Reliable Communication Across Ad Hoc Networks

Francois N. Daniels and William D. Tucker
Department of Computer Science, University of the Western Cape, Private Bag X17 Bellville, 7535, South Africa
Telephone: +(27) 21 959-2461, Fax: +(27) 21 959-3006, Email: {**fndaniels**, btucker}@uwc.ac.za

***Abstract*—This paper presents a fully decentralised peer-to-peer voice communication tool intended for use across mobile ad hoc networks (MANET) by distributed groups who desired collaboration. We examined the synergy between MANETs and peer-to-peer virtual overlay networks which allowed the creation of ad hoc applications. One style of communication considered suitable for task oriented distributed group collaboration was push-to-talk. This research was focused on providing a push-to-talk communication platform suitable for deployment across MANETs. The research methodology employed was a proof of concept approach within a classical experimental computer science paradigm. We developed a prototype which used JXTA, a peer-to-peer virtual overlay network, to provide push-to-talk functionality across MANETs. Guaranteed delivery of messages was provided via a peer-to-peer voicemail delivery system. While the system did what intended we show that JXTA had a problem with the *efficient* delivery of voice samples.**

***SATNAC Classification*—The Intelligent IP Edge: Packetised voice (VoATM, VoIP)**

***Index Terms*—ad hoc applications, mobile IP, peer-to-peer, voice over IP, multimedia, instant messaging**

I. Introduction

HAVING the means to communicate anytime, anywhere is valued by many. This is especially so for distributed groups who need a way to collaborate. Two way radio communication has been around for a long time. These public shared communication channels allow multiple parties to participate in distributed conversations. These systems function on a Push-to-Talk premise. Audio is only transmitted while the "talk" button is depressed, and audio transmissions are only received while the button is not depressed. This is a half duplex form of communication – one can either transmit audio or one can receive audio, but not both simultaneously. With the advent of digital trunked-radio networks came the ability of cellular providers to provide similar services to their subscribers [24]. This also introduced a new form of communication, one-to-one private push to talk.

A problem facing mobile half-duplex transceivers, such as *walkie talkies*, is the fact their range is limited due to the laws which govern their power output levels. Mobile Ad Hoc Networks (MANET) solved this limitation by allowing nodes to forward signals on behalf of each other. This enables the network reach to grow as the number of mobile nodes attached to the network increases. We developed a Push-to-Talk system intended for deployment across MANETs in order to provide the type of "anytime, anywhere" communication afforded by technologies such as *walkie-talkies*. It offered one-to-one private push-to-talk, public channel push-to-talk, as well as private multiparty push-to-talk sessions. Additionally it provided text mode conversations transport for low network capacity scenarios as well as a voicemail feature for use when a path to the recipient could not be established.

Section II examines the relationship between mobile ad hoc networks and peer-to-peer architectures. Section III we provide some insight into related work. The research questions and experimental design is then covered in Section IV. Section V justifies the research methodology. Section VI examines the design and implementation of the prototype application. In Section VII we discuss our findings and Section presents our conclusions.VIII .Section IX mentions future work.

II. Background

Traditional wireless network installations have fixed network topologies. Ad hoc wireless networks, on the other hand, are dynamically reconfigurable and do not depend on a static infrastructure or a central point of control such as an access point. A MANET is an ad hoc network in which the nodes are free to roam about arbitrarily. In these ad hoc networks, each node acts both as a peer as well as a router. Wireless routers have limited ranges, so allowing nodes to forward packets on behalf of each other increases the reach of the network and creates multi-hop paths to a destination if required, as shown in Figure 1.

MANETs can easily and quickly be deployed since they are self-organising and do not rely on any fixed infrastructure. The hardware required for wireless ad hoc networks are cost priced due to a competitive global market this makes setting up MANETs not only fast, but cheap as well. These networks are therefore desirable in situations where there is no time to set up a fixed network such in disaster response scenarios, or when it would be too costly to set up a fixed network.

There is an inherent congruency between peer-to-peer applications and ad hoc networks. In peer-to-peer applications the arrangement of peers are ad hoc, and in ad hoc networks all nodes are peers. Self organisation and the idea of decentralisation are key attributes of both ad hoc networks as well peer-to-peer virtual overlay networks. The goal of ad hoc computing, which can be realised by the union of ad hoc networks and peer-to-peer overlay networks, is to make distinct types of information available to peers "anytime and anywhere".

