Internet protocol-based push to talk

Hlabishi I. Kobo, William D. Tucker and Michael J. Norman
Department of Computer Science
University of the Western Cape, Private Bag X17, Bellville 7535 South Africa
Tel: +27 21 9592461, Fax: +27 21 9593006
E-mail: {**2654952**, btucker, mnorman}@uwc.ac.za

Abstract- This paper discusses a way of offering voice instant messaging based on Internet Protocol using Session Initiation Protocol. The purpose of this investigation is to enhance the modern social communication amongst the people of South Africa who are already accustomed to text-based instant messaging. The proposed application aims to implement the traditional Push-to-Talk technology using Internet Protocol. Thus the proposed IP-based Push-to-Talk is a new approach to voice communication which emulates a walkie-talkie system. On the mobile phone IP-Push-to-Talk herein referred to as Push-to-Talk over a cell phone can be viewed as a voice SMS. The adoption of a Push-to-Talk service was inspired by the fact that it applies half-duplex communication. This enhances the primary objective of offering a cheap voice instant messaging. In half-duplex communication, only one person can talk at a time, thereby avoiding bidirectional charging. The project was implemented on two platforms, a PC and a mobile phone. The PC Push-to-Talk was implemented through client server approach whilst the mobile Push-to-Talk through a peer-to-peer approach. Several software engineering strategies were used for user requirements gathering as well for testing. Six users participated in the test and the results were gathered through questionnaires. The results showed that, half-duplex communication is efficient and yet very economical as it makes less usage of system resources.

Index Terms— Network services, Protocols, Push-to-Talk, Session Initiation Protocol, Real-Time Transport Protocol

I. INTRODUCTION

Internet Protocol-based push-to-talk (IP-PTT) is a new instant messaging (IM) type of communication that is voice-based rather than text-based like many popular IM services. IP-PTT uses the PTT concept of radio systems that have been in existence for years, similar to a walkie-talkie or citizen's band (CB) radio. IP-PTT makes use of Internet protocol over a variety of networks instead of analog radio frequencies. IP-PTT can operate using standards of Voice over Internet Protocol (VoIP). The communication model for IP-PTT is half-duplex whereby only one person can talk at a time [1] [2] [3], following on the walkie-talkie and CB radio models. This can allow IP-PTT more efficiency than real-time VoIP since only one person can talk at a time eliminating interruptions [www.mobilein.com]. Transmission of voice occurs through a push or press of a dedicated PTT button. A PTT session is initiated when button is pressed and terminated when released. The call set-up and termination can easily be controlled by the Session Initiation Protocol (SIP), a protocol more often used for real-time VoIP. The transmission of IP-PTT voice data packets occurs as normal. Because of IP's ubiquitous nature, IP-PTT can eliminate the geographical limitations imposed on traditional PTT systems.

Many South Africans have joined the community of mobile instant messaging through MXit [4]. MXit is free mobile text-based IM system developed locally in South Africa. Since MXit is both mobile and text-based, a voice-based IP-PTT similar in functionality to MXit would no doubt increase use of mobile data. Modern communication technology worldwide is converging to a mobile environment, and South Africans are much more likely to have a cell phone than a computer [http://mybroadband.co.za/news/cellular/11723.html].

Mobile IP-PTT is often referred to as PTT over a cell phone (PoC). PoC can support 2.5 and 3rd generations of mobile data networks as well as the next generation network [3] [www.mobilein.com]. The concept of PoC was introduced in 2003 and standardization started in 2004 with the first version finalized in 2005 [www.mobilein.com] [6]. The Open Mobile Alliance (OMA) is the official standards body that oversees all of the infrastructures and processes supporting PoC. In June 2006, the first version OMA PoC 1.0 was released. PoC is available and popular on many cellular networks around the world, but unfortunately not in South Africa. We find this situation curious since PoC seems a natural way to increase data usage, and therefore, revenue generation, in South Africa. In the USA, Nextel communications and Motorola network providers offer the service, and both Orange and Vodafone offer PoC in the UK. The Nextel PTT service is called Direct Connect [1]. As shown in [12], PoC can even be implemented using Bluetooth. Thus there are many options for building PoC systems, some for revenue and some for recreation.

