. Further information regarding the installation and configuration of Apache Tomcat and the SIP-enabled Mobicents variant can be found at the respective websites [66], [60].

A.1.2.2 Subscribing / Receiving PUA

The PUA subscribes to its resource list, and then receives the presence information aggregated by the RLS. The role of the PUA in the implementation using SIPp, an open source tool that generates SIP traffic according to a user-defined script. The SIPp script implementing the PUA initiates the subscription by sending a SUBSCRIBE message to the RLS. The RL is included in the payload of the SIP message, rather than actually held and managed by the RLS. The RL received by the RLS is parsed and the presentities contained therein subscribed to, with the resulting presence information aggregated and returned to the PUA entity. While the SIPp script adheres to SIP signalling requirements through these interactions, only the NOTIFY messages received and sent by the RLS are included in the operational logs for later analysis.

A number of different PUA scripts were prepared and used, each subscribing to different numbers of presentities.

A.1.2.3 Presence Servers

The evaluation framework used four separate PS entities, all implemented in SIPp. Each PS was bound to a unique port (from port 5071 to 5074), allowing them to work independently of each other, and reduce the operational load per SIPp execution thread. Each PS received subscriptions from the RLS, which were distributed evenly between the servers, and responded with the requested information.

The initial response from the PSs differed if the tests were simulating initialisation or steady-state operation. If in the initialisation state, a full set of presence information was provided to the RLS almost immediately (some delay was built in to simulate transmission and processing time), while a steady-state subscription encountered a random statistical delay at the RLS before an "update" to the presence information was made. The duration of the delay before the "update" was forwarded determined the presence update period, which was examined as one of the parameters affecting aggregation.

The update period delay is implemented using the SIPp 'pause' function, which is in effect a timer. The function implements delay which can be specified as either being of fixed duration, or statistically distributed. The statistical distributions can be specified with an upper and lower bound for the delay, between which the expires according to the specified distribution parameters. The function offers a normal and lognormal distribution, for which the mean and standard deviation can be set; and exponential distribution for which the mean may be specified, and a uniform distribution, which is the default.

The update period in the implementation uses a uniform distribution, with a lower limit of a second and an upper limit of 2, 5, 7 or 10 minutes. Hence, the timer expiration is not the value of the update period, but a random time between 1s and the specified upper limit, which when spread across a number of subscriptions and presence servers, results in a relatively random data stream for modelling presentity update activity.

A.2 Tests Performed

For each aggregation method examined a set of tests was performed, with a relevant parameter being changed for each test. Details regarding these tests, such as the parameter values, RL sizes and test duration are presented in the following tables.

A.2.1 Size Aggregation Tests

The range of parameter combinations tested in size aggregation is provided in Table A.2. Tests performed are marked with an 'x'. The set of tests presented here is independent of the update period under investigation - each test in the table was repeated, for each of the four update periods examined. Each test was run over a 15 minute period.

Size / RL Size	5 pres.	10 pres.	15 pres.	20 pres.	30 pres.	50 pres.
0.5kB	x	x	x	x	x	x
0.75kB	x	x	x	x	x	x
1kB	x	x	x	x	x	x
1.5kB	x	x	x	x	x	x
2kB	x	x	x	x	x	x
2.5kB	x	x	x	x	x	x
3kB	x	x	x	x	x	x
5kB	x	x	x	x	x	x
7.5kB	x	x	x	x	x	x
10kB	x	x	x	x	x	x

Table A.2: Size Aggregation Parameters and Resource List Size Combinations Tested

A.2.2 Quantity Aggregation Tests

The combinations of aggregation parameter and RL size tested for quantity aggregation are provided in Table A.3. Each test performed is marked with an 'x'. As can be seen, not all the possible tests were performed - parameters of 1 and 3 presentities for aggregation are considered very low, and only performed with small RL sizes, due to the limited aggregation performance. The set of tests presented here is again independent of the update period examined. Each test was performed four times, for each of the update periods examined. Each test was run over a 15 minute period.