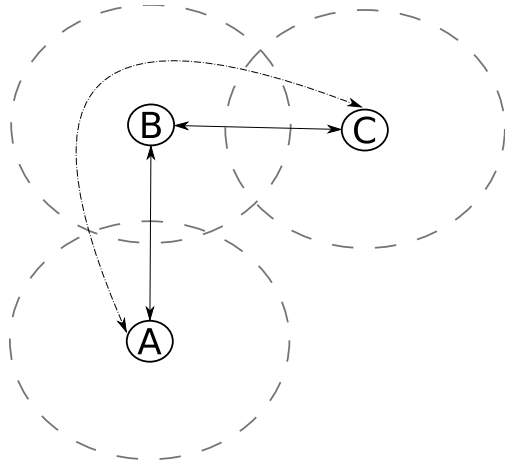

Fig. 1. This figure illustrates multi-hop paths in MANETs. In conventional wireless networks node A and B, and node B and C would be able to communicate, but node A and C would not because they were not within radio distance of each other. In MANETs, node B would forward signals on behalf of A if C was the recipient, and vice versa. This cooperation of nodes in this manner increased the size of the network.

III. Related Work

A. Service Delivery

Konark [8] was a service discovery and delivery protocol designed for ad hoc networks. It used the underlying network infrastructure for peer naming and routing. With the help of *Konark* researchers designed a system to deploy generic peer-to-peer systems across ad hoc networks. Garbinato and Rupp [10] developed a mobile peer-to-peer communication framework. The key feature of their system was that it was not dependant on the underlying network technology. This work was similar to *Proem* [14], a computing platform built by the University of Oregon for use in mobile ad hoc networks. The main purpose of *Proem* was to provide a framework for the development of applications for mobile ad-hoc network environments. The most mature publicly available virtual overlay network infrastructure was Sun Microsystem's JXTA (Juxtapose) [11]. Originally pioneered by Sun, it is now an open-source project that is still supported by them. JXTA defines a set of protocols and APIs for general-purpose peer-to-peer communication in a fully decentralised environment.

Researchers at the Communications Law Centre [20] developed a prototype application called *vuCRN*, based on JXTA technology, which enabled effective control over who could upload content in a peer-to-peer network by using existing user authentication. A framework for synchronised distributed group collaboration and knowledge sharing was developed by researchers from the National University of Singapore [22]. It was aimed at distributed collaborating environments such as e-Sharing and e-Learning, it contributed technologies such as shared white boards and threaded discussion groups. Siri Birgitte Uldal from the University of University of Tromsø developed a prototype [23] application capable of sharing resources across ad hoc networks. This prototype was only capable of file transfers. The main contribution was that of a shared virtual folder from which peers shared resources. A peer-to-peer forum was developed at the University of Saskatchewan [12]. This was an example of software with a client-server architecture redesigned as a peer-to-peer solution.

B. Distributed Audio

Impromptu [19] was a distributed mobile audio framework. It allowed for the deployment of network-centric applications and services. Various applications such as a baby monitor, a chat application, and an mp3 player was implemented using this framework. *TattleTrail* [13] was a multiparty VoIP application created using the *Impromptu* framework. It allowed for three modes of operation. The first mode, called "catch up" mode, sent a user a review of the current conversation when they joined the chat room. This was accomplished by sending them a high-speed version of the recorded audio. The second mode called "push to talk" mode was activated once the user was caught up. Users pushed a button to talk and released it to receive audio. The final mode of operation was called "background" mode, which was activated once a user left a chat. Once in this mode a user only received *alerts* of new messages. The above projects were all client-server based.

C. Push-to-Talk

Woodruff and Aoki performed a qualitative analysis on a group of young adults using push-to-talk handsets over the period of one week [24]. It was discovered that the half-duplex nature of push-to-talk afforded participants with several conversational styles ranging from sustained turn taking, to bursts of conversational activity, to occasional responses followed by periods of silence. They went on to note that while half-duplex was often seen as being low quality and offering limited functionality, the "reduced interactional commitment" was seen as desirable as it gave users alternative communication mechanisms. Users were released from the need to respond immediately because audio was only transmitted while a button was explicitly pressed (pushed) [2], which gave recipients a feeling of "plausible deniability". [24] concluded by saying that push-to-talk offered an unique balance between availability and obligation to participate in conversations.

