Presence in the IP multimedia subsystem

Ling Lin and Antonio Liotta[*]
Department of Electronic Systems Engineering, University of Essex, Colchester, UK

Abstract. With an ever increasing penetration of Internet Protocol (IP) technologies, the wireless industry is evolving the mobile core network towards all-IP network. The IP Multimedia Subsystem (IMS) is a standardised Next Generation Network (NGN) architectural framework defined by the 3^{rd} Generation Partnership Project (3GPP) to bridge the gap between circuit-switched and packet-switched networks and consolidate both sides into on single all-IP network for all services. In this paper, we provide an insight into the limitation of the presence service, one of the fundamental building blocks of the IMS. Our prototype-based study is unique of its kind and helps identifying the factors which limit the scalability of the current version of the presence service (3GPP TS 23.141 version 7.2.0 Release 7 [1]), which will in turn dramatically limit the performance of advanced IMS services. We argue that the client-server paradigm behind the current IMS architecture does not suite the requirements of the IMS system, which defies the very purpose of its introduction. We finally elaborate on possible avenues for addressing this problem.

Keywords: Presence service, IMS prototype

1. Introduction

With today's multimedia-capable, multi-purpose mobile devices, end users are demanding rich-media, interactive services which can take greater advantages of the capabilities of their devices anytime and anywhere. The communication industry as a whole is undergoing an evolutionary transformation. The traditional isolated network is giving way to a new service delivery framework which requires industry standards based services across multiple networks.

The IP Multimedia Subsystem (IMS) is a network framework based on the Internet Protocol (IP), which fundamentally allows service providers to deliver services to end users independently of their device type, network type, or physical location. With IMS, service providers are able to address fundamental concerns as they migrate from circuit-switched networks to packet networks. By adopting key technologies from the IT domain, IMS is also well integrated with existing mobile and Internet networks. On longer term, IMS enables a migration to an all-IP architecture. Thus, IMS can provide true mobility for desktop, notebook, handheld and mobile phone with single access privileges.

IMS has evolved from its initial scope set around the Universal Mobile Telephony System (UMTS) – in 3GPP Release 5 – to an access-agnostic, to access agnostic framework, which takes advantages of multiple access technologies, including fixed access (DSL, Ethernet), mobile access (W-CDMA, GSM, GRPS), and wireless access (WLAN, WiMAX) – 3GPP Release 6 and 7. Other non IMS-compatible systems are also supported through gateways. Because of its general applicability outside the wireless access domain, other standards bodies have subsequently adopted the majority of the 3GPP IMS specifications

[*] Corresponding author. University of Essex, ESE Department, Wivenhoe Park, Colchester CO4 3SQ, UK. Tel.: +44 (0)1206 87 4247; Fax: +44 (0)1206 87 2900; E-mail: antonio.liotta@ieee.org.

as the underpinning of their own architectural standards. These forums include the third Generation Partnership Project 2 (3GPP2) under the Multi-Media Domain (MMD) specification, the Open Mobile Alliance (OMA), and European Telecommunications Standards Institute (ETSI) [1].

The aim of IMS is not only provide new services but all the services, current and future, that the internet provides, like it is on the Internet. The profitability appeal of IMS for service providers lies in its ability to provide a unified service-centric network framework, providing an open interface for third party services to be easily integrated and an opportunity for end users to experience feature-rich services. IMS provides a number of fundamental common functions that are generic and can be reused by other services, for example, presence, authentication and authorization, charging and deployment.

The presence service is one of these basic services providing fundamental support to other services and applications. Presence information conveys the willingness and the ability of a user to communicate with others, including a variety of user status modalities. Through the presence service, end users are allowed to subscribe to each other's state and be notified of any changes in real-time. When combined with instant messaging, presence provides a simple and fast way of communication between end users [7].

IMS employs IETF defined text-based Session Initiation Protocol (SIP) as its call control protocol, which is a text-based application layer protocol for creating, modifying, and terminating calls with one or more participants. Text-based protocols have the advantage of being generally more interoperable but less efficient than their bit-wise counterparts [4].

In this article, we provide an insight into the performance of the IMS presence service based on a prototype. Our study indicates that, despite being such an important building block which has already been standardized, the presence service will be a burden in the IMS system. Our prototype-based study is unique of its kind and helps identifying the factors which limit the scalability of the current version of the presence service, which will in turn dramatically limit the performance of advanced IMS services relying on presence. We argue that the client-server paradigm behind the current IMS architecture does not suite the requirements of the IMS, which defies the very purpose of its introduction. We finally elaborate on possible avenues for addressing this problem.

For the benefit of those readers who are not familiar with IMS, Session 2 provides a brief description of its architecture, particularly the presence service. Section 3 explains the IMS prototype design and implementation. Section 4 describes our assessment of the IMS presence service, while in Section 5 we elaborate on how its shortcomings may be addressed. Conclusions are drawn in Section 6.

2. The IP multimedia subsystem

2.1. The IMS layered architecture

The intention of IMS is not to standardise applications within it but rather aid to build services across wired and wireless terminals. This is achieved by having a horizontal layered approach with open and standard interfaces to allow for seamless interconnection of standardized functions. The horizontal layered approach of the IMS allows for lower layers to be transparent to the upper layers, enabling operators and service developers to use different underlying networks. The IMS architecture can be divided in three logical layers, the Access/Transport Layer, the Session Control Layer and the Service/Application Layer. A simplified version of the IMS is shown is Fig. 1.

The Access/Transport Layer provides a common access interface to shield upper layers of IMS. This layer comprises many different type of access networks, such as packet based network, (such as GRPS, UMTS, WLAN, xDSL), and circuit switched network (such as PSTN). End users might connect to the

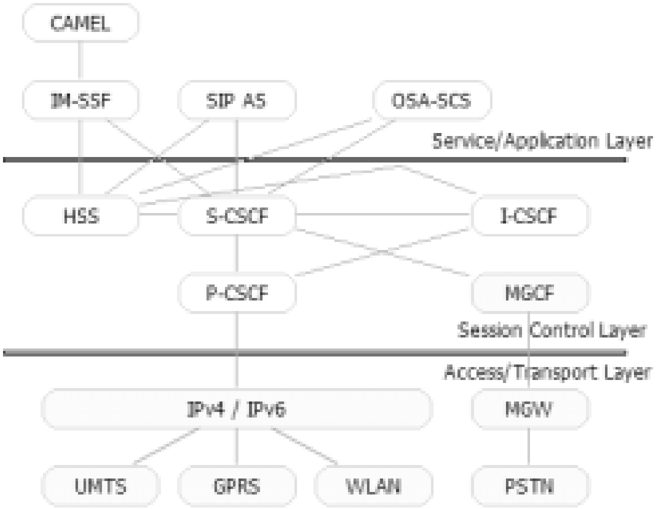


Fig. 1. The IMS Architecture [3].