Quantity / RL Size	5 pres.	10 pres.	15 pres.	20 pres.	30 pres.	50 pres.
1 pres.	x	x				
3 pres.	x	x				
5 pres.	x	x	x	x	x	x
7 pres.	x	x	x	x	x	x
10 pres.	x	x	x	x	x	x
15 pres.	x	x	x	x	x	x
20 pres.	x	x	x	x	x	x
25 pres.	x	x	x	x	x	x
30 pres.	x	x	x	x	x	x

Table A.3: Quantity Aggregation Parameters and Resource List Size Combinations Tested

A.2.3 Time Aggregation Tests

The duration of the time aggregation tests was not constant, as bigger time parameter values led to significantly fewer data points over a 15 minute period. The larger time value tests were run over 20 and 30 minute-long tests, while the smaller time value tests were run over 15 minutes. The split between test duration is shown in Table A.4.

Test Duration	Time / RL Size	5 pres.	10 pres.	15 pres.	20 pres.	30 pres.	50 pres.
15 min	30s	x	x	x	x	x	x
15 min	1 min	x	x	x	x	x	x
15 min	2 min	x	x	x	x	x	x
20 min	3 min	x	x	x	x	x	x
30 min	7.5 min	x	x	x	x	x	x
30 min	10 min	x	x	x	x	x	x

Table A.4: Time Aggregation Parameters and Resource List Size Combinations Tested, with Test Duration

The combinations of the aggregation parameters and RL sizes tested are also presented, with each test performed marked with an 'x'. The set of tests presented in the table was performed four times, for each of the presence update periods examined.

A.2.4 Time and Size Aggregation

Testing of the hybrid aggregation methods involved four separate parameters, with a range of values from each being tested. The range of each parameter examined is presented, with the time and size parameter values in Table A.2.4. The update time and RL size values examined are presented in Table A.6.

Time / Size	0.75kB	1.5kB	2.5kB	5kB	10kB
1 min	x	x	x	x	x
2 min	x	x	x	x	x
5 min	x	x	x	x	x
7.5 min	x	x	x	x	x

Table A.5: Time and Size Aggregation Parameter Combinations Tested

Each combination of parameters examined in Table was performed for each combination of parameters tested in Table A.6. This resulted in 400 tests, each with a unique set of parameters.

Update Time / RL Size	5 pres.	10 pres.	15 pres.	20 pres.	30 pres.
2 min	x	x	x	x	x
5 min	x	x	x	x	x
7 min	x	x	x	x	x
10 min	x	x	x	x	x

Table A.6: Update Time and RL Size Combinations Tested

Although time aggregation is used in the hybrid methods, the duration of each test was kept at 15 minutes, as it was not expected to be the method primarily performing aggregation.

A.2.5 Time and Quantity Aggregation

As discussed previously, the hybrid aggregation methods require the consideration of four separate parameters during testing. For each combination of parameters presented in Table A.8,

Update Time / RL Size	5 pres.	10 pres.	15 pres.	20 pres.	25 pres.
2 min	x	x	x	x	x
5 min	x	x	x	x	x
7 min	x	x	x	x	x
10 min	x	x	x	x	x

Table A.8: Update Time and RL Size Combinations Tested

all of the parameter combinations in Table A.7 have to be performed. This results again in 400 separate tests being performed.

Time / Quantity	5 pres.	10 pres.	15 pres.	20 pres.	25 pres.
1 min	x	x	x	x	x
2 min	x	x	x	x	x
5 min	x	x	x	x	x
7.5 min	x	x	x	x	x

Table A.7: Time and Quantity Aggregation Parameter Combinations Tested

Appendix B

Introduction to the IMS

This first Appendix provides a very brief overview of the IMS as an easily accessible reference. The interested reader is encouraged to read further using references such as "The 3G IP Multimedia Subsystem (IMS) - Merging the Internet and Cellular Worlds" by Camarillo and Garcia-Martin, and published by John Wiley and Sons [1].