Research has been done on implementing push-to-talk across various networks. The Open Mobile Alliance (OMA) [16] was in the process of standardising push-to-talk as an IP Multimedia Subsystem (IMS) application. This ensured that push-to-talk implementations was interoperable between complying cellular providers [6]. By using candidate releases of the OMA Push to Talk across Cellular (PoC) specifications, Burman [7] created a prototype of PoC for a Personal Digital Cellular (PDC) network. The performance of push-to-talk across General Packet Radio Service (GPRS) [3] and High Speed Downlink Packet Access (HSDPA) [1] was investigated in two separate studies. While Akerfeldt found that HSDPA had enough capacity for 65 concurrent users, Balazs showed that GPRS supported significantly less. The OMA PoC standard specified a client-server architecture, which

the aforementioned studies all followed. One project which attempted to deviate from this specification was [18]. They designed a framework for deploying PoC across bluetooth scatternets – which were peer-to-peer networks. Some PoC functionality was truncated in the process.

IV. Research Questions

Group communication allows for rapid information exchange, helping to increase the productivity of distributed members performing a group related task [21]. Group collaboration across MANETs is one way of providing this functionality. This need led us to the following research questions:

- How can one provide group communication in a volatile network topology where access to a centralised control point can not be guaranteed?
- How does one deal with peer discovery and network fragmentation?
- Which communication paradigm best fits such network conditions?

One style of communication was considered the most suitable for task oriented group collaboration, Push-to-Talk [24], [19]. We were interested in providing push-to-talk functionality across MANETs but the dynamic nature of the network topology meant that a client-server based solution would be not appropriate, so a peer-to-peer solution was necessary. In order for two or more peers to communicate they needed a way to discover each other's location, and be informed of any change of their online presence. Network fragmentation meant that a path to the destination could not be guaranteed and conversations could be prematurely disconnected. We also needed a way to provide communication even when the network experienced capacity issues.

V. Methodology and Experimental Design

We employed a proof of concept approach within a classical experimental computer science paradigm in order to explore those research questions. According to [4], experimentation was of central importance to any science or engineering field of study. This methodology associated experimentation both with laboratory as well as field experimentation [17]. It was argued in [9] that experimentation has a place in the real world, just as it does in the natural sciences. They go on to claim that scientific method should be employed so that the experiment was repeatable and the results were reproducible. We used an iterative approach of prototyping with experimental phases of qualitative and quantitative data collection.

Prototypes were constructed using the requirements specifications obtained from the research questions covered in section IV. The prototypes were designed for desktop computers running a Java virtual machine. At each iteration of development we evaluated the capabilities not only of the prototype, but on the paradigms and frameworks they were based upon. The prototype needed an efficient presence implementation capable of functioning in a peer-to-peer environment, it required a way of handling network fragmentation and a way of storing audio and forwarding it when a path to the recipient was available, and lastly it needed a means of communicating under varying network load scenarios.

Initial prototypes were based on hybrid peer-to-peer design principals. This, however, did not resolve the problem of continuity of communication after network fragmentation. Later, a switch to the JXTA virtual overlay network put us in a position to start answering the research questions. Small-scale experimentation took place in the computer laboratory. A private wireless network was created for several machines running the prototype application. This enabled us to evaluate functionality such as peer discovery and the one-to-one and group communication modes. Network fragmentation was tested by running another instance of the prototype on a virtual machine. When the WiFi adapter of the host machine was turned off, the ability of the host to communicate with the virtual client was evaluated. Larger scale experimentation is planned using the NS-2 network simulator in conjunction with *AgentJ*, a Java interface to NS-2.

VI. Design and Implementation

The latest version of the prototype used the JXTA overlay network to form an ad hoc peer-to-peer network across which computer mediated communication could take place. To reduce development time we decided to extend the open-source myJXTA project by means of writing several plugins for it. myJXTA was a peer-to-peer collaboration application based on JXTA technology. We extended upon the work done by Schmandt et al. [19] by allowing multiple simultaneous groups to collaborate on discrete "open channels" across an ad hoc IP network. The following sections will describe the overall system architecture, as well as the implementation details of the coded plugins.