This paper presents exploratory work to design, build and evaluate a purely IP-based PTT system, first on a PC and then on a mobile phone. Two methods of implementation were explored: a) a client-server approach was employed on the desktop PC application where the server was used to relay the real-time packets between two IP-PTT clients, and b) a peer-to-peer (P2P) approach was employed for the mobile phone PoC prototype. The latter approach does not involve any middle layer as it transmits directly from one endpoint to the other.

The rest of this paper is organized as follows; Section II comprises the bulk of the paper and presents the methods for the design and evaluation of the prototypes. Section III discusses the findings, and Section IV highlights conclusions and avenues for future work.

II. METHODS

This section presents the methods used to define and analyse user requirements, the design of the prototypes and the protocols used to implement them, and finally the procedure for testing the prototypes and the results obtained from those tests.

Standard software engineering (SE) approaches were used throughout the development process. A generic SE life cycle was adopted and used iteratively as in the incremental model (see Fig. 1). User requirements are always a determinant factor in software development. User requirements data were gathered through structured questionnaires and interviews with students in our university. Fig. 1 shows the overall iteration of the project development with the final product, in this case, being the PC and mobile phone prototypes.

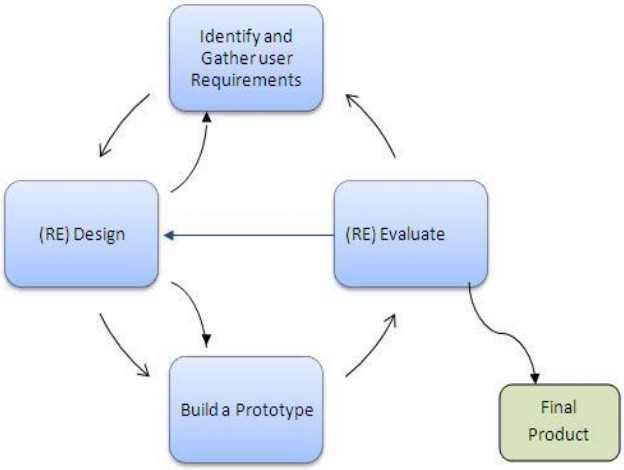

Fig. 1. The customary project life cycle provides an overview of the entire project, and consequently provides an outline for this section. These steps were iteratively applied to achieve the development of the prototypes.

A. *User requirements*

Many instant messaging services offered on mobile phones are text-based. According to a survey of 20 students at our university, users are still using short messaging service (SMS) even though they could send thousands of messages with a cellular IM service for the cost of a single SMS. This is because IM usage is charged by byte rather than by message. We found that the users interviewed want a convenient way of exchanging messages. According to most of them, SMS is not convenient in urgent situations due to the amount of time it takes to key in a message and they recognise that it is also expensive. Because cellular voice calls are even more expensive, these were key areas that the users identified as beneficial with a voice-based IM-like application such as PoC. The users' concerns include emergency situations, planning events like parties, as well as general social interaction amongst students on campus.

For the users interviewed, youths in particular want to engage in social interactive communications with friends, family and colleagues. They want voice instant messaging in place of textual IM and they would also prefer that the service be affordable. They also want the system to support real-time communication without any difficulties. About 70% of the people interviewed have a PoC application integrated on their mobile phones but they cannot use it because network service providers do not support it. We discovered that Vodacom offers a PTT service, but for only corporate customers. None of the students interviewed worked for a corporation. It is interesting that some of the corporate areas supported include construction, transport, security, distribution, manufacturing, and surface mining, as well as companies operating in catering, hospitality and courier industries (Pieter Uys, Vodacom Chief Operating Officer, December 2005.) Yet this does not apply to the vast majority of South African users.