B.1 The IMS

The Internet Protocol (IP) Multimedia Subsystem (IMS) is a service provisioning framework for the delivery of multimedia content to subscribers over packet-switched IP networks. It is a driver for the convergence of fixed and mobile services, because of the abstraction of service delivery away from the bearer technologies. The IMS is being specified by groups such as the Third Generation Partnership Project (3GPP) [4] and Open Mobile Alliance (OMA) [67], which focus on specifying functionality and standardised interfaces, rather than specific nodes, allowing some freedom of implementation [1]. The original intent for the IMS was an evolution of the GSM platform, but has been adopted as a Next-Generation Network architecture by groups such as ETSI-TISPAN and the ITU-T [18].

The IMS allows for the rapid development and deployment of novel services and applications as well as the easy provisioning of required resources, making it an attractive framework for service providers. Services on offer can also be integrated, adding value and a competitive edge to the service bouquet [68]. The inclusion of possible third party services can provide additional revenue for network operators, while leveraging the collective creativity of a large, disparate body of third party application developers (see Appendix C).

B.2 IMS Architecture

Functionality in the IMS is divided into three logical layers: a transport layer, core network layer and application layer. Each layer performs separate functions, with the transport layer

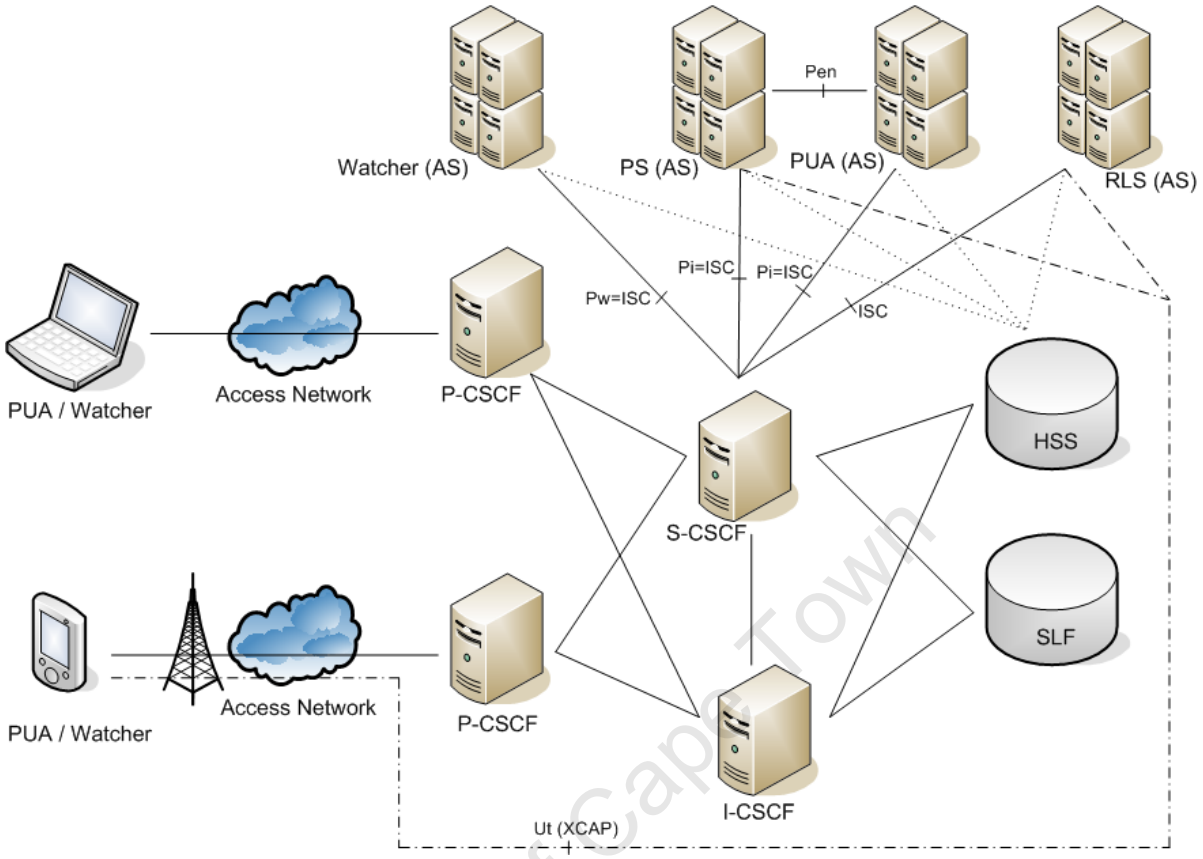

Figure B.1: IMS Architecture, with Presence Service Functions and Interfaces [3]

facilitating access from the IMS terminal (UA) to the IMS core. The IMS core network controls the delivery of services between a UA and the application layer, performing authentication, authorisation and accounting (AAA) tasks, provides gateways and interfaces between legacy and other networks, and manages the available resources. The application layer contains the application servers which provide services to subscribers. The abstraction of the application layer from the core network makes the addition of new services simple, as the introduction of new services and functions is not disruptive.

Figure B.1 shows a schematic of the IMS architecture, with additional presence service functions and interfaces included. The figure, and the details of each function are provided in the reference recommended at the beginning of this Appendix.

Appendix C

Open Source Telecommunications

This appendix examines the impact of Open Source applications and development methodologies on the IMS, and the diversification of service offerings from incumbent network operators.

Open Source Software (OSS) development has defied traditional software business models, relying on the goodwill of a large and disparate developer community. The open source movement can be leveraged by service and application providers to help develop robust, efficient software much faster and cheaper than possible in a corporate environment, with the aid of potentially thousands of developing contributors. Using this model, revenue streams are still available to operators, who can provide services further downstream.