IMS infrastructure either through a packet based network or via a circuit switched network that interfaces the IMS through the Media Gateway Control Function (MGCF).

The Session Control Layer handles session setup and teardown, and manages the handover of sessions between service providers. There are three types of Call Session Control Functions (CSCF), which are the most important elements in this layer. The Proxy Call Session Control Function (P-CSCF) is a SIP proxy server that provides the first contact point for IMS terminals. A SIP proxy server is an element which routes SIP requests and responses message from and to User Equipment (UE), which could be seen as application layer router [2].

The Serving Call Session Control Function (S-CSCF) is a central node for SIP signalling. It is the S-CSCF that plays the roles of SIP proxy server and SIP Register server, routing SIP messages to other UE or appropriate application servers and registering user information with the Home Subscriber Server (HSS). The SIP Register Server accepts REGISTER requests and stores the binding between Address of Record and Contact URI.

The Interrogating Call Session Control Function (I-CSCF) is also a SIP proxy server used as a gateway to the home network from the visited network, hiding network capacity and topology to the outside. I-CSCF queries the HSS to obtain the address of the appropriate S-CSCF where the request must be forwarded, then assigns the S-CSCF that will handle the SIP request.

The Home Subscriber Server (HSS) is a central database which maintains the unique service profiles for end users. A service profile stores all user service information and preferences, including end user's current registration information (such as IP address), roaming information, instant message service information (such as buddies list), voice mail box (such as greeting), etc.

The Service/Application Layer offers a platform for value-added services which go beyond integration of network and devices. It consists of different application servers for the execution of various IMS services and the provision of end user service logic. Application servers can operate in SIP Proxy Mode, SIP User Agent mode or SIP Back-to-Back User Agent mode.

There are three different types of application servers defined by IMS, SIP Application Server (SIP AS), Open Service Access – Gateway (OSA-GW), and IP Multimedia Service Switching Function (IM-SSF).

The Sip AS is a pure native SIP Application Server designed for IMS. The IM-SSF provides the ability to support traditional Customized Applications for Mobile Network Enhanced Logic (CAMEL) servers. To provide a simple API for IT background companies that are not familiar with the variety of complex telephony signalling protocols, the Parlay Forum, working closely with the 3GPP and ETSI standards development organization, have jointly defined a Parlay API for telephony network. The cooperation between SIP and the Parlay API is provided in the OSA-GW, which allows Parlay applications to manipulate IMS services [3].

2.2. The IMS presence service

As specified by 3GPP TS 23.141 version 7.2.0 Release 7 [1], the presence service uses the SIP Event Notification model as a basis for the presence model. The presence Service is composed of two different abstract roles, the provider of the presence information, or "presentity", and the consumer of the presence information, or "watcher". The watcher can be either a subscriber of presence information or a fetcher of presence information.

Figure 2 illustrates the UML sequence diagram of three typical operations: the subscription to presence information; the publication of presence information; and the notification of new presence information. The entities included in Fig. 2 are described below.

- Watcher (UE-A): A watcher is an entity which is interested in the presence information of SIP presentities. It sends a SUBSCRIBE request to the SIP Presence Agent, and then waits for the NOTIFY message from the Presence Agent.
- Watcher Presence Proxy (P-CSCF and S-CSCF): The main function of the watcher presence proxy is to route SIP messages to and from the watcher. It is a combination of P-CSCF and S-CSCF in the watcher's network.
- Presentity Presence Proxy (I-CSCF, S-CSCF and HSS): The presentity presence proxy first determines the presence server associated with the presentity and then routes the SIP message to it. The presentity presence proxy is a combination of I-CSCF, S-CSCF and HSS in IMS.
- Presence Server: known as presence agent, is a SIP user agent capable of receiving SUBSCRIBE requests, responding to them, and generating NOTIFY message when the presence state changes.
- Presence User Agent (UE-B): manipulates presence information for a SIP presence entity (called presentity); it decides when to report presence changes to the Presence Agent.

Both publication and subscription are valid for a period negotiated in the SIP message. In order to stay active, presentities have to periodically publish or subscribe to presence information. This behavior periodically generates signaling traffic on the radio interface that is the most critical one in the IMS architecture. Figure 2 highlights a crucial shortcoming of the IMS presence service, i.e. its critical use of continuous client-server signaling. The prototype-based study described below assesses this issue in greater detail.

3. Prototyping the IMS

3.1. The current status of IMS

The deployment of IMS will have immense impact on the whole future network. Nearly all network equipment providers, system integrators and network operators consider IMS as the most important

Table 1
Mapping between IMS nodes and SIP components

	Proxy	Register	Routing table	Presence	XCAP
P-CSCF	Yes		Yes		
I-CSCF	Yes		Yes		
S-CSCF	Yes	Yes	Yes		
Presence Server	Yes		Yes	Yes	Yes

NGN reference service delivery platform for aligning their product and service roadmaps, although standard-compliant IMS products are not yet fully available. This is probably due to the newness of IMS and the complexity of its protocol stack, which is still evolving and changing. IT hardware and software companies are facing a steep learning curve. Also, the heterogeneity of end terminals will present daunting obstacles to test IMS applications.

IMS has not yet been proved to be a reliable network framework, especially under heavy traffic conditions. The interoperability of IMS could be another problem when it comes to troubleshooting, handshaking, and guaranteeing quality of service. Our IMS prototype aims at identifying the factors which limit the scalability of the current version of IMS.

3.2. Mapping IMS servers onto SIP components

The 3GPP Core Network can be logically divided into three different domains, Circuit Switched Domain, Packet Switched Domain, and IP Multimedia Domain [2]. It is in the IP Multimedia Domain that SIP is used between user equipment terminals and networks, and between network nodes.

Figure 3 illustrates the UML deployment diagram of our IMS Prototype, including four types of servers, P-CSCF, I-CSCF, S-CSCF, and presence server. As seen in Section 2, these can be mapped onto four different SIP components: proxy, registrar, presence and XCAP component, labelled as ‘artefact’. Our prototype follows the mapping of Table 1.

The mapping between IMS servers and SIP components facilitates our design by component reuse.