Users interviewed told us they encounter difficulties to use text instant messaging in urgent situations. Users often make use of 'language compression' to enhance the speed of the keying in a message, such as "b4" and "2cu", as well reducing the amount of data to be sent and the consequent cost. This type of spelling and vocabulary can easily lose the contextual meaning of the message. This is mainly because people have different understandings of any given 'mobile' language, and it is usually based on phonetic contractions of spoken English. This can be acceptable by English-literate people. However, the English literacy of South African people is typically very low [www.eee.co.za]. For these reasons, people we interviewed deduced that text-based instant messaging is not 'multilingual friendly'. Thus, overall, the concept of a voice-based approach to IM seemed feasible to the users we interviewed.

B. *Requirements analysis and design*

Thorough analysis of the user requirements provided us with the design direction of the system. Fig. 2 illustrates a use-case view of the requirements analysis in terms of the functional activities that a user can perform. Fig. 3 shows a high level view of the design relationships between technology objects, in this case, a client and multiple servers.

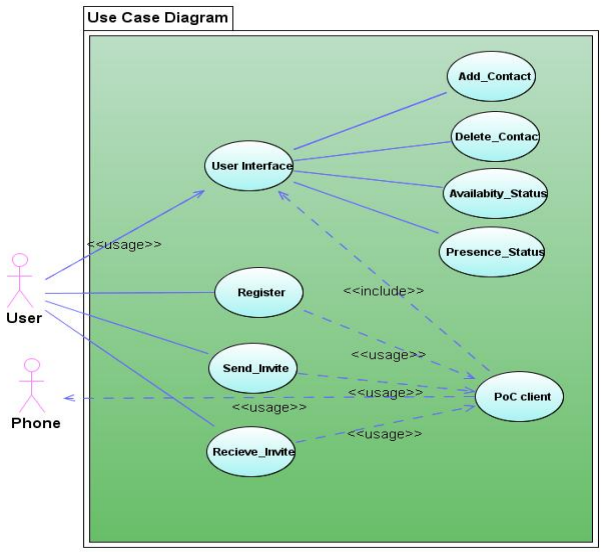

Fig. 2. A use-case diagram shows the users (on the left outside the box) as role players. They use the entities of the application via the phone's user interface to the underlying functionality.

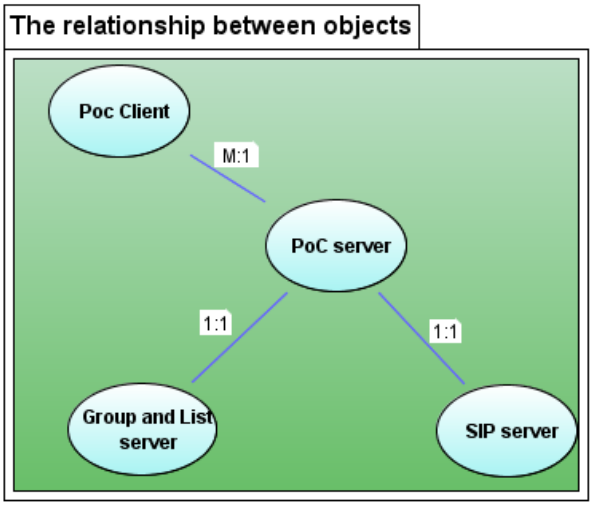

Fig. 3. The relationships between objects identify how one object maps to another. Many clients can use a single server. The M represents 'many' while a 1 represents 'one', i.e. M:1 qualifies the many-to-one relationship of multiple clients for one PoC server.

Floor control is also of vital importance because it controls the allocation of the floor during IP-PTT sessions. To obtain the floor, or the ability to speak, a client sends the PTT request to the PoC server and waits for feedback. The status of the floor is broadcast to all parties by the PoC server. Fig. 4 provides a high level description of the floor control class with UML (unified modelling language).

Fig. 4. Floor control methods illustrate the high level design of the floor control of the PoC application. The methods manage the PoC application's functional operation to give a user the ability to speak in half-duplex mode.