The promise of the IMS to operators is the rapid development, and simple deployment of innovative new services. While a variety of applications suitable for the IMS have been identified, the killer applications over the last few years have tended to come from third-party developers. Encouraging this development, while providing an integrated platform for services and billing, maintains an attractive offering to both third-party developers, and customers.

C.1 Open Source Tools for Development of Telecommunications Applications

The convergence of mobile and fixed-line technologies is driving the adoption of IP networks for telecommunications. The IMS is the leading NGN service-provisioning framework, enabling operators to transition from being access providers with increasingly commoditised networks, to service providers implementing new, innovative and profitable services across their network infrastructure. Part of this transition that should be embraced and utilised is the concept of open source development and standardisation, in order to incorporate the vast potential of the Open Source Software (OSS) movement [69], [70].

C.1.1 Embracing Open Source Software Development Cycles

Mulligan [71], presents an argument for the inclusion of open source methodologies into the standardisation of NGN platforms. The author examines standardisation processes in the converged telecommunications networks that are currently under development, and compares the process in the rigidly structured telecommunications sector to modern software development practises.

As services and applications in NGN networks become increasingly software-based, the design and implementation methodologies need to be updated to match requirements. This is problematic, because standardisation bodies such as the 3GPP [4], Open Mobile Alliance (OMA) [67] and ETSI-TISPAN [5] have traditionally used waterfall-based development models for the standardisation of their hardware. The development of hardware standards is also partly driven by operators pushing for their patented technology to be included, yielding the promise of licensing fees from competitors. However, the waterfall model has been rejected as being the most appropriate for software development. This is because such projects rarely follow the waterfall models' proposed sequential flow, working implementations only appear very late in the project time-span, and detailed product requirements are not always known at the start of the project [72]. This is particularly true when used in the standardisation of service enablers and open APIs. This is because the service enablers and APIs sit higher in the software stack, and are affected more by issues relating to software development in the IT domain, such as changes in specifications and requirements, than hardware.