Instead of implementing the four IMS servers, we only need to complete the SIP components, which can then be configured to obtain the various servers. This allows combining more functions in one node, or splitting a single function over two or more nodes. Each node can also be present multiple times in a single network, for load balancing or organizational issues. As an example, Fig. 4 shows a configuration table for the presence server.

This modular design also offers maximum flexibility to enable the different combination of the IMS servers over one or more machines. This is because, in the IMS, the interfaces between IMS nodes are standardized while IMS nodes are not, allowing vendors to combine as many functions as they wish into a particular node [3].

3.3. System design

Figure 5 shows the UML package diagram of our IMS prototype. Most of the packages follow the façade design pattern, which makes modules and classes manageable. JAIN SIP provides a standardized network interface, which makes the system network independent. Upon receiving a SIP messages, JAIN SIP invokes its listener, proxy component to process the message.

This proxy component implements a stateless proxy, as described in RFC 3261 [4], which is a listening component that routes SIP requests to user agent servers and SIP responses to user agent clients. Following successful authentication, the proxy makes its routing decisions based on the routing

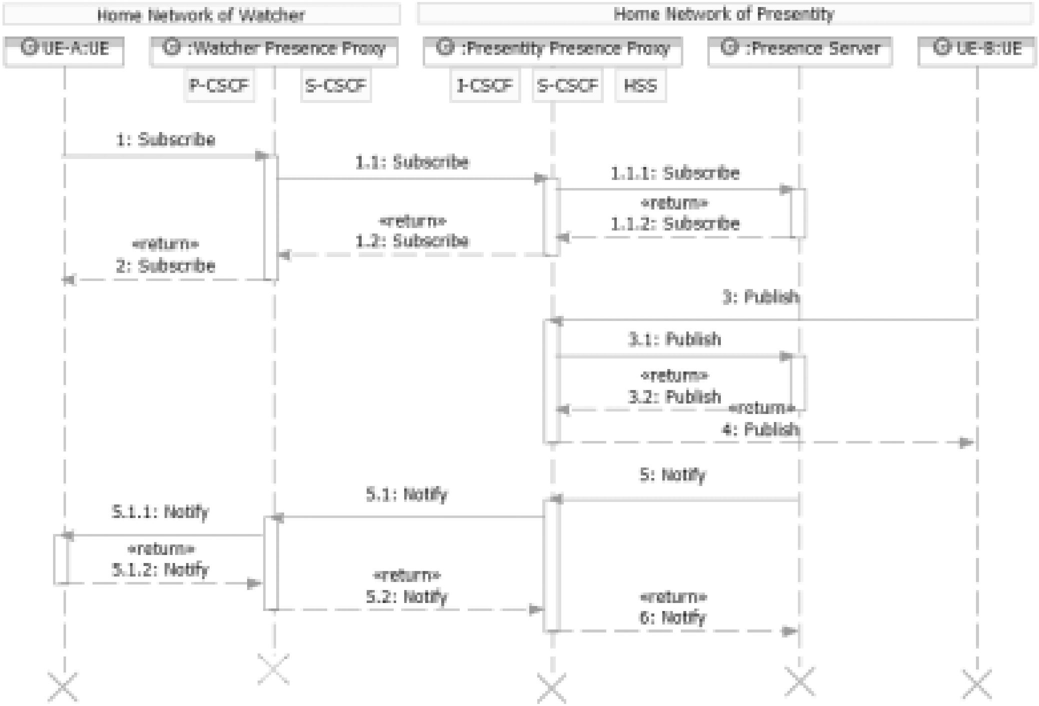


Fig. 2. IMS presence service – sequence diagram [3].

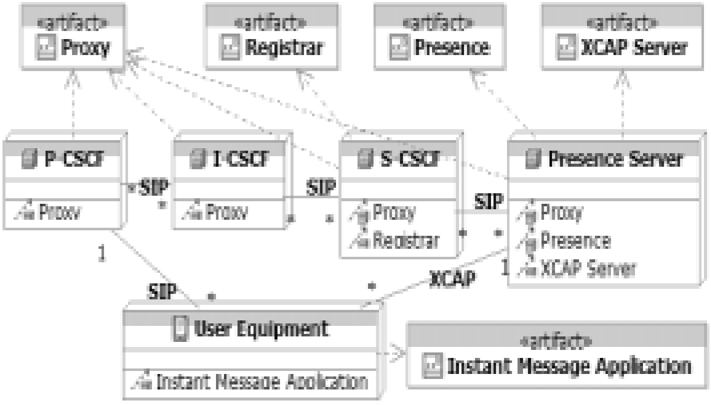
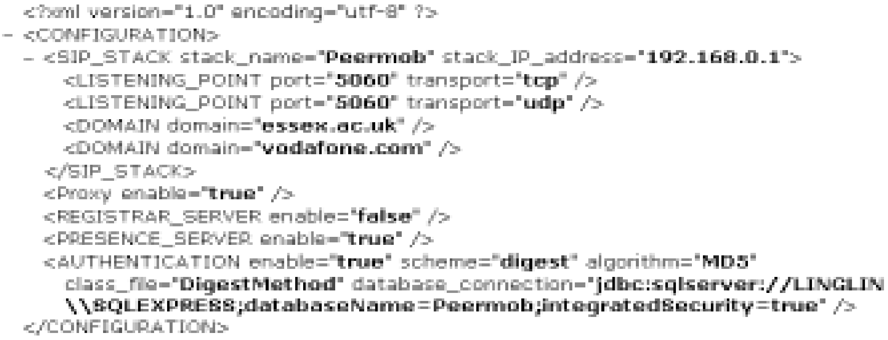


Fig. 3. System deployment diagram.

package. If the message is not within the proxy's domain, the message is modified before being forwarded to the next proxy. A request may traverse several proxies before reaching its destination. The relevant response will be routed through the same set of proxies but in the reverse order. If the message is within the proxy's domain, the proxy will transfer the SIP message to the relevant SIP component to handle it.

The registrar component is an implementation of SIP registrar as defined in RFC 3261 [4], which handles REGISTER requests sent by user agents to register a binding from a user's public address to



```
  <?xml version="1.0" encoding="utf-8" ?>
- <CONFIGURATION>
  - <SIP_STACK stack_name="Peermob" stack_IP_address="192.168.0.1">
      <LISTENING_POINT port="5060" transport="tcp" />
      <LISTENING_POINT port="5060" transport="udp" />
      <DOMAIN domain="essex.ac.uk" />
      <DOMAIN domain="vodafone.com" />
    </SIP_STACK>
    <Proxy enable="true" />
    <REGISTRAR_SERVER enable="false" />
    <PRESENCE_SERVER enable="true" />
    <AUTHENTICATION enable="true" scheme="digest" algorithm="MD5"
      class_file="DigestMethod" database_connection="jdbc:sqlserver://LINGLIN
      \\SQLEXPRESS;databaseName=Peermob;integratedSecurity=true" />
  </CONFIGURATION>
```

Fig. 4. Example configuration of presence service.