C. *Protocols and prototypes*

The prototypes runs on two platforms, PC and mobile, and although they are not explicitly compliant, both use common protocols defined in the IP Multimedia Subsystem (IMS). IMS is a functional architecture based on IP that provides multimedia services [7]. IMS is specified by the 3rd generation partnership project 3GPP [7] [8] and extended further by the European Telecommunication Standard Institute (ETSI) [7] to accommodate the convergence of multimedia services in the next generation network [7] [8]. Session Initiation Protocol (SIP) forms the core of the IP-PTT architecture. SIP is a signalling protocol used for multimedia session set-up and termination [9]. SIP is used in this context for the creation and termination of PTT calls. Fig. 5 shows the basic flow of voice messages.

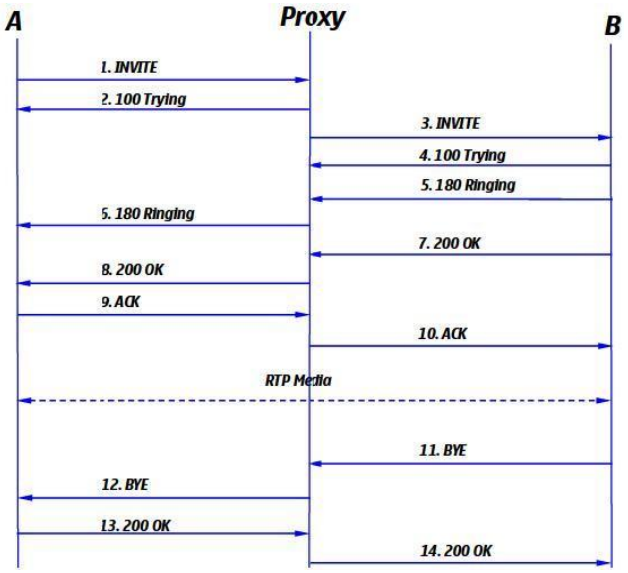

Fig. 5. Message flow in SIP, as taken from [13]. User A first sends an invite to user B that goes through the network proxy before the endpoints can communicate. The message flow is described in more detail in Table I.

TABLE I
SIP MESSAGE FLOW TAKEN FROM [13].

Step	Action	Description
1 A calls B	INVITE	A sends an INVITE request to the proxy.
2	100 Trying	Proxy sends a 100 response to A to acknowledge the request.
3	INVITE	Proxy forwards invite to B
4	100 Trying	B acknowledges request with a 100 response.
5	180 Ringing	B sends a 180 response to the proxy to indicate that B is being alerted.
6	180 Ringing	The proxy forwards B's180 response to A.
7 B answers	200 OK	B sends a 200 response to the proxy for connection established.
8	200 OK	The proxy forwards B's 200 response to A.
9	ACK	A acknowledges the 200 response from the proxy.
10	ACK	The proxy forwards the acknowledgement to B.
11 B terminates	BYE	B sends a BYE request to the proxy.
12	BYE	The proxy forwards B's BYE request to A.

The fourteen steps in Fig. 5 are described in more detail in this table. The flow here shows a typical SIP call set-up and termination between two user agents acting via a proxy. The last two steps from Fig. 5 are ignored.

The voice components within a given IP-PTT prototype is synchronously streamed over the IP network using Real-time Transport Protocol (RTP) [10]. RTP works together with RTP Control Protocol (RTCP) that provides RTP with statistics and control information. Presence on the application is provided by the SIP presence protocol extension, SIMPLE (Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions). The process of passing voice over the Internet is managed by these IP protocols, and others such as User Datagram Protocol (UDP), and is collectively referred to as VoIP.