Open APIs for the NGN are being developed in order to create platform openness, that would attract third party developers, and as such would be a key factor driving the platforms' adoption. Network operators need to create developer communities and interest around their platforms, so as to utilise the community to produce applications on their networks that are innovative and attractive to subscribers. The development of such APIs can therefore reap significant benefits, through the utilisation of open source methodologies, within the standardisation process. By adopting the Internet model of community involvement during the development of these interfaces, standardisation bodies can ensure they deliver APIs that are appropriate for application developers, rather than APIs that satisfy the considerably different perspectives of telecommunication vendors or operators.

Standards previously have focused on providing guidelines for development, rather than rudimentary working code, as in OSS development cycles. This has suited vendors and operators, who dislike contributing to running code that may be used beneficially by their competitors. This approach needs to be reconsidered, however, as the best way to improve the overall product quality is to get a beta product to end-users in the development community as soon as possible, to take advantage of the creativity, experience and programming ability available. Operators investing in OSS product development typically have related or downstream assets, which provide them with a considerable advantage over competitors, and a steady revenue stream [71].

C.1.2 Open Source Tools

Considering the wide range of activities and interests in OSS community, it is no surprise that NGN and IMS technologies have received considerable attention. Some of the tools and applications being developed are primarily for research purposes, while others such as Mobicents JBoss and Tomcat, and SIP Servlets, have bigger project bases, and are supported by numerous industry partners.

C.1.2.1 FOKUS OpenSourceIMS Core

The Open IMS Core [73] is an open source implementation of the Call Session Control Function (P-, S-, and C-CSCF) network entities, and the Home Subscriber Server (HSS), which together comprise the primary elements of the IMS core network as specified by the 3GPP, 3GPP2, ETSI-TISPAN and PacketCable initiative. It is intended solely as an academic and industry research tool, and to provide an IMS-core reference implementation for the testing of IMS technologies. All four of the available components are based on open source software implementations, and are released under the terms of the GNU General Public License [74].

The Open Source IMS Core is released and maintained by FOKUS, a Fraunhofer Society [75] institute for open systems.

C.1.2.2 UCT IMS Client

The University of Cape Town (UCT) IMS Client [76] is an open source, research implementation of an IMS client, designed to be used in conjunction with the FOKUS Open IMS Core. The client emulates IMS signalling in as much detail as possible, serving as a platform upon which research projects are implemented.

Some of the related projects that use the client as a foundation are the UCT Advanced Internet Protocol Television (IPTV) solution, the UCT Back-to-back User Agent (B2BUA) and the UCT IPTV Streaming Server. These are research projects to illustrate proof-of-concept, or to provide implementation testbeds for testing and validation.

The UCT IMS Client is provided as an installation package for the Ubuntu [77] and Debian linux distributions, and has the distinction of being the first open source IMS client available.

C.1.2.3 SIP Servlets

SIP servlets are the SIP equivalent of HTTP servlets - java-based application components, managed by a SIP servlet container, that perform SIP signalling [59]. The servlets handle SIP signalling, while presenting a simplified interface for developers. The servlet engine and container (Section C.1.2.4) manage network listening points, retransmissions, and the automatic populating of SIP headers such as CSeq, Call-ID and others, as required. SIP servlets offer easy

third-party development through open APIs, and simple integration with the Java business logic elements in an application.

SIP servlets differ from HTTP servlets due to differences in the operation of the protocols involved. An HTTP server is typically the end point for HTTP requests, while a SIP server may be a terminating or proxy server in the signalling route, depending on the characteristics of the application, and the state of the SIP dialog. HTTP also does not use a peer-to-peer model for signalling, and as such HTTP servers never initiate HTTP requests, only terminate them. SIP servers may terminate, proxy or initiate new SIP sessions, as required.

C.1.2.4 Web Application Containers

The rapid development and deployment of new and innovative services for an NGN such as the IMS requires suitable tools, support, and development frameworks in order to be successful. It is this ability to remain dynamic and engaging to consumers that will attract and maintain profitability and market share.