A. Simple Presence Implementation

Using JXTA propagation pipes we implemented a simple presence notifier. Presence is normally handled by a central server (or server cluster), but having static servers on ad hoc networks is impractical. We realised that there were three solutions to this problem. Firstly we could assign a dynamic server selected from one of the current online peers. The complication with this solution was that new server(s) would have to be selected as the ad hoc network fragments and merges. The second solution involved the peers registering their presence with every online peer. This meant that each peer would be a server for every other peer. While this solution was simpler, it also meant that a lot more precious bandwidth would be wasted on presence updates and other control messages. We decided to use a fully distributed approach in which propagate pipes were used to send periodic presence updates to multiple peers simultaneously. The program flow for the presence plugin is illustrated in Figure 2.

Propagate pipes allowed for many-to-many message transport. They were similar in function to the Internet Group Management Protocol (IGMP) in that only peers listening on a specific pipe (or IP address in the case of IGMP) received

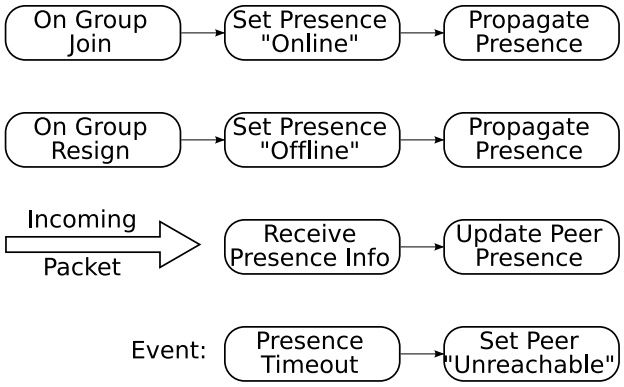

Fig. 2. This is a representation of the four main events which occurred in our implementation of presence. When a peer joined a group, their own presence was set to "online", this presence was then sent to every other peer on the network. Unless a peer *received* periodic presence updates, it would flag the expected sender's presence to "unreachable"

the message. It was, however, superior to simple multicasting in that it used multicasting where possible, and one-to-one message forwarding via rendezvous peers otherwise.

When a peer joined a group it sent the entire group an "online" presence update. It repeated this status update periodically. This model still allowed the user to change their status to "away" or "busy". An "offline" status update was propagated to all peers in the group when a peer resigned from that group. This behaviour also illustrates the fact that the presence status was group dependent and not global. If a peer did not receive a presence update from another peer for longer than a minute, that peer's presence was set to "unreachable" – the presence was reset the moment a presence update was received. This allowed the system the keep tabs on who it was still available to communicate with. Implementing it in this fashion also meant that no ACK messages needed to be transmitted by peers who received presence updates.

B. Open Channel Push-to-Talk

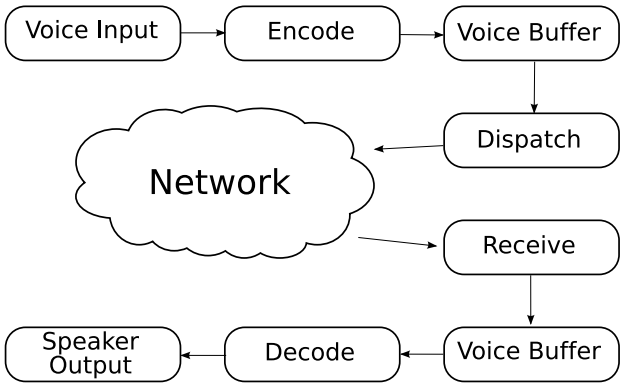

Fig. 3. A simplified model of the voice data flow in the prototype is illustrated in this figure. Encoded voice samples were stored in a voice buffer and periodically transmitted to the recipient. The recipient stored the received samples in their buffer until enough samples were collected for playback. This value was depended on voice quality and jitter compensation.

Figure 3 shows a simplified version of the data flow of voice through the application. Once some voice chunks had been read from the microphone it was encoded with the speex codec, which reduced the number of bits used to represent the stream. The encoded data was then stored in a data buffer. Once enough data had been stored in order to fill a packet it will be dispatched across the network. The receiver then stored the data into its data buffer. Once enough voice data (the size depended on the encode quality – higher quality meant more bytes) had been buffered it was decoded and played back.

Voice data was transported via propagation pipes. This meant that every peer listening on this propagation pipe received the voice data. The operational mode of this module was push-to-talk. A peer only transmitted audio while the user held down the push-to-talk button. While the push-to-talk button was depressed no audio would be processed or stored by the peer. This meant that a peer could only transmit or receive voice data at any instant, but not both simultaneously.