The implementation of our IP-PTT prototype used two architectures. The first was a client-server approach used for desktop PC IP-PTT (see Fig. 6). This method places a server in between IP-PTT clients. The server relays real-time voice as packet data from one end to the other. Another purpose of the server is the registration and authentication of users. The server used in this case was an open source Asterisk server (www.asterisk.org). We used an open source Java-based SIP environment called SIP communicator (sip-communicator.org) as the basis for the desktop IP-PTT client. We used SIP communicator because of its simplicity and object-oriented style that enabled us to implement the IP-PTT in a straight-forward fashion using well-documented classes and methods. We used a wired local area network (LAN) as the transport medium for this approach.

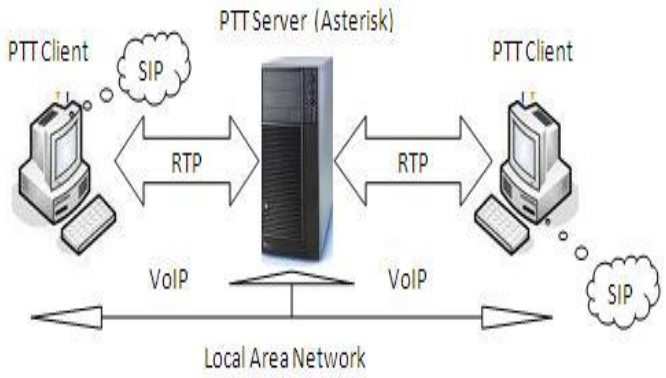

Fig. 6. Client-Server approach, The application on the clients uses SIP signalling with RTP as the voice carrier between the server and the clients. This is a typical VoIP application on a LAN.

The second approach was PoC deployed in P2P fashion. Another open source SIP stack, PJSIP (www.pjsip.org) was used on the mobile phone as the basis for the P2P approach. PJSIP is an open source SIP stack for the Symbian platform, and we chose to work with a Nokia phone. The network medium for this approach was WiFi. Fig. 7 shows the overview of this P2P approach. The network media in this case is WiFi to ensure free mobility, although it should be noted that any cellular data protocol may also be used. WiFi is essentially 'free' in our lab, whereas 2.5G and 3G data are not.

The most distinct factor of IP-PTT is the use of half-duplex real-time voice communication. We used the WireShark packet analyzer to observe the packet flow for the assurance of one-way communication.

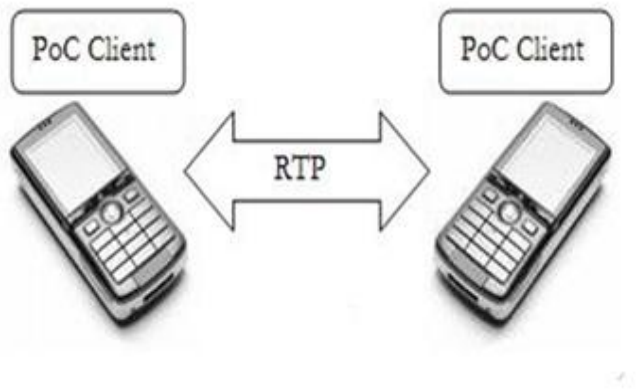

Fig. 7. Peer-to-peer architecture. This approch allows the clients to communicate directly with each other without the need for an interim relay server.

D. Evaluation

The evaluation of the prototypes was based on testing using various strategies commonly used by computer scientists as a part of the overall software development life cycle. Validation testing was carried out to examine whether the functionality of the software functioned in a manner reasonably expected by the end-user [11]. Stress testing was also conducted to test the system under immense pressure from a variety of traffic-related factors. We also conducted performance testing to examine the run-time performance of the software within the context of the overall integrated system [11]. Performance was tested with end-users and systematically measured with the Windows task manager. Six users 'hands on' tested the system, and structured questionnaires and interviews were again used to gather user feedback. The users were given three task scenarios to choose from: social interaction, emergency and corporate (secretariat) situations. The task scenarios were used to test the efficiency of the application from a user's perspective. We tested both prototypes as follows: on the local area network (LAN) using two PCs and on a WiFi network using Nokia E51 and E71 handsets.