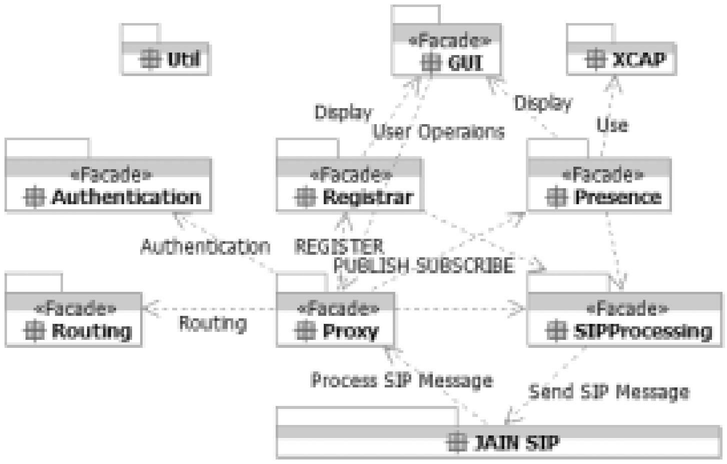


Fig. 5. System package diagram.

physical network address of their user. A SIP registrar is a listener responding to REGISTER messages and maintains users' related information stored in HSS. Register requests are processed atomically, meaning that a particular Register request is either processed completely or not at all.

Figure 6 depicts the high-level UML class Diagram of the Presence Component, which can accept PUBLISH and SUBSCRIBE messages. When receiving a PUBLISH message from the proxy component, the presence component verifies the identity of the source of the PUBLISH request and performs authorization. Then it updates the corresponding presence information of the user agent and sends a 200 OK message back, by invoking the SIP processing package, which is typical for constructing and sending SIP messages.

The XML Configuration Access Protocol (XCAP) component is an implementation of the XCAP protocol, allowing a client to read, write and modify application configuration data stored in XML format on a server. XCAP maps sub-trees, elements, and attributes of XML documents to HTTP URLs, so that they can be directly accessed by HTTP. Reading an XCAP resource is accomplished via the HTTP GET method. Creating or modifying one entry is done via the HTTP PUT method. Removing one of the resources is done via the HTTP DELETE method [4].

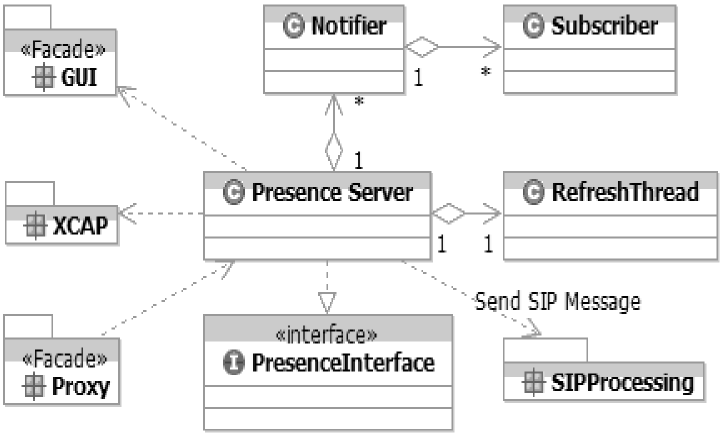


Fig. 6. Presence Component high level class diagram.

3.4. System implementation

In order to demonstrate and evaluate the performance of the IMS Presence Service, we have developed a Java-based prototype, as described in the previous section. We have adopted SUN JAIN SIP 1.2, specified by JAIN Protocol Expert Group Java Community Process Participants for SIP. JAIN SIP is a low-level Java API for the SIP stack and its extensions enable voice over IP gateways, user terminals, and other communication systems to interface with each other using SIP. The JAIN SIP API provides an interface into proprietary SIP protocol stacks.

National Institute of Standards and Technology (NIST) SIP [6] is an open source SIP stack reference implementation for JAIN SIP. This project has implemented a number of libraries and tools including JAIN SIP API and JAIN SDP API. NIST SIP 1.2 has been adopted to be the implementation for JAIN SIP 1.2 API. NIST-SIP can be replaced by any alternative reference implementation based on the JAIN SIP 1.1 API.

The IMS client application is based on SIP for J2ME API, which defines a Java SIP interface for mobile platforms. It enables SIP applications to be executed in memory-limited terminals such as mobile phones. The SIP for J2ME specification is based on the Connected Limited Device Configuration (CLDC) 1.1 and the Mobile Information Device Profile (MIDP) 2.0, supported by most Java phones. The NIST SIPLITE project has been chosen to be the implementation for SIP for J2ME.

3.5. System in operation

Figure 7 presents the sequence of interactions of IMS clients and the messages exchanged between SIP entities. Figures 8 to 11 illustrate key steps (for login to notification) through the application GUI running on HP IPAQ. Presence and proxy server are installed in ordinary desktops. Some peers run on HP IPAQ while others are executed on J2ME terminal emulators. The access network for all entities is an ordinary 802.11b LAN.