Applications deployed on web-servers typically consist of business logic, which implements the functionality of the application, and signalling logic, which enables the application to interact with other network entities in an acceptable and standardised manner. Programmers in the OSS and corporate communities are not necessarily well-versed in the intricacies of SIP and IMS signalling, but can quickly develop robust and efficient business logic suites for applications. Very important aspects in server design, such as multi-threading support, support for multiple interfaces, database connection pooling and resource management, require considerable technical experience and know-how, especially when the application experiences high call rates and resource loads.

Web application containers are server containers that abstract some of the underlying signalling and application management tasks (such as the management of multithreading and database connection pooling). Using web containers allows application developers to focus on developing the business logic of the application, rather than get caught up in network interface and signalling management. The Apache HTTP Server is a web server that played a significant role in the growth of the World Wide Web [78]. The Apache Tomcat web server is another, subsequent Apache Foundation project [79], which implements Java Servlet and JavaServer Page technologies, allowing dynamic Java-based web applications to be deployed, rather than static HTTP websites. Apache Tomcat is widely used for numerous large-scale and mission-critical web applications, by companies such as Wal-Mart and The Weather Channel [80].

Apache Tomcat provides an operating environment for Java-based web applications, but is limited to primarily HTTP interaction. While web applications are driving the provisioning of services on the World Wide Web, the converging telecommunications environment is not catered for. Mobicents [61], a research division of Red Hat, has addressed this requirement by incorporating SIP Servlets into Tomcat, turning Tomcat into a SIP application container. Applications developed with SIP Servlets, running in the Mobicents Tomcat [60] release, now

become capable of operating as SIP application servers. This provides a large part of the framework for the rapid introduction of services into the IMS, by serving as an attractive and readily available deployment platform with support for open-source APIs for the OSS community to use.

Mobicents have also incorporated the SIP Servlets framework into Red Hat's JBoss [81], an open source Java-Enterprise Edition-based application server, as well as the Java Advanced Intelligent Network Service Logic Execution Environment (JAIN-SLEE) [82] application environment [83]. JAIN-SLEE is the Java intelligent NGN standard for SLEE, an application environment offering high throughput and low latency event processing [84]. Mobicents JBoss is the only open source platform currently (Dec 2008) certified for both JAIN-SLEE and SIP Servlet compliance.

SIP Servlets and JAIN-SLEE, while similar, do not serve exactly the same markets. Deruelle [83], presents a proof-of-concept example illustrating the interoperability capabilities of the JAIN-SLEE application environment and SIP Servlets, as technological enablers in Next Generation Network (NGN) converged applications. The example used is developed for Mobicents JBoss platform. Deruelle is a Senior Software Engineer at JBoss [85], and is the Project Lead for Mobicents Sip Servlets, and a member of the SIP Servlets 1.1 (JSR289) Expert Group [59].

The author dissects the argument that the choice between SIP Servlets and JAIN-SLEE for a development framework is either / or decision, concluding that the specifications target different audiences and sets of needs. The two specifications can be used in conjunction with each other, to achieve converged application implementations, rather than in a mutually exclusive manner. This is achievable because the focus of SIP Servlets is kept specifically on SIP, keeping the standard simple, and providing a greatly reduced time-to-market for product development. JAIN-SLEE on the other hand, while powerful and protocol agnostic, introduces much extra complexity and has a steep learning curve for new developers.

The advantages gained from each specification can be utilised through encouraging interoperability. One of Mobicents goals with respect to the JAIN-SLEE and SIP Servlet converged platform, is to leverage the best of both implementations into a unified environment and programming model [83].

Appendix D

Accompanying CD-ROM

The CD-ROM accompanying this document (inside the back cover) contains relevant implementation and other files in electronic format.

The contents are as follows:

- Evaluation Framework
  - SIPp Files
  - Testing Shells
  - Evaluation Framework Software
- Evaluation Results
- Reference Material
  - Referenced Papers
  - RFC Standards
- Resource List Server Development Files
  - Servlet 1.1 API
  - sip-rls (RLS Project Files)
- Thesis Document Files
  - LyX-format Chapter and Appendix Files
- This document in Adobe's Portable Document Format (.pdf)