The results were gathered from the various testing methods outlined above, thus examining the application from different angles in order to triangulate a firm picture of how the prototypes performed from both technical and user orientations. The validation test was successful as 100% of the users completed the task at hand, and about 95% of the users were satisfied with the results. Fig. 8 shows a graphical representation of the results for the following two questions:

Question 1: *Did you manage to complete your task*?
Question 2: *Are you satisfied with the efficiency of the application?*

Note that most of the questions from the questionnaires are not included in this paper.

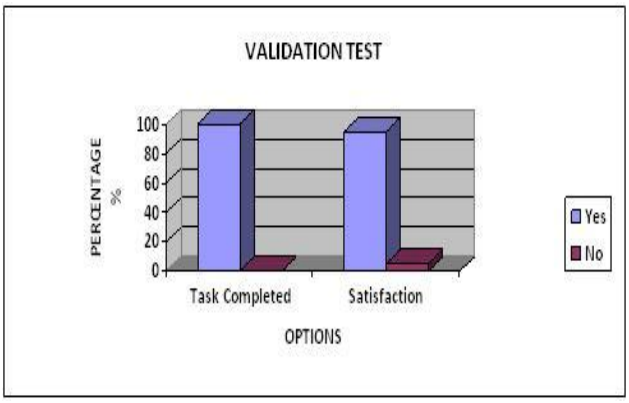

Fig. 8. Validation test results. The graph shows user satisfaction as well as the task completion rate. The x-coordinate shows the test factors and the y-coordinate shows the percentages of Yes/No answers.

The reason given for any unsatisfactory answers was mainly about the voice stream cuts, which was not the same for all users as the testing was conducted at different times under varying network traffic levels.

For stress testing, the desktop IP-PTT application was tested under high network traffic to deal with a relatively high number of users. With traffic on the network, real-time VoIP communication tends to suffer due to latency (delay of voice stream packets on the network). This was confirmed by the breakdown of the voice streams and the delay in time for voice packets to arrive. With the mobile phone scenario, many WiFi networks on our campus can bear negative effects on the PoC prototype due to interference. The interference can cause lot of unnecessary background noise. Nonetheless the peer-to-peer approach overcomes the latency problems encountered in the PC application because even in the presence of the noise, the voice still arrives on time and clear. Software should conform to performance otherwise it will be unacceptable even if it is validated. Fig. 9 shows how we used the Windows task manager to measure performance during the PTT session on a PC.

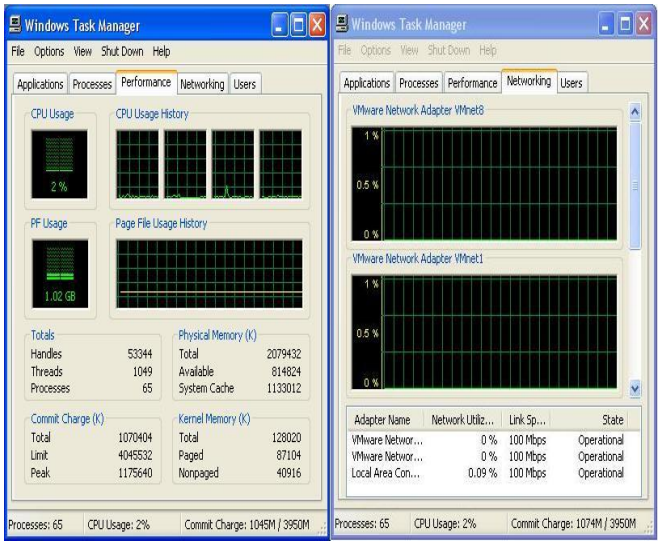

Fig. 9. Windows task manager. This screenshot illustrates the use of this tool during IP-PTT sessions to examine CPU usage, shown on the left. Bandwidth consumption is shown on the right hand side.