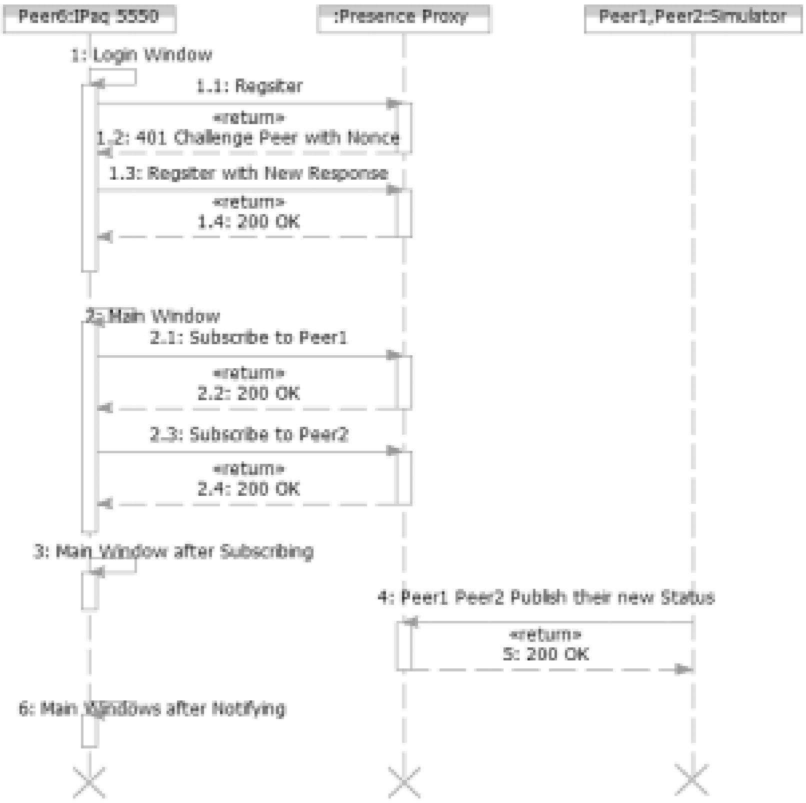


Fig. 7. System in operation sequence.

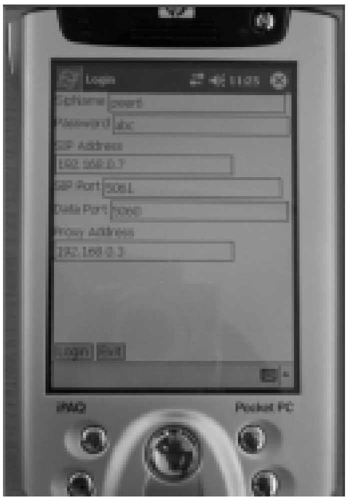
4. Assessment

The purpose of this assessment is to evaluate the performance of the IMS presence server. We measure response time, packet loss and CPU and memory utilization of the presence server running in one desktop. Response time and packet loss are recorded in the SIP client. Response time is measured as the delay between sending a PUBLISH message and receiving the corresponding NOTIFY message. Packet loss refers to the percentage of NOTIFY messages that should have been received but are lost, indicating that the presence server is overload. CPU and memory utilization are sampled in the presence server.

The test environment, as shown in Fig. 12, incorporates two machines. The presence server runs on an x86 based desktop PC, AMD Athlon 64 3400+ with 1 GB memory. The Presence and proxy components have been enabled and so the system is setup as a presence server. A new SIP traffic generator application is implemented on Jain SIP, which is capable of generating and receiving SIP messages. The traffic generator runs on an x86 based laptop with Intel 1.6 G CPU and 1 GB memory. Both machines run Windows XP SP2 with JDK 1.5 and connect to a switch through wired 100BaseT Ethernet. UDP has been chosen as underlying transport protocol.

The experiment follows the steps below:

1. Start the presence server.
2. Start SIP client and send a PUBLISH message to the presence server to login in as "SIP:llini@essex.ac.uk". Then, send a SUBSCRIBE message to subscribe the terminal's own



SIPname owen
Password abc
SIP Address
192.168.0.7
SIP Port 5061
Data Port 5060
Proxy Address
192.168.0.3

Fig. 8. Login window.

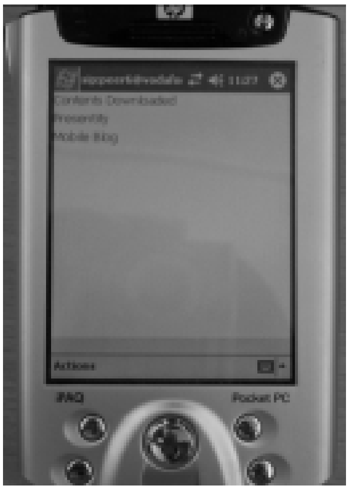


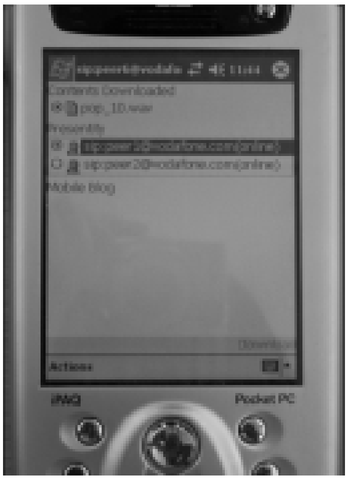
Fig. 9. Main window.

status “SIP:llini@essex.ac.uk”. This means that when the SIP client changes status he will receive a notification message.

3. Send PUBLISH messages to the presence server at different rates and different quantities. Measure the average response time and the percentage of packet received.
4. Measure CPU usage and memory usage in the presence server.

The first experiment was setup to measure the performance of the presence server in a given period. To achieve that, the SIP client sends 1000 PUBLISH messages to the presence server at different rates, ranging from 1 packet per second to 150 packets per second. The result is illustrated in Figs 13 and 14.

We can observe that the average response time remains at about 50 millisecond and there is no packet loss when the PUBLISH rate is below 30 packets per second. For higher rates, response time increases dramatically. In the meantime, there is a 10% to 20% packets loss. This indicates that the presence



sippeer6@vodafo 11:44

Contents Downloaded

⊗ pop_10.wav

Presentity

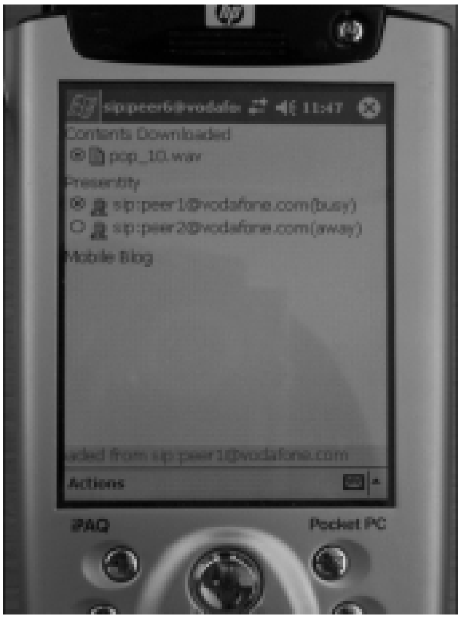
⊗ sip:peer1@vodafone.com(online)

○ sip:peer2@vodafone.com(online)

Mobile Blog

Actions

Fig. 10. Main window after subscribing.



sippeer6@vodafo 11:47

Contents Downloaded

⊗ pop_10.wav

Presentity

⊗ sip:peer1@vodafone.com(busy)

○ sip:peer2@vodafone.com(away)

Mobile Blog

aded from sip:peer1@vodafone.com

Actions

Fig. 11. Main window after notifying.

server works fine at lower PUBLISH rate. When the PUBLISH rate reaches 40 packets/sec, the presence server becomes overloaded, skipping NOTIFY messages. So, the presence server can only handle about 30 SIP packets per sec.