C. Multimodal One-to-One

In addition to open channel communication we were also interested in exploring one-to-one communication across volatile networks. In ad hoc networks there were no guarantees of either connectivity or capacity which led us to examine which counter measures could be taken in order to provide a communication platform which was stable under such unpredictable conditions. The result of our research was the multimodal one-to-one module.

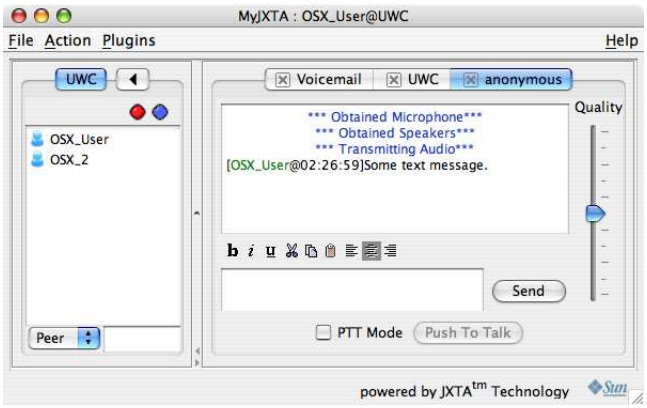

Fig. 4. A view of the multimodal one-to-one user interface. This interface allowed for simultaneous voice and text transmission. The voice *transmission* could be switched from hands-free to push-to-talk mode. This allowed a user to both conserve bandwidth as well as gain some privacy. Additionally the interface allowed a user to lower the quality of *incoming* audio, thus allowing a user to trade quality for latency.

Voice was the primary form of communication used in this module as it allowed for a richer communication environment than text alone could provide. The transportation of the voice data is illustrated by Figure 3. During times of network congestion voice packets may be dropped, which resulted in poor Quality of Service (QoS). By lowering the quality slider a user could reduce the quality of the audio session in order to reduce the size of the voice chunks (see Figure 4).

This only reduced the size of the *incoming* packets, which meant that the other peer would still receive audio at their preferred quality setting. The quality of the conversation was thus asymmetrical, much like the network conditions on ad hoc networks. Furthermore a user could switch into push-to-talk mode. This too was asymmetrical, allowing one party to be in live VoIP mode, while the other was in push-to-talk mode. Push-to-talk mode was useful both as a mechanism for conserving bandwidth usage, as well as for activating "privacy" mode.

In addition to voice, it was also possible to communicate via text. Text was included due to its lightweight nature and was there to complement the voice. The text feature was used during periods of network congestion as well as to facilitate call co-ordination during dual push-to-talk mode (while both parties had push to talk activated). If the conversation was prematurely terminated the voicemail feature would automatically be activated.

D. Voicemail

Asynchronous reliable VoIP was made possible by the voicemail plugin. When network conditions were turbulent, and synchronous communication was not possible one had to resort to fully asynchronous communication.

There were two ways in which the voicemail module was used within the prototype. Firstly when two peers were disconnected mid-conversation, the voicemail plugin would be activated. This will allowed parties to get their "last say" across. Upon activation the voicemail plugin recorded a voice message up to 20 seconds in length. The second instance in which the plugin was activated was when a user wanted to leave another party a message without engaging in a transactional conversation. A peer could do this whether or not the other party was online. The message was dispatched once the recipient of the messages came back online, in other words once there was once again a path between the sender and receiver. The delivery of these voicemail messages were guaranteed as the file transfer was capable of resuming from where it last got cut off if the process was interrupted due to network failure or fragmentation.

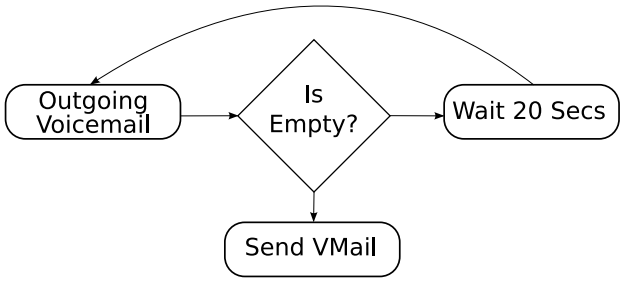

Fig. 5. Voicemail allowed users to leave messages for both offline as well as online peers. If the recipient is offline, the mail would be delivery as soon as a path to the destination is available. Online users received their mail immediately. The outgoing mail folder was polled every twenty seconds, at which point the system would try and deliver any undelivered messages.