CPU usage during the session shown in Fig. 9 is 2%, which confirms minimal use of system resources consumption by the application prototype. Before the application was run, we stopped every application on the PC such that the CPU usage and bandwidth consumption was 0%. The left side of Fig. 9 shows the CPU usage of 2%. The bandwidth consumption is depicted on the right hand side and is 0.09% during the solitary session. This is due to the half-duplex nature of the application. The bandwidth usage on the remote side (receiver) is 0%, as is expected for half-duplex communication.

The user's perspective of the prototype's performance was gathered with questionnaires. Fifty percent of the users view the performance in terms of response in time as good and the other half said it is acceptable. Fig. 10 depicts the response of the users to this question:

Question: *How would you rate the performance of the system in terms of the response time?*

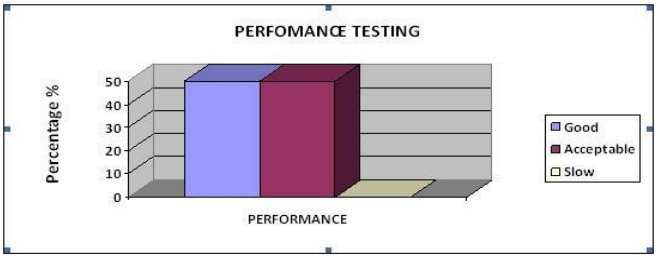

Fig. 10. User's view of performance. This graph shows the user's perspective of the prototype's performance, as collect with a questionnaire after hands-on use by the end-user.

III. Discussion

Analysis of the results allows us to deduce that both prototypes appear to meet the desired user needs in terms of basic PTT functionality. However, following on the incremental iterative nature of the software engineering life cycle, there is a need for improvement in some aspects of the prototypes. All participants completed their tasks at hand successfully although a few were unsatisfied. The main concern amongst the participants was the latency on the desktop IP-PTT prototype and the interference on the mobile PoC prototype. Overall, we found that users prefer PoC since the interference tends to fade away within a period of seconds. The mobility and portability of the dynamic PoC also appears to play a role as far as preference is concerned. It must be noted that the interference encountered during the tests is actually very realistic since more and more WiFi networks are being established.

From careful evaluation of the prototypes, and a literature survey, we deduce that the client-server IP-PTT that was employed on the PC desktop had a negative effect towards the application under extreme load. The ideal implementation of the prototype might be the use of the two approaches combined together. We kept them separate to evaluate the feasibility of half-duplex communication in both instances. In essence, a client-server server approach is usually employed in large communication networks, with large traffic, in order to keep strict control and monitoring

over the usage. This approach can be more secure as a result of robust authentication on the servers. Peer-to-peer, on the other hand, is very inexpensive due to its instability [technet.microsoft.com/en-us/library/cc751396.aspx]; it is thus suitable for small networks. It was for this reason that peer-to-peer was employed on the WiFi environment as a WiFi network can be a small controllable network.

Interviews with users indicate that they are more concerned about the cost of IP-PTT. Costs are based solely on bandwidth consumption. The half-duplex technique enables one-way bandwidth usage as compared to full-duplex VoIP applications. Although the majority of the users were satisfied with the efficiency of the application, about half of the participants only rated the performance 'in respect of time' as "good". The other half felt it was "acceptable".

IV. Conclusions and Future work

This paper discussed the implementation of IP-based Push-to-talk prototypes on a PC and mobile phone using Session Initiation Protocol. IP-PTT is a voice instant messaging service that operates like a walkie-talkie and/or CB radio system. PTT on a cellular phone is called PoC. IP-PTT employs half-duplex communication that proved to be very efficient as far as bandwidth consumption is concerned.

This paper roughly compared two architectural approaches, client-server and peer-to-peer, on two different platforms and network media: desktop PC on a LAN and mobile phone on a WiFi network, respectively. We conclude that the client-server approach was good for the PC on a LAN since such a network could be very big. A wired network is very stable and much more secure as compared to a wireless network. On the other hand, it is very difficult to apply the client-server approach to dynamic environments like PoC. Considering the amount of time it would take to handle the instability and interference of WiFi networks, we decided to rather use a peer-to-peer approach for PoC using WiFi. Both prototypes were tested with three task scenarios: social, emergency and corporate. Based on results from user questionnaires and interviews, we can conclude that IP-PTT and PoC offer acceptable voice messaging possibilities.