The purpose of the second experiment was to evaluate the performance of the presence server under a load of bulks of requests, which emulates several users who simultaneously issue requests. The SIP client sends bulks of PUBLISH packets at fixed speed, 100 packets per second. So, the presence server receives a bulk of packets virtually at the same time. Figures 15 and 16 illustrate the resulting response time and received NOTIFY packets.

The presence server behaves well when receiving a bulk of packet which is less than 100 packets.

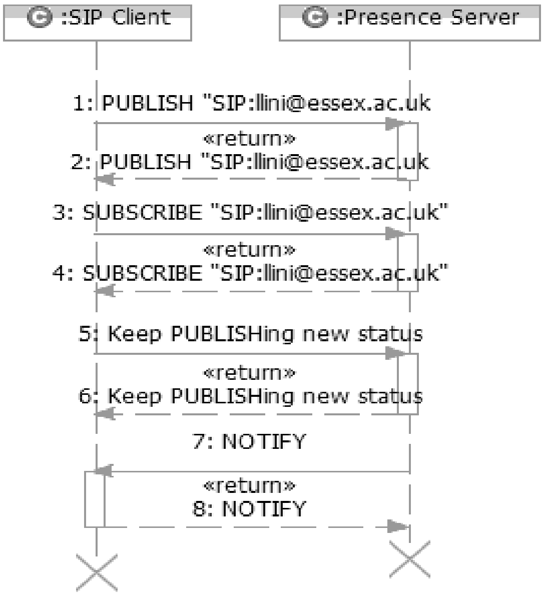

:SIP Client
:Presence Server
1: PUBLISH "SIP:llini@essex.ac.uk
«return»
2: PUBLISH "SIP:llini@essex.ac.uk
3: SUBSCRIBE "SIP:llini@essex.ac.uk"
«return»
4: SUBSCRIBE "SIP:llini@essex.ac.uk"
5: Keep PUBLISHing new status
«return»
6: Keep PUBLISHing new status
7: NOTIFY
«return»
8: NOTIFY

Fig. 12. Sequence diagram.

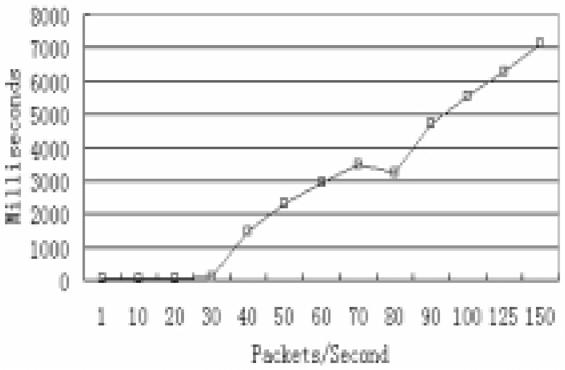


Fig. 13. Experiment 1: Response time.

However, the average response time rises significantly with the increase of bulk size – there is a 5%–10% packets loss at same time. This experiment indicates that the presence server is able to handle up to 100 packages but is overloaded for larger packages.

The third experiment gives a snapshot of CPU and memory usage of the presence server at no request, normal load and heavy load, as illustrated in Figs 17 and 18. The SIP client keeps sending PUBLISH at 0, 10 and 150 packets per second. In Fig. 17 the CPU usage remains at 5% at no request and it goes up to 10% when the client requests at 10 packets per second. However, when the request rate is 150 packets per second, all of the CPU power seems to be consumed. In Fig. 18, the memory usage is almost constant when there is no request or 10 packets per second. However, when the PUBLISH rate goes to 150 packets per second, the memory usage varies from 38 MB to 39 MB. This is because the system is so busy that the Java virtual machine can not release the allocated memory on time.

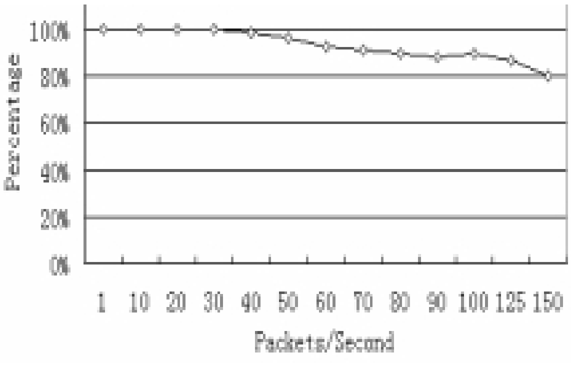


Fig. 14. Experiment 1: Percentage of received packet.

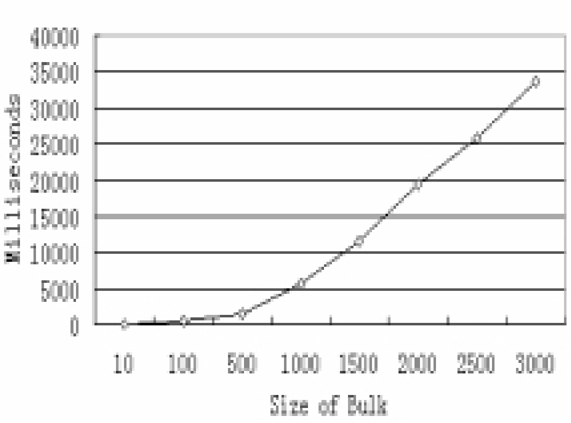


Fig. 15. Experiment 2: Response time.

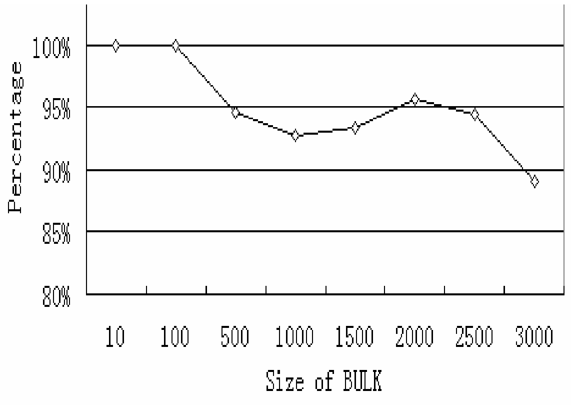


Fig. 16. Experiment 2: Percentage of received packets.