The logic behind the voicemail module is illustrated by Figure 5. The system checked the voicemail directory for any outgoing voicemail messages. If undelivered messages were found and the recipients were online, qualifying messages would be dispatched.

VII. Discussion

Presence was especially important within mobile ad hoc networks, as it allowed peers to know which other peers were reachable. Presence updates were wrapped within a single packet that was periodically propagated across the network. Receipt of these messages was not acknowledged. It was, however, the receipt of these messages that allowed the system to determine which peers were still able to communicate with each other, either directly or via multi-hop paths.

Providing push-to-talk functionality within a peer-to-peer framework meant that greedy floor control had to be implemented. That is, the users of the system handled the floor control. Private and public shared push-to-talk channels allowed groups of peers to communicate. The private shared channels could be password protected to limit access, while the open channel push-to-talk allowed any peer to join the channel (group). This too was implemented using propagation pipes, which allowed for the relatively efficient delivery of voice to multiple distributed peers.

Private communication between two peers was provided via a one-to-one multimodal communication channel. The interface allowed for a mixed text-voice communication environment. For low network capacity conditions adjustable quality sliders were provided to reduce or increase the quality of the *incoming* voice. This allowed users to adjust the amount of traffic passing through their node and thereby adjust the amount of latency due to anti-jitter buffering while still transmitting audio at a quality desired by the recipient. This was especially useful for asymmetrical MANET topologies. Additionally the interface provided a method to switch from hands-free voice over IP to push-to-talk mode. These different communication mechanisms provided different ways of communicating depending of the level of interactional commitment desired by users.

For the guaranteed delivery of messages, the prototype provided a voicemail delivery system. Voicemail could be recorded and sent to peers even if they were offline. This was accomplished by caching voicemail messages locally. As soon as a path was available to the recipient, the message would be transferred. The system also provided automatic voicemail recording. When two users participating in a one-to-one communication were disconnected due to a network error, the automatic voicemail module would be activated to allow users to get their "last word" across. The recording could be cancelled at the user's discretion.

VIII. Conclusion

The main outcome of this research was a push-to-talk communication application that supported public open channel, private shared channel as well as one-to-one commu-

nication. JXTA was designed for wired infrastructure mode network topologies [15], and thus was not ideal for use within MANETs. A problem of efficient *delivery* existed within the JXTA framework. Each JXTA message was wrapped in XML messages that assisted the framework to accomplish certain tasks such as NAT and firewall traversal. As pointed out in [5], when delivering small payloads such as compressed voice samples using JXTA, the bulk of a message was in fact composed of JXTA XML headers. They demonstrated that these headers increased the size of packets by more than 10 fold.

IX. Future Work

We propose a hybrid delivery scheme for JXTA in order to adapt it for use across MANETs. Whenever possible deliver packets in a peer-to-peer manner using raw UDP/TCP protocols, bypassing extraneous XML headers. In the case of MANETs, allow MAC level protocols to do any needed routing when possible, otherwise fall back to JXTA transport mechanisms. This is still being implemented as part of the next prototyping cycle.

We are currently working on making JXTA more suitable for MANETs, while still retaining interoperability with vanilla JXTA distributions. In the future we hope to port a solution to mobile devices such as cellular phones and PDAs.

Acknowledgments

The authors thank Telkom, Cisco and THRIP for financial support via the Telkom Centre of Excellence (CoE) programme.

References

[1] E. Akerfeldt, "Push-to-talk over cellular over high speed downlink packet access - a performance evaluation," Master's thesis, KTH Signals Sensors and Systems, January 2005.

[2] P. M. Aoki and A. Woodruff, "Making space for stories: Ambiguity in the design of personal communication system," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. New York, NY, USA: ACM Press, April 2005, pp. 181–190.

[3] A. Balazs, "Push-to-talk performance over GPRS," in *Proceedings of the 7th ACM international Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '04)*. New York, NY, USA: ACM Press, October 2004, pp. 182–187.

[4] V. R. Basili and M. V. Zelkowitz, "Empirical studies to build a science of computer science," *Communications of the ACM*, vol. 50, no. 11, pp. 33–37, 2007.

[5] L. Bernardo, R. Oliveira, S. Gaspar, D. Paulino, and P. Pinto, "A telephony application for manets," Online. Available at http://tele1.dee.fct.unl.pt/papers/winsys2006.pdf, 2006.