The test results showed that the user oriented view regarding the performance leaves a lot to be improved for the next prototypes. Future work includes adding a video capability, thus developing a Push-to-Video in accordance with the NGN's IMS multimedia convergence. Adding a video will enhance the communication at large as well as accommodating other social groups like the Deaf people. It would also be ideal to consider an asynchronous communication mode. This would allow users to still be able to sent audio or video IM where real time is not possible.

Acknowledgements

The authors thank the National Research Foundation (NRF) for the scholarship offered to the first author. We also thank Telkom, Cisco and THRIP (Technology and Human Resources for Industry Programme) for financial support via the Telkom Centre of Excellence (CoE) programme. THRIP funding is managed by the National Research Foundation (NRF). Any opinion, findings and conclusions or recommendations expressed in this material are those of the author(s) and therefore the NRF does not accept any liability in regard thereto.

References

[1] C. Schmandt, J. Kim, K. Lee, G. Vallejo, and M. Ackerman, "Mediated voice communication via mobile IP," *Proceedings of the 15th annual ACM symposium on User interface software and technology*, 2002, pp. 141–150.

[2] E. O'Regan and D. Pesch, "Performance Estimation of a SIP based Push-to-Talk Service for 3G Networks," *Cork Institute of Technology, Ireland, Adaptive Wireless Systems Group*, 2004.

[3] L.Y. Wu, M.H. Tsai, Y.B. Lin, and J.S. Yang, "A client-side design and implementation for push to talk over cellular service," *Wireless Communications and Mobile Computing*, vol. 7, 2007, pp. 539–552.

[4] R. Thomas, "Parents Guide to MXit." South Africa, 2006.

[5] R. Koivisto, "Towards the Next Wave of Mobile Communications: Push-to-Talk over a Cellular: Still Searching the Flow of Success" *In Proceedings of the Research Seminar on Telecommunications Business*, TML-C19, pp 45-96, 2005.

[6] Open Mobile Alliance, "Push to Talk over Cellular Architecture," *OMA-AD-PoC_V1_0-20050428-C Candidate Version 2.0*, February 2008.

[7] G. Bertrand, "The IP Multimedia Subsystem in Next Generation Networks," *Rapport technique, ENST Bretagne*, 2007.

[8] C. Menkens and N Kjellin, "IMS Social Network Application with J2ME compatible Push-To-Talk Service," *Next Generation Mobile Applications, Services and Technologies, 2007. NGMAST'07. The 2007 International Conference on*, 2007, pp. 70–75.

[9] J. Rosenberg, et al., "SIP: Session Initiation Protocol" *IETF RFC 3261*, June 2007.

[10] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications" *IETF RFC 3550*, July 2007.

[11] R.S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed. McGraw-Hill, 2004, pp 406-408.

[12] V. Ronnholm, "Push-to-Talk over Bluetooth," *proceedings of the 39th Annual Hawaii International Conference on System Sciences*, vol. 9, 2006, pp232c.

[13] Forum Nokia, "SIP/VOIP Call Flaw Messages", October 2008.

Hlabishi I. Kobo obtained a BSc Honours degree in Computer Science from the University of the Western Cape in 2009. The first author is presently studying towards an MSc degree with the Bridging Applications and Networks Group (BANG) at the same institution. His main research interest is now wireless mesh routing protocols on mobile phones.

William D. Tucker is a Senior Lecturer in Computer Science at UWC and leads BANG research. He obtained a PhD from University of Cape Town in 2009. His main research interest is applying Internet Protocol to the ICT4D context.

Michael J. Norman is a Senior Lecturer in Computer Science at UWC. His interests are in Software Engineering.