5. Discussion

The assessment described in Section 4 shows that the presence service, as specified in the latest IMS standards, does not meet the important requirement of scalability. Bottlenecks arise at three levels. First,

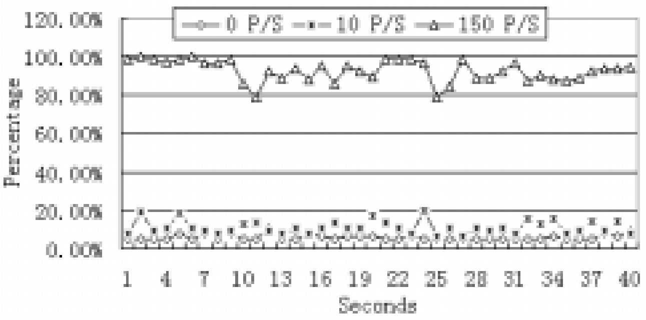


Fig. 17. Experiment 3: CPU usage.

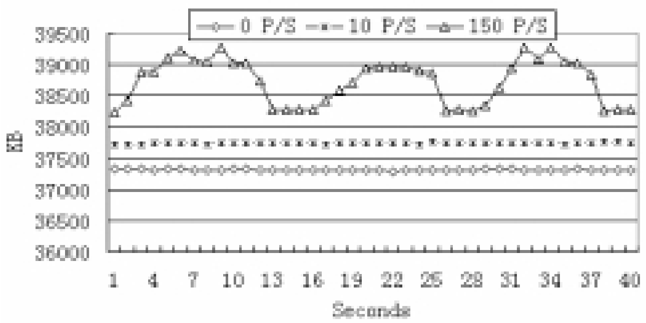


Fig. 18. Experiment 3: Memory usage.

the presence server is a bottleneck. Figures 17 and 18 show that CPU overloading is the predominant factor causing packet loss. SIP processing saturates the CPU well before exhausting memory. This also suggests that CPU power is the first factor to consider when running Java SIP phones on PDAs. Second, the network around the presence server is another inevitable bottleneck – SIP messaging increase dramatically with number of users, amount of presentities subscribed to, and frequency of presence updates.

A possible solution to this issue is to introduce Peer-to-Peer (P2P) technologies in the presence service, integrating P2P protocols onto the IMS system. P2P systems inherently have higher scalability, robustness and fault tolerance because there is no centralized server. A P2P-based presence service can be viewed as an application where the participants form a self-organizing P2P overlay network to locate and communicate with other participants.

Our next priority is to extend the IMS prototype developed so far, including a P2P-based presence service – Fig. 19 shows its UML package diagram. The necessary protocols are currently the subject of intensive studies under the banner of P2P SIP (http://www.p2psip.org) and are expected to reach a more stable status soon. These will also have to go through a standardization process before their full employment.

The P2P network layer of Fig. 19 may be realized via CHORD [8], which realizes typical P2P functionality (e.g. distributed publish and search) and can be used to isolate the presence service from the actual P2P network. The P2P SIP layer may act as a bridge between the P2P network and the IMS domain, translating P2P messaging into SIP.

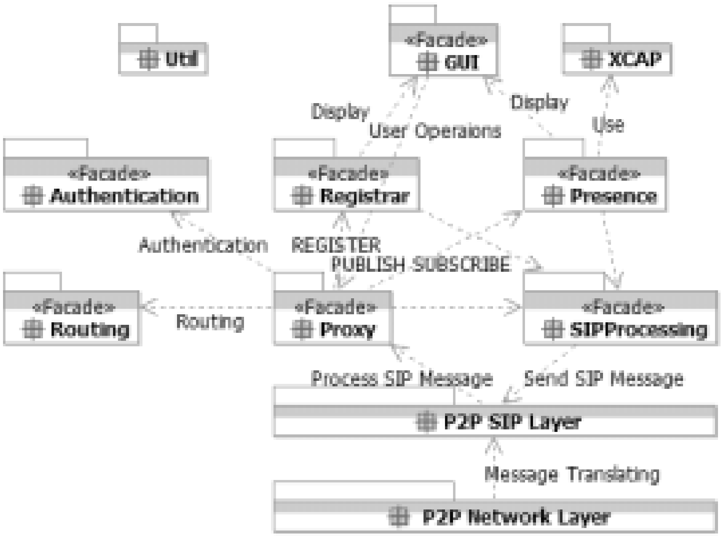


Fig. 19. P2P presence service package diagram.

6. Conclusions

This paper introduces a unique approach to implementing an IMS system prototype based on JAIN SIP. We have prototyped an instance messaging application running on a personal digital assistant, the HP IPAQ 5550, proving the viability of realizing an IMS client application on a relatively thin mobile terminal. While this demonstrates that the IMS architecture is suited to the deployment of mobile services, and this was the original motivation behind the specification of the IMS, our work aims at quantifying the scalability of key IMS components.

Herein we focus on the assessment of the IMS presence service, which is a fundamental building block for most IMS services. We find that the presence service, as currently specified by 3GPP, does not show the sort of scalability that one would expect in the context of the IMS. The IMS has been introduced to deploy ubiquitous services over converged, all-IP networks. By contrast, our experiments unveil a substantial processing bottleneck on the presence server (due to the processing of SIP messages) – 30 messages per second lead to server overload.

We conclude that the signalling overheads incurred by our IMS-compliant presence service will be the limiting factor of IMS services and applications that make use of presence information, which is a large majority. These will be limited in terms of number of concurrent users, response time (presence refresh rate), and number of presentities etc, defying the very purpose for introducing a standard (inter-operable) presence service for the ubiquitous user.

The study carried out so far, confirms our initial hypothesis, i.e. the need to look at new ways for managing presence information in a large-scale telecommunication system, beyond the means available today. Current efforts in the area of P2P systems, particularly P2P SIP are particularly promising in this case, since these have the potential to allow the deployment of distributed approaches to service management.

The present paper complements our earlier work [5], where we look at mechanisms to introduce P2P applications on top of the IMS, with particular attention to P2P publish and discovery of resources. Our next stage will be to develop a P2P presence service for the IMS and test its effectiveness and scalability in supporting P2P applications.