[6] N. Blum and T. Magedanz, "PTT + IMS = PTM - towards community/presence-based ims multimedia services." Los Alamitos, CA, USA: IEEE Computer Society, 2005, pp. 337–344. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ISM.2005.93

[7] O. Burman, "Push-to-talk in PDC packet data network," Master's thesis, Umeå University, October 2004.

[8] N. Desai, V. Verma, and S. Helal, "Infrastructure for Peer-to-Peer Applications in Ad-Hoc Networks," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, USA, February 2003.

[9] D. G. Feitelson, "Experimental computer science," *Communications of the ACM*, vol. 50, no. 11, pp. 24–26, 2007.

[10] B. Garbinato and P. Rupp, "From ad hoc networks to ad hoc applications," in *Proceedings of the 7th International Conference on Telecommunications (ConTEL 2003)*, vol. 1, 2003, pp. 145–149.

[11] L. Gong, "Jxta: A network programming environment," *IEEE Internet Computing*, vol. 5, no. 3, pp. 88–95, May 2001.

[12] E. Halepovic and R. Deters, "Building a p2p forum system with jxta," *Proceedings of the Second International Conference on Peer-to-Peer Computing*, 2002.

[13] J. S. Kim, "Tattletrail: An archiving voice chat systems for mobile users over internet protocol," Master's thesis, Massachusetts Institute of Technology, June 2002.

[14] G. Kortuem, J. Schneider, D. Preuitt, T. G. C. Thompson, S. Fickas, and Z. Segall, "When peer-to-peer comes face-to-face: Collaborative peer-to-peer computing in mobile ad hoc networks," in *Proceedings of the First International Conference on Peer-to-Peer Computing*. Los Alamitos, CA, USA: IEEE Computer Society, 2001, p. 75.

[15] R. Oliveira, L. Bernardo, N. Ruivo, and P. Pinto, "Searching for pi resources on manets using jxta," in *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT-SAPIR-ELETE'05)*. IEEE Press, July 2005, pp. 371–376.

[16] OMA, *Enabler Release Definition for Push-to-Talk over Cellular*, approved version 1.0.1 ed., Open Mobile Alliance (OMA), November 2006.

[17] L. Peterson and V. S. Pai, "Experience-driven experimental systems research," *Communications of the ACM*, vol. 50, no. 11, pp. 38–44, 2007.

[18] V. Rönnholm, "Push-to-talk over bluetooth," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06) Track 9*, vol. 9. Los Alamitos, CA, USA: IEEE Computer Society, 2006, p. 232c.

[19] C. Schmandt, K. H. Lee, J. Kim, and M. Ackerman, "Impromptu: Managing networked audio applications for mobile users," in *Proceedings of the 2nd international Conference on Mobile Systems, Applications, and Services (MobiSys '04)*. New York, NY, USA: ACM Press, June 2004, pp. 59–69.

[20] H. Shi, Y. Zhang, J. Zhang, E. Beal, and N. Moustakas, "Collaborative Peer-to-Peer Service for Information Sharing Using JXTA," *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences-Volume 1 (IMSCCS'06)-Volume 01*, pp. 552–559, 2006.

[21] R. Shtykh, G. Zhang, and Q. Jin, "Peer-to-peer solution to support group collaboration and information sharing," *International Journal of Pervasive Computing and Communications*, vol. 1, no. 3, p. 187, September 2005.

[22] J.-Y. Tham, S.-L. Lee, C.-E. Tan, Roger, and L.-C. Tee, "A distributed peer-to-peer platform for synchronized group collaboration and knowledge sharing," in *Proceedings of the 2004 International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES)*, Wuhan, Hubei, China, September 2004.

[23] S. B. Uldal, "Casual resource sharing with shared virtual folders," Master's thesis, University of Tromsø, June 2007.

[24] A. Woodruff and P. M. Aoki, "How push-to-talk makes talk less pushy," in *Proceedings of the 2003 international ACM SIGGROUP Conference on Supporting Group Work (GROUP '03)*. New York, NY, USA: ACM Press, November 2003, pp. 170–179.

Francois Daniels is a Telkom Centre of Excellence M.Sc student at the University of the Western Cape. Currently he is doing research on peer-to-peer applications over ad hoc networks.

William D. Tucker is the director of the Broadband Applications and Networks Group (BANG) research group at the University of the Western Cape.