Acknowledgements

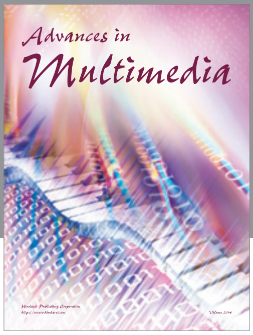
This work was initiated in the context of the PeerMob project, which was funded by Vodafone Group Plc. The work is now being carried out in the context of IST-034115 PHOSPHORUS (Lambda User Controlled Infrastructure for European Research) – www.ist-phosphorus.eu.

References

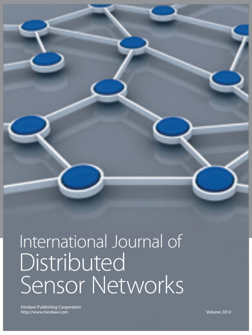
[1] 3GPP, http://www.3gpp.org/.
[2] Alan B. Johnston, SIP: Understanding the Session Initiation Protocol, Second Edition, Artech House © 2004.
[3] Gonzalo Camarillo, M.-A. G.-M., *The 3G IP Multimedia Subsystem* (*IMS*)*: Merging the Internet and the Cellular Worlds*, John Wiley & Sons (August 4, 2004) ISBN: 0470871563.
[4] IETF, http://www.ietf.org.
[5] A. Liotta et al, *Service-driven Group Management for Mobile P2P Services*, Proc. of the IFIP IntellComm'05, Montreal, Canada, KLUWER Academic Publishers, October 17–19, 2005.
[6] NIST, *NIST-SIP 1.2 – SIP Libraries and Tools for the People*, http://www-x.antd.nist.gov/proj/iptel/.
[7] Miikka Poikselka, Georg Mayer and Hisham Khartabil, *The IMS: IP Multimedia Concepts and Services in the Mobile Domain*, John Wiley and Sons Ltd (2004).
[8] I. Stoica et al., Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications, *IEEE/ACM Transactions on Networking* **11**(1) (Feb. 2003), 17–32.

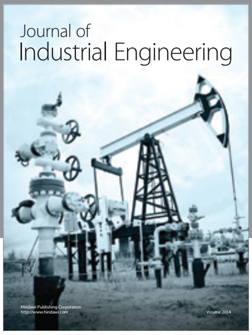
Ling Lin is a PHD student in computer networks and service managementin at the University of Essex, sponsored by Vodafone Group. He received his MSc in the university of surrey with distinction. He is fluent in both system design and system implementation. His main research domain is P2P secure IMS.

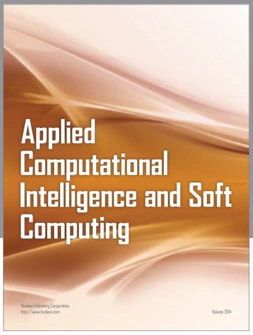
Dr. Antonio Liotta is a Senior Lecturer in computer networks and service management. He has over 70 papers and 2 patents to his credit and is known for his pioneering work on mobile peer-to-peer networks and systems. Antonio is a Registered Practitioner of the U.K. Higher Education Academy; a member of the Peer Review College of EPSRC; and a Member of the Board of Editors of JNSM.

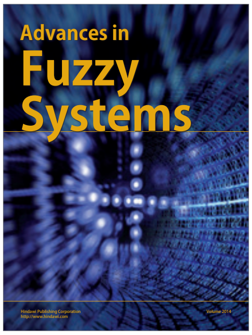

Advances in
Multimedia

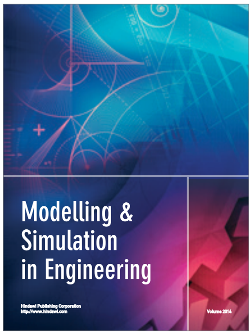
The Scientific
World Journal

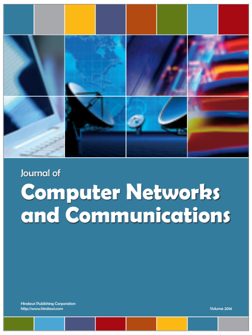

International Journal of
Distributed
Sensor Networks

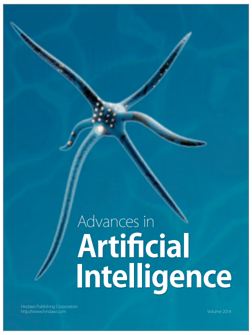

Journal of
Industrial Engineering

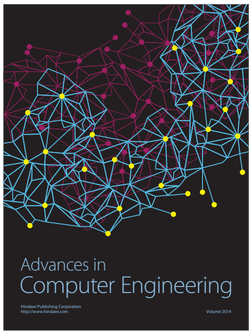

Applied
Computational
Intelligence and Soft
Computing


Advances in
Fuzzy
Systems

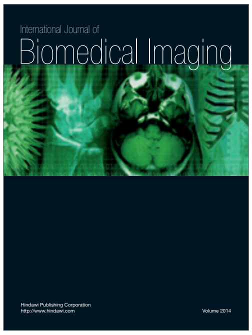

Modelling &
Simulation
in Engineering

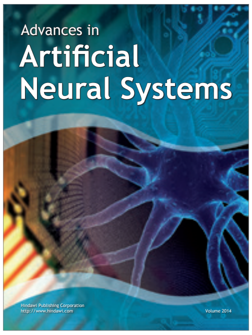

Journal of
Computer Networks
and Communications

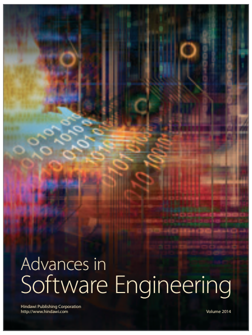

Advances in
Artificial
Intelligence


Advances in
Computer Engineering

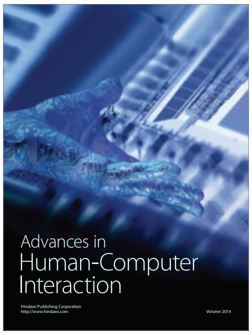
International Journal of
Computer Games
Technology

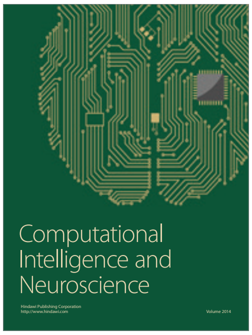

International Journal of
Biomedical Imaging


Advances in
Artificial
Neural Systems


Advances in
Software Engineering

Journal of
Robotics


Advances in
Human-Computer
Interaction


Computational
Intelligence and
Neuroscience

International Journal of
Reconfigurable
Computing

Journal of
Electrical and Computer
Engineering